


LCD, 서보모터, DC모터, 초음파센서, 릴레이

강성관
(스마트인재개발원 부장, 공학박사)



 LCD

 서보모터

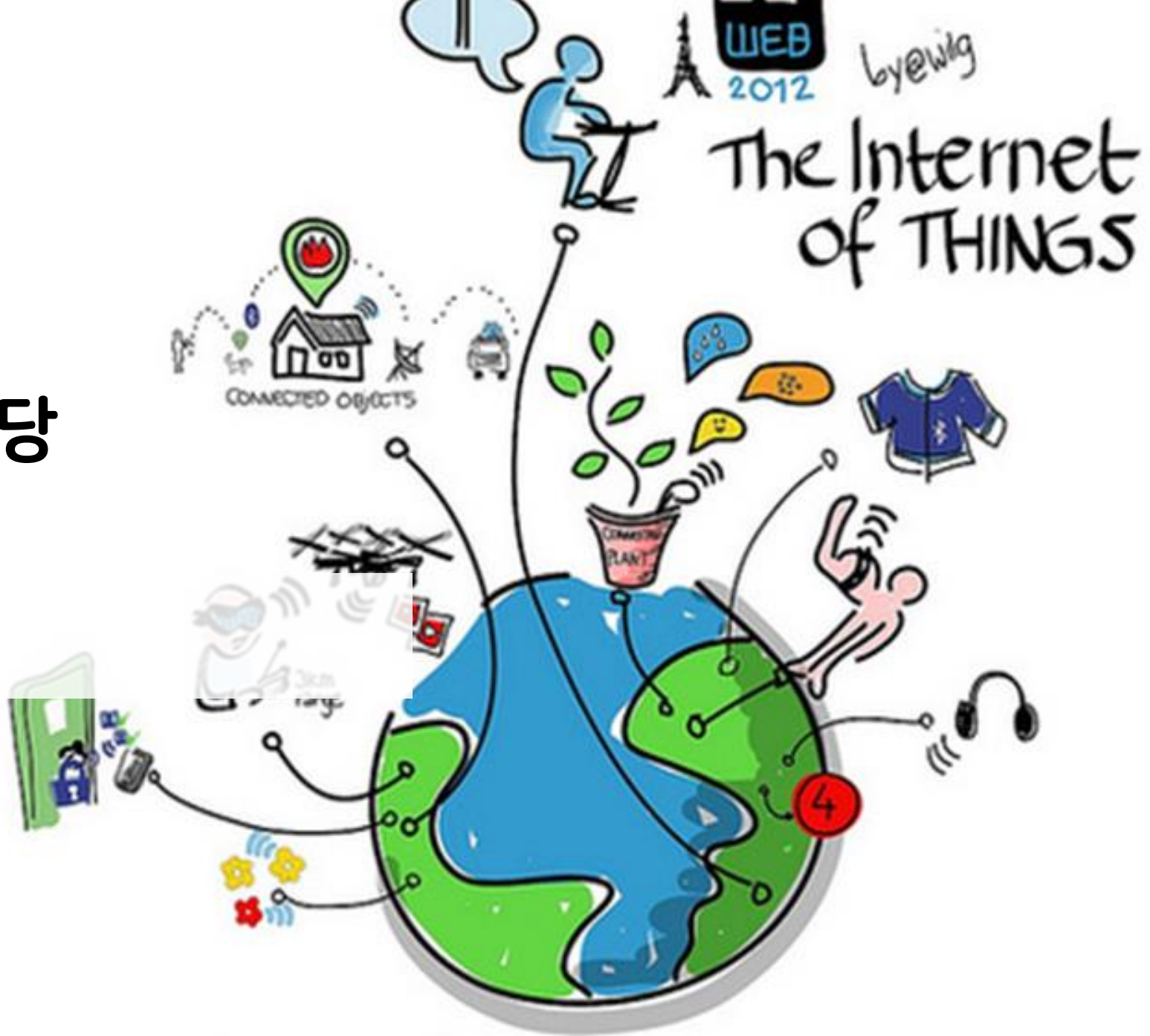
 DC 모터

 초음파 센서

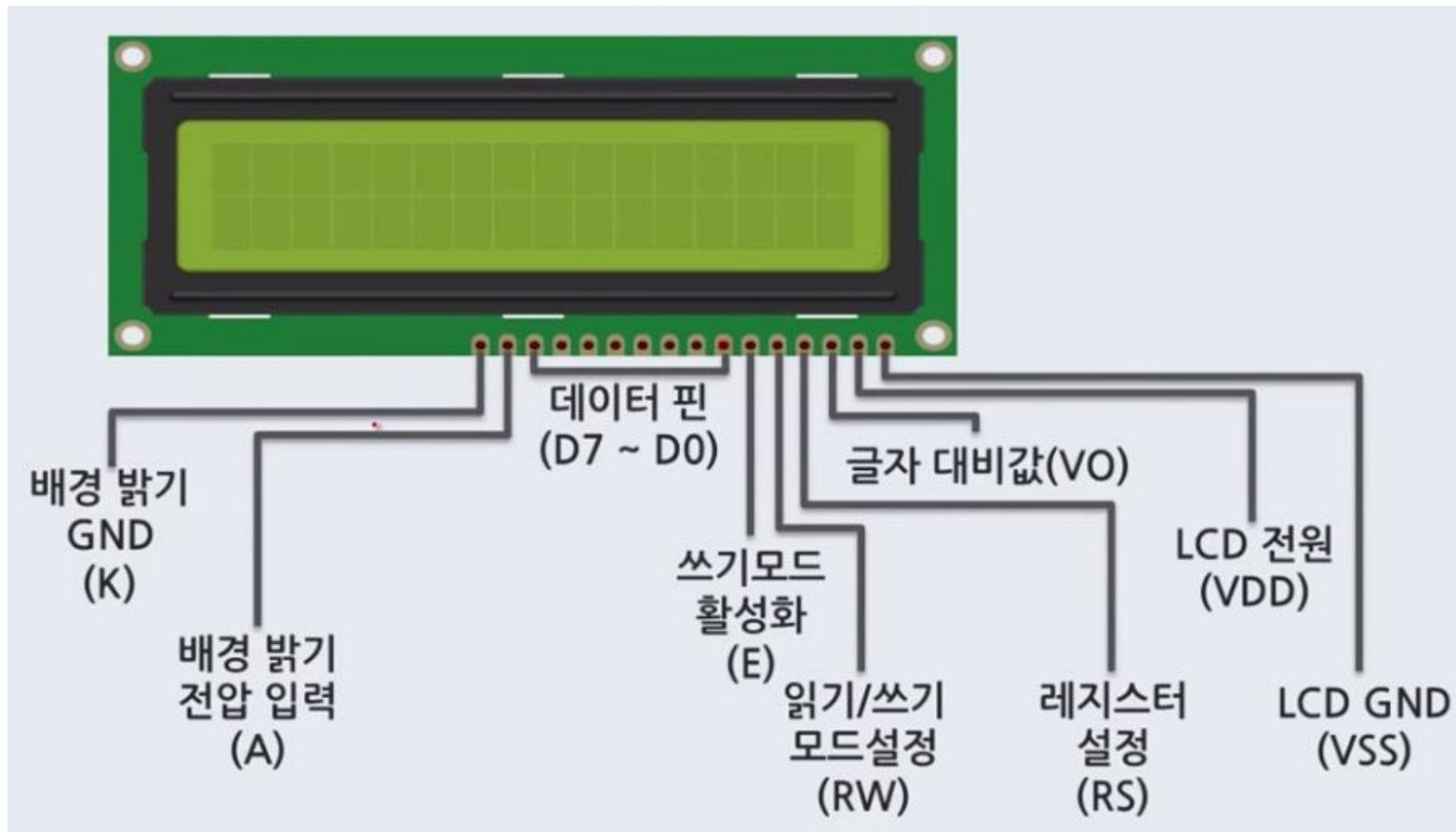
 릴레이

첫째 마당

LCD



- 2 x 16열의 문자 (숫자, 영문, 특수문자)를 출력하는 LCD 패널
- 15번과 16번은 Back Light 전원으로 BL1이나 BL2에 저항을 사용



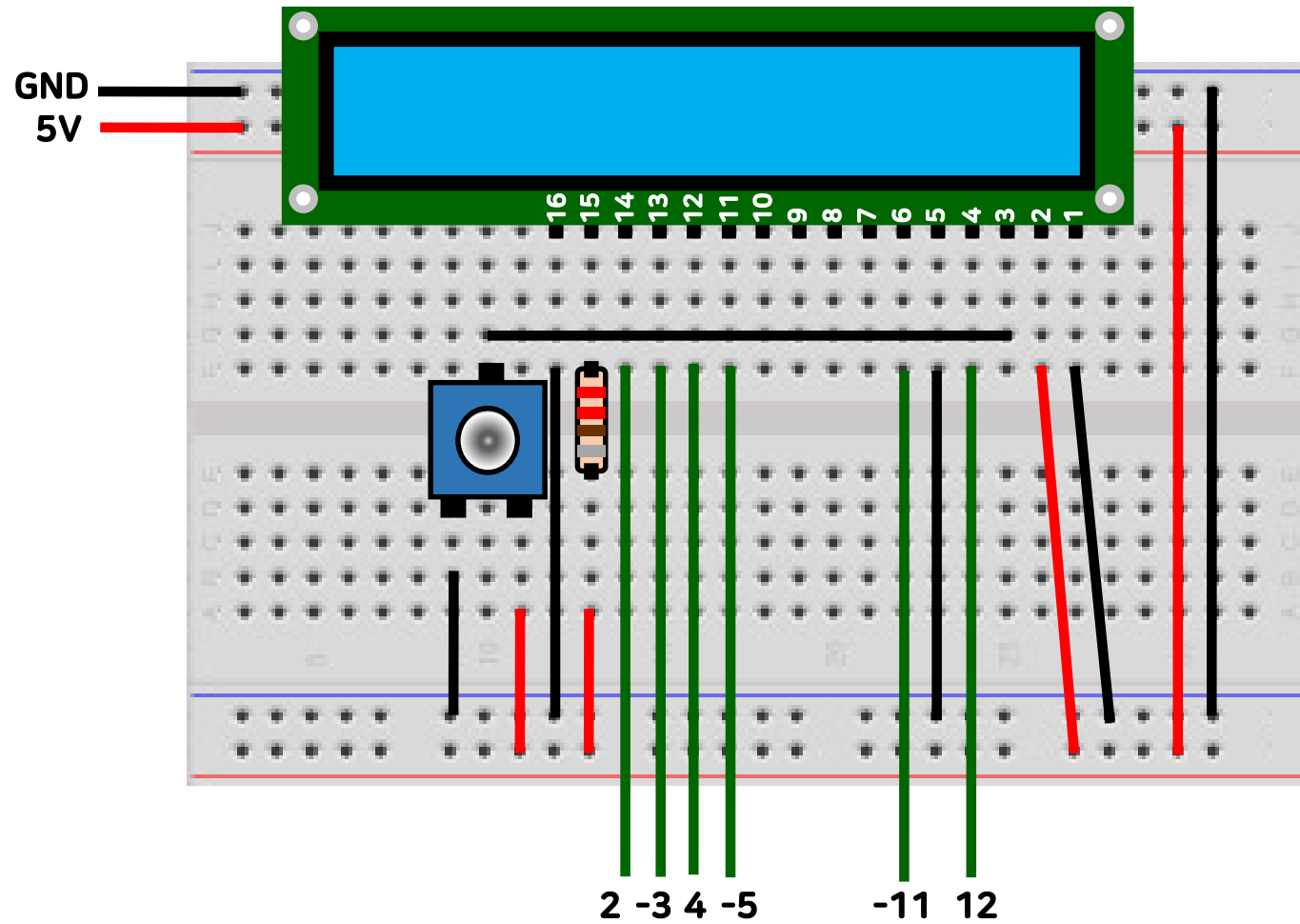
핀 번호	기 호	레 벨	기 능		
1	VSS	-	pV		
2	VDD	-	+5V		
3	Vo		도트 밝기 조정(가변저항)		
4	RS	H/L	L : 명령 입력 (IR 선택) H : 데이터 입력 (DR 선택)		
5		H/L	L : 쓰기(CPU → LCD module) H : 읽기(CPU ← LCD module)		
6	E	H	LCD 모듈의 허가 신호		
7	DB0	H/L	4비트 데이터 버스 이용시 사용 불가	8비트 데이터 버스 이용시 모두 사용	
8	DB1	H/L			
9	DB2	H/L			
10	DB3	H/L			
11	DB4	H/L	4비트 데이터 버스 이용시 사용 가능		
12	DB5	H/L			
13	DB6	H/L			
14	DB7	H/L			
15	LED+		LCD 백라이트		
16	LED -				

+, -

글자 밝기

Data 저장, 읽기, 쓰기

LCD배경 밝기



- **LiquidCrystalcd()** : LCD와 연결되는 아두이노 핀 번호 설정

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

- **begin(col, row)** : LCD의 열의 크기와 행의 크기를 설정 (16, 2)
- **print()** : LCD에 해당 내용을 출력
- **setCursor(col, row)** : LCD의 커서를 (col열, row행)에 위치시킴

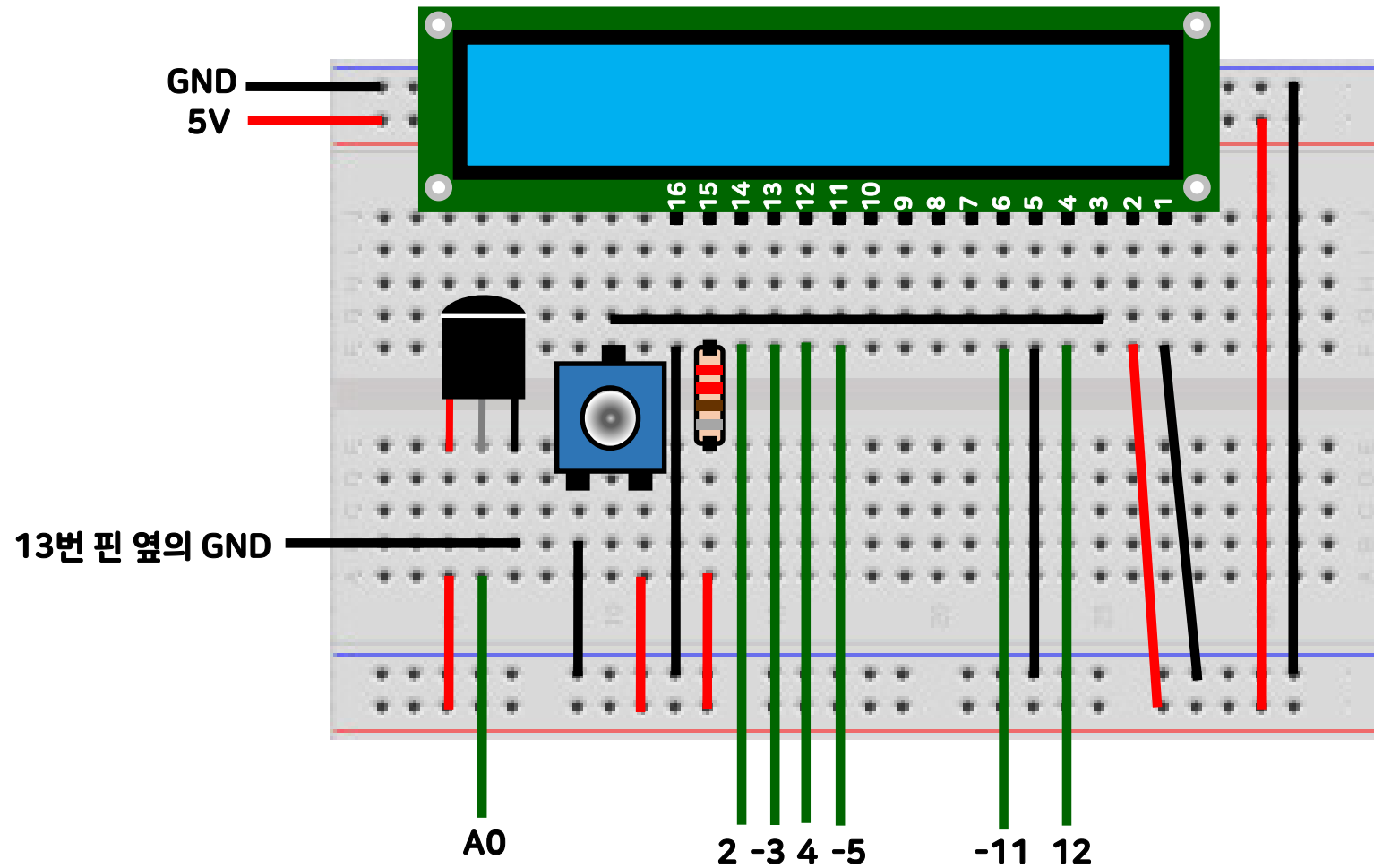
0,0	1,0	2,0	...	13,0	14,0	15,0
0,1	1,1	2,1		13,1	14,1	15,1

- **clear()** : LCD의 내용을 지운다

- LCD에 "Hello World !!"가 출력되고 다음 줄에 시간이 1초마다 초로 출력

1	#include <LiquidCrystal.h>	1	라이브러리 등록
2		2	
3	const int numRows = 2;	3	LCD의 행의 수
4	const int numCols = 16;	4	LCD의 열의 수
5		5	
6	LiquidCrystal lcd(12, 11, 5, 4, 3, 2);	6	아두이노 핀 번호로 라이브러리를 초기화
7		7	
8	void setup() {	8	
9	lcd.begin(numCols, numRows);	9	LCD 상에 열과 행의 수를 설정
10	lcd.print("Hello World !!");	10	LCD에 내용을 출력
11	}	11	
12		12	
13	void loop() {	13	
14	lcd.setCursor(0, 1);	14	0열 1행에 커서를 위치시킴 (2번째 줄)
15	lcd.print(millis() / 1000);	15	시간을 1초마다 초로 출력
16	delay(1000);	16	
	}		

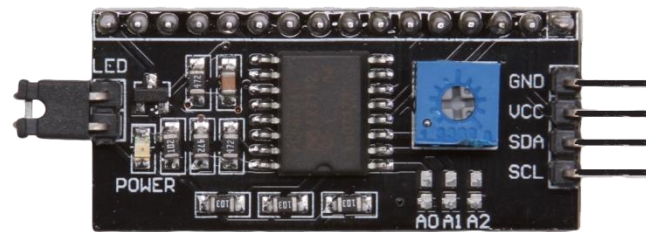
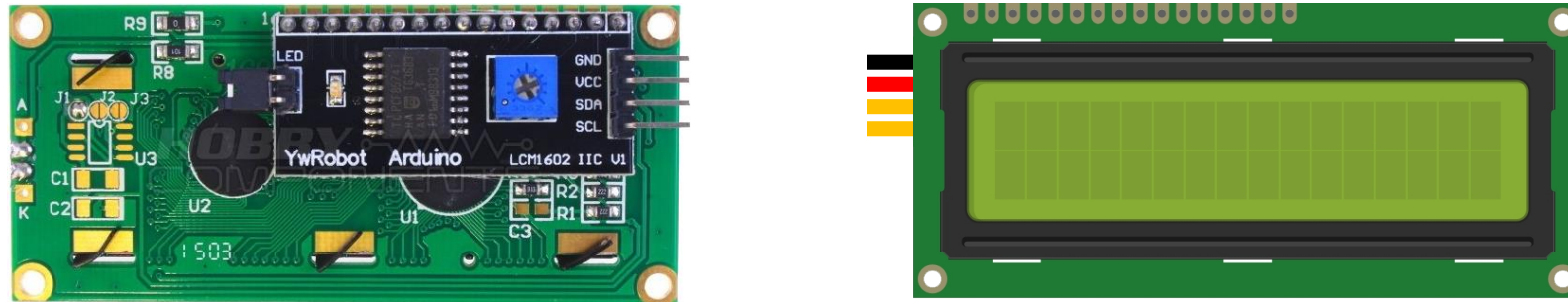
온도 센서값 출력하기



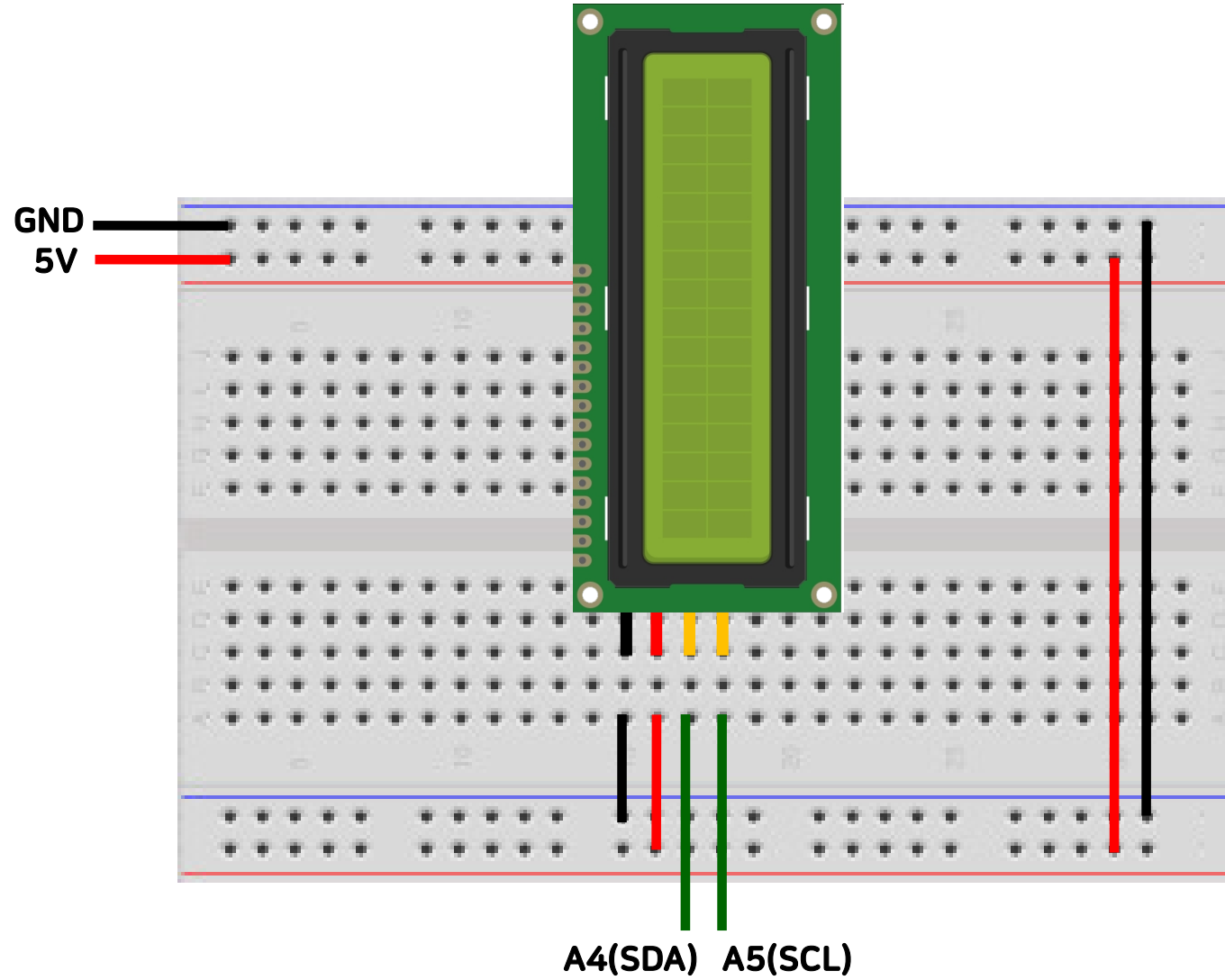
온도 센서값 출력하기

1	#include <LiquidCrystal.h>	1	LCD 라이브러리
2		2	
3	const int numRows = 2;	3	LCD 줄 수
4	const int numCols = 16;	4	LCD 컬럼 수
5	const int sensorPin = A0;	5	온도 센서 핀
6		6	
7	LiquidCrystal lcd(12, 11, 5, 4, 3, 2);	7	LCE 연결 핀 번호
8		8	
9	void setup() {	9	
10	lcd.begin(numCols, numRows);	10	LCD 시작
11	}	11	
12	void loop() {	12	
13	int sensorVal = analogRead(sensorPin);	13	
14	float voltage = (sensorVal / 1024.0) * 5.0;	14	
15	float temp = (voltage - 0.5) * 100;	15	
16		16	
17	lcd.setCursor(0, 1);	17	LCD에 커서를 0,1 에 위치 시킨다
18	lcd.print(temp);	18	LCD에 온도를 출력한다.
19	delay(500);	19	
20	}	20	

