# Team Design Project and Skills Report

## Team No.31: FFAASSTT

| Student's Name | Student's UESTC ID# | Student's UoG ID# |
|---|---|---|
| Liu Jinyuan | 2016200302012 | 2289243L |
| Ding Shuhan | 2016200302005 | 2289236D |
| Peng Zhendong | 2016200302008 | 2289239P |
| Xu Yang | 2016200301035 | 2289231X |
| Li Yueshan | 2016200302011 | 2289242L |
| Xu Mengying | 2016200302006 | 2289237X |
| Yu Xi | 2016200302009 | 2289240Y |
| Gao Haihuan | 2016200302007 | 2289238G |

## Abstract

Team design has been considered as the most efficiently working module and skills for project management are requisite in most subject areas. During the Team design and Project Skill (TDPS) session, we design and build a car which can perform different tasks including line tracking, obstacle avoidance, color recognition, releasing item and communication. To create a motivated working atmosphere, our team design our own logo, team name and uniform. We divide our team into power group and visual group to work more efficiently. Specific tasks are allocated based on the superiority of each group member. Gantt Chart and lab notebooks are utilized to ensure the working progress and record the experiment process respectively. Components based on the Arduino UNO and Raspberry Pi are applied to the construction of the car, and we also design our algorithm containing line tracking, color recognition and priority algorithm based on both Arduino and Python programming platform during our project. Moreover, the materials applied for our car are light and environmental-friendly. For the patio 1 and patio 2, our car could finish all the tasks successfully within ten minutes and eleven minutes respectively. In the future, the car can be improved in three aspects: firstly, PID algorithm could be applied to guide the car driving along the straight line. Secondly, the threshold setting for line tracking algorithm only stays on multiple attempts and needs to be adjusted for different environments. Thirdly, the number of GPIO ports used for the communication between the Arduino and Raspberry Pi can be increased to enable high accuracy control.

# Contents

# 1 Introduction

## 1.1 Team Introduction

In this section we will demonstrate how we build a team with strong cohesion and create a comfortable working atmosphere. As we have learned during the TDPS course, we know that the foundation of a successful project is a good team. In this case, we designed our team name called FFAASSTT, shown in **Figure 1(a)**, which is the combination of eight letters and represents that our team is the combination of eight students and we will work as a whole team. Also, it means the car we build will run much faster and perform better than other teams. Additionally, we design our team logo shown in **Figure 1(b)**, which is just like a car and has the close relationship with the topic of team design and project skill. Moreover, we have our own team uniform and we do believe these things can help us build a strong and united team. The financial statement of this whole project for our team is shown in **Appendix Figure32**. The contribution details for each member are shown in in **Appendix Figure31**.

(a) Team name (b) Team logo

Figure 1: Team Introduction

## 1.2 Plan and Task Allocation

In this section, we will demonstrate the main task we need to finish, the basic strategies and task allocation for each member.

- **Plan and Basic Strategy:** to successfully manage the project, we firstly consider about what specific tasks should be handled with and the corresponding strategies. In **Figure 2**, the main tasks for the project are to drive the car, process the image information, perform obstacle avoidance and communication between car and PC. For car driving task, we use driving module based on the Arduino UNO, and for obstacle avoidance, ultrasonic sensor is applied. Moreover, for detecting road path and image recognition, we use camera based on the Raspberry pi to obtain the images information, and for communication part, HC-12 is applied to send the information to the computer. Raspberry pi and ultrasonic sensor can send the instructions to the Arduino UNO, which can control the movement of car.

- **Task Allocation:** after we design the strategies, then we start the task allocation. **Figure 3** shows the task allocation of our team, which we divide our

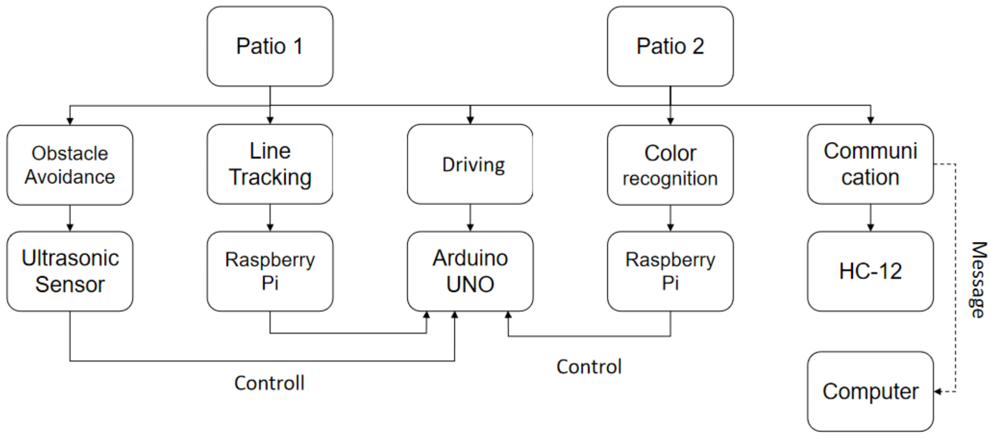


Figure 2: Plan and Basic Strategy

team into two groups: power group and visual group. The power group is mainly responsible for tasks based on Arduino including motor driving, ultrasonic obstacle avoidance and mechanical arm. The visual group is mainly responsible for tasks based on Raspberry Pi and computer including line tracking, color recognition and communication.

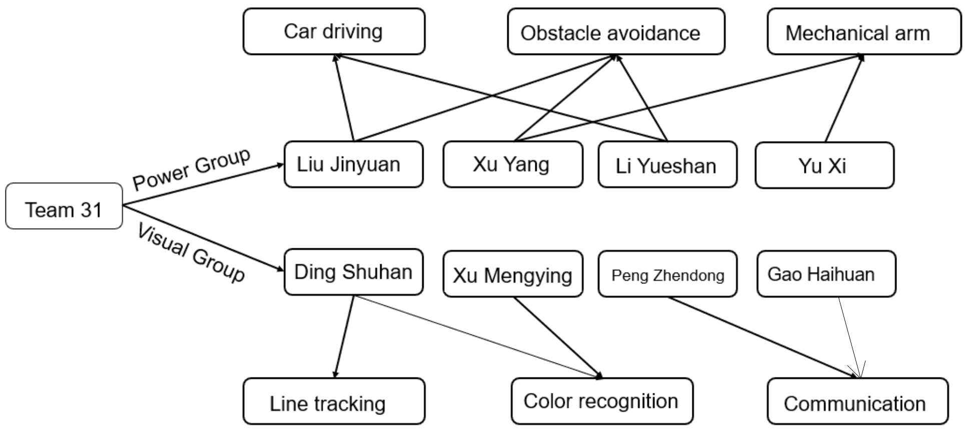
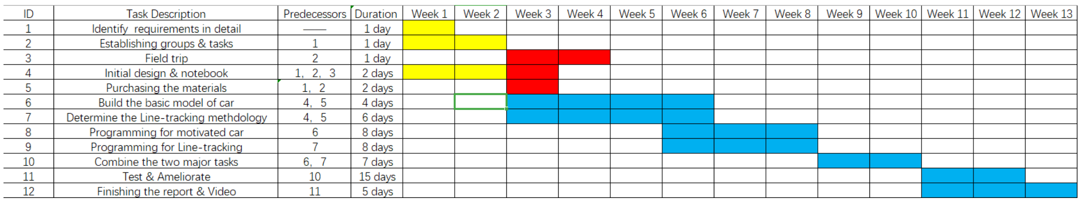


Figure 3: Task Allocation

## 1.3 Working Module

In this section, we demonstrate our working module. Our team has the group meeting each week, and during the group meeting each group would illustrate their working process and what problems occur. And team leader Liu Jinyuan also designed the Gantt Chart[1] which is shown in **Figure 4**, where the yellow part refers to the accomplished tasks, the red part refers to the task which have exact time arrangement and the blue part refers to the tasks whose timetables are not fixed. To record each group progress and experiment results, two lab notebooks are used for power group and visual group respectively in our team.



| ID | Task Description | Predecessors | Duration | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 | Week 11 | Week 12 | Week 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Identify requirements in detail | —— | 1 day | 🟨 | | | | | | | | | | | | |
| 2 | Establishing groups & tasks | 1 | 1 day | 🟨 | 🟨 | | | | | | | | | | | |
| 3 | Field trip | 2 | 1 day | | | 🟥 | 🟥 | | | | | | | | | |
| 4 | Initial design & notebook | 1, 2, 3 | 2 days | 🟨 | 🟨 | 🟥 | | | | | | | | | | |
| 5 | Purchasing the materials | 1, 2 | 2 days | | | 🟥 | | | | | | | | | | |
| 6 | Build the basic model of car | 4, 5 | 4 days | | | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | |
| 7 | Determine the Line-tracking methdology | 4, 5 | 6 days | | | 🟦 | 🟦 | 🟦 | 🟦 | | | | | | | |
| 8 | Programming for motivated car | 6 | 8 days | | | | | | 🟦 | 🟦 | 🟦 | | | | | |
| 9 | Programming for Line-tracking | 7 | 8 days | | | | | | 🟦 | 🟦 | 🟦 | | | | | |
| 10 | Combine the two major tasks | 6, 7 | 7 days | | | | | | | | | 🟦 | 🟦 | | | |
| 11 | Test & Ameliorate | 10 | 15 days | | | | | | | | | | | 🟦 | 🟦 | |
| 12 | Finishing the report & Video | 11 | 5 days | | | | | | | | | | | 🟦 | 🟦 | 🟦 |

Figure 4: Gantt Chart

# 2 Technical Details and Experiment Results for Power Part

## 2.1 Car Driving

### 2.1.1 Task Description

In this section, we introduce how to achieve the car driving along the track. The car driving is the most important part because it is the foundation of the whole project. All the other tasks are based on the movement of car. The following two subsections will introduce some of the hardware and principle we used, illustrate the experiment results and what we could improve in the future design.

### 2.1.2 Principle and Experiment Results of Car Driving

Firstly, we introduce the driving modules based on the Arduino[2, 3].

- **DC deceleration motors:** To give the car power, we utilize four DC deceleration motors for four wheels. The supplying voltage for the motor is 6-12V, which is given by outer battery. And the motor could provide a large range of speed with high torque, which adapts the characteristics of our tasks because the path in both patios are rough.

