리사이클봇 주행 코드 작성 및 수정

상태 완료

[arduino_motor.h]

```
arduino_motor.h
// ESP32 사용 코드입니다.
// 2MB APP / 2MB SPIFFS
// 모터드라이버 실드 세팅
#define BAT_READ_PIN 36
// 배터리 전압을 측정하기 위한 분압 저항
#define R1 20000.0
#define R2 4700.0

#define BOLT_FACTOR 1.1
#define VOLT_LIMIT 7000 // 저전압 경보 값
#define BUZZER_PIN 17

#define M1_A 33
#define M1_B 32

#define M3_A 25
#define M3_B 26

#define M2_A 2
#define M2_B 27

#define M4_A 12
#define M4_B 13

void stop();
```

```
void motor_init()
{
  pinMode(BUZZER_PIN, OUTPUT);
  // 모터 초기화
  pinMode(M1_A, OUTPUT);
  pinMode(M1_B, OUTPUT);
  pinMode(M2_A, OUTPUT);
  pinMode(M2_B, OUTPUT);
  pinMode(M3_A, OUTPUT);
  pinMode(M3_B, OUTPUT);
  pinMode(M4_A, OUTPUT);
  pinMode(M4_B, OUTPUT);

  ledcAttachPin(M1_A, 0);
  ledcAttachPin(M1_B, 1);
  ledcAttachPin(M2_A, 2);
  ledcAttachPin(M2_B, 3);
  ledcAttachPin(M3_A, 4);
  ledcAttachPin(M3_B, 5);
  ledcAttachPin(M4_A, 6);
  ledcAttachPin(M4_B, 7);

  ledcSetup(0, 5000, 8);
  ledcSetup(1, 5000, 8);
  ledcSetup(2, 5000, 8);
  ledcSetup(3, 5000, 8);
  ledcSetup(4, 5000, 8);
  ledcSetup(5, 5000, 8);
  ledcSetup(6, 5000, 8);
  ledcSetup(7, 5000, 8);
  stop();
}

void m1(int value)
{
  if ( value > 0 )
  {
    ledcWrite(0, value);
```

```
    ledcWrite(1, 0);
  }
  else
  {
    ledcWrite(0, 0);
    ledcWrite(1, -value);
  }
}

void m2(int value)
{
  if ( value > 0 )
  {
    ledcWrite(2, value);
    ledcWrite(3, 0);
  }
  else
  {
    ledcWrite(2, 0);
    ledcWrite(3, -value);
  }
}

void m3(int value)
{
  if ( value > 0 )
  {
    ledcWrite(4, value);
    ledcWrite(5, 0);
  }
  else
  {
    ledcWrite(4, 0);
    ledcWrite(5, -value);
  }
}

void m4(int value)
```

```
{
  if ( value > 0 )
  {
    ledcWrite(6, value);
    ledcWrite(7, 0);
  }
  else
  {
    ledcWrite(6, 0);
    ledcWrite(7, -value);
  }
}

void stop()
{
  m1(0);
  m2(0);
  m3(0);
  m4(0);
}
```

