

소프트웨어 프로젝트 2

거리센서, Part I

2024년 2학기

국민대학교
소프트웨어학부/인공지능학부
주용수, 최진우, 한재섭, 허대영
{ysjoo, jaeseob, jnwochoi, dyheo}@kookmin.ac.kr

센서

- 센서

- 자연(센서 주변)에 존재하는 물리량을 읽어들이는(인지) 장치
- 물리량을 전압의 형태로 표현하는 경우가 일반적
 - 우리가 사용할 초음파 센서는 거리를 파형이 high인 구간의 길이로 계산

- 아날로그 센서


- 물리량의 변화를 연속적인 전압의 변화로 출력
 - 예: $0^{\circ}\text{C} \sim 100^{\circ}\text{C} \rightarrow 0\text{V} \sim 5\text{V}$ 로 변환
 - $50^{\circ}\text{C} \Rightarrow 2.5\text{V}$, $50.3^{\circ}\text{C} \Rightarrow 2.515\text{V}$, $50.4^{\circ}\text{C} \Rightarrow 2.52\text{V}$

ADC

- 디지털 센서

- 물리량의 변화를 특정 범위의 숫자로 양자화(quantize)하여 표현
 - 예: $0^{\circ}\text{C} \sim 100^{\circ}\text{C} \rightarrow 0 \sim 255$ 로 변환
 - $50^{\circ}\text{C} \Rightarrow 128$, $50.3^{\circ}\text{C} \Rightarrow 128$, $50.4^{\circ}\text{C} \Rightarrow 129$

디지털 센서

- 구조
 - 아날로그 센서 + ADC(아날로그-디지털 변환기)가 하나의 부품으로 구성됨
(즉, 아날로그 센서 값을 내부에서 디지털로 바꾸어 출력)
- 직렬통신 인터페이스 사용
 - 디지털 신호를 주고받기 위해 필요
 -  UART, SPI, I2C 등
- 장점
 - 아날로그 신호 경로 최소화 -> **노이즈 억제에 유리**
- 단점
 - ADC 내장->비용 증가
 - 통신을 위해 SPI 등의 프로토콜 이해 필요

