IT프로그래밍

한성대학교
IT융합공학부
오희석
(ohhs@hansung.ac.kr)

센서 / 모터 제어하기

1. 아날로그 초음파 센서
2. 모터 소개
3. 서보 라이브러리와 함수의 사용

학습목표

- 초음파(ultrasound) 센서를 이용하여 가까운 거리를 측정할 수 있다.
- 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.

1. 아날로그 초음파(ultrasound) 센서

[초음파 센서]

- 병원에서 초음파 진단기로 질병을 진단
- 초음파를 이용하여 기기 세척
- 바다에서 고기잡이를 하기 위해 수중 초음파 장비를 사용
- 초음파는 인간이 들을 수 없는 범위의 주파수
- 20KHz 이상인 음파를 말함
- 사람이 들을 수 있는 주파수 범위는 일반적으로 16Hz에서 20KHz 사이

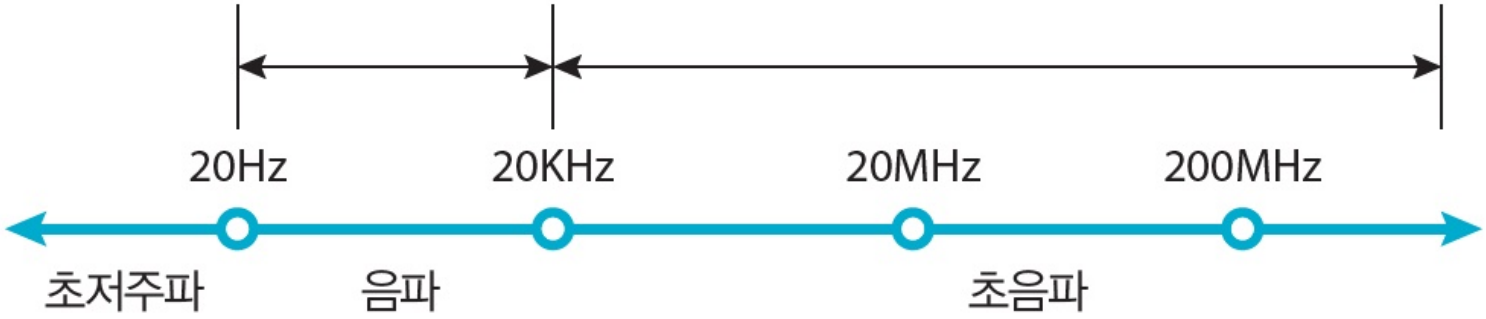


그림 4.14 주파수별 음파의 종류

1. 아날로그 초음파(ultrasound) 센서

[초음파 센서]

- 신호를 보내는 발신부와 물체에 반사되어 돌아오는 수신부가 분리됨
- 발신부에서 신호를 보내고 반사되어 돌아오는 시간을 계산하여 거리 측정
- 신호를 보내는 부분이 Trig, 받는 부분이 Echo

