**Embedded Controller \_ Final Project Report**

21 황시환

21700242 문선빈

**Introduction**

In this project, our team required to build an RC Car with automatic parking system. With the most weight, design the algorithm to drive and park car automatically. After design the algorithm about the overall program, choose the number of sensors to use and direction to drive an RC car. The program proceeds to the outer yellow lane tracking by giving a start sign with the Bluetooth button. As you approach the wall at the end of Lane, park the car automatically. Our team focus on, optimized number of sensors, stabilized lane tracking, and accurate parking.

**Methods**

**Pin Maps**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sensor | Clock | GPIO- | Pin | Usage |
| Ultra Sonic |  | GPIOC | 8 | Trigger |
| TIM4 | GPIOB | 8 | Echo |
| IR Sensor | TIM5 | GPIOA | 0 |  |
| GPIOA | 1 |  |
| RC Motor |  | GPIOA | 6 |  |
| Bluetooth |  |  |  |  |
| DC motor |  | GPIOA | 8 | Left High |
|  | GPIOA | 9 | Left Low |
|  | GPIOA | 10 | Right High |
|  | GPIOA | 11 | Right Low |

Table 1. Pin Map

여기서 pin map이랑 clock이랑 다 설명하고 싶은데 ,, 쉽지 않음,,

**Sensors**

여기서 왜 센서 개수 설정 이유와 그래서 방향이 한 방향으로만 간다 이거 설명하려고 했음

Purpose of this project is using optimized number of sensors and using smallest sensor.

Two IR sensor for tacking yellow lane. Lane traking can be performed by adjusting the distance between the two sensors as appropriate. Therefore, only the IR sensor on both sides is necessary with the yellow lane in the middle and IR sensor traking the yellow lane in the middle is unnecessary. A total of two IR sensors were used.

When the end of the lane is reached, an Ultra sonic sensor must be attached to perform the parking algorithm. The purpose of using a small number of sensors is to use one Ultra sonic sensor, but the servo motor can be used to measure the distance of the front, side and rear of the vehicle. The parking mode was implemented according to the measured distance value, and the parking program was executed smoothly.