- **Driver chip:** In our project, TB6612FNG driver chip is applied to drive four DC deceleration motors. By controlling the level of driving chip AIN1, AIN2, BIN1, BIN2, PWMA and PWMB, the motor can be controlled to turn forward, reverse and stop. We could control AIN1 for high level, AIN2 for low level, BIN1 for high level, BIN2 for low level, and then by controlling PWMA, PWMB between 0 and 255 to control the speed of the car. One-way PWM controls the speed of the motor on one side of the car. AIN1, AIN2, PWMA, BIN1, BIN2, PWMB are connected to Arduino main control board 8, 7, 2, 4, 6 and 5 pins respectively. Among them, pins 6 and 5 have PWM function and support input PWM wave. The pin control logic table is shown in **Table 1**.

Table 1: Pin control logic table

| **AIN1** | **AIN2** | **BIN1** | **BIN2** | **PWMA** | **PWMB** | **A01/A02** |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 1 | Forward |
| 0 | 1 | 0 | 1 | 1 | 1 | Backward |
| 1 | 1 | 1 | 1 | 1 | 1 | Stop |
| 0 | 0 | 0 | 0 | 0 | 0 | Free Stop |
| X | X | X | X | X | X | Stop |

Then our first experiment is to build the overall architecture of our car, the details for installing the car are as follows:

**STEP1.** For how to assemble motor, we pierce a fixed piece from the front motor mounting hole of the car's bottom plate. Then we paste a fixed piece next to the DC motor mounting hole and pierce two long screws. Then, we install the motor in the designated position. Finally, we can screw two nuts on separately.

**STEP2.** For how to assemble Arduino, in the position where the Arduino motherboard is installed in the smart car, screw from the opposite side of the smart car's motherboard. Put it in front and screw it on with a double-pass copper column. Secondly, four Arduino mounting holes and four copper pillar holes are paired, and then three screws are screwed respectively.

**STEP3.** For Installation of adapter plates and wheels, firstly insert the insert on the back of the adapter board into the black row mother hole of the Arduino motherboard. Then, the red and white wire connectors on the voltage detection module are connected to the voltage interface of the adapter board. Finally, two wheels are installed respectively.

Also there are two ways we could choose to drive the car turning the direction.

- **Turning during the motion:** in this turning model, one side wheels of the car will keep stationary, and the other side of wheels will rotate, which means when the car turns left, the right side wheels will rotate and left side wheels will keep static.
- **Turning in place:** in this turning model, one side of our wheels will turn backwards and the other side of wheels turns forwards. In this case, the car will turn in place.

Based on the experiment we did in the competition venue, we found that the radius of circles are not very long, which means it is not suitable for the car turning during the motion. Moreover, owing to our basic methodology, we choose to use the raspberrypi to control the movement of the car. And process one picture taken from the camera needs time, therefore during the lull our car is supposed to stop and wait for the next instruction. Hence, turning in place is the appropriate turning model.

### 2.1.3 Conclusion and Recommendation

- **Strengths:** Based on the overall structure and turning methodology we applied, the driving of our car becomes very stable. Our car has a wide range of velocity which can be utilized, and the higher power of the motor can drive the car on the rough ground.
- **Weaknesses and improvement:** however, our car cannot go very straight because It is almost impossible that the driving characteristics of the four motors are identical. Although this deviation does not have a big effect on our car because we use camera to detect the path, in the future design we will apply PID algorithm to control the driving of car.

## 2.2 Obstacle Avoidance

### 2.2.1 Task Description

The obstacle avoidance function is used in both patio 1 and patio 2, which will be demonstrated in detail. Ultrasonic distance measuring module and priority algorithm are used to achieve obstacle avoidance function.

- **Patio 1:** for patio 1, as shown in **Figure 5**, in patio 1 task 2, beacon1 is put in front of the bridge. When the distance between the car and beacon1 is less than 70 centimeters, the car will turn right and cross the bridge. To help the car do the line tracking process again, beacon2 is put in the position shown in **Figure 5**. When the distance between the car and beacon2 is less than 55 centimeters, the car could turn left and the camera in front of our car will detect the line again.

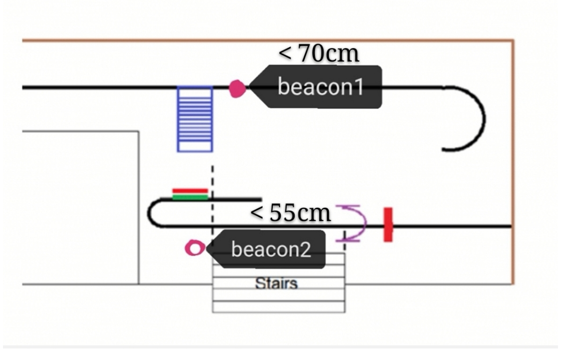


Figure 5: The location of beacon in patio 1

- **Patio 2:** In patio 2 task1, the car can move to the release point with the help of beacon and walls. We put a beacon in the position shown in **Figure 6** and use a priority algorithm combined with servo motor and sensor to complete this task. The car will detect whether there are obstacles in three directions and the priority of them are front, right and left. All the appropriate distances are shown in **Figure 6**.

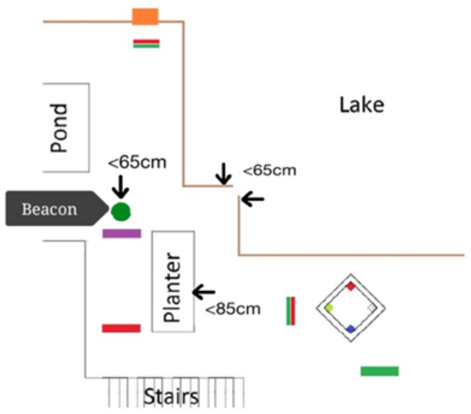


Figure 6: The location of beacon in patio 2

### 2.2.2 Principle and Experiment Result of Obstacle Avoidance

- **Principle of sensor:** The ultrasonic distance measuring sensor[4] operation schematic is shown in **Figure 7**. Give the sensor a 10us high level signal to trigger the ranging process. The ultrasonic module sends eight 40khz square waves in one direction and starts timing. Ultrasonic waves travel through the air and return as soon as they encounter obstacles. At the same time, the module detects if there is a signal return. If signal is returned, the sensor stops timing and outputs an echo signal, and the pulse width of the echo signal is proportional to the distance.

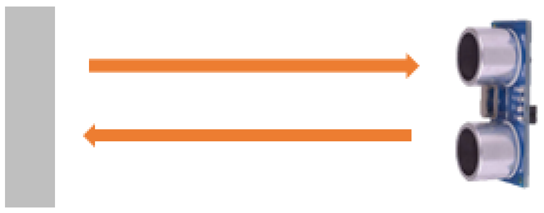


Figure 7: Principle of sensor

- **Obstacle avoidance methodology:** In the beginning, we found that only one sensor could not meet multiple situations at the same time for the reason that car must turn certain angles when detecting distance of different direction. Then we considered using three sensors to detect the distance in three directions simultaneously. However, the use of three sensors greatly increases the complexity of code and car structure, and the direction of the measurement is fixed. To solve this problem, we tried to use a combination of a servo motor and an ultrasonic distance measuring sensor. In this case, the sensor can turn in any direction within the range of 0 to 180 degrees. This makes it possible to judge complex situations with the simplest structure. The combination model of ultrasonic distance measuring module and servo motor to achieve the obstacle avoidance function is shown shown in **Figure 8**.

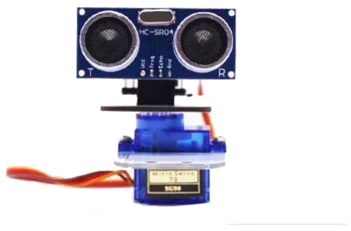


Figure 8: The structure of sensor

- **Priority algorithm:** to guide the car drive to the correct releasing point and effectively use the ultrasonic distance sensor, we develop a priority algorithm. The flow chart of the priority algorithm is shown in **Figure 9**. According to the algorithm and the beacon we located shown in **Figure 6**, the whole process is: the car will turn right when meets the planter. In the corner, as shown in **Figure 6**, there are obstacles in two directions, so car will turn left. When meet the beacon, car will turn right. After that the car will manage to move to the release point.

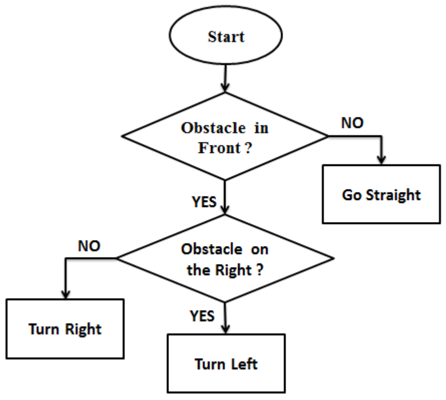
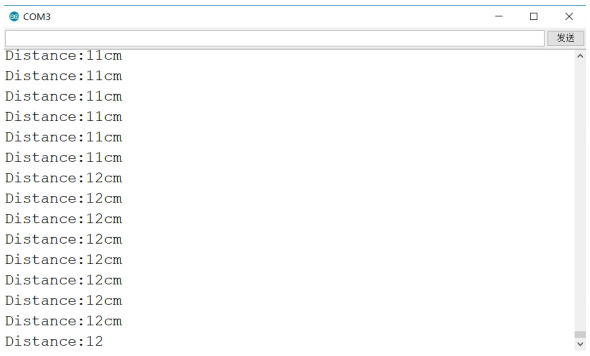


Figure 9: Priority algorithm

- **Experiment Process:** before we start using the ultrasonic module, The related parameters should be tested before using the module. First, as **Figure 10** shows, we use the serial communication to display the distance detected by the ultrasonic module. Next the module is installed on the car. We brought the car to the lakeside, using serial communication to find the appropriate obstacle avoidance distances.



```
Distance:11cm
Distance:11cm
Distance:11cm
Distance:11cm
Distance:11cm
Distance:11cm
Distance:12cm
Distance:12cm
Distance:12cm
Distance:12cm
Distance:12cm
Distance:12cm
Distance:12cm
Distance:12cm
Distance:12
```

Figure 10: Experiment result

### 2.2.3 Conclusion and Recommendation

To achieve this obstacle avoidance function, we choose a conventional method of using ultrasonic distance measuring module. According to the task requirements, we make improvements to this approach by using a combination of servo motor and senor and priority algorithm. These improvements make it possible for our car to make corresponding judgment efficiently with simplest structure. In the experiment, we repeatedly measured the appropriate obstacle avoidance distance and calculated the average to get a more accurate number. We also adjust the position of the module to achieve the best recognition results. With these improvements, car successfully realized the obstacle avoidance function

## 2.3 Mechanical Arm

### 2.3.1 Task Description

In this part, we demonstrate how the car delivers the fish food. A mechanical arm has been applied for this task, which is based on steering servos, which are controlled by PWM wave provided by Arduino.

