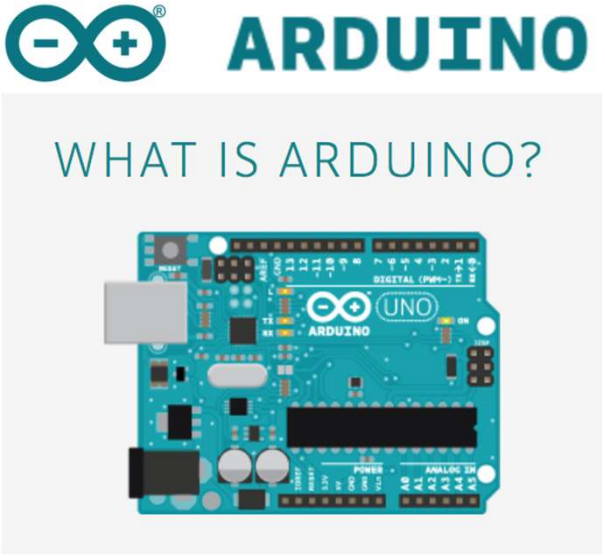
Chapter 10

초음파센서 사용해보기



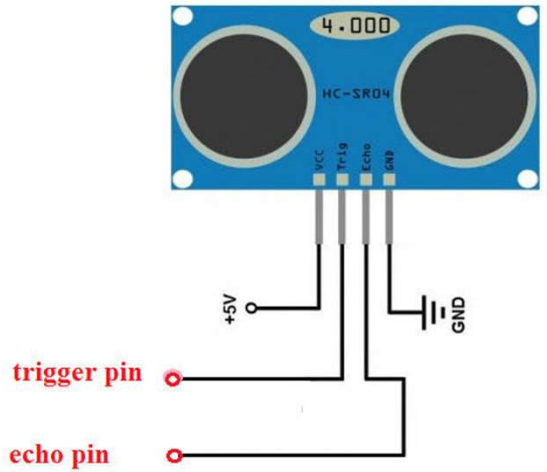
Step 1. 초음파센서를 이용하여 거리 감지하기

Step 2. 후방감지기 만들기

초음파센서를 이용하여 거리 감지하기

HC-SR04

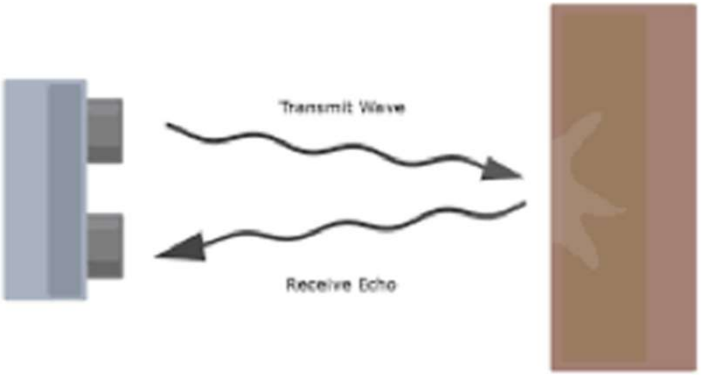
- 대표적인 초음파 센서 모듈
- 핀맵
  - VCC (5V)
  - GND
  - TRIG (Digital Output)
  - ECHO (Digital Input)
- 정격
  - 고려하지 않아도 됨



초음파센서를 이용하여 거리 감지하기

초음파 센서의 원리

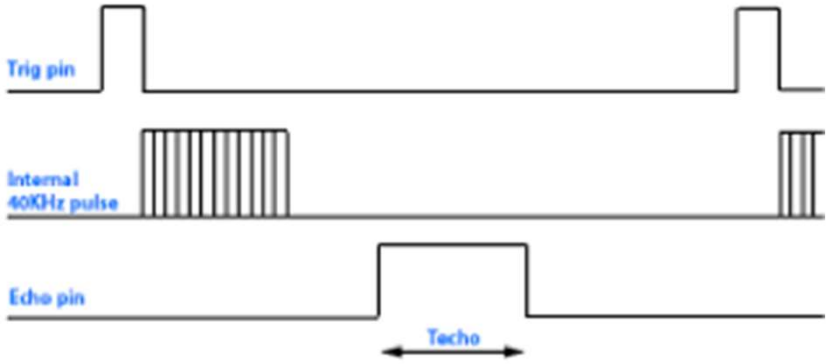
- 소리의 반사 시간을 측정하여 거리를 알아냄
  - 소리는 340m/s의 속도를 가진다.
  - 거리 = 속도 x 시간
  - 측정된 시간은 ½배를 해야 한다.
    (왕복 시간이므로)
  - 먼 거리를 측정할 수록 시간이 오래 걸린다.



초음파센서를 이용하여 거리 감지하기

HC-SR04 제어

- 초음파를 발생시킨다.
  - TRIG핀에 HIGH를 출력한다.
  - 짧은 시간 후 TRIG핀 LOW 출력
- 초음파가 돌아오기를 기다린다.
  - 초음파가 들어오면 ECHO핀 HIGH
  - 초음파가 없으면 ECHO핀 LOW
- 측정된 시간에서 거리 환산
  - 거리(cm) = 측정시간(usec) x 58



Time = Width of Echo pulse, in uS (micro second)

- Distance in centimeters = Time / 58
- Distance in inches = Time / 148
- Or you can utilize the speed of sound, which is 340m/s