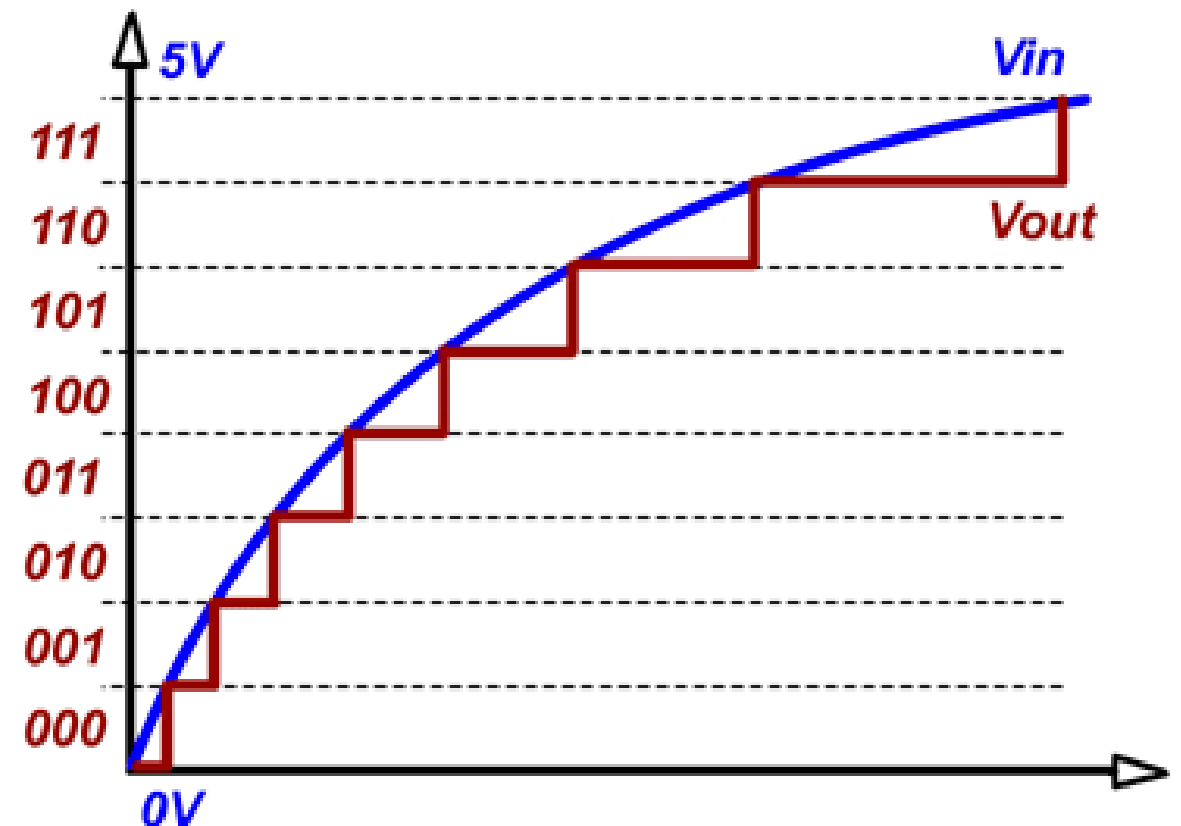
아날로그 센서 ADC

- 별도의 ADC 필요
 - 센서 출력은 연속적으로 변하는 아날로그 전압
 - 컴퓨터에 의한 처리 위해서는 디지털로 변환해야 함
- 하드웨어 인터페이스
 - 센서 출력핀과 ADC 사이는 signal (전압)과 ^{ground}gnd, 2개의 핀으로 연결 가능
- 장점
 - 아두이노와 같이 자체 ADC 보유한 CPU 사용시 비용 절감 가능
- 단점
 - 노이즈에 취약 ->
 - 센서와 ADC 사이에 노이즈 방지 대책 필요 (PCB 또는 shield 등의 설계시)

아날로그-디지털 변환기 (ADC)

- Analog-to-Digital Converter (ADC)
 - 전압을 특정 자리수(=해상도)의 이진수 숫자로 변환
 - 8-bit 해상도 ADC: 출력값을 0 ~ 255($=2^8-1$)의 숫자로 표현
 - 아두이노: 10bit 해상도(0~1023)의 ADC 내장 (A0~A5 핀)

- 양자화 과정에서 연속적인 측정값이 불연속 값으로 변함 => 정보 손실
- 일단 양자화된 이후에는 디지털 처리가 되어 더 이상의 정보 손실을 방지할 수 있음



아두이노와 아날로그 센서 연결

- 아날로그 센서 연결
 - 아두이노에 내장된 ADC(A0~A5 핀)와 센서 출력을 연결
 - 센서 출력전압은 0~1023 사이의 숫자로 읽을 수 있음
 - analogRead(pin) 사용

<https://www.arduino.cc/reference/ko/language/functions/analog-io/analogread/>



```
int analogPin = 3;  // 가변 저항의 가운데 핀이 아날로그 핀 3에 연결됨
                    // 바깥쪽은 그라운드와 +5v에 연결됨
int val = 0;        // 읽은 값을 저장할 변수
void setup() {
  Serial.begin(115200); // 시리얼 설정
}
void loop() {
  val = analogRead(analogPin); // 입력 핀 읽기
  Serial.println(val);         // 입력 값 모니터링
}
```

