사물인터넷

Chapter 7. 모터 제어

1. 모터 소개
2. 서보 라이브러리와 함수의 사용
3. 스테퍼 모터 제어
4. DC모터 제어

학습목표

- 각종 모터의 원리와 쓰임새를 이해할 수 있다.
- 서보 라이브러리를 설치할 수 있다.
- 서보 라이브러리의 함수를 이해하고 적절하게 활용할 수 있다.
- 각종 모터를 아두이노 보드에 연결할 수 있다.
- LCD 표시장치의 행과 열의 좌표를 이해하고 적절한 위치에 원하는 문자를 출력할 수 있다.

1. 모터 소개

- 모터는 라틴어의 "moto"(움직인다)에서 온 단어로 플레밍의 왼손법칙에 따라 전기적 에너지를 운동(기계) 에너지로 변환하는 장치

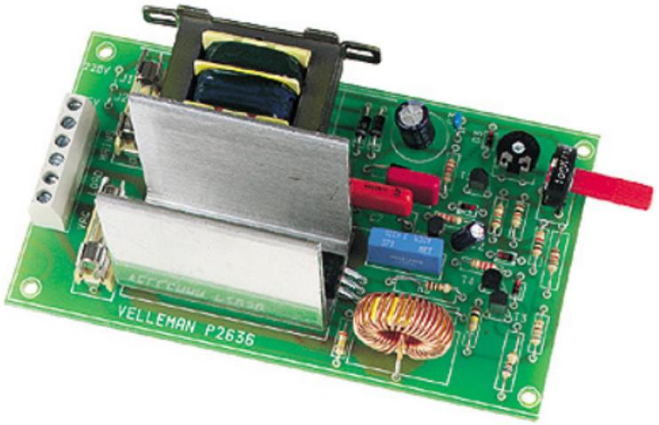
그림 7.1 모터의 정의

1. 모터 소개

- AC 모터

  - 우리나라에서 일반적으로 사용하는 220V 전기인 교류 전원을 사용하는 모터
  - 장점 : 소음과 진동이 적고 수명이 반영구적이며 안정적인 성능을 가지고 있음
  - 단점 : 속도와 방향 제어가 어려움

AC모터

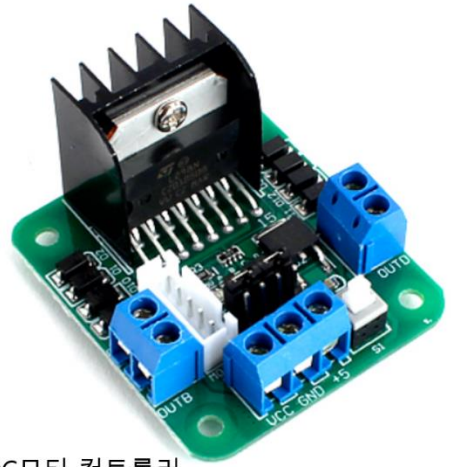


AC모터 컨트롤러

1. 모터 소개

- DC 모터
  - 배터리와 같은 직류 전원을 사용
  - 고정자로 영구자석을 사용하고 전기자로 코일을 사용
  - 전기자에 흐르는 전류의 방향을 전환시키면서 자력의 밀어내는 힘이나 끌어당기는 힘으로 회전력을 생성
  - 적당한 전력만 공급해주면 회전하므로 RC자동차, 무선 조종용 장난감 등 여러 용도로 많이 사용
  - 전기를 끊으면 멈추게 되는데, 이때 관성에 의하여 정확한 정지 위치를 지정하기는 어려움

DC모터



DC모터 컨트롤러

1. 모터 소개

- 스테퍼 모터
  - 아날로그시계의 초침처럼 고유의 분할 각도가 있고 이 각도에 따라 단계적으로(step-by-step) 움직이는 모터
  - 스탭모터는 360도 회전이 가능하며, 회전수, 정확한 각도, 방향 등을 제어해줄 수 있다.

