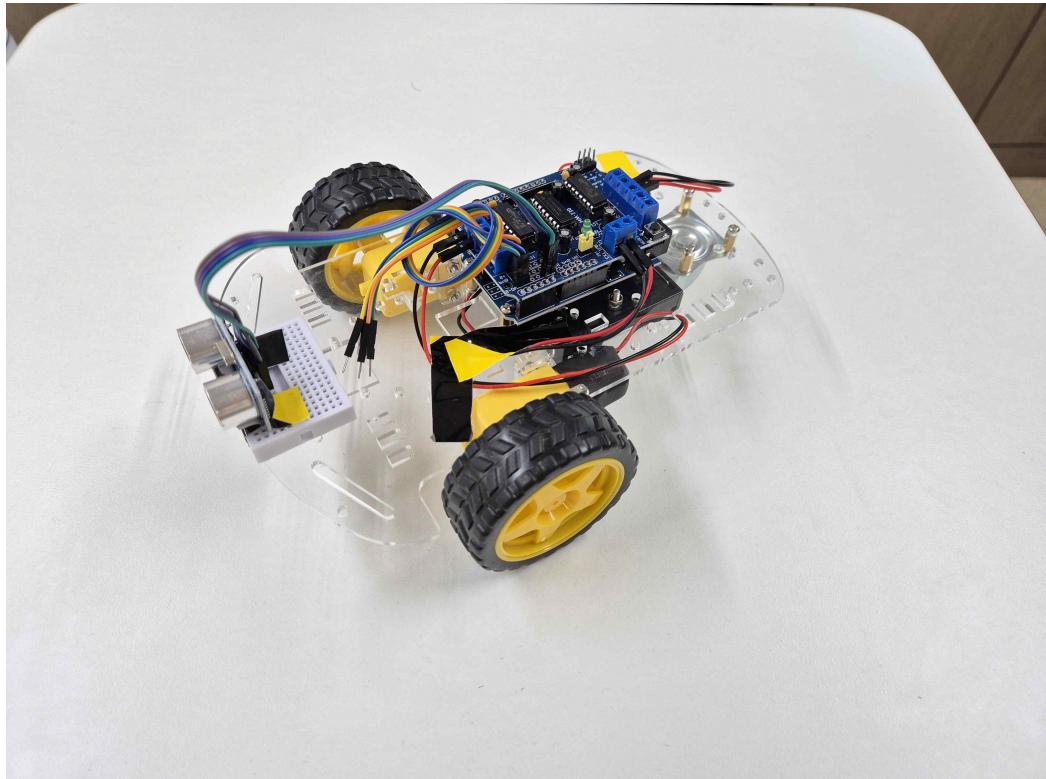


# 자율주행 RC카 프로젝트



마이크로프로세서 실습  
영남대학교/컴퓨터공학과

22112042 김동욱

22211998 김민준

22311947 김서현

22112110 김성민

# 목차

## I. 서론

## II. 배경 및 사전 조사

1. 자율주행 RC카
2. 사용 부품

## III. 수행 과정 및 절차

1. 몸체 제작
2. 아두이노 프로그램

## IV. 결론

1. 프로젝트 결과
2. 향후 개선 방안

## I . 서론

최근 자율주행 기술은 다양한 분야에서 급격히 발전하고 있다. 이를 적용한 여러 프로젝트가 활발하게 이루어지고 있으며 이번 자율주행 RC카 프로젝트에서는 아두이노를 활용한 자율주행 RC카를 실습하게 될 것이다.

아두이노는 기본적으로 다양한 함수와 라이브러리를 지원하며 센서와 부품들을 쉽게 활용하는 것이 가능하다. 이러한 이점들로 인해 자율주행 RC카의 경로 인식 및 주행 제어에 필요한 핵심 기술들을 쉽고 효율적으로 구성할 수 있다.

자율주행 RC카를 제작하기 위해선 주행 중 다양한 센서 정보를 바탕으로 실시간 경로를 분석하고 또 조정하는 단계들이 필요하다. 이 과정에서 아두이노는 센서의 입력을 처리하고 모터를 제어하며 자율주행 RC카 작동의 핵심적인 역할을 맡는다.