**Overall of the RC Car**

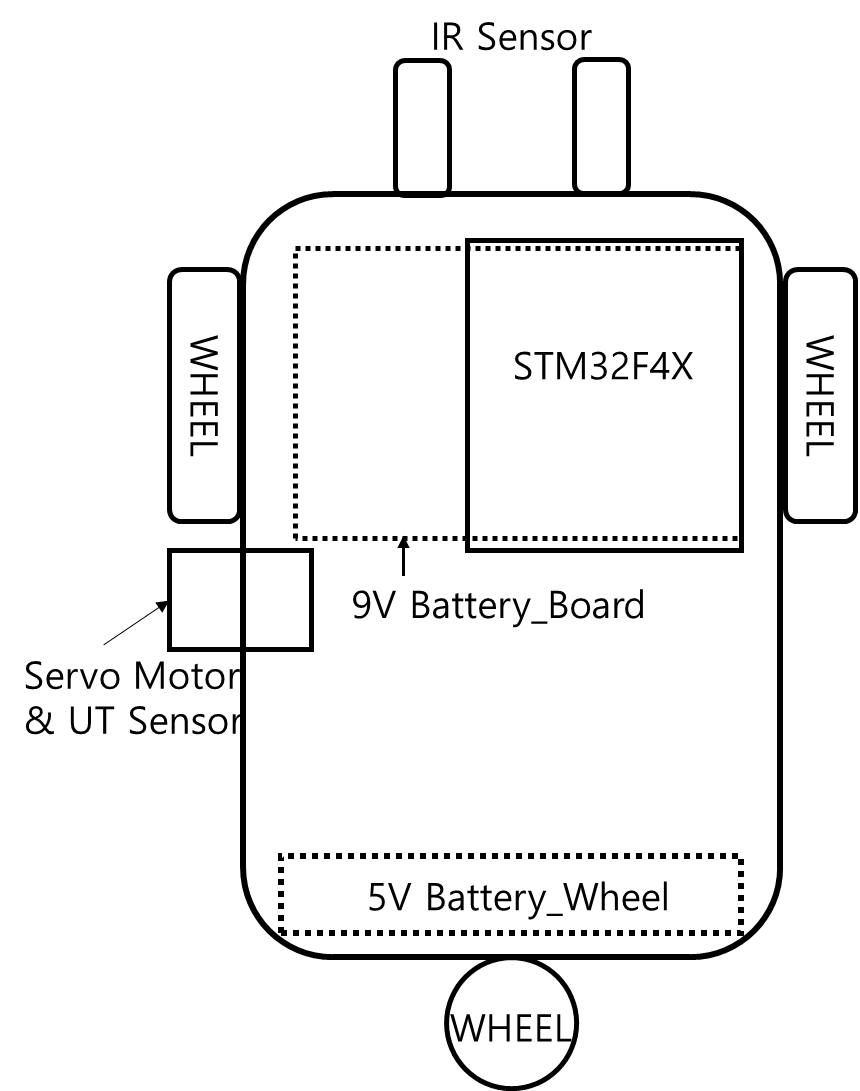


Figure 1. Topside of the RC Car

The figure above is a front view of the entire RC car seen from above. Two IR sensors were installed in the second closest position among the four cases that could be installed. A yellow lane passes between the two IR sensors. The board and board power 9V battery were installed on the front of the wheel. With all-wheel drive, the weight carried on the back of the RC car in order not to interfere with the momentum of the car. When the entire load is loaded on the front, the car moves forward with a strong propulsion. Originally, there was a problem with the 9V battery that drives the wheels, and instead of the 9V battery, the 5V battery was used. Low voltage and low power with 5V 1A (????).

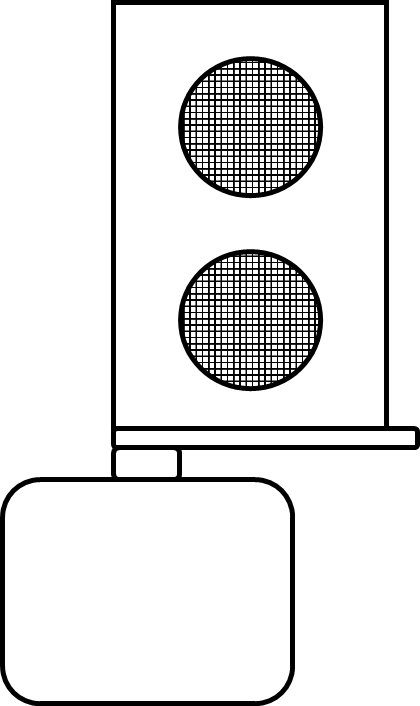


Figure 2. Servo Motor & UT Sensor

The servo motor and Ultra sonic are positioned in the middle and to the left of the body. Position the vehicle in the middle of the car body so that the front, side, and rear sides of the vehicle are detected fairly. However, since it is located on the left side and the parking logic is applied accordingly, there is a limit that the vehicle can go in only one direction. (Our team can only drive the sensors inward.) In addition, the Ultra sonic is mounted vertically on the servo motor, allowing precise detection of the edge of the parking area. In case of attaching horizontally, if the edge is located between trigger and echo of UT sensor, it is not detected. When attached vertically, the edge can be detected accurately in any case.