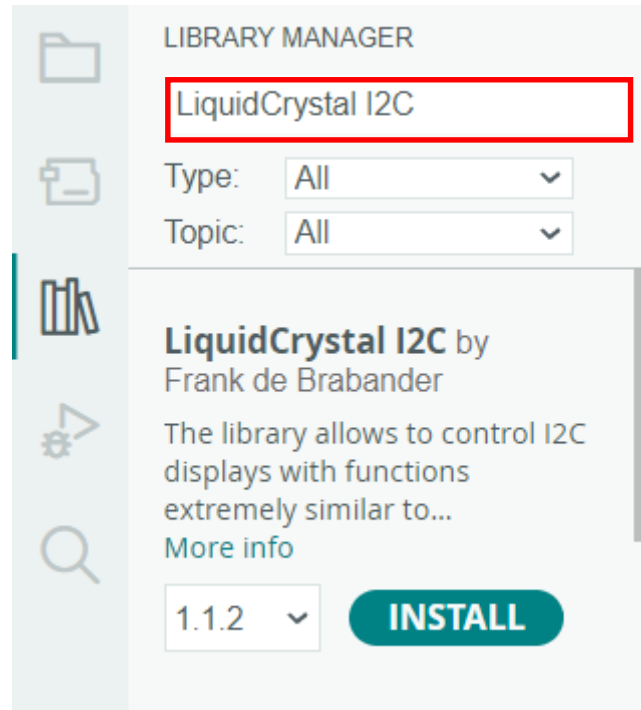
- I2C 모듈을 부착한 LCD 패널을 사용할 것을 권장



I2C 통신 모듈



- Library Manager 아이콘을 클릭하고 LiquidCrystal I2C로 검색하여 라이브러리를 설치한다.



- **LiquidCrystal_I2C** : LCD의 주소와 가로 세로 셀의 갯수를 설정
- **init()** : LCD를 시작
- **backlight()** : LCD의 백라이트를 켜다
- **print()** : LCD에 해당 내용을 출력
- **setCursor(col, row)** : LCD의 커서를 (col열, row행)에 위치시킴

- I2C 주소를 검색하는 코드

1	#include <Wire.h>	1	라이브러리 등록
2		2	
3	void setup() {	3	
4	Serial.begin (9600);	4	
5	Wire.begin();	5	I2C 통신 시작
6	while (!Serial) { }	6	시리얼 통신 장치가 있는 경우만 통과
7	Serial.println("");	7	
8		8	
9	for (byte i = 1; i < 120; i++) {	9	
10	Wire.beginTransmission (i);	10	1-120번 주소의 장치를 검색
11		11	
12	if (Wire.endTransmission () == 0) {	12	I2C 통신이 가능한 장치가 있다면
13	Serial.print("Address: ");	13	주소를 출력
14	Serial.print("0x"); Serial.print(i, HEX);	14	
15	delay (10);	15	
16	}	16	
17	}	17	
18	}	18	
19		19	
20	void loop() {}	20	

- I2C 주소 검색 결과

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM7')

Address: 0x27

- 예제 : LCD에 "Hello World !!"가 출력되고 다음 줄에 시간이 1초마다 초로 출력

1	#include <LiquidCrystal_I2C.h>	1	라이브러리 등록
2		2	
3	LiquidCrystal_I2C lcd(0x27, 16, 2);	3	검색된 I2C 주소와 열과 행의 수를 설정
4		4	
5	void setup() {	5	
6	lcd.init();	6	LCD를 시작
7	lcd.backlight();	7	LCD 백라이트를 ON
8	lcd.setCursor(0, 0);	8	
9	lcd.print("Hello World !!");	9	LCD에 내용을 출력
10	}	10	
11		11	
12	void loop() {	12	
13	lcd.setCursor(0, 1);	13	0열 1행에 커서를 위치시킴 (2번째 줄)
14	lcd.print(millis() / 1000);	14	시간을 1초마다 초로 출력
15	}	15	

- **blink()** : 커서를 깜빡이게 한다
- **noBlink()** : 커서를 깜빡이지 않게 한다
- **noDisplay()** : LCD의 내용을 보이지 않게 한다
- **display()** : LCD의 내용을 보이게 한다

- **scrollDisplayLeft()** : LCD 내용을 왼쪽으로 스크롤한다
- **scrollDisplayRight()** : LCD 내용을 오른쪽으로 스크롤한다

LCD 패널 제어

1	#include <LiquidCrystal_I2C.h>	1	
2		2	
3	LiquidCrystal_I2C lcd(0x27, 16, 2);	3	
4		4	
5	void setup() {	5	
6	lcd.init();	6	
7	lcd.backlight();	7	
8	}	8	
9		9	
10	void loop() {	10	커서를 0열 0행에 위치 시킨다
11	lcd.setCursor(0,0);	11	
12	lcd.print("Cursor blink");	12	커서를 깜빡인다
13	lcd.blink();	13	
14	delay(2000);	14	
15		15	
16	lcd.clear();	16	커서를 깜빡이지 않게 한다
17	lcd.noBlink();	17	
18	lcd.print("Cursor noBlink");	18	
19	delay(2000);	19	
20	lcd.clear();	20	
21	lcd.print("Display OFF");	21	
22	delay(1000);	22	
23	lcd.noDisplay();	23	LCD의 내용을 보이지 않게 한다
24	lcd.clear();	24	
25	delay(2000);	25	
26		26	
27	lcd.print("Display ON");	27	
28	lcd.display();	28	LCD의 내용을 보이게 한다
29	delay(2000);	29	
30	lcd.clear();	30	
31	}	31	

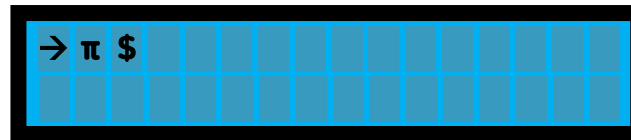
[illegible][illegible]

```
Cursor noBlink
```

[illegible][illegible][illegible]

1	#include <LiquidCrystal_I2C.h>	1	
2		2	
3	LiquidCrystal_I2C lcd(0x27, 16, 2);	3	
4		4	
5	const char data[] = "Hellow World !!";	5	
6	const int textSize = sizeof(data) - 1;	6	
7		7	
8	void setup() {	8	
9	lcd.init();	9	
10	lcd.backlight();	10	
11	lcd.print(data);	11	
12	}	12	
13		13	
14	void loop() {	14	
15	for(int pos=0; pos<textSize; pos++) {	15	LCD 내용을 왼쪽으로 한 칸씩 이동한다
16	lcd.scrollDisplayLeft();	16	
17	delay(200);	17	
18	}	18	
19	}	19	

- 문자 코드는 <http://www.sparkfun.com/datasheets/LCD/HD44780.pdf>의 17페이지에서 확인할 수 있다.

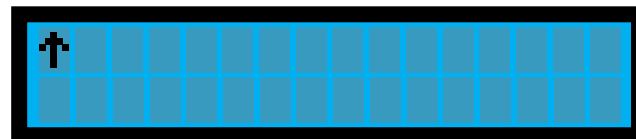


1	#include <LiquidCrystal_I2C.h>	1	
2		2	
3	LiquidCrystal_I2C lcd(0x27, 16, 2);	3	
4		4	
5	const byte arrowSymbol = B01111110;	5	→ 문자 코드
6	const byte piSymbol = B11110111;	6	π 문자 코드
7	const byte dollarSymbol = B00100100;	7	\$ 문자 코드
8		8	
9	void setup() {	9	
10	lcd.init();	10	
11	lcd.backlight();	11	
12	lcd.clear();	12	문자 코드를 출력한다
13	lcd.write(arrowSymbol);	13	
14	lcd.write(piSymbol);	14	
15	lcd.write(dollarSymbol);	15	
16	}	16	
17		17	
18	void loop() {	18	
19	}	19	

- 8행 5열로 된 배열로 사용자 문자를 정의할 수 있다

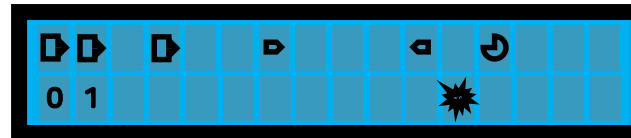
B00000					
B00100					
B01110					
B10101					
B00100					
B00100					
B00100					
B00000					

- **createChar(1, 변수)** : 변수에 정의된 데이터를 사용하여 문자를 만들어서 문자 1에 할당한다



1	#include <LiquidCrystal_I2C.h>	1	
2		2	
3	LiquidCrystal_I2C lcd(0x27, 16, 2);	3	
4		4	
5	byte arrow[8] = {	5	사용자 문자 정의
6	B00000,	6	
7	B00100,	7	
8	B01110,	8	
9	B10101,	9	
10	B00100,	10	
11	B00100,	11	
12	B00100,	12	
13	B00000	13	
14	};	14	
15		15	
16	void setup() {	16	
17	lcd.createChar(1, arrow);	17	정의된 문자를 문자 1에 할당한다
18	lcd.init();	18	
19	lcd.backlight();	19	문자 1에 정의된 문자를 출력한다
20	lcd.write(1);	20	
21	}	21	
22		22	
23	void loop() {	23	
24	}	24	

- 전투기, 총알, 적군, 점수, 폭발 이미지
- 버튼 1, 2로 전투기 상하 이동
- 버튼 3으로 공격 - 총알 발사
- 적군은 전투기와의 거리보다 1-2초만큼만 보여주고 사라짐
- 적군의 총알은 피해야 함



Serial 통신으로 수신한 결과 출력하기

1	#include <LiquidCrystal_I2C.h>	1	
2		2	
3	LiquidCrystal_I2C lcd(0x27, 16, 2);	3	
4		4	
5	void setup() {	5	
6	lcd.init();	6	
7	lcd.backlight();	7	
8	}	8	
9		9	
10		10	
11	void loop() {	11	
12	if (Serial.available()) {	12	
13	delay(100);	13	
14	lcd.clear();	14	
15	while (Serial.available() > 0) {	15	
16	lcd.write(Serial.read());	16	
17	}	17	LCD에 수신 결과 쓰기
18	}	18	
19	}	19	
20		20	
21		21	

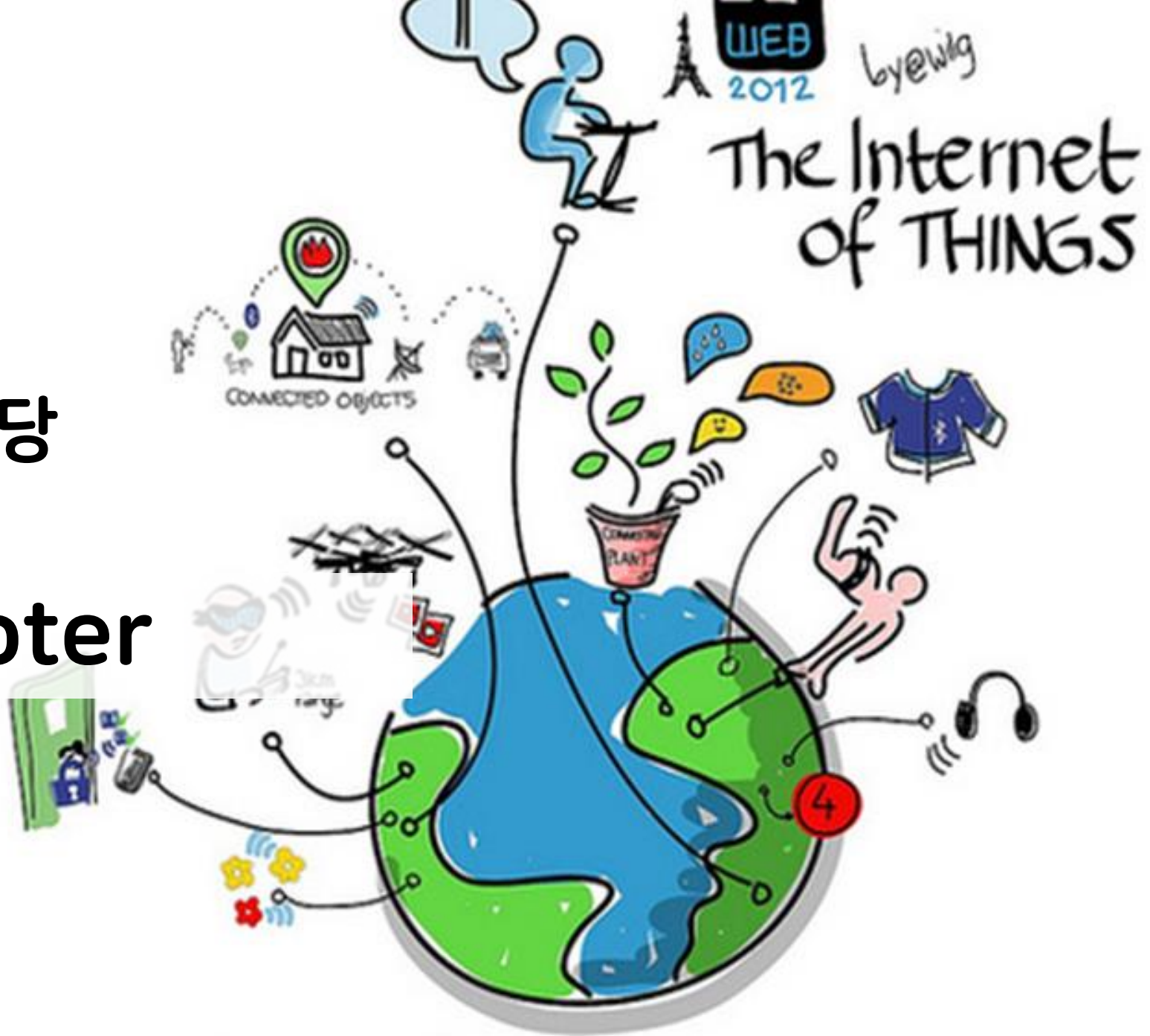
Serial 통신으로 LCD 스크롤 제어하기

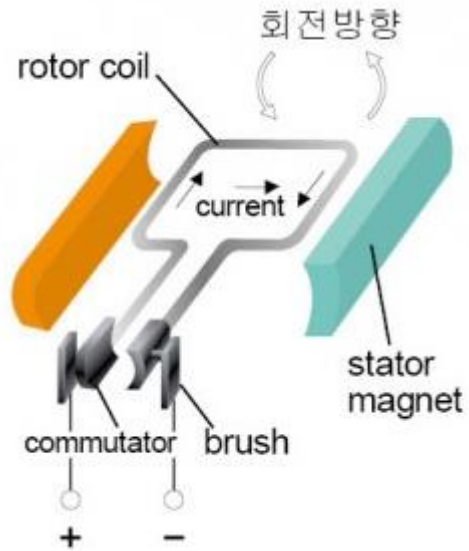
```
1 #include <LiquidCrystal_I2C.h>
2
3 LiquidCrystal_I2C lcd(0x27, 16, 2);
4
5 void setup() {
6     lcd.init();
7     lcd.backlight();
8 }
9
10 void loop() {
11     if (Serial.available()) {
12         delay(100);
13         while (Serial.available() > 0) {
14             char lcdChar = Serial.read();
15             if(lcdChar == '6'){
16                 lcd.scrollDisplayRight();
17             } else if(lcdChar == '4'){
18                 lcd.scrollDisplayLeft();
19             } else if(lcdChar == '2'){
20                 lcd.clear();
21                 lcd.setCursor(0,1);
22                 lcd.write("Hello, IoT");
23             } else if(lcdChar == '8'){
24                 lcd.clear();
25                 lcd.setCursor(0,0);
26                 lcd.write("Hello, IoT");
27             }
28             delay(200);
29         }
30     }
31 }
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14 수신한 문자가 6이라면
15 오른쪽으로 스크롤
16 수신한 문자가 4라면
17 왼쪽으로 스크롤
18 수신한 문자가 2라면
19 화면을 지우고 (0, 1)로 이동해서 내용을 출력
20
21
22 수신한 문자가 8이라면
23 화면을 지우고 (0, 0)로 이동해서 내용을 출력
24
25
26
27
28
29
30
31
```

 둘째 마당

Servo Moter

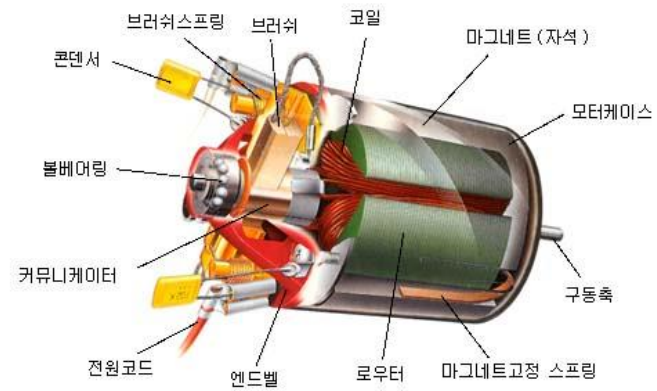




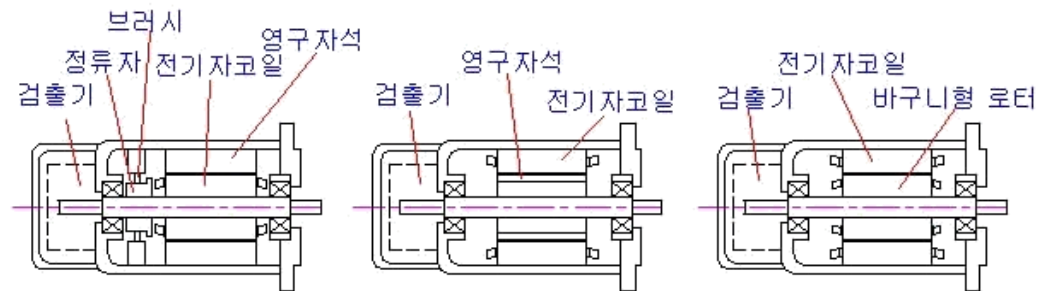
DC Motor



정류자



스텝 모터 (Step Motor)



(a)DC 서보모터 (b)SM형 AC 서보모터 (c)IM형 AC 서보모터

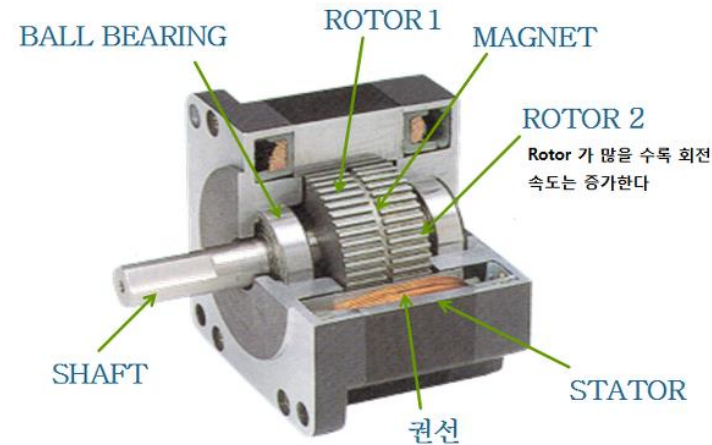
서보 모터 (Servo Motor)

- RC카 등의 무인 자동차 제작에 유용하게 사용할 수 있음 - 바퀴 1~2개 장착



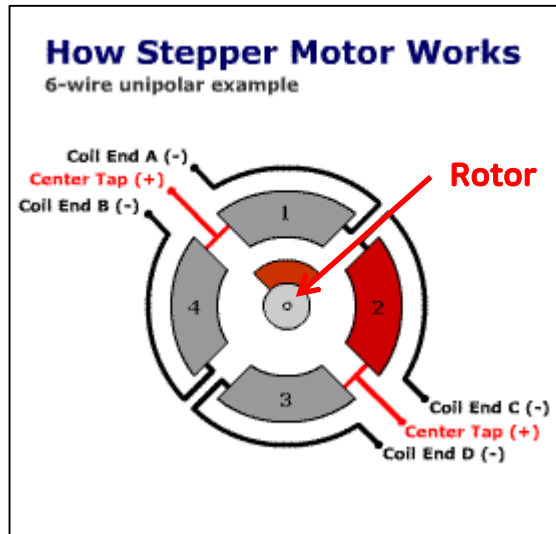
Step 모터

- 1920년 영국 군함에서 어뢰, 포신의 방향 조정에 사용 → Step by step 모터
- 기계적 구조나 전자회로가 간단하고 디지털 제어에 적합한 모터
- 긴 거리의 위치제어는 서보 모터가 유리하지만 짧은 거리의 위치 제어에는 스텝 모터가 빠른 응답
- 각도 제어 및 속도 제어가 간단 → 입력되는 펄스 신호에 따라 1 step씩 회전하는 모터
- 명칭 : 스텝퍼 모터, 스텝 (Step) 모터, 스텝핑 (Stepping) 모터, 기어드 모터 등