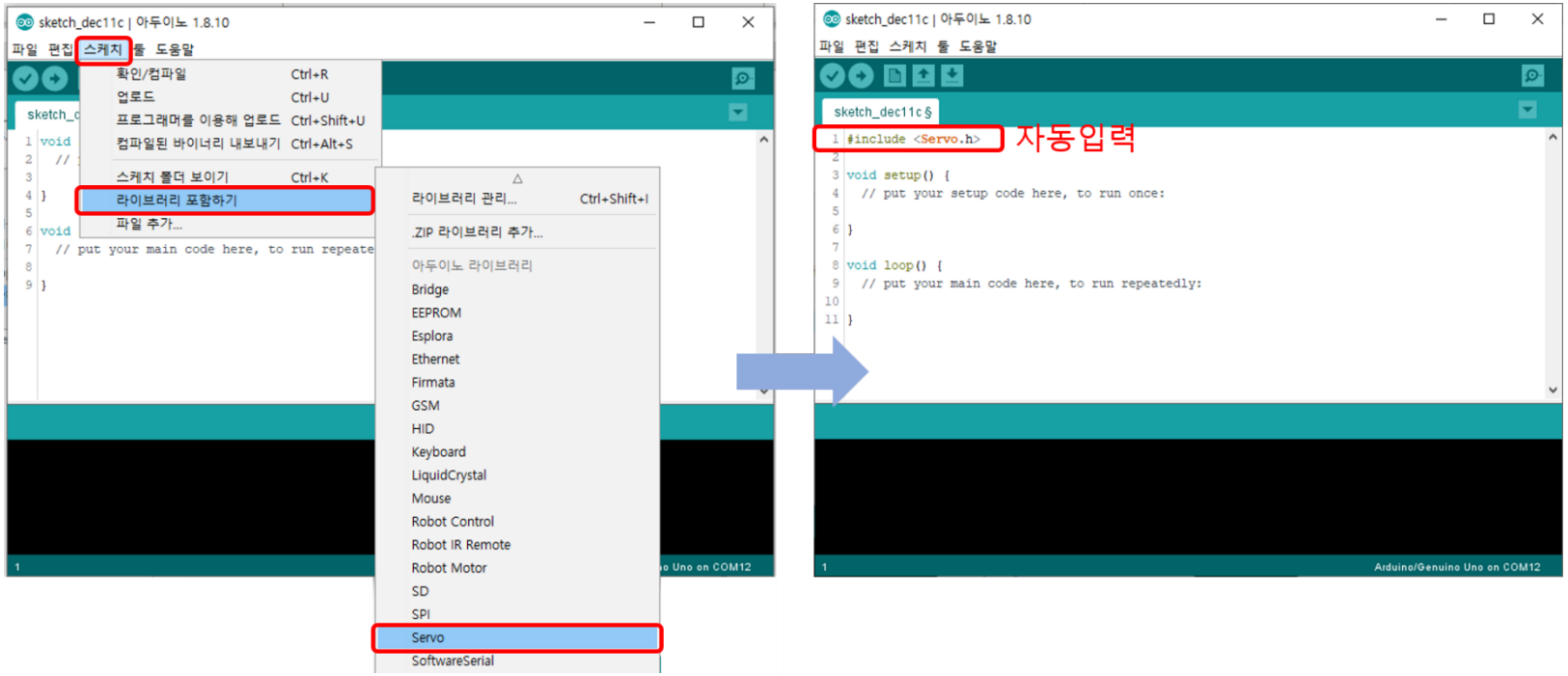
1. 모터 소개

- 서보 모터
  - 정확한 위치 제어가 어려운 DC모터의 단점을 극복하기 위하여 DC모터에 귀환 회로를 추가하여 0° ~ 180°의 각도를 정밀하게 제어할 수 있는 모터
  - 주로 로봇 분야에서 관절이나 손과 같은 정밀함이 필요한 쪽으로 많은 사용

2. 서보 라이브러리와 함수의 사용

▶ 서보 라이브러리 설치

프로그램의 상단에 #include <servo.h>를 입력하거나 [스케치] 메뉴에서 [라이브러리 포함하기] → [Servo]를 클릭하면 자동으로 #include <servo.h>가 표시



2. 서보 라이브러리와 함수의 사용

▶ 서보 라이브러리 함수

표 7.1 서보 라이브러리 함수

함수	설명
void attach(pin)	서보모터를 제어할 핀을 설정
void attach(pin, min_us, max_us)	펄스 폭 A의 최소, 최댓값까지 지정
void write(angle)	특정 각도만큼 모터를 회전(angle: 0~180)
void writeMicroseconds(us)	펄스 폭 A를 us단위로 지정
int read()	현재 각도를 읽음(반환 값은 [0, 180]의 정수)
boolean attached()	서보모터 핀이 지정되어 있는지 검사
void detach()	서보모터의 지정된 핀을 내부적으로 제거

2. 서보 라이브러리와 함수의 사용

- 서보 모터의 제어

[필요한 HW장치 및 부품]
- 아두이노 우노 보드 1개
- SG-90 서보모터 1개

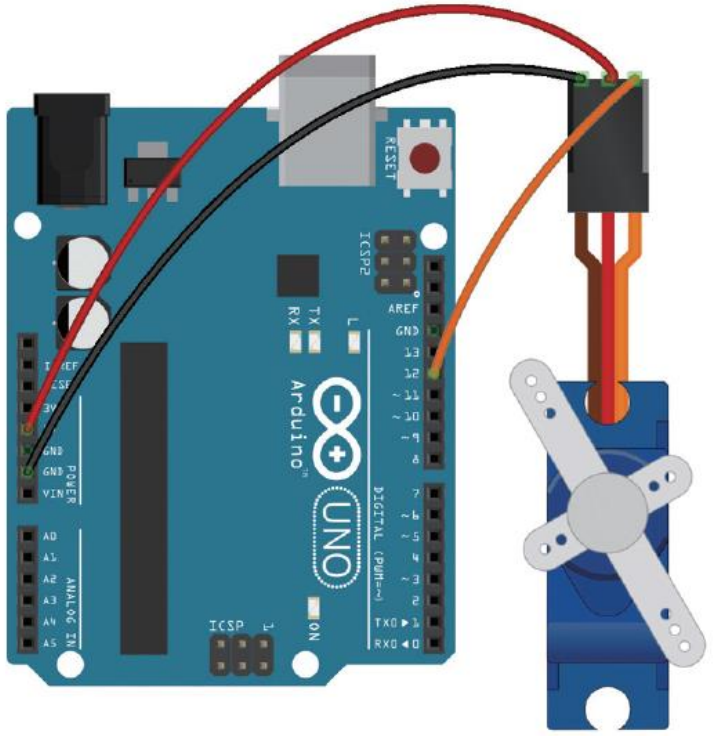


그림 7.7 서보모터와 아두이노의 회선 연결

표 7.2 서보모터와 아두이노 보드의 회선 연결

SG-90 서보모터	VCC	GND	신호선
아두이노 보드	5V	GND	12번 핀

2. 서보 라이브러리와 함수의 사용

- 서보모터의 샤프트를 0도→180도, 180도→0 이동시키는 프로그램

```
#include <Servo.h>
Servo myServo; // 서보모터를 제어하기 위한 서보 객체의 생성
// 대부분의 보드에서 서보 객체를 12개까지 생성할 수 있음
int angle = 90; // 서보 위치(각도)를 저장하기 위한 변수
void setup() {
  myServo.attach(12); // 핀 12번에 연결된 서보를 서보 객체에 배속시킴
}
void loop() {
  for (angle = 0; angle <= 180; angle += 1) { // 0도에서 180도로 1도씩 이동
    myServo.write(angle); // pos가 가진 각도의 위치로 이동
    delay(15); // 서보가 해당 위치에 도달할 때까지 15ms 대기
  }
  for (angle = 180; angle >= 0; angle -= 1) { // 180도에서 0도로 1도씩 이동
    myServo.write(angle); // pos가 가진 각도의 위치로 이동
    delay(15); // 서보가 해당 위치에 도달할 때까지 15ms 대기
  }
}
```