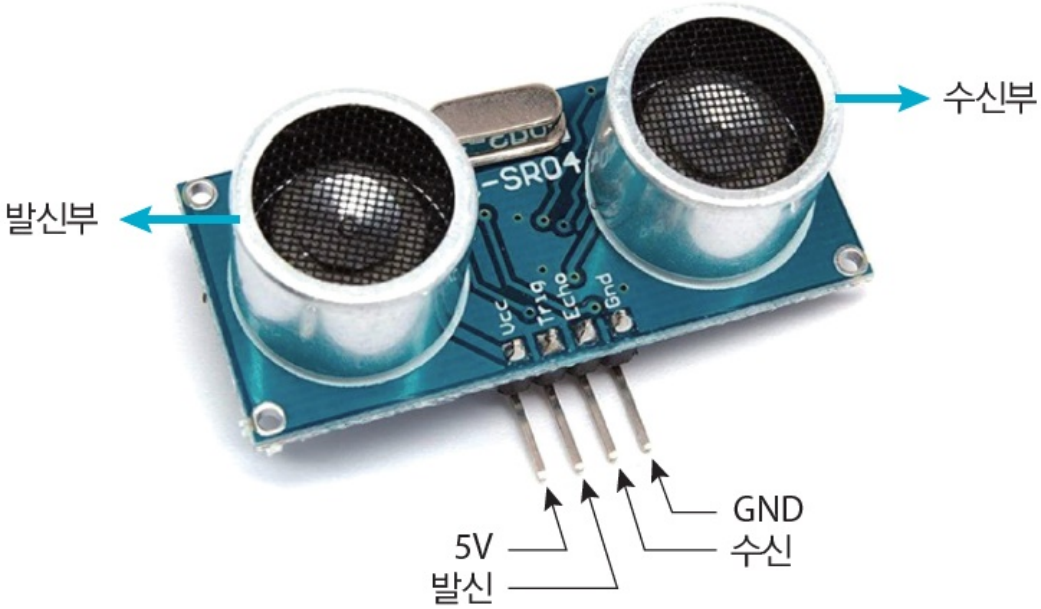


그림 4.15 초음파센서 핀 설명

1. 아날로그 초음파(ultrasound) 센서

- HC-SR04 초음파 센서
- 송신부와 수신부가 분리
- 동작 전압: 5V
- 발생한 주파수: 40kHz
- 측정 거리: 2~400cm
- 핀은 4개가 있고 5V, GND, Trig, 핀번호에 연결