### 2.3.2 Principle and Experiment Results of Tracking Algorism

The model of our mechanical arm is shown in **Figure 11**. There are three steering servos in our Mechanical arm, controlling different directions of arm respectively. The control PWM wave can determine the rotation angle of the output shaft, driving the gear group and structure. The code sets the initial values and the range of rotation angle of output shaft, determines the state of arm. And the function of the three servos are shown in below:

- First steering servo: control the jaw which catches fish food and releases.
- Second steering servo: control the movement of arm in x direction.

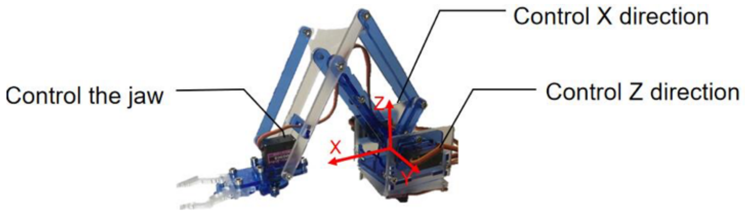


Figure 11: Model of mechanical arm

- Third steering servo: control the movement of arm in z direction.

In initial state, the jaw catches fish food, and the arm shrinks. The rotation angle ranges of three steering servos are in the **Table 2**. After the car stops, the output

Table 2: Pin control logic table

| SERVO JAW | SERVO X | SERVO Z |
|---|---|---|
| 30° | 50° | 60° |
| 120° | 160° | 120° |

shaft of second and third steering servos rotate at the same time, extending the mechanical arm to the box. Then the first steering servo control the jaw releasing the fish food after a delay time. Finally, the first steering servos return to initial state, and the arm return to initial state in a delay time.

### 2.3.3 Conclusion and Recommendation

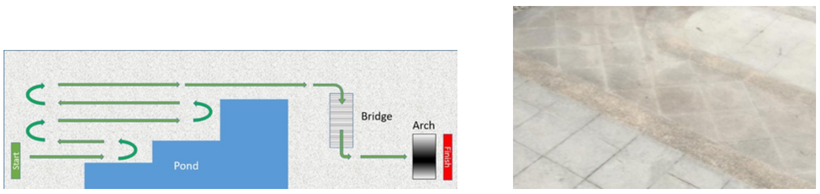
The mechanical arm can achieve the task and it is stable. It never dropped down the fish. Though the mechanical arm run perfectly, we cannot catch some heavy things. In the future, we will build a more robust arm our our car.

# 3 Technical Details and Experiment Results for Visual Part

## 3.1 Line Tracking

### 3.1.1 Task Description

This task is to identify the boundary between the two ground materials to complete the tracking and the whole path includes partial straight lines and four turns, as shown in **Figure 12(a)**. The difficulty in this task is that two road surfaces do not have a clear color difference and the ground is unevenness, which causes a lot of noise. The road to be detected is shown in **Figure 12(b)**, which is an outdoor road made up of two materials. If Canny algorithm is directly used to binarize image, it is difficult to form a stable boundary line to determine the direction of the car. As for this situation, the overall flow chart is shown in **Figure 13**, Python and OpenCv are used to implement the image processing part.



(a) (b)

Figure 12: (a)Schematic diagram of the Patio 1 task; (b)The road conditions

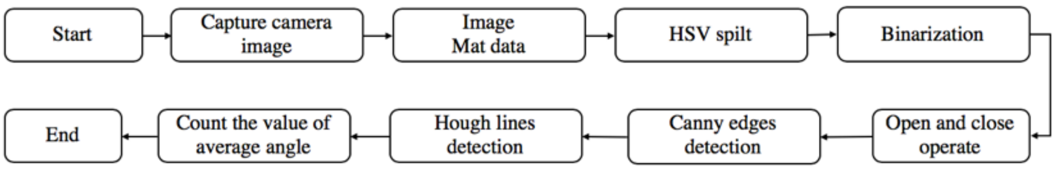


Figure 13: The image processing flow chart

### 3.1.2 Principle and Experiment Results of Tracking Algorism

- **HSV spilt processing:** HSV (Hue, Saturation, Value)[5] is a color space created by A.R.Smith based on the visual characteristics of the color. It can be represented by a hexagonal cone model, as shown in **Figure 14**. The parameters represented by HSV are hue (H), saturation (S), and brightness (V). In the process of image binarization, the unimportant parts of the image are filtered out by setting thresholds for each channel of the HSV value. In this task, the HSV filters out region outside the path. The HSV values of the left and right roads in the HSV window is detected separately. The results are shown in **Figure 15(b)**, **Figure 16(b)**. The above process has divided the image into two parts, which highlights the shape of the left and right boundary lines that need to be paid attention to in the line tracking task. The use of HSV filtering basically achieves image binarization, removing noise and greatly increasing the accuracy and stability of subsequent processes.

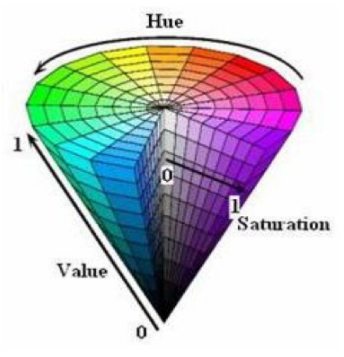


Figure 14: The HSV color space

- **Image Dilate:** Dilate is a basic morphological operation provided by OpenCv and is commonly used to eliminate salt and pepper noise in binarized images[6]. The edges in the image expand outward to fill the voids in the image, eliminating slender and unimportant noise.The dilate algorithm has a library function of OpenCv, which is used to enlarge and overlap the discrete pixels. The image results after dilating is shown in **Figure 15(c)**, **Figure 16(c)**, whose left and right parts of the image have been clearly divided.

- **Canny edge detection operation:** The Canny edge detection operator is a multi-level edge detection algorithm proposed by John F. Canny in the paper "A Computational Approach to Edge Detection" in 1986[7]. This algorithm can extract useful structural information from different visual objects to reduce the amount of data processing. It has been widely used in the development of various computer vision systems. The Canny edge detection is used to trace the boundary between the two colors regions. This equation will return a binary matrix with only 1 position at the boundary and 0 at the remaining position, that is, only the boundary line is displayed as white. As shown in **Figure 15(d)**, **Figure 16(d)**, in the image processed on the right side, only the middle boundary line remains. Since the program requires at least 120 consecutive pixel points to generate a Hough line, it is necessary to use a Gaussian blur equation to compress the boundary line and eliminate discontinuities on the boundary line to obtain a smoother curve. The result of Gaussian blur is shown in **Figure 15(e)**, **Figure 16(e)**.

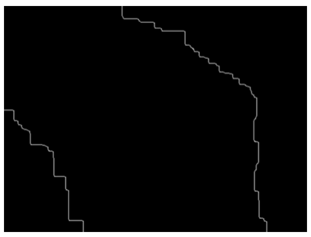
- **Hough Line Detection:** Hough Transform was first proposed by Paul Hough in 1962, which is a feature extraction technique in image processing. It uses a voting algorithm to detect objects with specific shapes[8]. The classical Hough transform is used to detect straight lines in images. The basic principle of Hough line detection is to exploit the duality of points and lines. The standard Hough transform under OpenCv can be implemented with the

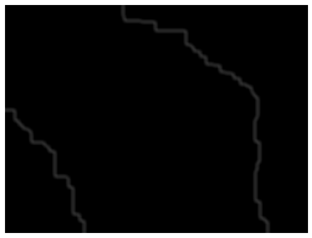
Houghlines function. The code requires that 120 consecutive pixels be detected to generate a Hough line and result of adding a Hough line is shown in **Figure 15(f)**, **Figure 16(f)**. The coordinates of this line will be expressed in polar coordinates, then we calculate the slope k of the detected edge line and use it for subsequent control determining the direction of the car.

(a) The original image

(b) The HSV filtering result

(c) The dilate processing result


(d) The Canny processing result


(e) The Gaussian Blur processing result

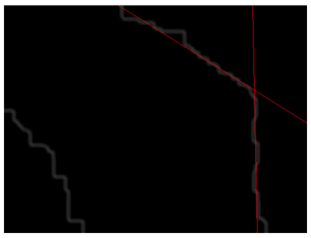
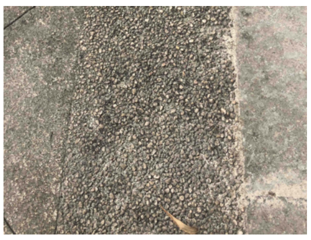
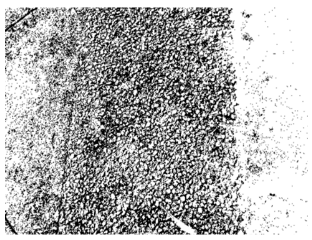
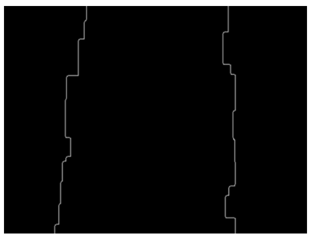

(f) The Hough line detection result

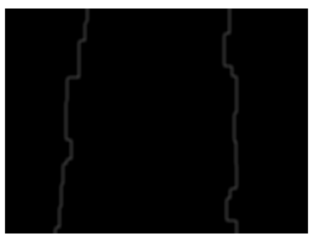
Figure 15: The line tracking results of turning situation


(a) The original image


(b) The HSV filtering result

(c) The dilate processing result


(d) The Canny processing result


(e) The Gaussian Blur processing result

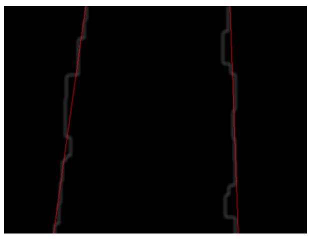

(f) The Hough line detection result

Figure 16: The line tracking results of going straight situation

### 3.1.3 Conclusion and Recommendation

The car tracking algorithm design introduced in this paper is based on the image processing of camera capture. Firstly, the image is filtered by HSV color, and then the image boundary is extracted by Dilate, Canny edge detection and Gaussian blur. Finally, the Hough line detection function is used to generate the direction line and calculate the slope. The process is simple to program and easy to implement, and it has reference value for using image processing to solve the tracking problem. However, the threshold setting for each library functions currently only stays on multiple attempts and needs to be adjusted for different environments.