2. 서보 라이브러리와 함수의 사용

- 푸시 버튼을 이용한 서보 모터의 제어

[필요한 HW장치 및 부품]
- 아두이노 우노 보드 1개
- 브레드보드 1개
- SG-90 서보모터 1개
- 푸시 버튼 1개
- 10kΩ 저항 1개
- 약간의 점퍼선

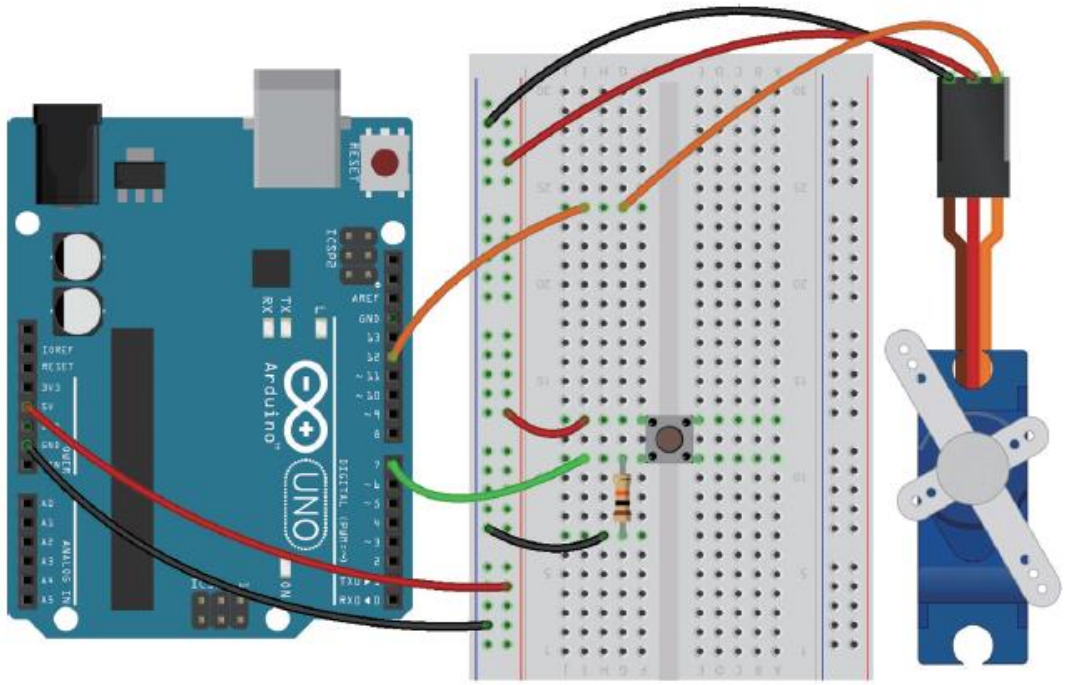


그림 7.8 서보모터, 푸시버튼 그리고 아두이노 보드의 연결

2. 서보 라이브러리와 함수의 사용

- 푸시 버튼을 누르면 서보모터의 샤프트를 시계 방향 또는 반시계 방향으로 이동시키는 프로그램

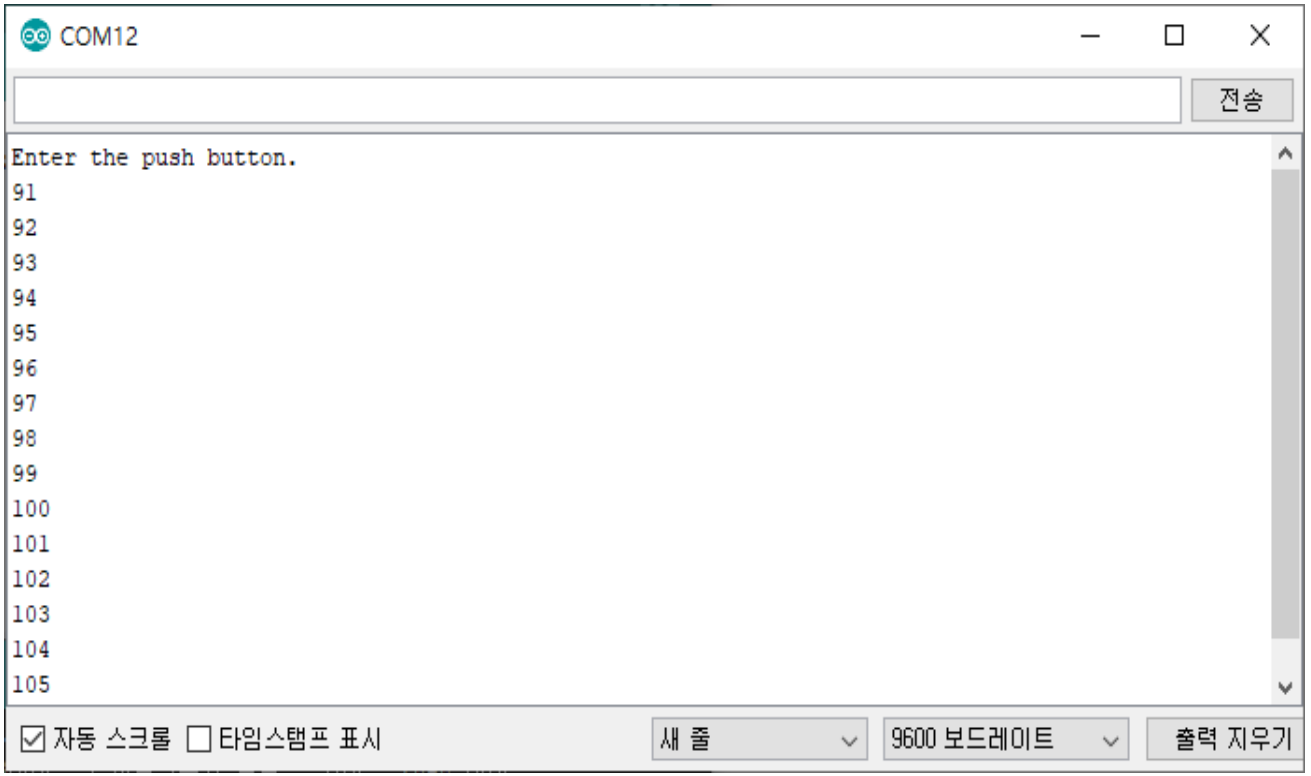
```
#include <Servo.h>
Servo myServo; // 서보모터 객체를 생성하고 이름을 myServo로 설정
int motorPin = 12; // 서보모터를 12번 핀에 연결
int pushPin = 7; // 푸시 버튼을 7번 핀에 연결
int angle = 90; // 초기에 샤프트는 중간에 위치
int state=0; // 회전방향 설정 (state 값 0 :  시계 방향 / 1 : 반시계 방향)
void setup() {
  myServo.attach(motorPin); // 서보 모터를 아두이노의 해당 핀에 연결
  pinMode(push, INPUT); // 푸시 버튼이 연결된 핀을 입력 모드로 설정
  Serial.begin(9600); // 시리얼 모니터와 통신
  Serial.println("Enter the push button."); // 시리얼 모니터에 문자열 출력
}
```

2. 서보 라이브러리와 함수의 사용

```
void loop() {
  if (digitalRead(pushPin) == HIGH){
    if (state == 0) {
      angle += 1; // 각도를 1도씩 증가
      if (angle>=180) state = 1; // 회전 방향을 반시계 방향으로 전환
      myServo.write(angle); // angle 값을 서보에게 전달하여 기록
      delay(10);
      Serial.println(angle); // 현재 각도를 시리얼 모니터에 표시
    }
    else {
      angle -= 1;
      if (angle<=0) state = 0; // 회전 방향을 시계 방향으로 전환
      myServo.write(angle);
      delay(10);
      Serial.println(angle);
    }
  }
}
```

