

실험 9

자동차 키트 조립 및 동작 제어

2025. 10. 22. (Wed)

실험 목표

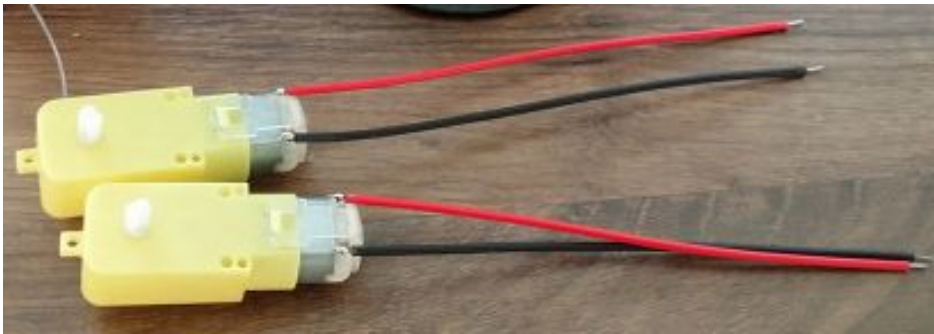
- 자동차 키트 조립 및 조립된 자동차를 주행하기 위한 제어 프로그램 작성
- 사용할 모듈
 - 모터
 - 라인 트레이서
 - 모터 드라이버
 - 초음파 센서
- 기타 준비물
 - 드라이버

실험 목표

- 조립된 자동차를 주행하기 위한 제어 프로그램 작성
- 라인 트레이서 센서를 이용한 추적 주행 프로그램 작성
- 초음파 센서를 이용한 거리 조절 주행 프로그램 작성

DC모터 동작

- DC모터와 wire 연결 (전선과 니퍼 필요)
- 모터 제어 드라이버
- DC모터와 모터 제어 드라이버 연결
 - OUT1, OUT3: 모터+
 - OUT2, OUT4: 모터-



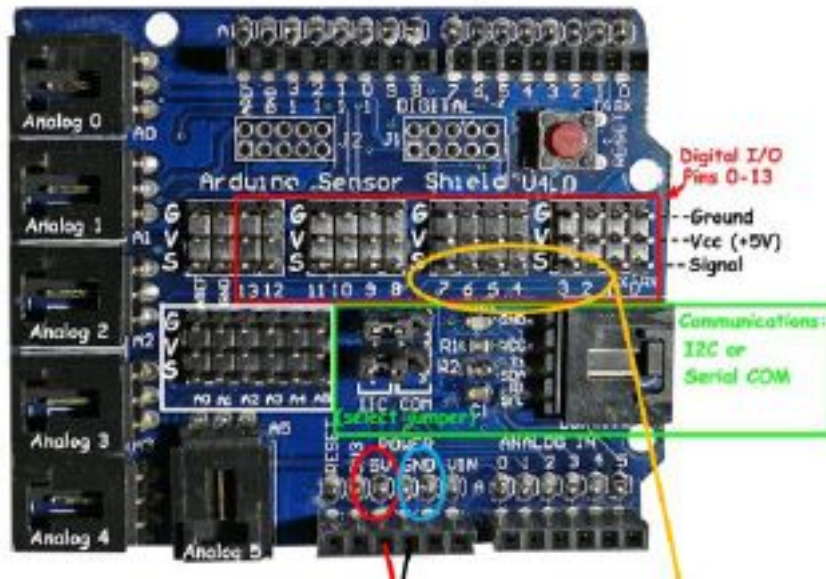
출처 :
http://www.app123.kr/xe/s4a_pds/2098



모터 제어 드라이버

DC모터 동작

- 센서 쉴드를 아두이노 보드에 장착
- 센서 쉴드와 모터 제어 드라이버 연결



센서 쉴드

모터 제어드라이버	아두이노 보드
ENA	S6
IN1	S7
IN2	S3
ENB	S5
IN3	S4
IN4	S2
5V	5V
GND	GND

DC모터 동작

■ 센서 쉴드와 모터 제어 드라이버 연결

- Arduino VCC pin – **5V** or **12V**

- 일단 5V로 연결해보고,
드라이버 **LED**는 켜지는데
모터가 돌지 않는 경우
건전지 **12V** 전원에 연결

(노트북 저전력 설정으로 인한
POWER 부족이 원인)