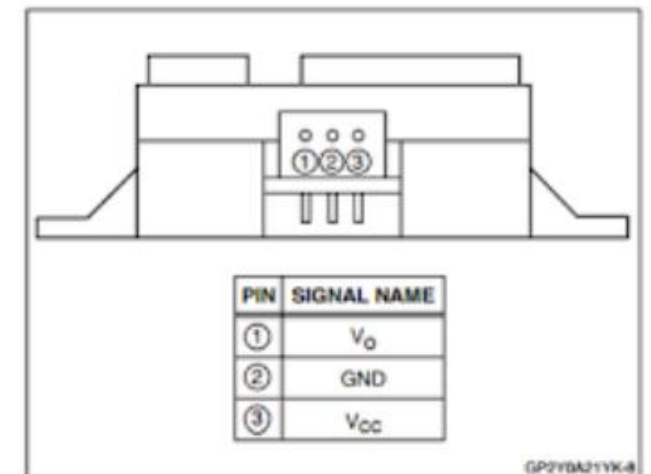
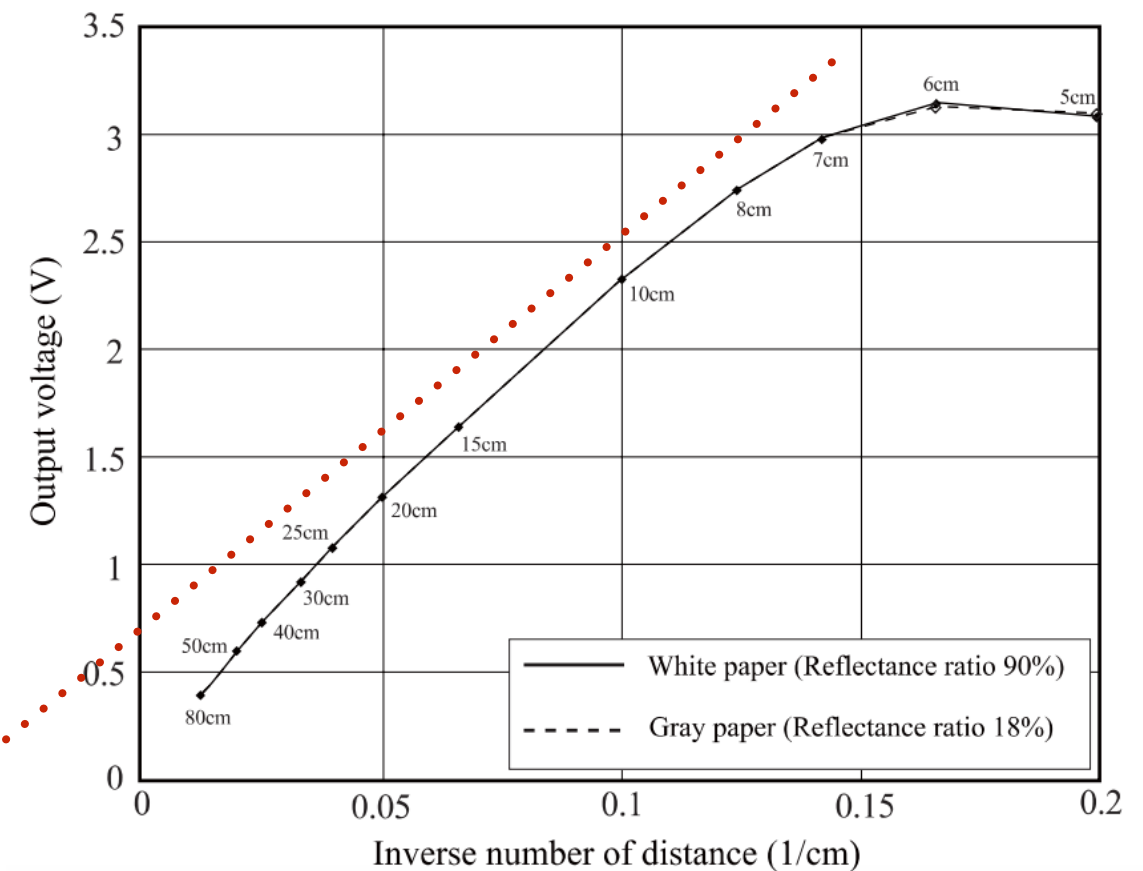