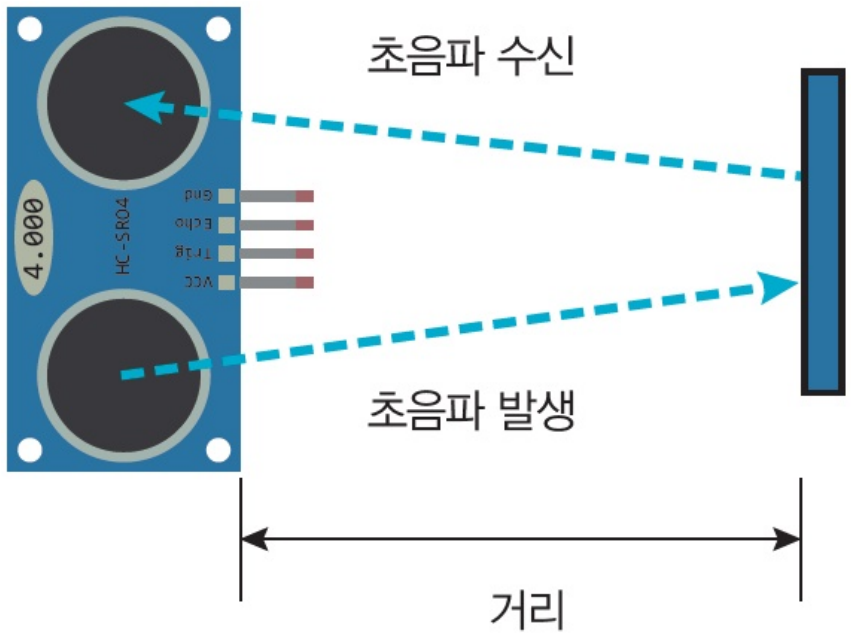


그림 4.16 초음파 거리 측정

1. 아날로그 초음파(ultrasound) 센서

(1) 회로 만들기

- 초음파 센서는 4개의 핀
- VCC 부분은 아두이노의 5V에 연결.
- Trig는 2번 핀, Echo는 3번 핀, GND는 GND에 연결

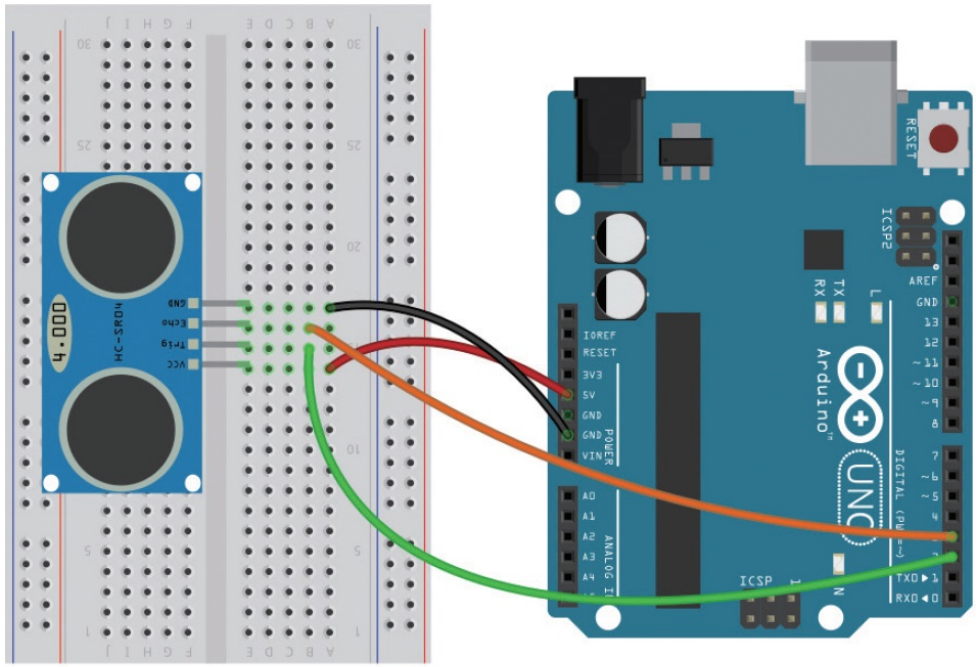


그림 4.17 아두이노와 초음파센서의 연결도

```
// 출력핀(trig)과 입력핀(echo) 연결 설정, 다른 핀을 연결해도 됨
int trigPin = 2; int echoPin = 3;
// 시리얼 속도 설정, trigPin을 출력, echoPin을 입력으로 설정
void setup() {
Serial.begin(9600);
  pinMode(trigPin, OUTPUT);  // 출력 핀으로 설정
  pinMode(echoPin, INPUT);  // 신호를 받는 핀 입력 설정
}
void loop() {
  float duration, distance;
  digitalWrite(trigPin, HIGH);    delay(10);
  digitalWrite(trigPin, LOW);
  // echoPin이 HIGH를 유지한 시간을 저장
  duration = pulseIn(echoPin, HIGH);
  // HIGH일 때 초음파가 보냈다가 돌아오는 시간을 가지고 거리를 계산
  // 340은 초당 소리의 속도이고 10000은 밀리세컨드를 세컨드로,
     왕복거리이므로 2로 나눔
  distance = ((float)(340 * duration) / 10000) / 2;
  // 시리얼모니터에 Echo가 HIGH인 시간 및 거리를 표시
  Serial.print("Duration:");      Serial.print(duration);
  Serial.print("₩nDistance:");   Serial.print(distance);
  Serial.println("cm₩n");         delay(500);
}
```

1. 아날로그 초음파(ultrasound) 센서

[코드 설명]

- int trigPin = 2;
  - 트리거 핀 번호 설정
- int echoPin = 3;
  - 에코 핀 설정
- Serial.begin(9600);
  - 초음파 센서의 거리 측정 결과를 시리얼 모니터에 출력하기 위해 시리얼 통신을 선언
- pinMode(trigPin, OUTPUT);
  - 초음파 트리거 핀을 출력(OUTPUT)으로 설정
- pinMode(echoPin, INPUT);
  - 초음파 에코핀을 입력(INPUT)으로 설정
- digitalWrite(trigPin, HIGH);
  - 트리거 신호를 발생시키는 신호
- duration = pulseIn(echoPin, HIGH);
  - 에코 핀 포트가 HIGH가 될 때까지의 시간을 측정.
  - 단위는 microseconds