## 3.2 Color Recognition

### 3.2.1 Task Description

In patio2 task 1, the grayscale recognition method is used to identify color square which is placed before, then guides the car to the corresponding area. In the patio2 task2, the camera finds the colored beacon and feedback the center coordinates, guiding the car from planter to feeding site.

### 3.2.2 Principle and Experiment Results of Color Recognition

The photoresistor is made by utilizing the photoconductivity of the semiconductor to change the resistance value according to the intensity of the incident light. The incident light intensity, the resistance is reduced, vice versa. Different colors are recognized according to the principle that the illuminance of different colors of light differs depending on the resistance of the returned light. Its pin definition can be seen in **Figure 17**. Where pin 1 for power supply, pin 2 is for Ground and pin 3 is for output signal. And the color recognition module is connected to A5 on the Arduino. The final results can be seen from **Figure 18** the

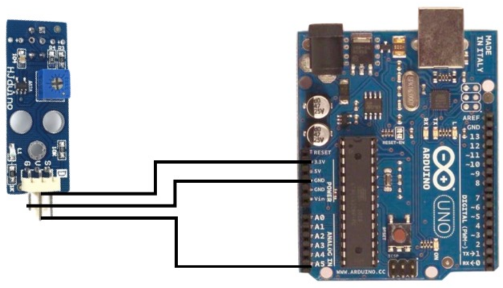
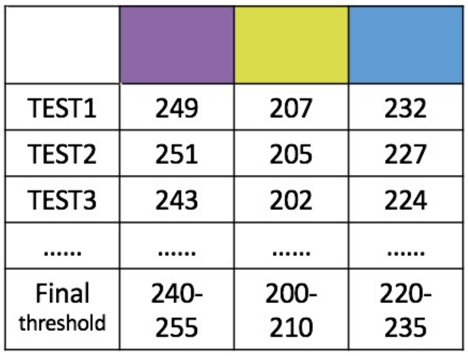


Figure 17: Connection between model and Arduino

intensity of purple is between 240-255 of yellow is between 200-210 of blue is between 220-235.



| | 🟪 | 🟨 | 🟦 |
|---|---|---|---|
| TEST1 | 249 | 207 | 232 |
| TEST2 | 251 | 205 | 227 |
| TEST3 | 243 | 202 | 224 |
| …… | …… | …… | …… |
| Final threshold | 240-255 | 200-210 | 220-235 |

Figure 18: Final results from grayscale module

### 3.2.3 Conclusion and Recommendation

- **Advantage:** Grayscale module has a wide operating voltage range and can work normally even when the power supply voltage fluctuates greatly. It outputs a continuous analog signal, so it is easy to determine the reflectivity of the object through the a/d converter or a simple comparator. In this case, the car recognizes the color and then goes to the corresponding position.
- **Drawbacks and Improvement:** The intensity of the external light has a great influence on its return value, and the fluctuation of the same color is very obvious, which will directly affect the detection effect. In this case, a LED light is installed at the bottom of the module. The light source emitted by the LED makes the light source of the grayscale module recognition part stable to reduce the influence on the test result, thereby realizing the accurate test result.
  According to its working principle, the photosensitive probe determines the color depth of the detection surface according to the intensity of the light reflected from the detection surface, so the accuracy of the measurement is directly related to the distance from the sensor to the detection surface. The vibration of the body during the movement of the robot also affects its measurement accuracy. In his case, we paste the grayscale module on the bottom of the cart and fix it to the cart with cardboard around the module to slow down the impact of the vibration when the car is driving.

## 3.3 Color detection based on OpenCV

### 3.3.1 Task Description

In task2, we drive the car to the designated location for feeding by placing a color block. The overall flow chart is shown in **Figure 19**, Python and OpenCv are used to implement the image processing part.

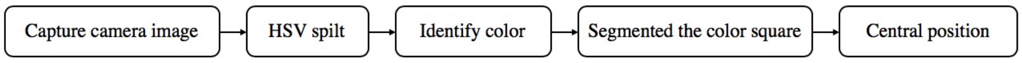
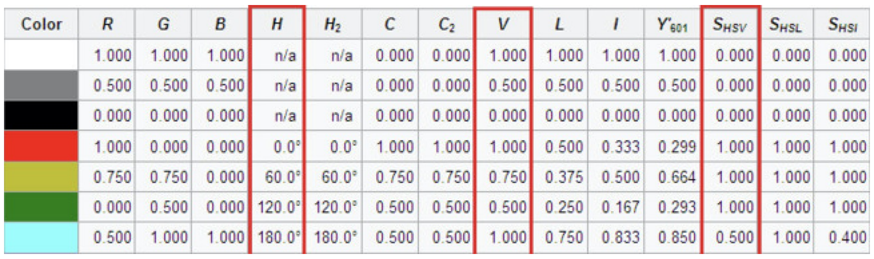


Figure 19: The process of color recognition

### 3.3.2 Principle and Experiment Results of Detecting Color

To help the car drive to the correct location, the commonly used models in digital image processing are rgb (red, green, blue) models and hsv (hue, saturation, brightness), and our usual pictures are generally rgb models. The hsv model is more in line with the way people describe and interpret colors. The color description of hsv is natural and very intuitive for people. After experiments, the value of recognizing blue is h between 100 and 140, and both s and v are between 90 and 255. Some basic color hsv values can be taken from **Figure 20**. By setting



| Color | $R$ | $G$ | $B$ | $H$ | $H_2$ | $C$ | $C_2$ | $V$ | $L$ | $I$ | $Y'_{601}$ | $S_{HSV}$ | $S_{HSL}$ | $S_{HSI}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⬜ | 1.000 | 1.000 | 1.000 | n/a | n/a | 0.000 | 0.000 | 1.000 | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| ▨ | 0.500 | 0.500 | 0.500 | n/a | n/a | 0.000 | 0.000 | 0.500 | 0.500 | 0.500 | 0.500 | 0.000 | 0.000 | 0.000 |
| ⬛ | 0.000 | 0.000 | 0.000 | n/a | n/a | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 🟥 | 1.000 | 0.000 | 0.000 | 0.0° | 0.0° | 1.000 | 1.000 | 1.000 | 0.500 | 0.333 | 0.299 | 1.000 | 1.000 | 1.000 |
| 🟨 | 0.750 | 0.750 | 0.000 | 60.0° | 60.0° | 0.750 | 0.750 | 0.750 | 0.375 | 0.500 | 0.664 | 1.000 | 1.000 | 1.000 |
| 🟩 | 0.000 | 0.500 | 0.000 | 120.0° | 120.0° | 0.500 | 0.500 | 0.500 | 0.250 | 0.167 | 0.293 | 1.000 | 1.000 | 1.000 |
| 🟦 | 0.500 | 1.000 | 1.000 | 180.0° | 180.0° | 0.500 | 0.500 | 1.000 | 0.750 | 0.833 | 0.850 | 0.500 | 1.000 | 0.400 |

Figure 20: HSV value of common color

the hsv value of the target color we can eliminate other colors and only recognize the target color. The results can be seen in **Figure 21**. In the field of computer

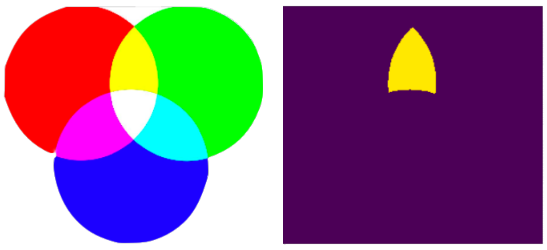


Figure 21: Test results of opencv

vision and image processing, the moments of an image are often used to describe the shape of an object in an image. The center value of the color can be calculated by these process. The experiment and test Result is shown in **Figure 22**.

### 3.3.3 Conclusion and Recommendation

In patio 2, we first use the grayscale module to complete the color recognition and guide the car to the corresponding position, then identify and calculate the

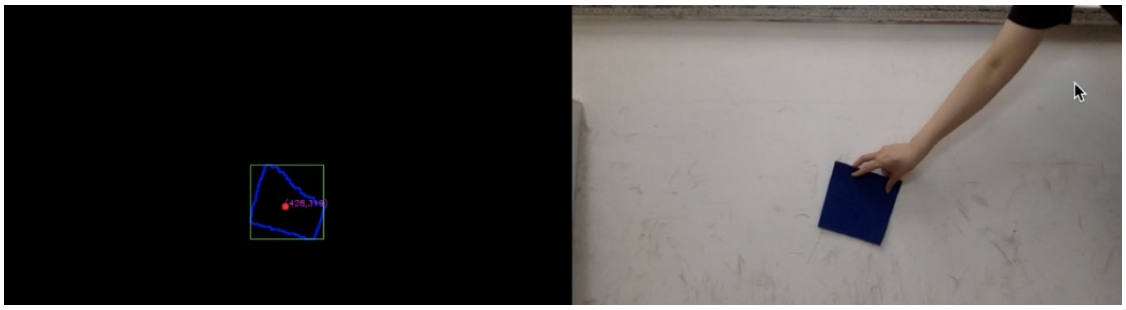


Figure 22: Test results of center calculation

color block placed at the feeding place and its center value through the OpenCV algorithm, and finally guide the car to the feeding place.

# 4 Technical Details and Experiment Results for Wireless Communication

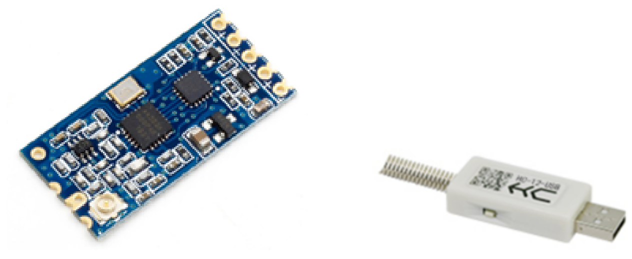
## 4.1 Task Description

In patio 2, task 3, our car needs to stop at the planter area and transmit message regarding teamąŕs information as well as time signal to laptop. To achieve this task, wireless communication module HC-12 and clock module DS-1302 are used for transmitting message wirelessly and recording the time during the process respectively. In the following part, we will introduce basic operations of two important modules mentioned above and explain the principles of our groupąŕs wireless communication process in task 3 based on the combination of these modules. Also, result and future work will be discussed in last part.