초음파센서를 이용하여 거리 감지하기

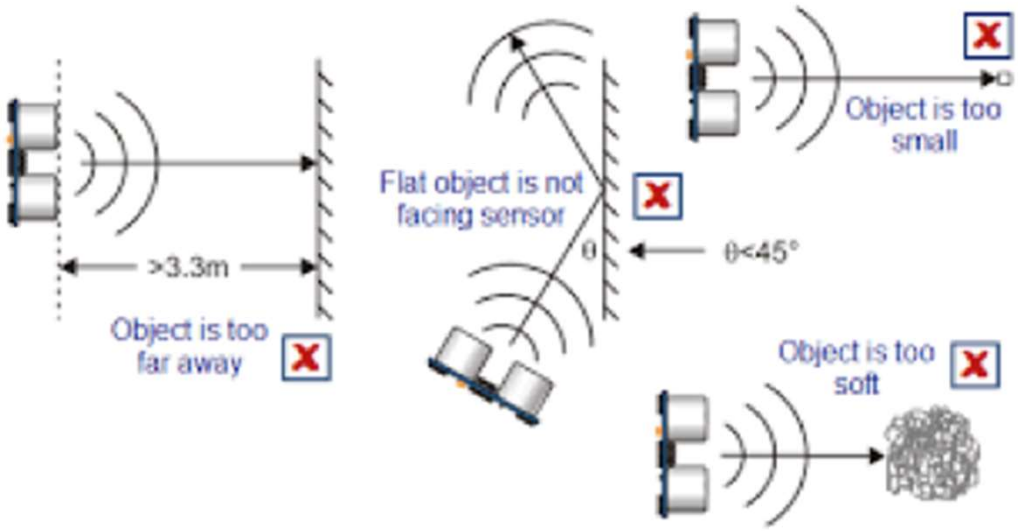
pulseIn

- 지정한 디지털 상태가 입력될때까지 기다린다.
- C언어 문법
  - unsinged long pulseIn(pin, value, timeout);
  - 반환값 있음, 인자 3개
- 사용방법
  - pin: 기다림에 사용할 입력 핀
  - value: 기다릴 입력 상태
  - timeout: 최대 기다릴 시간 (단위: msec)
  - 반환: 기다린 시간 (단위: usec)

초음파센서를 이용하여 거리 감지하기

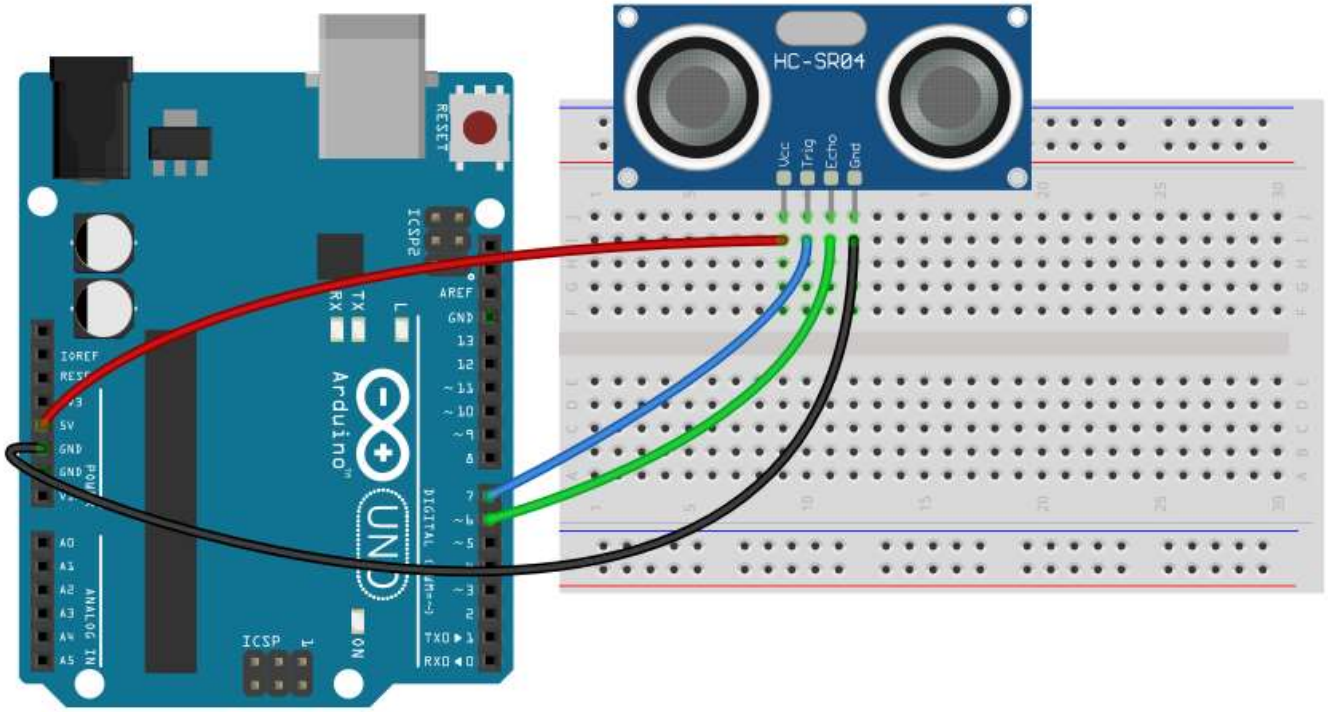
초음파센서의 단점

- 소음이 심한 곳에서는 측징이 잘 안 될 수도 있다.
- 여러 초음파 센서를 동시에 사용할 경우 서로 간섭을 일으킬 수 있다.
- 반사각이 수직이 아니면 측정이 안될 수 있다.
- 먼 거리를 측정하려면 시간이 오래 걸린다.
  (아무것도 못하고 기다려야 한다.)