2. 서보 라이브러리와 함수의 사용

- 푸시 버튼을 누르면 서보모터의 샤프트를 시계 방향 또는 반시계 방향으로 이동시키는 프로그램
- 시리얼 모니터에 표시된 서보모터의 위치 값



2. 서보 라이브러리와 함수의 사용

- 가변저항을 이용한 서보 모터의 제어

[필요한 HW장치 및 부품]
- 아두이노 우노 보드 1개
- 브레드보드 1개
- SG-90 서보모터 1개
- 가변저항 1개
- 약간의 점퍼선

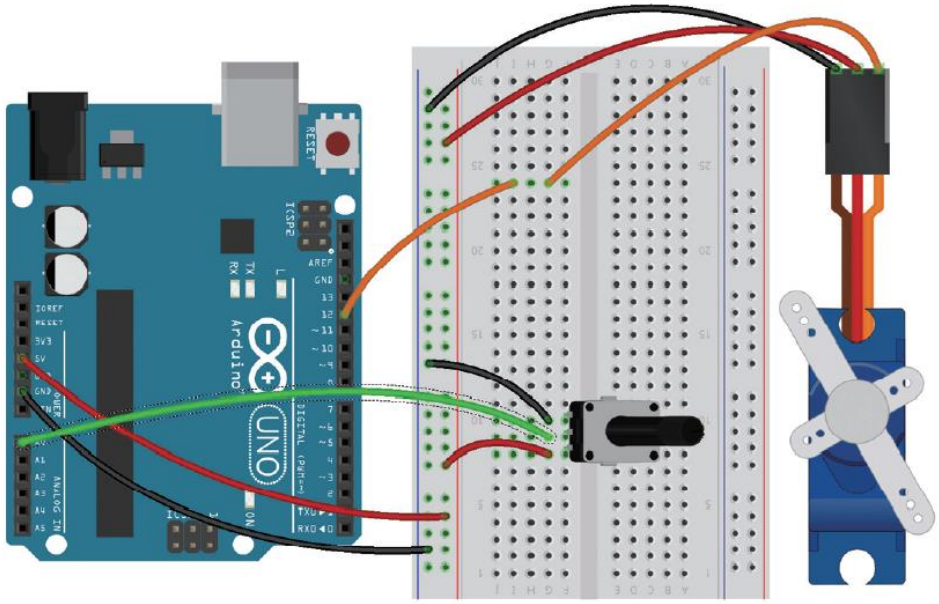


그림 7.11 서보모터, 가변저항기와 아두이노의 연결

표 7.3 서보모터, 가변저항기와 아두이노의 연결

가변저항	아두이노
Left Arm	5V
Center Arm	A0
Right Arm	GND

서보모터	아두이노
Vcc(Red)	5V
Data(Orange)	Digital 12
GND(Brown)	GND

2. 서보 라이브러리와 함수의 사용

```
#include <Servo.h>
Servo myServo;
void setup() {
  Serial.begin(9600); // 시리얼 모니터와의 통신
  myServo.attach(12); // 디지털 12번 핀을 이용하여 서보모터를 제어
}
int val; // 가변저항의 출력 값을 읽어서 저장할 변수
int angle;
void loop() {
  val = analogRead(A0);
  angle = map(val, 0, 1023, 0, 180); // 0~1023의 가변저항 출력값을 0~180도로 매핑
  myServo.write(angle);
  Serial.println(angle); // 현재 angle 값을 시리얼 모니터에 표시
  delay(50);
}
```

2. 서보 라이브러리와 함수의 사용

- 시리얼 모니터 입력으로 서보모터 제어

  시리얼 모니터 입력창에 U(u) 또는 D(d)를 한 번 또는 여러 번 입력하고 아두이노로 전송

