

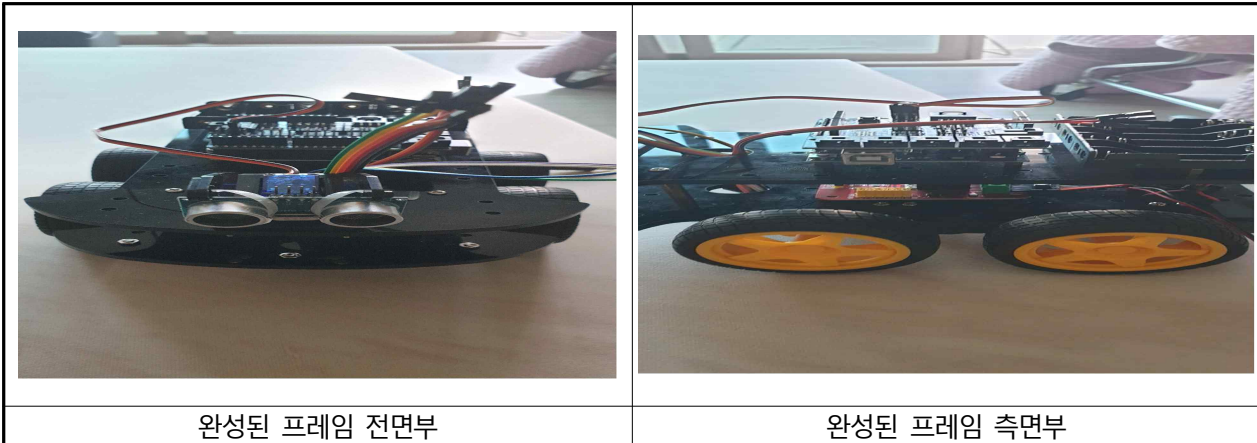
# 캡스톤디자인 결과보고서

(팀별 작성)

## 1. 교과목 및 팀 구성

학부(과)	과목명	학점	담당 교수	요일/교시
ICT	캡스톤디자인	2	홍진근	수/4,5
시제(작)품 명	자율주행 스마트카			
참여학생명단	학부	학번	성명	
	ICT학부	20161267	김동준	
	ICT학부	20181242	김도형	

## 2. 캡스톤디자인 과제 결과(시제(작)품을 사진으로 증빙)



완성된 프레임 전면부

완성된 프레임 측면부

```

9_RC_CAR_Line.ino
35 pinMode(trigPin, OUTPUT); // trigPin 출력
36
37 // 모터 핀 설정
38 pinMode(rightMotor_1_pin, OUTPUT); // 오른쪽 모터 전동부 출력
39 pinMode(leftMotor_1_pin, OUTPUT); // 왼쪽 모터 전동부 출력
40 pinMode(rightMotor_2_pin, OUTPUT); // 오른쪽 모터 앞바퀴 출력
41 pinMode(leftMotor_2_pin, OUTPUT); // 왼쪽 모터 앞바퀴 출력
42 pinMode(rightMotor_3_pin, OUTPUT); // 오른쪽 모터 뒷바퀴 출력
43 pinMode(leftMotor_3_pin, OUTPUT); // 왼쪽 모터 뒷바퀴 출력
44
45 Serial.begin(9600); // PC와의 시리얼 통신 값 9600
46 Serial.println("Activate!"); // 통신에 성공했을 경우, 시리얼 모니터 창에 Activate! 출력
47
48 // 라인트레이싱 관련 설정
49
50 void loop() {
51   int L = digitalRead(L_line);
52   int C = digitalRead(C_line);
53   int R = digitalRead(R_line);
54
55   Serial.print("digital : ");
56   Serial.print(L);
57   Serial.print(", ");
58   Serial.print(C);
59   Serial.print(", ");
60   Serial.print(R);
61   Serial.print("\n");
62   Serial.print(" ");
63
64   // 라인트레이싱 완성 설정
65   if ( L == LOW && C == LOW && R == LOW ) { // 0 0 0

```

Arduino IDE 통한 개발 환경

## 3. 자체평가

하드웨어 프레임 조립, 코드 작성, 트랙제작, 주행테스트 / 구글과 유튜브를 참고함

#### 4. 담당교수 평가 및 의견

--

2022 년 12월 14일

담당교수: 홍진근 (서명 또는 날인)

전공주임교수: 김귀정 (서명 또는 날인)

**교무처장·커리큘럼인증센터소장 귀하**

# 캡스톤 디자인 결과보고서

프로젝트 이름 : 자율주행 스마트카

정보보호학과 20161267 김동준

정보보호학과 20181242 김도형

## 목차

1 . 프로젝트 정의서

2 . 프로젝트 설계서

3 . Use-case

4 . 프로젝트 소스 코드

5 . 주차별 보고서

6 . 프로젝트 결과물

7 . 프로젝트를 진행하며 느낀점

# 1 . 프로젝트 정의서

## 프로젝트 이름

자율주행 스마트카

### 1. 프로젝트의 개발배경 및 필요성

미래 자동차 산업은 C.A.S.E라는 이름의 4가지 방향으로 변화할 것이라고들 이야기한다. 커넥티드카(Connected), 자율주행차(Autonomous), 공유서비스(Share), 전기차(Electric)의 앞 글자를 따온 단어이다.

특히 요즘 전기차에 대한 사람들의 관심과 발전속도가 남다른데, 많은 전문가들이 미래에는 내연기관 자동차가 아닌 전기 자동차가 도로위에서 주를 이룰 것이라고들 이야기 한다.

그리고 또 하나는 전기차하면 빼놓을수 없는 주제가 자율주행이다.

이러한 자율주행은 기술은 여러 가지 장점들 때문에 많은 자동차 회사에서 이러한 자율주행 기술을 도입하려 노력하고 있다.

#### 1.활용률 극대화

오랜 시간 지구상의 자동차는 하루의 90%이상을 주차장에서 멈춰서서 보내왔지만 2010년대에 들어 이런 문제를 해결하기 위한, 이른바 ‘공유경제’가 떠오르기 시작했다.

우버(Uber)와 같은 차량 호출 서비스(Car Hailing), 쏘카(Socar)와 같은 차량 공유 서비스(Car Sharing)가 출현하면서, 자동차를 좀 더 효율적으로 이용하고 하는 노력들이 지속되어왔다.

하지만 이런 서비스의 단점 중 하나는, 운영에 인간 노동자가 필수적이라는 것이다.

우버 기사가 하루 10시간 이상 운전하며 승객을 받기는 쉽지 않다.

쏘카 공유차량 역시 한 사람의 사용이 끝나면 오랜 시간 새로운 사용자를 기다리며 주차장에서 대기해야 한다.

자율주행은 이러한 한계를 극복하고 차량의 활용률을 극대화하기 위한 해결책이다.

무인 운전이 가능한 수준의 완전 자율주행이 상용화 된다면, 인간 기사 없이 24시간 내내 새로운 승객을 스스로 찾아다니는 무인 택시, 차량 대여 서비스가 가능해질 것이다.

지금 많이 상용화되고 있는 무인가게에서 더 나아간 발전형태로 볼 수 있을 것이다.

#### 2.교통사고 감소

자동차 사고는 주로 운전자들의 실수에서 비롯되는데 졸음과 반응 시간 등의 여러 이유로 항상 좋은 컨디션이 지속되기는 힘들다.

반면 자율주행 자동차는 360도의 시야와 다른 사물을 바로 인지할 수 있는 능력을 통해 사고의 발생을 줄일 수 있다.

따라서 보다 정교하고 정확한 자율주행 시스템이 도입된다면, 교통사고의 확률이 감소하고, 결론적으로 사망률까지도 낮출 수 있다.

## 2. 프로젝트의 목표

현재 IoT 시대에는 단순한 자동차 기업이 아니라 IT기업이 자율주행차 분야에서 가장 앞선 기술력을 가진 기업으로 평가 받는 만큼 구글, 애플과 같은 세계적인 IT기업들이 자동차 산업에 관심을 가지고 있다.

앞으로 펼쳐질 자율주행이 완벽하게 상용화된다면 자동차 시장은 혁명을 맞이할 것이고 스마트폰이 우리의 삶을 바꾸었듯 자동차 사용자들의 생활도 바뀔 전망으로 보고 있다.

그리하여 우리가 진행할 프로젝트에서는 자동차가 각광받는 시대의 흐름에 맞춰 인간의 조작 없이 스스로 자율주행이 가능하여 몸이 불편하여 운전을 하지 못하는 사람들이나 무인 배달 서비스, 무인 택시, 차량 대여 서비스 등 사람들이 각종 편리한 시스템을 사용할 수 있는 자율주행 스마트카와 그에 맞는 트랙을 제작하여 스마트카의 자율주행을 구현해보는 것이 이 프로젝트의 목표이다.

## 3. 프로젝트의 설계 구상

우리가 진행하려는 프로젝트의 기술 구상에 대해서 간단하게 설명해보려고 한다.

자율주행 스마트카는 기본적으로 아두이노 우노 호환 보드를 사용하여 프로젝트를 진행할 것이다.

초음파 센서를 통해 장애물을 감지할 수 있고, 적외선 리모컨 수신 모듈, 적외선 리모컨 등으로 원격제어가 가능하다.

그뿐만 아니라 블루투스 모듈HC-06을 사용하여 안드로이드 기반의 스마트폰으로도 원격제어가 가능하도록 설계할 것이다.

자율주행 작동에 필요한 DC모터 드라이버, 서브모터 SG90을 기본적으로 사용할 것이며, 원활한 주행을 위해 바퀴4개를 사용할 것이고, 각 바퀴에 DC모터 4개를 장착 후 작동을 하려고 한다.

또한 라인트레이서 모듈 센서를 사용하여 적외선 감지 원리를 파악하고, 자율주행 스마트카가 바닥에 그려진 라인을 따라서 주행을 가능하도록 코드를 구성해보도록 할 것이다.

하지만 그전에 자율주행을 위한 스마트카의 프레임은 먼저 완성시켜야 하는데 우리 조는 상부 바디 프레임과 하부 바디 프레임을 기본 뼈대 구조로 이용할 예정이다.

상부 바디 프레임에는 초음파 센서, 아두이노 보드, 배터리 홀더등의 부품들과 결합해줄 구멍을 만들어 케이블을 통해 연결해줄 것이다.

그 다음 하부 바디 프레임에는 라인트레이서, 상하부 연결 서포트, 모터 등의 부품들과 결합해줄 구멍을 만들어 케이블을 통해 연결해줄 것이다.

바디프레임 조립을 완성하면 각 부품들에 대한 간단한 코드를 구현하여 구동테스트를 진행하면서 완성된 스마트카 프레임이 전체적으로 정상 작동되는지 확인한다.

마지막으로 자율주행 스마트카라는 컨셉에 맞게 자율주행에 대해서 좀더 심도있는 작품을 만들기 위하여 트랙을 만들어 그 위에 스마트카가 자율주행이 가능하게 만들어 볼 것이다.

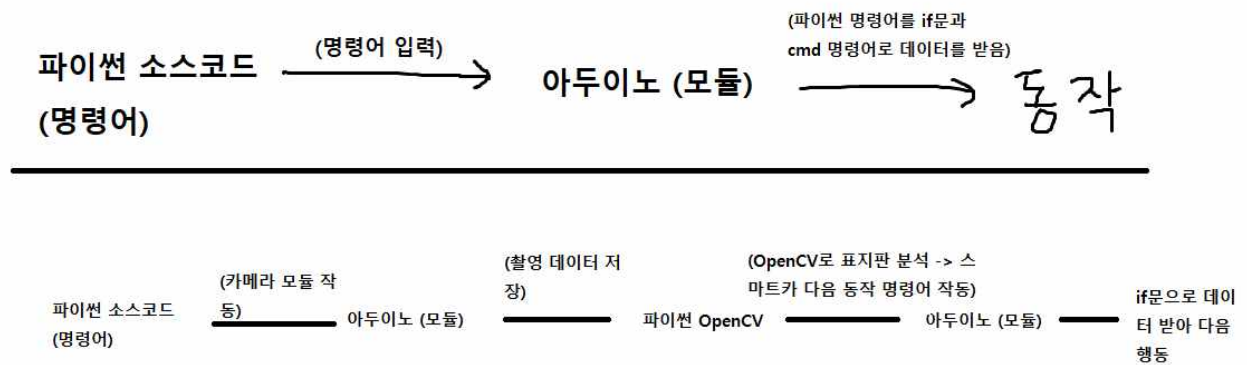
트랙에서 구현해 볼 자율주행 테스트로는 라인트레이서를 활용하여 차선유지 및 곡선 주행, 자

을 주차 가능 등을 구현해볼 것이며 그뿐만 아니라 opencv 파이썬과 ESP32 CAM OV2640 카메라 4M PSRAM 장착 보드를 활용하고 led센서를 통해 신호등을 구현하여 빨간불에는 멈추고 초록불에 주행을 하도록 해볼 것이다.

