Report for Embedded System Integrated Project

- *describe your work in Demonstrations 1, 2, 3, and 4.*
- *show your design in all demonstrations and tell the difficulties you have and how to overcome them.*

Demonstration 1: Robot Car Assembly

Task of demonstration 1 is to assemble the robot car according to the car assembly guide, and then program the robot car (STM32F103RBT6) and the remote controller (Arduino Uno), to let the robot car act according to the commands sent by the remote controller.

The Robot Car uses two PWM Signals generated from timer to control the speed of the two motors, and two GPIO Signals to control the direction of the two motors. The robot car communicates with the remote controller through Bluetooth connection. Characters will be sent upon button pressing on the remote controller, and parsing will be triggered through USART2 RX interrupt each time characters are received by the Robot Car.

Demonstration 2: Line Tracking

The task of the demonstration 2 is to control the robot car to track the black line on a given arena (Figure 1), through designated route: A > B > C > D > A > X > C > D > A > B > C > X > A. The overall control mechanism involves two levels: line tracking mechanism and route controlling mechanism.

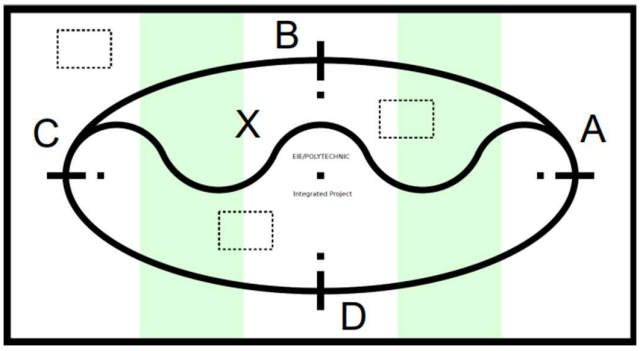

Figure 1. Designated Arena for Task Achievement

Line Tracking Mechanism

In the real experiment, the motors of the car are found very inaccurate and unstable to control. Specifically, the speed of the wheel can exhibit huge fluctuation even if the same PWM level is applied. When the power (or voltage) of the battery drops, the speed of the wheel also changes significantly. In addition, it is suspected that the gear of the motor is weak in durability as the gear sometimes cannot occlude well and just run in idle.

In this connection, plus the fact that the line tracker only has five lines, I gave up the idea of PID control in the later development. Instead, a system initially implemented by Yan is employed, which conducts differential adjustments when the car is deflected from the line, and full speed forward when the car is perfectly tracking the line. `adjust_l_r` sets the priority of the direction adjustment, which is intended for the car to continue move forward in absence of the line and enables the route selection. Detailed control flowchart is explained in Figure 2. Under the condition that the motors with plastic gears are troublesome to control, this system have significantly reduced the chances that the car runs off track when turning.

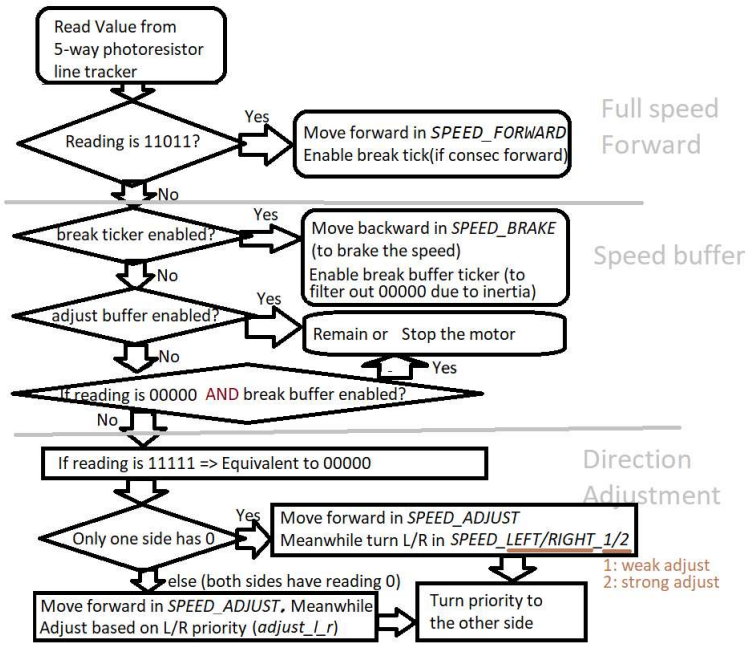

Figure 2. Control flowchart for line tracking.

However, there are still several limitations for this scheme. Firstly, the car still have trouble in braking after a long distance of driving forward. Secondly, the car is still volatile to off track in crossings, as the car often reaches the crossings not in a just position. Thirdly, as the motor is inaccurate in controlling, position adjustments can easily go overshoot, which makes the car wriggles around the line. These limitations are believed to be resulted from the terrible physical properties of the motor, as in the Demonstration 4 after changing to metal gears, the stability of the car increases significantly.

Route Controlling Mechanism

The route controlling mechanism is built upon the line tracking mechanism. It only controls the forwarding speed and conducts route selection by manipulating variable `adjust_l_r` (Figure 3). The only decision input is the distance travelled of the car, which is achieved by reading the counter value of the timer that takes the wheel encoder as clock input.