초음파센서를 이용하여 거리 감지하기

회로구현



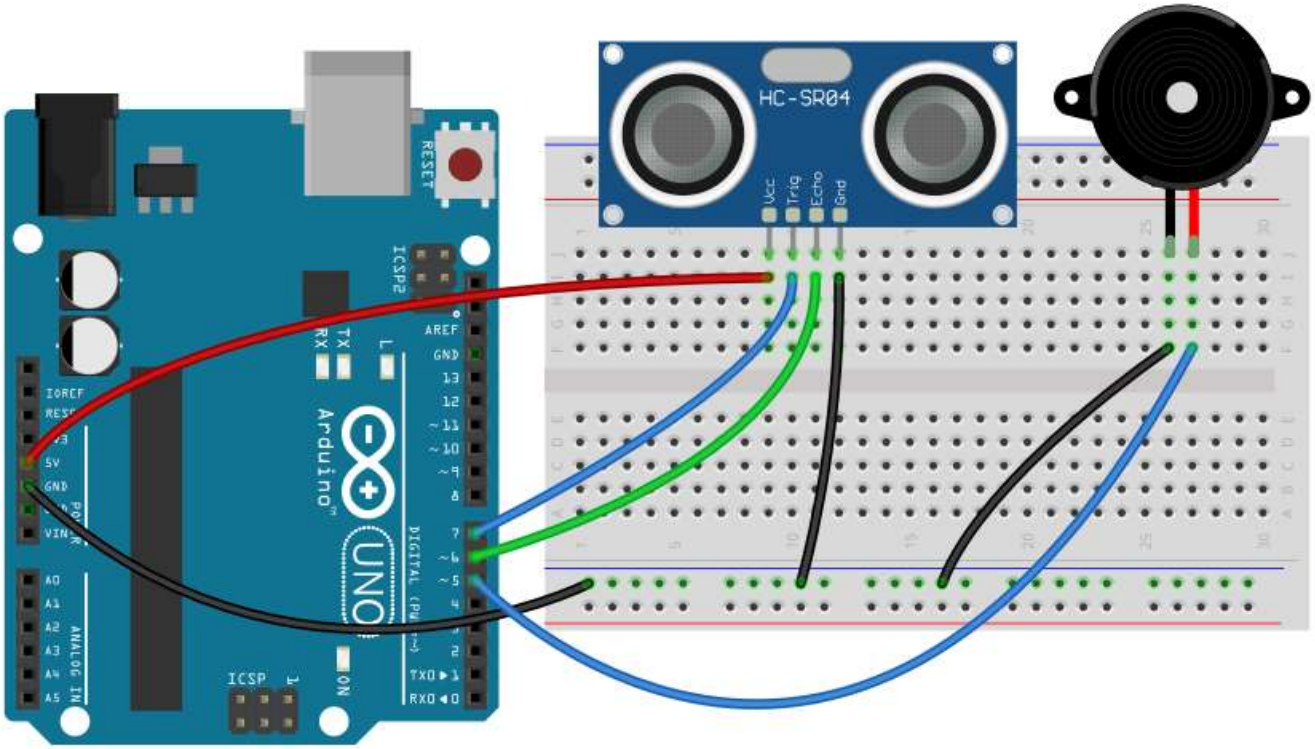
초음파센서를 이용하여 거리 감지하기

소스코딩

```
#define TRIG_PIN  7
#define ECHO_PIN  6

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}
void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(4);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(20);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 5000);
  float distance = duration / 58.0;

  Serial.print(distance); Serial.println("cm");
  delay(500);
}
```

후방 감지기 만들기

회로구현



후방 감지기 만들기

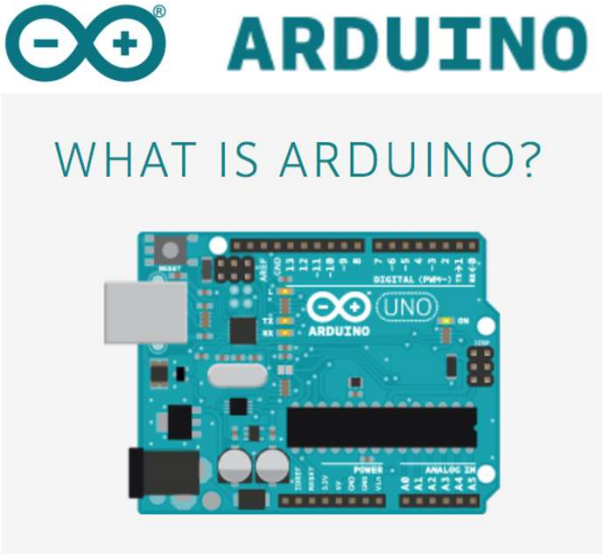
소스코딩

```
#define TRIG_PIN  7
#define ECHO_PIN  6
#define BUZZER_PIN  5

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(4);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(20);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 5000);
  float distance = duration / 58.0;
```

```
  if(distance >= 20){
    tone(BUZZER_PIN, 262);    //도
    delay(700);
    noTone(BUZZER_PIN);
    delay(700);
  }
  else if(distance >= 15){
    tone(BUZZER_PIN, 262);    //도
    delay(500);
    noTone(BUZZER_PIN);
    delay(500);
  }
  else if(distance >= 10){
    tone(BUZZER_PIN, 262);    //도
    delay(300);
    noTone(BUZZER_PIN);
    delay(300);
  }
  else if(distance >= 5){
    tone(BUZZER_PIN, 262);    //도
    delay(100);
    noTone(BUZZER_PIN);
    delay(100);
  }
  else {
    tone(BUZZER_PIN, 262);    //도
  }
}
```

Chapter 11

서보모터 제어해보기



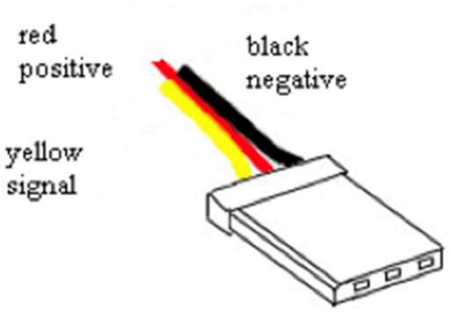
Step 1. 서보모터 제어하기

Step 2. 입력값을 이용하여 서보모터 제어하기

서보모터 제어하기

서보모터(Servo Motor)

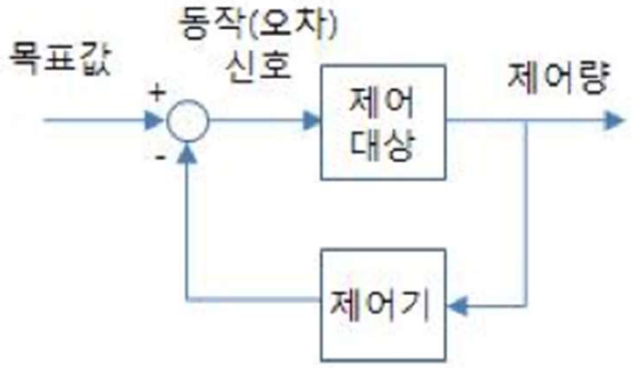
- 각도를 제어할 수 있는 모터
- 핀 순서에 주의
- 정격
  - 모터 종류마다 다르다.



서보모터 제어하기

서보모터의 원리

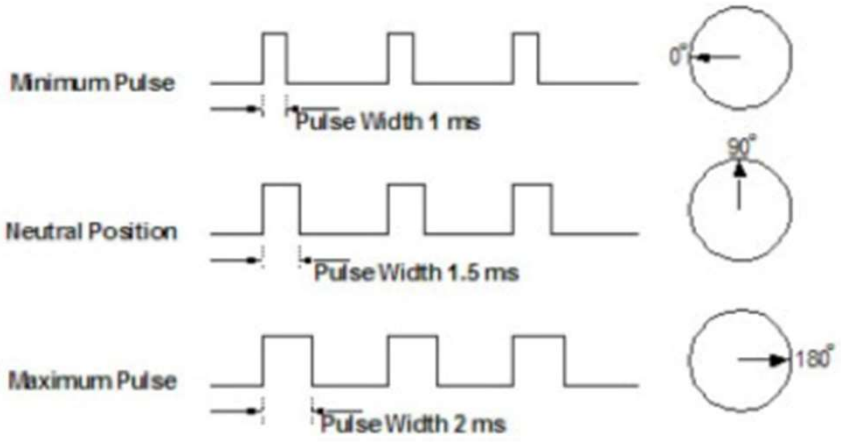
- 피드백(Feedback) 제어
  - 센서를 사용해서 상태에 따라 출력을 제어하는 기법
    - 가변저항을 사용하여 각도 인식
    - PWM 신호로 모터의 출력 제어
  - 피드포워드(Feedforward) 제어의 반대



