THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT
OF
ELECTRONIC AND INFORMATION ENGINEERING

EIE3105 Integrated Project
Final Report

GUAN Zhong, 14110265d

March 31, 2019

1 Introduction

The robot car used in the EIE3105 course is a DC-motor-driven two-wheel robot car with a mobile power bank as power source, which is designed for handling multiple missions including following specific tracks and locating objects through the WIFI communication.

With the given Main Board, Floor Board and LED Board, I have to integrate them together and program on the *STM32F103RB Cortex-M3* Microcontroller Unit (MCU). There are mainly three tasks need to be completed in this project by applying hardware and software programming skills. The descriptions of each task is as follows:

- **Track Follower:** A well-designed speed and direction control system will be implemented so that the robot car is able to follow the track automatically by using the information collected from the Floor Board Infra-Red (IR) photo transistors.
- **Hit Three Balls:** The robot car is required to hit three balls automatically to a green zone by locating the position of the balls and car through WIFI communication.
- **Pass Ball:** Two robot cars, placed in two different regions, can be able to pass a ball to each other. The robot car will consistently receive the same location information from a camera via WIFI communication. Theoretically, with good control of the speed and orientation of two robot cars, they can pass the ball forever until battery dies.

All the programming codes can be found on my GitHub.

2 Design Methodology

2.1 Track Follower

In order to let the robot car follows the track automatically, the robot car should be able to detect the track. By enabling the SPI communication, the robot car can perform two tasks depending on the information it gathers from the Infra-Red (IR) photo transistors: 1) deciding whether the car moves along the track, and 2)

adjusting the wheel speed to keep moving on the track. The algorithm logic of the implemented controller is shown in Figure 1.

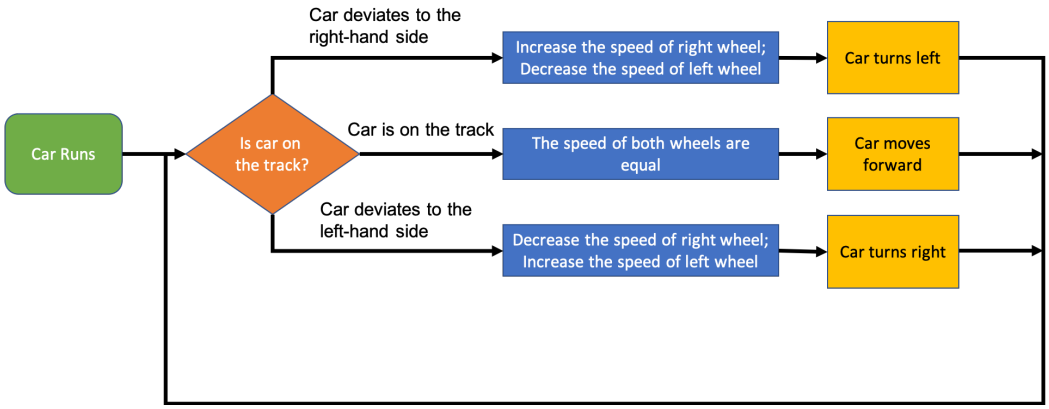

Figure 1: Algorithm of the track follower.

2.1.1 Deciding if the car is on the track

A SPI logical data array comprised of the readings from IR photo transistors are used to perform track detection. When the ground is white, the IR ray is absorbed and the photo transistors remain OFF state, which can generate logic signal "1". When the ground is black, the IR ray is reflected to the photo transistors and photo transistors go to ON state, which can generating logic signal “0”. Therefore, the relative position between the car and black trace can be decided by knowing which photo transistor generates “0”.