- Arduino GND pin – **GND**



모터 제어 드라이버

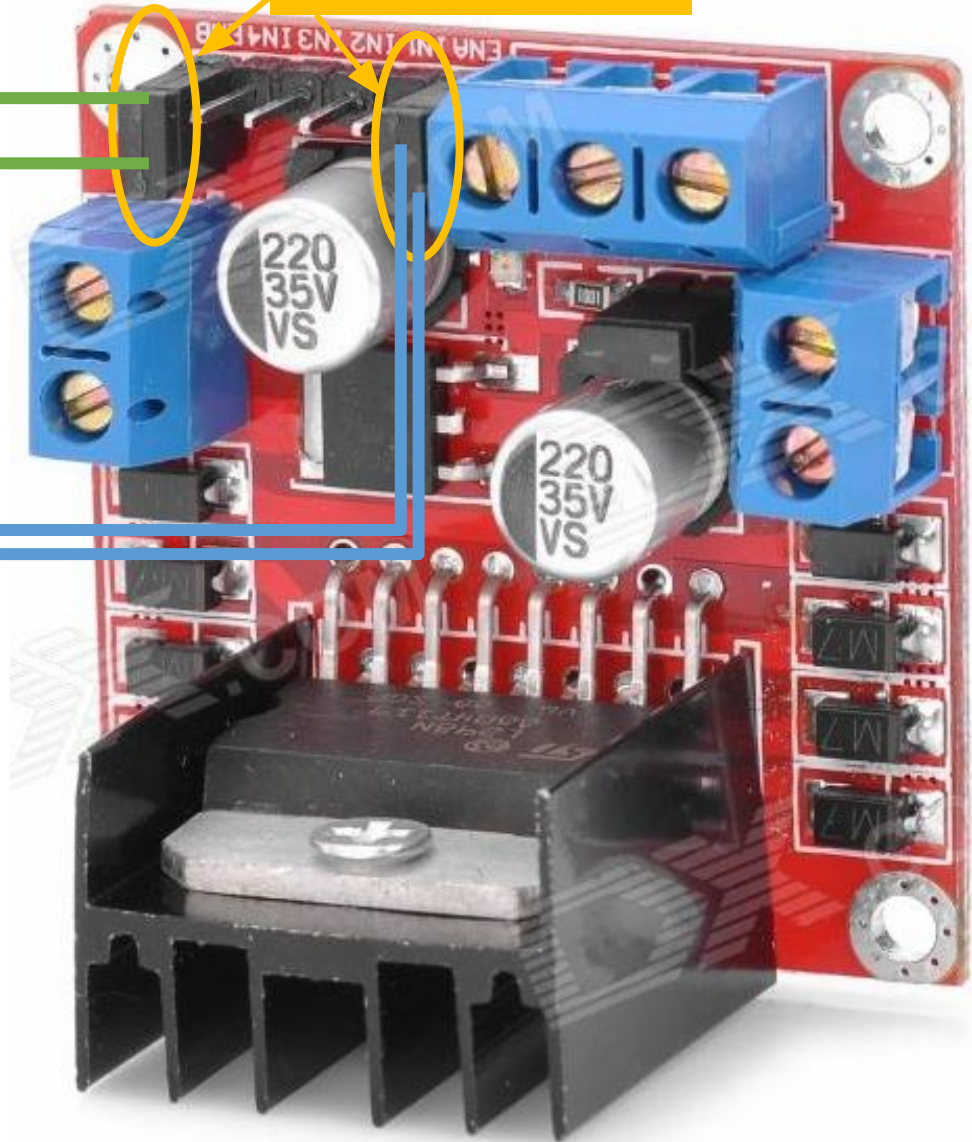
DC모터 동작

- 센서 쉴드와
모터 제어 드라이버
연결
 - ENA, ENB 캡을 제거하고
핀 2개를 각각
센서 쉴드에 연결
(breadboard 이용)