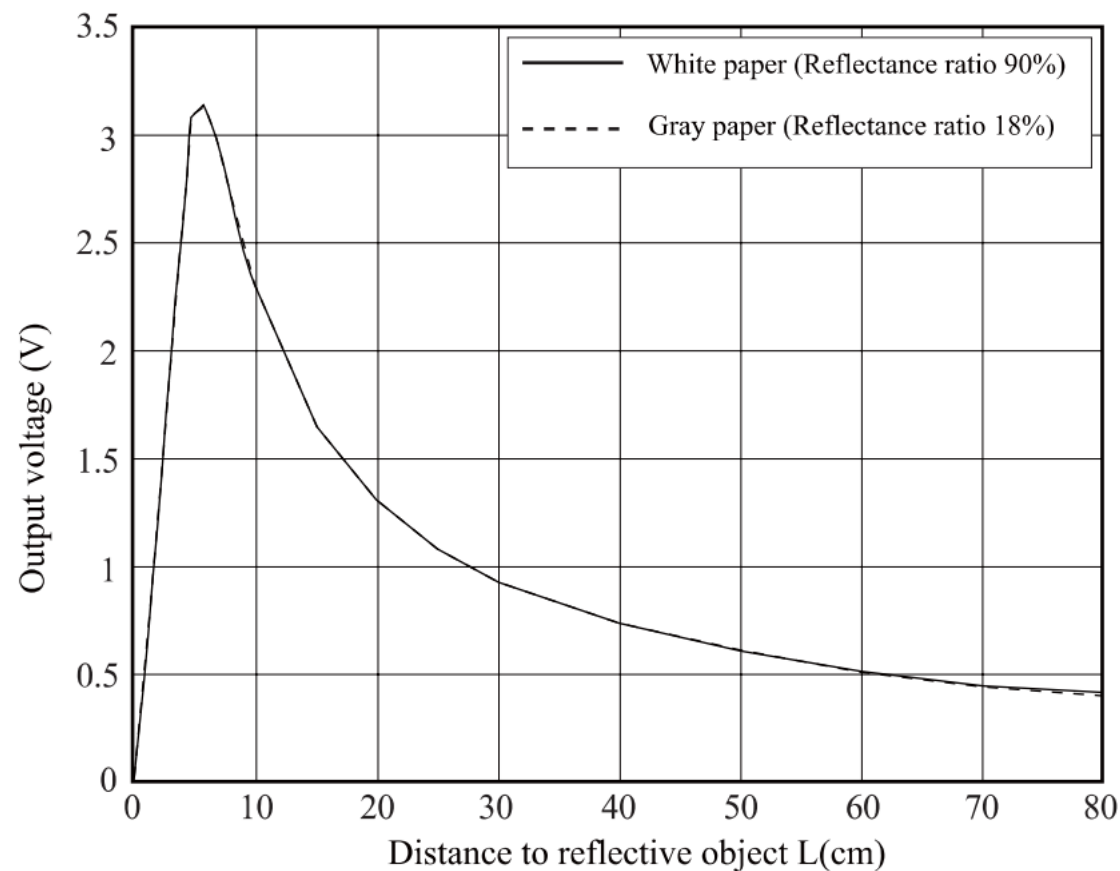


