

15. 블루투스로 DC 모터 제어하기

1. 소프트웨어 시리얼

아두이노에는 시리얼 통신을 위해 Serial 클래스가 마련되어 있으며 Serial 클래스는 디지털 0번과 1번 핀을 통해 시리얼 통신을 수행한다. 또한 아두이노 보드에는 USB 연결을 가상의 시리얼 포트와 연결시켜 주는 기능의 칩을 내장하고 있어 USB를 통해서도 컴퓨터와 시리얼 통신이 가능하다. 아두이노 프로그램에서 시리얼 모니터는 USB 연결을 통해 생성된 가상의 시리얼 포트와 연결되어 있다. 한 가지 주의할 점은 USB를 통한 시리얼 통신과 디지털 0번과 1번 핀을 통한 시리얼 통신은 동일한 핀을 사용하므로 동시에 사용할 수 없다는 점이다. 디지털 0번과 1번 핀 또는 USB를 통한 시리얼 통신은 UART(Universal Asynchronous Receiver Transmitter)로 알려진 방식으로 RS-232C 프로토콜에 정의된 내용 중에서 데이터 송수신을 위한 RXD (receive data), TXD (transmit data), GND (ground) 세 개의 핀만을 사용하여 통신을 수행한다. 그림 1은 UART 시리얼 통신을 이용하기 위해 컴퓨터와 아두이노를 연결한 예로 두 기기를 연결할 때 송신과 수신 핀은 서로 교차하여 연결하여야 한다. 단, 아두이노의 시리얼 통신은 5V 로직을 사용하므로 연결하고자 하는 기기가 5V 로직을 사용하지 않는 경우에는 별도의 변환 장치가 필요하다. 대표적으로 컴퓨터의 D-SUB 시리얼 커넥터는 반전된 12V 로직을 사용하므로 직접 연결하여서는 안된다.

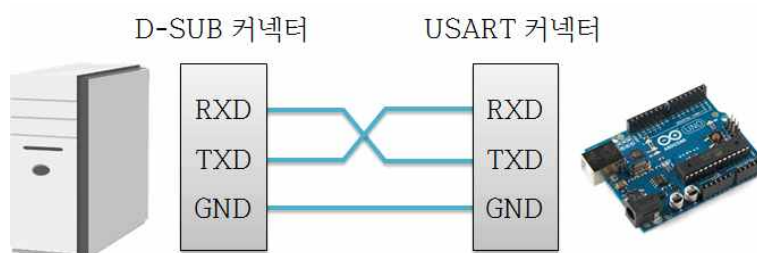


그림 1. PC와 마이크로컨트롤러의 UART 연결¹⁾

이처럼 디지털 0번과 1번 핀을 이용하여 수행하는 UART 시리얼 통신 기능을 하드웨어 시리얼이라고 하며 글자 그대로 하드웨어 수준에서 지원되는 시리얼 통신 기능이다. 아두이노와 연결하여 사용할 수 있는 장치 중에서는 시리얼 통신을 사용하는 주변 장치들을 흔히 볼 수

1) 그림 출처: <http://arduino.cc/en/Main/ArduinoBoardUno>

있다. 하지만 아두이노 우노의 경우 하나의 하드웨어 시리얼 통신만을 지원하므로 여러 개의 시리얼 장치를 연결하여 사용할 수 없다. 이를 보완하기 위해 아두이노에는 소프트웨어 시리얼 라이브러리(SoftwareSerial)가 기본으로 포함되어 있다. SoftwareSerial을 사용하면 하드웨어 시리얼 포트가 아닌 디지털 입출력 핀을 통해 시리얼 통신이 가능하며 아두이노 우노의 경우 모든 핀을 시리얼 통신을 위해 사용할 수 있다. 소프트웨어 시리얼은 글자 그대로 소프트웨어를 통해 시리얼 통신을 구현한 것이다. SoftwareSerial을 통해 여러 개의 시리얼 통신을 사용할 수 있지만 한 번에 하나의 시리얼 통신만을 수행할 수 있는 한계는 존재한다. SoftwareSerial을 사용하기 위해서는 스케치에 해당 헤더 파일인 SoftwareSerial.h를 포함시켜야 한다. 소프트웨어 시리얼의 경우 하드웨어 시리얼과 사용하는 핀의 번호만 다를 뿐 기본적인 사용 방법은 Serial 클래스를 사용하는 것과 동일하다.

2. 블루투스

블루투스(Bluetooth)는 1994년 에릭슨(Ericsson)이 개발한 개인 근거리 무선 통신 (Personal Area Network, PAN)을 위한 표준으로 RS-232를 대체하기 위한 저가격, 저전력의 무선 기술로 개발되었다. 블루투스는 기본적으로 10m 이내에서의 통신을 목표로 하며 이 범위는 100m까지 확장 가능하다. 블루투스는 프로파일(profile)이라 불리는 통신 방식을 지원하며 그 중 직렬 포트 프로파일(Serial Port Profile, SPP)은 직렬 통신과 거의 동일한 방식으로 무선 통신이 가능하도록 해준다. 사용하는 블루투스 모듈이 SPP를 지원한다면 소프트웨어 시리얼 라이브러리를 사용하여 블루투스 통신이 가능하다. 이 장에서 사용할 블루투스 모듈은 SPP를 지원하는 HC-06 블루투스 모듈이다. 블루투스는 마스터-슬레이브 구조를 가지며 HC-06 블루투스 모듈의 경우 마스터 모듈과 슬레이브 모듈이 별도로 존재하므로 실험을 위해서는 반드시 슬레이브 모듈을 구입하도록 한다.²⁾

2) HC-06 이외에도 SPP를 지원하는 블루투스 모듈이면 동작에는 문제가 없지만 이후 모듈의 설정 부분은 모듈의 종류에 따라 다를 수 있으므로 데이터시트를 참고하여 연결하도록 한다.