배터리 5V 설명

**Overview of the Algorithm**

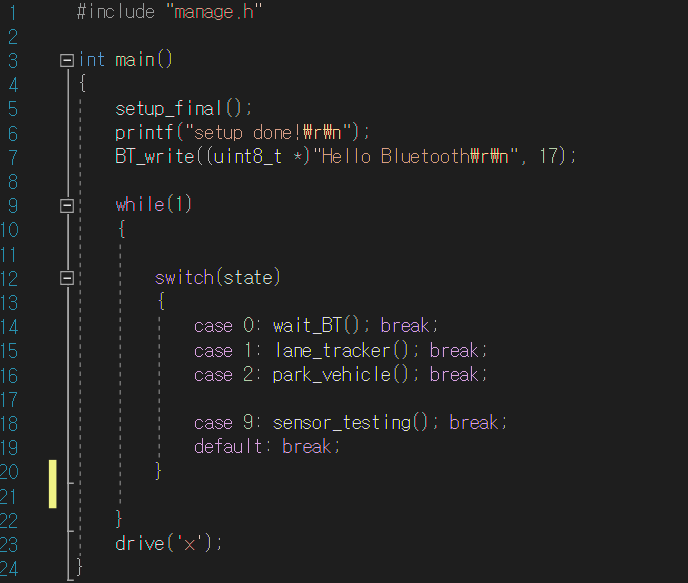


Figure 3. main.c

The overall program algorithm is shown in the figure above. The program is organized into three states. In the main statement, setup is completed with the setup\_final () function. In the setup\_final () function, we initialize each necessary clock and make it function by running the Bluetooth, RC motor, UT, DC motor, IR sensor, and TIM9\_init. Returning to the main statement, the default value is case 0, which waits in drive ('X') mode before giving a start sign with Bluetooth. After that, when the Bluetooth gives the start sign, the state changes to 1 and case 1: lane\_tracker () mode is executed. If you get close to 500 or less on the wall, the state will change to 2, and you will switch to case 2: park\_vehicle () mode. Each case is described in the following categories. In case 9, we made it for the engineer to make sure that the sensor works reliably while preparing the program.

**Lance Tracking Algorithm**

lane traking 설명, 값 설정 이유,

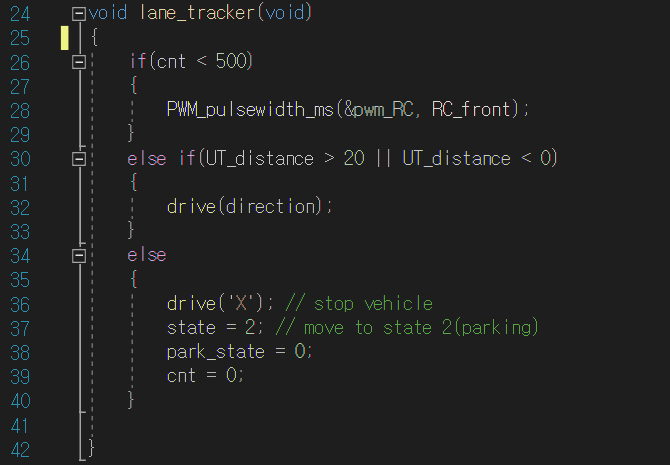


Figure 4. Lane\_tracker()

First, enter the lane tracking algorithm, case 1. Depending on the count calculated by TIM9, the UT sensor will continue to look forward. Afterwards, if the UT sonic is 20 or more and the garbage value is received in the middle, that is, the drive (direction ) Function. Then the else statement is when RC car is near 0 to 20 [cm] on the wall. Stop the car, change the state to 2, and move on to the park\_vehicle step. The park\_vehicle function proceeds according to park\_state. Therefore, the first step, park\_state, is initialized.

Count에 대한 설명 보충 필요

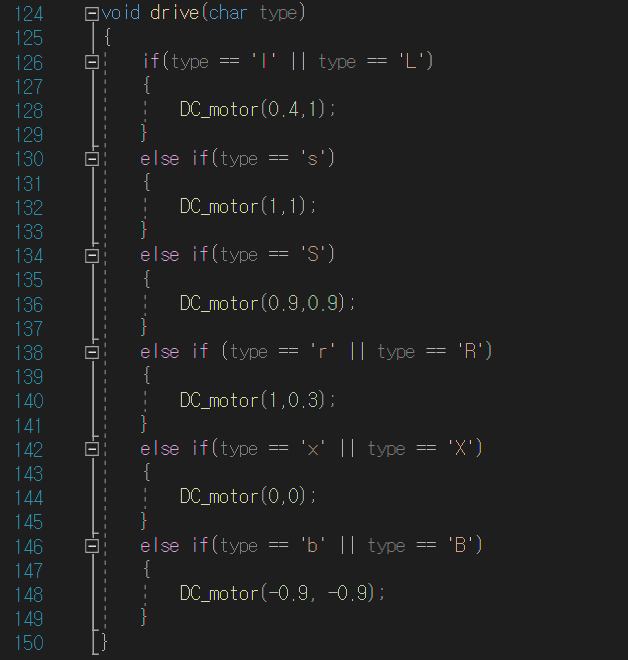


Figure 5. drive()