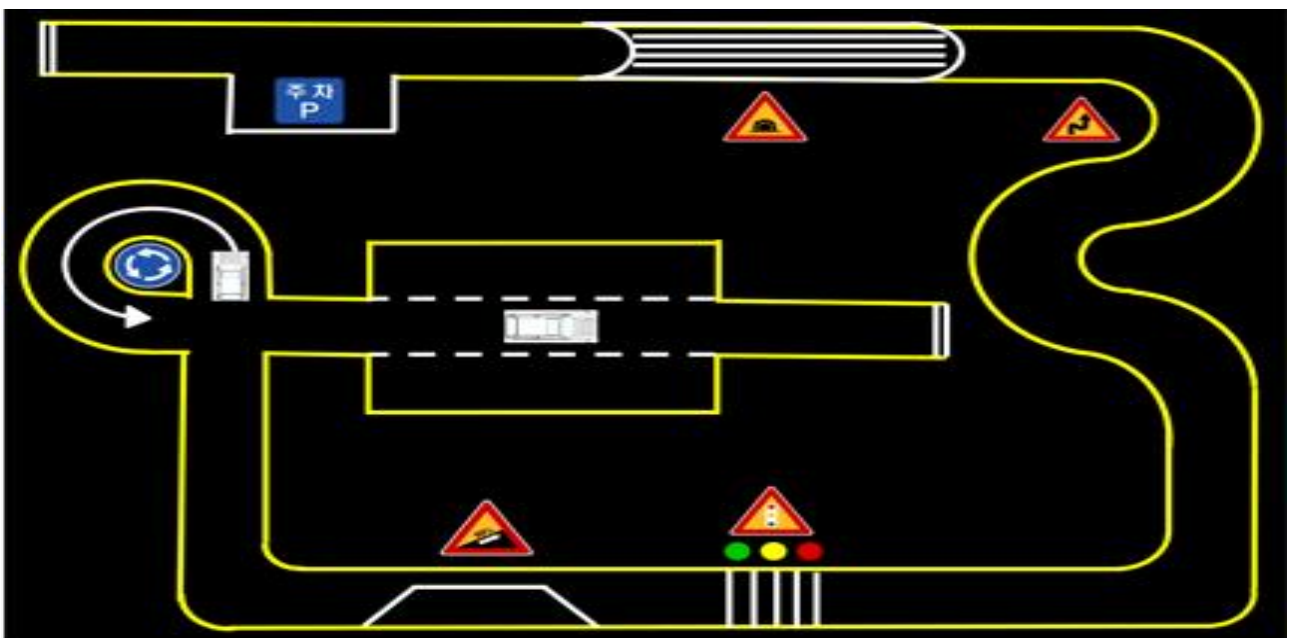
그리고 이미지 인식을 활용하여 표지판 등을 인식해 표지판이 의미하는 것을 자동적으로 인식하여 주행하도록 설계해보도록 할 예정이다.

기본적으로 파이썬-아두이노 연동제어를 통해 결과를 도출해볼 예정이다.

### [파이썬-아두이노 개발환경 분석]



### [자율주행 트랙 구상도]

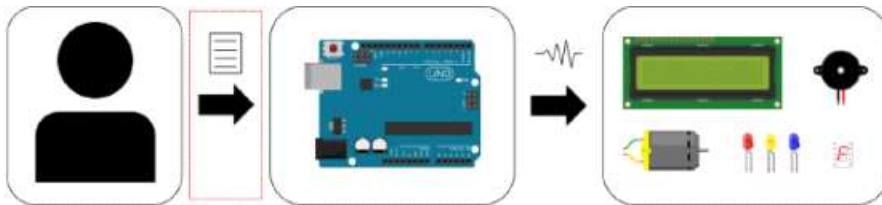


## 2 . 프로젝트 설계서

### 1. 프로젝트의 시스템 환경

-기본적으로 아두이노 우노 호환 보드를 사용하고, 아두이노 프로그램 IDE(Integrated Development Environment)를 설치해준다.

-IDE는 소프트웨어 개발을 위한 통합 개발 환경으로 아두이노 환경에서 개발할 때, 프로그래밍, 컴파일, 업로드 등의 과정을 하나의 화면에서 작업할 수 있도록 제공하는 개발 툴이다.



아두이노 호환보드를 사용하기 위해서는 사용하는 PC에 전용 드라이버를 설치해야 아두이노 IDE에서 아두이노 보드를 인식할 수 있기 때문에 꼭 아두이노 PC전용 드라이버를 설치해주도록 한다.

### 2. 프로젝트 준비물 및 기능



우노 호환 보드  
수량: 1개



우노 호환 보드  
수량: 1개

▶프로젝트를 진행하는데 제일 중요하고 기본적인 준비물로 PC에 아두이노 프로그램 IDE를 설치하여 개발 및 프로그래밍, 컴파일, 업로드를 한다.



미니 브레드보드  
수량: 1쌍



미니 브레드보드  
수량: 1쌍

▶아두이노와 케이블을 연결해주는 매개체

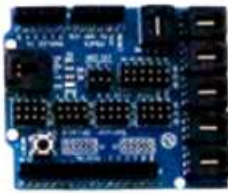


우노용 USB 케이블  
수량: 1개



우노용 USB 케이블  
수량: 1개

▶아두이노 우노용 USB B케이블을 아두이노 우노 보드의 USB B소켓에 연결해주고, 아두이노 우노 보드와 연결된 아두이노 우노용 USB B케이블의 반대쪽 부분을 컴퓨터 본체의 USB 소켓에 연결해준다.



센서확장 쉴드  
수량: 1개

▶아두이노 우노에 적층하여 사용하는 확장형 쉴드로 센서들을 확장하여 사용하는 용도로 사용된다.

우리가 프로젝트에 사용하는 V4.0은 센서 확장 쉴드 4번째 버전으로 브레드보드 없이 쉽게 각종 센서 등 부품을 plug&play 형식으로 연결이 가능하다.



바퀴  
수량: 4개

▶자율주행 스마트카의 원활한 주행을 위해 4개의 바퀴를 사용한다.



DC모터  
수량: 4개

▶각 바퀴를 구동시킬 DC모터 4개





서보모터 SG90  
수량: 1개

▶모터와 제어 구동 보드를 구축하고 위치, 속도를 명령으로 제어하는 센서이다.  
PMW 제어로 180도, 360도까지 제어가 가능하다.



초음파 센서  
수량: 1개

▶사용하기 가장 쉽고 보편적인 HC-SR04모델을 사용하였으며 본 프로젝트에서는 거리를 측정하고, 장애물을 감지하는 역할로 사용하려고 한다.



라인트레이서 센서  
수량: 3개

▶우리가 사용하는 라인트레이서의 명칭은 ‘TCRT5000’으로 적외선 빛을 송수신하여 흰색과 검정색을 구분하는 센서이다.  
인식거리는 12mm로, 자율주행 스마트카가 바닥에 그려진 라인을 따라서 주행이 가능하도록 해주는 역할로 사용할 것이다.



적외선 리모컨  
수량: 1개

▶적외선 신호를 만들어 방출하는 포토 트랜지스터와 적외선 신호를 수신하는 수광 다이오드가 한 쌍이 되어 작동하고, 자율주행 스마트카를 원격제어 해주는 역할로 사용할 것이다.



적외선 리모컨 수신 모듈  
수량: 1개

▶적외선 리모컨을 수신하기 위해 필요한 모듈



DC모터 드라이버  
수량: 1개

▶모터 드라이버 'L298N'은 스마트카의 바퀴를 움직이는 모터의 회전 방향이나 회전 속도를 조절하기 위해 필요한 장치이다.

아두이노 보드를 활용해서 직접 모터속도, 회전 방향을 조절할 수 있지만, 사용하는 모터가 요구하는 전원을 공급할 수 없어 모터 드라이버 'L298N'을 사용해야 한다.



바디 프레임  
수량: 1set

▶자율주행 스마트카의 뼈대가 될 바디 프레임



초음파 센서 브라켓  
수량: 1set

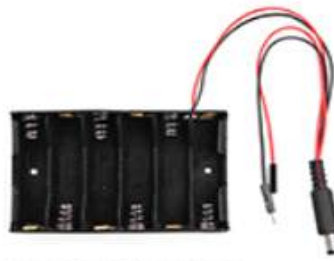


DC 모터 브라켓  
수량: 4개

▶초음파 센서, DC모터 브라켓



9V 배터리 홀더  
수량: 1개



1.5V 배터리 홀더  
수량: 1개



볼트 / 너트M3



▶배터리 홀더, 볼트/너트, 케이블 등 각종 부자재



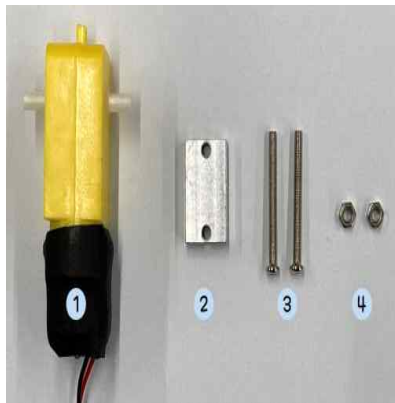
▶아두이노 3색 RGB LED 신호등 모듈



▶ESP32 CAM OV2640 카메라 4M PSRAM 장착 보드

### 3 . Use-case

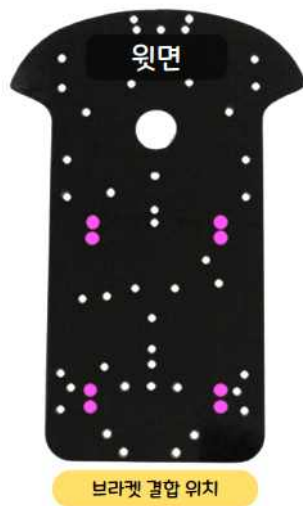
#### 1. 하부 바디 프레임 결합순서 및 센서위치



- 1 DC모터
- 2 DC모터 브라켓
- 3 M3 30mm 볼트 2개
- 4 M3 너트 2개



▶DC모터와 DC모터 브라켓을 조립하기 위해 필요한 부품들을 준비후 결합해준다.



브라켓 결합 위치

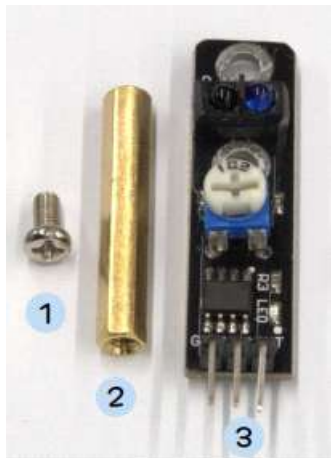


선 정리 구멍 위치



모터 4개, 10mm 볼트 8개

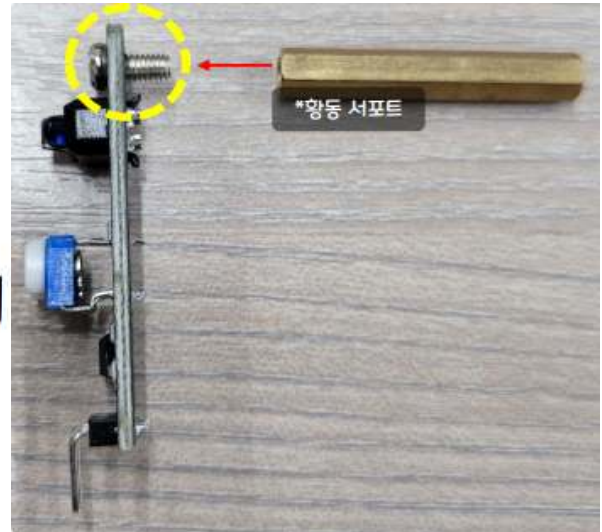
▶하부 바디 프레임과 DC모터 4개를 10mm 볼트를 이용해 결합위치를 확인 후 결합해준다.