Figure 1. Pinout

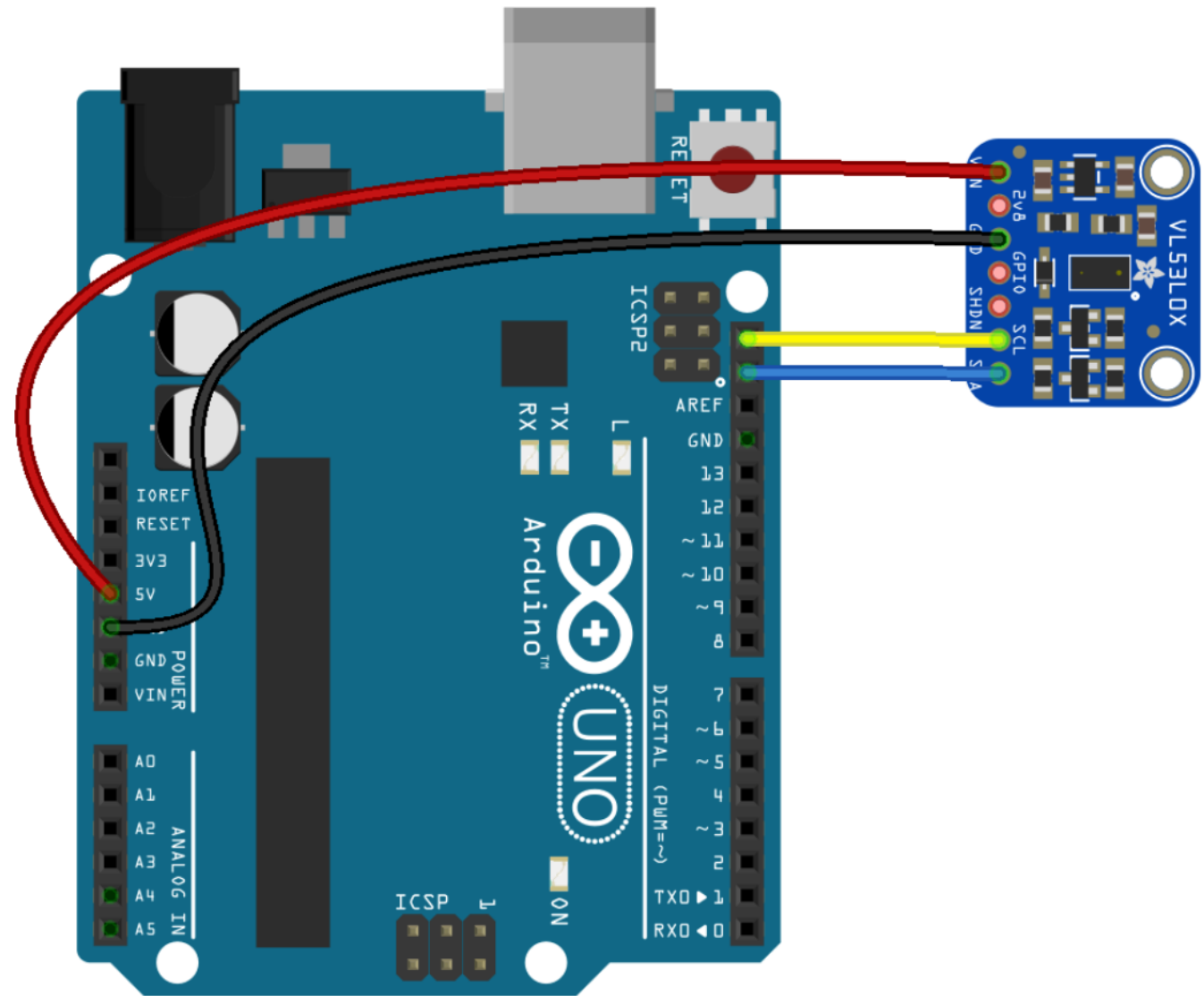
전압-물리량 변환

- 변환 방법
 - Linear model
 - 전압-물리량 관계를 1차 방정식으로 근사
 - lookup table
 - 다양한 전압에 대한 물리량을 테이블로 기록
 - 테이블에 존재하지 않는 전압에 대한 물리량 변환은 보간법 사용



아두이노와 디지털 센서 연결

- 직렬 통신 인터페이스
 - I²C 또는 SPI 등을 사용
 - 해당 센서로 검색하여 I²C (또는 SPI) 모듈 초기화 및 데이터 통신 예제 코드를 확보하여 사용