1. 아날로그 초음파(ultrasound) 센서

(3) 초음파로 거리 측정한 결과 보기

- 스케치를 업로드하고 시리얼 모니터를 실행

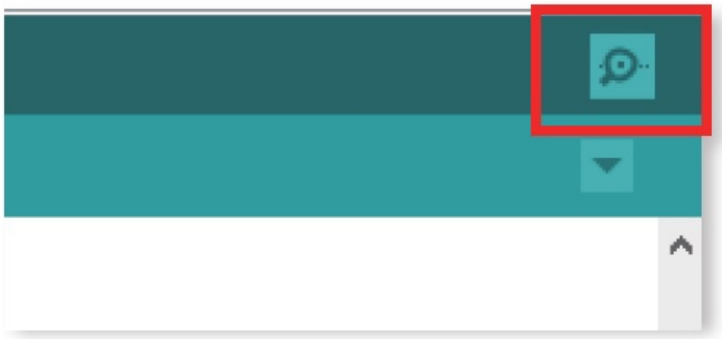


그림 4.18 시리얼 모니터 열기 버튼

1. 아날로그 초음파(ultrasound) 센서

- Duration: 에코핀 포트가 HIGH가 될 때까지의 시간을 측정한 값
- 단위는 microseconds
- Distance: 초음파 측정 거리 결과

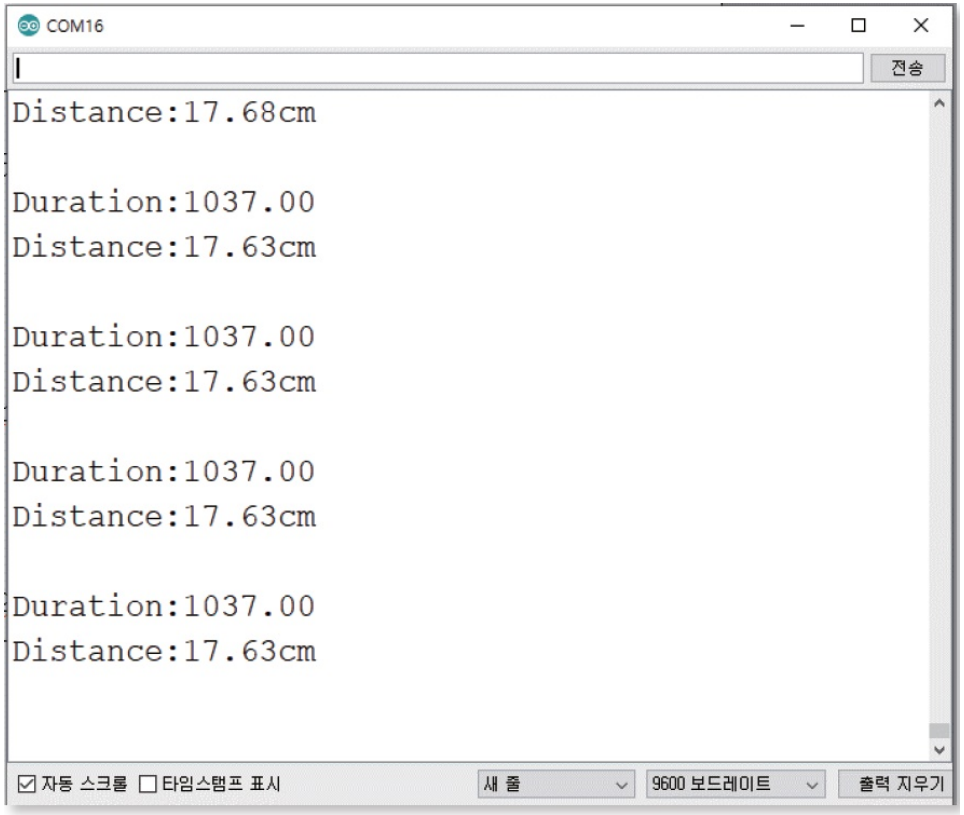


그림 4.19 초음파 측정 값 시리얼 모니터 출력 결과

1. 아날로그 초음파(ultrasound) 센서

- [테스트 모습]

