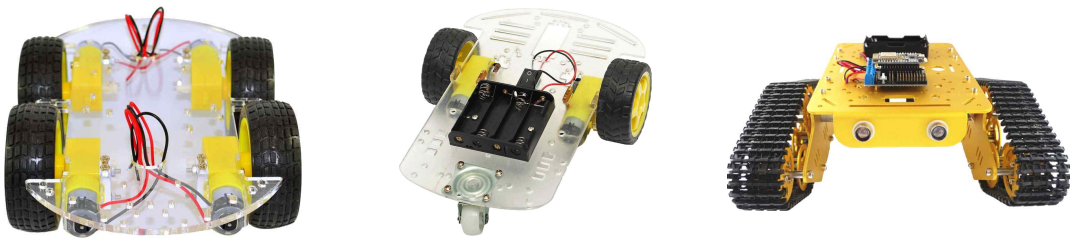
# Chapter 14. 아듀카 만들기

1. 아듀카(ArduCar : Arduino Controlled Car)

RC카는 Radio Controled Car의 약자로 무선 조종이 가능한 자동차를 말한다. 시중에는 완제품뿐만 아니라 조립이 가능한 키트도 여러 종류 판매되고 있으며 차체 역시 3D 프린터로 출력할 수 있는 등 다양한 형태와 다양한 기능을 구현해 볼 수 있다. 아두이노를 사용하여 만들 수 있는 무선 자동차에도 형태와 기능에 따라 다양한 종류가 있다.



(a) 4바퀴 자동차 (b) 2바퀴 자동차 (c) 탱크

그림 1. 아두이노를 이용한 무선 자동차 형태

이 책에서는 아두이노를 사용하여 만들 수 있는 자동차 중에서 4개의 DC 모터를 사용하는 4바퀴 아듀카(ArduCar)를 만들어본다. DC 모터는 정확한 위치 제어가 어려운 단점은 있지만 제어가 간단하고 가격이 저렴하므로 사용하였다. 4개의 DC 모터를 사용한 이유는 다양한 주변 장치를 추가할 수 있는 충분한 공간을 확보하기 위해서다. 온라인 또는 오프라인에서 저렴한 가격에 쉽게 구할 수 있는 부품을 사용하여 아듀카를 만들 수 있도록 하기 위한 것도 4바퀴 무선 자동차를 선택한 이유 중 하나다. 아듀카는 그림 2와 같은 형태를 가지고 있으며 필요에 따라 다양한 주변장치를 추가할 수 있다.

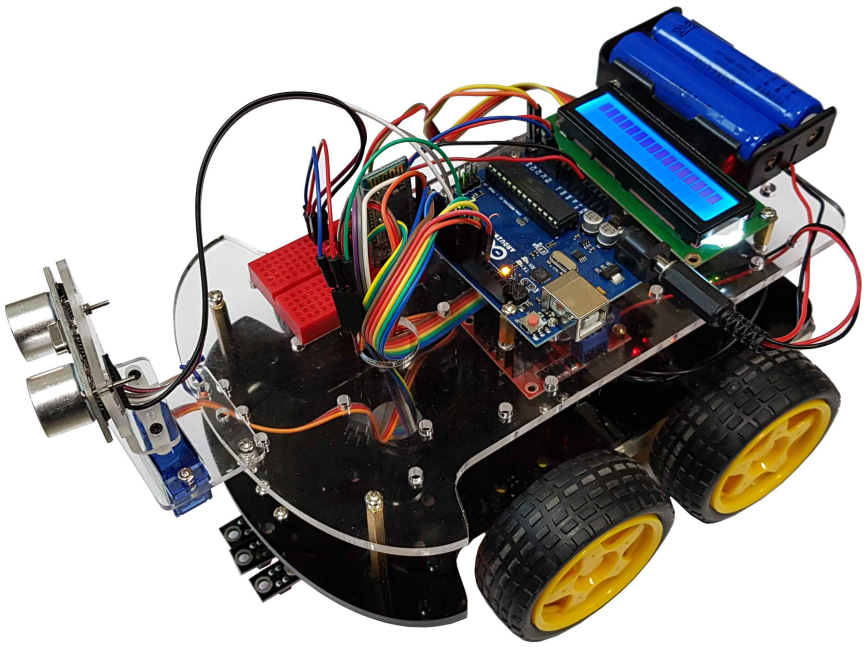


그림 2. 장애물 회피 아듀카

이 장에서는 먼저 아듀카의 뼈대를 만들어 테스트할 것이며 여기에는 모터와 텍스트 LCD가 포함된다. 이 장에서 만들어진 아듀카에는 무선 조정을 위해 블루투스나 적외선 수신기를 추가할 수 있고, 라인트레이서를 만들기 위해 라인트레이서 모듈을 추가할 수 있으며, 장애물을 피해 움직이는 자동차를 만들기 위해서 초음파 거리 센서를 추가할 수 있는 등 다양한 방법으로 응용이 가능하다. 아듀카를 사용하여 만들 수 있는 다양한 자동차에 대해서는 이후 장에서 하나씩 알아본다.