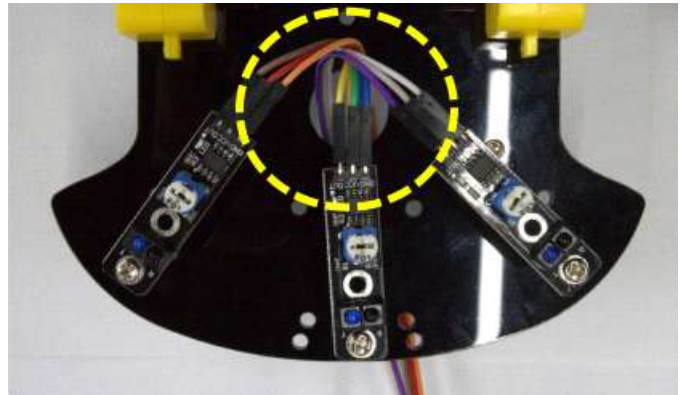
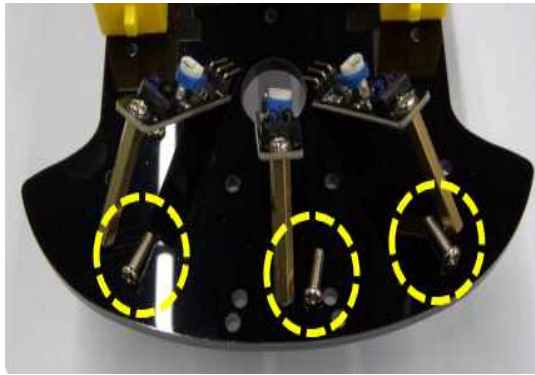
1 M3 6mm 볼트 3개

2 황동 서포트 30mm 3개

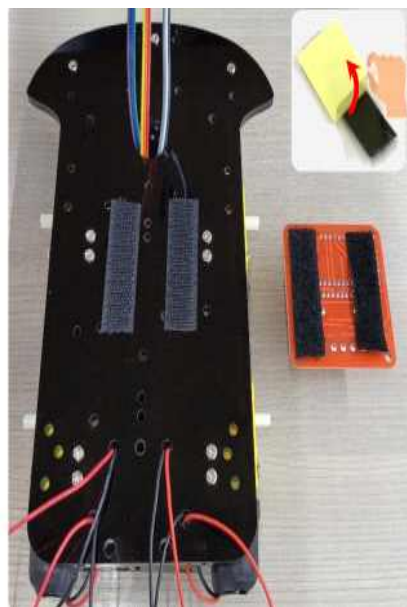
3 라인트레이서 3개



▶라인트레이서 조립에 필요한 부품들을 준비 후 황동 서포트를 6mm 볼트로 사진과 같이 결합해준다.

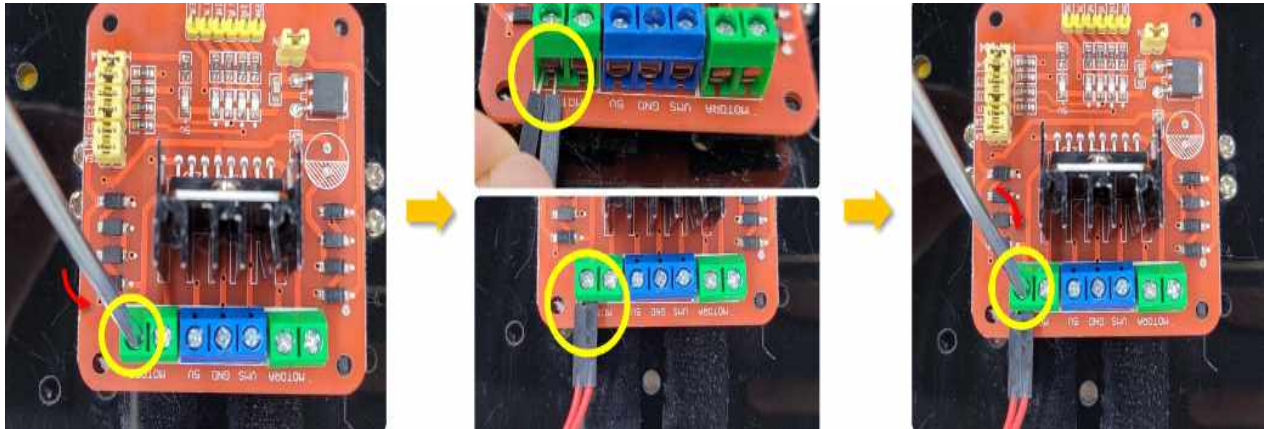


▶조립을 마친 라인트레이서 3개를 10mm 볼트를 이용하여 하부 바디프레임에 결합 후 라인 트레이서에 케이블을 결선한 뒤, 정리해준다.

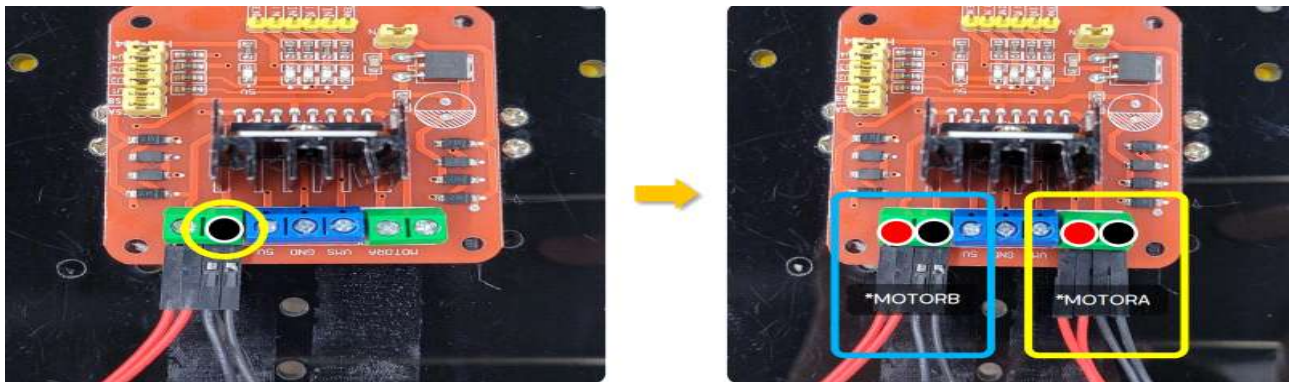


▶DC모터 드라이버, 하부 바디 프레임, 벨크로를 준비 후 프레임과 드라이버에 벨크로를 부착해준 다음 결합해준다.

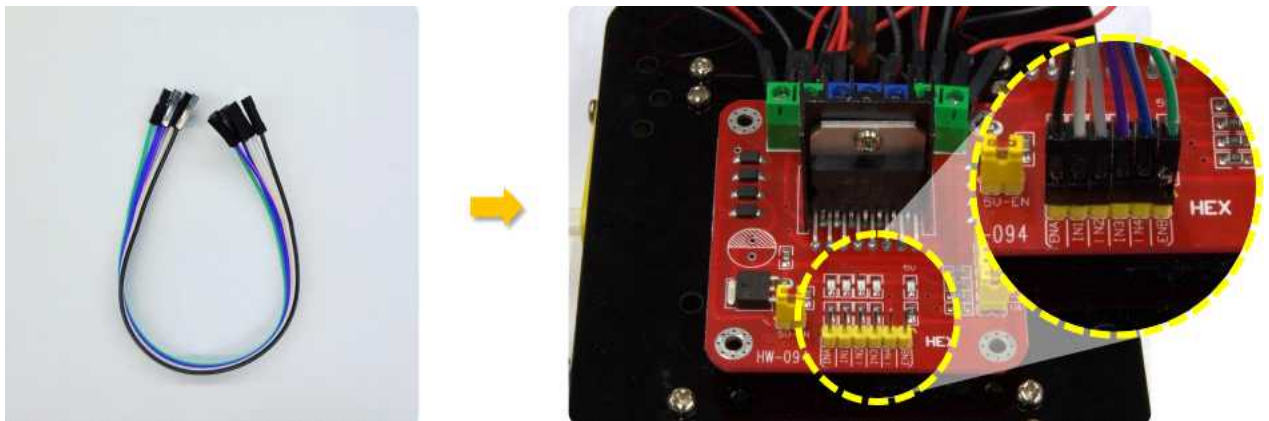




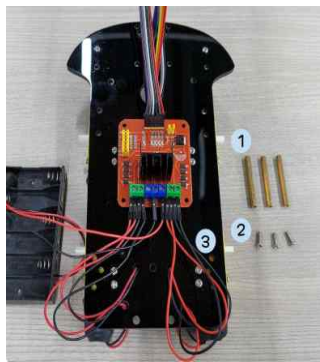
▶드라이버를 화살표 방향으로 돌려 볼트를 풀어준 후 왼쪽 모터 빨간색 선 2개르 모아서 커넥터에 연결해준다. 그 다음 풀어준 볼트를 다시 조여준다.



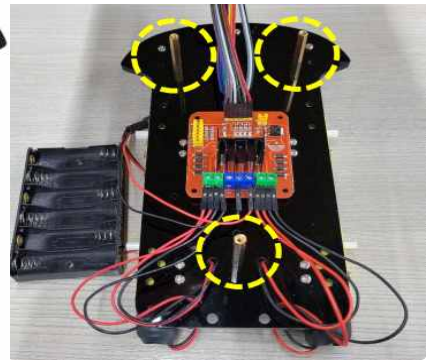
▶동일한 방법으로 검은색 선 2개를 DC모터 드라이버에 결선해준다.



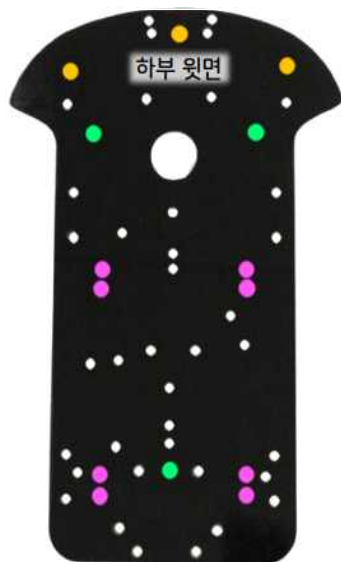
▶모터 제어핀 연결 부분과 케이블을 결선해준다. (여기까지가 DC모터 드라이버 조립완성)



- 1 황동 서포트 40mm 3개
- 2 M3 10mm 볼트 3개
- 3 하부 바디 프레임



▶결합에 필요한 부품들을 준비 후 서포트 결합 위치를 확인하여 하부 바디프레임 위에 청동 서포터를 올리고, 아래에서 볼트로 조여준다.