이번 프로젝트의 주된 목표는 이러한 아두이노 기술을 활용하여 자율주행 RC카가 주어진 주행 경로를 정확하게 따르고 목표 지점에 보다 더 빠르게 도달하는 것이다. 평가 기준으로는 목표 지점까지의 도착 시간, RC카의 경로 이탈 횟수, 이탈한 경로를 조정하기 위한 터치 횟수 등이 존재한다. 경로 이탈 횟수와 터치 횟수가 많을수록 감점 요소로 적용되며 이를 최소화하는 것이 중요한 개선 사항이 될 것이다.

## II . 배경 및 사전 조사

### 1. 자율주행 RC카

자율주행 RC카는 센서와 모터, 제어 장치 등을 활용해 프로그래밍 된 알고리즘에 따라 장애물에 부딪히지 않고 스스로 경로를 찾아 주행하는 원격 조종 자동차를 의미한다.

이번 프로젝트에서 DC모터는 차량의 이동을 실제로 구현하고 모터 드라이버 쇼드는 DC모터의 속도와 방향을 결정하게 될 것이며, 초음파 센서는 차량 주변의 장애물을 감지하는 역할을 맡게 될 것이다. 이러한 부품은 모두 아두이노에 연결되어 자율주행 RC카의 움직임과 진행 방향을 제어한다.

이때 차량을 이동시키는 DC 모터는 배터리 잔량, 모터의 사양 등의 여유 요소에 따라 출력이 조금씩 다르게 나오게 된다. 따라서 왼쪽 바퀴와 오른쪽 바퀴에 연결된 모터의 속도를 조정하여 양쪽 바퀴의 속도가 최대한 같게 만드는 것이 차량의 움직임에 있어 가장 중요한 핵심이 될 것이다.

장애물을 감지하는 초음파 센서는 초음파 송신부에서 초음파 신호를 발사하고 신호가 물체에 반사되어 돌아오는 시간을 측정해 장애물과의 거리를 계산한다. 이때 거리  $d$ 를 구하기 위한 식은  $d = (t * v) / 2$ 가 되며  $t$ 는 초음파가 왕복하는 데 걸린 시간,  $v$ 는 초음파의 속도이다.

## 2. 사용 부품

### 가. 아두이노 UNO



우노 R3 호환보드는 자율주행 RC카의 제어를 위한 마이크로컨트롤러 보드로 전체 시스템의 제어 및 계산을 담당한다. 초음파 센서와 모터 드라이버 쉴드 등을 연결해 센서 데이터 처리와 모터 제어 알고리즘을 실행한다.

### 나. DC 모터



자율주행 RC카 몸체의 바퀴를 구동한다. 전압에 따라 회전 방향과 속도를 제어할 수 있다. 회전을 통해 전진, 후진, 좌우 회전 등의 움직임을 만들어낸다.

모터 드라이버 쉴드에 연결해 보다 복잡한 명령을 실행할 수 있다.

### 다. 모터 드라이버 쉴드



DC모터 또는 서브 모터의 방향과 속도를 제어하기 위한 확장형 쉴드. 이번 프로젝트에서 사용된 모터 드라이버 쉴드는 4개의 DC모터 또는 2개의 서브 모터를 동시에 제어할 수 있다. 아두이노와 모터 사이의 전기적 인터페이스 역할을 수행한다. 모터를 제대로 작동시키기 위해서는 아두이노 외에 별도의 전력이 추가로 더 필요하다.

### 라. 초음파 센서



장애물 감지를 위한 거리 측정 센서. 차량 주변의 장애물을 감지하고 거리를 측정하여 자율주행에 필요한 환경 정보를 제공한다.

초음파 센서의 Trig는 초음파를 발생시키며 Echo는 장애물에 반사되어 돌아오는 초음파를 수신한다. 초음파 발생 시간과 반사되어 돌아오는 시간차를 이용해서 거리를 측정할 수 있다.

### 마. 기타 부품들

이 외에도 RC카의 여러 부품을 고정하는 역할을 하는 전체 프레임, 초음파 센서 등에서 전기 회로의 구성을 돋는 미니 브래드 보드, RC카의 동력원이 되어줄 배터

리와 배터리 홀더, DC모터와 연결되어 RC카를 구동시키게 될 바퀴와 RC카의 균형을 잡아줄 보조 바퀴 등의 부품들이 필요하다.