제거 가능

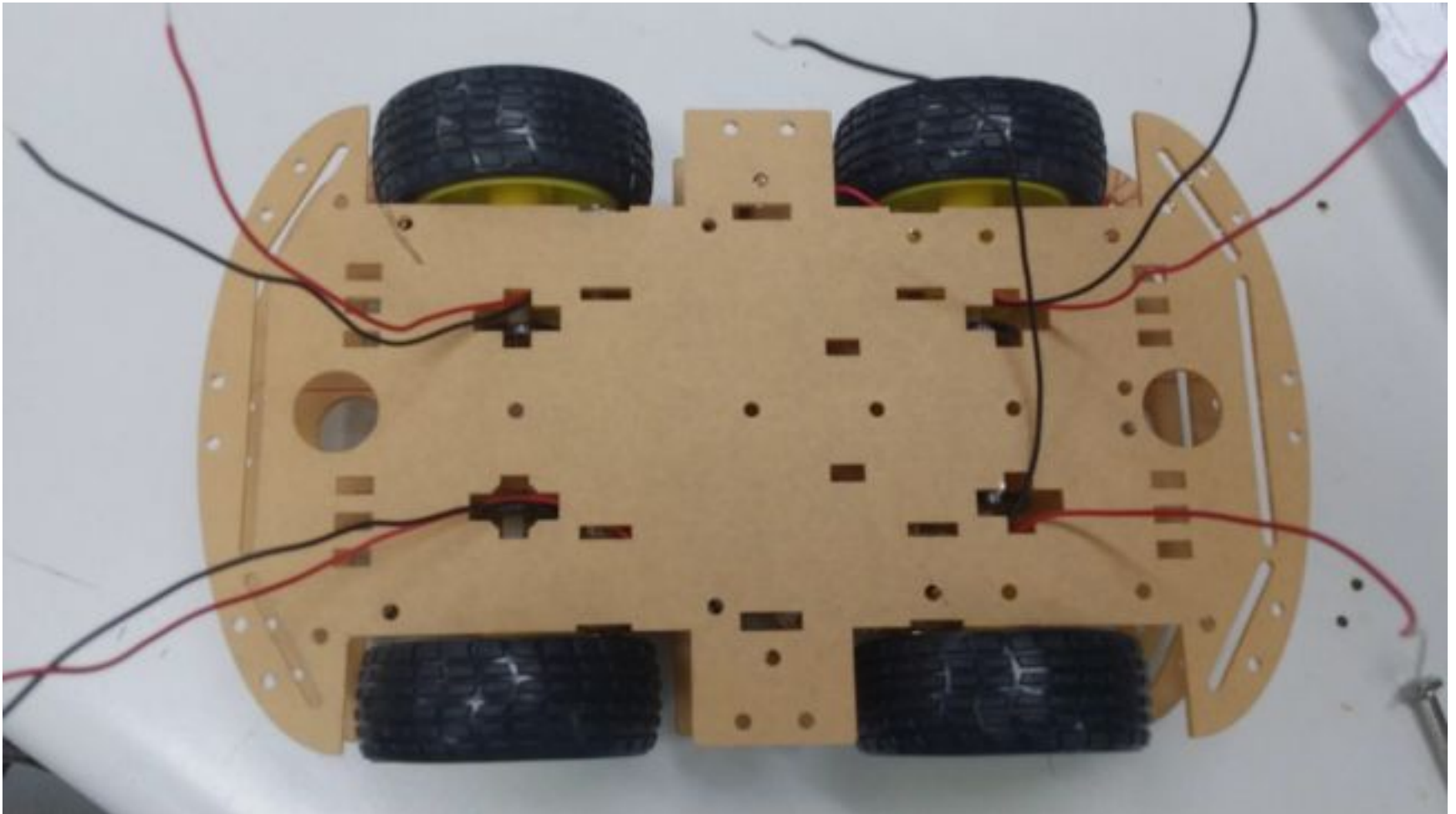
S6

S5



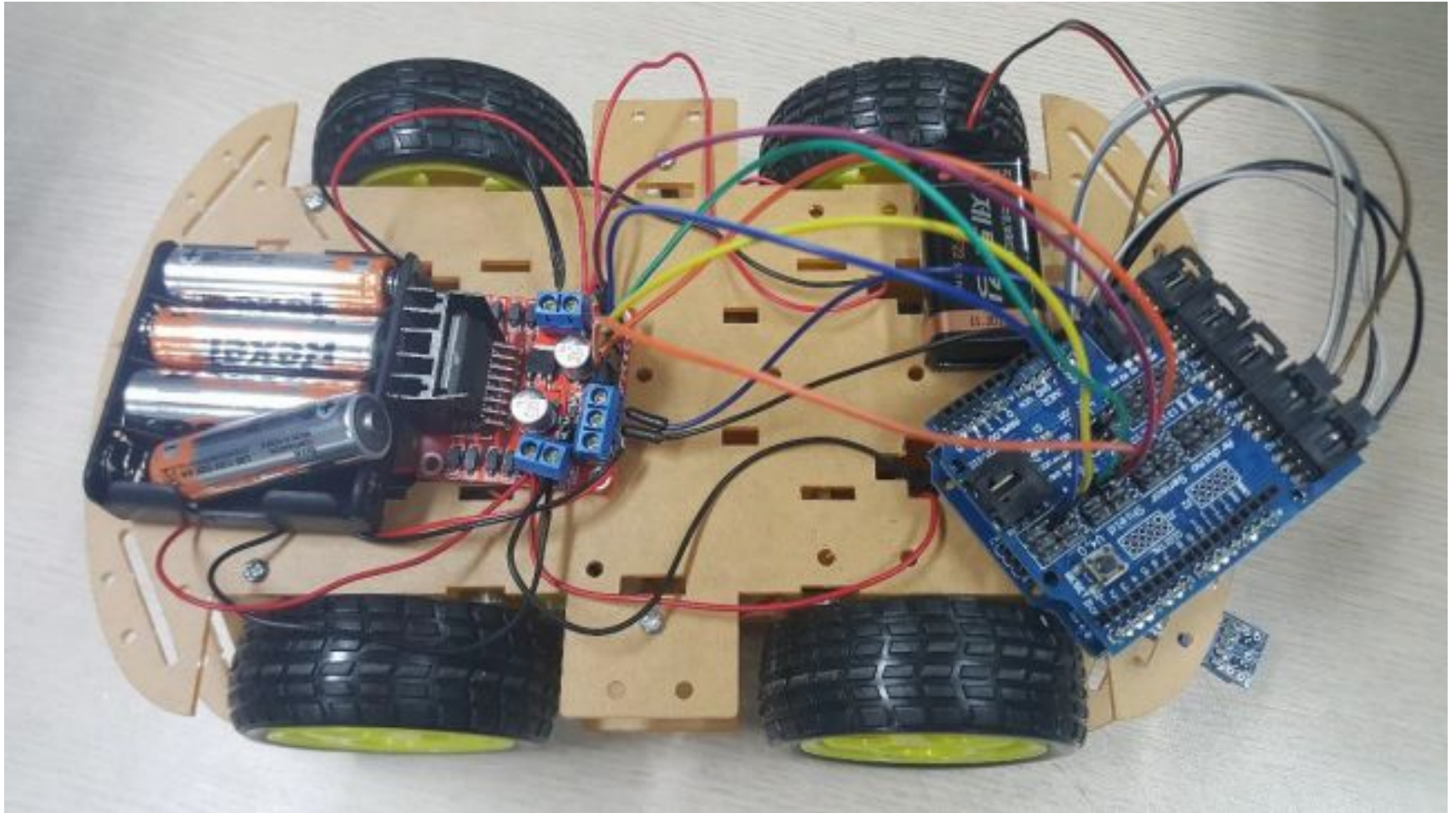
조립

- 상체 고정



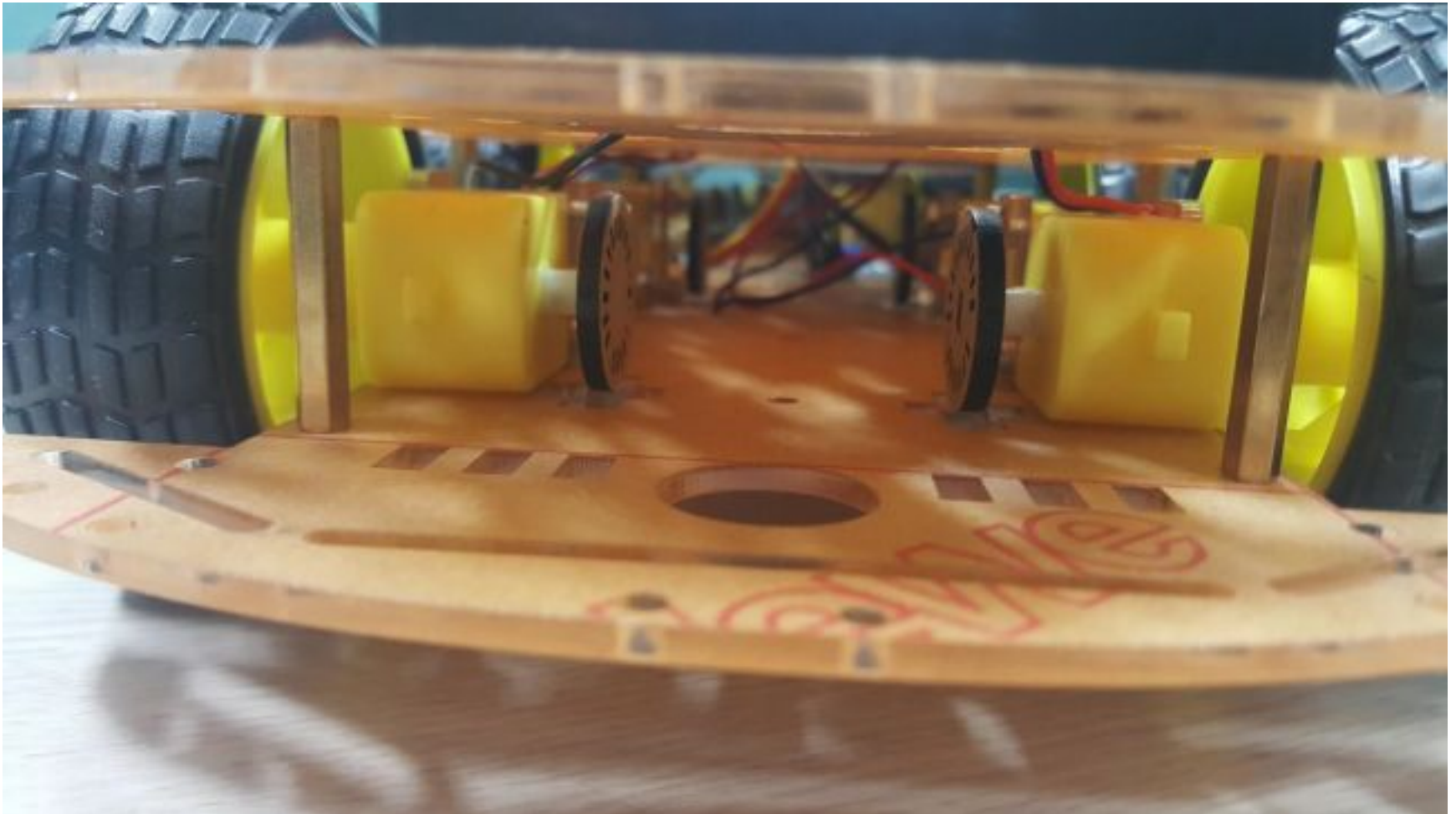
조립

- 보드와 모터제어드라이버 연결



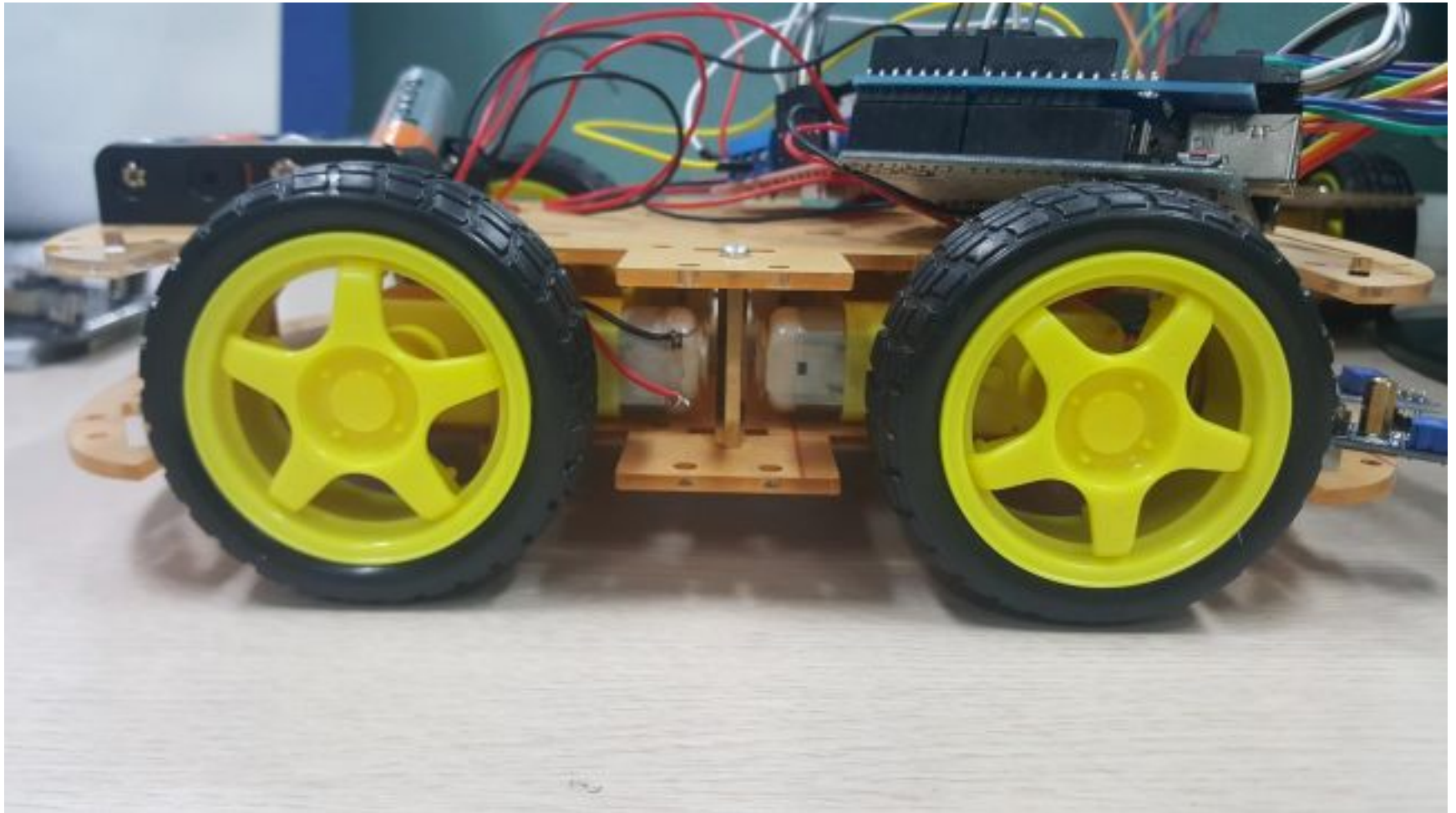
조립

- 차체 앞 모습



조립

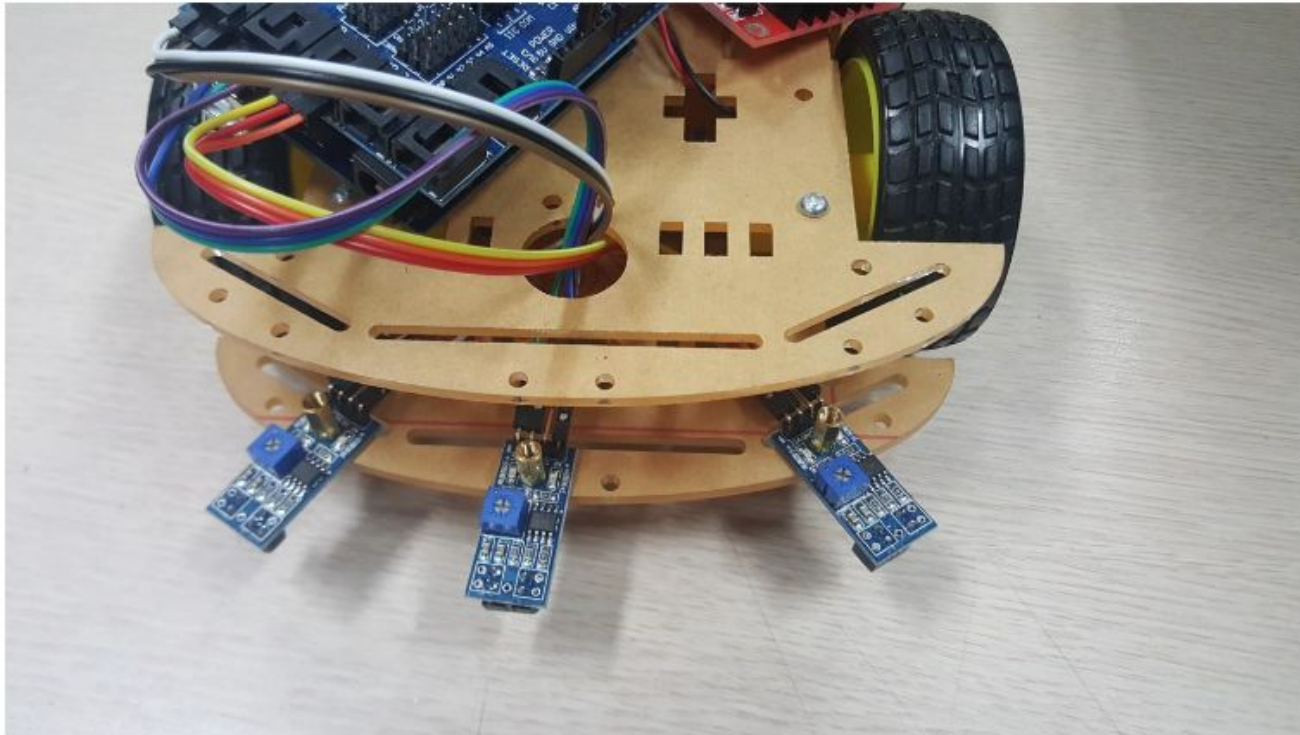
- 차체 옆 모습



조립