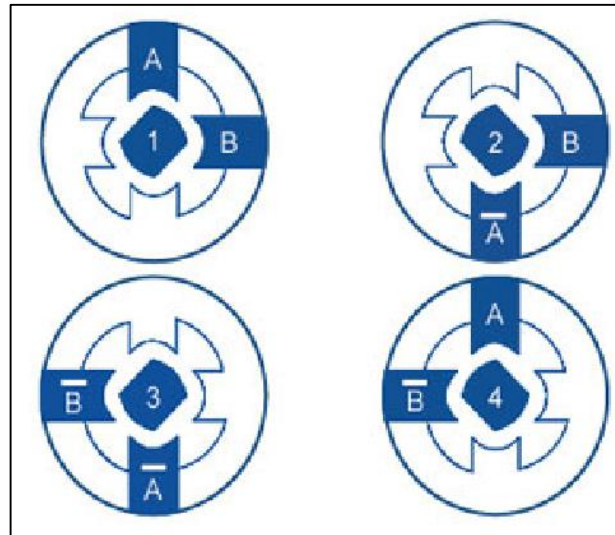


Step 모터 동작 방식

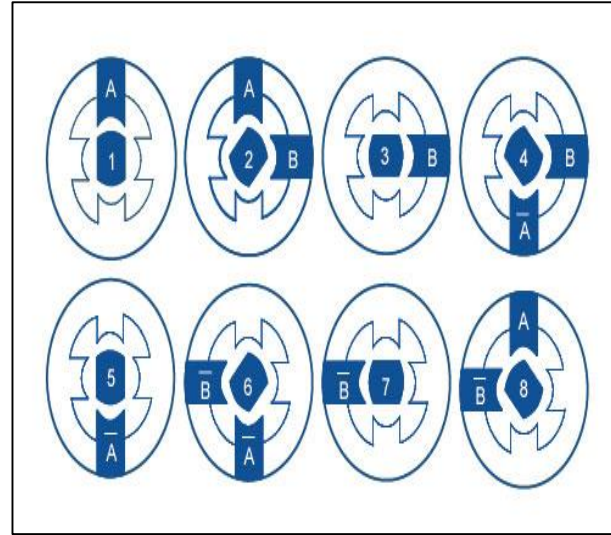
- (1) **싱글 코어 여자 방식** : 한 스텝에 하나의 코일에만 전류를 흘려 줌 → 4개의 Step
- (2) **Full Step** : 한 스텝에 2개의 인접한 코일에 전류를 흘려 줌 → 4개의 Step → 구동 토크가 더 큼
- (3) **Half Step** : Full Step을 쪼개서 한 스텝당 45도 씩 움직이게 하는 방법 → 8개의 Step → 정밀작업
- (4) **Micro Step** : 코일에 흐르는 전류의 양을 제어하여 회전자가 부드럽게 움직이게 하고 정확한 위치제어가 가능하게 함 → 가장 많이 사용



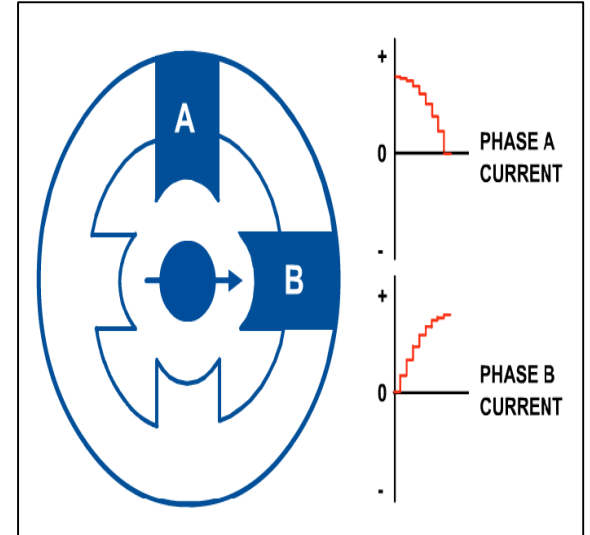
싱글 코어 여자 방식



Full Step



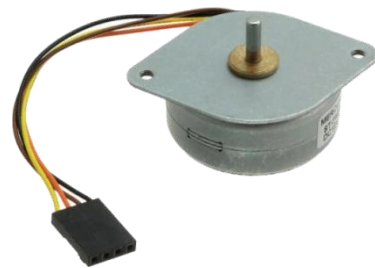
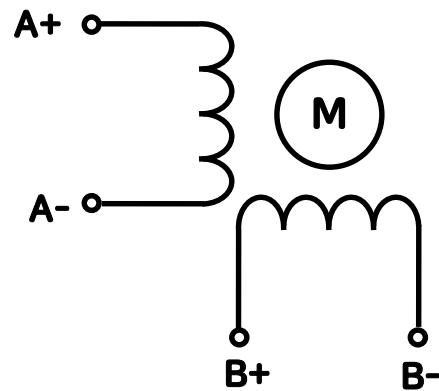
Half Step



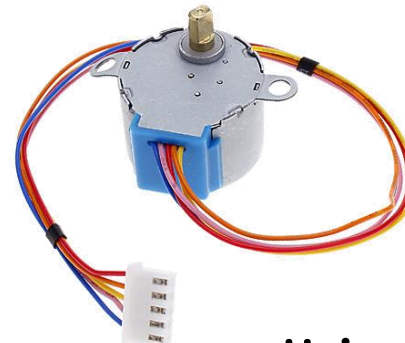
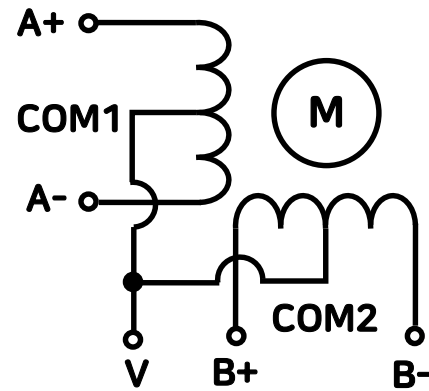
Micro Step

Step 모터 구조

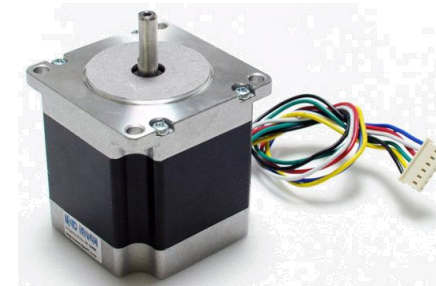
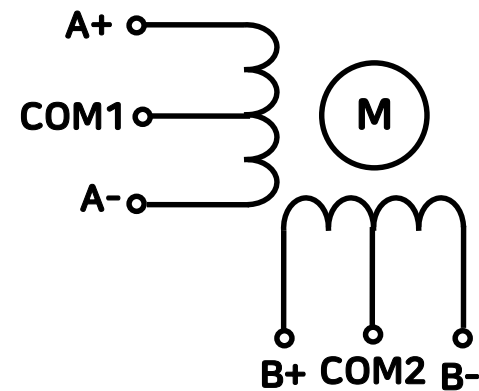
- 일반적으로 4선, 5선, 6선의 리드선을 가진 모터가 있음
- 4선 : **Bipolar** → 전류가 한쪽으로만 흐름 (고속용, 전력 비효율)
- 5선 / 6선 : **Unipolar** → 전류의 흐름이 변경되면서 흐름 (저속용, 전력 효율 좋음)



Bipolar 형태

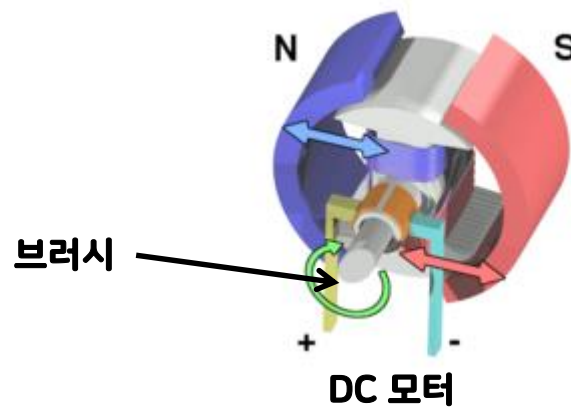


Unipolar 형태



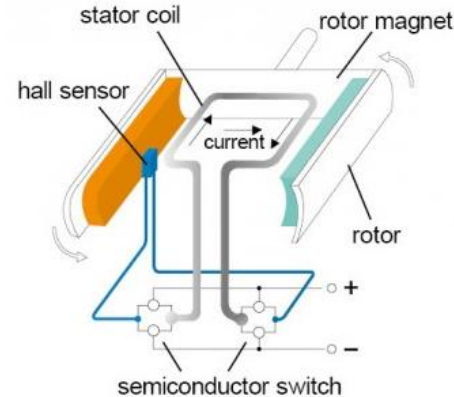
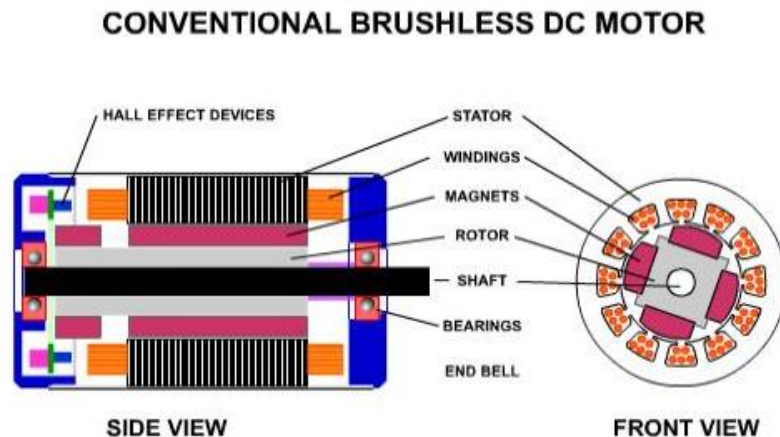
BLDC (Brushless DC) 모터

- DC 모터 (브러시드 모터) 는 모터축에 코일이 감겨있고 그 코일에 전류를 흘리기 위해서 브러쉬와 같은 절연도체를 이용해서 전력을 전달
- 따라서, 브러쉬와 모터 축에 연결된 전력 전달체 간에 마찰에 의해 불꽃이 발생(화재 위험)하거나 마모에 의해 접촉 불량 등으로 수명이 짧아지는 문제가 생김
- 또한, DC 모터의 경우 DC 전류로 속도를 제어하며 가격도 저렴하지만 소음이 큼
- 드론에 브러시드 모터를 사용한다면 ESC(변속기)가 필요없으며 전원만 연결하면 쉽게 회전한다는 장점을 가짐.



BLDC (Brushless DC) 모터

- BLDC 모터는 DC 모터의 문제를 해결하기 위해 브러시를 제거한 구조
- 모터 축에는 자석이 있고 모터 케이스 내부벽면에 코일이 있는 형태로 DC 모터와는 반대 구조로 모터가 회전하기 위한 전력 공급이 회전하지 않는 모터 내부 벽에 부착된 코일에 공급되므로 브러시가 필요 없어짐.
- BLDC 모터의 경우 동작 원리가 DC 모터와 달라, 추가 컨트롤러를 이용하여 제어가 필요
- 가격이 비싸나 넓은 속도 범위를 가지고 있고(즉, 고속에서 사용가능) 수명이 영구적
- 쿼드콥터에서 BLDC 모터를 사용하는 이유는 속도와 컨트롤의 정확성 때문

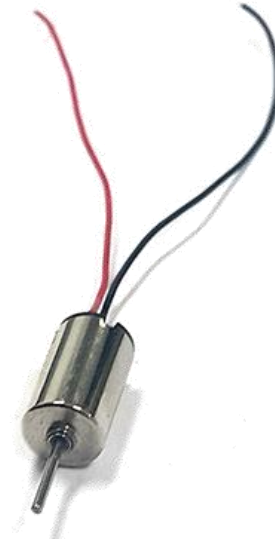
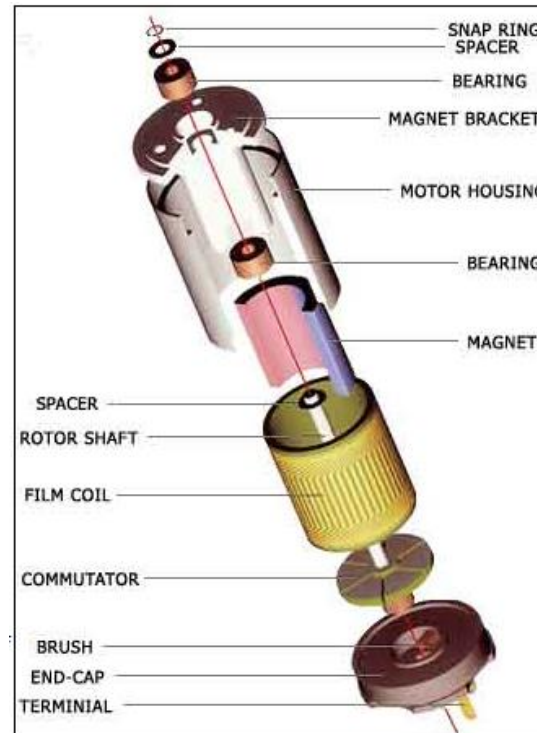


BLDC (Brushless DC) 모터

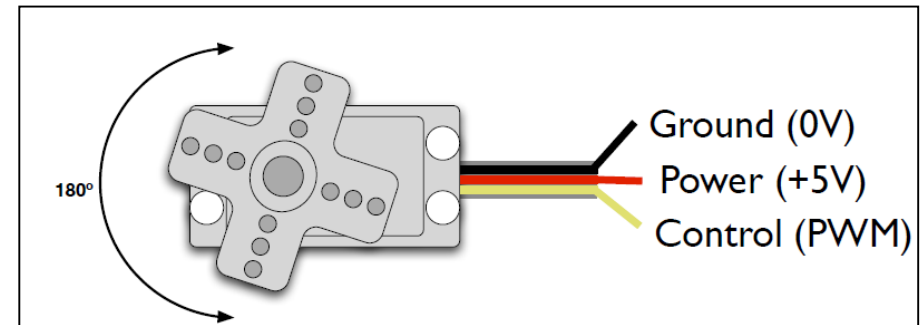
- BLDC 모터는 전원의 방향을 바꿔주는 역할을 하는 브러시가 없는 대신 BLDC용 모터 컨트롤러에서 전원의 방향을 변경 → **ESC (Electric Circuit Controller)**
- 주로 8개 선으로 구성 : 모터 연결선 3개, 전원선 2개, 신호선 3개 (1개의 선은 미사용)



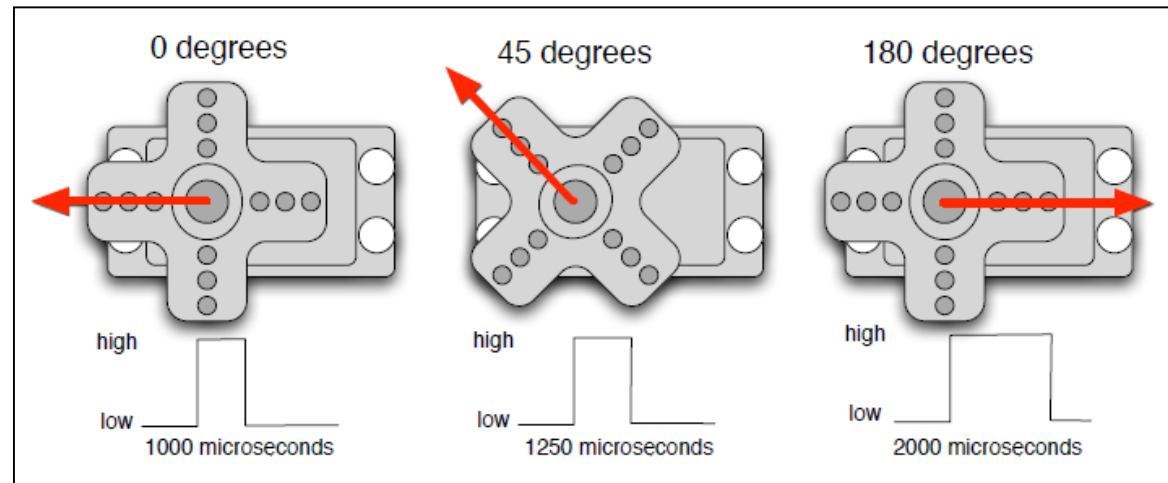
- 코일을 감는 철심을 제거하고 코일만 컵 모양으로 로터를 구성하고 그 속에 영구 자석을 넣은 모터
- 관성이 적음, 제어성이 우수, 기계적 시상수가 작음, 인덕턴스가 작고 정류 때 불꽃이 작음, 철손이 없고 출력효율이 좋음, 고응답성, 저전류, 수명이 김, 고효율, 고출력
- 높은 정밀도가 요구되는 기계나 장치에 이용



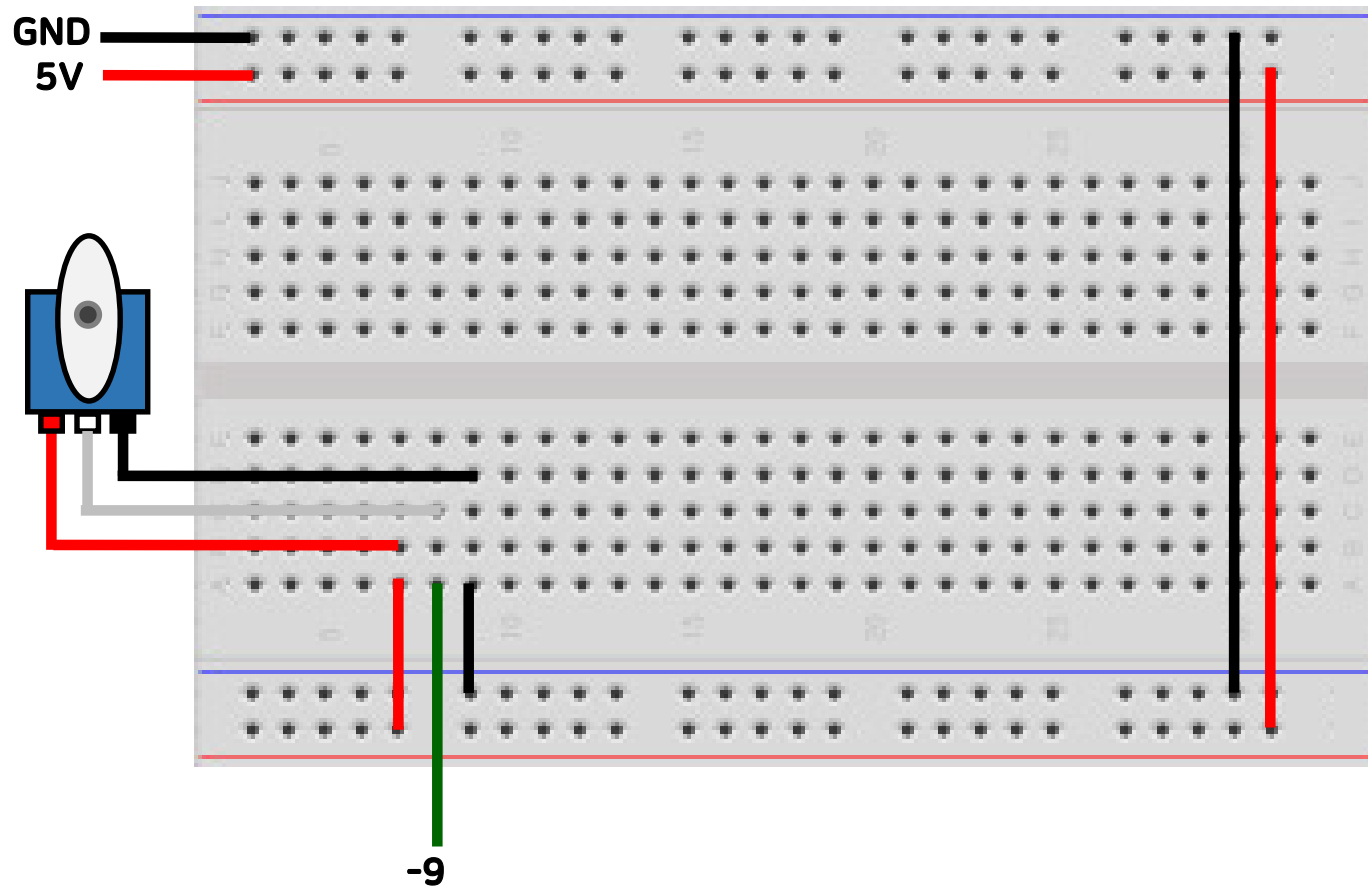
- 서보모터 : 각으로 모터의 회전을 제어할 수 있는 모터
- 서보모터의 범위 : 0도-179도
- 3개의 핀으로 구성 : 전원 핀, GND 핀, 제어 핀



- 일반적인 서보 모터 는 500 ~ 2500ms의 펄스 범위를 가짐 → 모터마다 약간씩 다를 수 있으므로 확인하고 사용 → **잘못 설정된 경우 떨림이나 소리가 날 수 있음 (헌팅 현상).**
- 1ms = 1000 μ s = 0도
- 2ms = 2000 μ s = 180도
- 서보 모터 밸런싱 하기
 - 0도에서 떨림 현상이 생기면 최소값을 높여주면서 테스트
 - 180도에서 떨림 현상이 생기면 최대값을 낮추면서 테스트