## 2. 모터 연결

아듀카는 그림 2에서 볼 수 있듯이 2단으로 구성된다. 먼저 자동차 샤시에 모터를 연결해 보자. 모터를 연결하기 위해서는 모터 고정을 위한 지지대를 먼저 연결해야 한다. 샤시의 밑판에 모터 지지대를 연결하자.

그림 3. 모터 지지대 연결

지지대를 연결한 후 4개의 모터를 지지대에 연결한다.

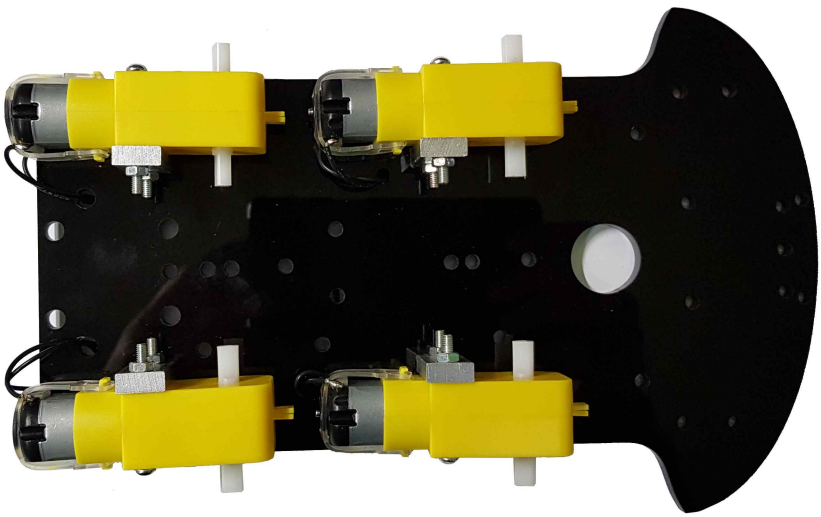


그림 4. 모터 연결

아듀카는 4개의 모터를 사용하지만 이 책에서 사용하는 모터 쉴드는 2개의 모터만을 제어할 수 있으므로 4개의 모터를 2개씩 묶어서 제어해야 한다. 4개의 모터는 왼쪽 2개와 오른쪽 2개로 묶는다. 그 이유는 왼쪽 2개의 모터와 오른쪽 2개의 모터는 항상 동일한 방향으로 회전하기 때문이다. 아듀카의 동작은 전진, 후진, 우회전, 좌회전의 4가지로 나눌 수 있으며 각 동작에서 모터의 회전은 표 1과 같다.

| 동작 | 왼쪽 모터 | 오른쪽 모터 |
|---|---|---|
| 전진 | 정회전 | 정회전 |
| 후진 | 역회전 | 역회전 |
| 우회전 | 정회전 | 역회전 |
| 좌회전 | 역회전 | 정회전 |

표 1. 아듀카의 동작[1)]

각각의 모터에 배터리를 연결하여 동일한 방향으로 회전하도록 하는 모터 연결선을 확인한 후 왼쪽 모터와 오른쪽 모터를 2개씩 묶어 모터 제어 모듈에 연결한다. 표 2는 모터 연결선을 모터 제어 모듈에 연결하는 방법을 나타낸 것으로 아듀카가 전진하도록 배터리를 연결하였을 때 배터리의 (+)와 연결되는 모터 연결선을 (+)로 표시하였다.

| 모터 | 모터 제어 모듈 |
|---|---|
| 왼쪽 모터 (+) | OUT1 |
| 왼쪽 모터 (-) | OUT2 |
| 오른쪽 모터 (+) | OUT3 |
| 오른쪽 모터 (-) | OUT4 |

표 2. 모터 연결선과 모터 제어 모듈 연결

표 2에서 OUT1과 OUT2를 바꾸어 연결하거나 OUT3과 OUT4를 바꾸어 연결한 경우라면 스케치에서 조정이 가능하다. 하지만 왼쪽이나 오른쪽 모터 2개가 동일한 방향으로 회전하도록 연결되어 있지 않다면 분해하고 다시 조립해야 하므로 모터의 회전 방향을 반드시 확인한 후 모터 제어 모듈에 연결해야 한다.

1) '정회전'은 아듀카가 전진하는 방향으로 모터가 회전하는 것을 말한다. 다만 '정회전'과 '역회전'은 모터 단위의 회전을, '전진'과 '후진'은 아듀카의 동작을 나타내는 것으로 구별하여 표시하였다.