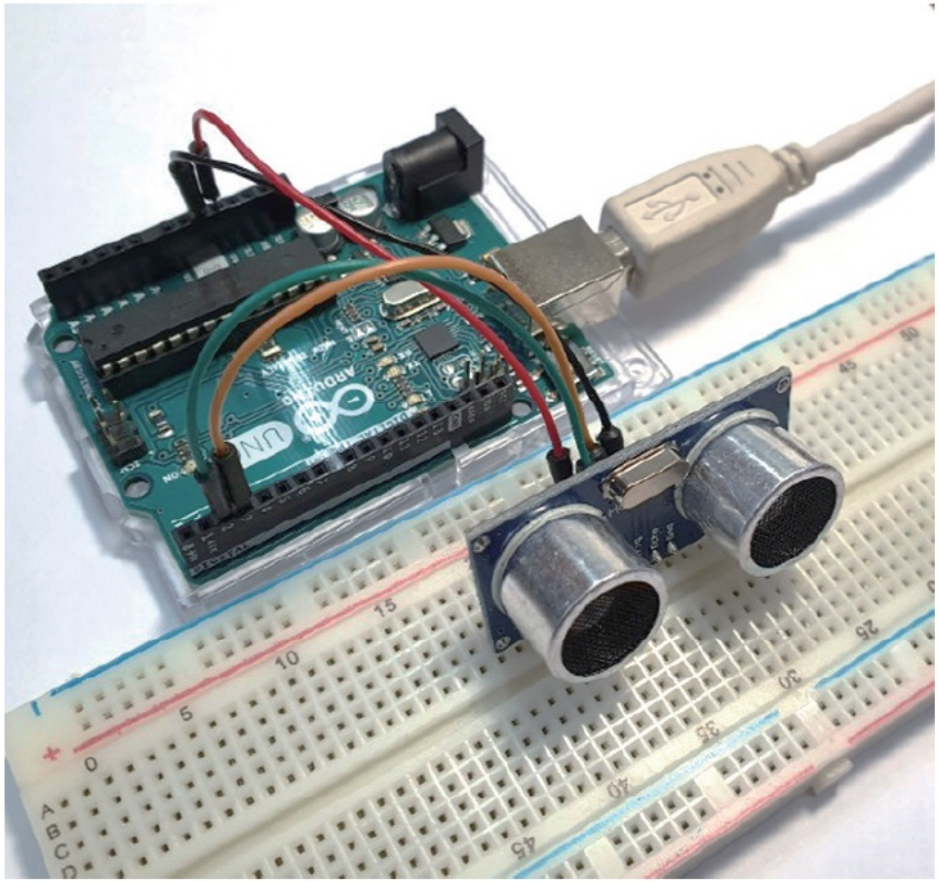


그림 4.20 초음파 테스트 모습

2. 모터 소개

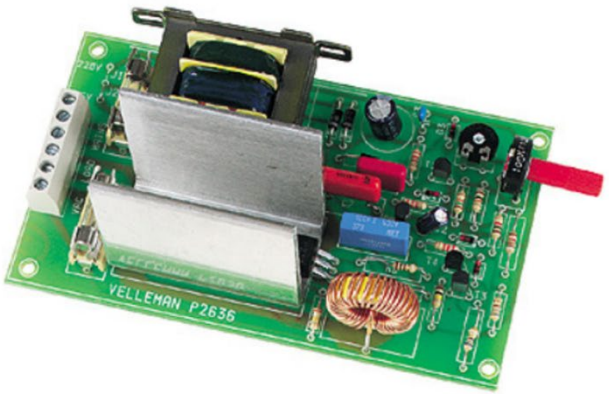
- 모터는 라틴어의 "moto"(움직인다)에서 온 단어로 플레밍의 왼손법칙에 따라 전기적 에너지를 운동(기계) 에너지로 변환하는 장치