Servo 모터 제어



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

ServoMotor 사용을 하기 위한 라이브러리

```
Servo myservo;
```

ServoMotor 사용을 위한 변수

```
myservo.attach(9);
```

ServoMotor 연결된 Pin 번호

```
myservo.write(angle);
```

ServoMotor 각(0-180)도 조절

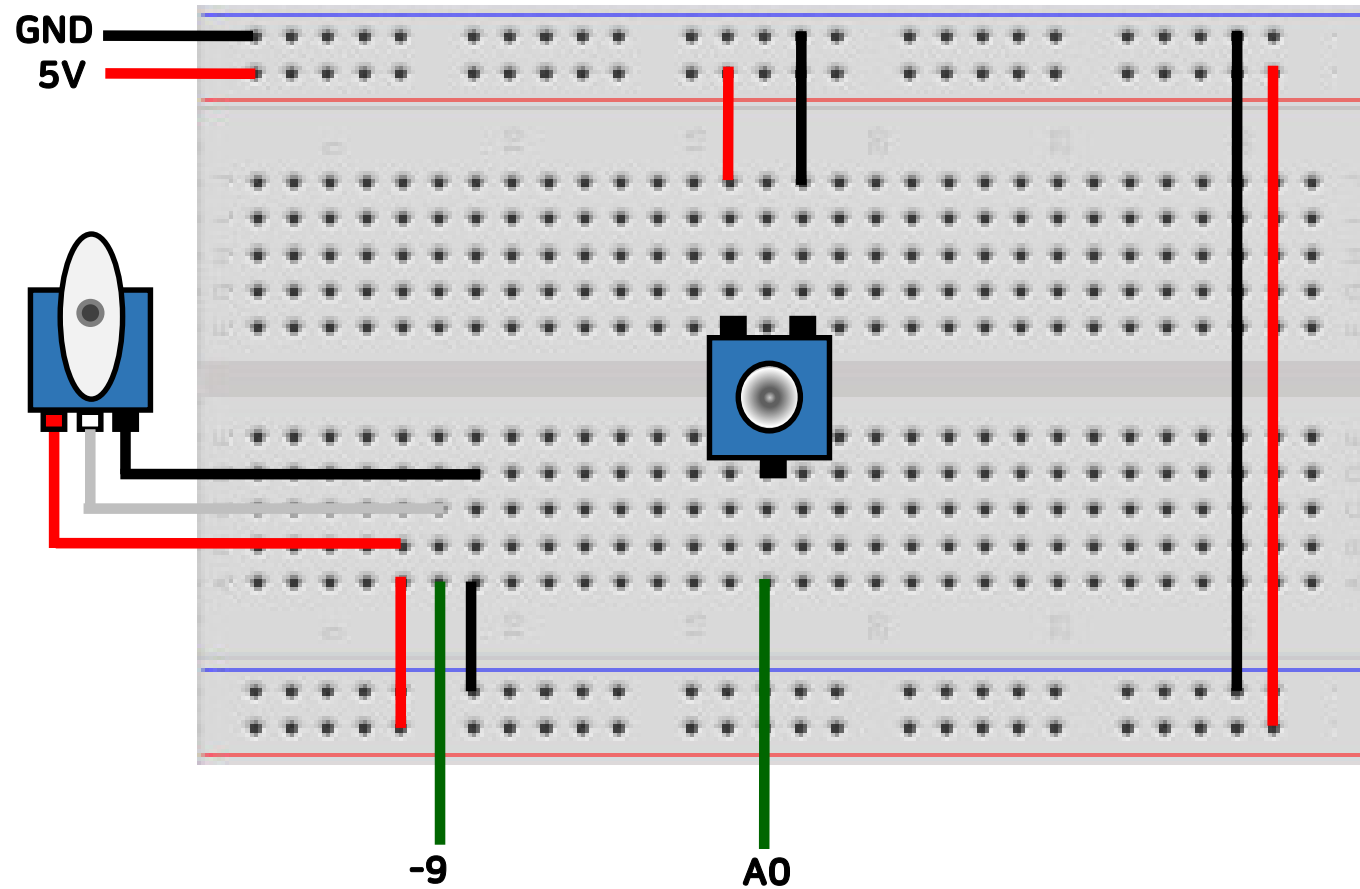
- **read()** : 서보 모터의 현재 각도를 반환
- **detach()** : 서보모터의 연결을 해제 → 9번, 10번 핀에 analogWrite() 사용 가능

- 헌팅 현상 제거 - detach() 함수를 사용하여 더 이상 PWM 신호를 보내기 않게 함

1	#include <Servo.h>	1	Servo 라이브러리를 등록
2		2	
3	Servo myServo;	3	서보 객체 선언
4		4	
5	void setup() { }	5	
6		6	
7	void loop() {	7	
8	myServo.attach(9);	8	9번 핀에 연결된 서보를 서보 객체에 연결
9	myServo.write(180);	9	서보 모터를 180도로 돌린다
10	delay(500);	10	서보 모터 회전 대기 시간
11	myServo.detach();	11	서보 모터의 연결을 해제한다
12	delay(3000);	12	
13	myServo.attach(9);	13	
14	myServo.write(0);	14	
15	delay(500);	15	
16	myServo.detach();	16	
17	delay(3000);	17	
18	}	18	

가변저항으로 서보모터 제어

- 가변저항의 값을 조절함에 따라 서보 모터의 범위인 0-180도 까지 회전시킴



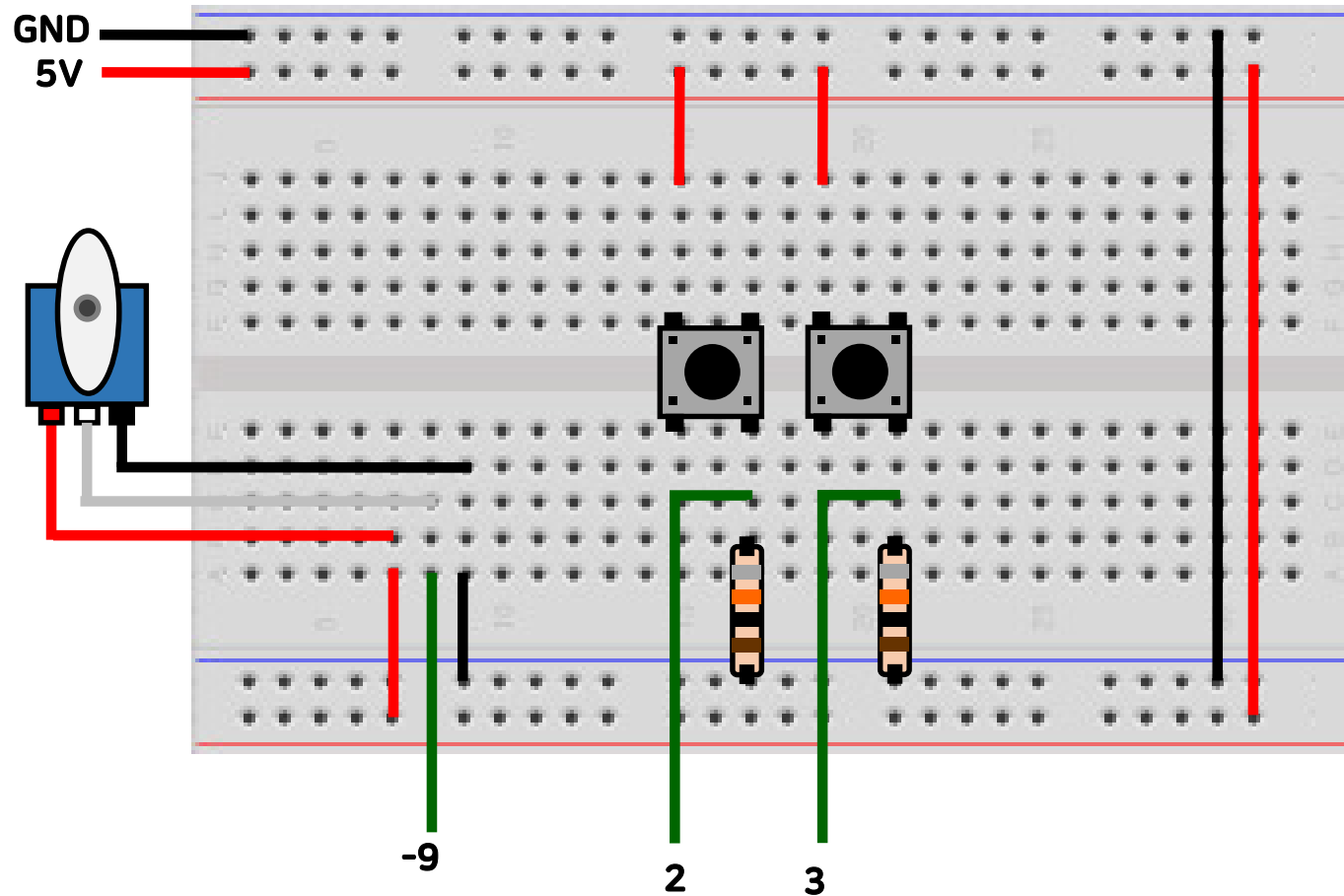
1	#include <Servo.h>	1	
2		2	
3	Servo myServo;	3	
4		4	
5	int const potPin = A0;	5	
6	int potVal;	6	가변저항의 값
7	int angle;	7	가변저항 값을 각도로 변환한 값
8		8	
9	void setup() {	9	
10	myServo.attach(9);	10	9번 핀에 연결된 서보를 서보 객체에 연결
11	Serial.begin(9600);	11	
12	Serial.println(myServo.read());	12	현재 서보 모터의 각도를 출력한다
13	}	13	
14		14	
15	void loop() {	15	
16	potVal = analogRead(potPin);	16	가변 저항 값을 읽는다
17	Serial.print("potVal : ");Serial.print(potVal);	17	
18		18	
19	angle = map(potVal, 0, 1023, 0, 179);	19	가변저항 값을 0-179 범위로 변환
20	Serial.print(", angle = ");Serial.println(angle);	20	
21		21	
22	myServo.write(angle);	22	각도에 따라 서보 모터를 회전시킨다
23	delay(15);	23	
24	}	24	



Quest

스위치 S1을 누르면 서보 모터를 90도 돌리고 스위치 S2를 누르면 0도로 다시 돌아오는 회로를 제작하시오.

PushButton으로 서보모터 제어



PushButton으로 서보모터 제어

```
1 #include <Servo.h>
2
3 Servo myServo;
4
5 int sw1 = 2;
6 int sw2 = 3;
7 int state = 0;
8
9 void setup() {
10   myServo.attach(9);
11   pinMode(sw1, INPUT);
12   pinMode(sw2, INPUT);
13   myServo.write(state);
14 }
15
16 void loop() {
17   if(digitalRead(sw1) == HIGH) {
18     state = 90;
19   }
20   if(digitalRead(sw2) == HIGH) {
21     state = 0;
22   }
23   myServo.write(state);
24 }
```

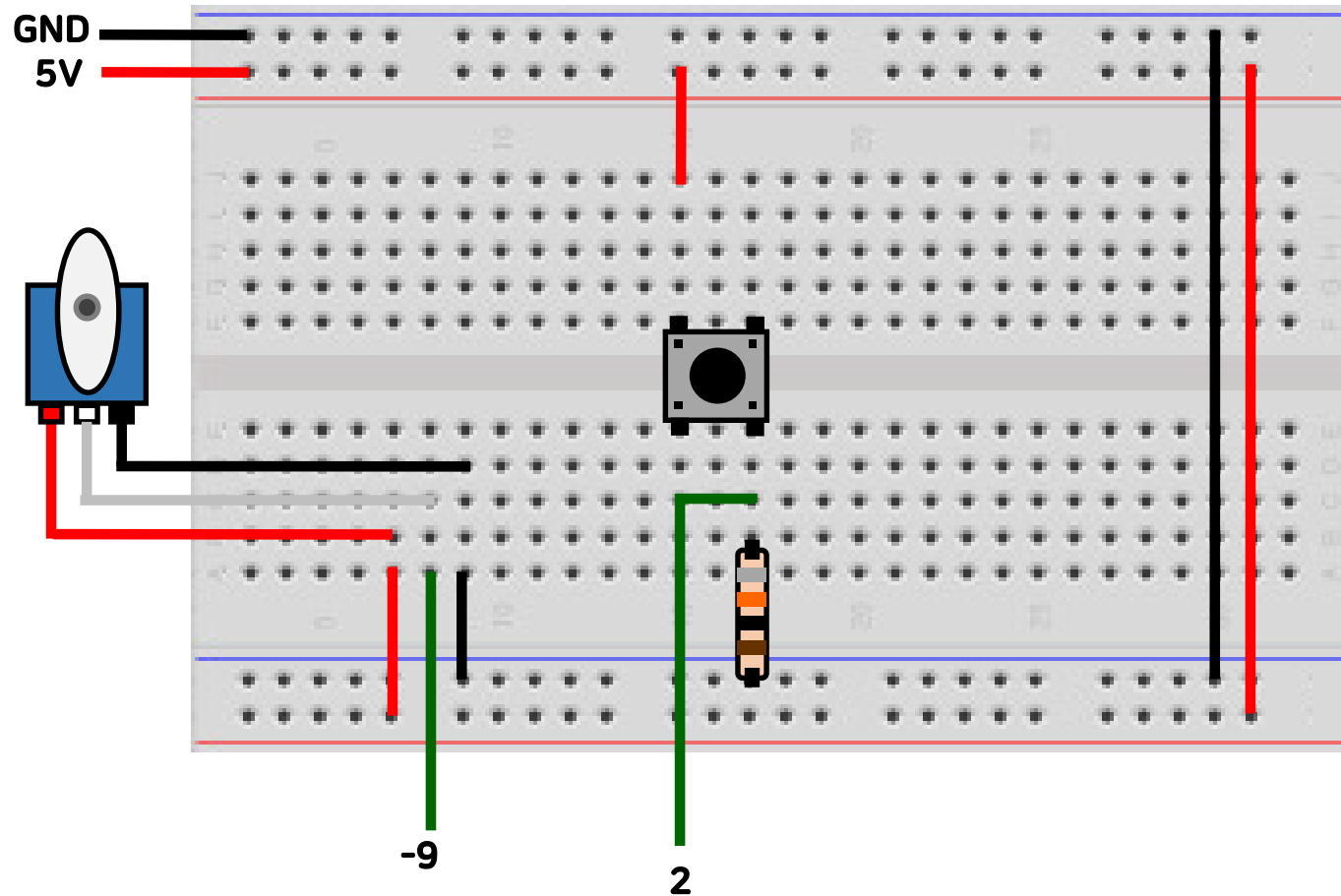
```
1
2
3
4
5
6
7 모터의 각도
8
9
10
11
12
13
14
15
16
17 스위치1이 LOW이면 90도로 설정
18
19
20 스위치2가 LOW이면 0도로 설정
21
22
23
24
```



Quest

스위치를 누르면 서보모터가 0-180도 사이를 왕복하고 스위치를 누르지 않으면 그 자리에 정지하는 회로와 스케치를 작성하시오.

PushButton으로 서보모터 제어



PushButton으로 서보모터 제어

```
1 #include <Servo.h>
2
3 Servo myServo;
4
5 int sw = 2;
6 int state = 0;
7 int direction = 1;
8
9 void setup() {
10   myServo.attach(9);
11   pinMode(sw, INPUT);
12   myServo.write(state);
13 }
14
15 void loop() {
16   if(digitalRead(sw) == HIGH) {
17     if(state <= 0) direction = 1;
18     if(state >= 179) direction = 0;
19
20     if(direction == 1) state += 10;
21     if(direction == 0) state -= 10;
22
23     myServo.write(state);
24   }
25   delay(100);
26 }
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 스위치가 LOW이면 90도로 설정
17 0보다 작으면 방향을 오른쪽으로 설정
18 179보다 크면 방향을 왼쪽으로 설정
19
20 방향이 오른쪽이면 10도씩 더함
21 방향이 왼쪽이면 10도씩 뺌
22
23
24
25
26
```

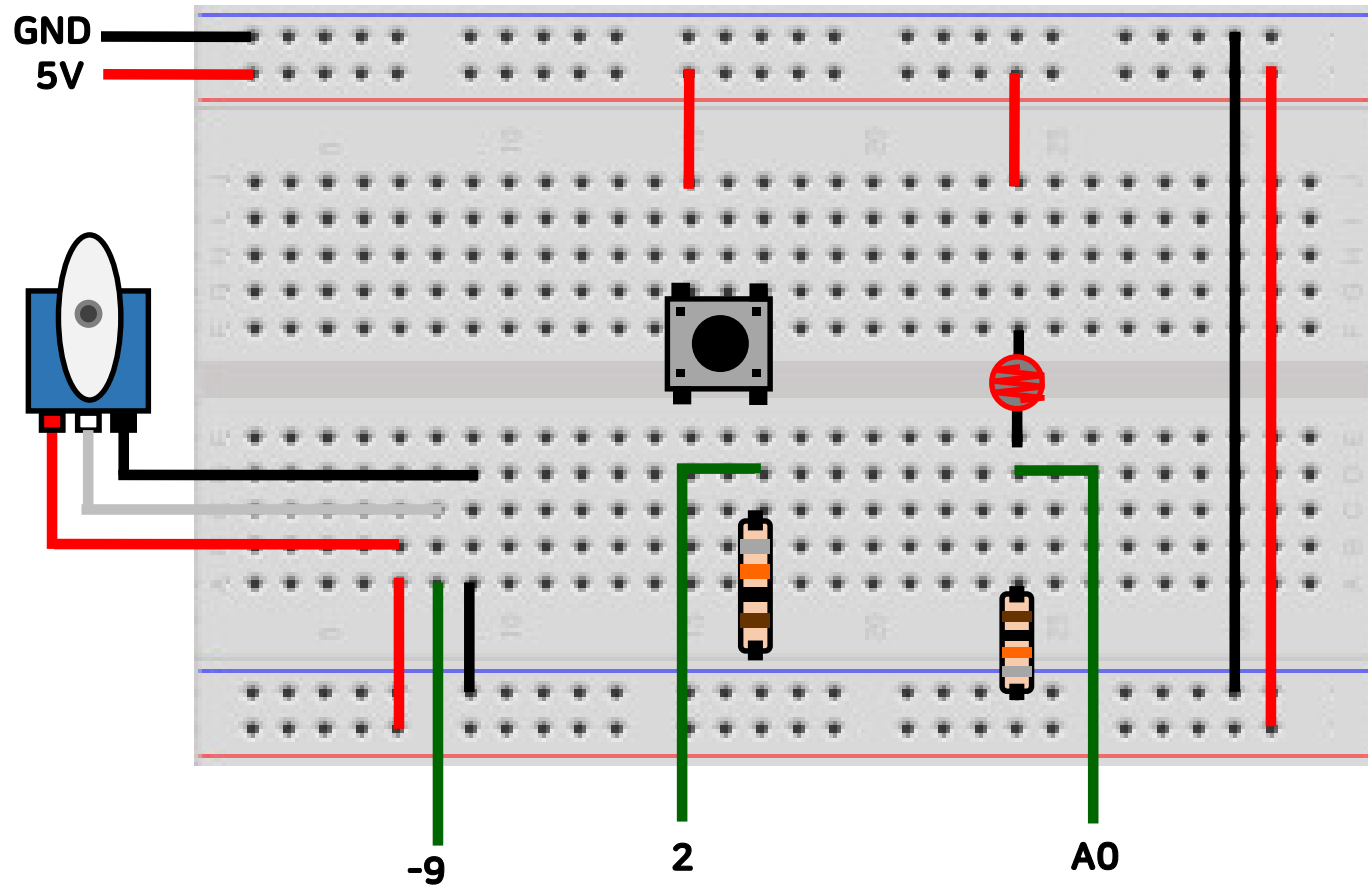


Quest

조도 센서를 이용하여 값에 따라 밝기에 따라 서보모터를 돌리도록 해보자 (빛에 따라 문이나 커튼 제어)

- 버튼을 누르면 서보모터가 초기 위치(0도)로 돌아가도록 한다.