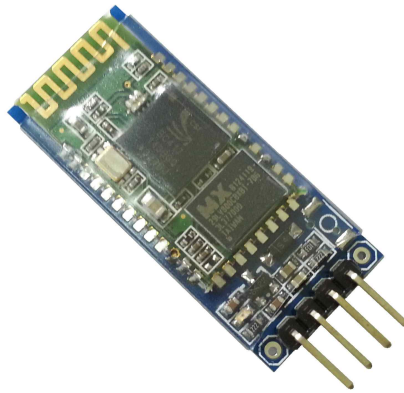


그림 2. HC-06 블루투스 모듈

스케치 1은 블루투스 모듈을 테스트하고 설정하기 위한 스케치로, 시리얼 모니터와 블루투스 모듈의 데이터를 상호 전달해주는 역할을 한다. 블루투스 모듈은 소프트웨어 시리얼을 통해 디지털 2번과 디지털 3번 핀으로 연결된다. 소프트웨어 시리얼은 매개변수를 (RX, TX) 순서로 지정하지만 블루투스 모듈과는 교차하여 연결하여야 하므로 2번 핀에는 블루투스 모듈의 TX를, 3번 핀에는 블루투스 모듈의 RX를 연결하여야 한다.

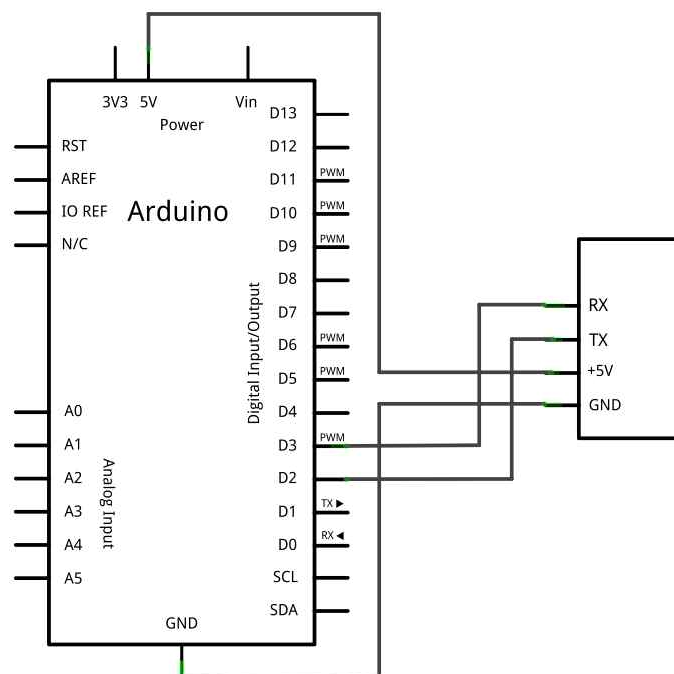


그림 3. HC-06 슬레이브 모듈 연결 회로도

스케치 1. 블루투스 모듈 테스트 및 설정

```

#include <SoftwareSerial.h>

// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결된다.
// 2 -> TX of Bluetooth Module
// 3 -> RX of Bluetooth Module
SoftwareSerial BTSerial(2, 3);

void setup()
{
    Serial.begin(9600);           // 컴퓨터와의 시리얼 통신 초기화
    BTSerial.begin(9600);        // 블루투스 모듈 초기화
}

void loop()
{
    // 블루투스 모듈 --> 아두이노 --> 시리얼 모니터
    if (BTSerial.available())
        Serial.write(BTSerial.read());

    // 시리얼 모니터 --> 아두이노 --> 블루투스 모듈
    if (Serial.available())
        BTSerial.write(Serial.read());
}

```

스케치 1을 업로드 한 후, 시리얼 모니터를 실행시키고 시리얼 모니터에서 'AT' 명령을 입력하였을 때 블루투스 모듈로부터 'OK' 문자열이 수신되어 시리얼 모니터에 표시되면 블루투스 모듈의 동작에는 이상이 없는 것으로 생각할 수 있다. 이 때 시리얼 모니터에서는 반드시 'No line ending' 옵션을 선택하여야 한다. 다른 옵션이 선택된 경우 'AT' 명령 뒤에 해당 문자가 첨부되어 전달되므로 'AT' 명령이 제대로 전달되지 않아 모듈이 정상 동작 중임에도 불구하고 'OK' 반환값을 받을 수 없다. 시리얼 모니터에서 'AT+NAMEDM-UStyle' 명령을 전송하여 블루투스 모듈의 이름을 'DM-UStyle'로 바꿀 수 있고, 'AT+PIN1111' 명령을 전송하여 비밀번호를 '1111'로 변경할 수 있다. 전송 속도는 'AT+BAUD4' 명령으로 변경 가능하며 지정한 숫자에 따른 전송 속도는 표 1과 같다.