```
#include <Servo.h> // 서보모터 라이브러리
Servo myServo; // 서보모터 사용을 위한 객체 생성
int motor = 12; // 서보모터의 핀
int angle = 90; // 서보모터 초기 각도 값
void setup() {
  myServo.attach(motor); // 서보모터 연결
  Serial.begin(9600); // 시리얼 모니터 시작
  Serial.println("Initial angle : 90 degrees"); // 초기값 출력
  Serial.println("Enter U(u) or D(d)"); // u 또는 d키 입력하기
  Serial.println("U : Increase the angle by 5 degrees."); // U : 현재 각도에서 5도 증가
  Serial.println("D : Decrease the angle by 5 degrees."); // D : 현재 각도에서 5도 감소
}
```

2. 서보 라이브러리와 함수의 사용

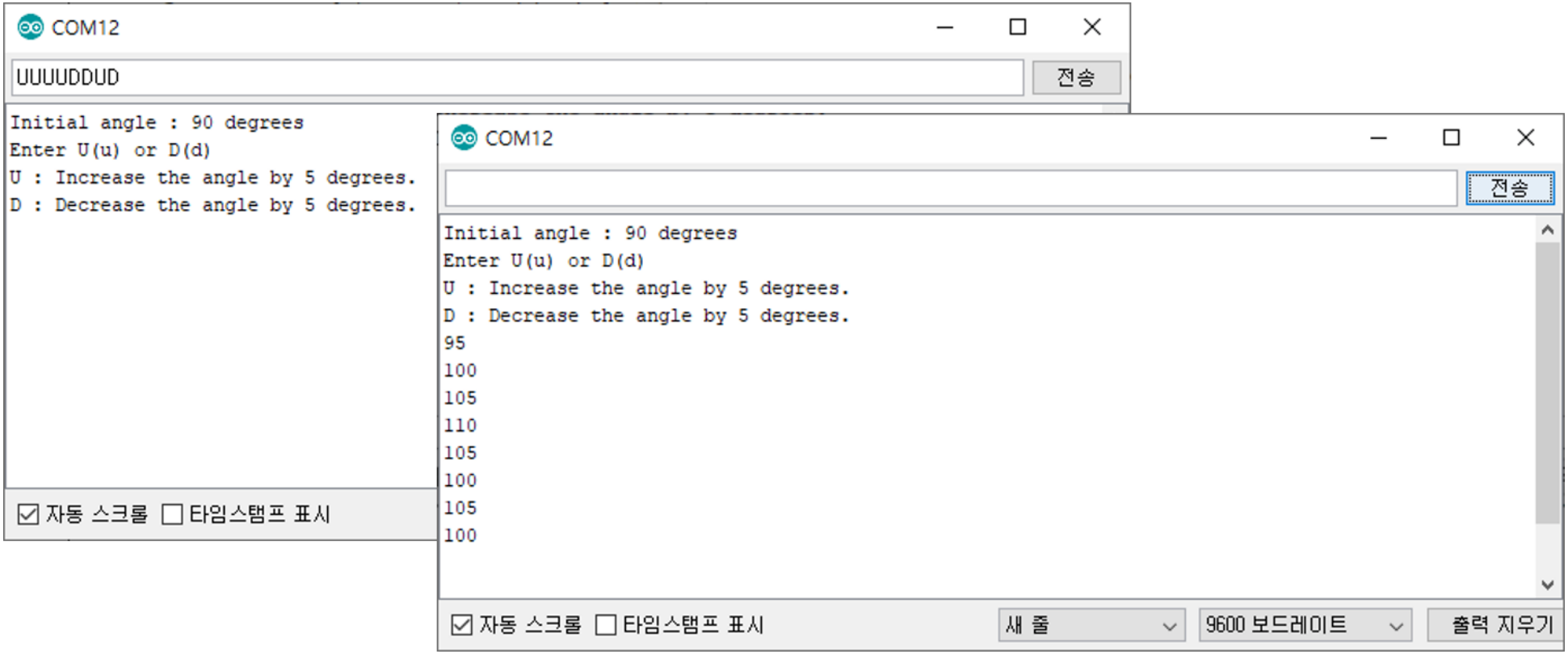
```
void loop() {
  if(Serial.available()) { // 시리얼모니터가 사용가능할 때
    char input = Serial.read(); // 문자 입력받기
    if(input == 'U' || input == 'u') { // U(u) 키를 누를 때
      for(int i = 0; i < 5; i++) {  // 현재 각도에서 15도 더해주기
        angle += 1;
        if(angle >= 180) angle = 180;
        myServo.write(angle);
        delay(10);
      }
      Serial.println(angle); // 현재 각도 출력
    }
```