The drive function moves for each type according to the value commanded by the ADC handler. The difference between lowercase and uppercase letters is capitalized when driving continuously in the same direction, and lowercase letters are used when subtle tunning is required such as turning curves. Depending on the type of ADC handler, we put a value into the DC\_motor function and control the wheel accordingly.

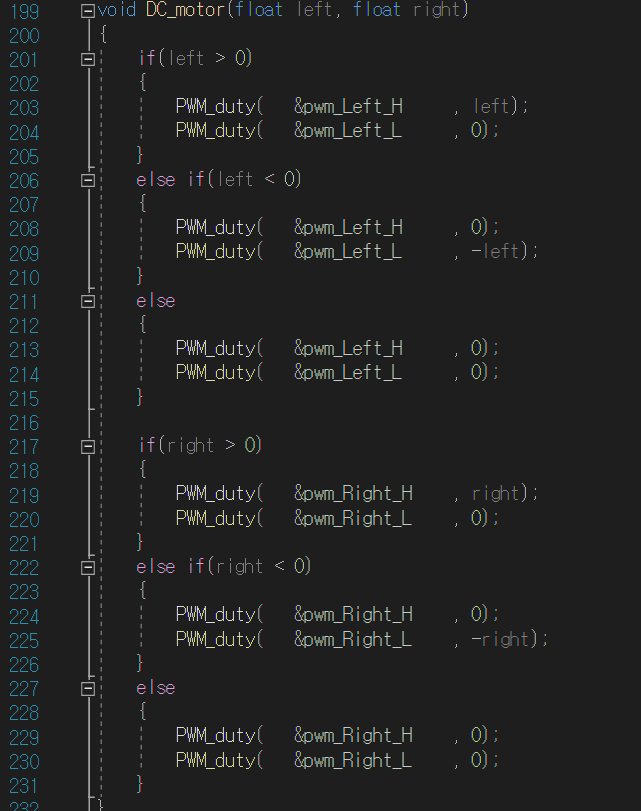


Figure 6. DC\_motor()

The DC\_motor function controls the wheel speed by adjusting the pwm duty given to each wheel. In main.c, we specified pwm to enter two motor drivers per wheel in setup\_final () and DC\_motor\_setup (). For example, & pwm\_Left\_H and & pwm\_Left\_L. The same is true for the right side. If the wheels drive forward, give high pwm a value and low pwm a zero value. If you want to reverse the wheel, give Low pwm a value and High pwm a zero value. By adjusting pwm in this way, you can control the wheels through the motor driver.

Again, the DC\_motor function gives a positive value when moving forward. For example, if the wheel turns to the left, the drive function gives DC\_motor (0.4,1). Since the value of the left wheel is positive, adjust the value to & pwm\_Left\_High to move forward, and the right wheel continues to give the maximum value of 1 pwm. For example, if backward, DC\_motor (-0.9, -0.9) is given. It is negative and gives 0 pack to pwm High and takes the absolute value to Low. The wheel goes backwards.

The values ​​given by the drive function, left, right, forward and backward values ​​are experimentally obtained. Since the power of the left wheel motor is weaker than that of the right foot motor, calibration was required to determine the right left direction. Accordingly, the value in the left right direction is different.