그림 7.1 모터의 정의

2. 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

2. 모터 소개

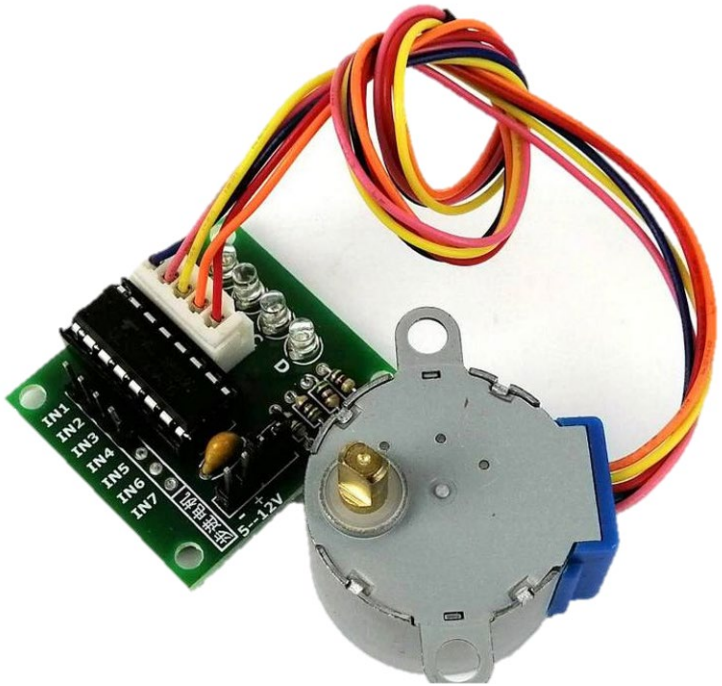
- DC 모터
  - 배터리와 같은 직류 전원을 사용
  - 고정자로 영구자석을 사용하고 전기자로 코일을 사용
  - 전기자에 흐르는 전류의 방향을 전환시키면서 자력의 밀어내는 힘이나 끌어당기는 힘으로 회전력을 생성
  - 적당한 전력만 공급해주면 회전하므로 RC자동차, 무선 조종용 장난감 등 여러 용도로 많이 사용
  - 전기를 끊으면 멈추게 되는데, 이때 관성에 의하여 정확한 정지 위치를 지정하기는 어려움

DC모터

DC모터 컨트롤러

2. 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.