숫자	전송 속도(Baud)	숫자	전송 속도(Baud)
1	1200	5	19200
2	2400	6	38400
3	4800	7	57600
4	9600	8	115200

표 1. 전송 속도 설정

그림 4는 블루투스 모듈의 이름, 비밀번호, 통신 속도를 설정한 예를 보여주고 있다. 이 때 통신 속도를 기본 값인 9600에서 다른 값으로 변경하면 스케치에서도 이에 맞게 속도를 변경하여 다시 프로그램을 업로드 해야 추가 설정이 가능하다.

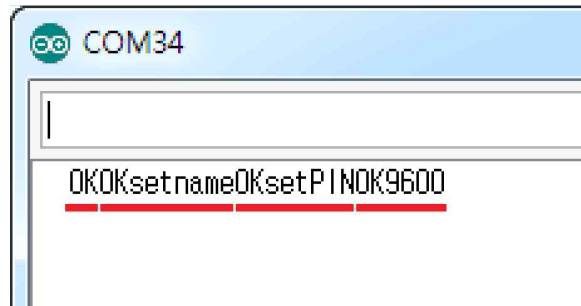


그림 4. 블루투스 모듈 설정

아두이노에 연결된 블루투스 모듈의 설정이 끝났으므로 이제 안드로이드 스마트폰과 아두이노 사이에 문자열 전송을 테스트해 보자. 안드로이드 스마트폰에는 블루투스 기능이 내장되어 있으므로 HC-06 슬레이브 모듈과 페어링하여 데이터를 주고받을 수 있다. 스마트폰용 앱을 작성하는 것은 이 책의 범위를 벗어나므로 블루투스의 SPP를 지원하는 앱을 스마트폰에 설치하여 HC-06 모듈의 동작을 확인하는 것만을 테스트한다. 아두이노에는 스케치 1을 업로드한 상태이면 된다. 블루투스 테스트용 어플리케이션 중 실험에서 사용한 어플리케이션은 'Bluetooth spp tools pro'로 플레이 스토어에서 무료로 다운받을 수 있다. 어플리케이션을 설치하고 실행시키면 주변의 블루투스 장치를 검색하여 보여준다.

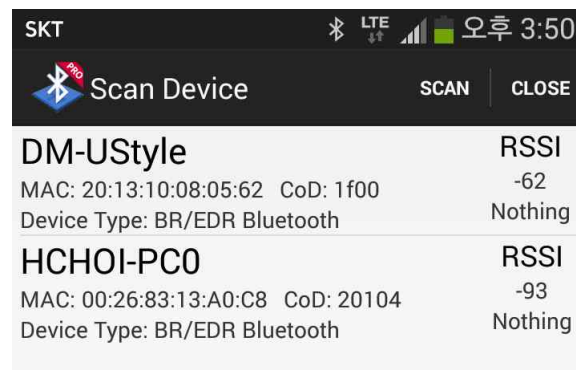


그림 5. 블루투스 모듈 검색

아두이노에 전원이 블루투스 모듈이 연결되고 전원이 공급되고 있다면 검색된 장치에서 'DM-UStyle'을 확인할 수 있다. 'DM-UStyle'을 선택하면 최초 연결 시 등록을 위해 PIN을 입력하는 화면이 나타난다. 설정한 PIN인 '1111'을 입력하면 등록이 완료된다.

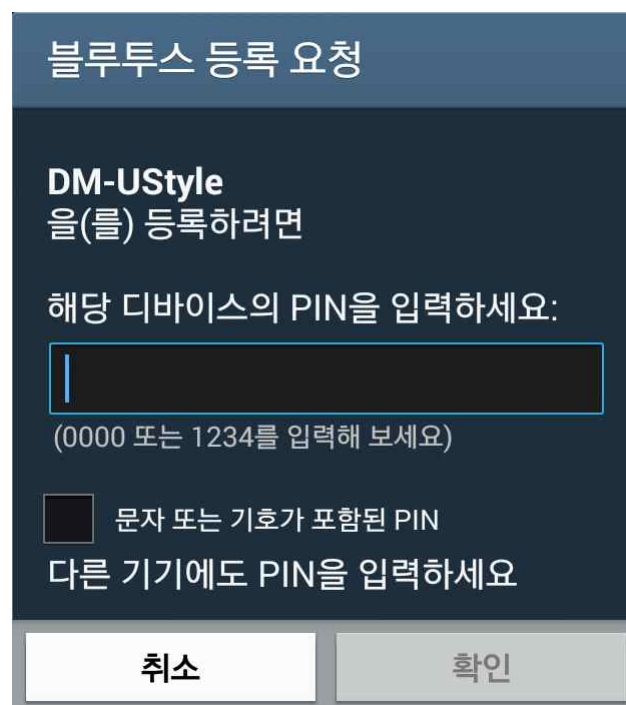


그림 6. 블루투스 등록

등록이 완료되었거나 이전에 이미 등록한 상태라면 'Connect' 버튼을 눌러 연결한다. 이 때 연결 모드는 'Byte stream mode'를 선택한다. 블루투스 연결이 완료되면 메시지를 입력하고 전송해 보자. 시리얼 모니터로 스마트폰에서 전송된 메시지가 출력되는가? 시리얼 모니터에

메시지를 입력하고 전송하면 스마트폰에서도 전송된 메시지가 출력된다.