2. 서보 라이브러리와 함수의 사용

```
    else if(input == 'D' || input == 'd') { // D(d)키를 입력했을 때
      for(int i = 0 ; i < 5 ; i++) { // 현재 각도에서 15도 빼주기
        angle -= 1;
        if(angle <= 0) angle = 0;
        myServo.write(angle);
        delay(10);
      }
      Serial.println(angle); // 현재 각도 출력
    }
    else if(input == '₩n') { // 엔터키의 입력에는 무반응
    }
    else { // U(u) 또는 D(d) 이외의 문자를 입력하였을 때
      Serial.println("wrong character!!");
    }
  }
}
```

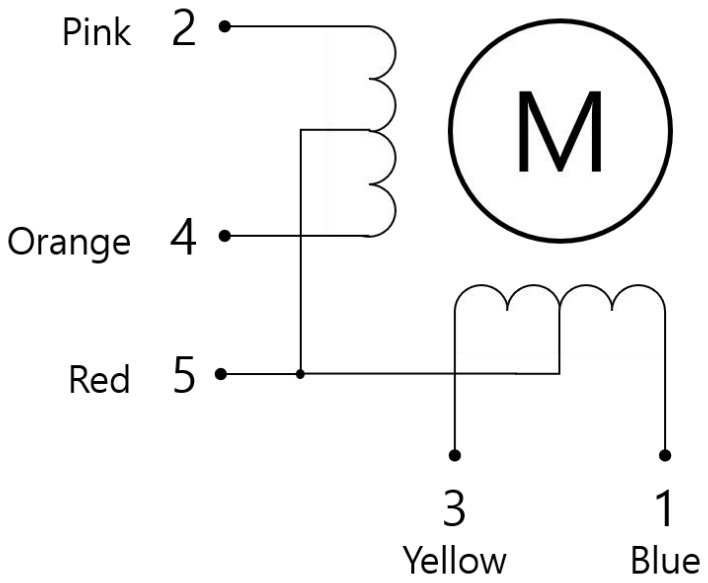
2. 서보 라이브러리와 함수의 사용

- 시리얼 모니터로 서보모터의 제어 실험 결과

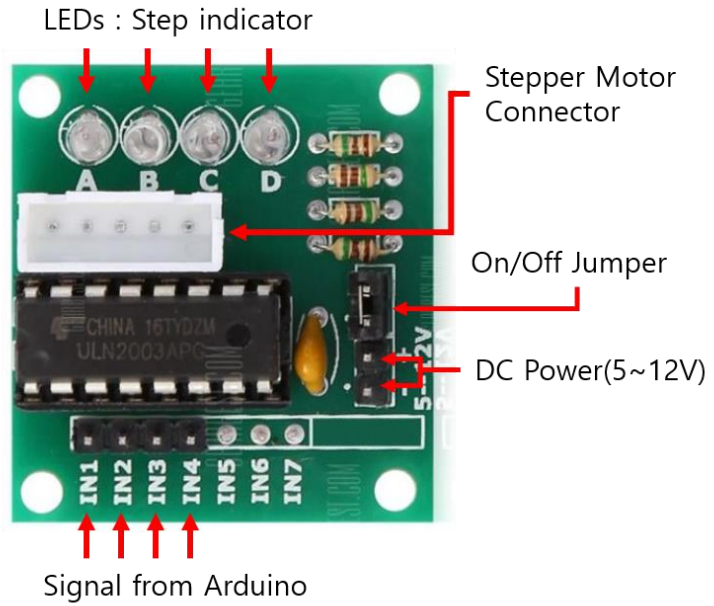


3. 스테퍼 모터 제어

- 스테퍼 모터 28-BYJ48 및 ULN2003 스테퍼 모듈



28BYJ-48 스테퍼모터

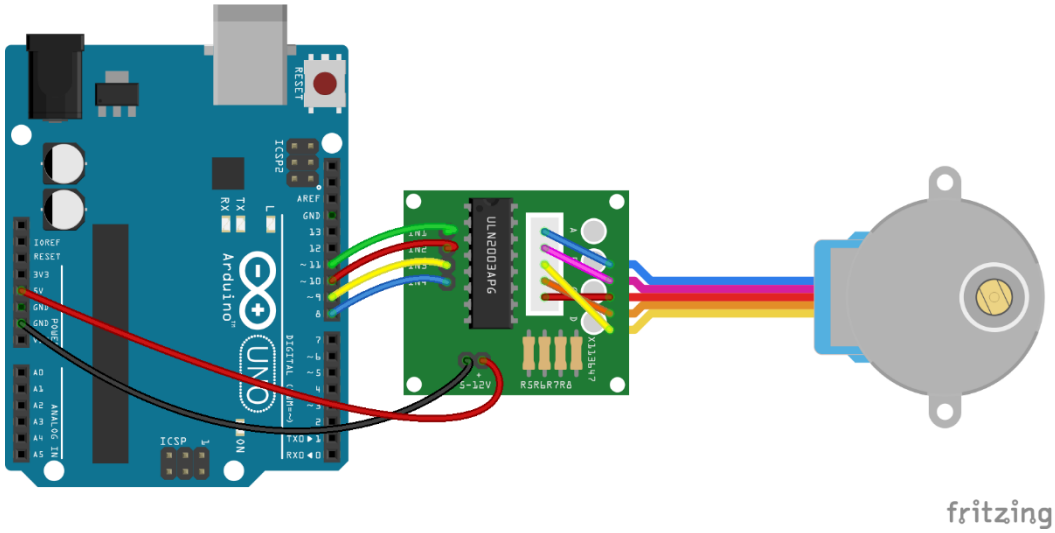


ULN2003 드라이버

3. 스테퍼 모터 제어

- 스테퍼 모터의 시계 방향 / 반시계 방향 회전 제어

[필요한 HW장치 및 부품]
- 아두이노 우노 보드 1개
- 28BYJ-48 스테퍼 모터 1개
- ULN2003 드라이버 1개
- 약간의 점퍼선



3. 스테퍼 모터 제어

```
#include <Stepper.h>
// 2048: 한바퀴(360도), 1024: 반바퀴(180도)...
const int stepsPerRevolution = 2048;
// Stapper class의 인스턴스인 myStepper를 생성
// 아두이노 보드의 디지털 8, 9, 10, 11번 핀을
// 스테퍼 모터 드라이브의 IN1, IN3, IN2, IN4에 연결
// 인스턴스 생성 시 1-3-2-4 순서로 대응되는 핀 번호 입력
Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);
void setup() {
  myStepper.setSpeed(14); // 스테퍼 모터의 스피드 설정