서보모터 제어하기

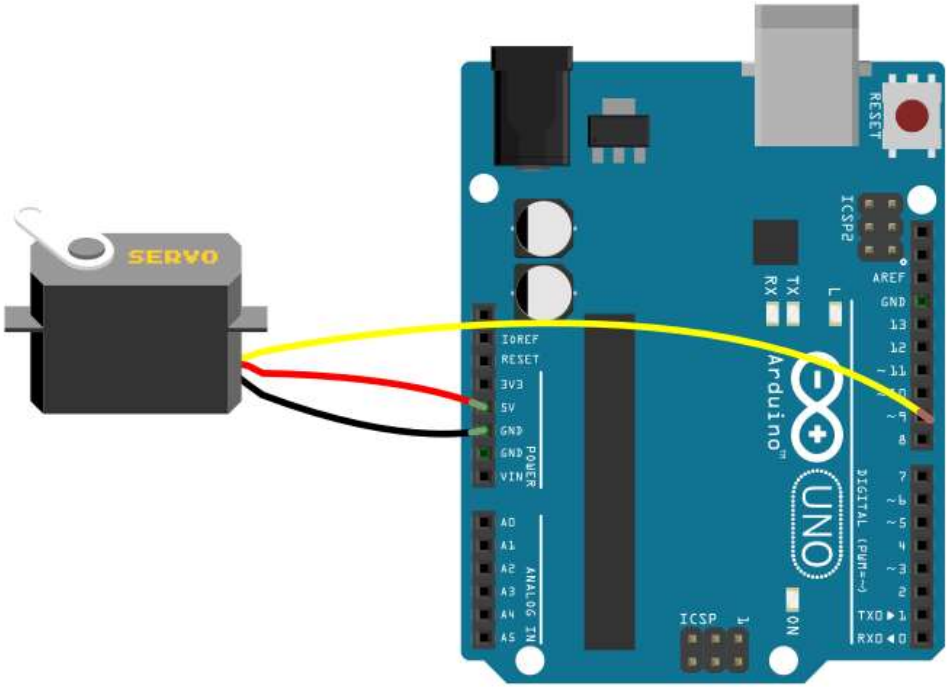
서보모터제어 방법

- PWM신호로 제어
  - analogWrite의 PWM과 주기가 다르다.
    - analogWrite: 31250Hz
    - 서보 모터: 500Hz
  - PWM 펄스 폭 시간에 따라 각도가 달라진다.



서보모터 제어하기

회로구현



서보모터 제어하기

아두이노 Servo 라이브러리

- 서보 모터를 쉽게 사용할 수 있도록 라이브러리가 만들어져 있다.
- 스케치에 Servo 라이브러리 추가
  - #include <Servo.h>
- Servo Class 선언
  - Servo [변수이름]
- Servo 함수 호출
  - Attach
  - Detach
  - Write

서보모터 제어하기

Servo.attach

- 아두이노 핀을 서보 모터 제어용 PWM 신호를 출력하도록 설정
- C언어 문법
  - void attach(int pin);
  - 반환값 없음, 인자 1개
- 사용방법
  - pin: 아두이노 핀 번호
  - 예)

    Servo myServo; // Servo Class 선언

    myServo.attach(3); // 3번 핀을 서보 모터용 PWM 출력으로 설정

서보모터 제어하기

Servo.detach

- 서보 모터 제어용 PWM 신호를 출력하는데 사용한 핀을 해제함
- C언어 문법
  - void detach();
  - 반환값 없음, 인자 없음
- 사용방법
  - 예)

```
Servo myServo;  // Servo Class 선언
myServo.attach(3); // 3번 핀에 서보 모터용 PWM 출력 시작
myServo.detach(); // 3번 핀에 서보 모터용 PWM 출력하지 않음
```

서보모터 제어하기

Servo.write

- PWM 펄스 폭을 조절하여 서보 모터의 각도 제어
- C언어 문법
  - void write(int angle);
  - 반환값 없음, 인자 1개
- 사용방법
  - angle: 서보 모터 회전 각도 (0 ~ 180 범위)
  - 예)

```
Servo myServo;  // Servo Class 선언
myServo.attach(3); // 3번 핀에 서보 모터용 PWM 출력 시작
myServo.write(90); // 3번 핀에 연결된 서보 모터 각도를 90도로 만든다.
```

서보모터 제어하기

소스코딩

```
#include <Servo.h>
Servo servo;

#define SERVO1_PIN 9

void setup() {
  servo.attach(SERVO1_PIN);
}

void loop() {
  servo.write(30);
  delay(500);
  servo.write(90);
  delay(500);
    servo.write(150);
  delay(500);
}
```

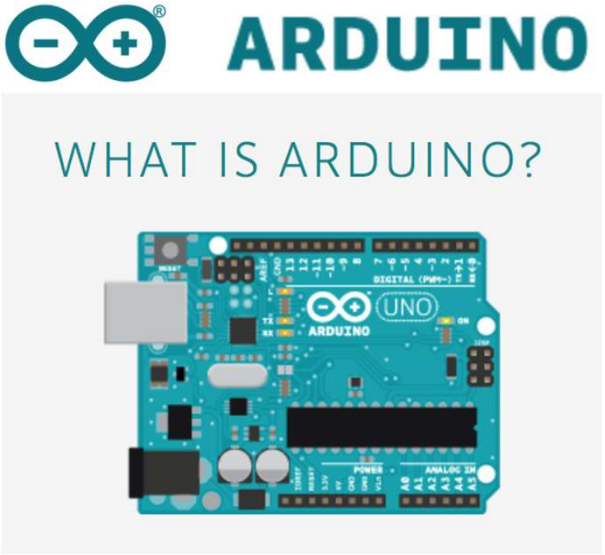
입력값을 이용하여 서보모터 제어하기

소스코딩

```
#include <Servo.h>
Servo servo;

#define SERVO_PIN 9

void setup() {
  Serial.begin(9600);
  servo.attach(SERVO_PIN);
}
void loop() {
  if (Serial.available() > 0) {
    String packet;
    packet = Serial.readStringUntil('\n');
    if (packet != 0) {
      int temp;
      temp =  packet.toInt();
      servo.write(temp);
      Serial.print("SERVO : ");
      Serial.println(temp);
    }
  }
}
```