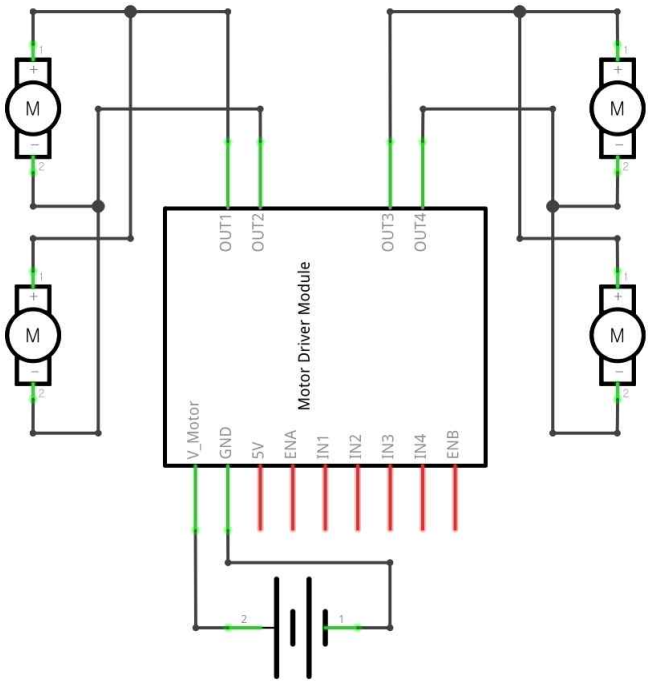



그림 5. 모터 제어 모듈과 모터 연결 회로도

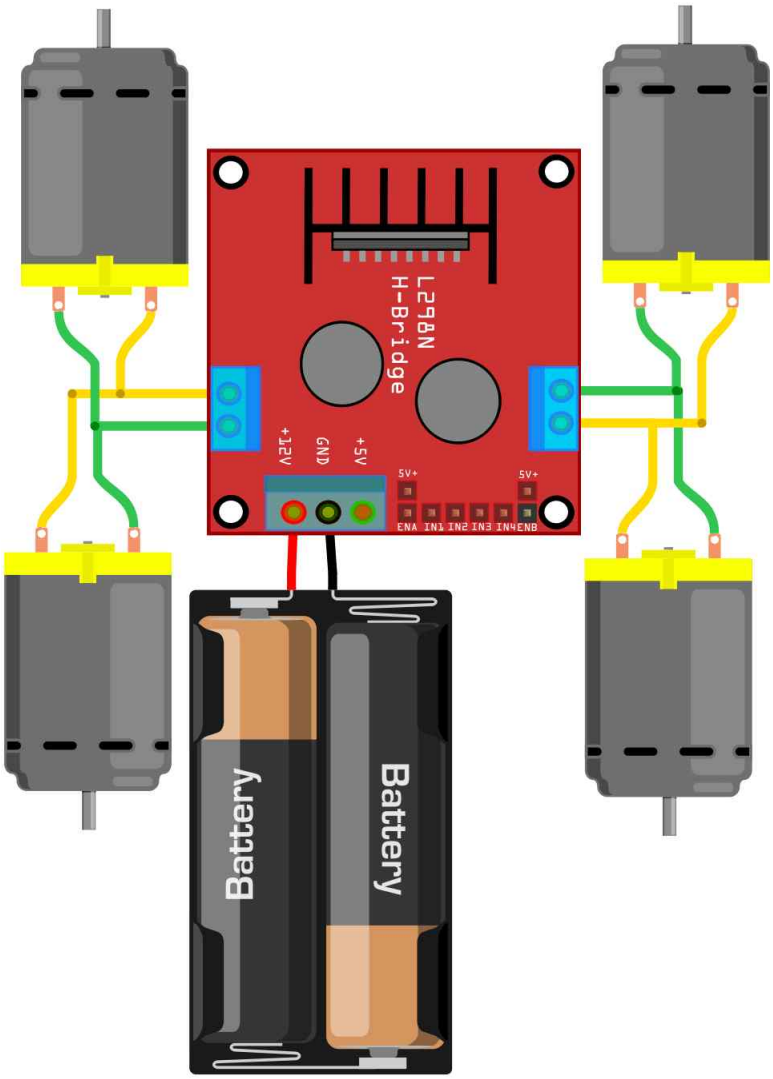



그림 6. 모터 제어 모듈과 모터 연결 회로

아듀카는 18650 충전지 2개를 사용한다. AA 건전지를 사용할 수도 있지만 모터가 2개인 경우에는 AA 건전지 4개, 모터가 4개인 경우에는 AA 건전지 8개를 사용해야 안정적인 동작이 가능하였으므로 많은 전력을 공급할 수 있는 18650 충전지를 사용하였다.[2)]

2) 모터 개수에 따라 필요로 하는 AA 건전지의 개수는 사용하는 모터 제어 모듈에 따라 달라질 수 있으며 위의 설명은 이 책에서 사용한 모터 제어 모듈을 기준으로 한 것이다.