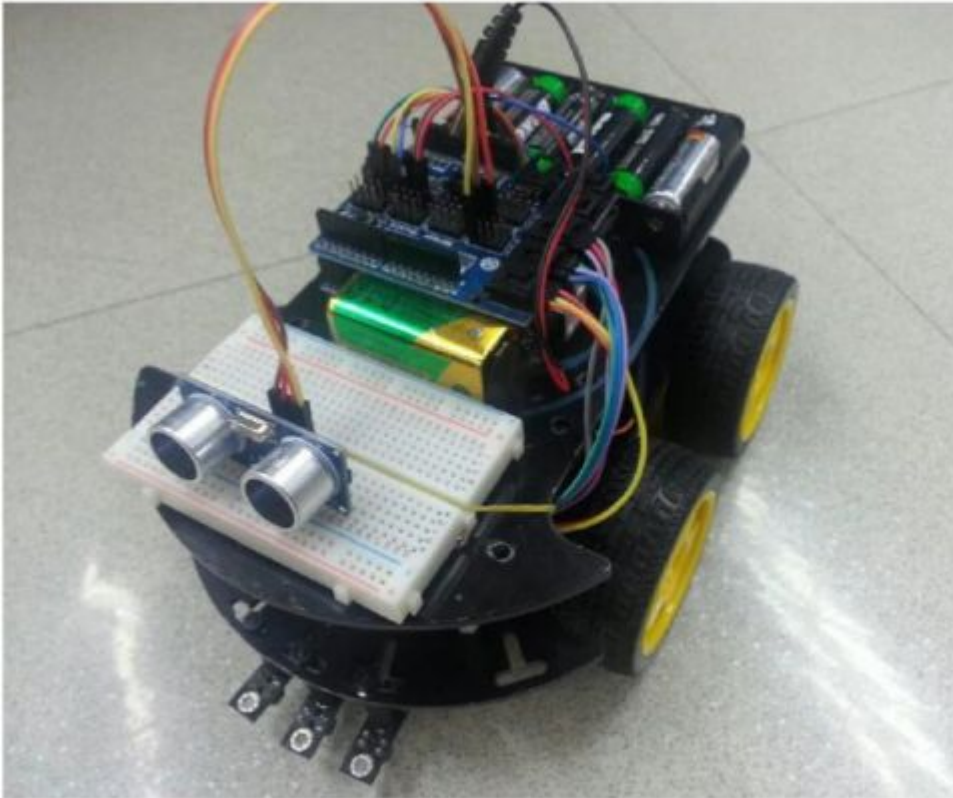
■ 라인 트레이서 장착

- **주의사항**: 키트 구성품으로 라인 트레이서 3개를 장착하려면 볼트가 부족함
- 기둥 6개 중 가운데 두개를 포기하고 여기서 나온 볼트 두 개를 이용하여 라인 트레이서 장착할 것!



조립

■ 초음파 센서 장착



초음파 센서 모듈	아두이노
VCC	5V
Trig	D12
Echo	D13
GND	GND

조립

■ 초음파 센서 장착

The diagram illustrates the connection of an ultrasonic sensor to an Arduino Sensor Shield. The shield is shown with various ports and components labeled. The connections are as follows:

- Buckled Analog Port:** A0~A4 (green lines)
- Buckled Analog Port A5:** (green line)
- Analog Port A0~A5:** (green lines)
- Digital IO port D0-D13:** (orange lines)
- Communication Port:** Selectable between I2C and Uart (blue lines)
- Arduino Port:** (yellow lines)
- Ground Vcc Signal:** (black lines)
- To Signal To VCC To GND:** (red and black lines)

The ultrasonic sensor module is shown in the bottom right corner, with its pins connected to the shield's headers. The sensor is labeled "HC-SR04" and "BUC4".