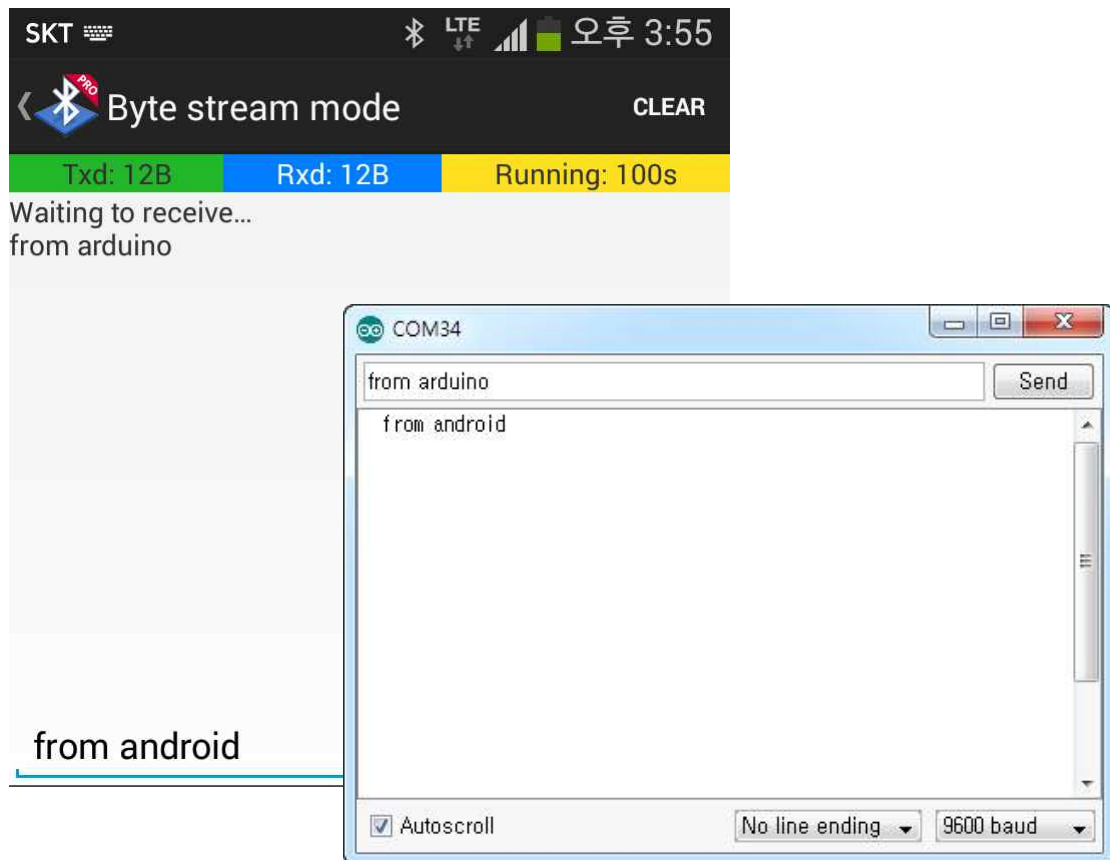


그림 7. 스마트폰과 아두이노의 메시지 송수신

3. 모터

모터는 움직이는 장치를 구현하기 위한 필수적인 부품으로 아두이노에서는 DC 모터, 스텝핑 모터, 서보 모터 등이 많이 사용된다.

- DC (Direct Current) 모터 : 축이 연속적으로 회전하는 형태의 모터로 전원이 끊어지는 경우에만 정지한다. 정지 시에는 관성으로 인해 정확한 정지 위치를 지정하기 어렵다.
- 스텝핑 (stepping) 모터 : 전원이 공급되면 축은 일정 각도를 회전하고 멈춘다. 따라서 축을 연속적으로 회전시키기 위해서는 모터로 펄스열이 전달되어야 한다. 스텝핑 모터에는 고유의 분할 각도가 존재하여 이 분할 각도에 의해 제어된다. 분할 각도가 1도인 경우 모터를 10도 회전시키기 위해서는 10개의 펄스를 전송해주어야 하며 9.5도 회전은 불가능하다.

- 서보 (servo) 모터 : DC 모터에 귀환 회로를 추가하여 정확한 위치 제어가 가능하도록 구성된 모터이다. 가격 면에서는 귀환 제어 회로로 인해 다른 모터에 비해 비싸지만 귀환 제어를 통해 정밀 제어가 가능한 장점이 있다.

이 장에서 사용할 모터는 DC 모터이다. DC 모터는 두 개의 연결선을 가지며 Vcc와 GND를 연결하는 것만으로 동작하고 연결 방향에 따라 정회전 또는 역회전이 결정된다. 따라서 하나의 선을 GND로 연결하고 다른 한 선은 디지털 핀에 연결하여 모터의 on/off 제어가 가능하다. 하지만 DC 모터는 일단 연결이 되고 나면 한 방향으로만 회전할 수 있는 문제점이 있다. 또 다른 문제점은 일반적으로 DC 모터 구동을 위해 많은 전류가 필요하다는 점이다. 아두이노의 출력 핀에서 공급할 수 있는 최대 전류는 40mA이고 이는 소형 DC 모터의 구동 전류에도 미치지 못하는 경우가 많으므로 대부분의 DC 모터를 구동할 수 없는 문제점이 있다. 이러한 문제점들을 해결하기 위해 아두이노에서는 모터 제어 모듈을 사용하여 DC 모터의 회전 방향과 회전 속도를 조절하고, 전용 전원을 통해 DC 모터에 별도로 전원을 공급한다. 모터 제어를 위해서는 전용 모터 드라이버 칩을 사용하며, 모터 드라이버 칩에는 아두이노 공식 모터 쉴드에 사용된 L298을 비롯하여 모터 구동을 위해 필요한 전압과 전류량에 따라 다양한 종류의 칩이 존재한다. 이 장에서 사용할 모터 제어 모듈은 아두이노 공식 모터 쉴드에서 사용된 칩과 동일한 기능의 L298N 칩을 사용한 모터 제어 모듈이다.