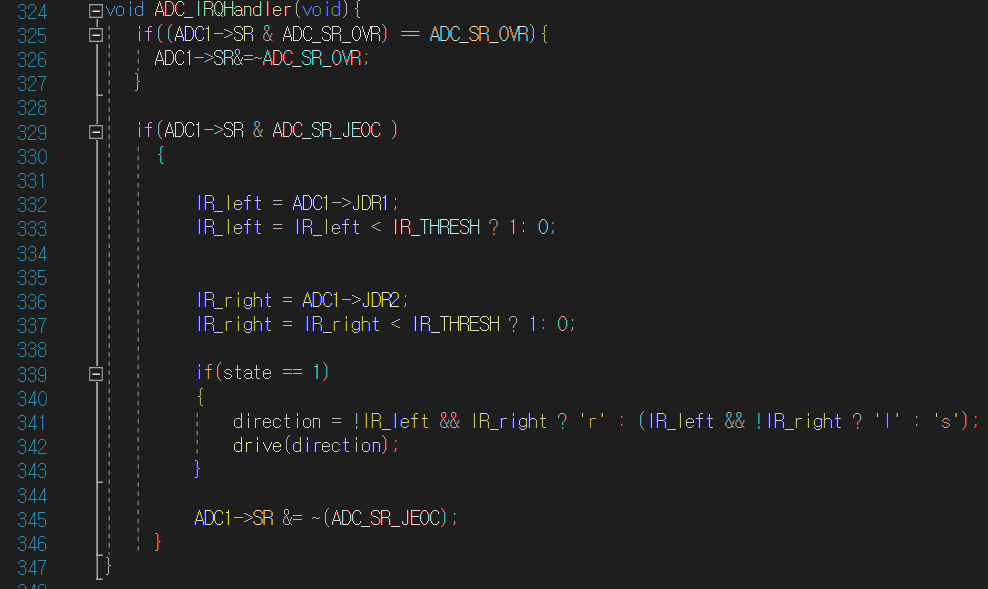


Figure 7. ADC handler

The ADC handler reads the IR value and sets the direction value. First, it is true if the IR threshold value is less than 1000 (experimentally obtained), and false if the threshold value is exceeded. If the threshold is not exceeded, the yellow lane is affected. Both IR algorithms are identical. After that, when the state is 1, that is, lane\_tracking mode, the direction is determined. If the value of IR\_Left is false and IR\_right is true, turn to the right. If the value of IR\_Left is true and IR\_right is false, turn left. If neither case, go straight to the yellow lane.

방향 맞게 적었는 지 체크 해보기

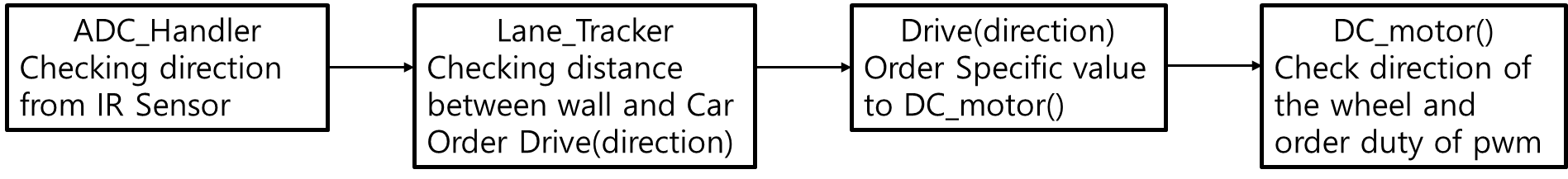


Figure 8. Simplify of Lane\_tracking mode

The above system is an open-loop control system. Therefore, the feedback is not applied, so proper tuning value and stable hardware are required.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Go Straight | Turn Right | Turn Left |

**Parking Algorithm**

Switch from lane tracking state 1 to state 2 parking vehicle mode. Parking vehicle mode also consists of states from 0 to 4. The above system is an open-loop control system. In order to use the fewest sensors, only one direction of parking is possible, so the parking algorithm proceeds according to each step and state.