## 4.2 Communication Module and Principle

The communication modules we used are demonstrated below.

- **HC-12:** HC-12 shown in **Figure 23(a)** is a multichannel embedded wireless serial data transmission module[9], which is widely used in integral communication system. In our task, HC-12 is paired with a helix external antenna[11] to increase efficient length, the communication distance can be beyond 1 kilometer in outdoor environment. Particularly, in task 3, we use

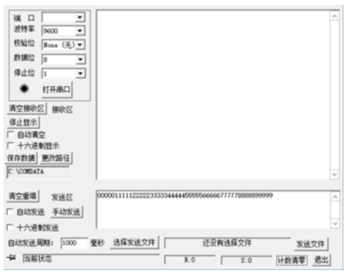


(a) (b)

Figure 23: Outline of $HC-12$ and $HC-12$ USB

  HC-12 USB shown in **Figure 23(b)**, which can help PC receive the message from HC-12 equipped on Arduino.

- **Detection Window:** Based on the two modules HC-12 USB and HC-12, then to detect the message transmitted from HC-12 on Arduino, we need a detection window to show the result received from HC-12 USB on PC[12]. To better illustrate it, the surface of this serial tooląŕs software window is shown in **Figure 24**. It can be seen clearly that the transmission parameters of HC-12 USB can be changed easily by us in the tooląŕs surface to improve communication performance.



串口
波特率 9600
校验位 None (无)
数据位 8
停止位 1
C:\COMDATA
0000011111222223333344444555556666677777888889999
自动发送周期: 1000 毫秒
还没有选择文件
R:0
S:0

Figure 24: Operation surface of HC communication tool

- **Clock Module DS-1302:** DS-1302 clock module shown in **Figure 25** is a real-time clock circuit with high performance and low power which can get an accurate clock signal concerning year, month, day, week, hour, minute and second from internal crystal oscillator with 32.768Khz. Some details could be referred in article written by Yao in 2006. [14]The reason we choose DS-1302 is that it is convenient for us to operate the time since it has an entire and strong library called <DS1302.h> based on Arduino[15]. Hence, we only construct an array to store the time information like day, hour, minute and operate it once the time information needs to be transmitted at specific moment.

Figure 25: Outline of DS-1302

Now we can combine the modules and tool we explained above to achieve the whole wireless communication process, the schematic diagram of it in our task is shown in **Figure 26**. In the **Figure 26**, the device, Arduino on the left sends the serial port data containing team and time information calculated above to the traditional HC-12. After receiving the serial port data, the RXD port of this HC-12 automatically communicate wirelessly by radio waves with HC-12 USB connected to PC. Similarly, PC on the right can be automatically received and the serial port data sent by the original left device is restored from TXD of HC-12 USB. Apparently, the process also can be achieved reciprocally.

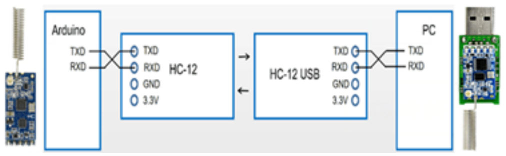
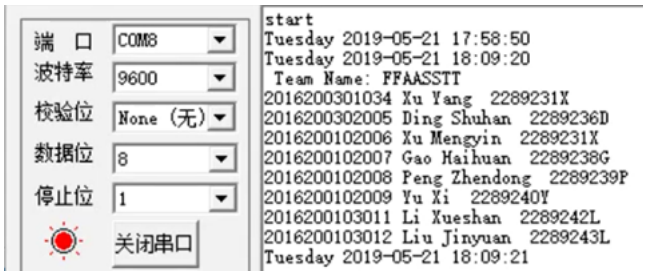


Figure 26: Schematic diagram of WCN in task 3

## 4.3 Conclusion and Recommendation

- **Results:** Based on the content we discussed in experimental details, now we consider how do we define the ąřtransmission timeąś, referring to the time the car transmits the message to PC. Apparently, we can define an input signal as ąřan alarm clockąś to remind Arduino should transmit the message to PC by using different sensors at the specific position. Sensors can capture information from the environment. In our task, to simplify it, we choose voltage signal as an input signal[16]. Hence our communication process can be described as: the clock starts recording initial time and HC-12 transmits starting signal to PC when our car starts running. Then after the car arrives at the destination, the car gets voltage signal at the same time. Arduino records time at this moment and transmits it as well as the team information to PC. The Result of our communication process is shown in **Figure 27**.



端 口: COM8
波特率: 9600
校验位: None (无)
数据位: 8
停止位: 1

```
start
Tuesday 2019-05-21 17:58:50
Tuesday 2019-05-21 18:09:20
 Team Name: FFAASSTT
2016200301034 Xu Yang  2289231X
2016200302005 Ding Shuhan  2289236D
2016200102006 Xu Mengyin  2289231X
2016200102007 Gao Haihuan  2289238G
2016200102008 Peng Zhendong  2289239P
2016200102009 Yu Xi  2289240Y
2016200103011 Li Xueshan  2289242L
2016200103012 Liu Jinyuan  2289243L
Tuesday 2019-05-21 18:09:21
```

Figure 27: Communication Results

- **Discussion:** However, wireless communication is always faced with multiple problems like ISI, IBI, CCI and Multipath effect[17] in some slight hostile conditions, especially outdoor environment. A simple method to release the fading in our communication task is setting different transmission parameter compared with other team like changing channel frequency, baud rate and adding parity based on the HC-12 operation document to resist the interference and improve reliability and efficiency of communication process. Based on the compromise of these two demands, some important transmission parameters are shown in **Table 3**. Because of the characteristic of half-duplex communication between HC-12[18], the modules cannot send and receive data at the same time in our task. This will be improved in the future.

Table 3: Communication parameter in our task

| **Baud Rate** | **Channel Frequency** | **Parity** | **Communication Type** |
|---|---|---|---|
| $9600Hz$ | $433MHz$ | None | Fu3 |

# 5 Task Integration

After finishing all the individual task, how to integrate these tasks and package them into one main program should be considered. Hence, in this section we discuss the programming logic by utilizing the state machine[19] for both patio 1 and patio 2 respectively.

## 5.1 Patio1

The state machine for patio 1 is shown in **Figure 28**. There are three states for our car in patio 1 including obstacle avoidance, stop and moving decision.

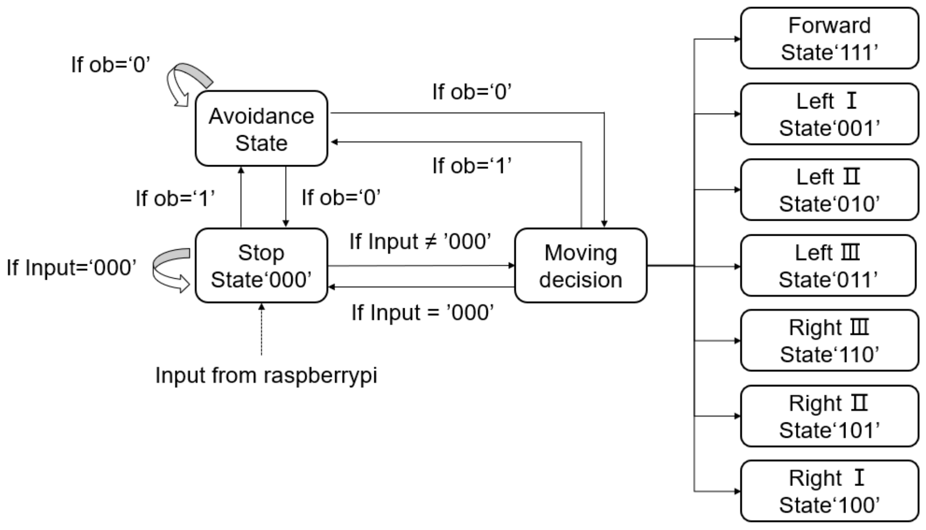


Figure 28: State machine for patio 1

- **Stop State:** During the stop state, the camera will take a picture reflecting the road path and raspberry pi will generate instruction based on the image information. Also, in this state Arduino waits for instructions which will be came from either raspberry pi or ultrasonic sensor.
- **Moving Decision State:** During this state, the movement of our car will be determined based on the information obtaining from the raspberry pi. The camera takes photo and the raspberry pi detects the correct path. Thus, the Raspberry Pi can calculate the angle of the path and based on the angle it will figure out the derivation of the car. Then to give the instruction to the car, we define three analog output pins for the communication between the raspberry pi and Arduino, which each pin has two states containing 1 and 0, and in this case we can have eight driving modules for our car shown in both **Figure 28** and **Table 4**. Based on the eight driving modules our car can move along the correct path.

Table 4: True table of controlling

| State | Pin23 | Pin24 | Pin25 |
|---|---|---|---|
| STOP [BUSY] | 0 | 0 | 0 |
| LEFT $[-90°, -50°]$ | 0 | 0 | 1 |
| LEFT $[-50°, -30°]$ | 0 | 1 | 0 |
| LEFT $[-30°, -15°]$ | 0 | 1 | 1 |
| RIGHT $[15°, 30°]$ | 1 | 0 | 0 |
| RIGHT $[30°, 50°]$ | 1 | 0 | 1 |
| RIGHT $[50°, 90°]$ | 1 | 1 | 0 |
| STRAIGHT $[-15°, 15°]$ | 1 | 1 | 1 |

- **Obstacle Avoidance State:**whenever the car detect obstacle in front of it, it will immediately go into the obstacle avoidance state containing going through the bright instruction. Therefore, this state has the priority in our program.

## 5.2 Patio 2

For patio 2 task 1, the state machine is shown in **Figure 29**, where there are three states for our car containing moving forward, line Tracking and color recognition.

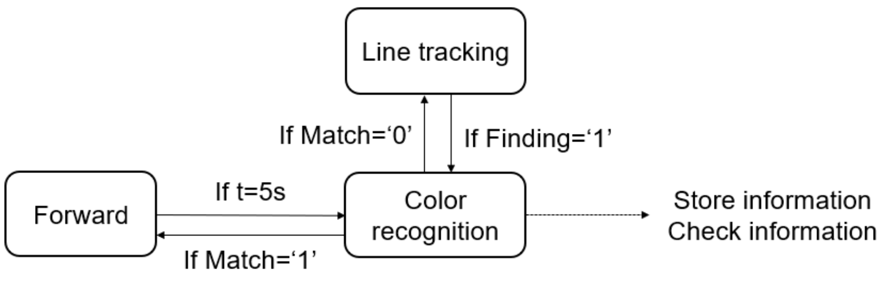


Figure 29: State machine for patio 2

- **Forward:** During this state, car will move forward for five seconds. This state will be performed at the beginning of the patio 2 and after the car detect the correct color.
- **Color Recognition:** During this state, the car will detect and store the color intensity. At the beginning this color intensity equals to zero, and if the color intensity is not matching, the car will go into line tracking state.