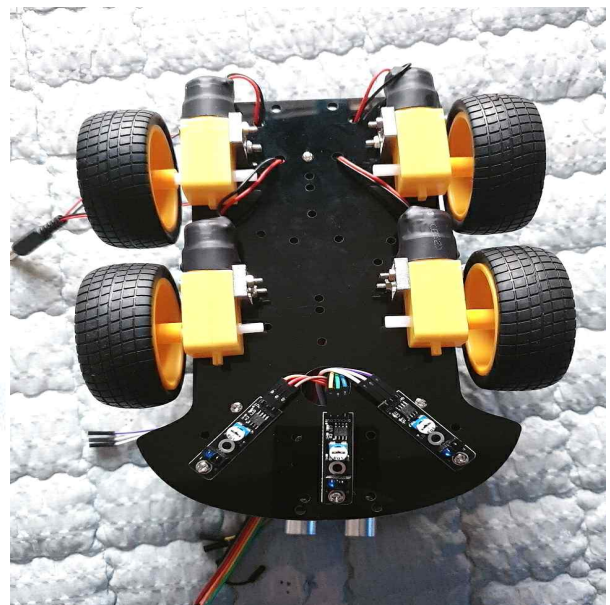


하부 바디 프레임

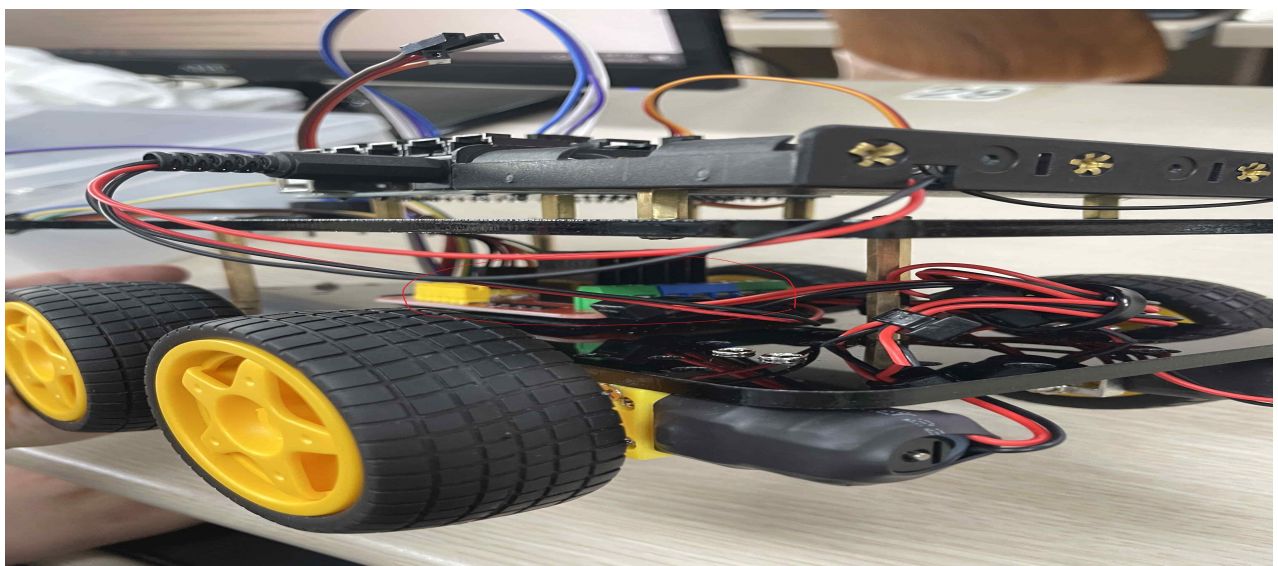
라인트레이서

상하부 연결 서포트

모터



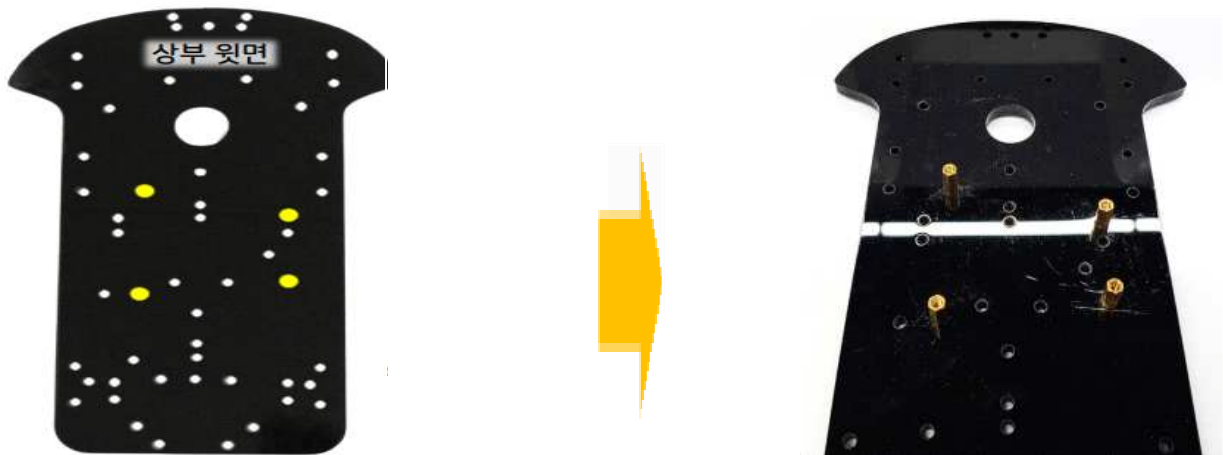
▶하부 바디 프레임에 라인트레이서, 상하부 연결 서포트, 모터를 케이블과 연결하여 완성된 하부 바디 프레임의 모습이다. (DC모터 드라이버는 하부, 상부 바디프레임 중간 결합부 부분에 감춰진 모습)





## 2. 상부 바디 프레임 센서위치 및 결합

[결합을 완료한 모습]



▶ 먼저 아두이노 우노 보드, M3 10mm볼트 4개, 황동 서포트 10mm 4개를 준비해준다.  
그 다음 노란색으로 표시된 부분이므로 아두이노 보드 결합 위치로 이곳에 볼트-바디 프레임- 황동 서포트 순으로 결합해준다.



▶ 황동 서포트 위에 아두이노를 올리고 6mm 볼트 4개로 서포트-아두이노-볼트 순으로 아두이노 보드를 상부 바디 프레임에 결합한다.

[확장섀드 결합을 완료한 모습]



▶ 아두이노 보드 위에 확장 섀드를 구멍에 핀을 올바르게 하여 결합해준다.





- 1 서보 모터
- 2 아래 브라켓
- 3 중심축 볼트
- 4 브라켓 내 동봉된 볼트, 너트

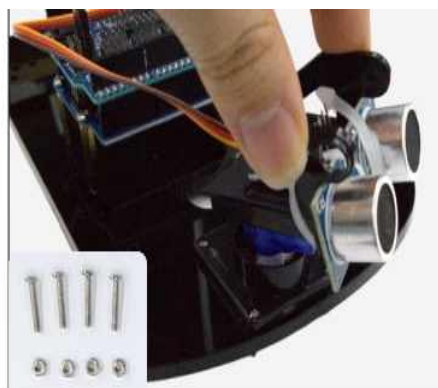


▶서보모터 조립에 필요한 부품들을 준비 후, 화살표 방향으로 서보모터와 아래 브라켓을 놓는다. 그 다음 파란색 동그라미 부분 아래 브라켓의 표시된 곳과 서보모터 축을 결합해준다.

[서브모터와 아래브라켓 결합된 모습]



▶중심축 볼트를 결합하여 서보모터와 아래 브라켓을 고정시켜준다.



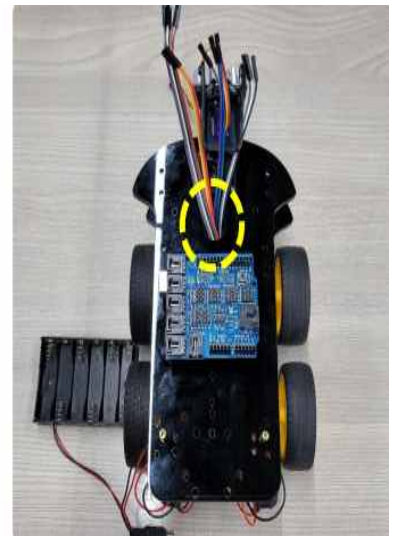
▶긴 볼트4개, 너트4개를 준비 후 상부 바디프레임 결합 위치를 확인한다. 확인 후 프레임 아래에서 너트를 조여 서보모터 브라켓을 고정 시켜준다.

1 황동 서포트 10mm 2개

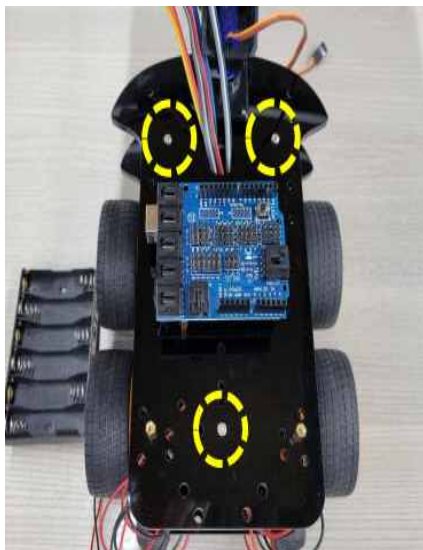
2 M3 10mm 볼트 2개



▶배터리 홀더 장착을 위해 서포트와 볼트를 준비 후 결합 위치를 확인해준다.  
확인한 결합 위치에 볼트-바디프레임-서포트 순으로 결합해준다.

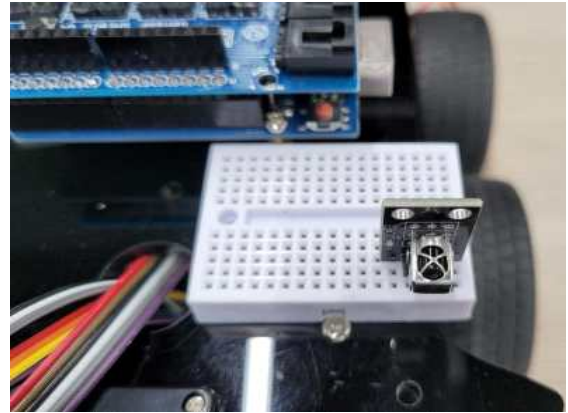
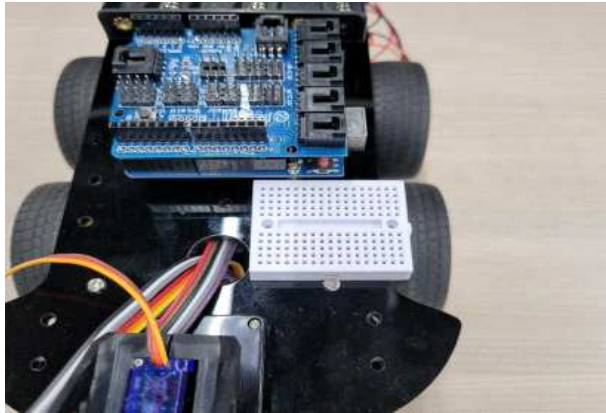


▶하부 바디 프레임과 상부 바디프레임을 조립하기 전, 위 사진에 표시한 상부 바디 프레임 구멍에 케이블을 빼준다.

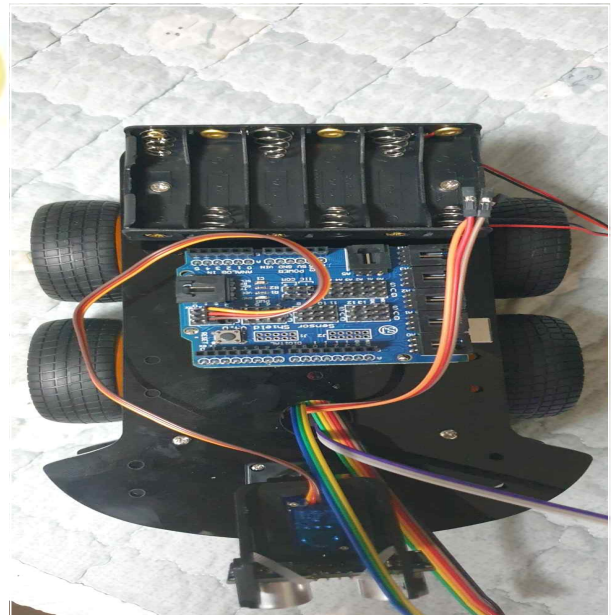




▶다음은 하부 서포트 위에 상부 바디 프레임을 얹고 10mm 볼트 3개를 통해 결합해준다. 그리고 둥근 납작머리 볼트를 사용하여 서포트와 배터리 홀더를 조립해준다.



▶마지막으로 미니브레드 보드를 붙여준 후 보드에 적외선 수신 센서를 결합해준다.



▶상부 바디 프레임에 초음파 센서, 아두이노 보드, 배터리 홀더를 케이블과 결선하여 완성된 모습이다.