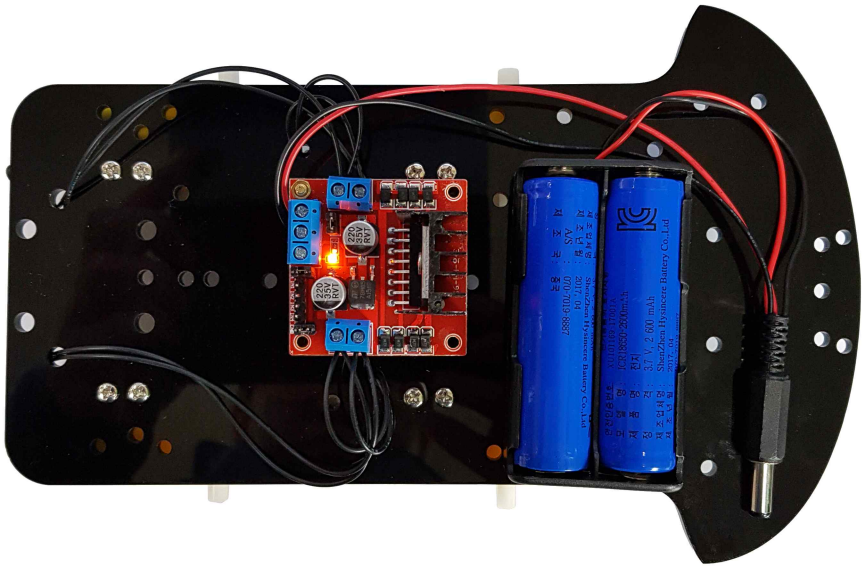


그림 7. 모터 제어 모듈과 모터 연결

밑판과 위판을 연결하고 아두이노 우노를 위판에 고정시킨다.

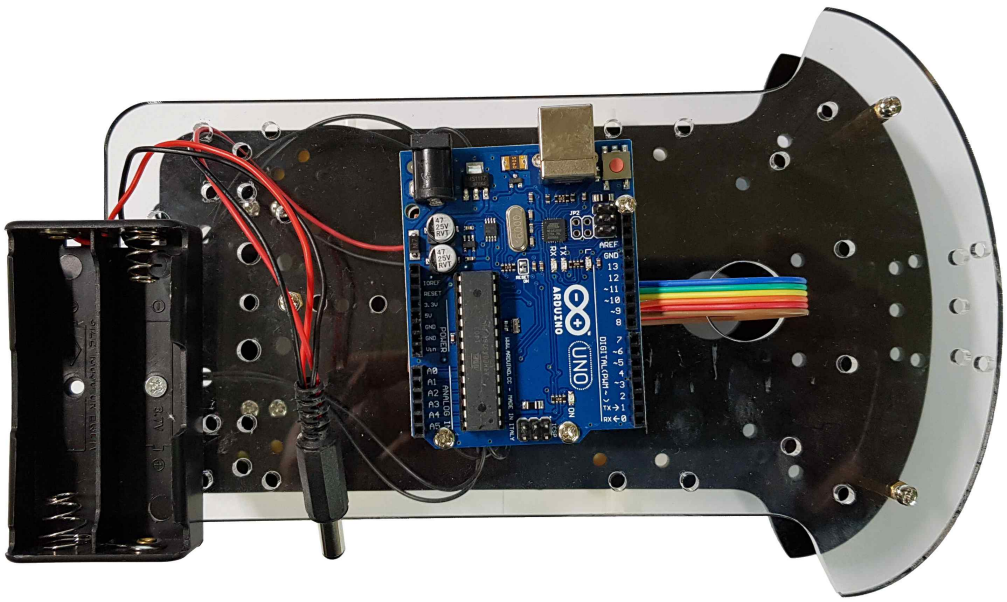


그림 8. 위판과 밑판 연결

아두이노 우노와 모터 드라이버를 연결한다. 모터의 회전 속도 조절이 가능하도록 모터 제어 모듈의 ENA와 ENB는 PWM 출력이 가능한 디지털 5번과 6번 핀에 연결하였다. 회전 방향 조절을 위한 INn(n = 1, … , 4)핀은 PWM 출력이 필요하지 않으므로 디지털 10~13번 핀에 연

결하였다.

