EIE 3106 Integrated Project

Final report

Chan Wai Chi, 19060801d

2022

1 Introduction

In the EIE3106 course, a two-wheel robot car with Li-ion batteries as the power source is given to handle multiple automation missions including following a specific track. The robot car uses STM32F103C8T6 as Microcontroller Unit (MCU) and Bluetooth communication for remote control.

There are four tasks needed to complete in this course by applying programming skills. Descriptions of each task are mentioned as follows:

- **Remote control and measure the distance:** remote control the robot car to move by using a joypad. And measure the distance by using an ultrasonic sensor.
- **Line Tracking:** A good speed and direction system is implemented so that the robot car should follow the line automatically.
- **Car Parking:** The robot car is required to get close to the wall but not hit the wall from three different positions.
- **Relay Race:** Two robot cars are put in two different positions. While car 1 should follow the specific track and cross the two maps, car 2 will wait for car 1 to come. Car 2 will turn 180 degrees and finish the remaining track. In the meantime, car 1 will turn 180 degrees and wait for car 2 to come. Then, it turns 180 degrees and follows the line until it goes to point F.

Then, the following describe the design and difficulties of the robot car in each task.

2 Remote control and measure the distance

Using the HC05 and HC06 Bluetooth module, the joypad can send the command to the robot car. Then, the car can receive the command through USART2 and complete the command, such as move forward, move backward, turn left and turn right. Besides, with the SR04 ultrasonic sensor, the robot car can measure the distance by using the input capture function in timer 1.

3 Line Tracking

To achieve the goal of following the path automatically, the robot car should be able to detect the track. Thus, the MCU can use the information collected by the Infrared phototransistors to control the robot car. The MCU also use the timer to determine the position of the car.

3.1 Determine if the car is on the track

The module of infrared LED and phototransistors is used to perform track detection. When the ground is white, the IR ray is reflected to phototransistors and the signal '0' is generated. When the ground is black, the IR ray is absorbed and the signal '1' is generated. Therefore, the ground information can be collected precisely by this module. And using shift register 74HC299, the 8-bit data generated by the module is shifted to the MCU via SPI serial port.

3.2 Adjusting the wheel speed

After the reading from phototransistors is acquired through SPI communication, the wheel speed should be adjusted by modifying the PWM of the motor control signal. If the duty cycle of the PWM pulse increase, the motor speed increases, and vice versa. Hence, the robot car can control its moving direction by adjusting the difference between left and right motors' s duty cycle. For this task, the duty cycle is controlled by the following strategy.

- The MCU will check the data from the module and get the error value. The error value for PD controller is computed as the following table. This table reflects the derivation between the current location and target location, i.e., if the moving direction is correct, the error value should be smaller. Due to the reason of the track is wide enough for three phototransistors to detect, the position value will use combination to present.

Position Value (Hex)	80 (Most left)	C0	E0	70	38	1C	0E	07	03	01 (Most right)	FF
Error	-12	-9	-6	-3	-1	0	3	6	9	12	-11

  - 0xFF is for point A and point C, because error value could be '0' if I do not add this if-conditional statement. It can optimize the performance in task 2 effectively.

- Then, the duty cycle is calculated by using the formula.

$$\mathrm{PWM}_{\mathrm{left}} = \text{ basic speed}_{\mathrm{left}} + kp \ \times error + kd \times (error - last\ error)$$

$$\mathrm{PWM}_{\mathrm{right}} = \text{ basic speed}_{\mathrm{right}} - kp \ \times error - kd \times (error - last\ error)$$

If the car deviates to left, the sensor will return a value of '01' and the corresponding error is '12'. To return the line, the speed of left wheel could be larger while the right wheel's speed could be smaller. Thus, the formula can follow this logic, i.e., duty cycle of left wheel is larger, and vice versa. The basic speed can confirm the car can forward if there is no error. The *kp*-term of PD controller decides the adjustment strength according to the error. The kd-term of PD controller can increase the response.

3.3 Changing the track

This is the main difficult that I encountered in this task. To solve this problem, the car can count the time that it passes the point A and C. Then, it can use timer to count when it goes to point B. And it moves anticlockwise for short time and then change from outer track to inner track.