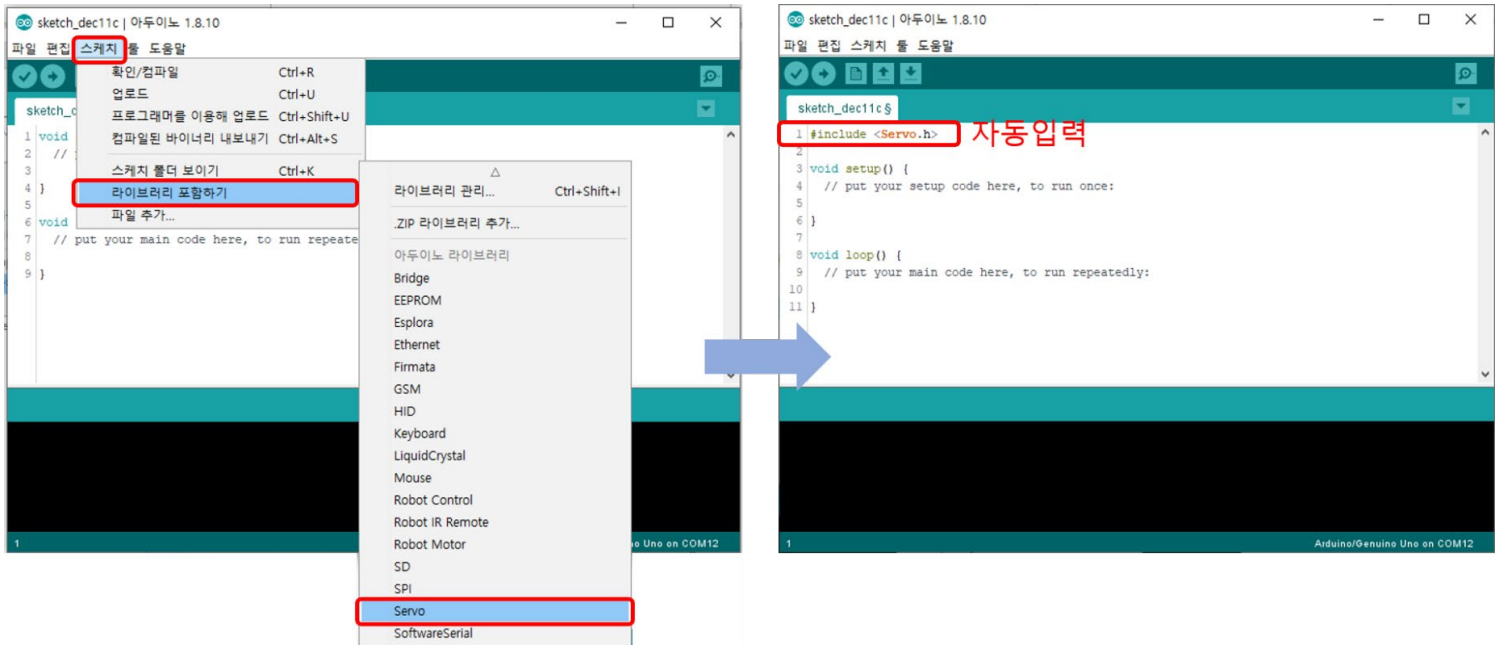
2. 모터 소개

- 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

3. 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



3. 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

3. 서보 라이브러리와 함수의 사용

- 서보 모터의 제어

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

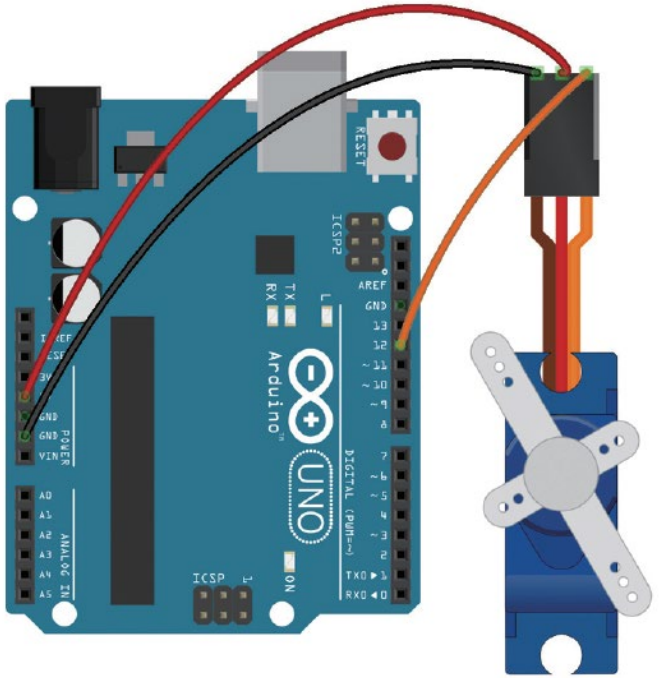


그림 7.7 서보모터와 아두이노의 회선 연결

표 7.2 서보모터와 아두이노 보드의 회선 연결

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

3. 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
  }
}
```

3. 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

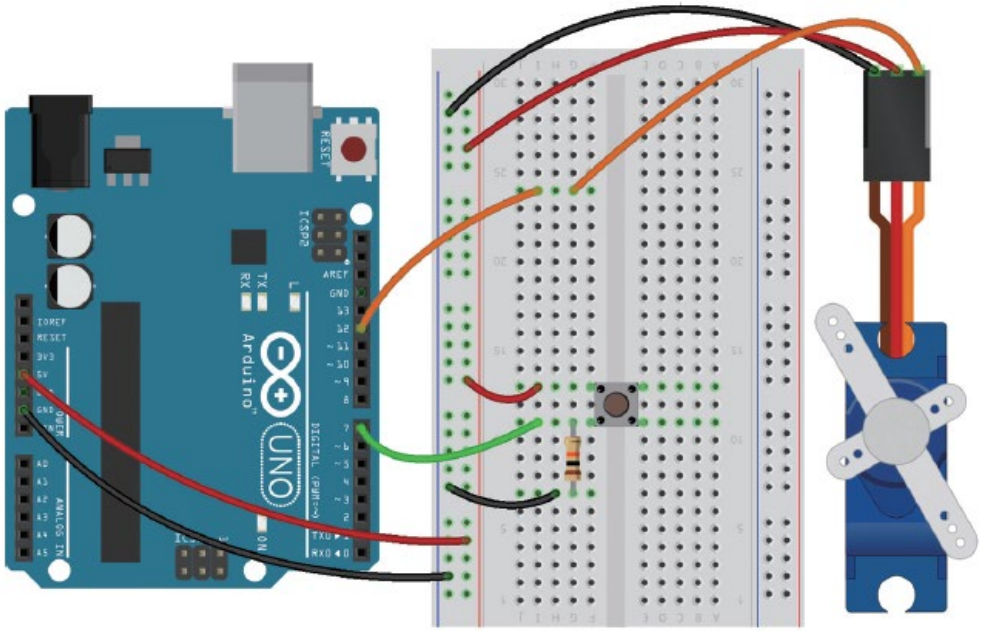


그림 7.8 서보모터, 푸시버튼 그리고 아두이노 보드의 연결

3. 서보 라이브러리와 함수의 사용

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

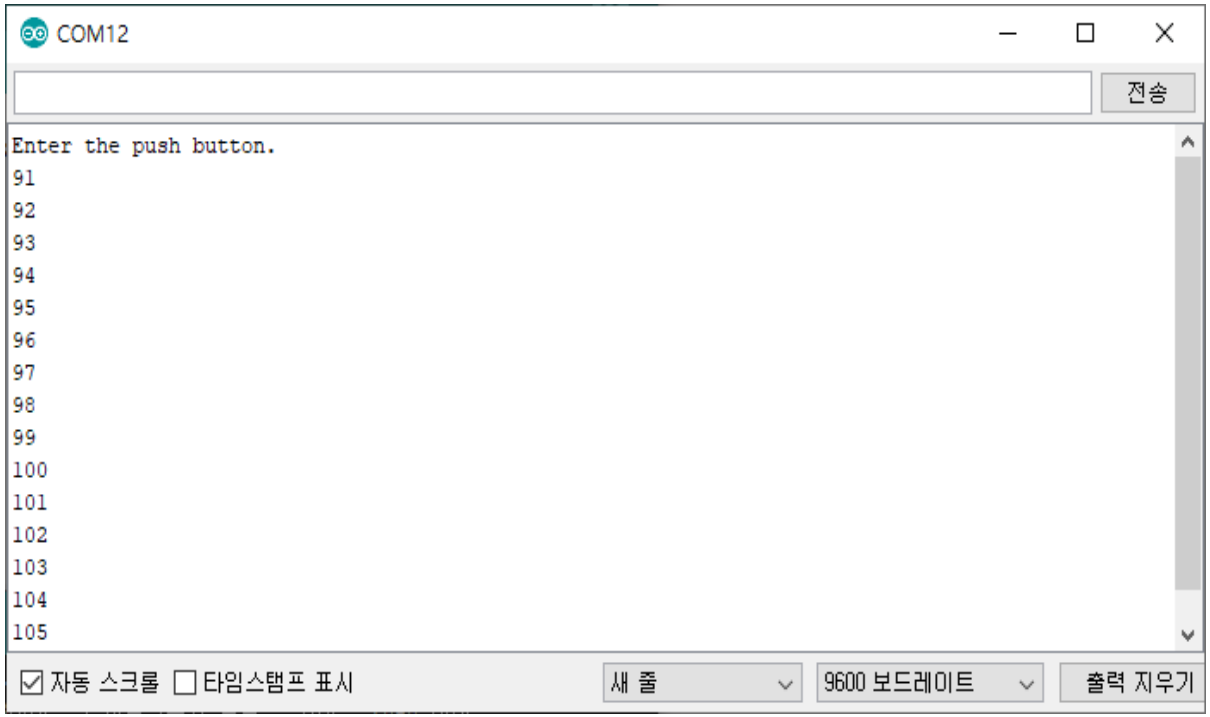
```
#include <Servo.h>
Servo myServo; // 서보모터 객체를 생성하고 이름을 myServo로 설정
int motorPin = 12; // 서보모터를 12번 핀에 연결
int pushPin = 7; // 푸시 버튼을 7번 핀에 연결
int angle = 90; // 초기에 샤프트는 중간에 위치
int state=0; // 회전방향 설정 (state 값 0 :  시계 방향 / 1 : 반시계 방향)
void setup() {
  myServo.attach(motorPin); // 서보 모터를 아두이노의 해당 핀에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

3. 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);
    }
  }
}
```

3. 서보 라이브러리와 함수의 사용

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



3. 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
}
```

3. 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
    }
```

3. 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!!");
    }
  }
}
```

3. 서보 라이브러리와 함수의 사용

- 시리얼 모니터로 서보모터의 제어 실험 결과