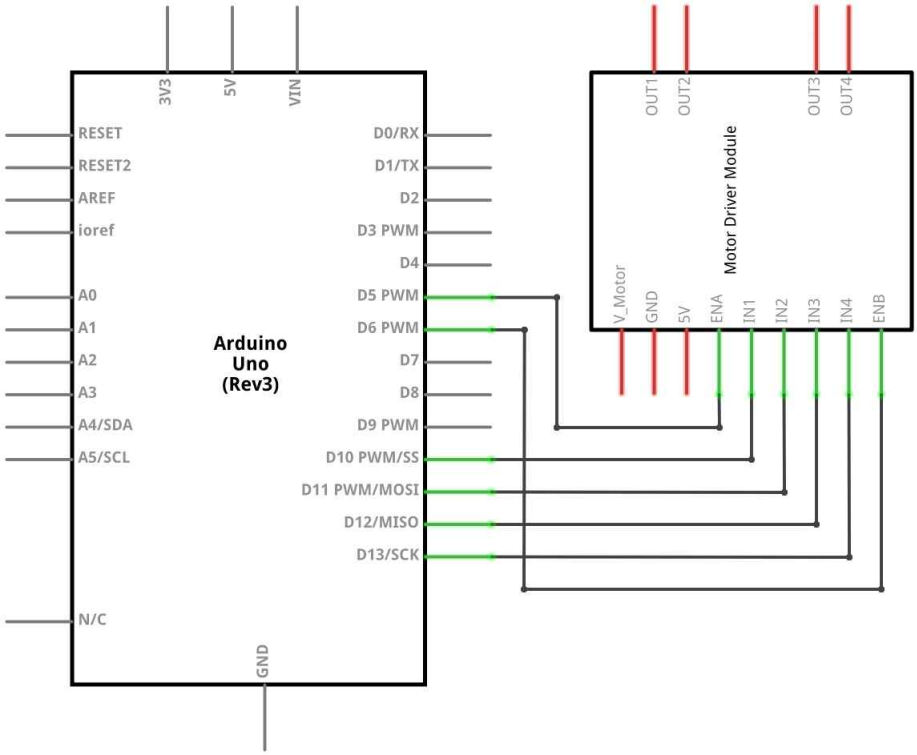



그림 9. 아두이노 우노와 모터 제어 모듈 연결 회로도

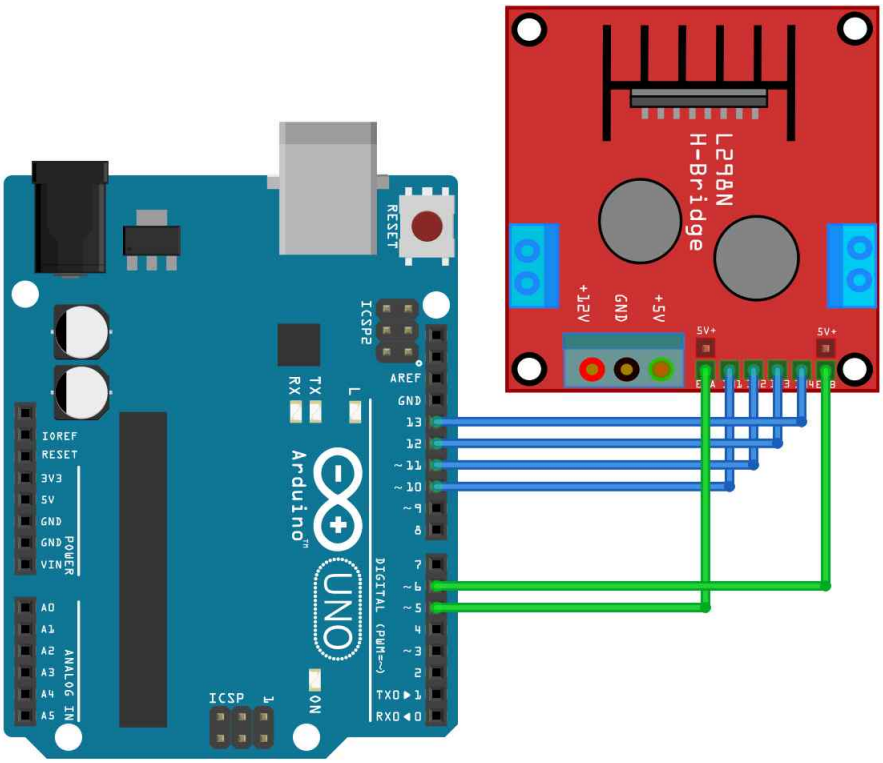



그림 10. 아두이노 우노와 모터 제어 모듈 연결 회로

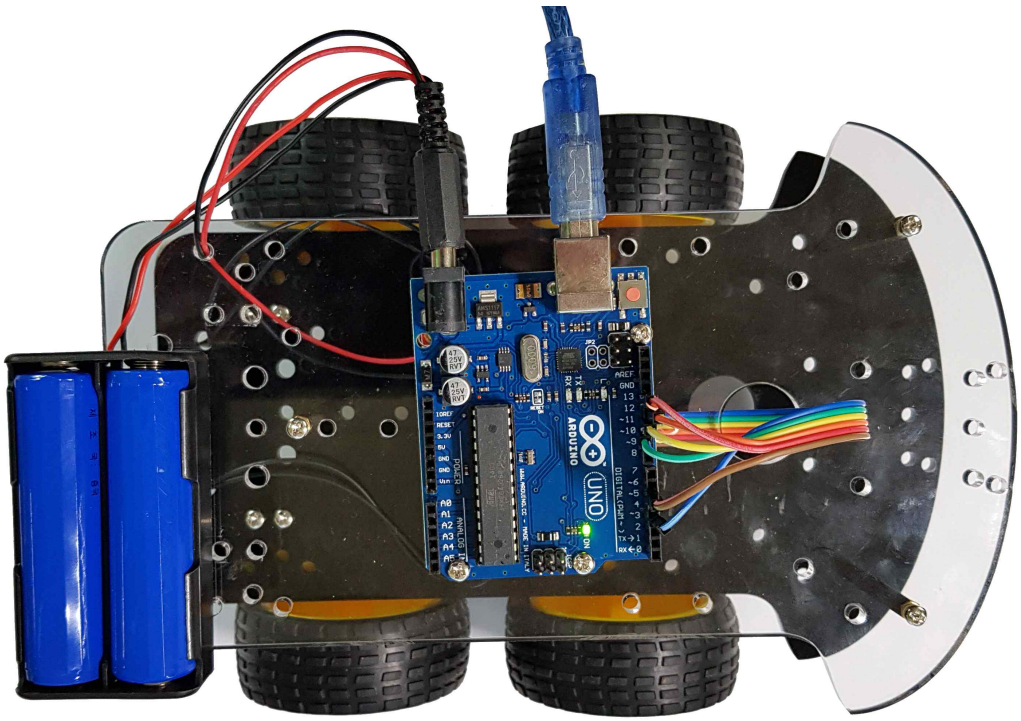


그림 11. 아두이노 우노와 모터 제어 모듈 연결

스케치 1은 시리얼 모니터로 전진(F), 후진(B), 우회전(R), 좌회전(L), 정지(S)에 해당하는 명령을 입력 받고 이에 따라 모터를 구동시키는 스케치의 예다.

스케치 1. 모터 테스트

```
const int SPEED = 100;                  // 모터 회전 속도

int IN1 = 10, IN2 = 11;                 // 왼쪽 모터 회전 방향
int IN3 = 12, IN4 = 13;                 // 오른쪽 모터 회전 방향
int ENA = 5, ENB = 6;                   // 모터 활성화

void setup() {
  Serial.begin(9600);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
}

void loop() {
  byte command;
  if(Serial.available()){               // 명령 수신
    command = Serial.read();
    motorControl(command);
  }
}

void motorControl(byte command) {
  if(command == 'F') {                  // Forward
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
```

```
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    Serial.println("Forward...");
  }
  if(command == 'B') {                    // Backward