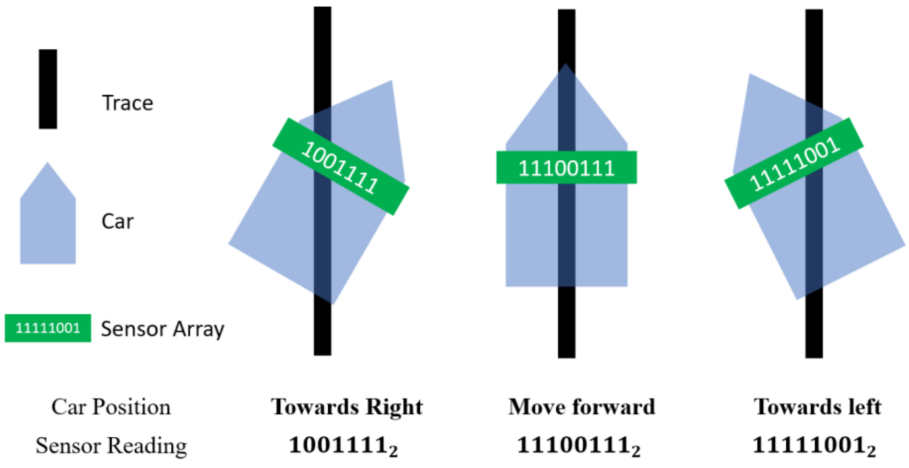

Figure 2: Relationship between car position and sensor array.

Using shift register 74HC299, the generated signals are shifted from the sensor array to the MCU through SPI serial port. The relationship between car position and sensor array reading is shown in Figure 2.

2.1.2 Adjusting the wheel speed

Once the reading from photo transistors can be acquired via SPI communication, the wheel speed will be adjusted accordingly by changing the PWM of the motor control signal. If the duty cycle of PWM pulse increase, the motor speed increases, and vice versa. Therefore, the robot car can change its moving direction when it has different speed for left and right motors. For this task, the PWM is controlled in the following manner:

- The MCU checks the logical data from the middle to the both sides, the car records the position of the first "0" signal i.e. the first-zero position.
- The error value for PID controller is computed as the following table. When the car direction is correct, the error value will be very small. Therefore, the error value reflects on how much the car deviates to either right or left side.

Position	1 (most left)	2	3	4	5	6	7	8 (most right)
Error	7	5	3	1	-1	-3	-5	-7

- The duty cycle is then calculated by using the following formula of typical PD controller:

$$PWM_{left} = ConstantPWM - P_{left} \times Error - D_{left} \times (Error - Error_{previous})$$
$$PWM_{right} = ConstantPWM + P_{right} \times Error + D_{right} \times (Error - Error_{previous})$$

If the car deviates to the right, the first or the second sensor will be "0", and the error will be a positive value. To return to the line, the left wheel speed should decrease while the right wheel speed should increase. Following the logic, the error is subtracted from the left-wheel PWM and the error is added to the right-wheel PWM. The constant PWM makes sure the car can move forward when there is no error. The D-term of the PD controller is used to suppress the oscillation of PD controller.

2.1.3 Changing the track

When the car passes the intersection of the inner and outer track, many sensors can detect the tracks, which can generate more than three "0" signals. If checking the sensor reading from right to left, the first zero comes from the inner track when the car passes the track intersection in clockwise direction. Hence, the car will change from the outer track to the inner track. When the car moves counterclockwise on the outer track, the first zero comes from the left, hence the car will track the outer loop if reading the sensr data from left to right. In a word, by changing the way of checking the first zero position, the car can easily change the track.

2.2 Hit Ball

In this task, I designed a two-stage system which is composed of “hit the ball” stage and “return to origin” stage. Firstly, the car will locate the position of the target ball and moves toward it. Secondly, the car will stop moving forward when the ball was pushed into the target region. Finally, the car will return back to the original point to prepare for the next hit, which would be the repetition of the above steps. The algorithm for this task is shown in Figure 3.

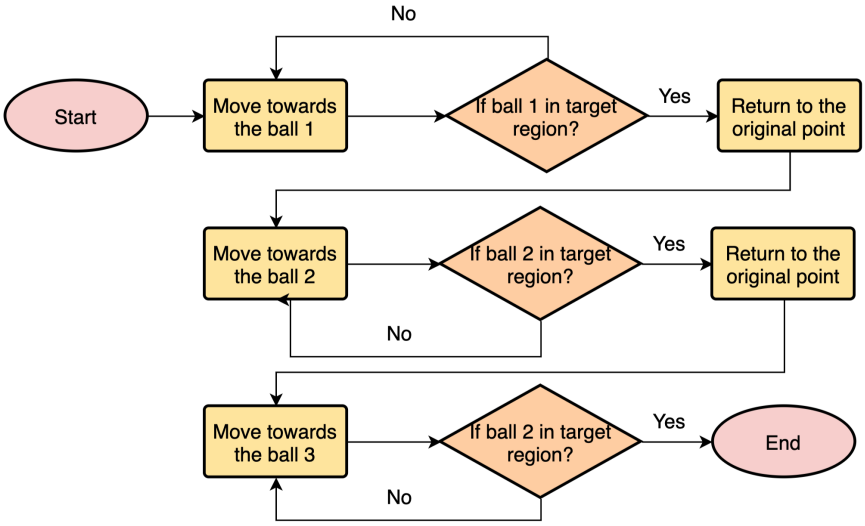

Figure 3: Algorithm for hitting three balls.