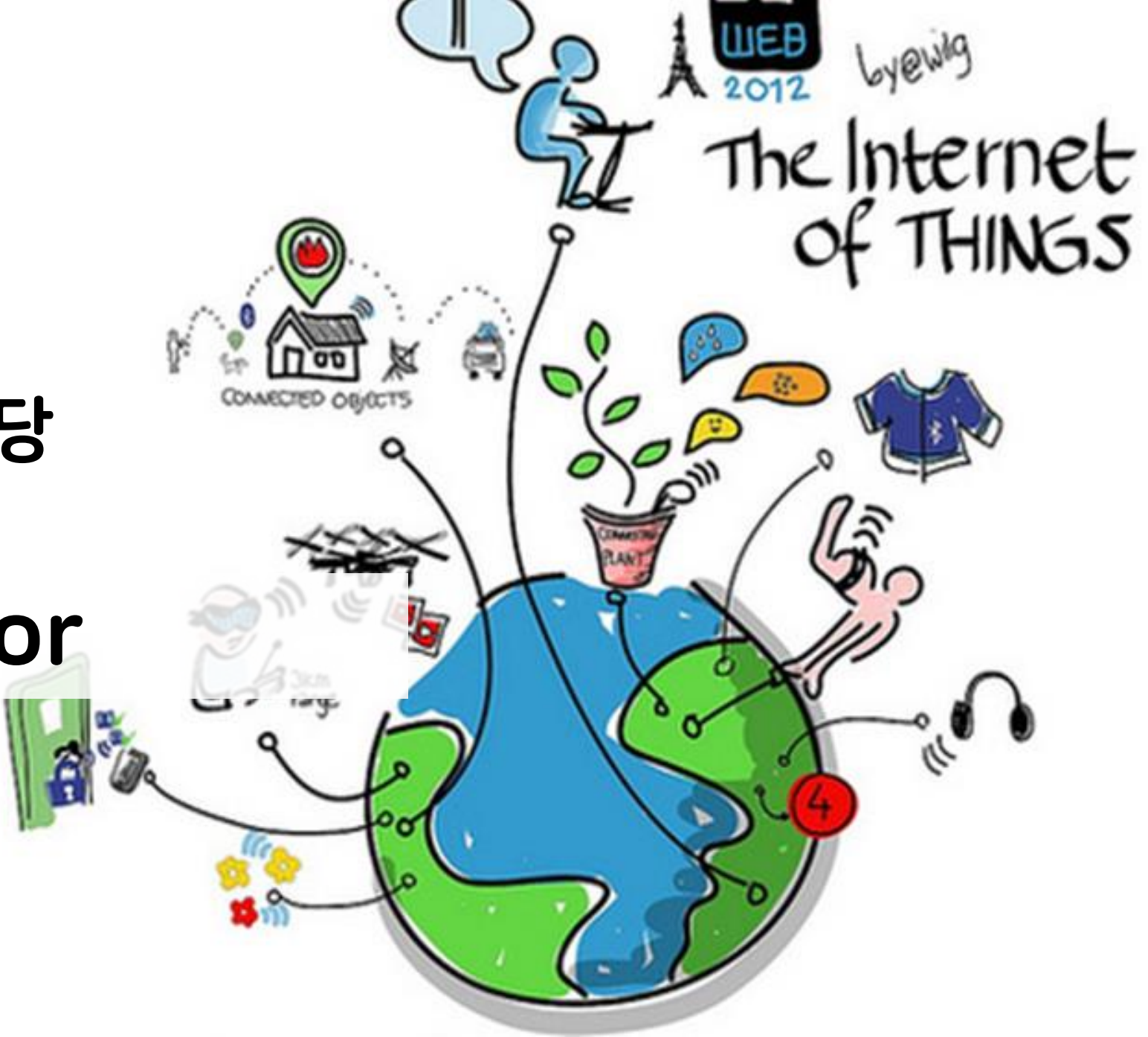
조도센서로 서보모터 제어





셋째 마당

DC Motor



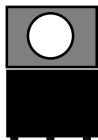
- DC 모터에 연결된 다이오드는 역방향 전류를 방지하기 위한 것



1N4007

역전압을 빠르게 견딜만한 용량 필요

정류 다이오드

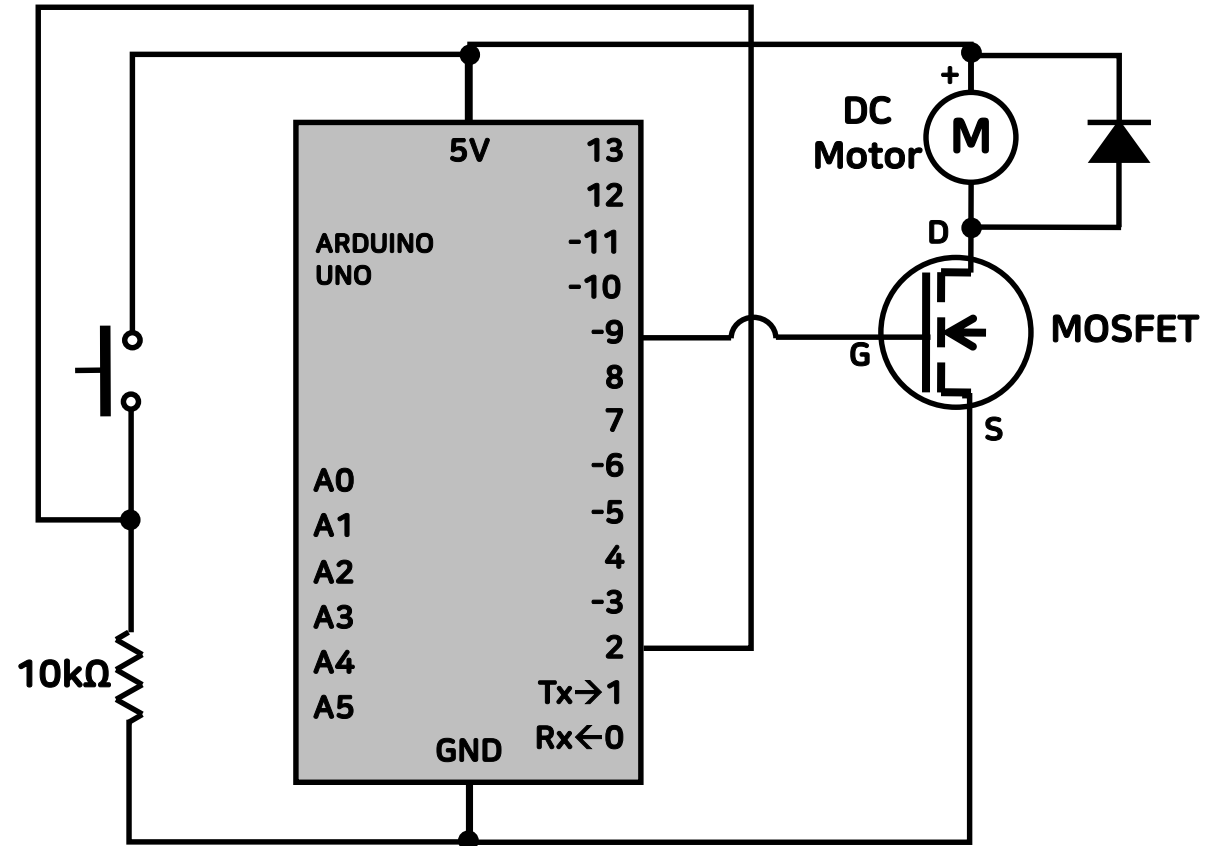


Gate

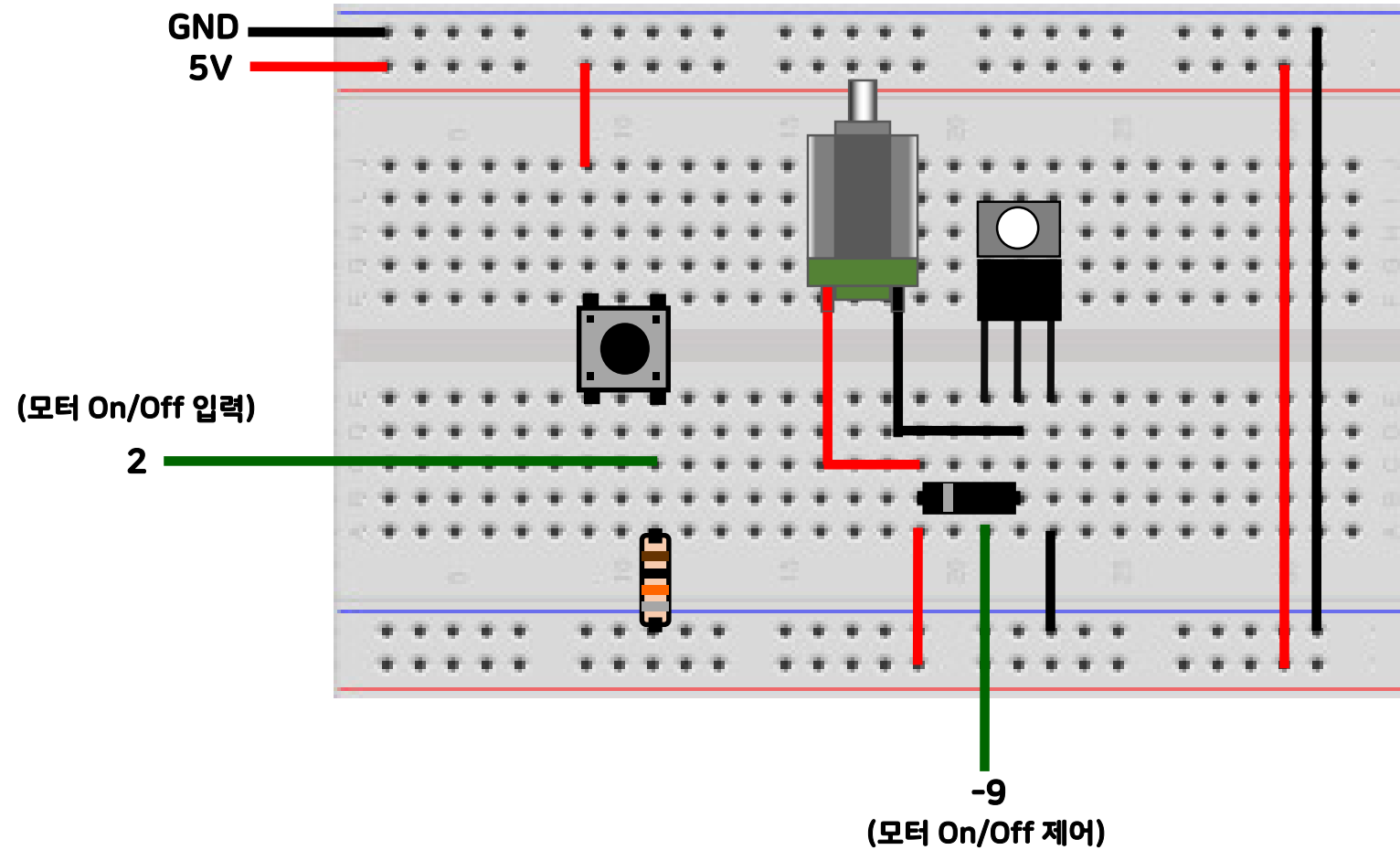
Drain

Source

MOSFET의 핀 구조



- 버튼을 누르면 모터가 돌아가고 떼면 정지



1	const int switchPin = 2;	1	푸쉬 버튼이 연결된 핀
2	const int motorPin = 9;	2	모터가 연결된 핀
3	int switchState = 0;	3	스위치의 상태
4		4	
5	void setup() {	5	
6	pinMode(motorPin, OUTPUT);	6	
7	pinMode(switchPin, INPUT);	7	
8	}	8	
9		9	
10	void loop() {	10	
11	switchState = digitalRead(switchPin);	11	
12		12	
13	if(switchState == HIGH) {	13	
14	digitalWrite(motorPin, HIGH);	14	
15	} else {	15	
16	digitalWrite(motorPin, LOW);	16	
17	}	17	
18	}	18	

Quest

스위치를 1번 누르면 모터가 돌고 2번 누르면 멈추도록 하세요.

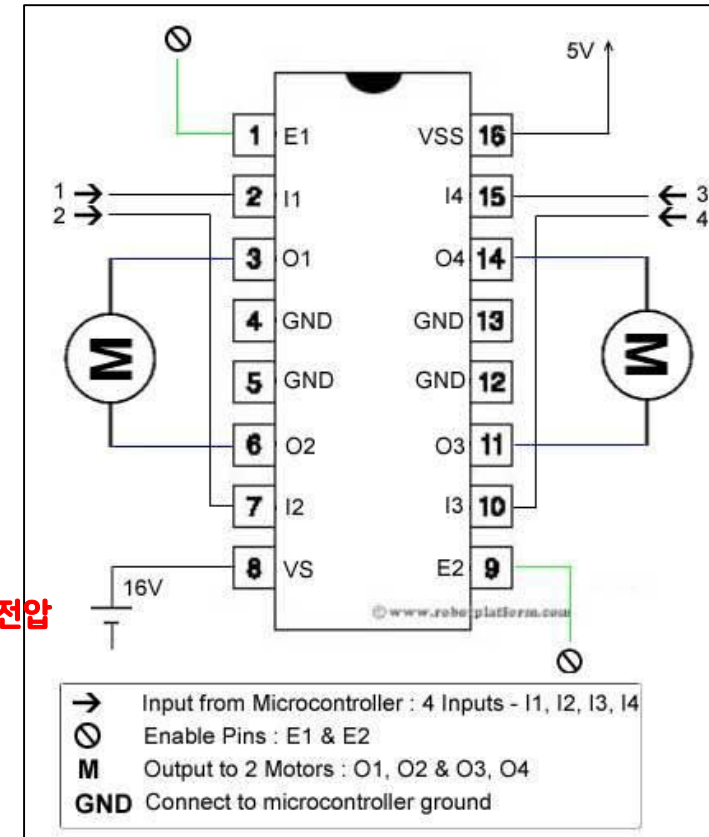
Quest

스위치를 1번 누르면 모터가 천천히 돌고 2번 누르면 빠르게 돌고, 3번 누르면 멈추도록 하세요.

H-브릿지를 이용한 DC모터 제어

- DC모터를 양방향으로 제어하기 위해서는 TR, FET, 저항, 콘덴서, 옵토커플러 등이 필요 -> 단일 칩으로 된 H-Bridge를 사용
- DC 모터들도 용량이 꽤 크기 때문에, 반드시 적절한 용량의 칩을 선택해야 함
- L293D : 초소형 모터의 작동 최대전류가 600mA 이하
- L293 : 작동 최대전류가 600mA ~ 1A
- L298 : 작동 최대전류가 1A ~ 2A
- L6203 : 작동 최대전류가 2A ~ 4A (별도 부품, 저항 및 콘덴서 필요)
- H-Bridge : 작동 최대전류가 4A 이상 (전력용 FET 사용)

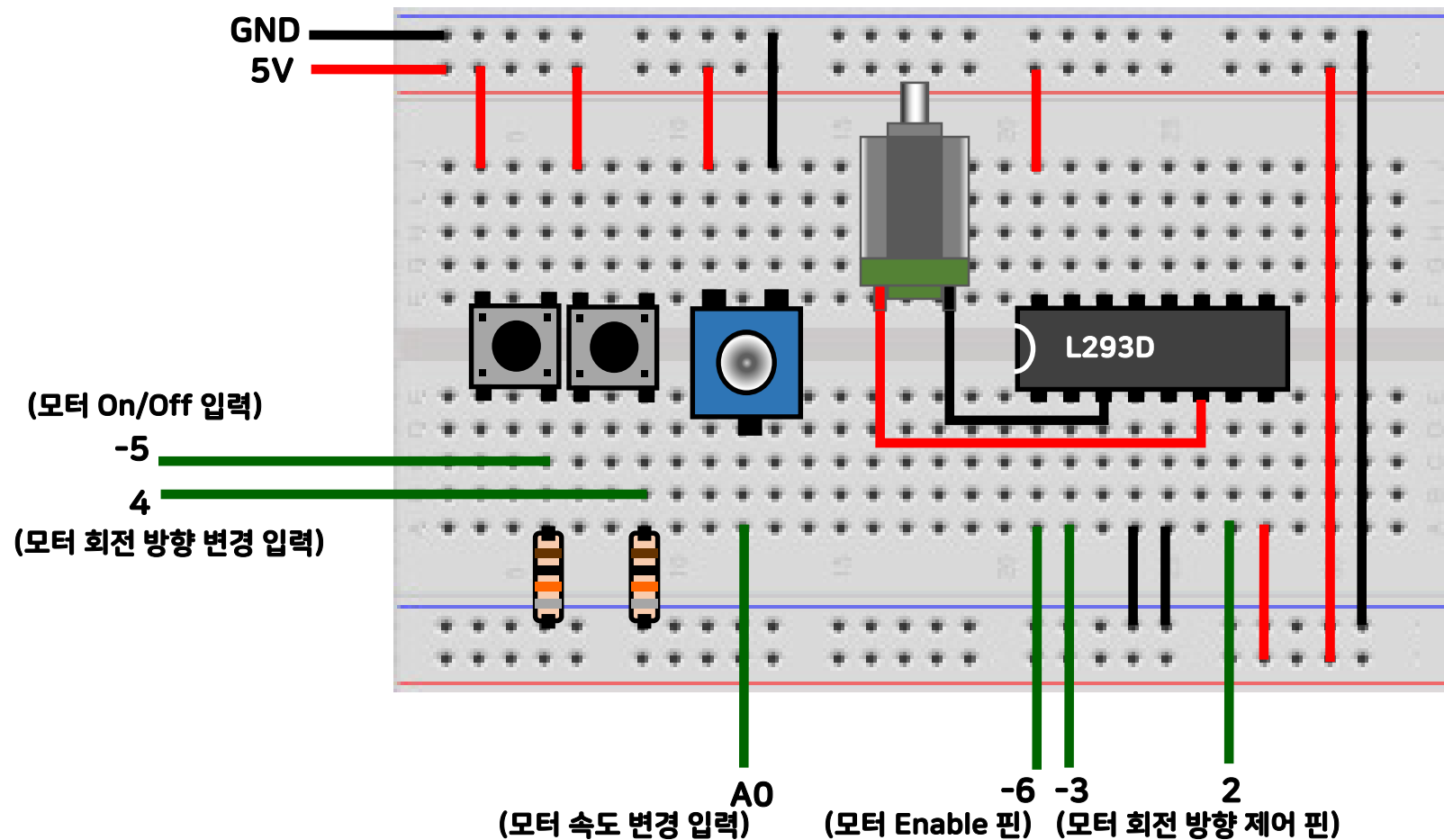
모터 동작 전압



L293D의 핀 구조

H-브릿지를 이용한 DC모터 제어

- 핀 2가 HIGH이고 핀 3이 LOW → 한 방향으로 회전
- 핀 2가 LOW이고 핀 3인 HIGH → 반대 방향으로 회전
- 핀 2와 3이 동시에 HIGH이거나 LOW → 정지



H-브릿지를 이용한 DC모터 제어

- On/Off 스위치 (5번에 연결된 스위치)를 누르면 모터가 회전 시작하고 On/Off 스위치를 다시 누르면 모터가 정지
- 방향 스위치 (4번에 연결된 스위치)를 누르면 모터의 회전 방향이 반대로 변경
- 가변저항을 돌리면 모터의 속도가 변경됨

1	const int controlPin1 = 2;	1	모터 제어 핀 (H 브릿지의 7번과 연결)
2	const int controlPin2 = 3;	2	모터 제어 핀 (H 브릿지의 2번과 연결)
3	const int directionSwitchPin = 4;	3	모터 방향 제어 스위치가 연결된 핀
4	const int onOffSwitchStateSwitchPin = 5;	4	모터 On/Off 스위치가 연결된 핀
5	const int enablePin = 6;	5	H 브릿지의 1번과 연결
6	const int potPin = A0;	6	모터의 속도 제어 입력 (가변저항 크기)
7		7	
8	int onOffSwitchState = 0;	8	모터 On/Off 스위치의 상태
9	int previousOnOffSwitchState = 0;	9	이전 모터 On/Off 스위치의 상태
10	int directionSwitchState = 0;	10	모터 방향 제어 스위치의 상태
11	int previousDirectionSwitchState = 0;	11	이전 모터 방향 제어 스위치의 상태
12		12	
13	int motorEnabled = 0;	13	모터의 동작 여부
14	int motorSpeed = 0;	14	모터의 속도
15	int motorDirection = 1;	15	모터의 회전 방향

H-브릿지를 이용한 DC모터 제어

16	void setup() {	16	
17	pinMode(directionSwitchPin, INPUT);	17	모터 방향 제어 스위치의 핀을 입력으로
18	pinMode(onOffSwitchStateSwitchPin, INPUT);	18	모터 On/Off 스위치의 핀을 입력으로
19	pinMode(controlPin1, OUTPUT);	19	모터 제어핀을 출력으로 설정
20	pinMode(controlPin2, OUTPUT);	20	
21	pinMode(enablePin, OUTPUT);	21	모터 동작 핀을 출력으로 설정
22		22	
23	digitalWrite(enablePin, LOW);	23	초기 모터는 정지 상태로 설정
24	}	24	
25		25	
26	void loop() {	26	
27	onOffSwitchState = digitalRead(onOffSwitchStateSwitchPin);	27	모터 On/Off 스위치 상태를 읽는다
28	delay(1);	28	
29	directionSwitchState = digitalRead(directionSwitchPin);	29	모터 방향 스위치 상태를 읽는다
30	motorSpeed = analogRead(potPin) / 4;	30	가변 저항 값 / 4로 모터의 속도를 결정
31		31	
32	if(onOffSwitchState != previousOnOffSwitchState) {	32	모터 On/Off 스위치를 누르면
33	if(onOffSwitchState == HIGH) {	33	모터 On/Off 스위치 상태가 HIGH이면
34	motorEnabled = !motorEnabled;	34	모터의 동작 상태를 반대로 변경
35	}	35	
36	}	36	

H-브릿지를 이용한 DC모터 제어

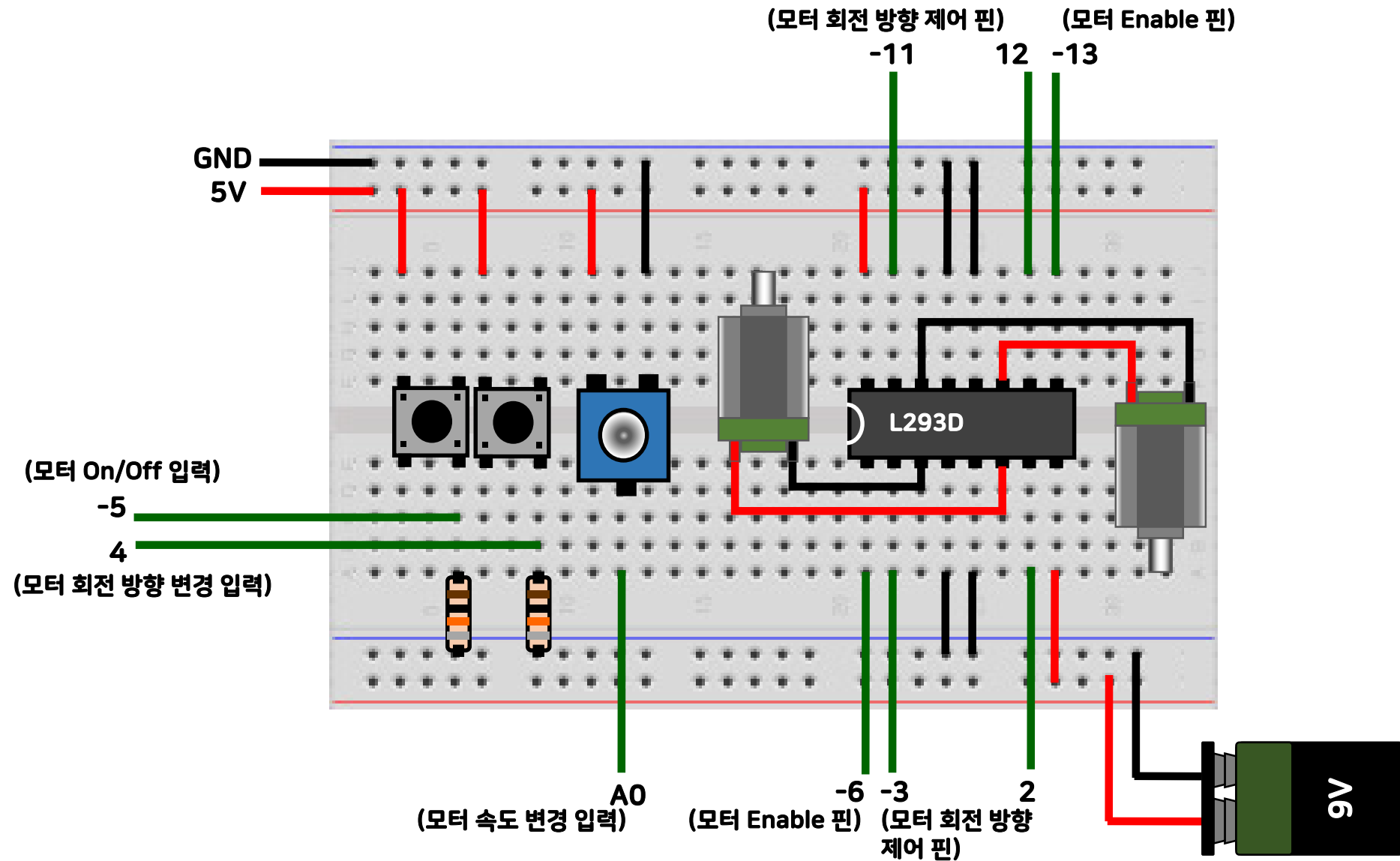
37	if(directionSwitchState != previousDirectionSwitchState) {	43	모터 방향 스위치를 누르면
38	if(directionSwitchState == HIGH) {	44	모터 방향 스위치 상태가 HIGH 이면
39	motorDirection = !motorDirection;	45	모터의 방향을 반대로 변경
40	}	46	
41	}	47	
42		48	
43	if(motorDirection == 1) {	49	모터의 방향이 1이면
44	digitalWrite(controlPin1, HIGH);	50	
45	digitalWrite(controlPin2, LOW);	51	
46	} else {	52	
47	digitalWrite(controlPin1, LOW);	53	
48	digitalWrite(controlPin2, HIGH);	54	
49	}	55	
50		56	
51	if(motorEnabled == 1) {	57	모터가 동작 가능하면
52	analogWrite(enablePin, motorSpeed);	58	
53	} else {	59	
54	analogWrite(enablePin, 0);		
55	}		
56			
57	previousDirectionSwitchState = directionSwitchState;		이전 모터 방향 스위치 상태 저장
58	previousOnOffSwitchState = onOffSwitchState;		이전 모터 On/Off 스위치 상태 저장
59	}		



Quest

이전 예제에서 H-브릿지 사용하여 2개의 DC 모터를 제어하라.