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    Serial.println("Backward...");
  }
  if(command=='L') {                      // Left turn
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    Serial.println("Left turn...");
  }
  if(command == 'R') {                    // Right turn
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    Serial.println("Right turn...");
  }
  if(command=='S') {                      // Stop
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
```

```
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, LOW);
        Serial.println("Stop...");
    }
}
```

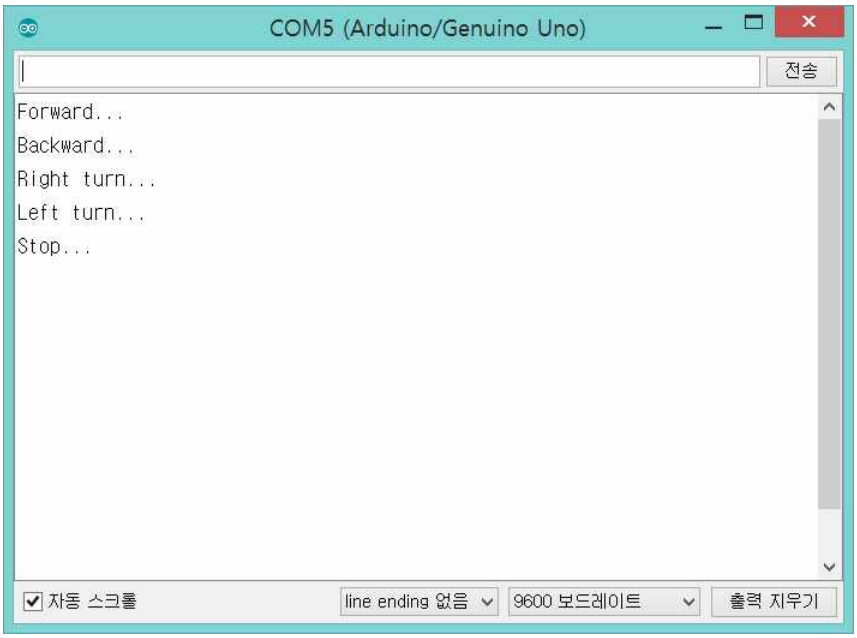



그림 12. 스케치 1 실행 결과

스케치 1에서는 모터의 속도 조절을 위해 SPEED 상수를 정의하여 사용하였으며 최대 속도로 동작시키기 위해서는 SPPED 변수의 값을 255로 설정하면 된다. 그림 11은 스케치 1을 테스트할 수 있는 아듀카의 상태를 나타낸다. 한 가지 주의할 점은 모터를 위한 배터리 전원이 아두이노에도 연결되어 있다는 점이다. 배터리는 모터 구동을 위해서도 사용되지만 아두이노의 전원 공급을 위해서도 사용된다. 스케치를 업로드 하기 위해 아두이노를 USB로 컴퓨터와 연결한 경우 아두이노는 USB를 통해 공급받는 전원으로 동작시킬 수 있다. 하지만 배터리 전원과 USB 전원의 GND는 연결시켜 주어야 하므로 배터리 전원을 반드시 아두이노에 연결시켜야 한다. 이처럼 배터리 전원이 아두이노에 연결되어 있으면 USB 연결을 제거하여도 아두이노는 동작한다.

