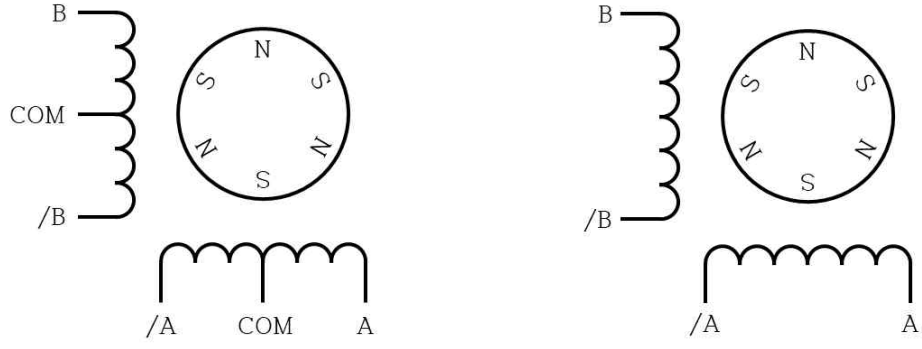
APPENDIX 05

스테핑 모터

필요한 부품

- 아두이노 우노
- 28BYJ-48 단극 스테핑 모터
- 스테핑 모터 드라이버 모듈(28BYJ-48 전용, ULN2003 칩 사용)

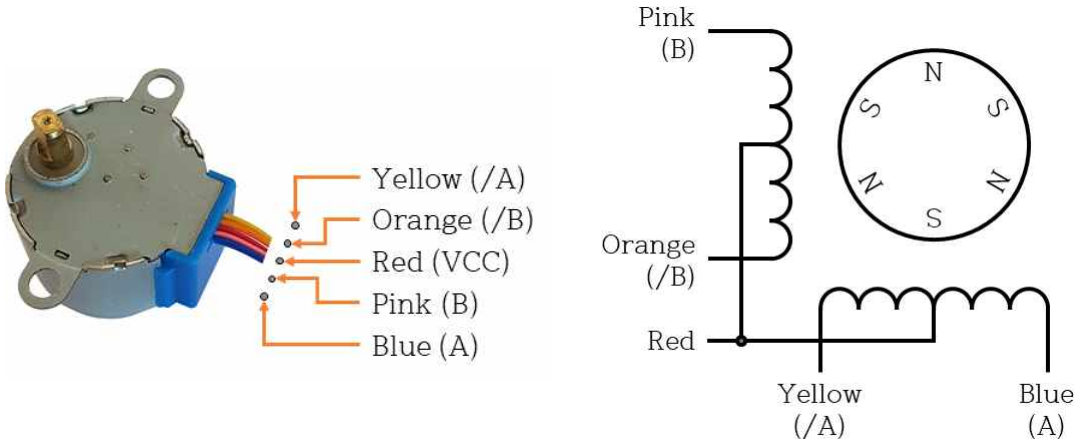
스테핑 모터는 코일에 전원을 가하는 방식에 따라 단극(unipolar) 모터와 양극(bipolar) 모터로 나눌 수 있으며 [그림 1]은 각 모터의 구조를 나타낸다. [그림 1]에서 알 수 있듯이 일반적으로 단극 모터는 6개의 연결선을 가지고 양극 모터는 4개의 연결선을 가지므로 연결선의 수로 구별할 수 있다.



(a) 단극 모터 (b) 양극 모터

[그림 1] 스테핑 모터의 구조

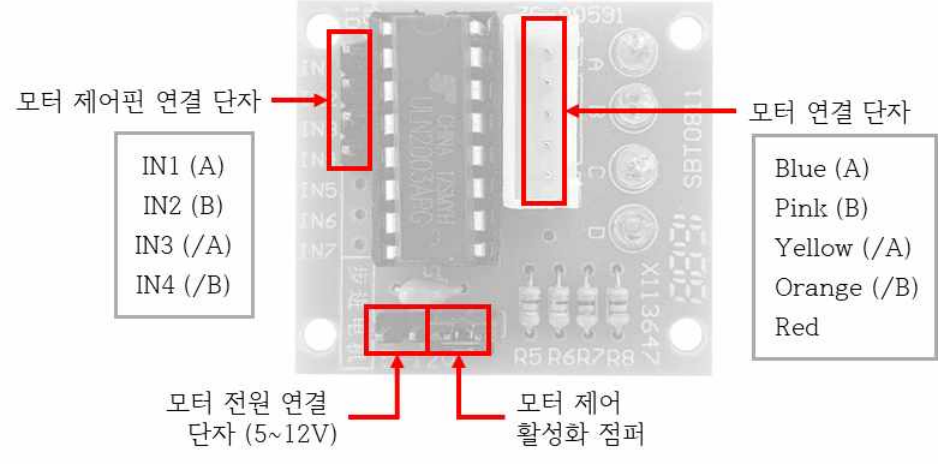
[그림 2]는 이 장에서 사용하는 28BYJ-48 스텝 모터를 나타낸다. 단극 모터이지만 일반적이 단극 모터와 달리 내부에서 공통 핀 2개를 연결해 놓아 5개의 연결선만을 가지고 있다. 28BYJ-48 모터는 감속 기어를 포함하고 있어 1회전을 위한 최대 스텝 수는 4,096이지만, 2,048 스텝에 1회전 하는 방식이 흔히 사용된다.