DC모터 제어



H-브릿지를 이용한 DC모터 제어

1	const int controlPin1 = 2;	1	
2	const int controlPin2 = 3;	2	
3	const int controlPin3 = 11;	3	2 번째 모터 제어 핀 1
4	const int controlPin4 = 12;	4	2 번째 모터 제어 핀 2
5	const int directionSwitchPin = 4;	5	
6	const int onOffSwitchStateSwitchPin = 5;	6	
7	const int enablePinA = 6;	7	2 번째 모터 enable 핀
8	const int enablePinB = 13;	8	
9	const int potPin = A0;	9	
10		10	
11	int onOffSwitchState = 0;	11	
12	int previousOnOffSwitchState = 0;	12	
13	int directionSwitchState = 0;	13	
14	int previousDirectionSwitchState = 0;	14	
15		15	
16	int motorEnabled = 0;	16	
17	int motorSpeed = 0;	17	
18	int motorDirection = 1;	18	
19		19	
20	void setup() {	20	
21	pinMode(directionSwitchPin, INPUT);	21	
22	pinMode(onOffSwitchStateSwitchPin, INPUT);	22	
23	pinMode(controlPin1, OUTPUT);	23	
24	pinMode(controlPin2, OUTPUT);	24	

H-브릿지를 이용한 DC모터 제어

```
25 pinMode(controlPin3, OUTPUT);
26 pinMode(controlPin4, OUTPUT);
27 pinMode(enablePinA, OUTPUT);
28 pinMode(enablePinB, OUTPUT);
29 digitalWrite(enablePinA, LOW);
30 digitalWrite(enablePinB, LOW);
31 }
32
33 void loop() {
34   onOffSwitchState = digitalRead(onOffSwitchStateSwitchPin);
35   delay(1);
36   directionSwitchState = digitalRead(directionSwitchPin);
37   motorSpeed = analogRead(potPin) / 4;
38
39   if(onOffSwitchState != previousOnOffSwitchState) {
40     if(onOffSwitchState == HIGH) {
41       motorEnabled = !motorEnabled;
42     }
43   }
44
45   if(directionSwitchState != previousDirectionSwitchState) {
46     if(directionSwitchState == HIGH) {
47       motorDirection = !motorDirection;
48     }
49   }
```

25 2번째 모터 제어핀 출력 설정

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

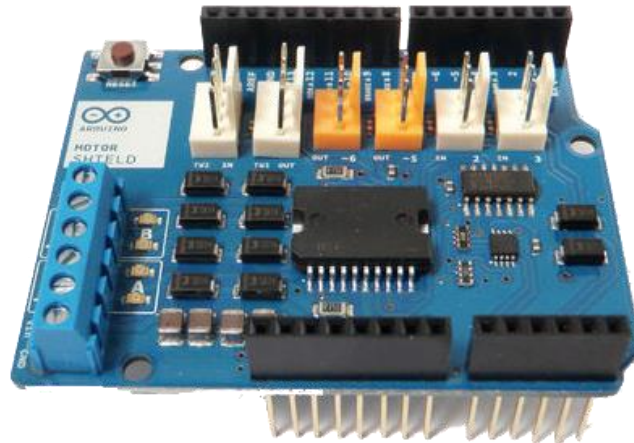
49

2번째 모터 활성화 핀 설정

H-브릿지를 이용한 DC모터 제어

50	if(motorDirection == 1) {	50	
51	digitalWrite(controlPin1, HIGH);	51	
52	digitalWrite(controlPin2, LOW);	52	
53	digitalWrite(controlPin3, HIGH);	53	2번째 모터 방향 설정
54	digitalWrite(controlPin4, LOW);	54	
55	} else {	55	
56	digitalWrite(controlPin1, LOW);	56	
57	digitalWrite(controlPin2, HIGH);	57	
58	digitalWrite(controlPin3, LOW);	58	2번째 모터 방향 설정
59	digitalWrite(controlPin4, HIGH);	59	
60	}	60	
61		61	
62	if(motorEnabled == 1) {	62	
63	analogWrite(enablePinA, motorSpeed);	63	2번째 모터 속도 설정
64	analogWrite(enablePinB, motorSpeed);	64	
65	} else {	65	
66	analogWrite(enablePinA, 0);	66	2번째 모터 정지
67	analogWrite(enablePinB, 0);	67	
68	}	68	
69		69	
70	previousDirectionSwitchState = directionSwitchState;	70	
71	previousOnOffSwitchState = onOffSwitchState;	71	
72	}	72	

- Arduino Motor Shield R3 모터 쉴드는 2개의 DC 모터나 1개의 스텝 모터를 구동할 수 있으며 2개를 연결할 수 있는 채널(A, B)을 가지고 있음.
- 모터 채널당 2A의 전류와 5-12V 전압을 제공하며 모터 제어는 PWM 방식을 사용
- L298P H 브릿지 사용



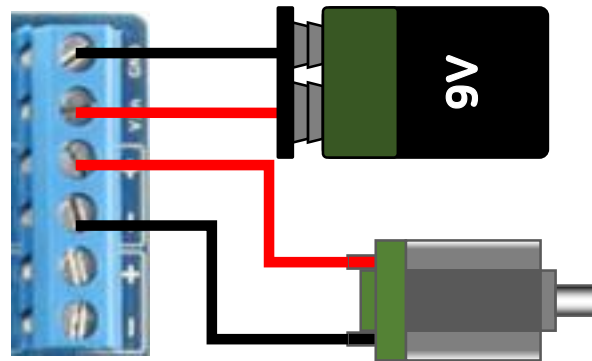
Arduino Motor Shield R3

모터쉴드를 이용한 DC모터 제어

- 모터쉴드를 사용하기 위해서는 아두이노 보드에 끼운다.



- USB 케이블, 모터 2개, 배터리를 연결한다.

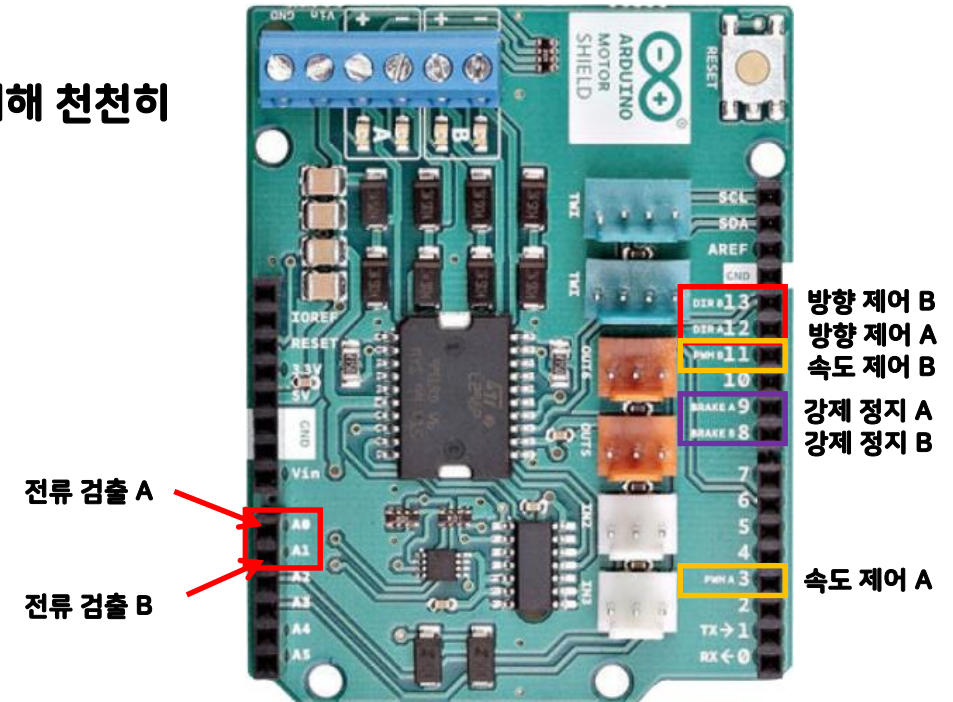


모터실드를 이용한 DC모터 제어

- 핀 연결

기능	Channel A	Channel B
방향 제어	Pin 12	Pin 13
속도 제어 (PWM)	Pin 3	Pin 11
강제 정지 (HIGH 인 경우)	Pin 9	Pin 8
전류 검출	A0	A1

- 모터 회전 시에 PWM을 0으로 하면 전압이 0이 되어 모터는 관성에 의해 천천히 정지하지만 강제 정지는 HIGH가 되는 경우 바로 정지함



- 예제 : 모터를 회전 시켜보자 (속도와 방향 변경)

1	const int DIR_A = 12, PWM_A = 3;	1	방향제어 핀, 속도 제어 핀
2		2	
3	void setup() {	3	
4	pinMode(DIR_A, OUTPUT);	4	
5	pinMode(PWM_A, OUTPUT);	5	
6	}	6	
7		7	
8	void loop() {	8	
9	digitalWrite(DIR_A, LOW);	9	LOW/HIGH 값으로 회전 방향 설정
10	analogWrite(PWM_A, 255);	10	모터를 최고속도로 회전시킨다
11	}	11	

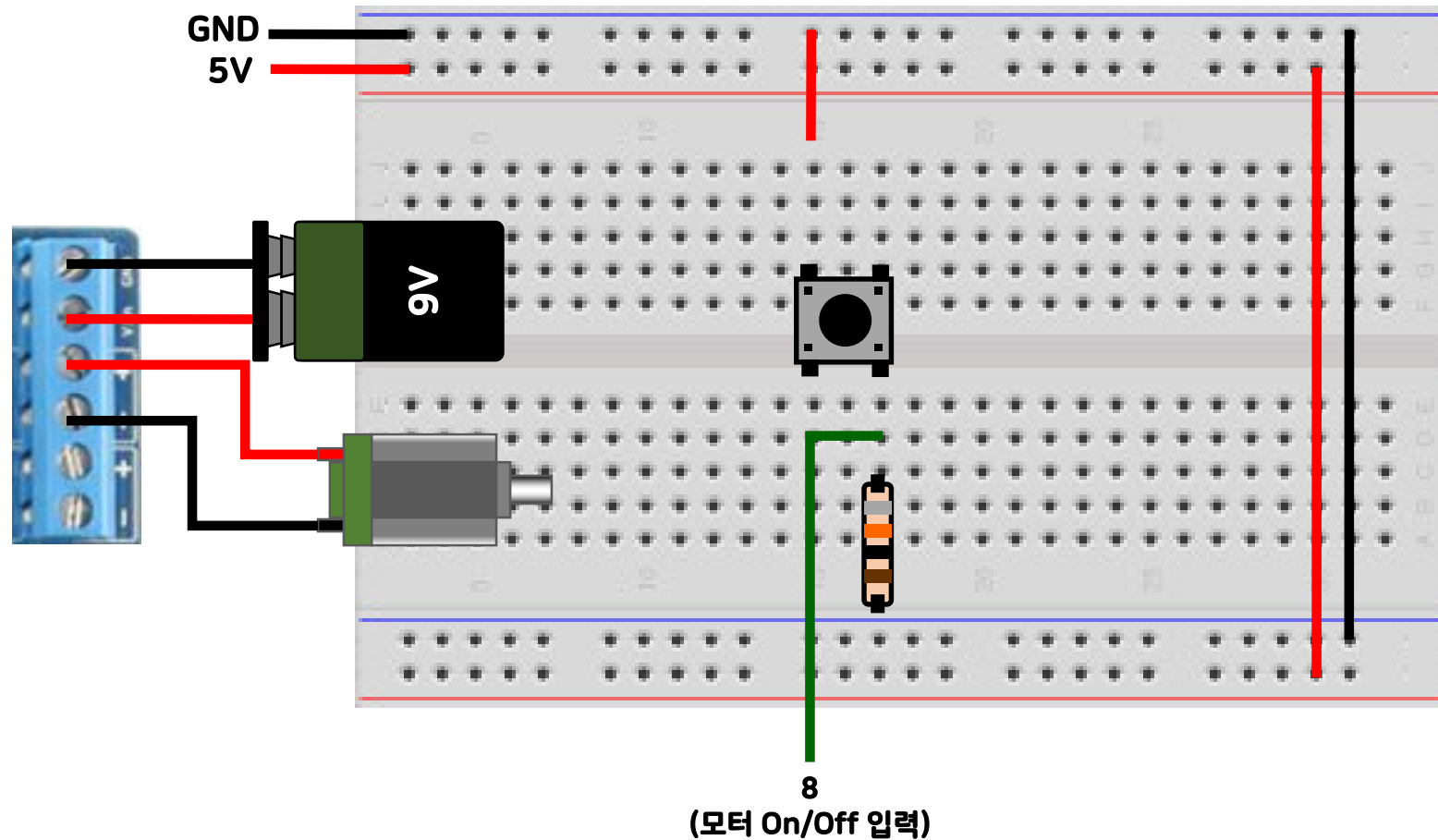
모터шил드를 이용한 DC모터 제어

- 예제 : 모터가 최대 속도로 5초간 회전하다가 급정지 한 후 5초 후에 반대 방향으로 최대 속도로 5초간 회전하다가 서서히 정지하는 스케치

1	const int PWM_A = 3, DIR_A = 12, BRAKE_A = 9, SNS_A = A0;	1	연결할 핀 설정
2		2	
3	void setup() {	3	
4	pinMode(BRAKE_A, OUTPUT);	4	채널 A의 출력 정지 핀 설정
5	pinMode(DIR_A, OUTPUT); pinMode(PWM_A, OUTPUT);	5	채널 A의 방향 제어 핀 설정
6		6	
7	Serial.begin(9600);	7	
8	Serial.println("Motor shield DC motor Test:\n");	8	
9	}	9	
10		10	
11	void loop() {	11	
12	digitalWrite(BRAKE_A, LOW);	12	정지 핀 값을 LOW로 설정
13	digitalWrite(DIR_A, HIGH);	13	모터 회전 방향을 앞으로 회전하도록 설정
14		14	
15	analogWrite(PWM_A, 255);	15	모터 속도를 최대로 설정 (255)
16		16	
17	delay(5000);	17	5초 동안 최대 속도로 모터를 회전시킨다
18	Serial.print("current consumption at full speed: ");	18	
19	Serial.println(analogRead(SNS_A));	19	최대 속도의 전류 값을 읽어온다
20		20	
21	Serial.println("Start braking\n");	21	
22	digitalWrite(BRAKE_A, HIGH);	22	모터를 빠르게 정지시킨다
23	delay(5000);	23	

24		24	
25	Serial.println("Backward");	25	
26	digitalWrite(BRAKE_A, LOW);	26	정지 핀을 LOW로 설정한다 (회전대기)
27	digitalWrite(DIR_A, LOW);	27	모터를 반대방향으로 회전시킨다
28		28	
29	analogWrite(PWM_A, 255);	29	속도를 최대로 설정한다
30	delay(5000);	30	
31	Serial.print("current consumption backward: ");	31	
32	Serial.println(analogRead(SNS_A));	32	
33		33	
34	analogWrite(PWM_A, 0);	34	모터의 속도를 0으로 설정한다 (서서히 정지한다)
35		35	
36	Serial.print("current brake: ");	36	정지 시의 전류 값을 읽어온다
37	Serial.println(analogRead(A0));	37	
38	Serial.println("End of the motor shield test with DC motors.!");	38	
39	}	39	

- 예제 : 버튼을 1번 누르면 150속도, 2번 누르면 255속도, 3번 누르면 정지가 되도록 한다



PushButton을 이용한 선풍기 제어

1	const int PWM_A = 3, swPin = 8;	1	속도 제어 핀, 스위치 핀
2	int switchState = 0, count = 0;	2	스위치 상태값, 스위치 누른 횟수
3		3	
4	void setup() {	4	
5	pinMode(PWM_A, OUTPUT);	5	
6	pinMode(swPin, INPUT);	6	
7	}	7	
8		8	
9	void loop() {	9	
10	switchState = digitalRead(swPin);	10	
11		11	
12	if (switchState == HIGH) {	12	
13	count++; delay(500);	13	
14	}	14	
15		15	
16	if (count == 0) {	16	스위치 누른 횟수에 따라 속도 제어
17	analogWrite(PWM_A, 0);	17	
18	} else if (count == 1) {	18	
19	analogWrite(PWM_A, 150);	19	
20	} else if (count == 2) {	20	
21	analogWrite(PWM_A, 255);	21	
22	} else {	22	
23	count = 0;	23	
24	}	24	
25	}	25	

Serial 통신을 이용한 선풍기 제어

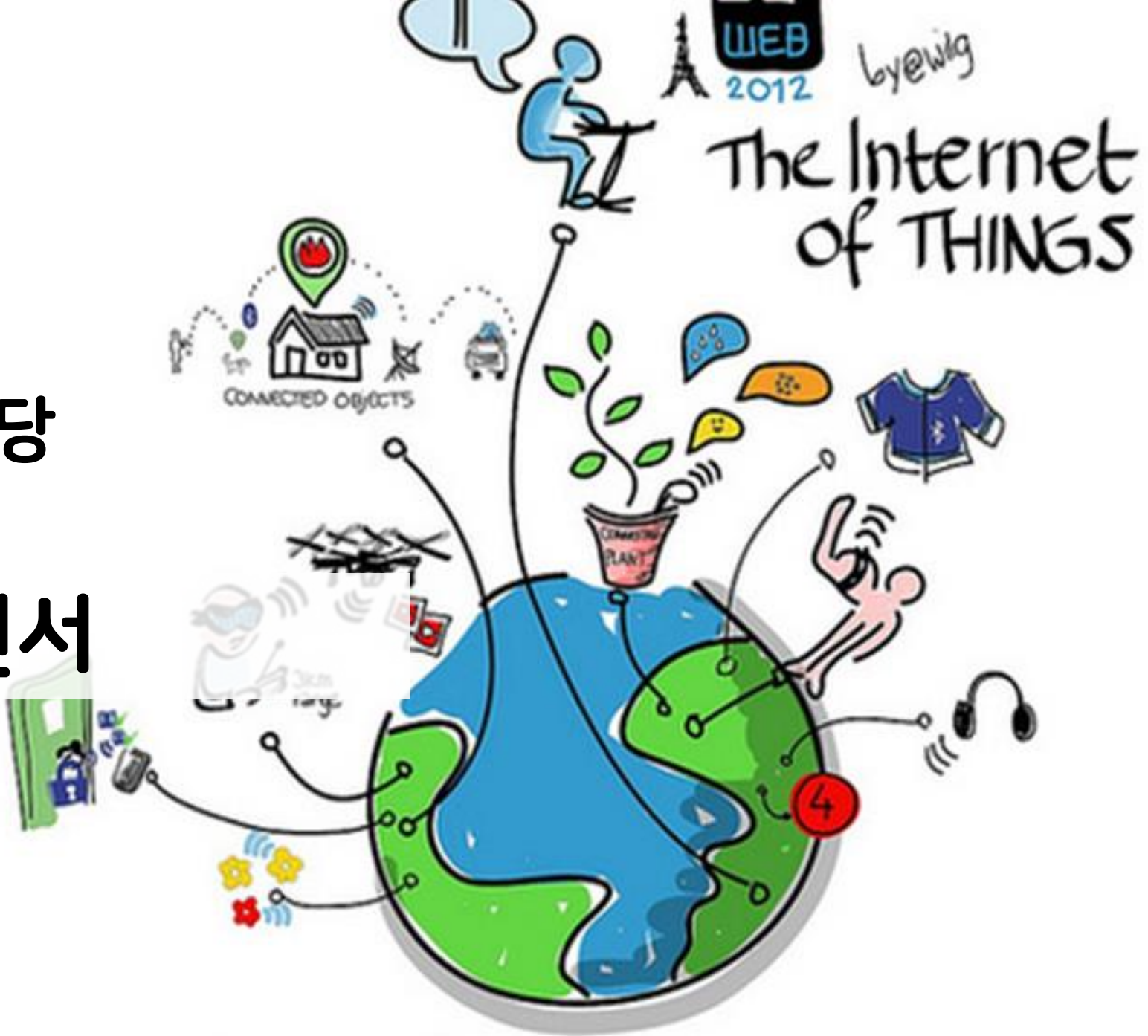
- 예제 : 시리얼에서 1을 보내면 150속도, 2를 보내면 255속도, 3을 보내면 정지가 되도록 한다

1	const int PWM_A = 3;	1	
2	int num = 0;	2	
3		3	
4	void setup() {	4	
5	pinMode(PWM_A, OUTPUT);	5	
6	Serial.begin(9600);	6	
7	}	7	
8		8	
9	void loop() {	9	
10	if(Serial.available()) {	10	
11	num = Serial.parseInt();	11	
12	}	12	
13		13	
14	if (num == 1) {	14	
15	analogWrite(PWM_A, 150);	15	
16	} else if (num == 2) {	16	
17	analogWrite(PWM_A, 255);	17	
18	} else if (num == 3) {	18	
19	analogWrite(PWM_A, 0);	19	
20	}	20	
21	}	21	