## 4 . 프로젝트 소스코드

### 1. 서보모터 구동 실습코드

```
#include <Servo.h>
Servo myservo;
void setup() {
  myservo.attach(2);
}
void loop() {
  myservo.write(0);
  delay(3000);
  myservo.write(90);
  delay(3000);
  myservo.write(180);
  delay(3000);
}
```

### 1. 서보모터 구동 실습코드(2)

```
#include <Servo.h>
Servo servo;
void setup() {
  servo.attach(2);
  Serial.begin(9600);
  servo.write(90);
  delay(1000);
}
void loop() {
}
```

### 2. 초음파 센서 실습코드

```
int trigPin = 13;
int echoPin = 12;
void setup() {
  Serial.begin(9600); // 시리얼 속도 설정
  pinMode(echoPin, INPUT); // echoPin 입력
  pinMode(trigPin, OUTPUT); // trigPin 출력
}
void loop() {
  long duration, distance;
  digitalWrite(trigPin, HIGH); // trigPin에서 초음파 발생(echoPin
도 HIGH)
  delayMicroseconds(10);
```

```

    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);           // echoPin 이 HIGH를 유지한 시간을
저장 한다.
    distance = ((float)(340* duration)/ 10000)/ 2;
    Serial.print("distance:");
    Serial.print(distance);                     // 물체와 초음파 센서간 거리를 표시
    Serial.println(" cm");
    delay(500);
}

```

### 3. 라인트레이서 구동 실습코드

```

void setup() {
    Serial.begin(9600);
}
void loop() {
    intval_1 = analogRead(A3);
    intval_2 = analogRead(A4);
    intval_3 = analogRead(A5);

    Serial.print("1:");
    Serial.print(val_1);
    Serial.print(" 2:");
    Serial.print(val_2);
    Serial.print(" 3:");
    Serial.println(val_3);
}

```

### 4. DC모터 드라이버 구동 실습코드

```

intRightMotor_E_pin = 5;    // 오른쪽 모터의 Enable & PWM
intRightMotor_1_pin = 8;    // 오른쪽 모터 제어선 IN1
intRightMotor_2_pin = 9;    // 오른쪽 모터 제어선 IN2
intLeftMotor_3_pin = 10;    // 왼쪽 모터 제어선 IN3
intLeftMotor_4_pin = 11;    // 왼쪽 모터 제어선 IN4
intLeftMotor_E_pin = 6;    // 왼쪽 모터의 Enable & PWM
void SmartCar_Go();
void SmartCar_Back();
//좌우 모터 속도 조절, 설정 가능 최대 속도 : 255

```

```

intL_MotorSpeed = 153; // 왼쪽 모터 속도
intR_MotorSpeed = 153; // 오른쪽 모터 속도
void setup() {
    pinMode(RightMotor_E_pin, OUTPUT); // 출력모드로 설정
    pinMode(RightMotor_1_pin, OUTPUT);
    pinMode(RightMotor_2_pin, OUTPUT);
    pinMode(LeftMotor_3_pin, OUTPUT);
    pinMode(LeftMotor_4_pin, OUTPUT);
    pinMode(LeftMotor_E_pin, OUTPUT);
}
void loop() {
    SmartCar_Go();
    delay(3000);
    SmartCar_Back();
    delay(3000);
}

void SmartCar_Go() { // 전진
    Serial.println("Forward");
    digitalWrite(RightMotor_1_pin, HIGH);
    digitalWrite(RightMotor_2_pin, LOW);
    digitalWrite(LeftMotor_3_pin, HIGH);
    digitalWrite(LeftMotor_4_pin, LOW);
    analogWrite(RightMotor_E_pin, L_MotorSpeed);
    analogWrite(LeftMotor_E_pin, R_MotorSpeed);
}

void SmartCar_Back() { // 후진
    Serial.println("Backward");
    digitalWrite(RightMotor_1_pin, LOW);
    digitalWrite(RightMotor_2_pin, HIGH);
    digitalWrite(LeftMotor_3_pin, LOW);
    digitalWrite(LeftMotor_4_pin, HIGH);
    analogWrite(RightMotor_E_pin, L_MotorSpeed);
    analogWrite(LeftMotor_E_pin, R_MotorSpeed);
}

```

## 5. 시리얼 통신 제어 (직진, 후진)

```

int RightMotor_E_pin = 5 ; // 오른쪽 모터의 Enable &PWM
int LeftMotor_E_pin = 6 ; // 왼쪽 모터의 Enable &PWM
int RightMotor_1_pin = 8 ; // 오른쪽 모터 제어선 IN1

```

```

int RightMotor_2_pin = 9 ;    // 오른쪽 모터 제어선 IN2
int LeftMotor_3_pin = 10 ;    // 왼쪽 모터 제어선 IN3
int LeftMotor_4_pin = 11 ;    // 왼쪽 모터 제어선 IN4
//좌우 모터 속도 조절, 설정 가능 최대 속도 : 255
int L_MotorSpeed = 153 ; // 왼쪽 모터 속도
int R_MotorSpeed = 153 ; // 오른쪽 모터 속도
void setup (){
    pinMode (RightMotor_E_pin, OUTPUT );    // 출력모드로 설정
    pinMode (RightMotor_1_pin, OUTPUT );
    pinMode (RightMotor_2_pin, OUTPUT );
    pinMode (LeftMotor_3_pin, OUTPUT );
    pinMode (LeftMotor_4_pin, OUTPUT );
    pinMode (LeftMotor_E_pin, OUTPUT );
    Serial .begin (9600 );
    Serial .println ("Welcome");
}
void loop (){
    if (Serial .available ()){
        char command = Serial .read ();
        Serial .print ("Recived command : ");
        if (command == 'g'){
            motor_role (HIGH, HIGH );
            Serial .println ("직진");
        }
        else if (command == 'b'){
            motor_role (LOW, LOW );
            Serial .println ("후진");
        }
        else {
            Serial .println ("Wrong command");
        }
    }
}
void motor_role (int R_motor , int L_motor){
    digitalWrite (RightMotor_1_pin, R_motor );
    digitalWrite (RightMotor_2_pin, !R_motor );
    digitalWrite (LeftMotor_3_pin, L_motor );
    digitalWrite (LeftMotor_4_pin, !L_motor );
    analogWrite (RightMotor_E_pin, R_MotorSpeed );
    analogWrite (LeftMotor_E_pin, L_MotorSpeed );
}

```

## 6. 초음파센서를 활용한 장애물 회피 자율주행 테스트

```

#include <Servo.h>
Servo Servo;
//출력핀(trig)과 입력핀(echo) 설정
int trigPin = 13;           // 디지털 13번 핀에 연결
int echoPin = 12;           // 디지털 12번 핀에 연결
int Ultra_d = 0;
int val = 0;                // 좌우 경로 설정 변수

```

```

intRightMotor_E_pin = 5;      // 오른쪽 모터의 Enable & PWM
intLeftMotor_E_pin = 6;      // 왼쪽 모터의 Enable & PWM
intRightMotor_1_pin = 11;    // 오른쪽 모터 제어선 IN1
intRightMotor_2_pin = 10;    // 오른쪽 모터 제어선 IN2
intLeftMotor_3_pin = 9;      // 왼쪽 모터 제어선 IN3
intLeftMotor_4_pin = 8;      // 왼쪽 모터 제어선 IN4
//좌우 모터 속도 조절, 설정 가능 최대 속도 : 255
intL_MotorSpeed = 153; // 왼쪽 모터 속도
intR_MotorSpeed = 153; // 오른쪽 모터 속도
void setup() {
    Servo.attach(2);          // 서보모터 PWM 디지털입출력 2번핀 연결

    pinMode(echoPin, INPUT);  // echoPin 입력
    pinMode(trigPin, OUTPUT); // trigPin 출력

    pinMode(RightMotor_E_pin, OUTPUT); // 출력모드로 설정
    pinMode(RightMotor_1_pin, OUTPUT);
    pinMode(RightMotor_2_pin, OUTPUT);
    pinMode(LeftMotor_3_pin, OUTPUT);
    pinMode(LeftMotor_4_pin, OUTPUT);
    pinMode(LeftMotor_E_pin, OUTPUT);
    Serial.begin(9600);        // PC와의 시리얼 통신 9600bps로 설정
    Serial.println("Welcome!");
}
void loop() {
    Ultra_d = Ultrasonic();
    Serial.println(Ultra_d);
    motor_role(HIGH, HIGH); // 직진
    if(Ultra_d < 250) {
        if (Ultra_d < 150) {
            Serial.println("150 이하.");
            motor_role(LOW, LOW); // 후진
            delay(1000);
            analogWrite(RightMotor_E_pin, 0);
            analogWrite(LeftMotor_E_pin, 0);
            delay(200);
        }
        else {
            analogWrite(RightMotor_E_pin, 0);
            analogWrite(LeftMotor_E_pin, 0);
            delay(200);
            Serial.println("150 이상.");
        }
    }
}

```



```

val = Servo_con();
if (val == 0) {
    Serial.println("우회전.");
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    delay(500);
    motor_role(LOW, LOW); // 후진
    delay(500);
    motor_role(LOW, HIGH); // 우회전
    delay(800);
}
else if (val == 1) {
    Serial.println("좌회전.");
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    delay(500);
    motor_role(LOW, LOW); // 후진
    delay(500);
    motor_role(HIGH, LOW); // 좌회전
    delay(800);
}
}
}
}

void motor_role(int R_motor, int L_motor){
    digitalWrite(RightMotor_1_pin, R_motor);
    digitalWrite(RightMotor_2_pin, !R_motor);
    digitalWrite(LeftMotor_3_pin, L_motor);
    digitalWrite(LeftMotor_4_pin, !L_motor);

    analogWrite(RightMotor_E_pin, R_MotorSpeed); // 우측 모터 속도값
    analogWrite(LeftMotor_E_pin, L_MotorSpeed); // 좌측 모터 속도값
}

int Ultrasonic(){
    long duration, distance;
    digitalWrite(trigPin, HIGH); // trigPin에서 초음파 발생(echoPin도 HIGH)
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH); // echoPin 이 HIGH를 유지한 시간을 저장 한다.
    distance = ((float)(340* duration)/ 1000)/ 2;
    //Serial.print("Distance:"); // 물체와 초음파 센서간 거리를 표시.
    //Serial.println(distance);
}

```

```

    returndistance;
}
int Servo_con(){
    Servo.write(30);
    delay(300);
    intUlt_30 = Ultrasonic();
    delay(700);
    Servo.write(150);
    delay(300);
    intUlt_150 = Ultrasonic();
    delay(700);
    if(Ult_30 > Ult_150){
        val = 1;
    }
    else{
        val = 0;
    }
    Servo.write(90);

    returnval;
}

```

## 7. led센서를 활용하여 제작한 신호등 작동 테스트

```

#define LED_R 3      // R의 핀 번호 3
#define LED_Y 4      // Y의 핀 번호 4
#define LED_G 5      // G의 핀 번호 5

void setup() {
    // put your setup code here, to run once:
    pinMode(LED_R, OUTPUT);
    pinMode(LED_Y, OUTPUT);
    pinMode(LED_G, OUTPUT);
}

void loop() {
    turnOffAll();                // LED 다 끕니다.

    digitalWrite(LED_R, HIGH);  // 빨간 불 켜기
    delay(3000);                 // 3초간 유지
    turnOffAll();                // 불 다 끄기
}

```

```

digitalWrite(LED_G, HIGH);    // 초록 불 켜기
delay(3000);                  // 3초간 유지
turnOffAll();                 // 불 다 끄기

digitalWrite(LED_Y, HIGH);    // 노란 불 켜기
delay(2000);                  // 2초간 유지
turnOffAll();                 // 불 다 끄기

}

void turnOffAll() {           // turnOffAll 함수 정의
    digitalWrite(LED_R, LOW); // 빨간 불 끄기
    digitalWrite(LED_Y, LOW); // 노란 불 끄기
    digitalWrite(LED_G, LOW); // 초록 불 끄기
}

```

## 8. 파이썬-아두이노 제어를 통한 시리얼 통신테스트

[파이썬]

```

import serial
arduino = serial.Serial('COM4',9600)

```

```

while(1):
    c=input()
    if c =='q':
        break
    else:
        c=c.encode('utf-8')
        arduino.write(c)