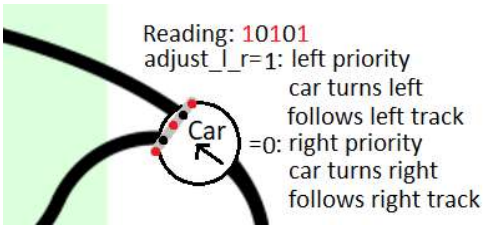

Figure 3. Illustration on route selection.

Demonstration 3: Car Parking

The task of demonstration 3 is to program the car to park in front of the barrier from 3 designated locations. The car needs to get close to the barrier in the shortest time possible but cannot hit the barrier. The Total time used for the car to finish the parking should be as short as possible.

To accomplish this task, an ultrasonic range finder is installed in the front of the car to detect the distance from the barrier. As the pulse of the range finder is so short that 1000Hz Timer 2 does not meet the resolution requirements, initially polling method was used for timing the high pulse.

However, probably due to the frequent peripheral activities (Timer 2 interrupt, USART 2), the result in this implementation is highly inaccurate. In this connection, I changed the right wheel to the timer 2 and cleared the timer 3 resource dedicated to the ultrasonic sensor, which make the reading much more accurate and stable. Based on the reading, the car will change its speed or brake using if-else control statement (Figure 4).

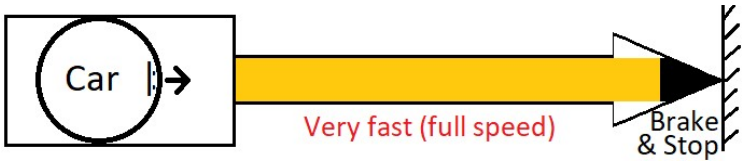

Figure 4. Control Scheme for Demo 3.

In the experiment, it is found that the car cannot cover the distance even if full speed is applied, and the car cannot follow a straight line due to deviation of the two motors. Therefore, PWM compensation have to be applied. However, no convenient solution is believed available to further speed up the car.

Demonstration 4: Relay Race

The task for demonstration 4 is to let two cars to track the line in relay through a specific route on a given arena, automatically without hitting each other. One car uses its ultrasonic sensor to detect whether the other car is coming. Additionally, Timer counting for control is not allowed.

This task combines the hardware technologies used in task 2 and 3, except that in route controlling mechanism, as using timer counting to get the distance is not allowed, the car have to rely purely on line patterns to determine when to turn and what route to select. Detailed route control scheme can be found in figure 5 and 6. In terms of hardware, since the motors used in demo 2 are terrible in stability, I purchased a pair of motors with full metal gear. The upgraded motor, although slow in speed, exhibits outstanding quality of stability when performing tasks on route 2 which is much more complicated than the route 1.

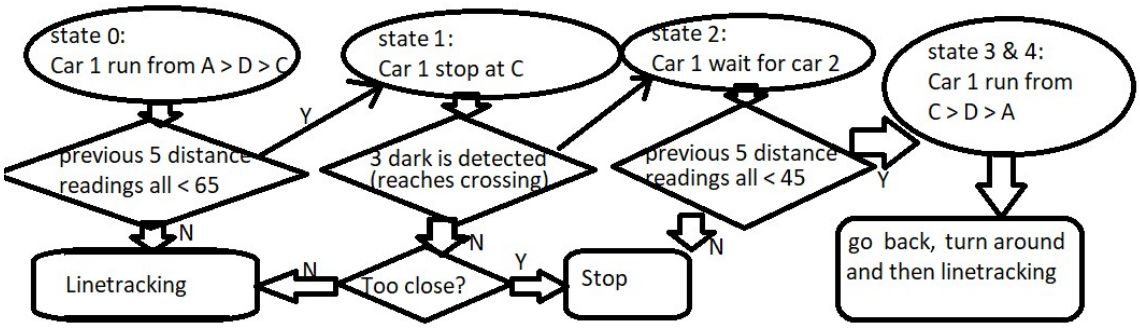

Figure 5. Route control flowchart for car route 1.

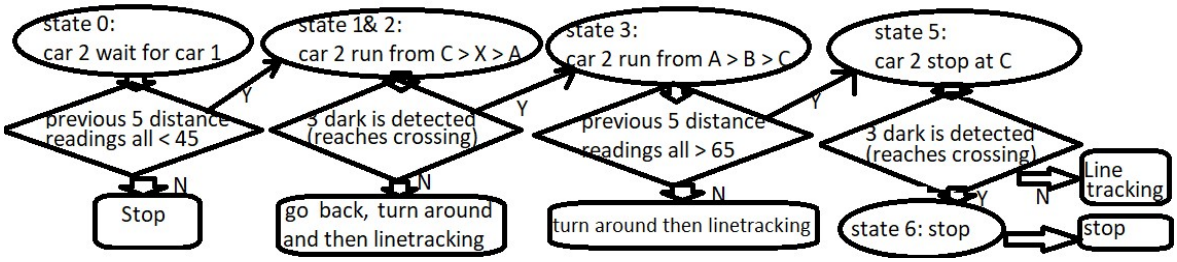

Figure 6. Route control flowchart for car route 2.

Note: This is an academic project. If you use (part of) the codes or documents in this repository for academic or commercial purpose, you will be required to **acknowledge** the corresponding components, otherwise it may be considered as **plagarism or IP theft**.

https://github.com/SS2867/STM32RobotCar