Chapter 12

스텝모터 제어해보기



Step 1. 스텝모터 제어하기

Step 2. 입력값을 이용하여 스텝모터 제어하기

스텝모터 제어하기

스텝모터

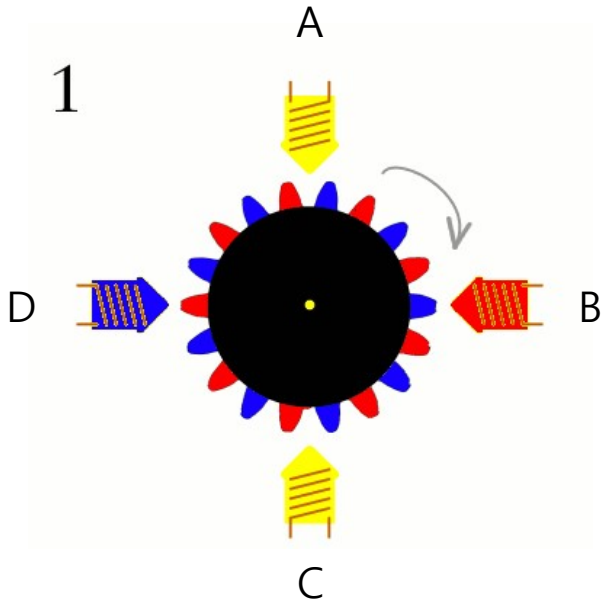
- 정확한 회전각, 회전속도를 제어할수 있는 모터
- 스텝모터, 스패터모터, 스테핑모터 등으로 불림
- 3D프린터 CNC 기계 등에 많이 사용됨
- 유니폴라(Unipolar)스텝모터, 바이폴라(Bipolar)스텝모터로 구분됨

스탭모터 28BYJ48 + ULN2003

스텝모터 제어하기

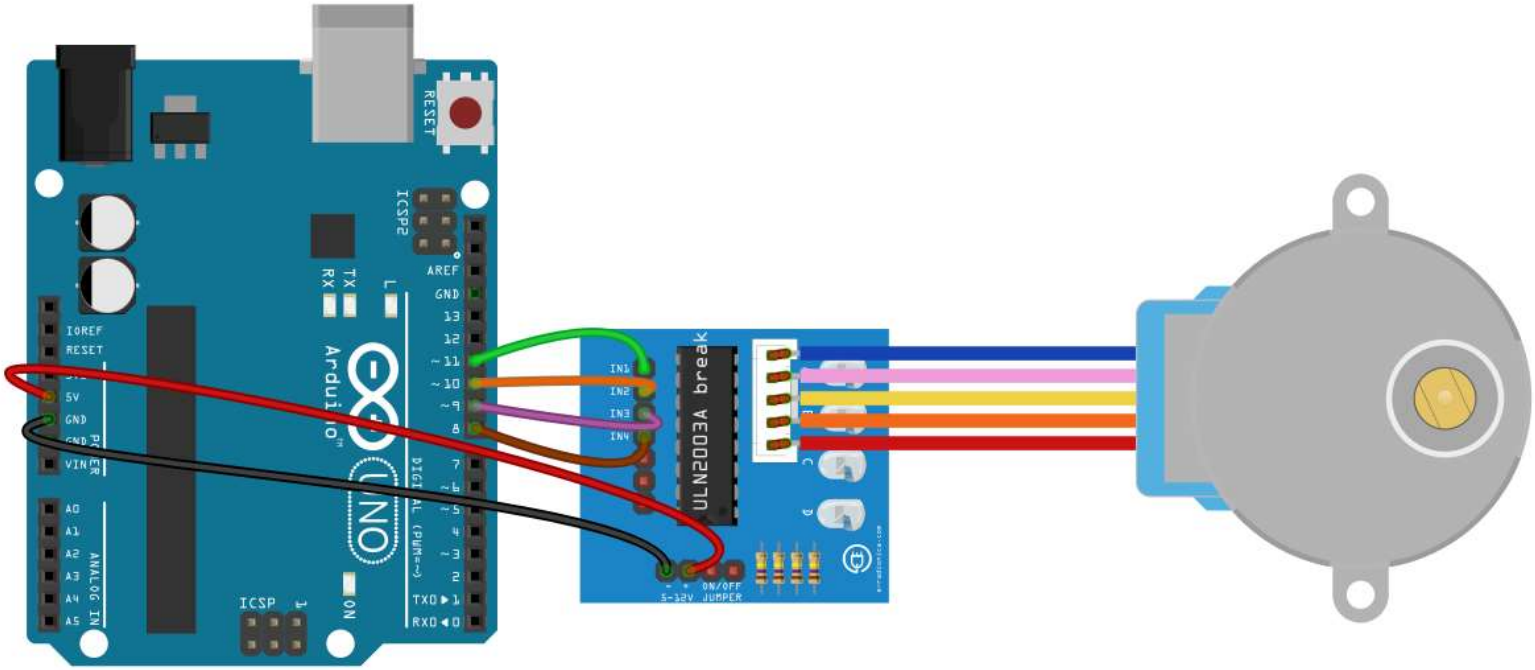
스텝모터의 원리

- 스텝모터는 크게 회전하는 회전자와 코일에 감겨 고정되어있는 고정자로 구성 됨
- 전자석은 서로 번갈아가면서 , 전기신호를 받고 그에 의해서 한 스텝을 회전함.
- A-B, B-C, C-D, D-A로 전자석에 들어오는 전류에 따라 회전자가 회전을 함



스텝모터 제어하기

회로구현 (28BYJ48, UNL2003)



스텝모터 제어하기

소스코딩

```
#define IN1 11
#define IN2 10
#define IN3 9
#define IN4 8

void setup() {
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
}

void loop() {
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  delay(10);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  delay(10);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  delay(10);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  delay(10);
}
```

스텝모터 제어하기

아두이노 Stepper 라이브러리

- 스텝모터를 쉽게 사용할 수 있도록 라이브러리가 만들어져 있다.
- 스케치에 Stepper 라이브러리 추가
  - #include <Stepper.h>
- Stepper Class 선언
  - Stepper [변수이름](steps,IN4,IN2,IN3,IN1)
- Stepper 함수 호출
  - setSpeed
  - step

스텝모터 제어하기

Stepper.setSpeed

- 분당 회전수(RPM)로 모터 속도를 설정.
- C언어 문법
  - void setSpeed(int rpm);
  - 반환값 없음, 인자 1개
- 사용방법
  - rpm : 모터가 분당 회전해야하는 속도
  - 예)