}
void loop() {
  // 시계 반대 방향으로 한 바퀴 회전
  myStepper.step(stepsPerRevolution);
  delay(500);
  // 시계 방향으로 한 바퀴 회전
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

3. 스테퍼 모터 제어

- 8단계로 한 바퀴 회전하는 스테퍼 모터의 구현

```
#include <Stepper.h>
const int stepsPerRevolution = 2048/8; // 45도(2048=360도)
// ULN2003 스테퍼 모듈의 IN1, IN2, IN3, IN4를
// 아두이노의 D8, D9, D10, D11에 순서대로 연결
Stepper myStepper(stepsPerRevolution, 8,10,9,11); // IN 1-3-2-4 시퀀스
void setup() {
  // 모터의 속도를 120으로 설정
  myStepper.setSpeed(120);
  // 시리얼 포트의 초기화
  Serial.begin(9600);
}
```

3. 스테퍼 모터 제어

```
void loop() {
  if(Serial.available()){
    Serial.println("clockwise");
    // stepsPerRevolution만큼 8번 이동 (시계 방향으로 1회전)
    for (int x=1;x<9;x++)
    {
      myStepper.step(stepsPerRevolution);
      Serial.println(x); // 현재 반복 횟수를 시리얼 모니터에 출력
    }
    delay(500);
    Serial.println("counterclockwise");
    // -stepsPerRevolution만큼 8번 이동 (반시계 방향으로 1회전)
    for (int x =1;x<9;x++)
    {
      myStepper.step(-stepsPerRevolution);
      Serial.println(x); // 현재 반복 횟수를 시리얼 모니터에 출력
    }
    delay(500);
  }
}
```

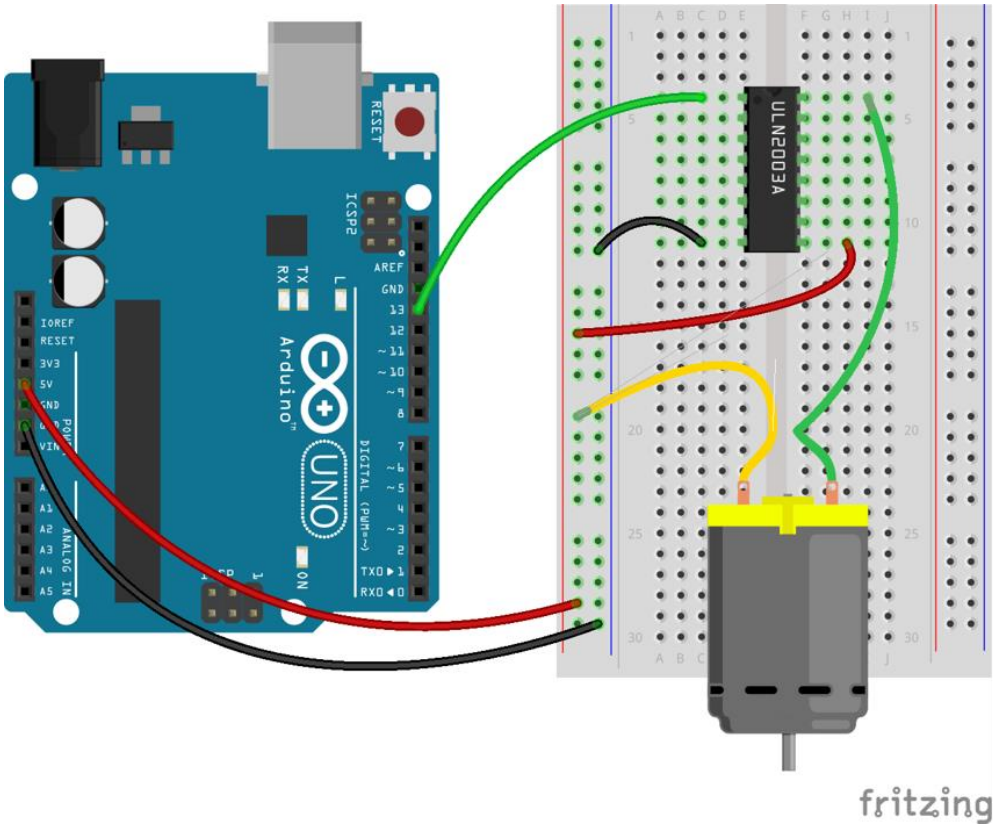
3. 스테퍼 모터 제어

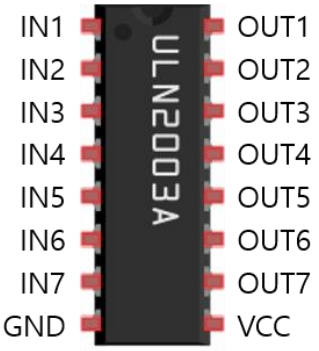
- 시리얼 통신을 이용한 스테퍼 모터의 제어