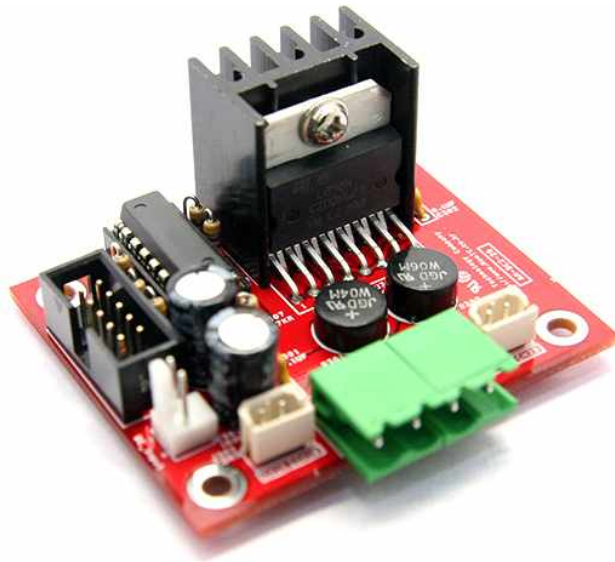


그림 8. 모터 제어 모듈³⁾

3) 이미지 출처: <http://www.newtc.co.kr>

L298N 칩은 5V ~ 45V의 전압에서 최대 2A의 전류를 공급할 수 있는 대용량 모터 컨트롤러 칩으로 두 개의 DC 모터를 제어할 수 있다. 모터 제어 모듈의 연결 방법은 그림 9와 같다.

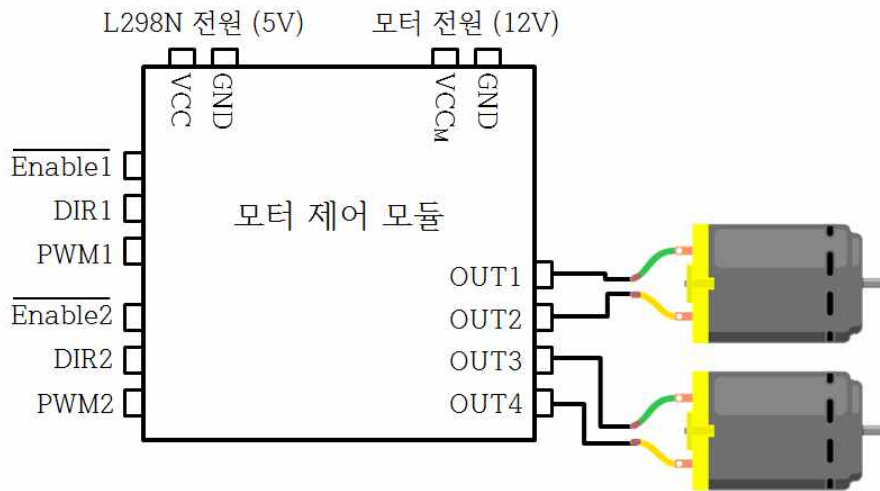


그림 9. 모터 제어 모듈 연결

모터 제어 모듈에는 4개의 출력 라인(OUT1 ~ OUT4)이 있으며 모터 2개를 연결할 수 있다. 모터 전용의 외부 전원은 12V를 연결하여야 하며 이는 L298N 구동을 위한 5V 전원과는 별도로 주어진다. 모터 제어를 위해서는 모터 하나에 3개의 입력(Enable, DIR, PWM)을 필요로 한다. Enable 신호를 LOW 값을 출력한 상태에서만 모터 구동이 가능하다. DIR 핀은 모터 회전 방향 제어를 위해 사용되며 LOW인 경우 역방향, HIGH인 경우 정방향 회전을 나타낸다. PWM 핀은 듀티 사이클에 따라 속도 조절을 위해 사용된다.

아두이노와 모터 제어 모듈의 연결은 그림 10과 같으며 표 2에 연결 핀을 요약하여 나타내었다. 속도 조절을 위해서는 펄스 폭 변조 (Pulse Width Modulation, PWM) 신호를 사용하므로 PWM 출력이 가능한 핀에 연결하여야 한다.

	모터 #1		모터 #2	
	모터 모듈 기호	아두이노 핀 번호	모터 모듈 기호	아두이노 핀 번호
속도	PWM1	9	PWM2	6
활성화	Enable1	10	Enable2	7
방향	DIR1	11	DIR2	8