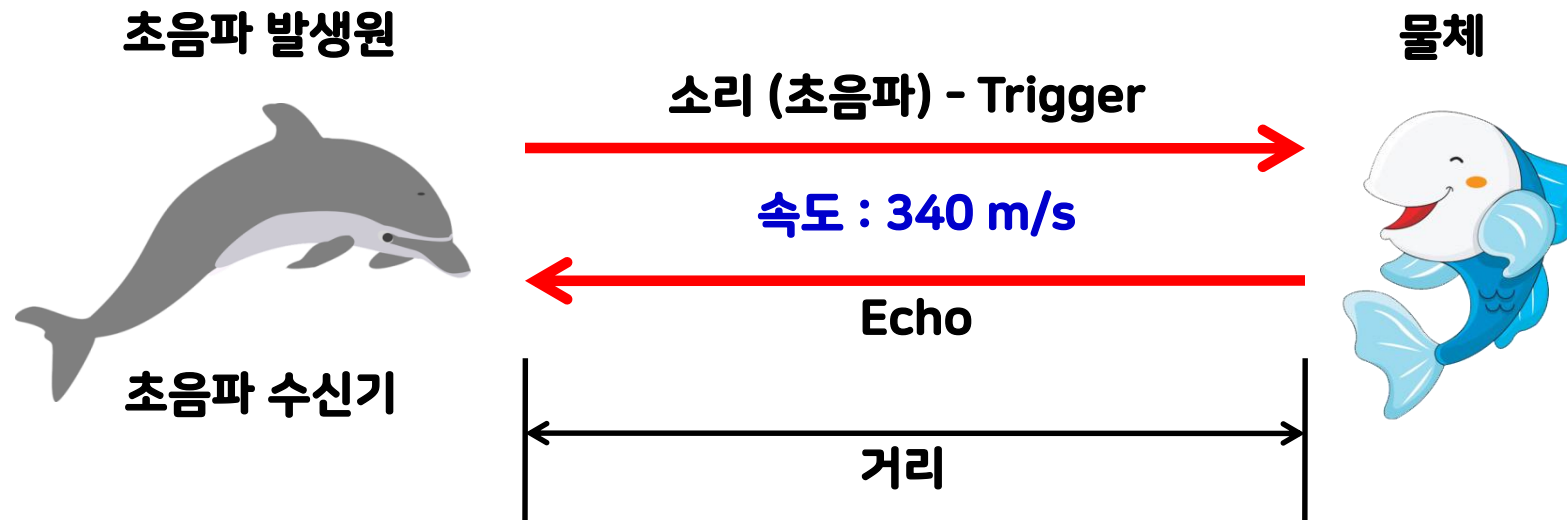


넷째 마당

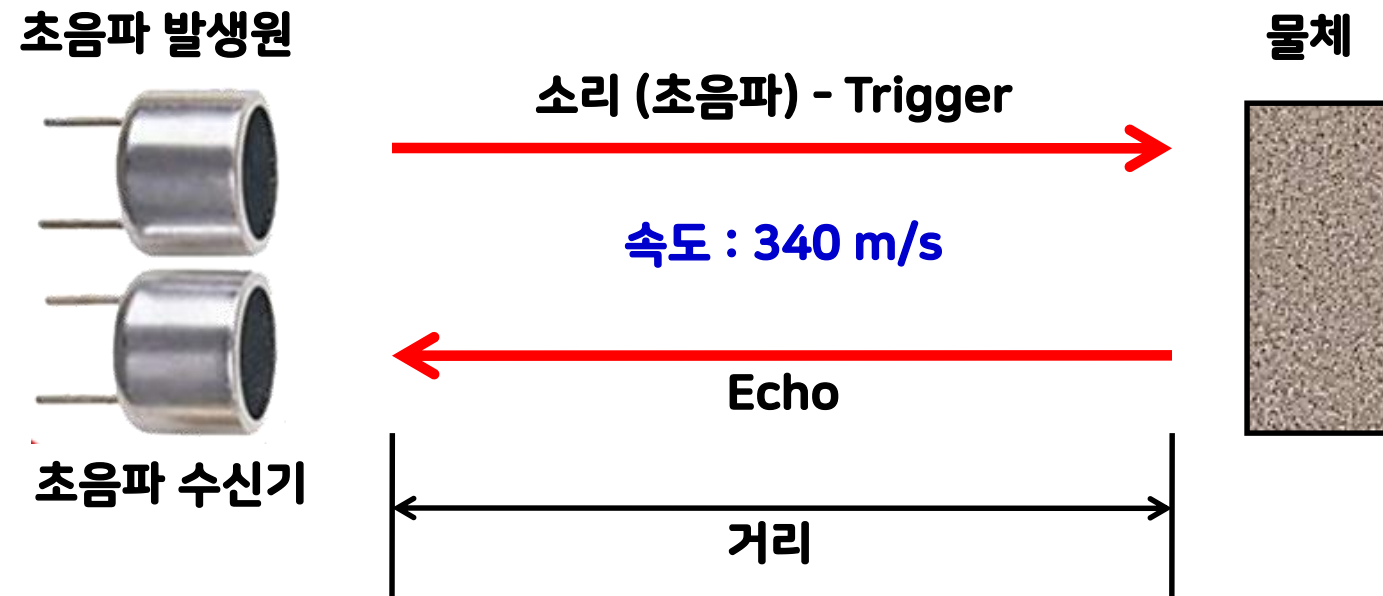
초음파 센서



- 초음파



- 초음파를 사용하여 거리를 측정하는 센서

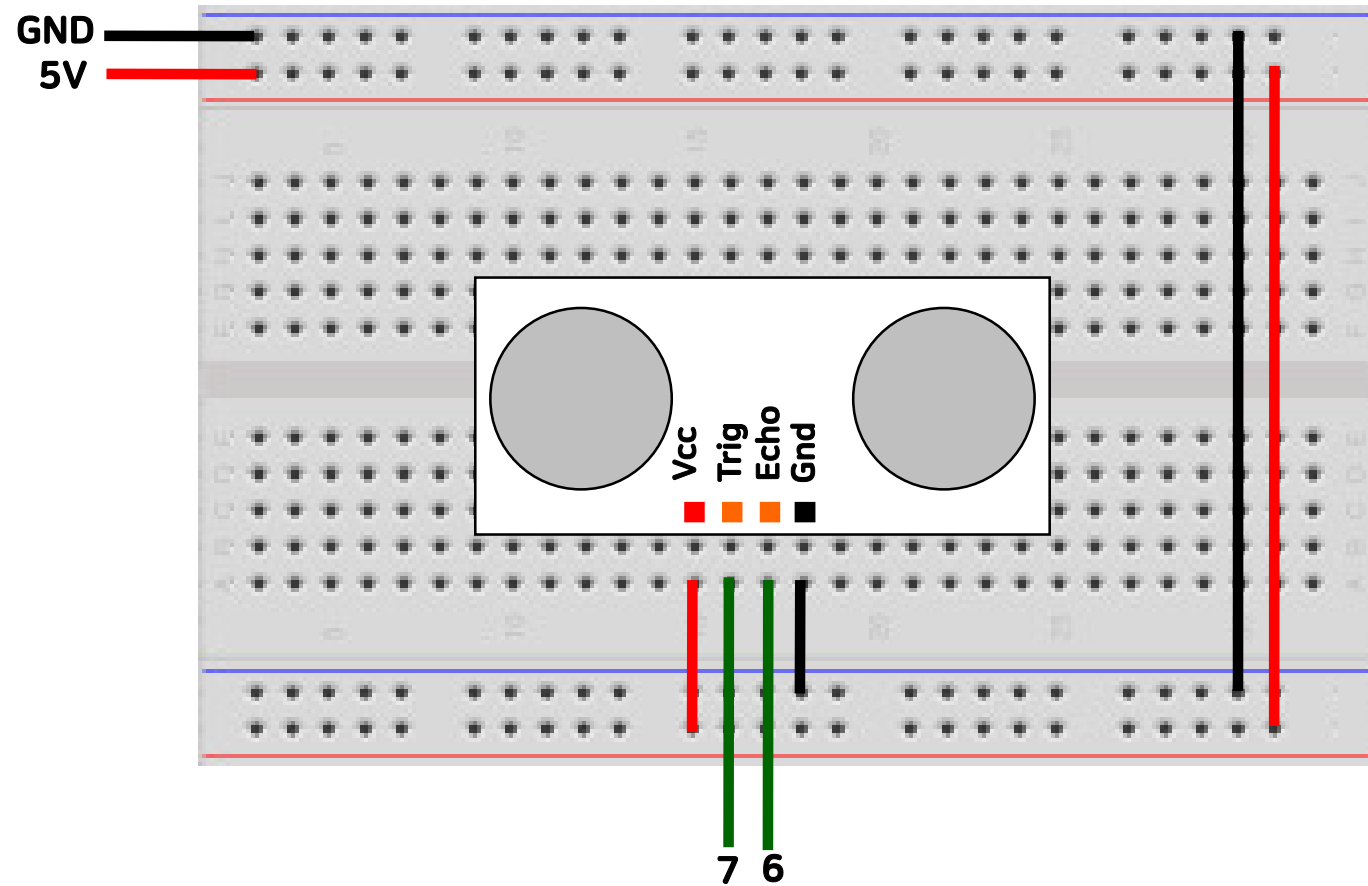


- HC-SR04는 4개의 핀으로 구성 (Vcc, Trig, Echo, Gnd)
- Trig 핀 : 신호를 보내는 핀
- Echo 핀 : 신호를 받는 핀
- 범위 : 3Cm - 4m, 30도



초음파 센서를 이용한 거리 측정

- 장애물의 거리를 Cm 단위로 계산하여 표시



초음파 센서를 이용한 거리 측정

- 예제 : 시리얼에서 1을 보내면 150속도, 2를 보내면 255속도, 3을 보내면 정지가 되도록 한다

1	const int triggPin = 7, echoPin = 6;	1	트리거 핀과 연결된 아두이노 핀
2		2	에코 핀과 연결된 아두이노 핀
3	void setup() {	3	
4	Serial.begin(9600);	4	
5	pinMode(triggPin, OUTPUT);	5	트리거 핀을 출력으로 설정
6	pinMode(echoPin, INPUT);	6	에코 핀을 입력으로 설정
7	}	7	
8		8	
9	void loop() {	9	
10	long duration, cm;	10	
11		11	
12	digitalWrite(triggPin, LOW);	12	HIGH 펄스를 확실하게 표현하기 위해 짧은 LOW
13	delayMicroseconds(5);	13	펄스를 먼저 발생시킨다
14	digitalWrite(triggPin, HIGH);	14	핑은 2μS 이상의 HIGH 펄스에 의해 실행
15		15	
16	duration = pulseIn(echoPin, HIGH);	16	
17		17	
18	cm = duration / 29 / 2;	18	ping으로부터 펄스 입력을 읽는다(펄스폭은 소리
19	Serial.print(cm); Serial.print("cm"); Serial.println();	19	의 이동거리에 비례)
20	delay(3000);	20	소리의 속도는 340m/s이므로 29μS/Cm이고
21	}	21	편도 거리를 구하므로 2로 나눈다

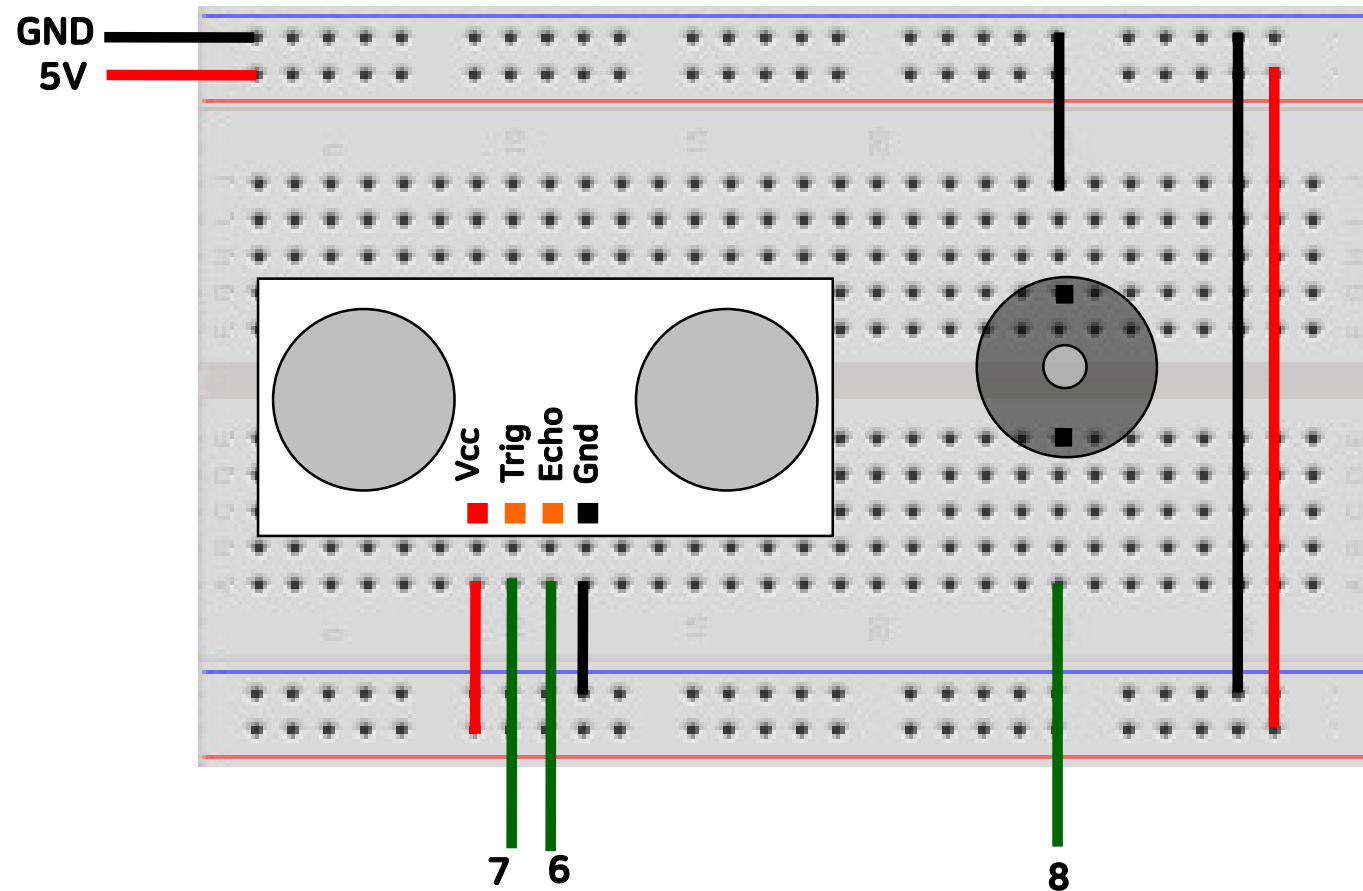


Quest

1m 내에 장애물이 있으면 피에조 센서로 소리를 내는 회로와 스케치를 작성하시오.

초음파 센서를 이용한 거리 알림

- 장애물이 1m 내에 있으면 소리를 발생



초음파 센서를 이용한 거리 알림

1	const int triggPin = 7;	1	트리거 핀과 연결된 아두이노 핀
2	const int echoPin = 6;	2	에코 핀과 연결된 아두이노 핀
3		3	
4	void setup() {	4	
5	pinMode(triggPin, OUTPUT);	5	트리거 핀을 출력으로 설정
6	pinMode(echoPin, INPUT);	6	에코 핀을 입력으로 설정
7	pinMode(8, OUTPUT)	7	
8	}	8	
9	void loop() {	9	
10	long duration, cm;	10	
11		11	
12	digitalWrite(triggPin, LOW);	12	
13	delayMicroseconds(2);	13	
14	digitalWrite(triggPin, HIGH);	14	
15	delayMicroseconds(5);	15	
16	digitalWrite(triggPin, LOW);	16	
17		17	
18	duration = pulseIn(echoPin, HIGH);	18	
19		19	
20	cm = duration / 29 / 2;	20	
21		21	
22	if(cm < 100) {	22	거리가 100cm이 하이면
23	tone(8, 262, 20);	23	피에조 센서로 소리를 발생시킨다
24	} else {	24	
25	noTone(8);	25	
26	}	26	
27	delay(3000);	27	
28	}	28	



Quest

거리 값에 따라 음계를 표현해보세요

- 음계의 주파수

도 : 262Hz

레 : 294Hz

미 : 330Hz

파 : 349Hz

솔 : 392Hz

라 : 440Hz

시 : 495Hz

도 : 524Hz

초음파 센서를 이용한 악기 제작

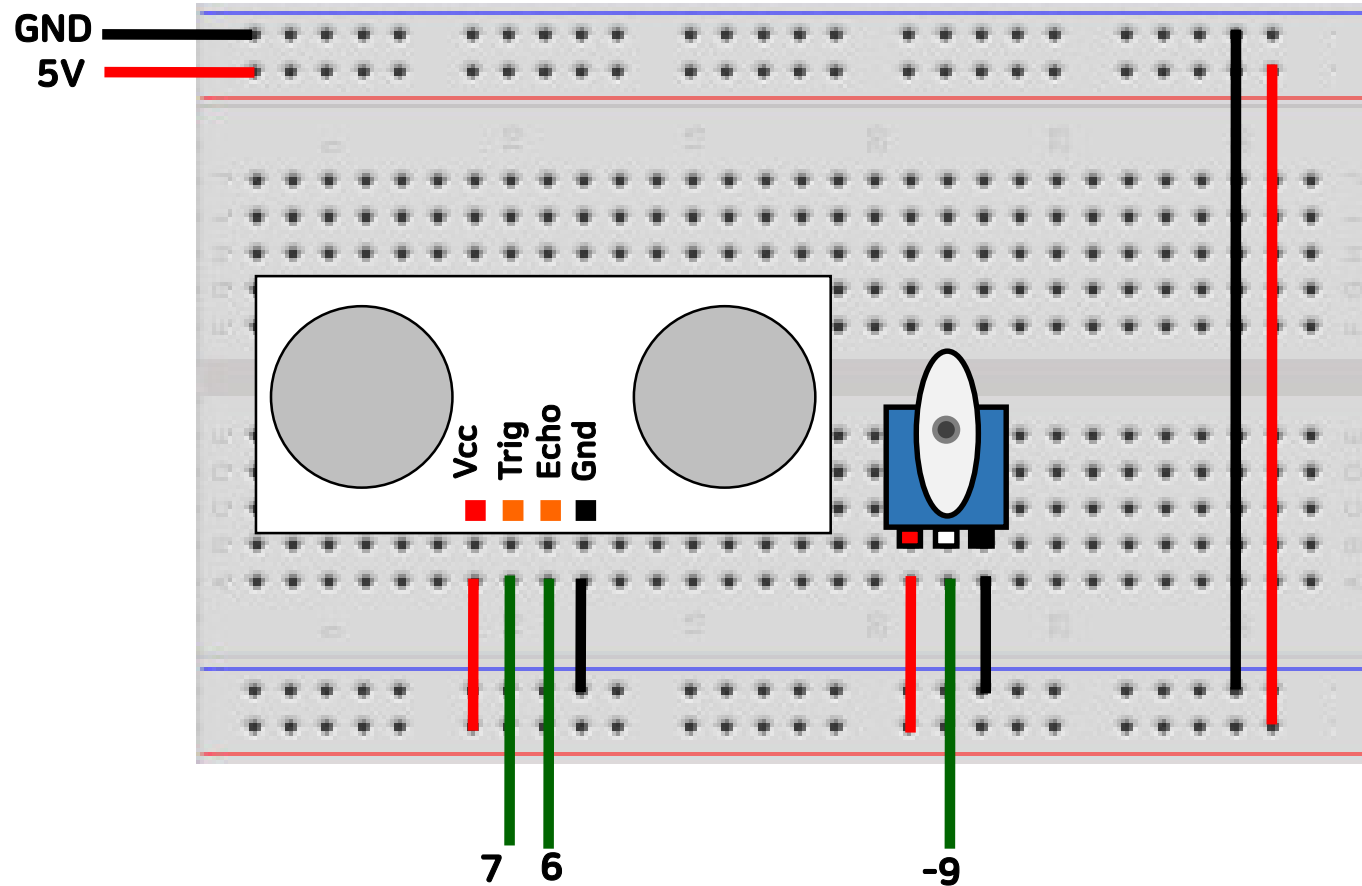
1	const int triggPin = 7, echoPin = 6;	1	
2		2	
3	void setup() {	3	
4	pinMode(triggPin, OUTPUT); pinMode(echoPin, INPUT);	4	
5	pinMode(8, OUTPUT);	5	
6	}	6	
7		7	
8	void loop() {	8	
9	long duration, cm;	9	
10		10	
11	digitalWrite(triggPin, LOW); delayMicroseconds(2);	11	
12	digitalWrite(triggPin, HIGH); delayMicroseconds(5);	12	
13	digitalWrite(triggPin, LOW);	13	
14		14	
15	duration = pulseIn(echoPin, HIGH);	15	
16	cm = duration / 29 / 2;	16	
17		17	
18	if(cm >= 10 && cm <= 40) {	18	10cm - 4cm 거리에서 소리내기
19	cm = map(cm, 10, 40, 262, 524);	19	거리를 주파수로 변환
20	tone(8, cm, 300);	20	해당 주파수로 300ms 동안 소리 내기
21	} else {	21	
22	noTone(8);	22	
23	}	23	
24	delay(350);	24	소리 내는 시간에 맞추어 설정
25	}	25	