### III. 수행 과정 및 절차

#### 1. 몸체 제작

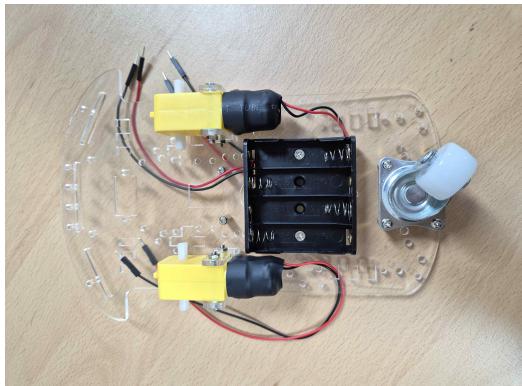
##### 가. DC모터 연결



모터선의 위치 및 블트의 결합 방향을 확인하며 아크릴 프레임 아래면에 모터를 고정한다.

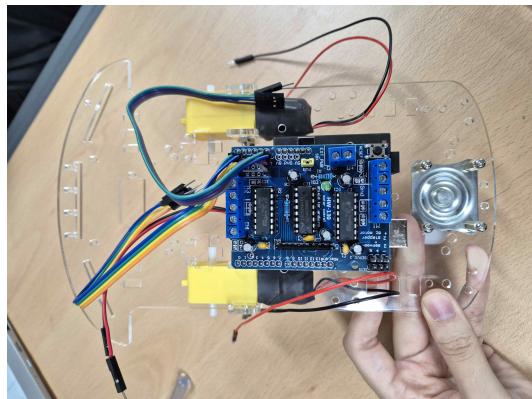
추가로 보조 바퀴와 아두이노 보드를 결합하기 위한 서포트를 각 위아래로 4개 2개씩 결합한다. 이때 보조 바퀴를 위한 서포트는 아크릴 프레임의 아래면, 아두이노 보드를 위한 서포트는 윗면 방향으로 결합한다.

##### 나. 배터리 홀더 및 보조 바퀴 연결



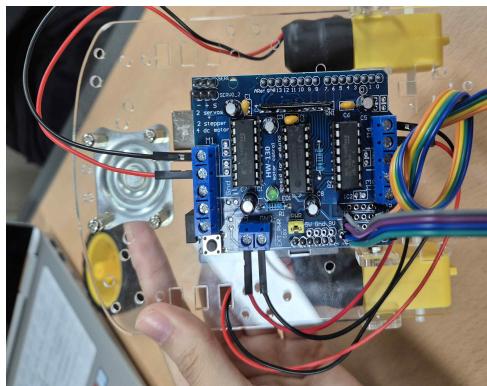
아크릴 프레임의 아래면에 보조 바퀴를 결합하고 배터리 홀더를 고정한다. 케이블 출구를 통해 DC모터와 배터리 홀더의 케이블들을 아크릴 프레임 윗면으로 빼내 정리한다.

##### 다. 아두이노 보드 및 모터 드라이버 쉘드 결합



윗면의 서포트에 아두이노 보드를 결합한다. 결합된 보드 위에 모터 드라이버 쉘드(L293D)를 설치한다.

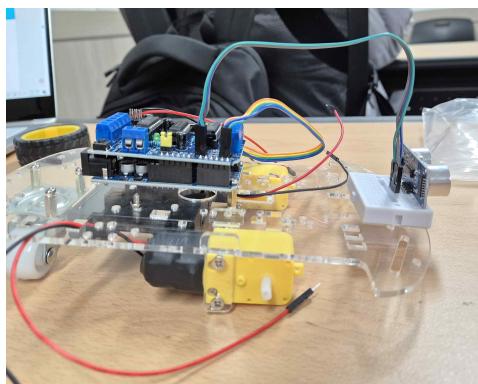
#### 라. 모터 드라이버 쉴드 연결



아랫면 서포트에 고정된 양쪽 DC 모터의 선을 모터 드라이버 쉴드와 연결 한다. 이때 각 모터의 방향과 모터 드라이버 쉴드의 커넥터 위치를 주의해야 한다.

모터 드라이버 쉴드의 외부전원 소켓에는 배터리 홀더를 연결한다.