```

[아두이노]

```

#include <Servo.h>
Servo myservoH; // create servo object to control a servo
Servo myservoV;
int posH = 90 ;
int posV = 90 ;
void setup (){
    myservoH .attach (2 ); // 서보모터를 쉼트의 2번에 연결한다.
    Serial .begin (9600 );
}
void loop (){
    while (Serial .available () >0 ){
        long value = Serial .parseInt ();
        int backUpH = posH;
        int backUpV = posV;
        switch (value ){
            case 1 :

```

```

    //posH = posH - 5; 같은 표현이다 posH -= 5;
    posH -= 30 ;
    break ;
case 2 :
    posH += 30 ;
    break ;
case 3 :
    posV -= 30 ;
    break ;
case 4 :
    posV += 30 ;
    break ;
}
if (posH<0 || posH>180 ){
    posH=backUpH;
}
if (posV<0 || posV>180 ){
    posV=backUpV;
}
myservoH .write (posH );
myservoV .write (posV );
}
}

```

## 9. 라인트레이서, 초음파센서 코드 (if, else if문)

```

#include <Servo.h>                // 서보 라이브러리
Servo Servo;
//출력핀(trig)과 입력핀(echo)
inttrigPin = 13;                  // 디지털 13번 핀에 연결
intechoPin = 12;                  // 디지털 12번 핀에 연결
intUltra_d = 0;                  // 초음파 센서
intval = 0;                       // 좌우 경로 설정 변수

intRightMotor_E_pin = 5;          // 오른쪽 모터 전원부 5번 연결핀
intLeftMotor_E_pin = 6;           // 왼쪽 모터 전원부 6번 연결핀
intRightMotor_1_pin = 8;          // 오른쪽 모터 8번 IN1
intRightMotor_2_pin = 9;          // 오른쪽 모터 9번 IN2
intLeftMotor_3_pin = 10;          // 왼쪽 모터 10번 IN3
intLeftMotor_4_pin = 11;          // 왼쪽 모터 11번 IN4
//모터 속도 조절, 최대속도 : 255
intL_MotorSpeed = 153;// 왼쪽 모터 속도
intR_MotorSpeed = 153;// 오른쪽 모터 속도
intL_Line = A5;// 왼쪽 라인트레이서 센서 - 확장설프드 A5
intC_Line = A4;// 가운데 라인트레이서 센서 - 확장설프드 A4
intR_Line = A3;// 오른쪽 라인트레이서 센서 - 확장설프드 A3

```

```

intSL = 1;
intSC = 1;
intSR = 1;
// 부품 셋업
void setup() {
    Servo.attach(2); // 서보모터 PWM 디지털입출력 2번핀 연결
    pinMode(echoPin, INPUT); // echoPin 입력
    pinMode(trigPin, OUTPUT); // trigPin 출력
    // 모터 핀 설치
    pinMode(RightMotor_E_pin, OUTPUT); // 오른쪽 모터 전원부 출력
    pinMode(LeftMotor_E_pin, OUTPUT); // 왼쪽 모터 전원부 출력
    pinMode(RightMotor_1_pin, OUTPUT); // 오른쪽 모터 앞바퀴 출력
    pinMode(RightMotor_2_pin, OUTPUT); // 오른쪽 모터 뒷바퀴 출력
    pinMode(LeftMotor_3_pin, OUTPUT); // 왼쪽 모터 앞바퀴 출력
    pinMode(LeftMotor_4_pin, OUTPUT); // 왼쪽 모터 뒷바퀴 출력
    Serial.begin(9600); // PC와의 시리얼 통신 값 9600
    Serial.println("Activate!"); // 통신에 성공했을 경우, 시리얼 모니터 창에 Activate! 출
    력
}
// 라인트레이서 값 읽기
void loop() {
    intL = digitalRead(L_Line);
    intC = digitalRead(C_Line);
    intR = digitalRead(R_Line);

    Serial.print("digital : ");
    Serial.print(L);
    Serial.print(", ");
    Serial.print(C);
    Serial.print(", ");
    Serial.print(R);
    Serial.print(" ");

    // 라인트레이서 인식 세팅
    if (L == LOW && C == LOW && R == LOW ) { // 0 0 0
        L = SL; C = SC; R = SR;
    }
    // 라인트레이서에서 센터만 인식이 됐을경우 직진,
    if (L == LOW && C == HIGH && R == LOW ) { // 0 1 0
        motor_role(HIGH, HIGH);
        Serial.println("직진");
    }
}

```

```

// 라인트레이서에서 오른쪽만 인식이 됐을경우 우회전,
else if (L == LOW && R == HIGH ){                                // 0 0 1, 0 1 1
    motor_role(LOW, HIGH);
    Serial.println("우회전");
}

// 라인트레이서에서 왼쪽만 인식이 됐을 경우 좌회전,
else if (L == HIGH && R == LOW ) {                                // 1 0 0, 1 1 0
    motor_role(HIGH, LOW);
    Serial.println("좌회전");
}

// 라인트레이서에서 왼쪽 오른쪽 전부 인식이 됐을경우, 정지
else if (L == HIGH && R == HIGH ) {                                // 1 1 1, 1 0 1
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    Serial.println("정지");
}

// 전방에 15cm 안에 물체가 감지됐을 경우 후진 후 정지,
else if (Ultra_d < 150) {
    Serial.println("150 이하.");
    motor_role(LOW, LOW); // 후진
    Serial.println("물체 감지! 정지 정지 정지");
    delay(1000);
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    delay(200);
}
SL = L; SC = C; SR = R;
}

// 디지털방식으로 모터의 출력 상태 0, 1로 기록
void motor_role(int R_motor, int L_motor) {
    digitalWrite(RightMotor_1_pin, R_motor);
    digitalWrite(RightMotor_2_pin, !R_motor);
    digitalWrite(LeftMotor_3_pin, L_motor);
    digitalWrite(LeftMotor_4_pin, !L_motor);
    // 아날로그 방식으로 모터의 출력 상태 0 ~ 255 기록
    analogWrite(RightMotor_E_pin, R_MotorSpeed); // 우측 모터 속도값 기록
    analogWrite(LeftMotor_E_pin, L_MotorSpeed); // 좌측 모터 속도값 기록
}

```

```

// 초음파 센서 작동
int Ultrasonic(){
    long duration, distance;
    digitalWrite(trigPin, HIGH);    // trigPin에서 초음파 발생
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH); // echoPin 이 HIGH를 유지한 시간을 저장 한다.
    distance = ((float)(340* duration)/ 1000)/ 2;
    //Serial.print("Distance:");    // 물체와 초음파 센서간 거리를 표시.
    //Serial.println(distance);
    return distance;
}
// 초음파 센서와 연결된 서보모터 작동
int Servo_con(){
    Servo.write(30);
    delay(300);
    int Ult_30 = Ultrasonic();
    delay(700);
    Servo.write(150);
    delay(300);
    int Ult_150 = Ultrasonic();
    delay(700);
    if(Ult_30 > Ult_150){
        val = 1;
    }
    else{
        val = 0;
    }
    Servo.write(90);

    return val;
}

```

## 10. 트랙주행 소스코드 (최종)

```

#include <Servo.h>    // 서보 라이브러리
Servo Servo;
//출력핀(trig)과 입력핀(echo)
int trigPin = 13;    // 디지털 13번 핀에 연결
int echoPin = 12;    // 디지털 12번 핀에 연결
int Ultra_d = 0;    // 초음파 센서
int val = 0;    // 좌우 경로 설정 변수

```

```

intRightMotor_E_pin = 5;      // 오른쪽 모터 전원부 5번 연결핀
intLeftMotor_E_pin = 6;      // 왼쪽 모터 전원부 6번 연결핀
intRightMotor_1_pin = 10;    // 오른쪽 모터 8번 IN1
intRightMotor_2_pin = 11;    // 오른쪽 모터 9번 IN2
intLeftMotor_3_pin = 8;      // 왼쪽 모터 10번 IN3
intLeftMotor_4_pin = 9;      // 왼쪽 모터 11번 IN4
//모터 속도 조절, 최대속도 : 255
intL_MotorSpeed = 140; // 왼쪽 모터 속도
intR_MotorSpeed = 140; // 오른쪽 모터 속도
intL_Line = A5; // 왼쪽 라인트레이서 센서 - 확장설프드 A5
intC_Line = A4; // 가운데 라인트레이서 센서 - 확장설프드 A4
intR_Line = A3; // 오른쪽 라인트레이서 센서 - 확장설프드 A3
//라인트레이서 적외선센서 값
intSL = 1;
intSC = 1;
intSR = 1;
// 부품 셋업
void setup() {
    Servo.attach(2);          // 서보모터 PWM 디지털입출력 2번핀 연결
    pinMode(echoPin, INPUT);  // echoPin 입력
    pinMode(trigPin, OUTPUT); // trigPin 출력
    // 모터 핀 설치
    pinMode(RightMotor_E_pin, OUTPUT); // 오른쪽 모터 전원부 출력
    pinMode(LeftMotor_E_pin, OUTPUT);  // 왼쪽 모터 전원부 출력
    pinMode(RightMotor_1_pin, OUTPUT);  // 오른쪽 모터 앞바퀴 출력
    pinMode(RightMotor_2_pin, OUTPUT);  // 오른쪽 모터 뒷바퀴 출력
    pinMode(LeftMotor_3_pin, OUTPUT);   // 왼쪽 모터 앞바퀴 출력
    pinMode(LeftMotor_4_pin, OUTPUT);   // 왼쪽 모터 뒷바퀴 출력
    Serial.begin(9600); // PC와의 시리얼 통신 값 9600
    Serial.println("Activate!"); // 통신에 성공했을 경우, 시리얼 모니터 창에 Activate! 출
    력
}
// 라인트레이서 값 읽기
void loop() {
    Ultra_d = Ultrasonic(); // 초음파 센서
    Serial.println(Ultra_d);
    intL = digitalRead(L_Line);
    intC = digitalRead(C_Line);
    intR = digitalRead(R_Line);

    Serial.print("digital : ");
    Serial.print(L);

```



```
Serial.print(", ");
Serial.print(C);
Serial.print(", ");
Serial.print(R);
Serial.print("  ");
```

// 라인트레이서 인식 세팅

```
if (L == LOW && C == LOW && R == LOW ) {
    L = SL; C = SC; R = SR;
}
```

// 라인트레이서에서 센터만 인식이 됐을경우 직진,

```
if (L == LOW && C == HIGH && R == LOW ) {
    if (Ultra_d < 70 ) { // 물체가 70mm 앞에 감지됐을 경우, 장애물 감지 및 위험회피 알고리
```

즘 작동

```
    Serial.println("위험, 물체 감지!");
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    delay(3000);
    motor_role(LOW, LOW); // 후진
    delay(1000);
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    delay(1000);
    Servo.write(30);
    delay(1000);
    Servo.write(180);
    delay(1000);
    Servo.write(90);
    delay(5000);
    motor_role(HIGH, HIGH);
    delay(500);
```

```
}
```

```
motor_role(HIGH, HIGH);
```

```
Serial.println("직진");
```

```
}
```

// 라인트레이서에서 오른쪽만 인식이 됐을경우 우회전,

```
else if (L == LOW && R == HIGH ){
```

```
    motor_role(LOW, HIGH);
```

```
    Serial.println("우회전");
```

```
}
```

```

// 라인트레이서에서 왼쪽만 인식이 됐을 경우 좌회전,
else if (L == HIGH && R == LOW ) {
    motor_role(HIGH, LOW);
    Serial.println("좌회전");
}
// 라인트레이서에서 왼쪽 오른쪽 전부 인식이 됐을경우, 정지
else if (L == HIGH && R == HIGH ) {
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    Serial.println("정지");
}
// 주차 감지 알고리즘
if (Ultra_d < 130 ) { // 130mm 앞에 물체가 감지됐을 경우
    if (L == HIGH && C == HIGH && R == HIGH ) { // 적외선 센서가 전부 켜져있을 경우
        Serial.println("주차 감지.");
        analogWrite(RightMotor_E_pin, 0);
        analogWrite(LeftMotor_E_pin, 0);
        delay(3000);
        motor_role(LOW, LOW); // 후진
        delay(700);
        motor_role(HIGH, LOW); // 회전
        delay(800);
        motor_role(HIGH, HIGH); // 전진
        delay(500);
    }
}

SL = L; SC = C; SR = R;
}

// 디지털방식으로 모터의 출력 상태 0, 1로 기록
void motor_role(int R_motor, int L_motor) {
    digitalWrite(RightMotor_1_pin, R_motor);
    digitalWrite(RightMotor_2_pin, !R_motor);
    digitalWrite(LeftMotor_3_pin, L_motor);
    digitalWrite(LeftMotor_4_pin, !L_motor);
}
// 아날로그 방식으로 모터의 출력 상태 0 ~ 255 기록
analogWrite(RightMotor_E_pin, R_MotorSpeed); // 우측 모터 속도값 기록
analogWrite(LeftMotor_E_pin, L_MotorSpeed); // 좌측 모터 속도값 기록
}

// 초음파 센서 작동