Quest 주차장 차단기 만들기

- 초음파 센서로 거리 측정
- 서보 모터로 차단기 열고 닫기

초음파 센서를 주차장 차단기 만들기

- 장애물이 1m 내에 있으면 서보모터를 회전



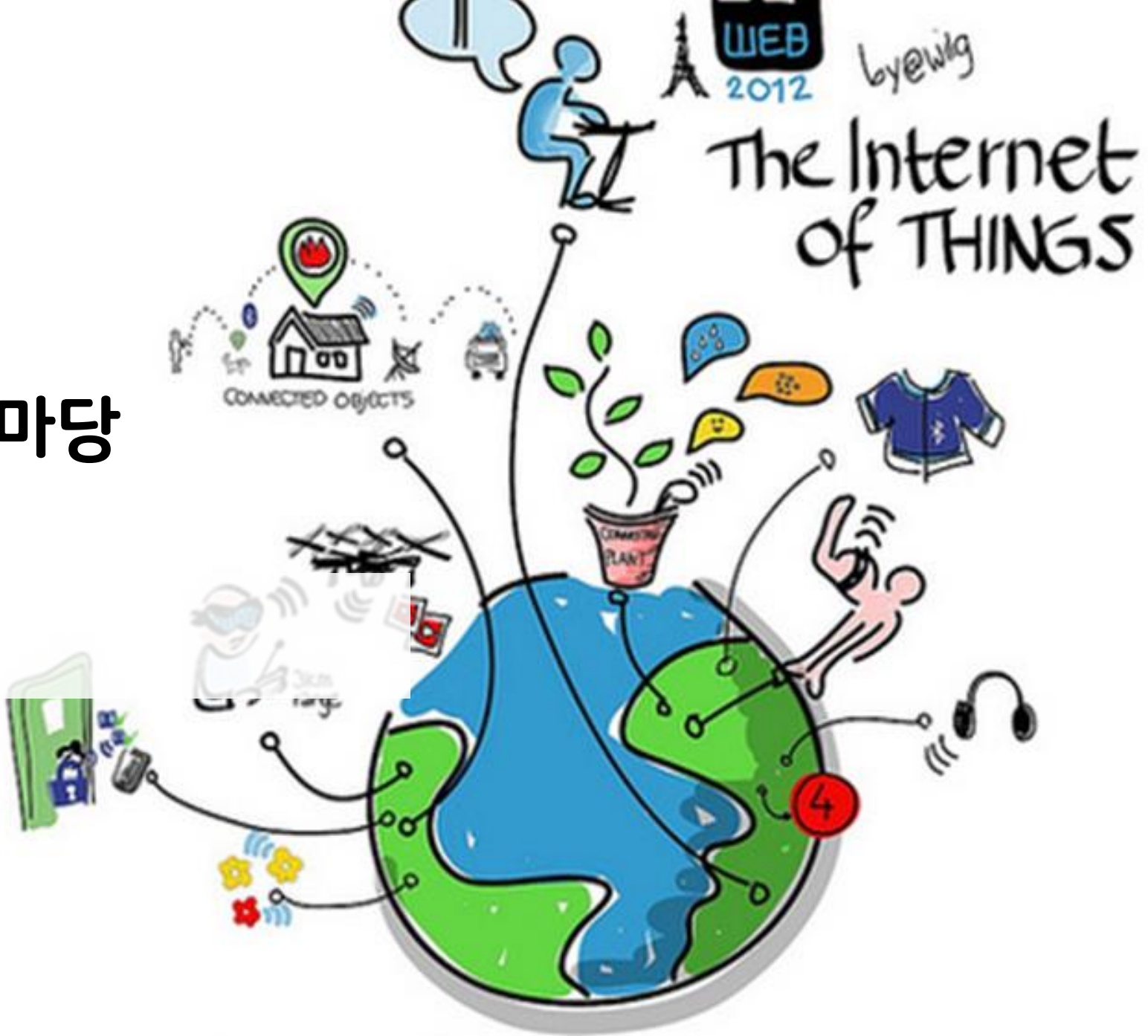
초음파 센서를 주차장 차단기 만들기

```
1 #include <Servo.h>
2 Servo myServo;
3 const int triggPin = 7, echoPin = 6;
4 int state = 0;
5
6 void setup() {
7   pinMode(triggPin, OUTPUT); pinMode(echoPin, INPUT);
8   pinMode(8, OUTPUT);
9 }
10
11 void loop() {
12   long duration, cm;
13
14   digitalWrite(triggPin, LOW); delayMicroseconds(2);
15   digitalWrite(triggPin, HIGH); delayMicroseconds(5);
16   digitalWrite(triggPin, LOW);
17
18   duration = pulseIn(echoPin, HIGH);
19   cm = duration / 29 / 2;
20
21   if (cm <= 30 && state == 0) {
22     myServo.attach(9); myServo.write(90); delay(300); myServo.detach();
23     state = 1;
24   } else if (cm > 30 && state == 1) {
25     myServo.attach(9); myServo.write(0); delay(300); myServo.detach();
26     state = 0;
27   }
28   delay(3000);
29 }
```

```
1
2
3
4 이전과 동일한 각도인지 검사하기 위함
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 만약 30cm보다 작고 이전이 0도였다면
22 서보 연결 → 회전 → 대기 → 연결 종료
23 상태 설정 (90도)
24 만약 30cm보다 크고 이전이 90도였다면
25 서보 연결 → 회전 → 대기 → 연결 종료
26 상태 설정 (0도)
27
28
29
```

다섯째 마당

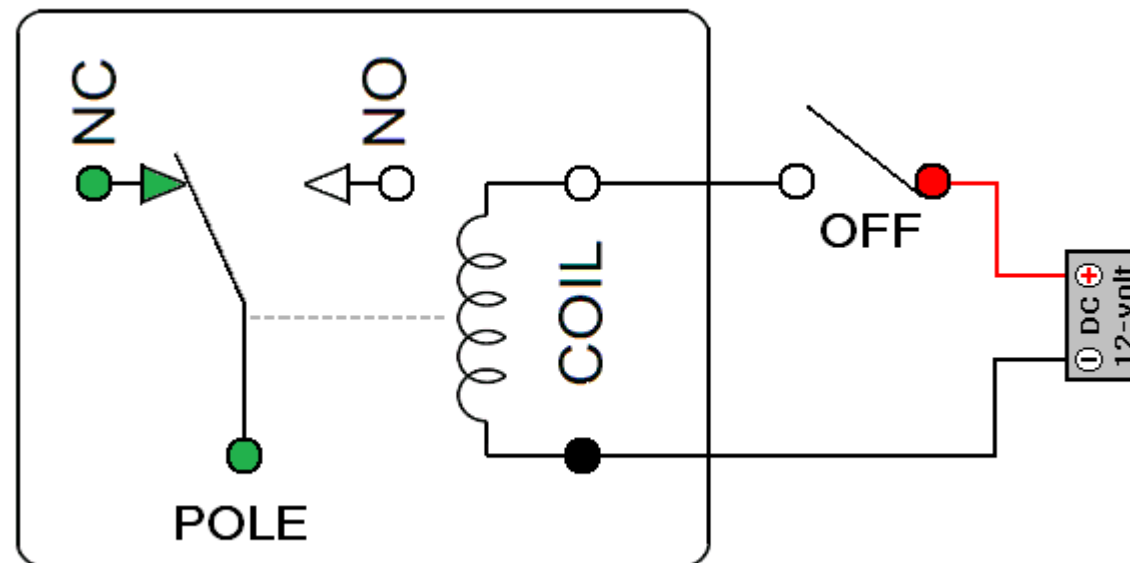
릴레이



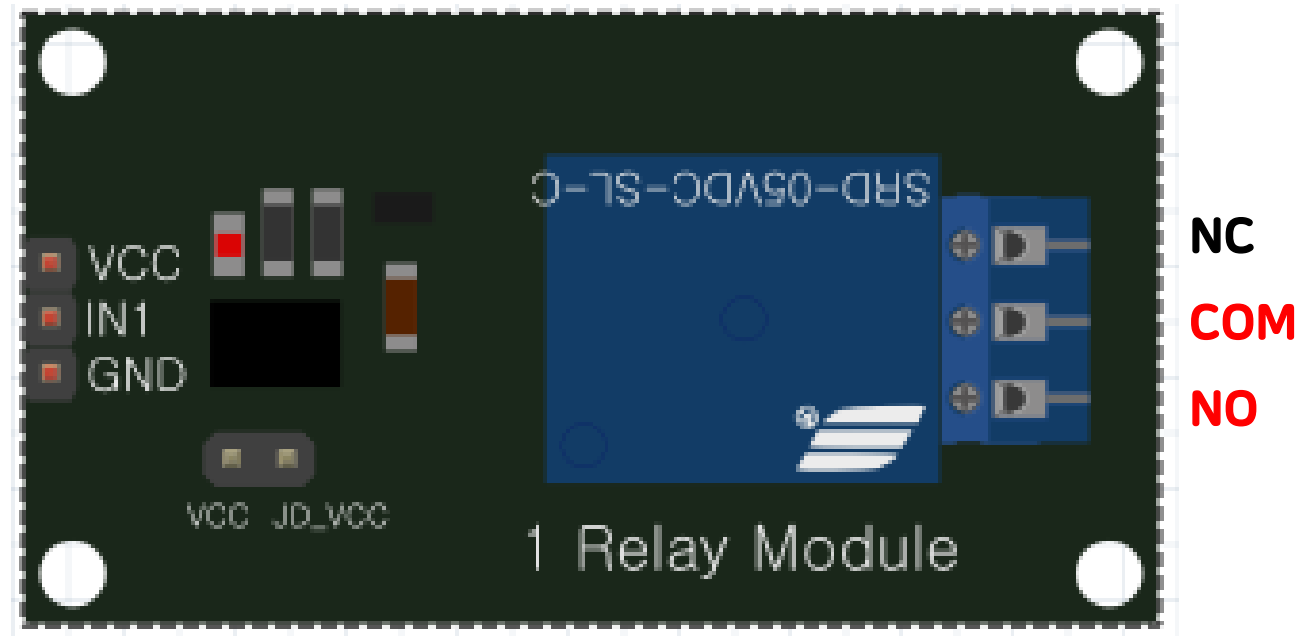
- 일반적으로 접하는 스위치는 수동으로 ON, OFF 해주지만, 릴레이는 **자동으로 ON, OFF** 할 수 있게끔 해주는 전자부품
- 별도로 분리되어 흐르는 전기를 스위칭할 수 있는 신호 또는 펄스를 만들어 줌
- 릴레이는 작동하기 위해 필요한 전압은 낮지만 입력될 수 있는 전압은 높음 → **흔히 낮은 전압/전류를 이용하여 더 높은 전압/전류를 제어하는데 많이 사용**



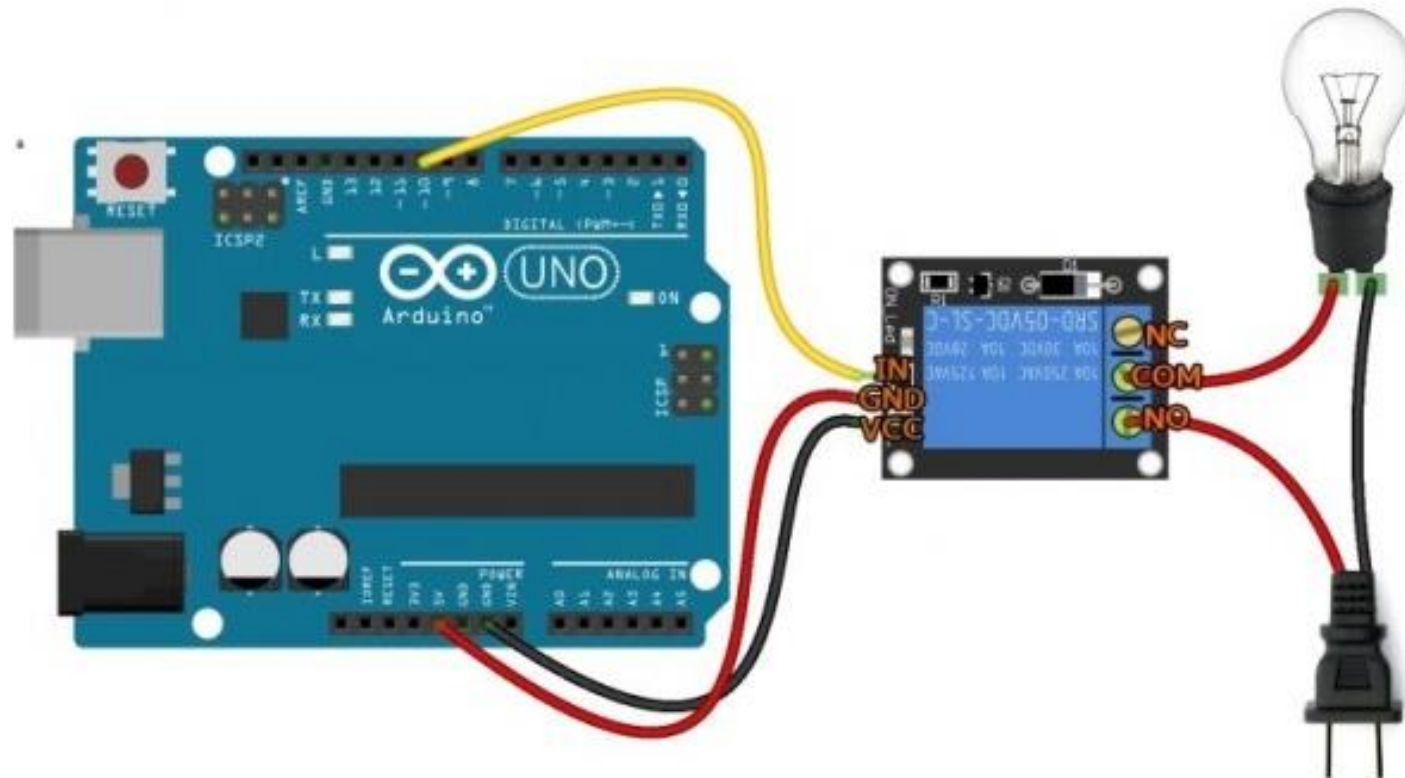
- 동작원리 : 코일에 전류가 흐르면 전자석이 되고 스위치를 끌어당겨 스위치가 ON이 되게 함
- NO(Normally Open) : 릴레이 OFF → 전기가 통하지 않음, 릴레이 ON → 전기가 통함
- NC(Normally Close) : 릴레이 OFF → 전기가 통함, 릴레이 ON → 전기가 통하지 않음



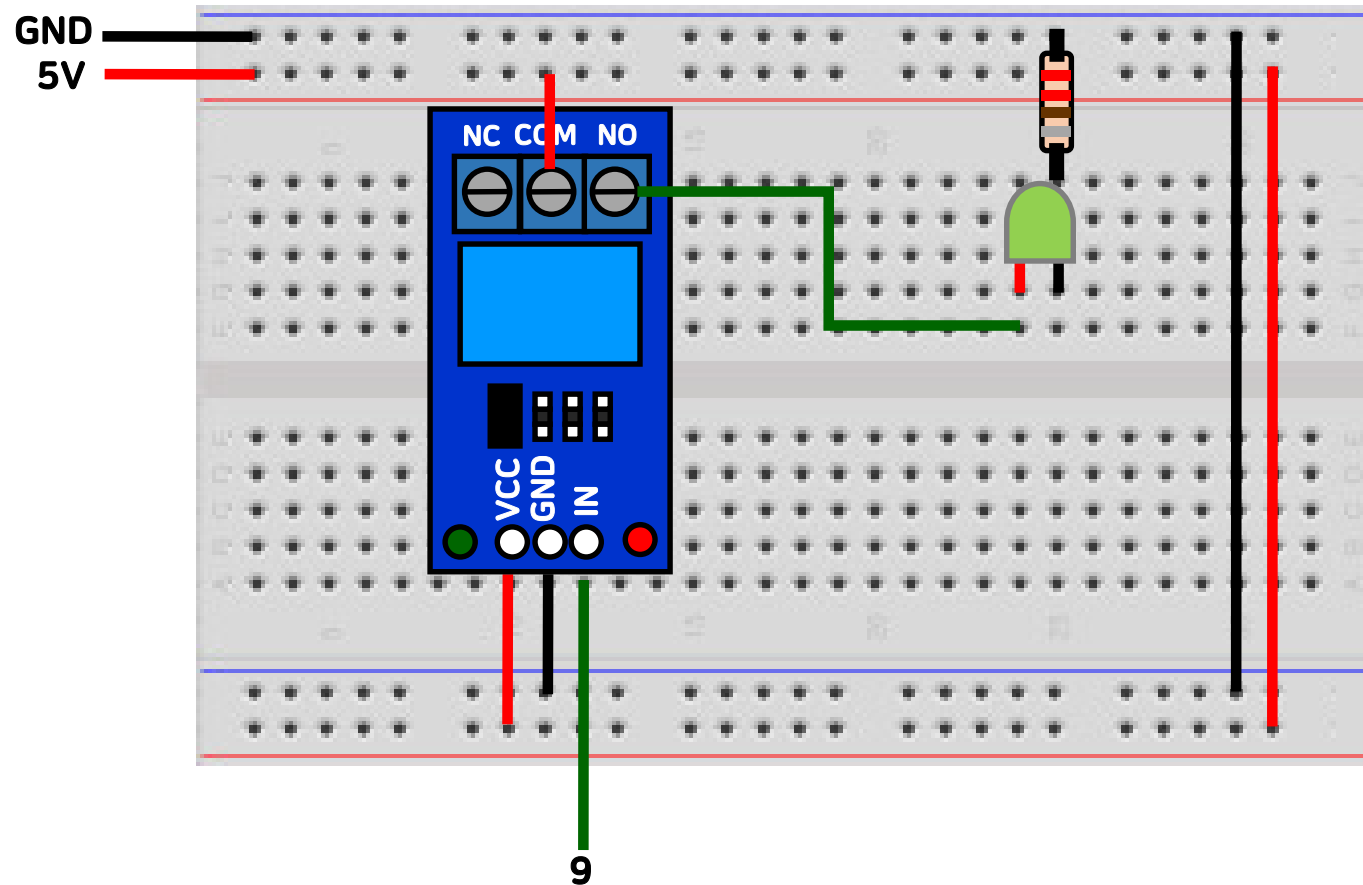
- 1채널 릴레이 모듈
- 동작전압 : 5V DC
- 릴레이 출력 최대 접점 : AC250V 10A / DC30V 10A



아두이노, 릴레이, 전등 연결



- 2초마다 LED를 ON/OFF

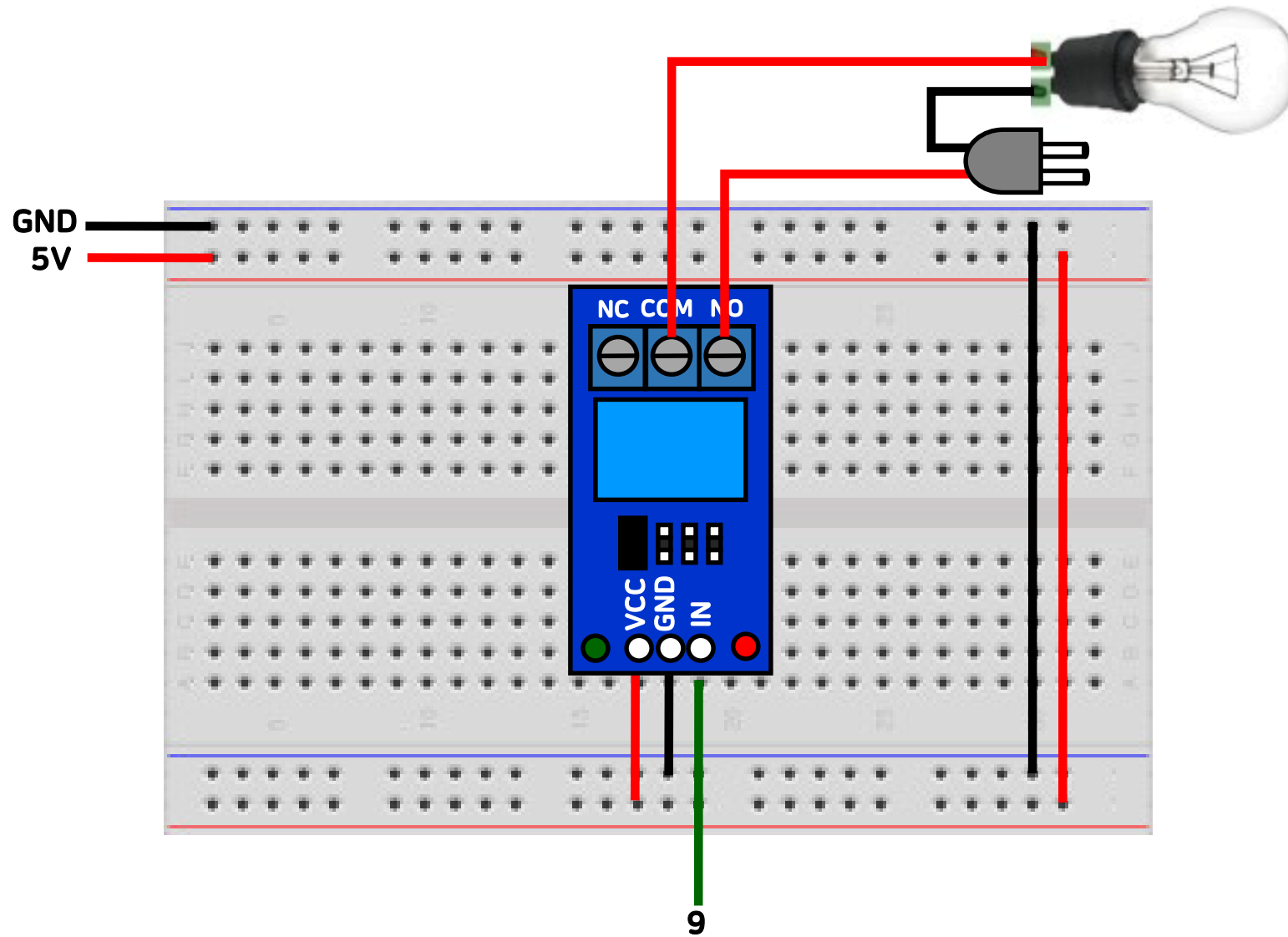


릴레이로 LED 제어

1	int relayPin = 9;	1	
2		2	
3	void setup() {	3	
4	Serial.begin(9600);	4	
5	pinMode(relayPin, OUTPUT);	5	
6	}	6	
7		7	
8	void loop() {	8	
9	digitalWrite(relayPin, HIGH);	9	
10	delay(2000);	10	
11	digitalWrite(relayPin, LOW);	11	
12	delay(2000);	12	
13	}	13	

Quest NC 대신에 NO에 연결하고 테스트해보자.

릴레이로 전등 제어





Quest

조도 센서를 이용하여 일정 조도 이하로 떨어지면 전등을 On 시키고 이상이면 Off 시키도록 하시오.

릴레이로 전등 제어