(a) 연결선 (b) 코일

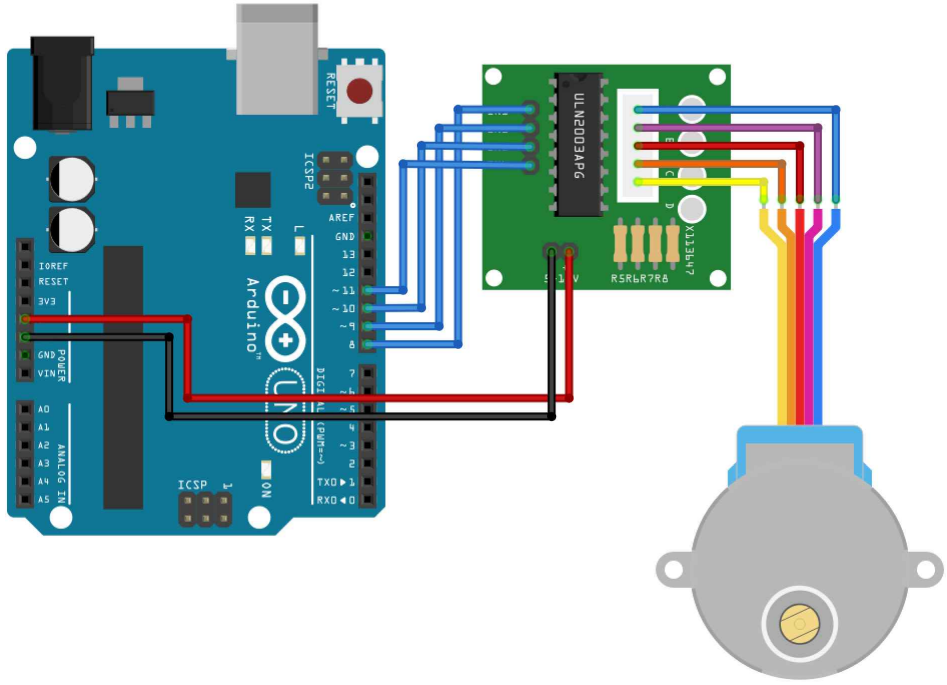
[그림 2] 28BYJ-48 단극 스텝 모터

28BYJ-48 모터는 [그림 3]과 같은 전용의 제어 모듈을 함께 사용하는 것이 일반적이다. 양극 모터와 달리 단극 모터는 가해지는 전원의 방향을 제어할 필요가 없고 모터 전용 전원의 연결 유무만을 제어하면 되므로 H- 브리지 회로를 사용하지 않아도 된다. [그림 4]의 모터 제어 모듈 역시 ULN2003의 달링턴 트랜지스터를 사용한다.



[그림 3] 모터 제어 모듈

모터를 [그림 4]와 같이 아두이노 우노에 연결하자.



[그림 4] 28BYJ-48 모터 연결

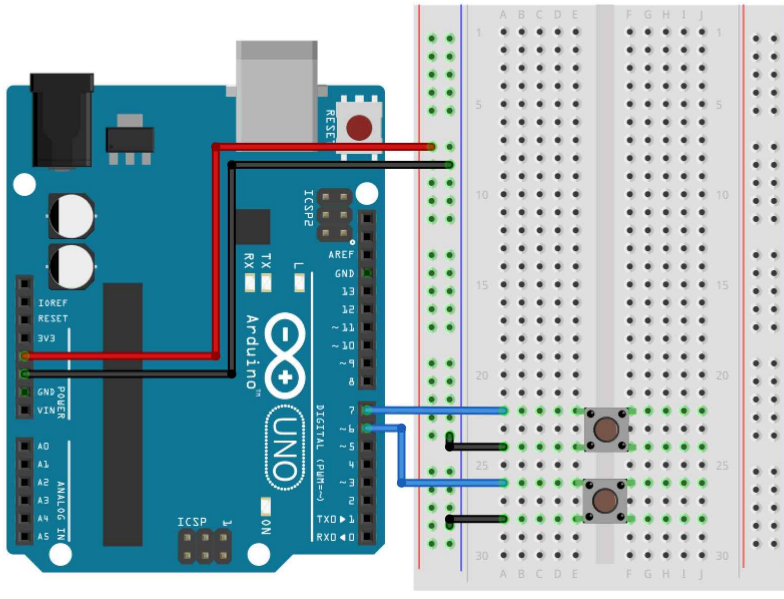
모터 제어를 위해서는 아두이노의 기본 라이브러리 중 하나인 Stepper 라이브러리를 사용하면 된다. Stepper 라이브러리는 객체 생성 후 **setSpeed** 함수로 분당 회전수를 설정한 후에 **step** 함수로 회전할 양을 지정하면 된다. [코드 1]은 28BYJ-48 모터를 앞뒤로 1바퀴씩 회전하는 스케치다. [그림 4]와 같이 아두이노의 5V 전원을 사용하는 경우 분당 회전수를 크게 하면 전력 부족으로 모터가 움직이지 않을 수 있으며 이 경우에는 모터 전용 전원을 사용해야 한다.