    Stepper myStepper(steps, 11, , 10, 11); // strpper Class 선언

    myStepper.setSpeed(15); // RPM 15으로 속도 설정

스텝모터 제어하기

Stepper.step

- setSpeed() 에서 결정된 속도로 특정 단계수만큼 회전
- C언어 문법
  - void step(int steps);
  - 반환값 없음, 인자 1개
- 사용방법
  - Steps : 모터를 돌리는 단계의 수를 뜻하며, 양수일때 정방향 음수일때 역방향
  - 예)

```
Stepper myStepper(steps, 8, 9, 10, 11); // strpper Class 선언
myStepper.setSpeed(60); // RPM 60으로 속도 설정
myStepper.step(steps); // step 만큼 회전
```

스텝모터 제어하기

소스코딩

```
#include <Stepper.h>
 #define IN1 11
 #define IN2 10
 #define IN3 9
 #define IN4 8
const int steps = 2048; //360도 :2048 180도 :1024

Stepper myStepper(steps,IN4,IN2,IN3,IN1);

void setup() {
  myStepper.setSpeed(17);  //모터의 회전속도 설정
}
void loop() {
  myStepper.step(steps);
  delay(1000);

  myStepper.step(-steps);
  delay(1000);
}
```

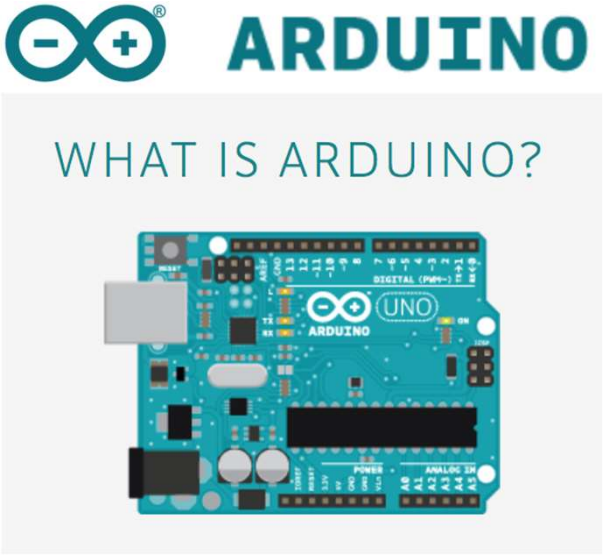
입력값을 이용하여 스텝모터 제어하기

소스코딩

```
#include <Stepper.h>
 #define IN1 11
 #define IN2 10
 #define IN3 9
 #define IN4 8
const int steps = 2048;

Stepper myStepper(steps,IN4,IN2,IN3,IN1);

void setup() {
 Serial.begin(9600);
 myStepper.setSpeed(17);
}
void loop() {
  if (Serial.available() > 0) {
    String packet;
    packet = Serial.readStringUntil('\n');
  if (packet != 0) {
    int temp;
    temp =  packet.toInt();
    temp = (2048.0 / 360.0) * temp;
    Serial.print("steeper : ");
    Serial.println(temp);
    myStepper.step(temp);
    }
  }
}
```

Chapter 13

DC모터 제어해보기



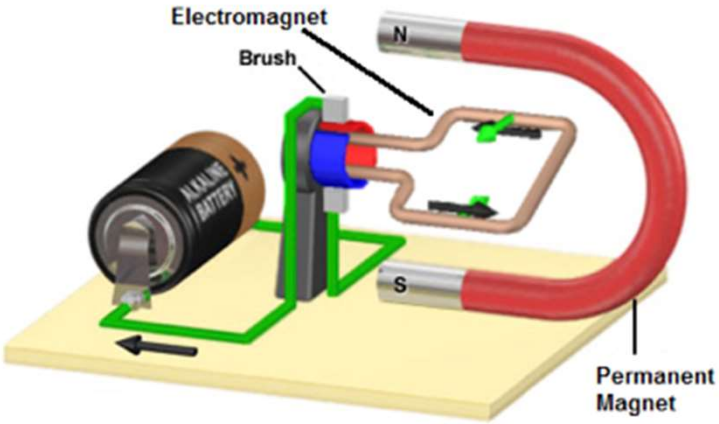
Step 1. DC모터 제어하기

Step 2. 입력값을 이용하여 DC모터 제어하기

DC모터 제어하기

DC모터

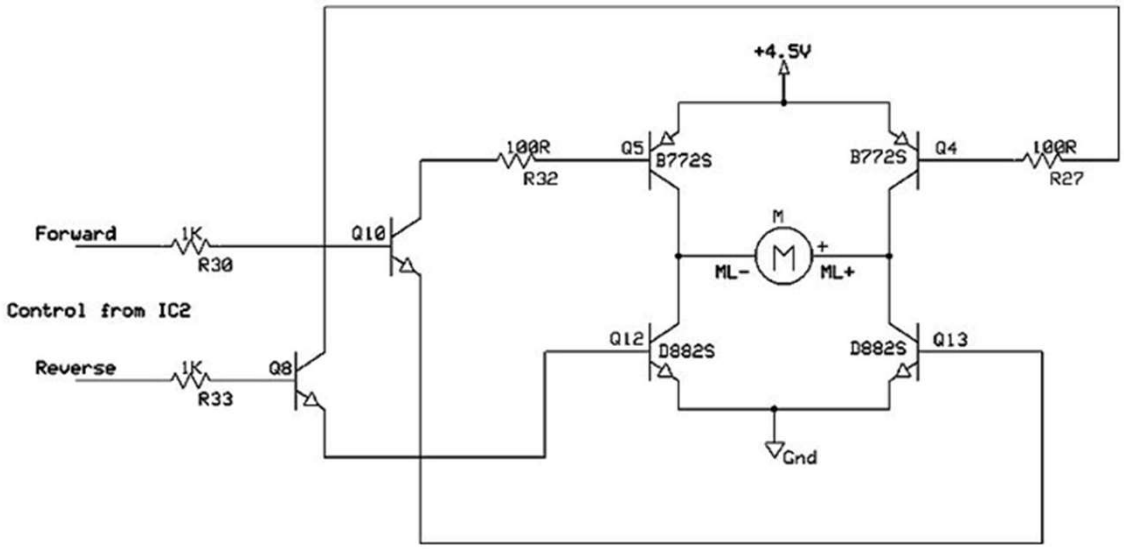
- 전기의 힘을 회전 운동으로 바꿔주는 장치
- 핀 극성 없음
- 정격
  - 모터 종류마다 다르다.