4 Car Parking

In this task, with the given SR04 ultrasonic module, the MCU can get the distance between the robot car and the wall. Then, it can move forward at different speed according to the distance. Thus, it can control the car to get close to the wall but not hit the wall. The algorithm for this task is shown in Figure 1.

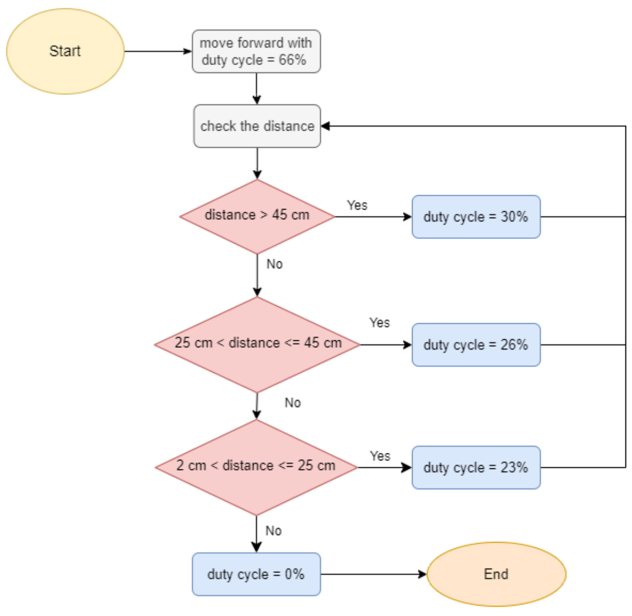

Figure. 1: The algorithm for car parking

5 Relay Race

In this task, I combined the method of "Line Tracking" and "Car parking".

For car 1, it follows the track at the beginning until the counter reaches a specific value. The car can cross the map and continue tracking the path for a distance. Then, it waits for car 2 to exit by using the SR04 module to detect. After car 2 exits, car 1 can move and turn to wait for car 2 to come. When it detects car 2 coming, it can turn and finish the track.

For car 2, it waits for car 1 to come and turn anticlockwise for specific wheel counter value. And it follows the line as same as car 1.

5.1 Difficulty

I found that there are some uncontrollable factors when doing this task. For example, motor damage and battery power. To solve them, I try to use PID controller to control the car speed. This is the algorithm for PID controller shown in Figure 2. This effectively improves the performance and stability of the robot car, compare with “Line tracking”. Its speed is not easily affected by battery life.

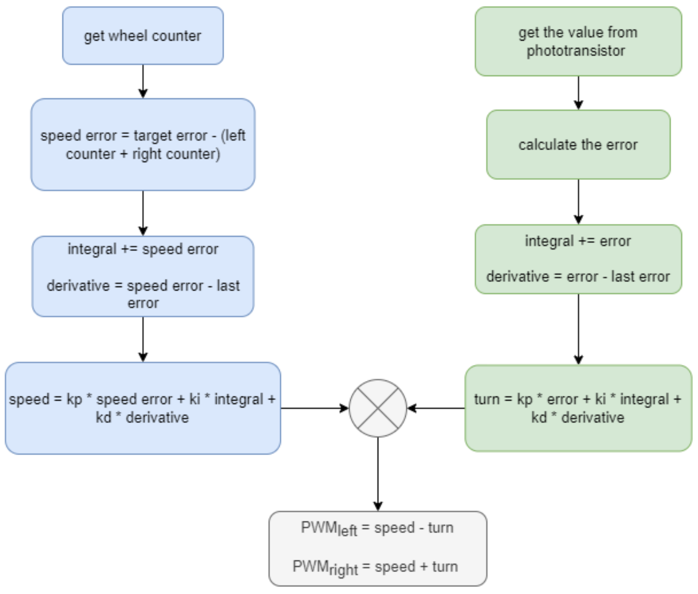
get wheel counter
speed error = target error - (left counter + right counter)
integral += speed error
derivative = speed error - last error
speed = kp * speed error + ki * integral + kd * derivative
get the value from phototransistor
calculate the error
integral += error
derivative = error - last error
turn = kp * error + ki * integral + kd * derivative
PWMleft = speed - turn
PWMright = speed + turn

Figure 2: algorithm for PID controller

Unlike “Line Tracking”, the robot car mainly uses the counter to determine which position it is at. Because I found that using timer is not enough particular to determine the position. This can also reduce the influence of the map more effectively.