표 2. 아두이노와 모터 제어 모듈 연결 핀

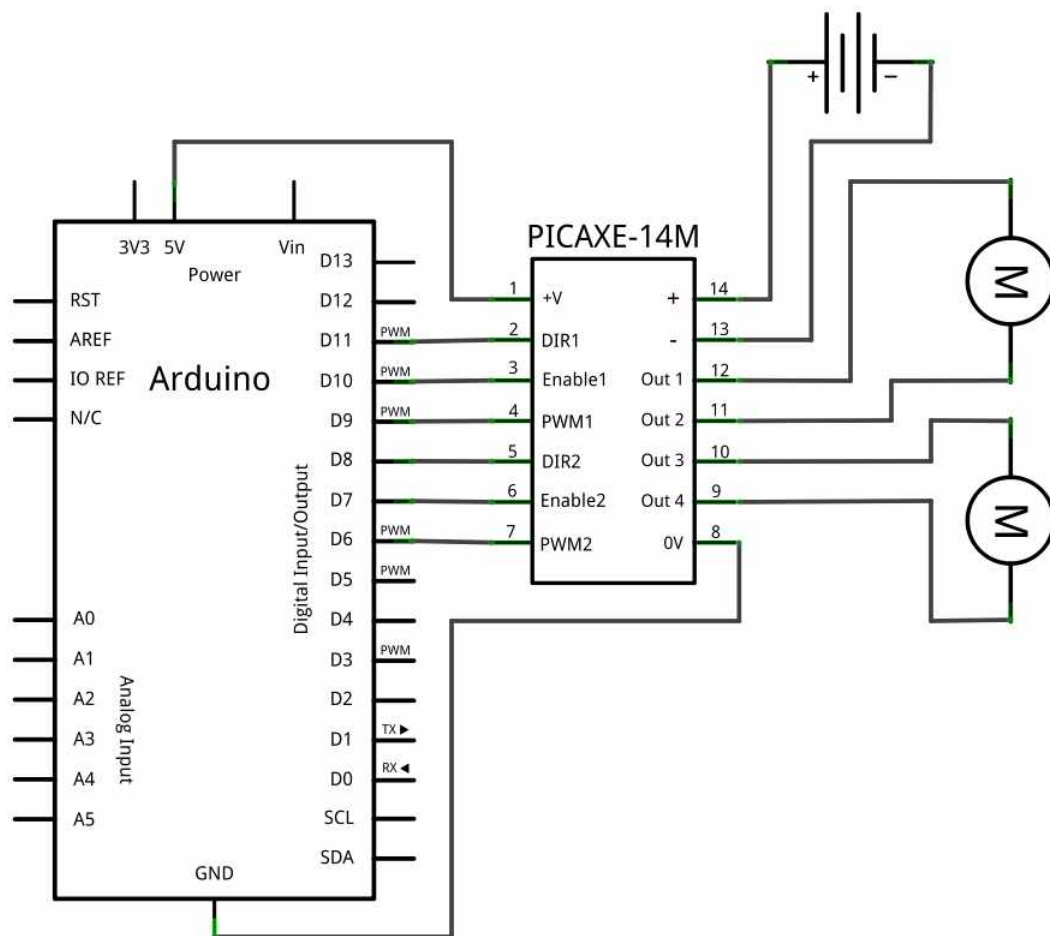


그림 10. 모터 제어 모듈 연결 회로도

스케치 2는 두 개의 모터를 연결하고 하나씩 번갈아 가면서 정방향과 역방향으로 최저 속도에서 최고 속도로 변화시키면서 모터를 구동시키는 스케치의 예이다.

스케치 2. 모터 제어

```
int PWM1 = 9;           // 모터 1 제어를 위한 연결 핀
int Enable1 = 10;
int DIR1 = 11;

int PWM2 = 6;           // 모터 2 제어를 위한 연결 핀
int Enable2 = 7;
int DIR2 = 8;
```

```
void setup(){
    // 모터 제어를 위한 핀들을 출력으로 설정
    pinMode(PWM1, OUTPUT);
    pinMode(Enable1, OUTPUT);
    pinMode(DIR1, OUTPUT);
    pinMode(PWM2, OUTPUT);
    pinMode(Enable2, OUTPUT);
    pinMode(DIR2, OUTPUT);

    digitalWrite(Enable1, HIGH);           // 초기 상태는 비활성화 상태
    digitalWrite(Enable2, HIGH);
}

void loop(){
    digitalWrite(Enable1, LOW);           // 모터 1 활성화
    digitalWrite(DIR1, HIGH);             // 정방향 회전
    for(int i = 0; i < 256; i++){         // 속도 조절을 위한 PWM 신호 출력
        analogWrite(PWM1, i);
        delay(25);
    }

    analogWrite(PWM1, 0);                 // 모터 정지
    delay(1000);
    digitalWrite(DIR1, LOW);              // 역방향 회전

    for(int i = 0; i < 256; i++){         // 속도 조절을 위한 PWM 신호 출력
        analogWrite(PWM1, i);
        delay(25);
    }

    analogWrite(PWM1, 0);                 // 모터 정지
    digitalWrite(Enable1, HIGH);          // 모터 1 비활성화

    digitalWrite(Enable2, LOW);
    digitalWrite(DIR2, HIGH);
```

```

    for(int i = 0; i < 256; i++){
        analogWrite(PWM2, i);
        delay(25);
    }

    analogWrite(PWM2, 0);
    delay(1000);
    digitalWrite(DIR2, LOW);

    for(int i = 0; i < 256; i++){
        analogWrite(PWM2, i);
        delay(25);
    }

    analogWrite(PWM2, 0);
    digitalWrite(Enable2, HIGH);

    while(true);
}