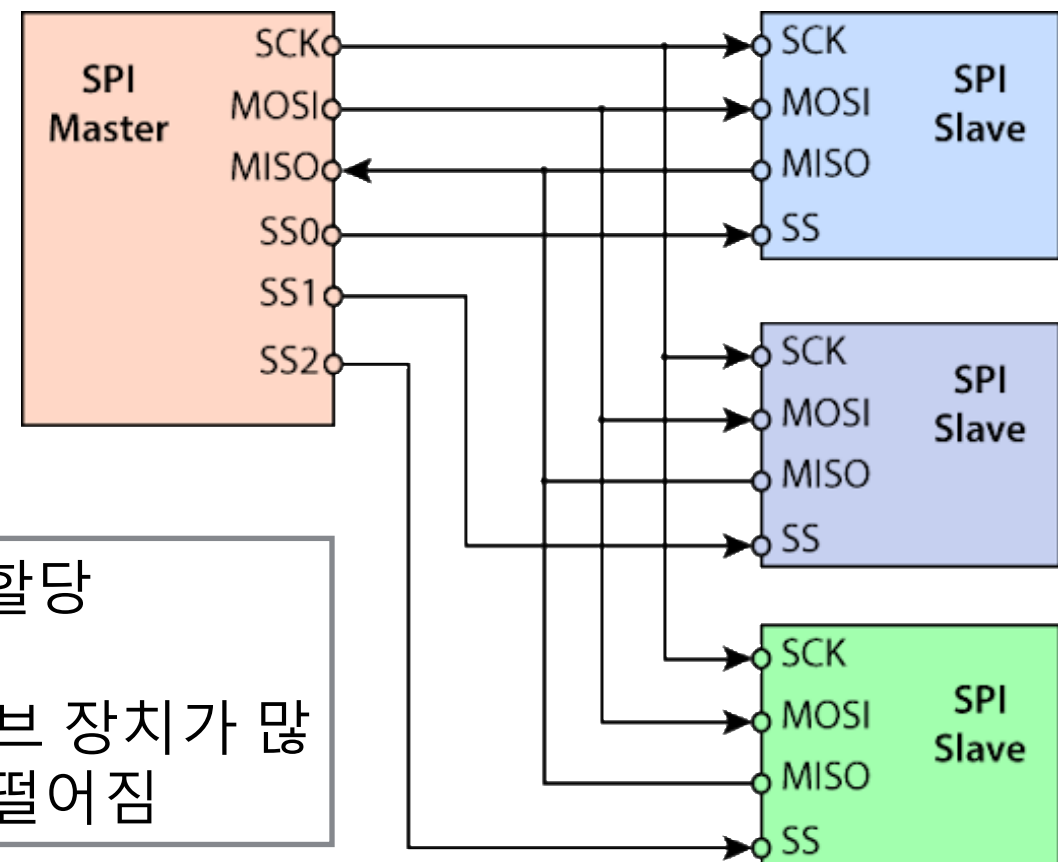
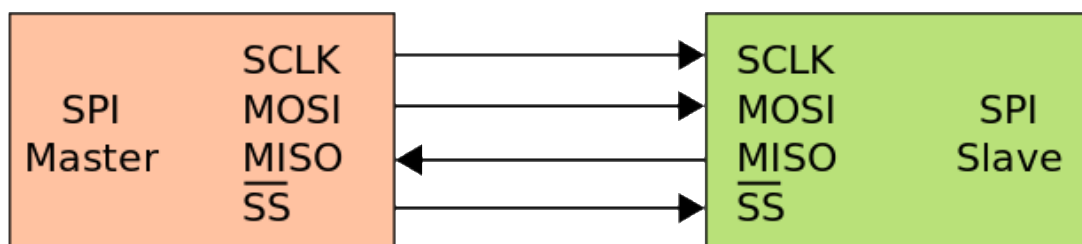
<https://learn.adafruit.com/adafruit-vl53l0x-micro-lidar-distance-sensor-breakout/arduino-code>

직렬통신 인터페이스 (참고)

- 직렬 통신 (serial communication)
 - 특정 순간에 하나의 비트(0 또는 1)만 전송하는 통신 방식
 - 단방향 통신
 - 송신(tx, transmitter)과 수신(rx, receiver)을 위해 별도의 선 할당
 - 두 모듈 사이의 통신에 적합
 - SPI (serial peripheral interface)
 - 양방향 통신
 - 하나의 선이 송수신을 모두 담당 (시분할 방식)
 - 다수의 모듈을 연결하는 통신 버스 구현에 적합
 - I²C (inter-integrated circuit)

직렬통신 인터페이스 (참고)

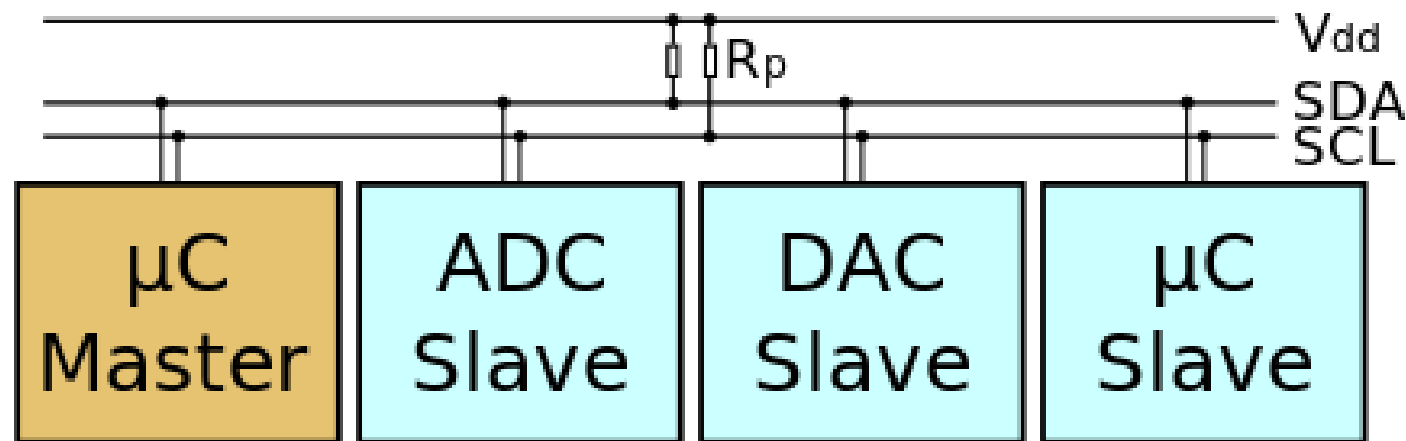
- SPI (serial peripheral interface)
 - SCLK: serial clock
 - MOSI: master output slave input
 - MISO: master input slave output
 - /SS: slave select (slave device 개수만큼의 /SS 핀 필요)