```
#include <Stepper.h>
const int STEPS = 2048;
Stepper stepper(STEPS, 8,10,9,11);
void setup()
{
  stepper.setSpeed(14);
  Serial.begin(9600);
}
void loop()
{
  if(Serial.available()) {
    int val=Serial.parseInt();      // 회전각 int형으로 읽기
    val=map(val,0,360,0,2048); // 회전각 스텝 수
    stepper.step(val);
    Serial.println(val);
    delay(10);
  }
}
```

4. DC 모터 제어

- 간단한 DC 모터의 제어
- DC모터와 디지털 푸시버튼의 회선 연결





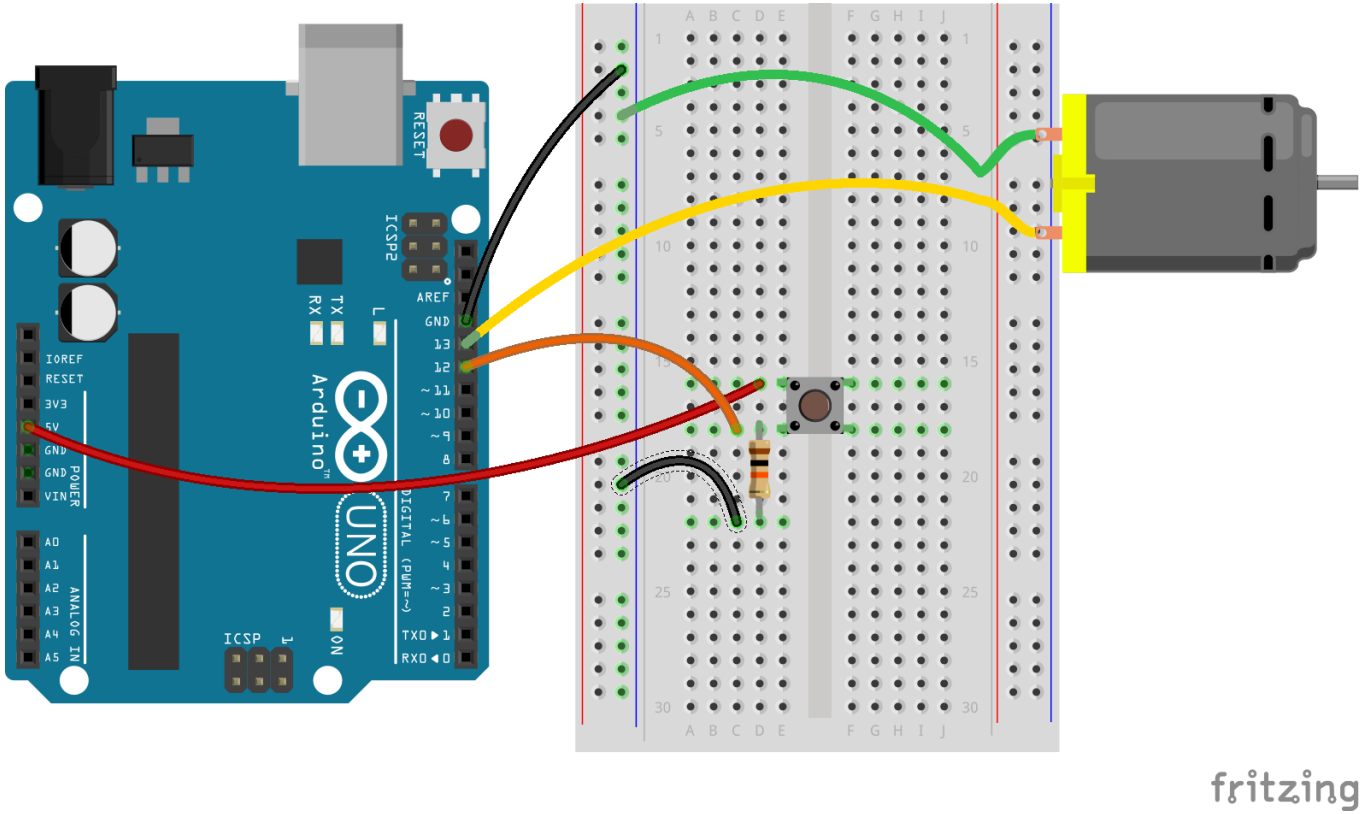
4. DC 모터 제어

- 간단한 DC 모터의 제어

```
int motor = 13;
void setup()
{
  pinMode(motor, OUTPUT);
}
void loop()
{
  digitalWrite(motor, HIGH); // DC 모터의 회전
  delay(1000);
  digitalWrite(motor, LOW); // DC 모터의 회전 멈춤
  delay(1000);
}
```

4. DC 모터 제어

- 푸시버튼을 이용한 DC모터의 제어
- DC모터와 디지털 푸시버튼의 회선 연결



4. DC 모터 제어

- 푸시 버튼을 이용한 DC모터의 제어

```
int state=0;
int motor = 13;
int btn = 12;
void setup() {
  pinMode(motor, OUTPUT);
  pinMode(btn, INPUT);
}
```

```
void loop() {
  if (digitalRead(btn) == HIGH){
    state=1;
  }
  else {
    state=0;
  }
  if (state == 1) {
    digitalWrite(motor, HIGH);
  }
  else {
    digitalWrite(motor, LOW);
  }
}
```