- **Obstacle Avoidance State:** During the line tracking state, the car will move along the small crack between these tiles. Whenever the car detects the color swatch, it will perform color recognition.

For patio 2 task 2, to guide the car driving to the specific releasing point, the state machine presenting our program logic is shown in **Figure 30**, where there are four states for our car containing moving state, search beacon, checking distance and releasing item. The state searching beacon and moving state are running simultaneously

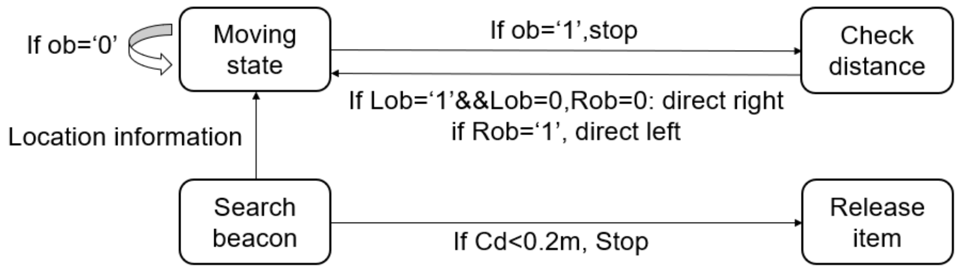


Figure 30: State machine for patio 2

- **Moving State:** During this state, car can perform different moving modules based on the different inputs. If there is an obstacle in front of our car, it will go into the checking distance state.
- **Check Distance State:** During this state, the car will stop and owing to the priority algorithm we have discussed, the car will perform different moving modules.
- **Searching beacon:** this state is running continuously, the camera will search for the beacon and give the instruction to control the movement of the car.
- **Releasing Item:** in this state, the mechanical arm will releasing the item into the box.

## 6 Overall Conclusion

For the patio 1 and patio 2, our car finally could finish all the tasks successfully within ten minutes and eleven minutes respectively. The movement of our intelligence car is stable due to the safety mechanism we build. In the future, there are still some aspects we can improve our car.

- Firstly, PID algorithm could be applied to guide the car driving along the straight line.
- Secondly, the threshold setting for line tracking algorithm only stays on multiple attempts and needs to be adjusted for different environments.
- Thirdly, the number of GPIO ports used for the communication between the Arduino and Raspberry Pi can be increased to enable high accuracy control.

## References

[1] Nahler M P G . Gantt chart[M]// Dictionary of Pharmaceutical Medicine. Springer Vienna, 2009.

[2] Sarik J , Kymissis I . Lab kits using the Arduino prototyping platform[C]// Frontiers in Education Conference. IEEE, 2010.

[3] Lu G Y , Wu T . Structure and Motion Analysis of Intelligent Tracking Car Based on SolidWorks and Arduino[J]. Advanced Materials Research, 2014, 951:11-14.

[4] Bischoff O , Wang X , Heidmann N , et al. Implementation of an ultrasonic distance measuring system with kalman filtering in wireless sensor networks for transport logistics[J]. Procedia Engineering, 2010, 5(none):196-199.

[5] Smith, Ray A . Color gamut transform pairs[C]// SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques. ACM, 1978:12-19.

[6] Zhang F L , Cheng M M , Jia J , et al. ImageAdmixture: Putting Together Dissimilar Objects from Groups[J]. IEEE Transactions on Visualization and Computer Graphics, 2012, 18(11):1849-1857.

[7] Canny, John. A Computational Approach to Edge Detection[J]. 1986, PAMI-8(6):679-698.

[8] Duda R O , Hart P E . Use of the Hough transformation to detect lines and curves in pictures[J]. Communications of the ACM, 1972, 15(1):11-15.

[9] Pack D J, Barrett S F. Microcontroller theory and applications: HC12 and S12[M]. Prentice Hall Press, 2007.

[10] Kandalkar S, Kadam P, Kadu A B. Arduino Based Head Gesture Controlled Robot Using Wireless Communication[J].

[11] Milligan T A, Milligan T A, Milligan T A, et al. Modern antenna design[M]. New York: IEEE press, 2005.

[12] HC-12: Communication Testing tool. 2016
`http://www.hc01.com/service`$_{d}ownload?keywords = type = tool$

[13] HC-12: Communication Module Specification. 2016
`http://www.hc01.com/service`$_{d}ownload?keywords = type = data$

[14] Defa Y, Honglin Z. How to use the trickle charge timekeeping chip DS1302[J]. Information Technology and Informatzation, 2006, 1: 92-94.

[15] Arduino-Home: An Fast-effective DS1032 Design Method. 2015
`https://www.arduino.cn/thread-75308-1-21.html`

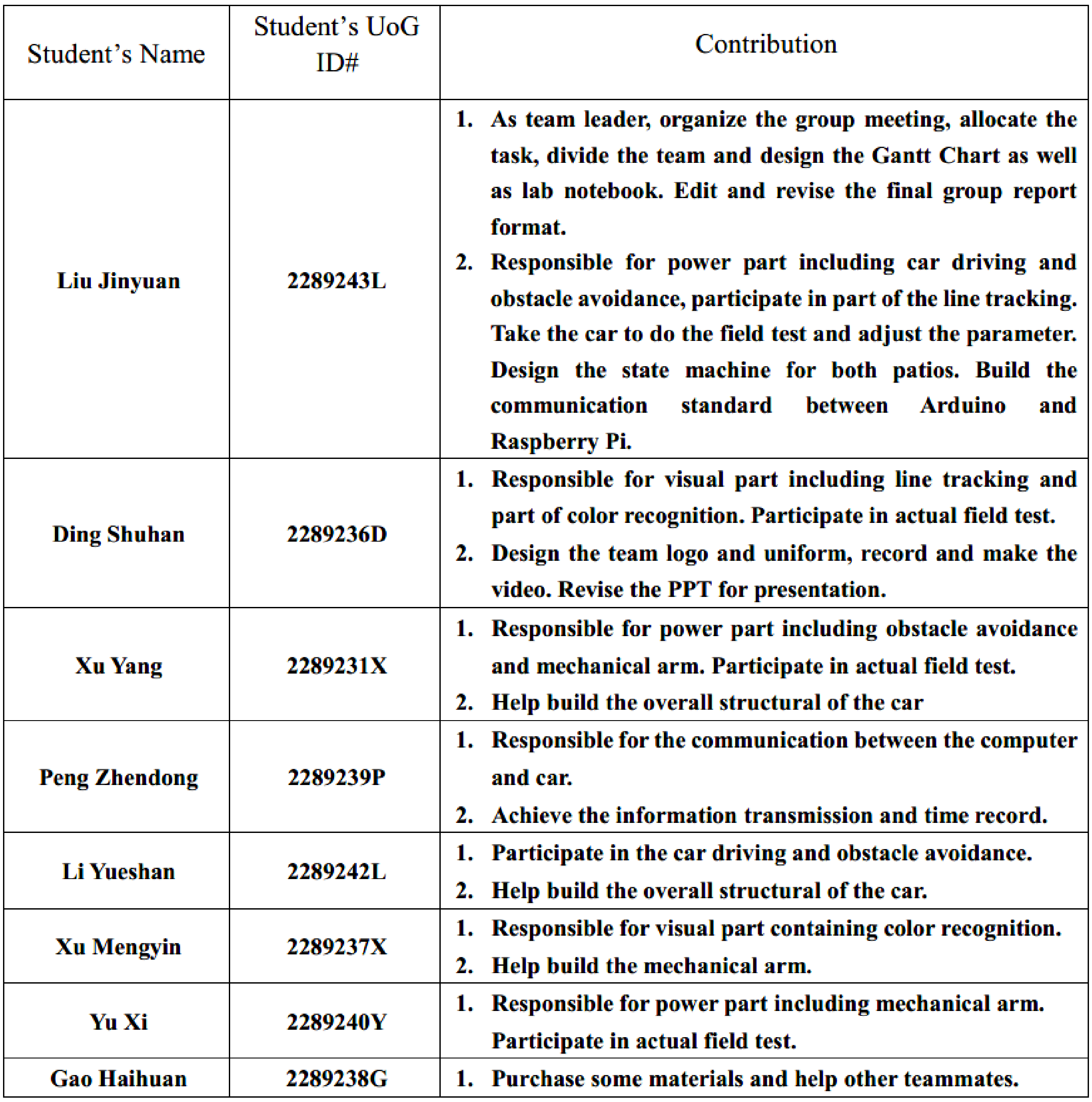
[16] Perez M S, Carrera E. Time synchronization in Arduino-based wireless sensor networks[J]. IEEE Latin America Transactions, 2015, 13(2): 455-461.

[17] Tse D, Viswanath P. Fundamentals of wireless communication[M]. Cambridge university press, 2005.