- Buckled Analog Port - Easy, Solid connection with Analog input with VCC/GND.
- Buckled Communication Port - Easy communication port with I2C and Uart
- Digital IO port - Standard servo pin compatible,
- Analog IO port - 2.54 grid male pin header connections

조립

- **작동 확인: 8주차 모터 드라이버 작동 코드 실행**
 - 4개 바퀴가 같은 방향으로 회전
 - RC카 전진

실험 1: 전진, 후진, 회전 동작 제어

- 주행 방향에 따른 변수 선언

```
#define Forward 1 // 전진  
#define Backward 2 // 후진  
#define Turn_Right 3 // 우회전  
#define Turn_Left 4 // 좌회전  
#define Stop 5 // 정지
```

```
int direction = 0; // 차량 운행상태 전역변수 선언  
int speed = 200;
```

실험 1: 전진, 후진, 회전 동작 제어

- Digital pin 연결 선언 – ENA, ENB 주의

```
#define ENA 6  
#define EN1 7  
#define EN2 3  
#define EN3 4  
#define EN4 2  
#define ENB 5
```

- Pin mode 선언

```
void setup()  
{  
  pinMode(ENA, OUTPUT); // ENA  
  pinMode(EN1, OUTPUT); // EN1  
  pinMode(EN2, OUTPUT); // EN2  
  pinMode(ENB, OUTPUT); // ENB  
  pinMode(EN3, OUTPUT); // EN3  
  pinMode(EN4, OUTPUT); // EN4  
}
```

실험 1: 전진, 후진, 회전 동작 제어

- 전진 프로그래밍
 - HIGH, LOW customizing 필요

```
delay(1000); // 1초 delay
direction = direction + 1; // 방향 전환 시 사용되는 변수

if(direction == Forward) // 전진 주행
{
    digitalWrite(EN1, HIGH);
    digitalWrite(EN2, LOW);
    analogWrite(ENA, speed);
    digitalWrite(EN3, HIGH);
    digitalWrite(EN4, LOW);
    analogWrite(ENB, speed);
}
```

실험 1: 전진, 후진, 회전 동작 제어

■ 우회전 프로그래밍

- HIGH, LOW customizing 필요

```
else if(direction == Turn_Right)
{
    digitalWrite(EN1, HIGH);
    digitalWrite(EN2, LOW);
    analogWrite(ENA, speed);
    digitalWrite(EN3, LOW);
    digitalWrite(EN4, HIGH);
    analogWrite(ENB, speed);
}
```

■ 정지

```
else if(direction == Stop)
{
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
}
```

실험2: 특정 위치로 자동차 이동시키기

■ 조별 실험 진행

- 130cm정도의 직진 주행 후 오른쪽으로 70cm 주행
- 조건문 사용하지 않고 delay 함수로 구현
- 검사 X

실험 3: 라인 추적 주행

■ Pin 변수 및 mode 선언

```
#define ENA 6
#define EN1 7
#define EN2 3
#define EN3 4
#define EN4 2
#define ENB 5

#define LT_MODULE_L A2
#define LT_MODULE_F A1
#define LT_MODULE_R A0

#define CAR_DIR_ST 1
#define CAR_DIR_FW 2
#define CAR_DIR_LF 3
#define CAR_DIR_RF 4

int speed = 200;
int g_carDirection = CAR_DIR_ST;

void init_line_tracer_modules(){
    pinMode(LT_MODULE_L, INPUT);
    pinMode(LT_MODULE_F, INPUT);
    pinMode(LT_MODULE_R, INPUT);
}
```

실험 3: 라인 추적 주행

- 라인 트레이서 작동 여부 체크
 - True, False 가르는 기준 값은 센서에 맞게 조정

```
bool lt_isLeft()
{
    int ret = analogRead(LT_MODULE_L);
    Serial.print("left: ");
    Serial.println(ret);
    return (ret > 200) ? (true) : (false);
}
```

```
bool lt_isRight()
{
    int ret = analogRead(LT_MODULE_R);
    Serial.print("right: ");
    Serial.println(ret);
    return (ret > 200) ? (true) : (false);
}
```

```
bool lt_isForward()
{
    int ret = analogRead(LT_MODULE_F);
    Serial.print("forward: ");
    Serial.println(ret);
    return (ret > 200) ? (true) : (false);
}
```

실험 3: 라인 추적 주행

■ 자동차 방향 조절

```
void lt_mode_update()
{
    bool ll = lt_isLeft();
    bool ff = lt_isForward();
    bool rr = lt_isRight();

    if (ll && ff && rr)
    {
        g_carDirection = CAR_DIR_ST;
    }
    else if (!ll && !ff && !rr)
    {
        g_carDirection = CAR_DIR_ST;
    }
    else if (ll)
    {
        g_carDirection = CAR_DIR_LF;
    }
    else if (rr)
    {
        g_carDirection = CAR_DIR_RF;
    }
    else if (ff)
    {
        g_carDirection = CAR_DIR_FW;
    }
}
```

실험 3: 라인 추적 주행

■ 차량 운행

- 후진, 좌회전, 우회전, 정지 동작을 추가하여 프로그래밍

```
void car_update()
{
    Serial.print("Car update: ");
    Serial.println(g_carDirection);
    if (g_carDirection == CAR_DIR_FW) // 전진
    {
        Serial.println("Forward");
        digitalWrite(EN1, HIGH);
        digitalWrite(EN2, LOW);
        analogWrite(ENA, speed);
        digitalWrite(EN3, HIGH);
        digitalWrite(EN4, LOW);
        analogWrite(ENB, speed);
    }
}
```

실험 3: 라인 추적 주행

- 자동차 동작을 위한 loop
 - car_update에서 방향에 맞게 주행
 - It_mode_update에서 방향 변경

```
void setup() {  
    Serial.begin(9600);  
    init_line_tracer_modules();  
  
    pinMode(ENA, OUTPUT);  
    pinMode(EN1, OUTPUT);  
    pinMode(EN2, OUTPUT);  
    pinMode(ENB, OUTPUT);  
    pinMode(EN3, OUTPUT);  
    pinMode(EN4, OUTPUT);  
}  
  
void loop(){  
    car_update();  
    It_mode_update();  
}
```