```

모터 제어를 위해 라이브러리를 작성해 보자. 클래스의 이름은 `_DMUStyleMotor`로 하고 전역 객체로 `DMUStyleMotor`를 생성하여 전역 객체를 통해 모터를 제어한다. 모터 제어를 위해 아두이노와 연결할 핀은 스케치 1에서 사용한 핀을 기본으로 사용하도록 생성자에서 지정하였으며, 다른 핀을 사용하고자 하는 경우를 위해 핀 번호를 바꿀 수 있도록 `SetPins` 함수를 제공하고 있다. 다른 핀을 사용하고자 하는 경우, 속도 조절을 위해서는 PWM 신호 출력이 가능한 핀을 PWM1과 PWM2에 지정하여야 한다는 점에 유의하여야 한다. 이외에 모터를 활성화시키는 `Enable`과 비활성화 시키는 `Disable` 함수, 그리고 모터를 지정한 방향과 속도로 회전하도록 하는 `Rotate`와 정지시키는 `Stop` 함수를 작성한다. 모터 제어 라이브러리는 `DMUStyleMotor.h` 헤더 파일과 `DMUStyleMotor.cpp` 소스 파일로 구성되며, 스케치북 디렉터리의 'libraries' 디렉터리 아래에 `DMUStyleMotor` 디렉터를 생성하고 그 아래에 복사하면 된다.

DMUStyleMotor.h

```

#ifndef _DMUStyleMotor_
#define _DMUStyleMotor_

#include "Arduino.h"

#define MOTOR1      1      // 모터 선택
#define MOTOR2      2

#define DIRECTION1  1      // 회전 방향 선택
#define DIRECTION2  2

class _DMUStyleMotor {
private:
    int DIR1, Enable1, PWM1;          // 모터 1 연결 핀
    int DIR2, Enable2, PWM2;          // 모터 2 연결 핀
public:
    _DMUStyleMotor();                // 기본 핀을 사용하는 생성자
    // 기본 핀 설정과 다른 핀을 사용하고자 하는 경우 핀 변경 함수
    void SetPins(int dir1, int enable1, int pwm1, int dir2, int enable2, int pwm2);
    void Init();                      // 핀 모드 설정 함수
    void Rotate(int motor, int dir, int speed); // 회전
    void Stop(int motor);             // 정지
    void Enable(int motor);           // 활성화
    void Disable(int motor);          // 비활성화
};

extern _DMUStyleMotor DMUStyleMotor;

#endif

```

DMUStyleMotor.cpp

```

#include "Arduino.h"
#include "DMUStyleMotor.h"

```

```
_DMUStyleMotor DMUStyleMotor:                // 전역 객체

_DMUStyleMotor::_DMUStyleMotor()
{
    PWM1 = 9;
    Enable1 = 10;
    DIR1 = 11;

    PWM2 = 6;
    Enable2 = 7;
    DIR2 = 8;

    Init();          // 핀 모드 설정
}

void _DMUStyleMotor::SetPins(int dir1, int enable1, int pwm1,
                             int dir2, int enable2, int pwm2)
{
    PWM1 = pwm1;
    Enable1 = enable1;
    DIR1 = dir1;

    PWM2 = pwm2;
    Enable2 = enable2;
    DIR2 = dir2;

    Init();
}

void _DMUStyleMotor::Init()
{
    pinMode(PWM1, OUTPUT);
    pinMode(Enable1, OUTPUT);
    pinMode(DIR1, OUTPUT);

    pinMode(PWM2, OUTPUT);
```

```
pinMode(Enable2, OUTPUT);
pinMode(DIR2, OUTPUT);

digitalWrite(Enable1, HIGH);          // 비활성화 상태
digitalWrite(Enable2, HIGH);
}

// 모터를 지정한 방향과 속도로 회전시킨다.
// motor : 모터 선택 (1 또는 2)
// dir : 방향 (1 또는 2)
// speed : 속도 (0~255)
void _DMUStyleMotor::Rotate(int motor, int dir, int speed)
{
    switch(motor){
    case MOTOR1:
        if(dir == DIRECTION1) digitalWrite(DIR1, HIGH);
        else digitalWrite(DIR1, LOW);
        analogWrite(PWM1, speed);
        break;
    case MOTOR2:
        if(dir == DIRECTION1) digitalWrite(DIR2, HIGH);
        else digitalWrite(DIR2, LOW);
        analogWrite(PWM2, speed);
        break;
    }
}

// 모터 회전을 정지시킨다.
void _DMUStyleMotor::Stop(int motor)
{
    switch(motor){
    case MOTOR1:
        analogWrite(PWM1, 0);
        break;
    case MOTOR2:
        analogWrite(PWM2, 0);
    }
```

```
        break;
    }
}

// 모터를 활성화 상태로 설정한다.
void _DMUStyleMotor::Enable(int motor)
{
    switch(motor){
    case MOTOR1:
        digitalWrite(Enable1, LOW);
        break;
    case MOTOR2:
        digitalWrite(Enable2, LOW);
        break;
    }
}

// 모터를 비활성화 상태로 설정한다.
void _DMUStyleMotor::Disable(int motor)
{
    switch(motor){
    case MOTOR1:
        digitalWrite(Enable1, HIGH);
        break;
    case MOTOR2:
        digitalWrite(Enable2, HIGH);
        break;
    }
}
```

모터 컨트롤 라이브러리를 사용하여 스케치 2와 동일한 동작을 하는 스케치를 작성해 보자. 먼저 스케치 → 라이브러리 가져오기... → DMUStyleMotor를 선택하여 DMUStyleMotor.h 파일을 포함시킨다. 헤더 파일이 포함되면 전역 객체인 DMUStyleMotor가 기본 핀을 사용하도록 설정되므로 별도의 설정은 필요하지 않으며 바로 사용할 수 있다. 스케치 3은 스케치 2를 모터 컨트롤 라이브러리를 사용하여 작성한 예이다.