[18] HC-12: Communication Module Data package. 2016
`http://www.hc01.com/service`$_{d}ownload?keywordstypedata_{p}ackage$

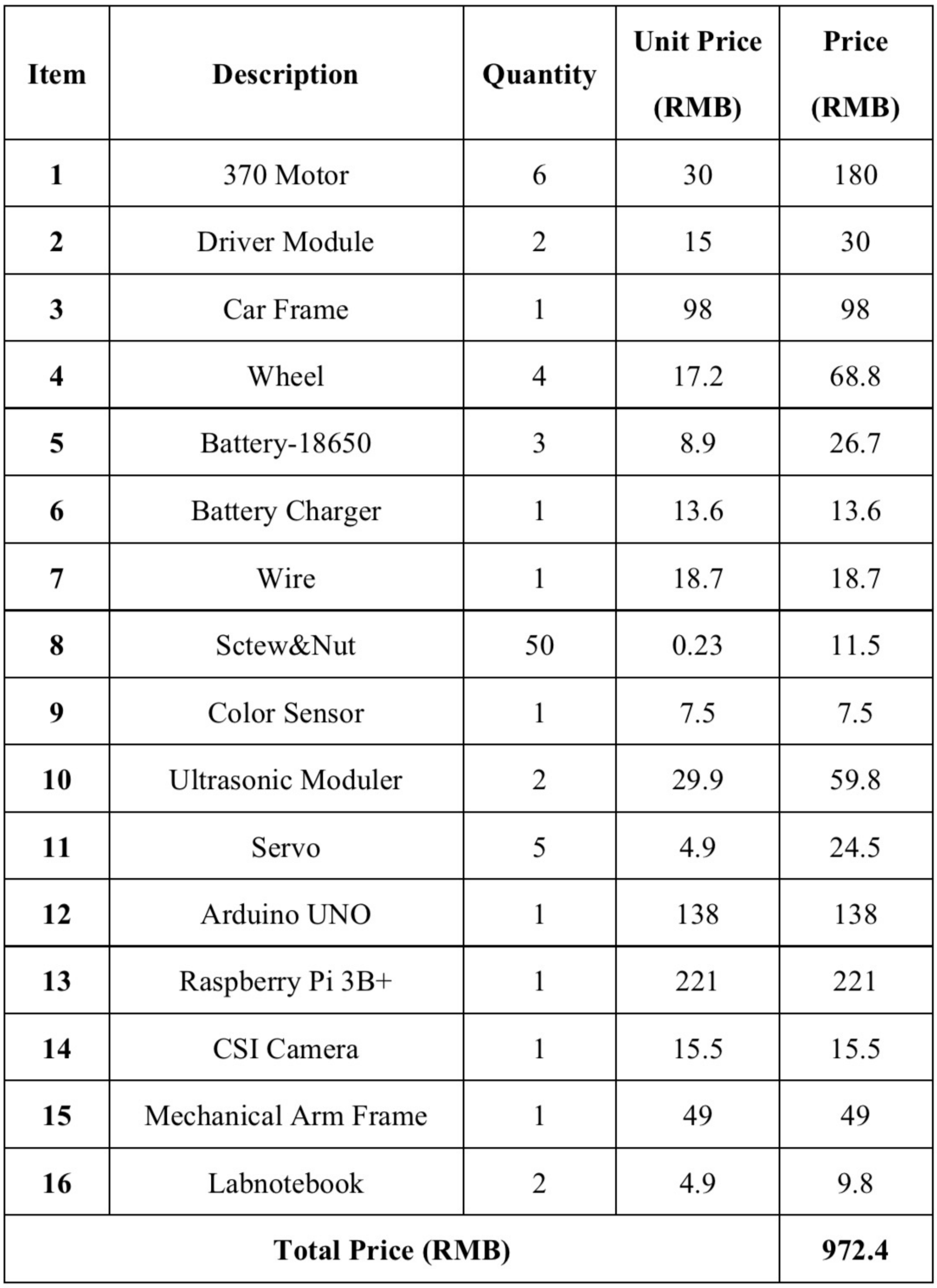
[19] Hung Y C , Chen G H . Communication system generator on layered communicating finite state machine[C]// IEEE International Conference on Communications. IEEE, 1992.

# A Appendix



| Student's Name | Student's UoG ID# | Contribution |
|---|---|---|
| **Liu Jinyuan** | **2289243L** | **1. As team leader, organize the group meeting, allocate the task, divide the team and design the Gantt Chart as well as lab notebook. Edit and revise the final group report format.**<br>**2. Responsible for power part including car driving and obstacle avoidance, participate in part of the line tracking. Take the car to do the field test and adjust the parameter. Design the state machine for both patios. Build the communication standard between Arduino and Raspberry Pi.** |
| **Ding Shuhan** | **2289236D** | **1. Responsible for visual part including line tracking and part of color recognition. Participate in actual field test.**<br>**2. Design the team logo and uniform, record and make the video. Revise the PPT for presentation.** |
| **Xu Yang** | **2289231X** | **1. Responsible for power part including obstacle avoidance and mechanical arm. Participate in actual field test.**<br>**2. Help build the overall structural of the car** |
| **Peng Zhendong** | **2289239P** | **1. Responsible for the communication between the computer and car.**<br>**2. Achieve the information transmission and time record.** |
| **Li Yueshan** | **2289242L** | **1. Participate in the car driving and obstacle avoidance.**<br>**2. Help build the overall structural of the car.** |
| **Xu Mengyin** | **2289237X** | **1. Responsible for visual part containing color recognition.**<br>**2. Help build the mechanical arm.** |
| **Yu Xi** | **2289240Y** | **1. Responsible for power part including mechanical arm. Participate in actual field test.** |
| **Gao Haihuan** | **2289238G** | **1. Purchase some materials and help other teammates.** |

Figure 31: Contribution of team members



| Item | Description | Quantity | Unit Price (RMB) | Price (RMB) |
|---|---|---|---|---|
| **1** | 370 Motor | 6 | 30 | 180 |
| **2** | Driver Module | 2 | 15 | 30 |
| **3** | Car Frame | 1 | 98 | 98 |
| **4** | Wheel | 4 | 17.2 | 68.8 |
| **5** | Battery-18650 | 3 | 8.9 | 26.7 |
| **6** | Battery Charger | 1 | 13.6 | 13.6 |
| **7** | Wire | 1 | 18.7 | 18.7 |
| **8** | Sctew&Nut | 50 | 0.23 | 11.5 |
| **9** | Color Sensor | 1 | 7.5 | 7.5 |
| **10** | Ultrasonic Moduler | 2 | 29.9 | 59.8 |
| **11** | Servo | 5 | 4.9 | 24.5 |
| **12** | Arduino UNO | 1 | 138 | 138 |
| **13** | Raspberry Pi 3B+ | 1 | 221 | 221 |
| **14** | CSI Camera | 1 | 15.5 | 15.5 |
| **15** | Mechanical Arm Frame | 1 | 49 | 49 |
| **16** | Labnotebook | 2 | 4.9 | 9.8 |
| **Total Price (RMB)** | | | | **972.4** |