## 2. 텍스트 LCD 연결

스케치 1에서는 아듀카의 동작 상태를 시리얼 모니터를 통해 출력하였다. 하지만 스케치가 업로드 된 후에 아듀카는 컴퓨터에 연결하지 않은 상태에서 사용하므로 메시지를 확인할 수 있는 방법이 필요하다. 간단한 메시지 출력이 필요하다면 텍스트 LCD를 사용할 수 있으며 I2C 방식의 텍스트 LCD는 연결선 수가 적으므로 쉽게 연결하여 사용할 수 있다. I2C 방식의 텍스트 LCD를 아두이노에 연결해 보자. 텍스트 LCD는 아날로그 4번(SDA)과 5번(SCL) 핀으로 연결한다.

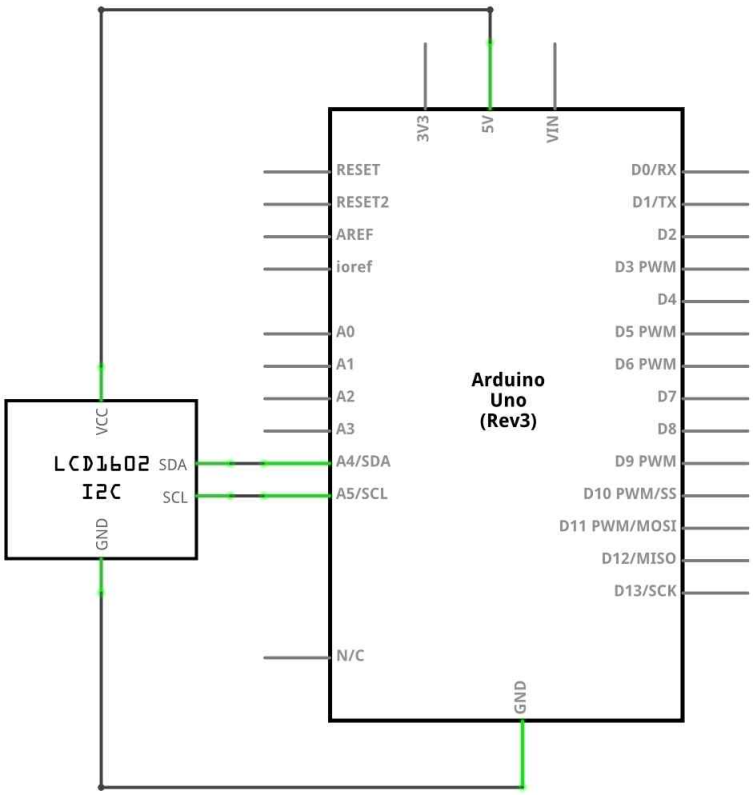



그림 13. I2C 방식 텍스트 LCD 연결 회로도

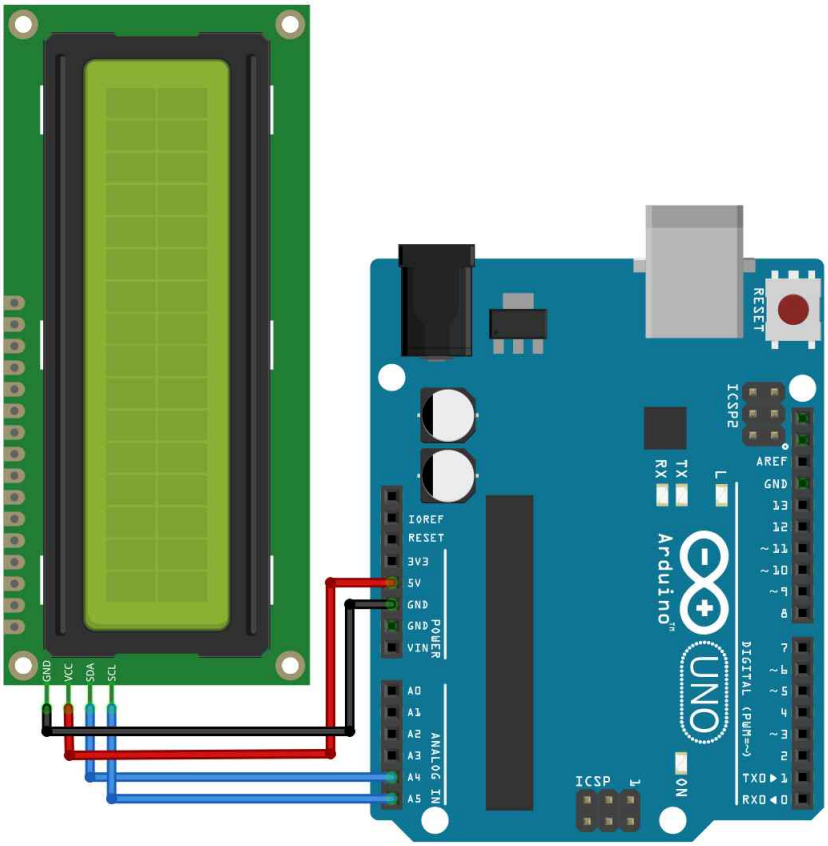



그림 14. I2C 방식 텍스트 LCD 연결 회로

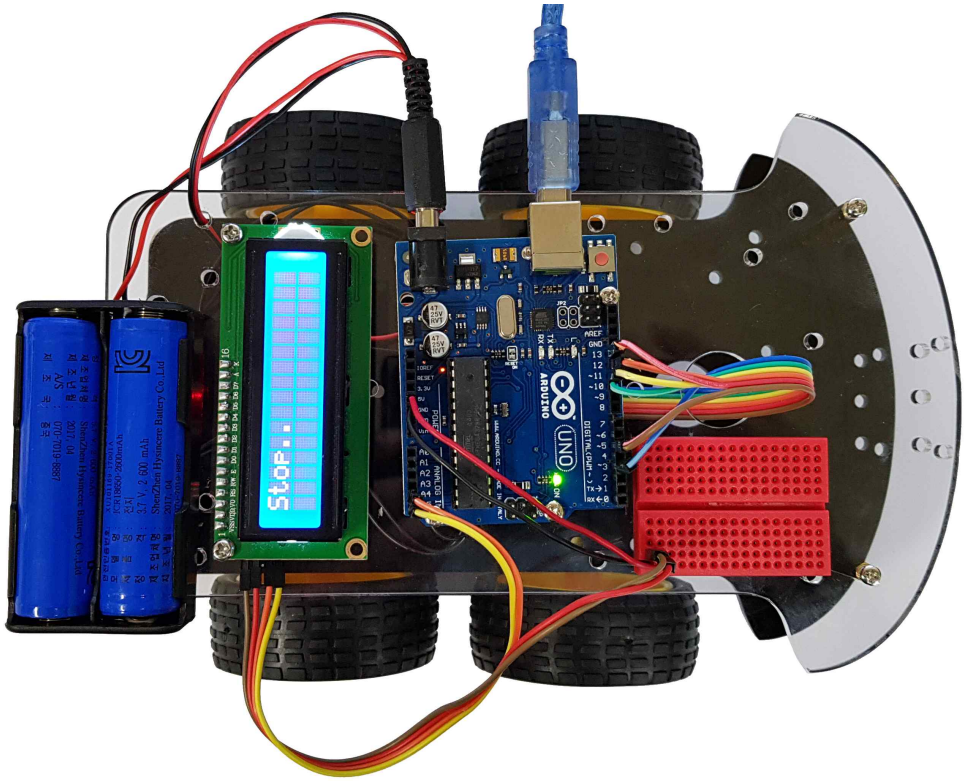



그림 15. I2C 방식 텍스트 LCD 연결

아듀카의 동작 상태를 텍스트 LCD로 출력하도록 스케치 1을 수정한 것이 스케치 2다. 또한 스케치 2에서는 스케치 1의 motorControl 함수를 보다 쉽게 이해할 수 있도록 여러 개의 함수로 나누어서 작성하였다.

스케치 2. 텍스트 LCD 테스트

```
#include <LiquidCrystal_I2C.h>

const int SPEED = 100;                  // 모터 회전 속도

int IN1 = 10, IN2 = 11;                 // 왼쪽 모터 회전 방향
int IN3 = 12, IN4 = 13;                 // 오른쪽 모터 회전 방향
int ENA = 5, ENB = 6;                   // 모터 활성화

LiquidCrystal_I2C lcd(0x27, 16, 2);     // I2C 방식 텍스트 LCD 객체 생성
```

```
void setup() {
  Serial.begin(9600);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);

  lcd.begin();                          // LCD 초기화
  lcd.print("Ready to start...");
  Stop();                               // 모터 정지 상태
}

void loop() {
  byte command;
  if(Serial.available()){               // 명령 수신
    command = Serial.read();
    motorControl(command);
  }
}

void motorControl(byte command) {
  if(command == 'F')            Forward();
  else if(command == 'B')       Backward();
  else if(command=='L')         LeftTurn();
  else if(command == 'R')       RightTurn();
  else if(command== 'S')        Stop();
}

void Forward() {
  analogWrite(ENA, SPEED);
  analogWrite(ENB, SPEED);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
```

```
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);

    lcd.clear();
    lcd.print("Forward...");
}

void Backward() {
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);

    lcd.clear();
    lcd.print("Backward...");
}

void LeftTurn() {
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);

    lcd.clear();
    lcd.print("Left turn...");
}

void RightTurn() {
    analogWrite(ENA, SPEED);
    analogWrite(ENB, SPEED);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
```

```
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);

    lcd.clear();
    lcd.print("Right turn...");
}

void Stop() {
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);

    lcd.clear();
    lcd.print("Stop...");
}
```

스케치 2를 업로드 하고 시리얼 모니터로 입력하는 명령에 따라 모터가 회전하고 텍스트 LCD에 상태가 출력되는지 확인해 보자.