코드 1 **28BYJ-48 모터 제어** APPENDIX_05-1_sttepping_test.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 2048;    // 1회전을 위한 스텝 수

// 모터 드라이브에 연결된 핀 IN1, IN3, IN2, IN4 => A, /A, B, /B 순서
Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {
```

```
  myStepper.setSpeed(15);                  // 분당 회전수
}

void loop() {
  myStepper.step(stepsPerRevolution);   // 1바퀴 정회전 : 양의 스텝 수
  delay(500);

  myStepper.step(-stepsPerRevolution); // 1바퀴 역회전 : 음의 스텝 수
  delay(500);
}
```

[그림 4]의 회로에 [그림 5]와 같이 6번과 7번 핀에 버튼을 추가하여 모터의 회전을 시작하고 멈추는 기능을 추가해 보자. 버튼은 내장 풀업 저항을 사용하므로 별도의 저항을 사용하지 않으며, 모터는 1분에 1회전 하도록 속도를 설정하여 스톱워치로 동작하도록 한다.



[그림 5] 버튼 연결

[코드 2]는 시작과 정지 기능이 있는 스톱워치로 동작하도록 하는 스케치다.

코드 2 **1분 스톱워치** APPENDIX_05-2_chrnograph.ino

```
#include <Stepper.h>

const int STEPS_PER_REV = 2048;         // 1회전을 위한 스텝 수

// 모터 드라이브에 연결된 핀 IN1, IN3, IN2, IN4 => A, /A, B, /B 순서
Stepper chronograph(STEPS_PER_REV, 8, 10, 9, 11);

// 1분에 1회전하기 위한 스텝 사이의 시간 간격 :
// 60초 * 1000밀리초/초 / 2048스텝
const float MS_PER_STEP = 60000.0 / 2048;

int BTN_START = 7;                      // 시작 버튼
int BTN_STOP = 6;                       // 정지 버튼

// 시간 및 스텝 수 관리를 위한 변수
unsigned long last_time = 0;
unsigned long curr_time = 0;
int steps_taken = 0;

void setup() {
  chronograph.setSpeed(15);             // 분당 회전수

  pinMode(BTN_START, INPUT_PULLUP);
  pinMode(BTN_STOP, INPUT_PULLUP);
}

void loop() {
  // 무한루프로 시작 버튼을 누를 때까지 대기한다.
  // 마지막의 세미콜론은 주어진 조건을 만족하지 않을 때까지
  // 조건 검사를 계속 시행하도록 한다.
  while (digitalRead(BTN_START) == HIGH);

  last_time = millis();                 // 스톱워치 시작 시간 얻기
```

```
  // 정지 버튼을 누르거나 1분이 경과할 때까지 반복
  while (digitalRead(BTN_STOP) == HIGH && steps_taken < STEPS_PER_REV) {
    curr_time = millis();
    // 스텝 간격 시간이 경과했을 때 1스텝 회전
    // 한 스텝 간격이 정수가 아닌 실수이므로 전체 스텝 수를 기준으로 계산
    if (curr_time - last_time >= MS_PER_STEP * (steps_taken + 1)) {
      chronograph.step(1);              // 1스텝 회전
      steps_taken++;                    // 스텝 수 관리 변숫값 증가
    }
  }

  // 이 단계는 정지 버튼을 누르거나 1분이 경과했을 때 실행된다.
  // 완전히 1회전이 이루어지지 않은 경우 시계바늘을 시작 위치로 옮긴다.
  if (steps_taken < STEPS_PER_REV) chronograph.step(-steps_taken);
  // 스텝 수 관리 변수 초기화
  steps_taken = 0;
}
```