스케치 3. 모터 컨트롤 라이브러리를 사용한 모터 제어

```
#include <DMUStyleMotor.h>

void setup(){ } // 헤더를 포함시키면 초기 설정 작업은 필요하지 않다.

void loop(){
    DMUStyleMotor.Enable(MOTOR1); // 모터 1 활성화

    for(int i = 0; i < 256; i++){ // 모터 1 정회전
        DMUStyleMotor.Rotate(MOTOR1, DIRECTION1, i);
        delay(25);
    }

    DMUStyleMotor.Stop(MOTOR1); // 모터 1 정지
    delay(1000);

    for(int i = 0; i < 256; i++){ // 모터 1 역회전
        DMUStyleMotor.Rotate(MOTOR1, DIRECTION2, i);
        delay(25);
    }

    DMUStyleMotor.Stop(MOTOR1); // 모터 1 정지
    DMUStyleMotor.Disable(MOTOR1); // 모터 1 비활성화

    DMUStyleMotor.Enable(MOTOR2);

    for(int i = 0; i < 256; i++){
        DMUStyleMotor.Rotate(MOTOR2, DIRECTION1, i);
        delay(25);
    }

    DMUStyleMotor.Stop(MOTOR2);
    delay(1000);
}
```

```

    for(int i = 0; i < 256; i++){
        DMUStyleMotor.Rotate(MOTOR2, DIRECTION2, i);
        delay(25);
    }

    DMUStyleMotor.Stop(MOTOR2);
    DMUStyleMotor.Disable(MOTOR2);

    while(true);

}

```

4. 안드로이드 폰으로 모터 제어하기

안드로이드 폰으로 모터를 제어해 보자. 안드로이드 폰에는 Bluetooth spp tools pro 어플리케이션을 설치하고 어플리케이션에서는 'L', 'R', 'S' 세 종류의 명령만을 전송한다. 'L'은 모터 1을 최대 속도의 50% 속도로 정회전 시키도록 하며, 'R'은 모터 2를 최대 속도의 50% 속도로 역회전 시키도록 한다. 'S'는 모터 회전을 멈춘다. 그림 3과 그림 10을 참고하여 블루투스 모듈과 모터 제어 모듈을 연결하고 소프트웨어 시리얼을 통해 명령이 전송되면 그에 맞게 모터를 회전시키도록 해보자.

스케치 4. 안드로이드 폰으로 모터 제어

```

#include <SoftwareSerial.h>
#include <DMUStyleMotor.h>

// SoftwareSerial(RX, TX) 형식으로 블루투스 모듈과 교차하여 연결된다.
// 2 -> TX of Bluetooth Module
// 3 -> RX of Bluetooth Module
SoftwareSerial BTSerial(2, 3);

void setup(){
    BTSerial.begin(9600);           // 블루투스 연결 초기화

```

```
Serial.begin(9600);                // 시리얼 포트 초기화

DMUStyleMotor.Enable(MOTOR1);      // 모터를 활성화 상태로 지정
DMUStyleMotor.Enable(MOTOR2);
}

void loop(){
  if(BTSerial.available()){        // 블루투스로 명령어가 수신된 경우
    char command = BTSerial.read(); // 명령어 읽기
    Serial.println(String("Command \"") + command + "\" received.");

    switch(command){
      case 'R':
      case 'r':                      // 모터 1 회전
        Serial.println("Motor 1 Start...");
        DMUStyleMotor.Rotate(MOTOR1, DIRECTION1, 127);
        break;
      case 'L':
      case 'l':                      // 모터 2 회전
        Serial.println("Motor 2 Start...");
        DMUStyleMotor.Rotate(MOTOR2, DIRECTION1, 127);
        break;
      case 'S':
      case 's':                      // 모터 정지
        Serial.println("Motor STOP !!");
        DMUStyleMotor.Stop(MOTOR1);
        DMUStyleMotor.Stop(MOTOR2);
        break;
    }
  }
}
```

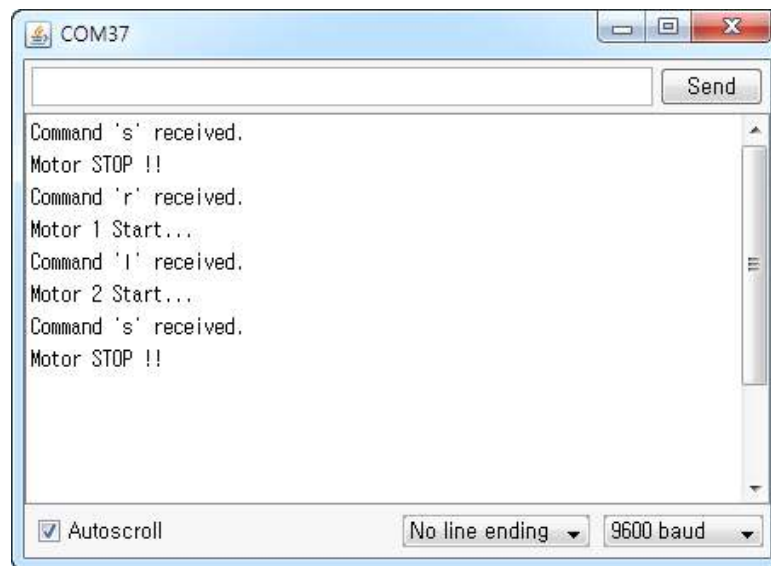


그림 11. 스케치 4 실행 결과