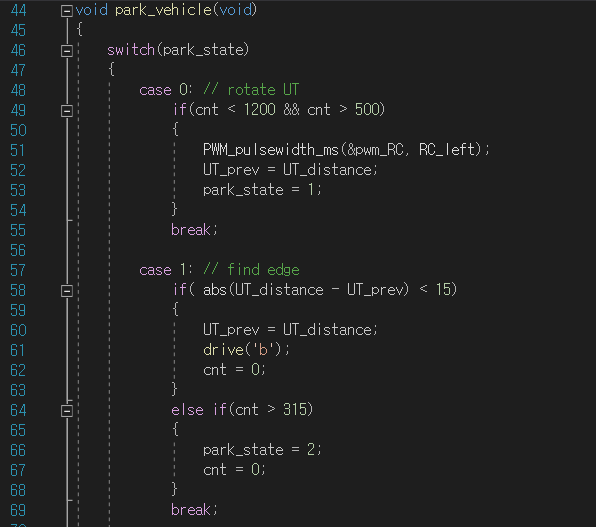


Figure 9. Park Vehicle Mode\_case 0 & case 1

In case 0, we set the UT sensor to look sideways. Then go to case 1 and adjust the RC car according to the value of the UT sensor. If the absolute value of the old value minus the new value of the UT sensor is greater than 15 [cm] (because the depth of the parking area is 15 [cm]), the previous value of the UT sensor is updated and the drive ('b Use the function to reverse the RC car backwards. In other words, the RC car is reversed until it reaches the edge of the parking area. However, this program is not feedbacked, so update the park state to 2 if the count being counted in TIM9 exceeds 315.

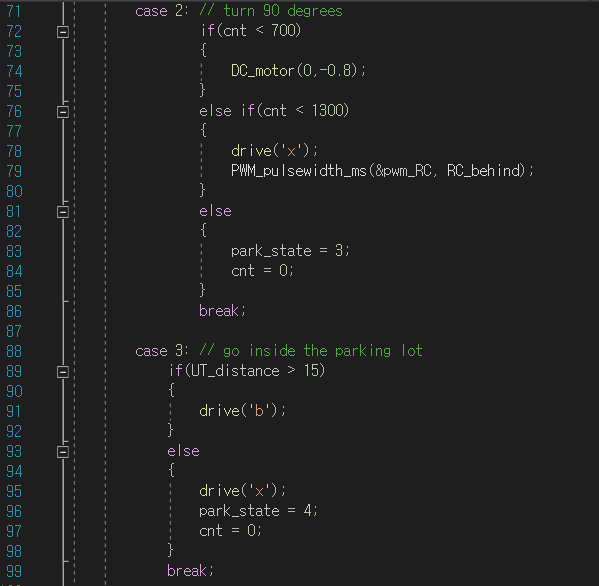


Figure 10. Park Vehicle Mode\_case 2 & case 3

In case 2, the RC car that reaches the edge is turned back. In other words, if you reverse the parking area, you are rotating 90 degrees so that you can park. Give an experimental value, DC\_motor (0, -0.8), and rotate it by 90 degrees. Still the UT sensor is looking to the side. Stop the RC car and stop the vehicle for a while using the experimentally obtained count. Set the UT sensor to look backwards for the next state progress. And update the park state to 3.

In case 3, reverse the RC car until the distance of the UT sensor looking at the rear of the vehicle is greater than 15. If the threshold is exceeded, update the park state to 4.

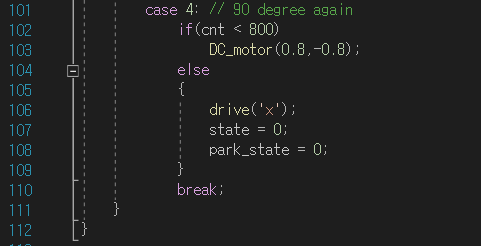


Figure 11. Park Vehicle Mode\_case 4

In case 4, the goal is to rotate the appropriately parked RC car 90 degrees into the correct parking space. Rotate by the number of counts and DC\_motor (0.8, -0.8) obtained experimentally. Then pause the vehicle and update the park state back to zero.

|  |  |
| --- | --- |
|  |  |
| Case 0 | Case 1 |
|  |  |
| Case 2 | Case 4 |
|  |  |
| Case 5 |

Figure 12. Overview of the Park Vehicle mode

**Results & Analysis**

**Conclusion**

Lane traking 성공 했다. 그 이유는 뭐라고 생각한당. Parking 도 잘 되었다.

**Appendix : Program Code**