DC모터 제어하기

DC모터 제어

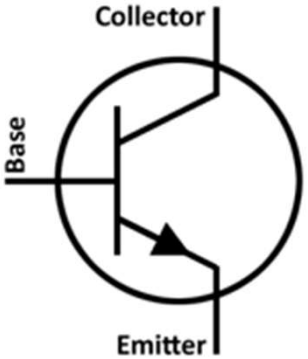
- H-bridge 회로
  - 모터의 회전 방향 및 속도를 제어하기 위한 회로

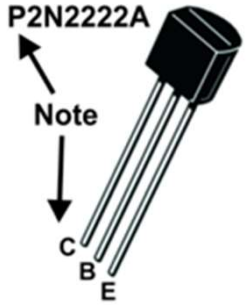


DC모터 제어하기

트랜지스터(Transistor)

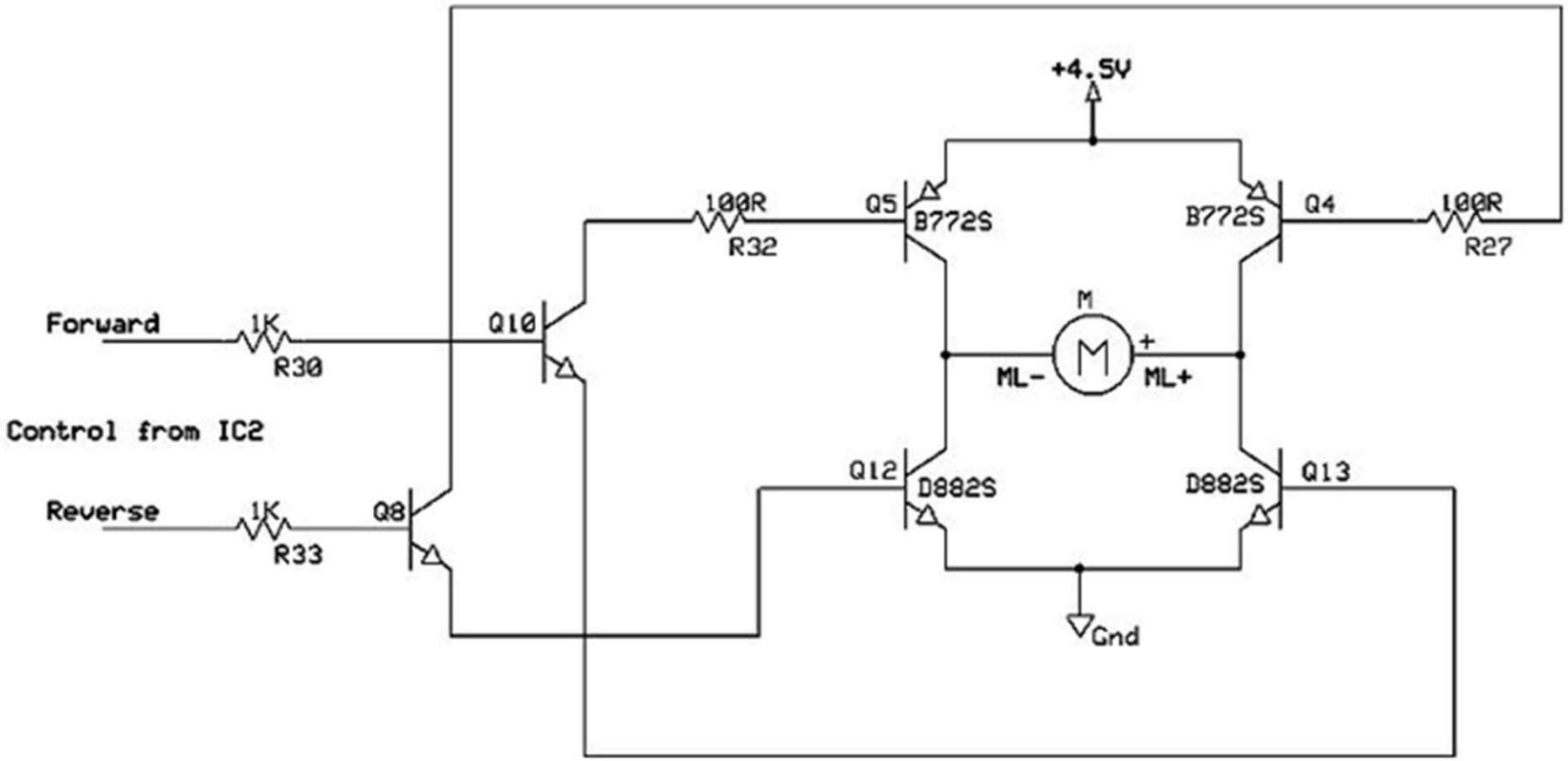
- 적은 전기적 힘으로 큰 전기적 힘을 제어하기 위한 부품
- 전기적 스위치 역할
  - Base 5V: Collector -> Emitter간 전류 흐름
  - Base 0V: Collector -> Emitter간 전류 흐르지 않음





DC모터 제어하기

H-bridge 회로 작동 방식



DC모터 제어하기

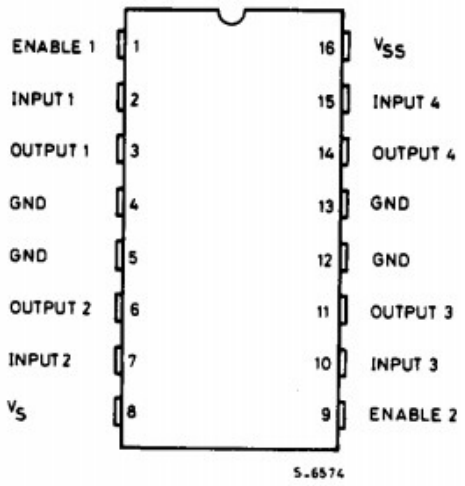
모터 드라이버 IC

- 모터 제어 회로의 복잡성으로 인해 직접 구현하지 않고 필요한 기능이 내장된 IC를 사용함
- 2개의 모터를 제어 할수 있음.
- 정회전, 역회전 제어 가능함.