```

```
int Ultrasonic(){  
    long duration, distance;  
    digitalWrite(trigPin, HIGH);    // trigPin에서 초음파 발생  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW);  
    duration = pulseIn(echoPin, HIGH); // echoPin 이 HIGH를 유지한 시간 저장  
    distance = ((float)(340* duration)/ 1000)/ 2;  
    return distance;  
}
```

## 5 . 주차별 보고서

작성일자: 2022년 9월 14일

프로젝트 명	자율주행 스마트카		
추진 활동	프로젝트 개발배경 및 필요성, 프로젝트 목표 작성	진도(%)	10%
팀 명	8조	성명	김동준, 김도형
추진 내용	<p>1.프로젝트 시스템 환경 작성 &gt;활용률 극대화 &gt;교통사고 감소</p> <p>2.프로젝트의 목표 작성 &gt;자율주행 자동차가 각광 받는 시대의 흐름에 맞춰 인간의 조작 없이 스스로 자율주행이 가능하여 몸이 불편해 운전하지 못하는 사람들이나 무인 배달 서비스, 무인 택시, 차량 대여 서비스 등 사람들이 각종 편리한 시스템을 사용할 수 있는 자율주행 스마트카를 제작하도록 한다.</p> <p>3.프로젝트 개발을 위해 필요한 부품 구매</p>		
문제점 및 대책	부품이 아직 도착하지 않아서 개발 진행 상황이 지연되고 있다.		
향후 추진 계획	부품이 도착할 때부터 프로젝트 작성과 함께 팀원과 회의하여 개발예정이다.		

작성일자: 2022년 9월 21일

프로젝트 명	자율주행 스마트카		
추진 활동	프로젝트 시스템 환경, 준비물 및 기능 / 부품 설계 및 조립	진도(%)	30%
팀 명	8조	성명	김동준, 김도형
추진 내용	<p>1.프로젝트 개발배경 및 필요성 작성            &gt;기본적으로 아두이노 우노 호환 보드를 사용하고, 아두이노 프로그램 IDE(Intergrated Development Environment)를 설치해준다.</p> <p>2.프로젝트 준비물 및 기능            &gt;프로젝트에 필요한 준비물들을 사진과 기능을 함께 첨부하였다.</p> <p>3.프로젝트에 필요한 부품조립 및 설계            &gt;프로젝트 작성에 필요한 부품이 도착하여 조립 및 설계를 진행하였다.</p>		
문제점 및 대책	<p>1.프로젝트 목적에 맞는 코드를 완벽히 구현하지 못하였다.            &gt;소스코드 검색 및 실습</p> <p>2.이 프로젝트만의 독자적이고 창의적인 아이디어를 찾지 못하였다.            &gt;팀원과 아이디어 회의</p> <p>3.조립 및 설계가 아직 완벽하지 않다.            &gt;팀원과 순서대로 프레임 조립</p>		
향후 추진 계획	<p>조립 및 설계를 완성해놓고 프로젝트 목적에 맞는 코드를 구현할 것이다.</p> <p>그리고 이 프로젝트만이 가지는 창의적인 아이디어를 팀원과 회의 및 탐색해볼 예정이다.</p>		

작성일자: 2022년 9월 28일

프로젝트 명	자율주행 스마트카		
추진 활동	보고서 추가 작성, 서보모터, 라인트레이서 구동 실습	진도(%)	50%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.보고서 추가 작성 2.스마트카 프레임 조립 완성 2.Use-Case 작성 진행중 3.서보모터, 초음파센서, 라인트레이서센서 코드구현후 작동 실습		
문제점 및 대책	1.배터리를 준비하지 못해서 주행 구동을 하지 못하였다. >건전지 구입 2.이 프로젝트만이 가지는 독창적인 아이디어 부족 >팀원과 아이디어 회의		
향후 추진 계획	프로젝트 소스 코드 구현 후 작동 실습 및 아이디어 회의		

작성일자: 2022년 10월 5일

프로젝트 명	자율주행 스마트카		
추진 활동	use-case작성, 프로젝트 코드 작성 및 실습	진도(%)	55%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.Use-Case 작성 2.프로젝트 코드 작성 및 실습 (DC모터 드라이버 / 전진 후진 제어 / 바퀴 정회전,역회전) 3.아이디어 회의		
문제점 및 대책	1.추가로 구매한 온습도 센서, 아두이노 카메라 부품이 아직 도착하지 않았다. 2.초음파센서 디버그창 출력문 오류 3.부품이 팀원에게 있는 상태로 팀원이 질병에 걸려 독자적으로 실습 불가하여 프로젝트 개발이 지연됨		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의		

작성일자: 2022년 10월 12일

프로젝트 명	자율주행 스마트카		
추진 활동	초음파 센서를 활용한 자율주행 테스트	진도(%)	60%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.팀원과 아이디어 회의 2.초음파 센서를 활용한 장애물 회피 자율주행 테스트		
문제점 및 대책	자율주행 테스트 결과 초음파 센서가 데드존(센서 측면 부분)을 제대로 인식하지 못하여 센서와의 대각선 방향 가까이 있는 장애물 인식이 원활하지 않다.		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의		



작성일자: 2022년 10월 26일

프로젝트 명	자율주행 스마트카		
추진 활동	트랙구상 및 제작, opencv 설치, 파이썬으로 아두이노 제어	진도(%)	65%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.팀원과 직접 만나서 아이디어 회의 2.트랙구상 및 제작 3.opencv설치 및 스마트카 활용방안 공부 4.파이썬으로 아두이노 연동 제어 테스트		
문제점 및 대책	Opencv를 다루어본적이 없어서 활용하는데 미숙하기 때문에 공부가 더 필요하다.		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의		

작성일자: 2022년 11월 09일

프로젝트 명	자율주행 스마트카		
추진 활동	트랙구상 및 제작, led센서로 신호등 만들기	진도(%)	70%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.팀원과 zoom을 통한 아이디어 회의 2.트랙구상 및 제작 3.led센서를 활용하여 트랙주행에 필요한 신호등 만들기 4.파이썬 아두이노 제어를 통한 시리얼 통신테스트		
문제점 및 대책	트랙제작이 번거로워 시간이 걸림, RC카에 카메라를 연동하여 카메라가 led센서 신호등 색상을 인식하는지 테스트해야 하는데 부품을 각자가 가지고 있어 테스트를 하지 못함		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의 / 트랙구상 및 제작		

작성일자: 2022년 11월 16일

프로젝트 명	자율주행 스마트카		
추진 활동	트랙구상 및 제작, led센서로 신호등 만들기, 파이썬 아두이노 제어	진도(%)	75%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.트랙구상 및 제작 2.led센서를 활용하여 트랙주행에 필요한 신호등 제작 3.파이썬 아두이노 제어를 통한 시리얼 통신테스트		
문제점 및 대책	RC카에 연동할 카메라를 주문하였는데 아직 도착하지 않아 실습을 해보지 못함		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의 / 트랙구상 및 제작		

작성일자: 2022년 11월 23일

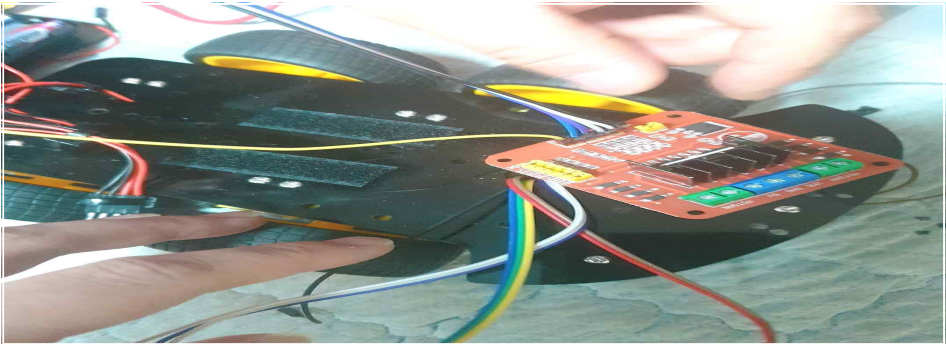
프로젝트 명	자율주행 스마트카		
추진 활동	트랙구상 및 제작, led센서로 신호등 만들기, 파이썬 아두이노 제어	진도(%)	80%
팀 명	8조	성명	김동준, 김도형
추진 내용	1.트랙구상 및 제작 2.스마트카 브레드보드에 카메라장착 및 연결 3.카메라 연결 테스트		
문제점 및 대책	카메라 장착 후 빛만 깜빡이고 카메라 화면이 활성화가 안된다.		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의 / 트랙구상 및 제작		

작성일자: 2022년 11월

프로젝트 명	자율주행 스마트카		
추진 활동	트랙구상 및 제작, led센서로 신호등 만들기, 파이썬 아두이노 제어	진도(%)	85%
팀 명	8조	성명	김동준, 김도형
추진 내용	1. 트랙구상 및 제작 2. led센서로 신호등 만들기 3. 파이썬 아두이노 제어		
문제점 및 대책	<p>저번주부터 스마트카에 카메라를 연결을 시도하였으나 빛만 깜빡이고 정상적으로 작동이 되지않아서 이번주에도 계속해서 연결을 시도했으나 실패했다.</p> <p>카메라를 활용하여 신호를 인식하여 주행 할 예정이었으나 지금 당장은 시간이 여유롭지 않기 때문에 기존에 활용했던 초음파 센서를 활용하여 사람 모형등이 앞에 있으면 주행을 멈춘다거나 하는 방법으로 대체하는 변경사항이 있을 예정이다.</p>		
향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의 / 트랙구상 및 제작		

30일

작성일자: 2022년 12월

프로젝트 명	자율주행 스마트카		
추진 활동	트랙 주행에 필요한 코드 작성, 마무리 작업	진도(%)	90%
팀 명	8조	성명	김동준, 김도형
07일	추진 내용	1. 트랙 주행에 필요한 코드 작성 및 테스트 2. 마무리 작업	
	문제점 및 대책	<p>코드 작성하면서 주행테스트하는 도중에 DC모터드라이버 전원부 기판이 고장나서 물건 재주문으로 인해 시간을 빼앗겼다.</p>  <p>어쩔수 없이 고장난 기판에 납땜 작업하면서 물건이 오기전까지 테스트 진행중이다.</p>	
	향후 추진 계획	계속해서 프로젝트 소스코드 개발 및 구현 실습 / 아이디어 회의 / 마무리 작업	

작성일자: 2022년 12월

프로젝트 명	자율주행 스마트카		
추진 활동	트랙 주행에 필요한 코드 작성, 마무리 작업	진도(%)	100%
팀 명	8조	성명	김동준, 김도형
추진 내용	1. 트랙 주행에 필요한 코드 작성 및 테스트 2. 마무리 작업 3. 발표 전 보고서 정리 및 발표자료 준비		
문제점 및 대책	기존에 계획했던 opencv차선감지와 카메라를 부착해 신호등 색상인식을 통한 트랙주행을 하려했지만, 각종 외부적 요인과 소스코드 오류를 해결하지 못하였기 때문에 보다 완성도 있는 프로젝트 작품을 제작할 수 없어 아쉬움이 남는다.		
향후 추진 계획	발표를 마치고, 기존에 계획했지만 미구현으로 마무리된 소스코드나 작업 추진해보기		