Figure 32: Financial Statement

```
## Line_tracking Code ##
# capture_image_HSV_segmented_dilate_Hough_line
# Final vision add output wiringPi and close the trackbar windows

import cv2
import numpy as np
import wiringpi
import time
import io
from picamera import PiCamera

stream = io.BytesIO()
camera = PiCamera()
#camera.start_preview()
camera.resolution=(160,120)
time.sleep(2)

def photoCap():
    camera.capture(stream,'jpeg')
    data = np.fromstring(stream.getvalue(),dtype=np.uint8)
    image = cv2.imdecode(data,1)
    image = image[:,:,::-1]
    stream.seek(0)
    stream.truncate(0)
    image = np.array(image,dtype = np.uint8)
    return image

# set output pins
wiringpi.wiringPiSetup()
wiringpi.pinMode (23,1)
wiringpi.pinMode (24,1)
wiringpi.pinMode (25,1)
wiringpi.digitalWrite(23, 0)
wiringpi.digitalWrite(24, 0)
wiringpi.digitalWrite(25, 0)

while(1):

    frame = photoCap()
    print(frame.shape)
    cv2.imshow("capture",frame) # capture image

    hsv=cv2.cvtColor(frame,cv2.COLOR_RGB2HSV)
    lower_hsv=np.array([15,50,150])
    upper_hsv=np.array([255,255,255])
    mask=cv2.inRange(hsv,lower_hsv,upper_hsv)
    cv2.imshow("hsv",mask) # HSV segmented image
```

```
# open and close operation
element=cv2.getStructuringElement(cv2.MORPH_RECT,(15,15))
threshold=cv2.morphologyEx(mask,cv2.MORPH_CLOSE,element)
threshold=cv2.morphologyEx(threshold,cv2.MORPH_OPEN,element)
canny=cv2.Canny(threshold,0,50,3)
gaussian=cv2.GaussianBlur(canny,(5,5),0)
cv2.imshow('edges',gaussian)

# Hough line detection
lines=cv2.HoughLines(gaussian,1,np.pi/180,80)
n=0
avgAngle=0
if lines == None :
    avgAngle = 888
else :
    for line in lines:
        rho,theta=line[0]
        a=np.cos(theta)
        b=np.sin(theta)
        x0 = a*rho
        y0 = b*rho
        x1 = int(x0 + 1000*(-b))
        y1 = int(y0 + 1000*(a))
        x2 = int(x0 - 1000*(-b))
        y2 = int(y0 - 1000*(a))
        cv2.line(frame, (x1,y1),(x2,y2),(0,0,255), 2)
        Angle=theta*57.3
        if Angle<90 :
            Angle=Angle
        elif Angle>90 :
            Angle=-180+Angle
        else :
            n=n+1
            Angle=0
        avgAngle = avgAngle+Angle
    avgAngle = avgAngle/(len(lines)-n)
print("avgAngle", avgAngle);
cv2.imshow("houghlines", frame)

# output 0 1 to communication with Arduino
if avgAngle>15 and avgAngle<=30 :
    wiringpi.digitalWrite(23, 1)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.delay(500)
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(25, 0)
```

```
elif avgAngle>30 and avgAngle<=60 :
    wiringpi.digitalWrite(23, 1)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(24, 1)
    wiringpi.delay(500)
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(25, 0)

elif avgAngle>60 and avgAngle<=90 :
    wiringpi.digitalWrite(23, 1)
    wiringpi.digitalWrite(24, 1)
    wiringpi.digitalWrite(25, 0)
    wiringpi.delay(500)

elif avgAngle>-15 and avgAngle<=15 :
    wiringpi.digitalWrite(23, 1)
    wiringpi.digitalWrite(24, 1)
    wiringpi.digitalWrite(25, 1)
    wiringpi.delay(500)
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(25, 0)

elif avgAngle>-30 and avgAngle<=-15 :
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 1)
    wiringpi.digitalWrite(25, 1)
    wiringpi.delay(500)
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(25, 0)

elif avgAngle>-60 and avgAngle<=-30 :
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 1)
    wiringpi.digitalWrite(25, 0)
    wiringpi.delay(500)
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(25, 0)

elif avgAngle>-90 and avgAngle<=-60 :
    wiringpi.digitalWrite(23, 0)
    wiringpi.digitalWrite(24, 0)
    wiringpi.digitalWrite(25, 1)
    wiringpi.delay(500)
```

```
        wiringpi.digitalWrite(23, 0)
        wiringpi.digitalWrite(24, 0)
        wiringpi.digitalWrite(25, 0)

    else :
        wiringpi.digitalWrite(23, 0)
        wiringpi.digitalWrite(24, 0)
        wiringpi.digitalWrite(25, 0)
        wiringpi.delay(500)
        wiringpi.digitalWrite(23, 0)
        wiringpi.digitalWrite(24, 0)
        wiringpi.digitalWrite(25, 0)

    if cv2.waitKey(1)&0xFF==ord('q'):
        break

    # time.sleep(0.5)

cap.release()
cv2.destroyAllWindows()



## Clor_recognition Code ##
# capture_image_HSV_calculate_center

import cv2
import numpy as np
import wiringpi
import time
import io
from picamera import PiCamera
import matplotlib.pyplot as plt

stream = io.BytesIO()
camera = PiCamera()
#camera.start_preview()
camera.resolution=(160,120)
time.sleep(2)

def photoCap():
    camera.capture(stream,'jpeg')
    data = np.fromstring(stream.getvalue(),dtype=np.uint8)
    image = cv2.imdecode(data,1)
    image = image[:,:,::-1]
    stream.seek(0)
    stream.truncate(0)
    image = np.array(image,dtype = np.uint8)
    return image

while(1):

    img = photoCap()
    print(frame.shape)
    cv2.imshow("capture",img) # capture image

    cv.namedWindow("Color_conversion",cv.WINDOW_NORMAL)
    cv.imshow('Color_conversion',img)
    cv.waitKey(5000)
    cv.destroyAllWindows()

    # Color_conversion
    img2 = cv.cvtColor(img,cv.COLOR_BGR2HSV)
    img3 = cv.cvtColor(img,cv.COLOR_BGR2GRAY)


    for i in range(1):
        plt.subplot(221), plt.imshow(img2)
        plt.title('HSV␣Image'), plt.xticks([]),plt.yticks([])
        plt.subplot(223), plt.imshow(img3)
        plt.title('GRAY␣Image'), plt.xticks([]),plt.yticks([])
        plt.show()

    # find color square
    lower_purple_hsv = np.array([125,43,46])
    upper_purple_hsv = np.array([155,255,255])
    mask1 = cv.inRange(img2, lower_purple_hsv, upper_purple_hsv)
    img =  cv.cvtColor(img,cv.COLOR_BGR2RGB)
    cv.namedWindow("inRange",cv.WINDOW_NORMAL)
    cv.imshow('inRange',mask1)
    cv.imwrite('Find_purple.jpg',mask1)
    cv.waitKey(5000)
    cv.destroyAllWindows()

    lower_blue_hsv = np.array([100,43,46])
    upper_blue_hsv = np.array([124,255,255])
    mask2 = cv.inRange(img2, lower_blue_hsv, upper_blue_hsv)
    img =  cv.cvtColor(img,cv.COLOR_BGR2RGB)
    cv.namedWindow("inRange",cv.WINDOW_NORMAL)
    cv.imshow('inRange',mask2)
    cv.imwrite('Find_blue.jpg',mask2)
    cv.waitKey(5000)
    cv.destroyAllWindows()

    lower_hsv = np.array([26,43,46])
    upper_hsv = np.array([34,255,255])
    mask3 = cv.inRange(img2, lower_hsv, upper_hsv)
    img =  cv.cvtColor(img,cv.COLOR_BGR2RGB)
    cv.namedWindow("inRange",cv.WINDOW_NORMAL)
    cv.imshow('inRange',mask3)
    cv.imwrite('Find_blue.jpg',mask3)
    cv.waitKey(5000)
    cv.destroyAllWindows()

    plt.subplot(221), plt.imshow(img)
    plt.title('origin␣Image'), plt.xticks([]),plt.yticks([])
    plt.subplot(222), plt.imshow(mask1)
    plt.title('Find_purple␣Image'),plt.xticks([]),plt.yticks([])
    plt.subplot(223), plt.imshow(mask2)
    plt.title('Find_blue␣Image'), plt.xticks([]),plt.yticks([])
    plt.subplot(224), plt.imshow(mask3)
    plt.title('Find_yellow␣Image'),plt.xticks([]),plt.yticks([])
    plt.show()
```

```
//Arduino codes for Raperberry pi and sensor controls the car
//Both for patio 1 and patio 2

int  Right_motor_go= 4;
int  Right_motor_back= 2;

int  Left_motor_go= 7;
int  Left_motor_back= 8;

int Left_motor_pwm= 6;
int Right_motor_pwm= 5;


int EchoPin = 12;
int TrigPin = 13;


int mei1=A2; //SI(blue)right
int mei2=A3; //CLK(red)middle
int mei3=A1; //A0(white)left

//ground green;
```

```
float distance = 0;


void setup()
{

  Serial.begin(9600);


  pinMode(Left_motor_go, OUTPUT);
  pinMode(Left_motor_back, OUTPUT);
  pinMode(Right_motor_go, OUTPUT);
  pinMode(Right_motor_back, OUTPUT);



  pinMode(EchoPin, INPUT);
  pinMode(TrigPin, OUTPUT);

  pinMode(mei1,INPUT);
  pinMode(mei2,INPUT);
  pinMode(mei3,INPUT);

}


void Run(int left_speed, int right_speed)
{

  digitalWrite(Left_motor_go, HIGH);
  digitalWrite(Left_motor_back, LOW);
  analogWrite(Left_motor_pwm, left_speed);


  digitalWrite(Right_motor_go, HIGH);
  digitalWrite(Right_motor_back, LOW);
  analogWrite(Right_motor_pwm, right_speed);
}


void left(int left_speed, int right_speed)
{

  digitalWrite(Left_motor_go, LOW);
  digitalWrite(Left_motor_back, HIGH);
  analogWrite(Left_motor_pwm,left_speed);
  digitalWrite(Right_motor_go, HIGH);
  digitalWrite(Right_motor_back, LOW);
  analogWrite(Right_motor_pwm, right_speed);
}

void right(int left_speed, int right_speed)
{

  digitalWrite(Left_motor_go, HIGH);
  digitalWrite(Left_motor_back, LOW);
  analogWrite(Left_motor_pwm,left_speed);


  digitalWrite(Right_motor_go, LOW);
  digitalWrite(Right_motor_back, HIGH);
  analogWrite(Right_motor_pwm, right_speed);
}


void Stop(int time)
{
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back, LOW);
  digitalWrite(Right_motor_go, LOW);
  digitalWrite(Right_motor_back, LOW);
  delay(time*100);
}

void Distancee()
{
  digitalWrite(TrigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin, LOW);
  float Fdistance = pulseIn(EchoPin, HIGH);
  Fdistance = Fdistance / 58;
  Serial.print("Distance:");
  Serial.print(Fdistance);
  Serial.println("cm");
  distance = Fdistance;
}

void spin_right(int Left_speed,int Right_speed)
{

  digitalWrite(Left_motor_go, HIGH);
  digitalWrite(Left_motor_back, LOW);
  analogWrite(Left_motor_pwm,Left_speed);


  digitalWrite(Right_motor_go, LOW);
  digitalWrite(Right_motor_back, HIGH);
  analogWrite(Right_motor_pwm, Right_speed);
}





void loop()
{





  int value1;
  int value2;
  int value3;
  value1=analogRead(mei1);
  value2=analogRead(mei2);
  value3=analogRead(mei3);

  Distancee();
  if (distance > 60)
  {
 if(value1<650&&value2<650&&value3<650)
      {
       Stop(10);
      }
  else if (value1>650&&value2>650&&value3>650)
    {
      Run(50,50);

    }
    else if (value1>650&&value2<650&&value3<650)
    {
      left(200,200);

      }
    else if (value1<650&&value2>650&&value3<650)
    {
      left(250,250);

      }
    else if (value1>50&&value2>650&&value3<650)
    {
      left(0,100);

      }
   else if (value1<650&&value2<650&&value3>650)
    {
      right(200,200);

      }
   else if (value1>650&&value2<650&&value3>650)
    {
      right(250,250);

      }
     else if (value1<650&&value2>650&&value3>650)
    {
      right(100,0);
      }
  }
  else if (distance < 60)
  {
    spin_right(50,50);
    delay(1000);
    Run(100,100);
    delay(1000);
    Stop(10);
  }

  }
```

```
//Communication between Arduino and computer
#include <stdio.h>
#include <DS1302.h>
#include <SoftwareSerial.h>

namespace {
SoftwareSerial HC12(12, 13);        // connect HC-12 Tx& Rx
```

```
int signal= 1;                              //signal input
DS1302 rtc(5, 6, 7);// Chip Enable, Input/Output, Serial Clock

String dayAsString(const Time::Day day) {
  switch (day) {
    case Time::kSunday: return "Sunday";
    case Time::kMonday: return "Monday";
    case Time::kTuesday: return "Tuesday";
    case Time::kWednesday: return "Wednesday";
    case Time::kThursday: return "Thursday";
    case Time::kFriday: return "Friday";
    case Time::kSaturday: return "Saturday";
  }
  return "(unknown␣day)";
}

void printTime() {
  Time t = rtc.time();
  const String day = dayAsString(t.day);
  char buf[50];
  snprintf(buf, sizeof(buf), "%s␣%04d-%02d-%02d␣%02d:%02d:%02d\n",
           day.c_str(),
           t.yr, t.mon, t.date,
           t.hr, t.min, t.sec);
  HC12.print(buf);
}
}

void setup() {
  HC12.begin(9600);

//initialize time
  Time t(2019, 5, 21, 17, 58, 50, Time::kTuesday);
  rtc.time(t);
}

// transmit student information and time if signal is received.
void loop() {
   if (signal == HIGH) {
    HC12.print(
"start␣\r\n"
  );
    printTime();
    delay(630000);
    printTime();
     HC12.print(
               "\r\n␣Team␣Name:␣FFAASSTT␣\r\n"
               "2016200301034␣Xu␣Yang␣␣2289231X␣\r\n"
```

```
                        "2016200302005␣Ding␣Shuhan␣␣2289236D␣\r\n"
                        "2016200102006␣Xu␣Mengyin␣␣2289231X␣\r\n"
                        "2016200102007␣Gao␣Haihuan␣␣2289238G␣\r\n"
                        "2016200102008␣Peng␣Zhendong␣␣2289239P␣\r\n"
                        "2016200102009␣Yu␣Xi␣␣2289240Y␣\r\n"
                        "2016200103011␣Li␣Xueshan␣␣2289242L␣\r\n"
                        "2016200103012␣Liu␣Jinyuan␣␣2289243L\r\n"
  );

    printTime();
    signal = LOW;
  }




}
```