DC모터 제어하기

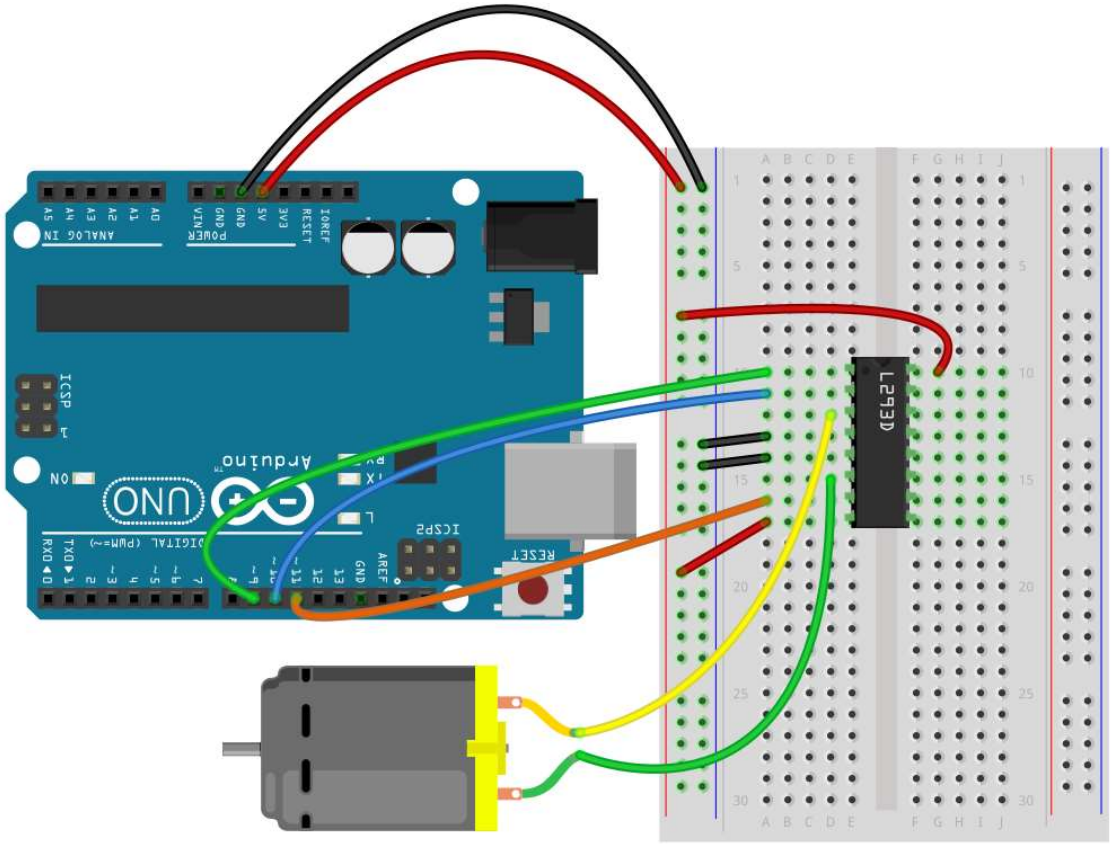
L293D 모터 드라이버

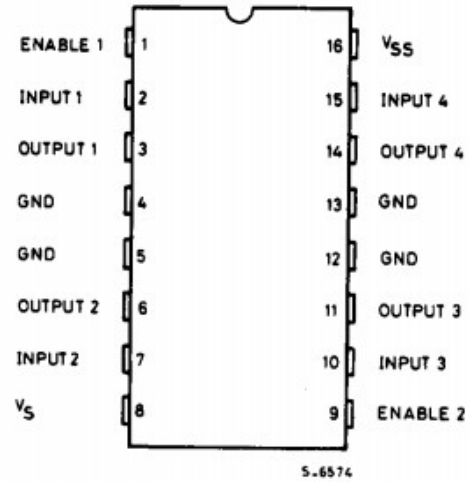
- Vs: 5V에 연결
  - 모터 드라이버 자체 전원 공급
- Vss: 5V에 연결
  - 내부로직 전원 공급
- GND : GND에 연결
- ENABLE1 : 속도제어 LOW시 정지
- INPUT1,INPUT2 : 아두이노 연결
  - 정회전, 역회전을 결정 함
- OUTPUT1,OUTPUT2 : DC모터에 연결



DC모터 제어하기

회로구현





DC모터 제어하기

L293D 모터 드라이버

ENABLE	INPUT1	INPUT2	모터 상태
PWM	LOW	LOW	정지
PWM	HIGH	LOW	A방향 회전
PWM	LOW	HIGH	B방향 회전
PWM	HIGH	HIGH	정지
GND(LOW)			정지

DC모터 제어하기

소스코딩

```
#define ENABLE_PIN 9
#define INPUT1_PIN 10
#define INPUT2_PIN 11

void setup() {
  pinMode(ENABLE_PIN, OUTPUT);
  pinMode(INPUT1_PIN, OUTPUT);
  pinMode(INPUT2_PIN, OUTPUT);

}

void loop() {
  analogWrite(ENABLE_PIN, 200);
  digitalWrite(INPUT1_PIN, HIGH);
  digitalWrite(INPUT2_PIN, LOW);
  delay(1000);

  analogWrite(ENABLE_PIN, 0);
  digitalWrite(INPUT1_PIN, LOW);
  digitalWrite(INPUT2_PIN, LOW);
  delay(1000);
```

```
 analogWrite(ENABLE_PIN, 200);
  digitalWrite(INPUT1_PIN, LOW);
  digitalWrite(INPUT2_PIN, HIGH);
  delay(1000);

   analogWrite(ENABLE_PIN, 0);
  digitalWrite(INPUT1_PIN, HIGH);
  digitalWrite(INPUT2_PIN, HIGH);
  delay(1000);
}
```

입력값을 이용하여 DC모터 제어하기

소스코딩

```
#define ENABLE_PIN 9
#define INPUT1_PIN 10
#define INPUT2_PIN 11

void setup() {
  pinMode(ENABLE_PIN, OUTPUT);
  pinMode(INPUT1_PIN, OUTPUT);
  pinMode(INPUT2_PIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String packet;
    packet = Serial.readStringUntil('\n');
    if (packet != 0) {
      int temp;
      temp =  packet.toInt();
      Serial.print("state : ");
      Serial.println(temp);
      if(temp == 1){
        analogWrite(ENABLE_PIN, 200);
        digitalWrite(INPUT1_PIN, HIGH);
        digitalWrite(INPUT2_PIN, LOW);
      }
      else if(temp == -1){
        analogWrite(ENABLE_PIN, 200);
        digitalWrite(INPUT1_PIN, LOW);
        digitalWrite(INPUT2_PIN, HIGH);
      }
      else{
        analogWrite(ENABLE_PIN, 0);
        digitalWrite(INPUT1_PIN, LOW);
        digitalWrite(INPUT2_PIN, LOW);
      }
    }
  }
}
```