In order to make sure the robot car can move towards the target ball, the PID controller is implemented where the error is the angle difference between the desired orientation and actual orientation. The Figure 4 illustrates how to measure the orientation.

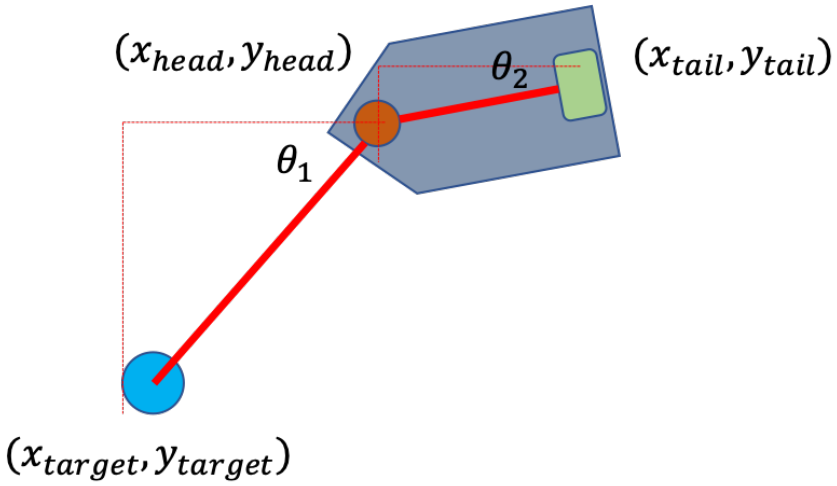

Figure 4: Angle error for hitting the ball.

The angles are calculated by using the following equations:

$$\theta_1 = \sin^{-1} \frac{y_{target} - y_{head}}{\sqrt{(x_{target} - x_{head})^2 + (y_{target} - y_{head})^2}}$$

$$\theta_2 = \sin^{-1} \frac{y_{head} - y_{tail}}{\sqrt{(x_{head} - x_{tail})^2 + (y_{head} - y_{tail})^2}}$$

The PID controller is implemented as follows:

$$Error = \theta_1 - \theta_2$$
$$Integral[n] = Integral[n-1] + Error$$
$$PWM_L = PWM - P_L \times Error - I_L \times Integral[n] - D_L \times (Error - Error_{previous})$$
$$PWM_R = PWM + P_R \times Error + I_R \times Integral[n] + D_R \times (Error - Error_{previous})$$

Therefore, with the use of PID controller, the robot car can keep moving towards the target position. When the car need to go back the original position after hit the ball, the target position is set to be the original point. With the use of same method, the car can also return back the original point.

2.3 Pass Ball

Same as the "hit ball", in order to complete the task of passing ball, the system has “hit the ball” stage and “return to origin” stage. The system starts with a conditional statement. The car won’t move unless the ball stops in the region charged by the

robot car. Then the car will follow several steps to hit the ball and go back to the origin. Finally, it checks if the ball has been successfully passed to the other side. If the ball has not moved to the other side, the car can hit the ball again when the ball stops moving until it arrived the other side. The whole system is a closed system so that the car can hit the ball no matter where it is. The detailed algorithm is shown in Figure 5.