14일

## 6 . 프로젝트 결과물

### -최종 소스코드

```
#include <Servo.h>                // 서보 라이브러리
Servo Servo;
//출력핀(trig)과 입력핀(echo)
int trigPin = 13;                 // 디지털 13번 핀에 연결
int echoPin = 12;                 // 디지털 12번 핀에 연결
int Ultra_d = 0;                  // 초음파 센서
int val = 0;                      // 좌우 경로 설정 변수

int RightMotor_E_pin = 5;         // 오른쪽 모터 전원부 5번 연결핀
int LeftMotor_E_pin = 6;          // 왼쪽 모터 전원부 6번 연결핀
int RightMotor_1_pin = 10;        // 오른쪽 모터 8번 IN1
int RightMotor_2_pin = 11;        // 오른쪽 모터 9번 IN2
int LeftMotor_3_pin = 8;          // 왼쪽 모터 10번 IN3
int LeftMotor_4_pin = 9;          // 왼쪽 모터 11번 IN4
//모터 속도 조절, 최대속도 : 255
int L_MotorSpeed = 140; // 왼쪽 모터 속도
int R_MotorSpeed = 140; // 오른쪽 모터 속도
int L_Line = A5; // 왼쪽 라인트레이서 센서 - 확장설프드 A5
int C_Line = A4; // 가운데 라인트레이서 센서 - 확장설프드 A4
int R_Line = A3; // 오른쪽 라인트레이서 센서 - 확장설프드 A3
//라인트레이서 적외선센서 값
int SL = 1;
int SC = 1;
int SR = 1;
// 부품 셋업
void setup() {
    Servo.attach(2);               // 서보모터 PWM 디지털입출력 2번핀 연결
    pinMode(echoPin, INPUT);       // echoPin 입력
    pinMode(trigPin, OUTPUT);      // trigPin 출력
    // 모터 핀 설치
    pinMode(RightMotor_E_pin, OUTPUT); // 오른쪽 모터 전원부 출력
    pinMode(LeftMotor_E_pin, OUTPUT);  // 왼쪽 모터 전원부 출력
    pinMode(RightMotor_1_pin, OUTPUT); // 오른쪽 모터 앞바퀴 출력
    pinMode(RightMotor_2_pin, OUTPUT); // 오른쪽 모터 뒷바퀴 출력
    pinMode(LeftMotor_3_pin, OUTPUT);  // 왼쪽 모터 앞바퀴 출력
    pinMode(LeftMotor_4_pin, OUTPUT);  // 왼쪽 모터 뒷바퀴 출력
    Serial.begin(9600);            // PC와의 시리얼 통신 값 9600
}
```



```

    Serial.println("Activate!"); // 통신에 성공했을 경우, 시리얼 모니터 창에 Activate! 출
    력
}
// 라인트레이서 값 읽기
void loop() {
    Ultra_d = Ultrasonic(); // 초음파 센서
    Serial.println(Ultra_d);
    intL = digitalRead(L_Line);
    intC = digitalRead(C_Line);
    intR = digitalRead(R_Line);

    Serial.print("digital : ");
    Serial.print(L);
    Serial.print(", ");
    Serial.print(C);
    Serial.print(", ");
    Serial.print(R);
    Serial.print(" ");

    // 라인트레이서 인식 세팅
    if (L == LOW && C == LOW && R == LOW ) {
        L = SL; C = SC; R = SR;
    }
    // 라인트레이서에서 센터만 인식이 됐을경우 직진,
    if (L == LOW && C == HIGH && R == LOW ) {
        if (Ultra_d < 70 ) { // 물체가 70mm 앞에 감지됐을 경우, 장애물 감지 및 위험회피 알고리
        즘 작동
            Serial.println("위험, 물체 감지!");
            analogWrite(RightMotor_E_pin, 0);
            analogWrite(LeftMotor_E_pin, 0);
            delay(3000);
            motor_role(LOW, LOW); // 후진
            delay(1000);
            analogWrite(RightMotor_E_pin, 0);
            analogWrite(LeftMotor_E_pin, 0);
            delay(1000);
            Servo.write(30);
            delay(1000);
            Servo.write(180);
            delay(1000);
            Servo.write(90);
            delay(5000);
        }
    }
}

```

```

        motor_role(HIGH, HIGH);
        delay(500);
    }
    motor_role(HIGH, HIGH);
    Serial.println("직진");
}

// 라인트레이서에서 오른쪽만 인식이 됐을 경우 우회전,
else if (L == LOW && R == HIGH ) {
    motor_role(LOW, HIGH);
    Serial.println("우회전");
}

// 라인트레이서에서 왼쪽만 인식이 됐을 경우 좌회전,
else if (L == HIGH && R == LOW ) {
    motor_role(HIGH, LOW);
    Serial.println("좌회전");
}

// 라인트레이서에서 왼쪽 오른쪽 전부 인식이 됐을 경우, 정지
else if (L == HIGH && R == HIGH ) {
    analogWrite(RightMotor_E_pin, 0);
    analogWrite(LeftMotor_E_pin, 0);
    Serial.println("정지");
}

// 주차 감지 알고리즘
if (Ultra_d < 130 ) { // 130mm 앞에 물체가 감지됐을 경우
    if (L == HIGH && C == HIGH && R == HIGH ) { // 적외선 센서가 전부 켜져있을 경우
        Serial.println("주차 감지.");
        analogWrite(RightMotor_E_pin, 0);
        analogWrite(LeftMotor_E_pin, 0);
        delay(3000);
        motor_role(LOW, LOW); // 후진
        delay(700);
        motor_role(HIGH, LOW); // 회전
        delay(800);
        motor_role(HIGH, HIGH); // 전진
        delay(500);
    }
}

SL = L; SC = C; SR = R;
}

```

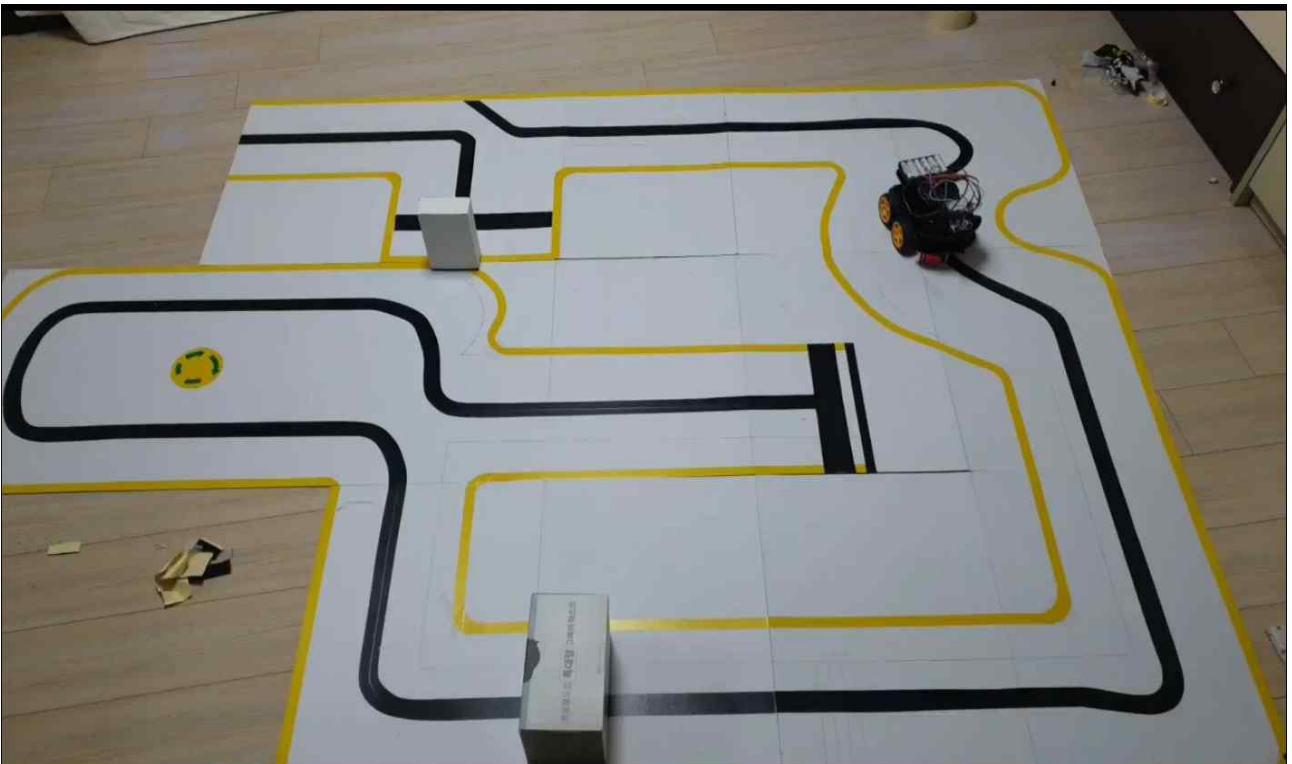
```

// 디지털방식으로 모터의 출력 상태 0, 1로 기록
void motor_role(int R_motor, int L_motor) {
    digitalWrite(RightMotor_1_pin, R_motor);
    digitalWrite(RightMotor_2_pin, !R_motor);
    digitalWrite(LeftMotor_3_pin, L_motor);
    digitalWrite(LeftMotor_4_pin, !L_motor);
// 아날로그 방식으로 모터의 출력 상태 0 ~ 255 기록
    analogWrite(RightMotor_E_pin, R_MotorSpeed); // 우측 모터 속도값 기록
    analogWrite(LeftMotor_E_pin, L_MotorSpeed); // 좌측 모터 속도값 기록
}

// 초음파 센서 작동
int Ultrasonic(){
    longduration, distance;
    digitalWrite(trigPin, HIGH);    // trigPin에서 초음파 발생
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH); // echoPin 이 HIGH를 유지한 시간 저장
    distance = ((float)(340* duration)/ 1000)/ 2;
    returndistance;
}

```

**[결과물 영상은 따로 준비]**



## 7 . 프로젝트를 진행하며 느낀점

이번 캡스톤 디자인 프로젝트를 팀원과 함께 진행하면서 느낀 점은 우리가 예상하고, 목표했던 것과 달리 잘못된 결과가 나올 수도 있고, 문제가 생겨 오랫동안 해결방안을 찾지 못할 수도 있는데 이러한 부분들은 항상 팀원과 협동하여 회의 및 검토하고, 테스트를 해보는 것이 프로젝트를 차질없이 진행하는데 제일 현명한 방법이라는 생각이 들었습니다.

하지만 그럼에도 기존에 계획했던 opencv파이썬-아두이노 연동을 통한 차선감지와 스마트카에 부착된 브레드보드에 카메라를 연결하여 신호등 색상인식을 통한 정교한 트랙주행을 구현하지 못한 것이 아쉬움이 남습니다.

각종 외부적 요인과 소스코드 오류를 해결하지 못한 것에 조금더 공부가 필요하다는 것을 느꼈습니다.

처음에는 스스로 움직이는 자율주행 자동차를 막연하게 생각하며 팀원과 함께 부품을 구하고 코드를 작성하여 프로젝트를 진행하면 졸업과제이지만 재미있게 진행할 수 있을 것이라고 생각했습니다.

하지만 생각과 달리 프레임 부품 조립부터 시작해서 아두이노 연동을 통해 스마트카에 부착된 각종 모터들과 부품들을 실습하며 코드를 작성 후 작동되는지 테스트하고, 라인트레이서 활용과 함께 opencv-파이썬과 카메라까지 연동시켜 프로젝트를 진행하다보니 막히는 부분도 솔직히 많았습니다.

그외중에 자율주행을 컨셉으로 하다보니 그에 필요한 큰 트랙까지 만들어야 하다보니 손도 정말 많이 갔고 중간중간 어려움이 있었습니다.

그럼에도 팀원과 함께 서로 역할을 분배하고 노력하다보니 결과물을 만들어냈고, 이 작품을 만들면서 공부하다보니 모르던 부분도 새롭게 많이 알게된 점도 많고, 프로젝트를 만드는 실력도 조금은 발전하지 않았나 생각해봅니다.

마지막으로 한 학기동안 좋은 수업해주신 홍진근 교수님께 감사하다는 말씀 전하며 프로젝트 발표를 마칠것습니다.

감사합니다.