실험 3: 라인 추적 주행

- 자동차 동작을 위한 loop
 - 라인트레이서 모듈 값에 따라 방향 변경 시리얼 모니터로 확인

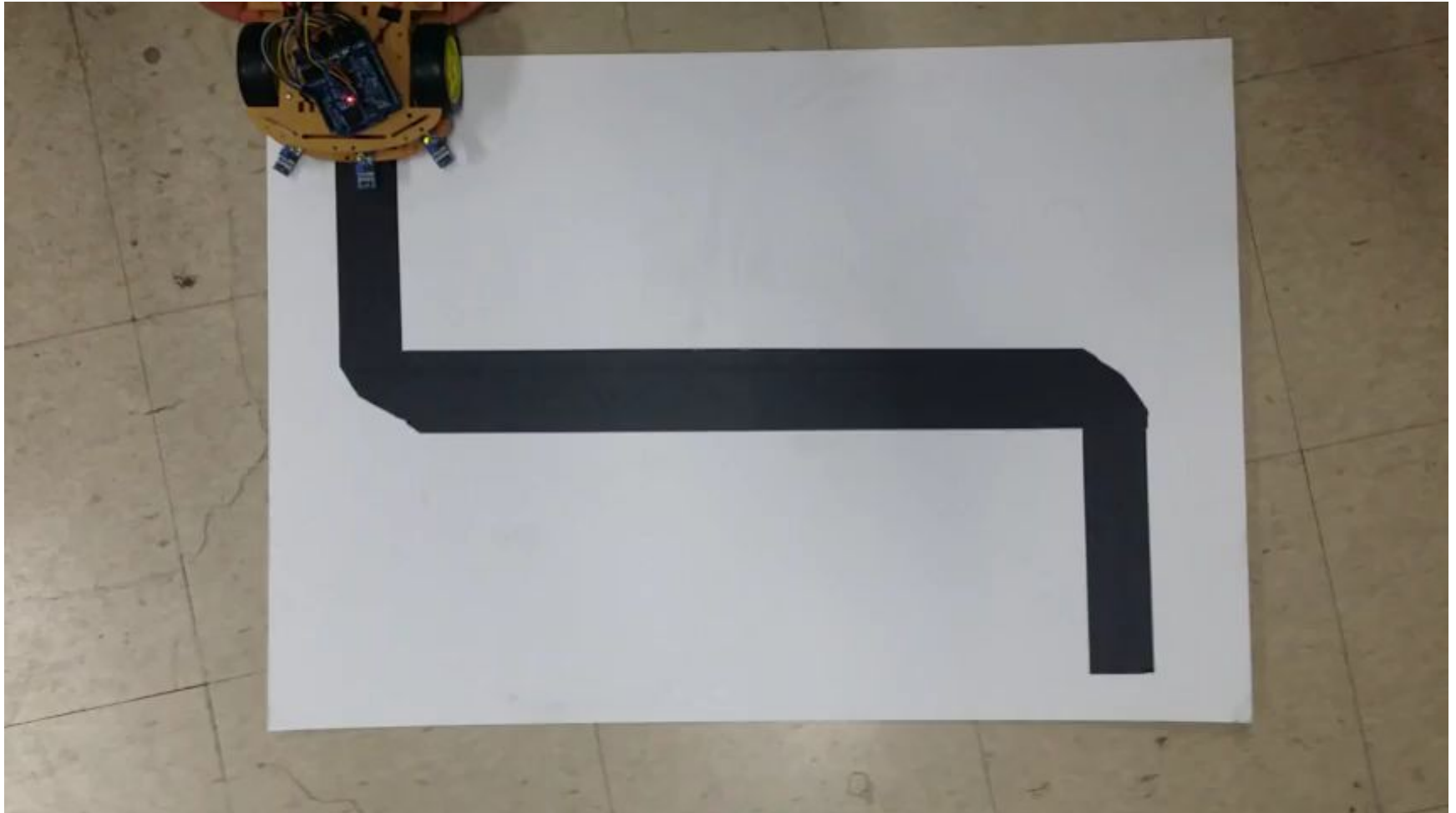
Output Serial Monitor X

Message (Enter to send message to 'Arduino Uno' on '/dev/ttyACM0')

```
Car update: 2 Forward
left: 31 right: 26 forward: 339
Car update: 2 Forward
left: 32 right: 27 forward: 26
Car update: 1 Stop
left: 31 right: 27 forward: 26
Car update: 1 Stop
left: 29 right: 25 forward: 25
Car update: 1 Stop
left: 29 right: 25 forward: 24
```


실험 3: 라인 추적 주행

■ 자동차 주행 시범



실험4: 거리 인식 주행

- 20cm 이내 거리에 장애물이 있으면 후진
 - 이외에는 전진
- Pin에 관한 설정은 실습4 참고

실험 검사 항목

■ 실험 1

- 전진, 후진, 우회전, 좌회전 순서로 주행

■ 실험 2

- 검사 생략

■ 실험 3

- 라인 추적을 하며 주행하는지 확인

■ 실험 4

- 전방 장애물 유무에 따라 전진 및 후진 주행

결과보고서 항목

- 없음