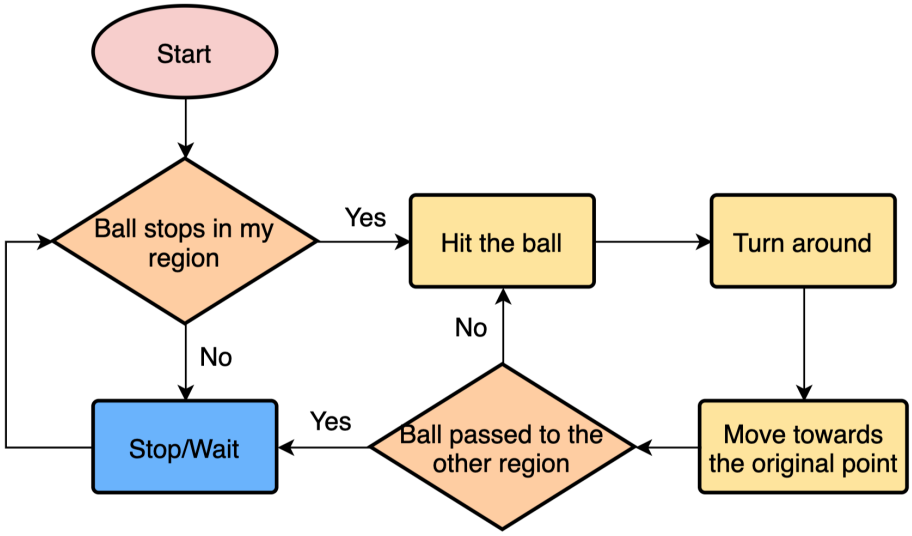

Figure 5: Algorithm for pass the ball

One of the most significant part is that the car has to wait for hitting until the ball stops moving. In order to check if the ball stops moving or not, the algorithm shown in Figure 6 is implemented.

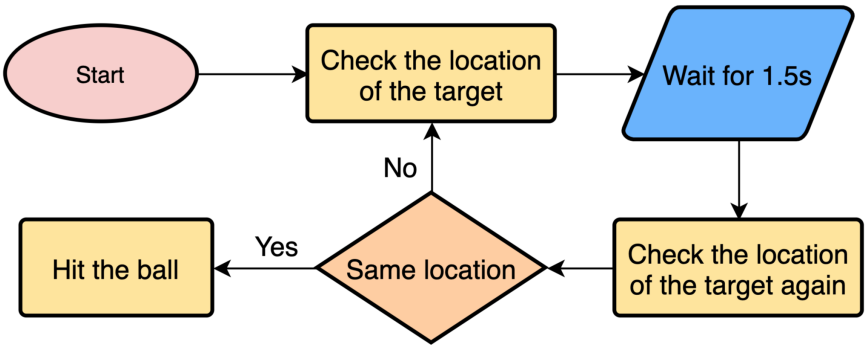

Figure 6: Algorithm for checking if ball stops moving

3 Verification Results

I verified my design in "Track Follower", "Hit Balls", and "Pass Ball" by using the same methodologies mentioned above.

- In "Track Follower", I achieved highly stable track following for both outer track and inner track and finished running two laps in 44 seconds. Video for Track Follower.
- In "Hit Balls", the car is initially placed at origin. It can locate and hit one ball and then return back to the origin. Then, it will hit another ball until all balls are hit by the car. I decreased the speed when car hit the ball while increased the speed when it moves back. Finally, the robot car finished the task in 20 seconds. Video for Hit Balls.
- In "Pass Ball", two cars are placed at their origins respectively. One of the cars firstly hit the ball to kick off the game. Then the other car will follow the flow chart in “Methodology” and switch from one state to another state continuously. Finally, the two robot cars can finish passing the ball for four times in 36 seconds, which proves that our system is fast and robust enough. Video for Pass Ball.

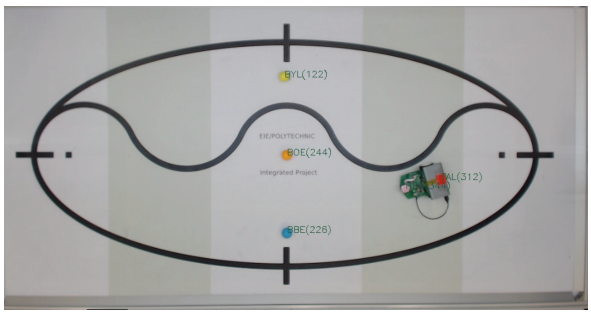

Figure 7: Hit Balls

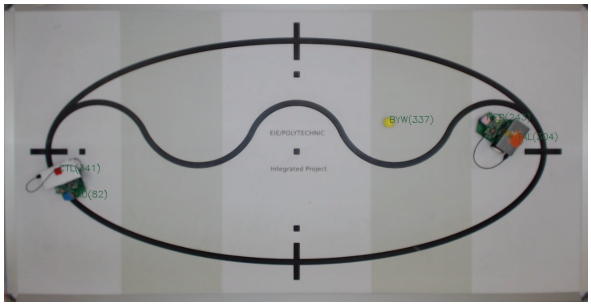

Figure 8: Pass Ball