[esp32_car.ino]

```
#include <LiquidCrystal.h>

// ARDUINO ESP32 BOARD 2.0.5
// ESP32 BOARD : ESP32 Dev Module
// Partition Scheme : "No OTA(2MB APP/2MB SPIFFS)

// 터치 센서 제어
#define TOUCH_PIN 27
int touchCount = 0;

// 모터 제어
#define MOTOR_POWER 800 // 230이었던 거
#include "arduino_motor.h"
```

```
#define CHECK_DISTANCE 700 // 장애물 감지 거리 mm
bool auto_driving_mode = false; // 시작시 수동 제어모드로 시작

int obstacle_distance = 10000;
// 장애물이 있는 경우 좌우측으로 회전하지만 모서리의 경우 좌측 우측을 반복하는 현상이
// 따라서 장애물 감지후 좌회전을 한적이 있는지 우회전을 한적이 있는지 기억을 해 두었다
// 좌회전 우회전 모두 한적이 있으면 우회전을 하는 것으로 함.
int left_right_flag = 0;

int distance_list[36];

#include "BluetoothSerial.h"
BluetoothSerial SerialBT;

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// 일반적으로 LCD I2C의 주소는 0x27 혹은 0x3F입니다.
// LCD 출력이 되지 않는 경우 아래의 주소를 0x27과 0x3F로 변경해 보시기 바랍니다.
// 변경후에도 출력이 되지 않을 경우 i2c Scanner로 주소를 확인해 주시기 바랍니다.
LiquidCrystal_I2C lcd(0x27,16,2);
//LiquidCrystal_I2C lcd(0x3F,16,2);

#define BT_NAME "kProject_Car"

void setup()
{
  SerialBT.begin(BT_NAME);
  Serial.begin(115200);
  Serial1.begin(115200, SERIAL_8N1, 14, 23);

  // LCD 출력
  Wire.begin(18,5);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("BT NAME =");
  lcd.setCursor(0,1);
```

```
  lcd.print(BT_NAME);

  // 모터 초기화
  motor_init();

  digitalWrite(BUZZER_PIN, HIGH);
  delay(100);
  digitalWrite(BUZZER_PIN, LOW);
  delay(100);
  digitalWrite(BUZZER_PIN, HIGH);
  delay(100);
  digitalWrite(BUZZER_PIN, LOW);

  // 터치 센서 초기화
  pinMode(TOUCH_PIN, INPUT);
}

void loop()
{
  // 터치 센서 시리얼 모니터 출력
  if (digitalRead(TOUCH_PIN) == HIGH){
    Serial.println("Touched");
    delay(1000);
  }

  // 전압 체크 및 경보
  int val = analogRead(36); // 배터리 값 측정
  int volt = map(val, 0, 4095, 0, 3300); // 측정값을 mV단위로 변경
  int real_volt = volt * ( ( R1 + R2 ) / R2 ) * BOLT_FACTOR; // 실제 전압으로 환산

  // 배터리 전압 출력
  Serial.print(real_volt);
  Serial.println("V");

  // 배터리 저전압일 경우 부저 울림
  if ( real_volt < VOLT_LIMIT )
  {
    digitalWrite(BUZZER_PIN, HIGH);
```

```
}
else
{
  digitalWrite(BUZZER_PIN, LOW);
}

if ( Serial1.available() >= 74 )
{
  int id = Serial1.read();
  if ( id == 20 )
  {
    int cmd = Serial1.read();
    if ( cmd == 6 ) // 라이다 값을 가져온 경우
    {
      uint8_t value[72];
      for ( int i = 0; i < 36; i++)
      {
        value[i*2] = Serial1.read();
        value[i*2+1] = Serial1.read();
      }

      if ( Serial1.read() == 21 )
      {
        Serial.println("LIDAR OK");
        SerialBT.write(255);
        SerialBT.write(20);
        for ( int i = 9; i < 36; i++)
        {
          SerialBT.write(value[i*2]);
          SerialBT.write(value[i*2+1]);
          distance_list[i] = ( value[i*2+1] << 8 ) | value[i*2];
        }
        for ( int i = 0; i < 9; i++)
        {
          SerialBT.write(value[i*2]);
          SerialBT.write(value[i*2+1]);
          distance_list[i] = ( value[i*2+1] << 8 ) | value[i*2];
        }
```

```
obstacle_distance = 10000;
obstacle_distance = min(obstacle_distance, int(distance_list[27]));
obstacle_distance = min(obstacle_distance, int(distance_list[28]));
obstacle_distance = min(obstacle_distance, int(distance_list[29]));
//      obstacle_distance = min(obstacle_distance, int(distance_list[21]));
obstacle_distance = min(obstacle_distance, int(distance_list[26]));
obstacle_distance = min(obstacle_distance, int(distance_list[25]));
//      obstacle_distance = min(obstacle_distance, int(distance_list[15]));

int obstacle_left = 10000;
int obstacle_right = 10000;

obstacle_right = min(obstacle_right , int(distance_list[28]));
obstacle_right = min(obstacle_right , int(distance_list[29]));
obstacle_right = min(obstacle_right , int(distance_list[30]));

obstacle_left = min(obstacle_left , int(distance_list[26]));
obstacle_left = min(obstacle_left , int(distance_list[25]));
obstacle_left = min(obstacle_left , int(distance_list[24]));

/*
// 터치 센서에 값이 입력된 경우 터치 횟수를 증가
if (digitalRead(TOUCH_PIN) == HIGH) {
  touchCount++;
}

// 터치 횟수가 1번일 때 자율주행 모드 꺼짐
if (touchCount == 1) {
  auto_driving_mode == false;
  stop();
}

//터치 횟수가 2번일 때 자율주행 모드 켜짐
if (touchCount == 2) {
  auto_driving_mode == true;
  //터치 횟수 초기화
  touchCount == 0;
```

```
        }
        */

        if ( auto_driving_mode == true )
        {
          if ( obstacle_distance < CHECK_DISTANCE )
          {
            if ( left_right_flag == 0b00000011 ) // 장애물 감지후 좌회전 우회전 모두 해
            {
              right();
            }
            else
            {
              if ( obstacle_right < obstacle_left )
              {
                left_right_flag = left_right_flag | 0b00000001; // 좌회전 한적이 있음을
                left();
              }
              else
              {
                left_right_flag = left_right_flag | 0b00000010; // 우회전 한적 있음을 설
                right();
              }
            }
          }
          else
          {
            left_right_flag = 0; // 좌회전 우회전 한적 없음으로 설정
            front();
          }
        }
      }
    }
  }
}

if ( SerialBT.available() )
{
```

```
int cmd = SerialBT.read();
if ( cmd == '2' )
{
  Serial.println("FRONT");
  auto_driving_mode = false;
  front();
}
if ( cmd == '4' )
{
  Serial.println("LEFT");
  auto_driving_mode = false;
  left();
}
if ( cmd == '5' )
{
  Serial.println("STOP");
  auto_driving_mode = false;
  stop();
}
if ( cmd == '6' )
{
  Serial.println("RIGHT");
  auto_driving_mode = false;
  right();
}
if ( cmd == '8' )
{
  Serial.println("BACK");
  auto_driving_mode = false;
  back();
}
if ( cmd == '9' )
{
  Serial.println("FRONT");
  auto_driving_mode = true;
  front();
}
if ( cmd == '0' )
```

```
    {
      Serial.println("STOP");
      auto_driving_mode = false;
      stop();
    }
  }
}


void front_left()
{
  m1(MOTOR_POWER); // 2분의 1
  m2(MOTOR_POWER * 2);
  m3(MOTOR_POWER); // 2분의 1
  m4(MOTOR_POWER * 2);
}

void front_right()
{
  m1(MOTOR_POWER * 2);
  m2(MOTOR_POWER); // 2분의 1
  m3(MOTOR_POWER * 2);
  m4(MOTOR_POWER); // 2분의 1
}

void front()
{
  m1(MOTOR_POWER);
  m2(MOTOR_POWER);
  m3(MOTOR_POWER);
  m4(MOTOR_POWER);
}

void back()
{
  m1(-MOTOR_POWER);
  m2(-MOTOR_POWER);
  m3(-MOTOR_POWER);
```

```
  m4(-MOTOR_POWER);
}

void left()
{
  m1(-MOTOR_POWER);
  m2(MOTOR_POWER);
  m3(-MOTOR_POWER);
  m4(MOTOR_POWER);
}

void right()
{
  m1(MOTOR_POWER);
  m2(-MOTOR_POWER);
  m3(MOTOR_POWER);
  m4(-MOTOR_POWER);
}
```