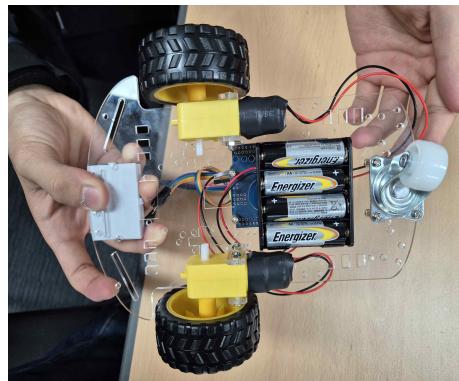
#### 마. 초음파 센서 연결



아크릴 프레임 전면부에 브래드보드를 부착하고 아래 결선표를 참고해 모터 드라이버 쉴드와 초음파 센서를 결합 한다.

| 모터 드라이버 쉴드<br>(L293D) | 초음파센서 |
|-----------------------|-------|
| 5V                    | VCC   |
| GND                   | GND   |
| A0                    | Trig  |
| A1                    | Echo  |

#### 바. 바퀴 및 배터리



마지막으로 DC모터에 바퀴를 결합하고 배터리 홀더에 건전지를 결합하여 RC카를 구동할 준비를 마친다.

## 2. 아두이노 프로그램

미리 작성된 완료한 프로그램을 아두이노에 업로드하여 자율주행 RC카에 필요한 소프트웨어를 삽입한다. 이때 모터 드라이버 쉴드를 제어하기 위한 라이브러리의 헤더 파일은 Arduino IDE에 기본적으로 포함되어 있지 않다. 따라서 <AFMotor.h> 헤더 파일을 사용하기 위해서는 별도로 “Adafruit Motor Shield” 라이브러리를 설치하는 과정이 꼭 필요하다.

특히 프로그램을 작성할 때 RC카가 장애물을 만났을 때의 행동을 결정하는 Obstacle\_Check() 함수에 가장 중점을 두었다. 초반에는 직진 후 장애물을 만났을 때 왼쪽으로 회전해 길을 먼저 살피고 만약 왼쪽 길이 막혀 있다면 오른쪽으로 회전해 길을 살피는 알고리즘을 사용했으나 코드 구현상의 문제와 센서 성능의 한계로 실패했다. 센서의 성능이 부족한 탓에 자주 장애물을 인식하지 못하는 경우가 생겼고 오른쪽으로 연속 2번을 회전해 왔던 길을 되돌아가는 문제점이 발생했다.

다음으로는 직진 여부와 관계없이 장애물을 만났을 때 왼쪽 또는 오른쪽으로 한번, 그래도 길이 존재하지 않는다면 반대쪽으로 두 번 회전하도록 함수를 구현했다. 이를 위해 정수값을 담은 변수 count를 선언하고 count 값의 변화에 따라 회전하는 방향을 조절할 수 있도록 하였다.

또한 여러 번의 실험 중 회전 이후에 바로 직진을 실행할 경우 바퀴에 남아있는 회전력 때문에 경로가 틀어지는 현상이 발생한다는 것을 알게 되었다. 이러한 현상을 해결하기 위해 회전 이후에는 반드시 완전히 정지하고 그 후 다시 직진을 시작하는 방식을 택했다.

아래는 자율주행 RC카 프로그램 코드와 해당 코드를 설명한 주석들이다.

```
#include <SoftwareSerial.h>
#include <AFMotor.h>
AF_DCMotor motor_L(1); //모터 드라이버 쉴드 변수 선언
AF_DCMotor motor_R(4);

//양쪽 모터의 출력 다를 수 있으므로 최대한 직진하도록 조정
int Lspeed = 164;           //좌측 모터 속도 164
int Rspeed = 200;           //우측 모터 속도 200

//초음파센서 출력핀(trig)과 입력핀(echo), 변수 선언
int TrigPin = A0;
int EchoPin = A1;
long duration, distance;
int count = 0; //Obstacle_Check() 함수를 위한 변수

//loop를 돌리기 위한 함수들을 선언
void Obstacle_Check();
```

```

void Distance_Measurement();
void Forward();
void Backward();
void Right();
void Left();
void Stop();

//setup//
void setup() {
    Serial.begin(9600);
    Serial.println("Eduino Smart Car Start!");

    pinMode(EchoPin, INPUT);           // EchoPin 입력
    pinMode(TrigPin, OUTPUT);         // TrigPin 출력

    motor_L.setSpeed(Lspeed);         // 왼쪽 모터의 속도
    motor_L.run(RELEASE);
    motor_R.setSpeed(Rspeed);         // 오른쪽 모터의 속도
    motor_R.run(RELEASE);
}

//loop//
void loop() {
    Forward(); //직진
    delay(50);
    Obstacle_Check(); //장애물을 확인 및 회피 방향 결정
}

//장애물을 확인하고 회피 방향을 결정하는 함수//
void Obstacle_Check() {
    Distance_Measurement();
    delay(50);

    Serial.println(distance);

    while (distance < 500) {
        //장애물과의 거리가 200 이하라면 후진 후 다시 거리 측정
        if (distance < 200) {
            Backward();
        }
    }
}