- 수정/추가 부분
  - 모터 속도 - 수거부의 무게로 인해 주행 속도가 느려짐. 따라서 모터 속도를 230 → 800으로 수정.
  - 터치 센서 사용을 위해 변수 추가.

    `#define TOUCH_PIN 27`

    `int touchCount = 0;`
  - 터치 센서 초기화를 위해 코드 추가.

    `pinMode(TOUCH_PIN, INPUT);`
  - 터치 센서 디버깅을 위해 조건문 추가.

    ```
    if (digitalRead(TOUCH_PIN) == HIGH){
        Serial.println("Touched");
        delay(1000);
      }
    ```
  - 터치 센서에 따른 주행 변경 코드 추가.

```
// 터치 센서에 값이 입력된 경우 터치 횟수를 증가
if (digitalRead(TOUCH_PIN) == HIGH) {
   touchCount++;
}

// 터치 횟수가 1번일 때 자율주행 모드 꺼짐
if (touchCount == 1) {
   auto_driving_mode == false;
   stop();
}

//터치 횟수가 2번일 때 자율주행 모드 켜짐
if (touchCount == 2) {
   auto_driving_mode == true;
   //터치 횟수 초기화
   touchCount == 0;
}
```

⇒ 다만 해당 코드의 경우 제대로 실행되지 않음. TOUCH_PIN이 활성화 되는 것을 시리얼 모니터를 통해 확인 했지만, touchCount 변화에 따른 자율주행 모드 변경이 제대로 이루어지지 않음.

해당 코드 테스트 중 라이다 센서 고장으로 진도를 나갈 수 없게 됨.