```

```

delay(800);
Stop();
delay(400); //완전히 정지한 뒤 거리를 측정하도록 정지 시간을 늘림
Distance_Measurement();
delay(100);
}

//장애물과의 거리가 200 이상 500 이하라면
//왼쪽과 오른쪽을 번갈아가며 순차적으로 거리 측정
else {
    if(count == 0){
        Left();
        delay(950);
        count++;
    }
    else if(count == 1){
        Right();
        delay(670);
        count++;
    }
    else if(count == 2){
        Right();
        delay(670);
        count++;
    }
    else {
        Left();
        delay(950);
        count=0;
    }
    Stop(); //회전을 완전히 정지하고 다음 동작을 하도록 딜레이를 줌
    delay(400);
    Distance_Measurement();
    delay(100);
}
}

//장애물과의 거리를 측정하는 함수//
void Distance_Measurement() {

```

```

digitalWrite(TrigPin, LOW);
delay(2);
digitalWrite(TrigPin, HIGH); //trigPin에서 초음파 발생(echoPin도 HIGH)
delayMicroseconds(10);
digitalWrite(TrigPin, LOW);
duration = pulseIn(EchoPin, HIGH); //echoPin이 HIGH를 유지한 시간을 저장
//duration값과 거리 공식으로 구해진 결과를 distance 변수에 저장
distance = ((float)(340 * duration) / 1000) / 2;
delay(50);
}

//모터를 제어하는 함수들//
//직진과 후진의 경우 왼쪽 모터와 오른쪽 모터를 동일하게 동작
void Forward() { //직진 함수//
    motor_L.run(FORWARD); motor_R.run(FORWARD);
    motor_L.setSpeed(Lspeed); motor_R.setSpeed(Rspeed);
}

void Backward() { //후진 함수//
    motor_L.run(BACKWARD); motor_R.run(BACKWARD);
    motor_L.setSpeed(Lspeed); motor_R.setSpeed(Rspeed);
}

void Right() { //오른쪽으로 회전하는 함수//
    //왼쪽 바퀴는 앞으로, 오른쪽 바퀴는 뒤로 회전
    motor_L.run(FORWARD); motor_R.run(BACKWARD);
    //오른쪽 바퀴의 속도를 50퍼센트 줄여 부드러운 회전이 가능하도록 설정
    motor_L.setSpeed(Lspeed); motor_R.setSpeed(Rspeed*0.5);
}

void Left() { //왼쪽으로 회전하는 함수//
    motor_L.run(BACKWARD); motor_R.run(FORWARD);
    motor_L.setSpeed(Lspeed*0.5); motor_R.setSpeed(Rspeed);
}

void Stop() { //정지 함수//
    //양쪽 모터를 모두 RELEASE 상태로 설정
    motor_L.run(RELEASE); motor_R.run(RELEASE);
    //양쪽 모터의 속도를 0으로 설정
}

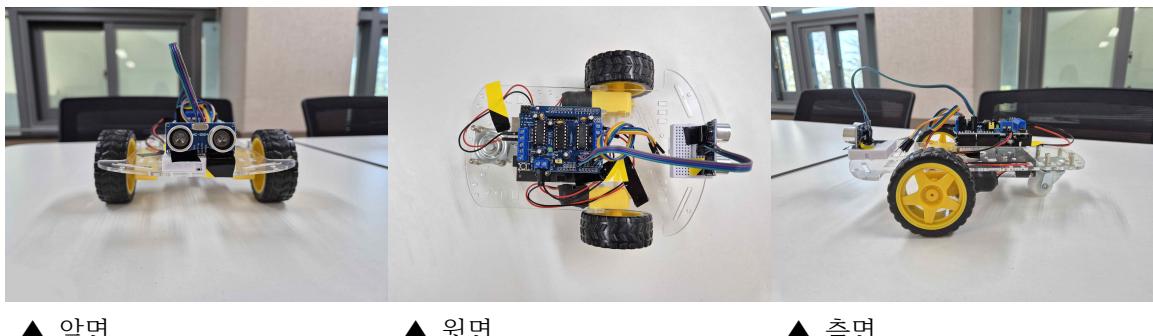
```

```
    motor_L.setSpeed(0);  motor_R.setSpeed(0);
}
```

## IV. 결론

### 1. 프로젝트 결과

아래는 완성된 자율주행 RC카의 앞면, 윗면, 측면을 찍은 사진과 동작을 촬영한 영상의 URL이다.



▲ 앞면

▲ 윗면

▲ 측면

[https://drive.google.com/file/d/14Z-ICQGF\\_p\\_HgS1A7So8FZJevn4raNOU/view?usp=drive\\_link](https://drive.google.com/file/d/14Z-ICQGF_p_HgS1A7So8FZJevn4raNOU/view?usp=drive_link)

이번 프로젝트에서 자율주행 RC카를 완성하기 위해 아두이노를 제외하고 DC 모터, 모터 드라이버 쉘드, 초음파 센서가 주요 부품으로 사용되었다. 프로젝트를 시작하기 전에는 이 세 가지 부품을 한 번에 사용해야 한다는 사실이 부담스럽게 느껴지기도 했었다. 하지만 막상 몸체 제작을 시작하자 부품을 사용하는 것에 있어 큰 어려움은 없었다.

아두이노를 포함한 부품들은 이미 수업 시간에 완전히 실습을 마친 부품들이었고 필요한 곳에 부품을 연결하고 몸체를 제작하는 과정은 모두 원활하게 이뤄졌다. 몇 가지 기본적인 부품들만으로 장애물을 피해 가는 자율주행 RC카를 만들 수 있다는 점이 놀랍기도 했고 추후 더 다양한 아두이노 프로젝트에 도전해 보고 싶다는 자신감이 생겼다.

특히 이번 프로젝트를 진행하면서 가장 의외라고 느껴졌던 부분은 실제 RC카를 제작하는 것만큼이나 제작된 기기를 검증하는 과정에서 긴 시간이 걸린다는 것이었다. Obstacle\_Check() 함수의 알고리즘 설정은 물론이고 양쪽 모터의 속도를 조절하고 장애물을 인식할 가장 적당한 거리를 찾기 위해 몇 번이나 변수의 값을 수정해야 했다.

이 과정에서 프로그래밍만큼이나 프로그래밍과 하드웨어와의 연계가 중요하다는 것을 알게 되었으며, 센서 활용과 모터 제어 등에서 많은 센스가 필요하다는 점을 다시 한번 깨달았다.

## 2. 향후 개선 방안

이번 프로젝트에서 자율주행 RC카의 주행 성능은 RC카가 ‘얼마나 빠르고 정확하게 주행 경로를 추적할 수 있는지’에 따라 좌우된다. RC카에 삽입된 프로그램에서는 정수형 변수를 사용하여 바퀴의 속도를 제어했다. 하지만 이 경우 모터의 속도를 정수형 변수로 조절한 탓에 바퀴 속도의 세밀한 조정이 불가능했고 따라서 완전한 직진을 만들기에는 한계가 있었다는 단점이 존재했다.

또한 직선 주행에서는 큰 문제를 보이지 않았지만 커브 구간을 지나면서 양쪽 바퀴 회전 속도 차이가 발생하며 우리가 원하던 경로와 방향을 벗어나는 현상이 자주 발생했다. 따라서 코너링 과정과 경로를 추적하는 것에 기존의 방식보다 더 섬세한 제어와 조정이 필요하다는 것을 체감하게 되었다.

자율주행 RC카가 제대로 경로를 따라가기 위해서는 다양한 센서를 통해 데이터를 분석하고 주행 경로를 설정하는 과정이 무엇보다 중요하다. 하지만 이번 제작에서 사용한 초음파 센서의 경우 장애물을 인식할 수 있는 범위가 매우 한정적이었다. 곡선 구간에 놓인 장애물이나 폭이 좁은 장애물 등 장애물이 바로 앞을 확실하게 막아선 상황이 아니라면 센서를 인식하는 것에 한계가 있었다.

예를 들어 커브를 돌다가 폭이 좁은 형태의 장애물을 만날 경우, 해당 장애물을 감지하지 못해서 경로를 벗어나는 경우가 빈번히 발생하였다. 이러한 문제는 자율주행 시스템에서 센서의 정확성과 범위가 중요하게 여겨져야 하는 이유를 보여주기도 했다.

만약 추후 다시 자율주행 RC카에 관련된 프로젝트를 진행하게 된다면 앞서 말한 프로그래밍의 아쉬움과 하드웨어 제작의 아쉬움을 개선해 더 나은 자율주행 RC카를 제작할 수 있을 것이다.