- 양방향 통신을 위해 MOSI, MISO의 두 개의 선 할당
- 클럭 SCLK에 동기화된 데이터 전송
- Select 신호선 기반 슬레이브 장치 선택: 슬레이브 장치가 많아질 경우 select 신호선 개수 증가로 확장성이 떨어짐

직렬통신 인터페이스 (참고)

- I²C (inter-integrated circuit)
 - SDA: serial data line
 - SCL: serial clock line
 - 7-bit address (최대 128개의 slave device 연결 가능)



- 클럭 SCL에 동기화된 데이터 전송
- 단일 신호선으로 양방향 통신 구현 (시분할)
- 단일 신호선으로 다수의 slave와 통신 (시분할)
- slave 선택을 위해 7-bit address 사용: SPI 대비 slave 선택 과정이 복잡
- slave 개수 증가시 확장성이 떨어짐:
최대 128개의 device를 SDA/SCL 버스에 연결하는 것만으로 회로 구현 가능

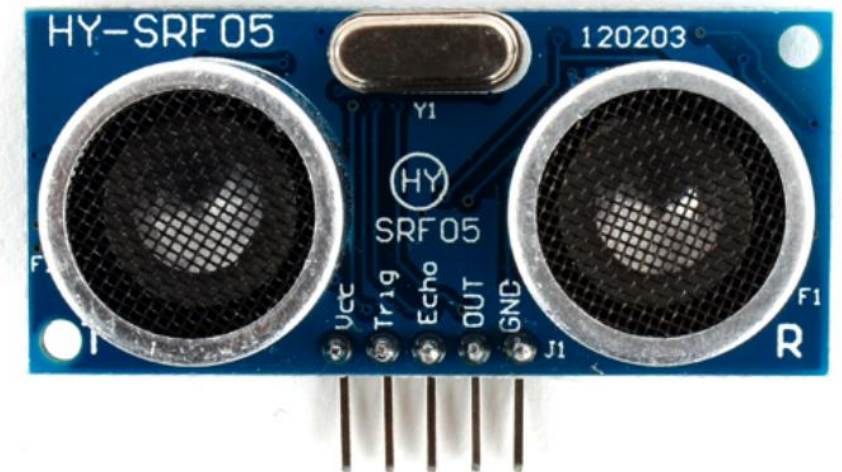
거리센서

- 전방의 물체(장애물)와 센서 사이의 거리를 측정
- 다양한 방식
 - 초음파(ultrasonic) / 적외선(infrared) / 레이저 (laser) / 전파 (radio)
- FOV (field of view, 화각 또는 시야각)
 - 센서가 물체를 인지할 수 있는 범위
 - FOV 내에 존재하는 물체 중 가장 가까운 물체의 거리를 인식
- 센서 종류에 따라 FOV가 달라짐
 - VL53L0X laser distancing sensor: 25°
 - HC-SR04 ultrasonic sensor: horizontal ~21°, vertical ~4°
 - Sharp infrared 2Y0A21: unknown (narrower)
 - DT-20P224B displacement measurement sensor: 0.68°

<https://www.st.com/resource/en/datasheet/vl53l0x.pdf>

초음파 센서

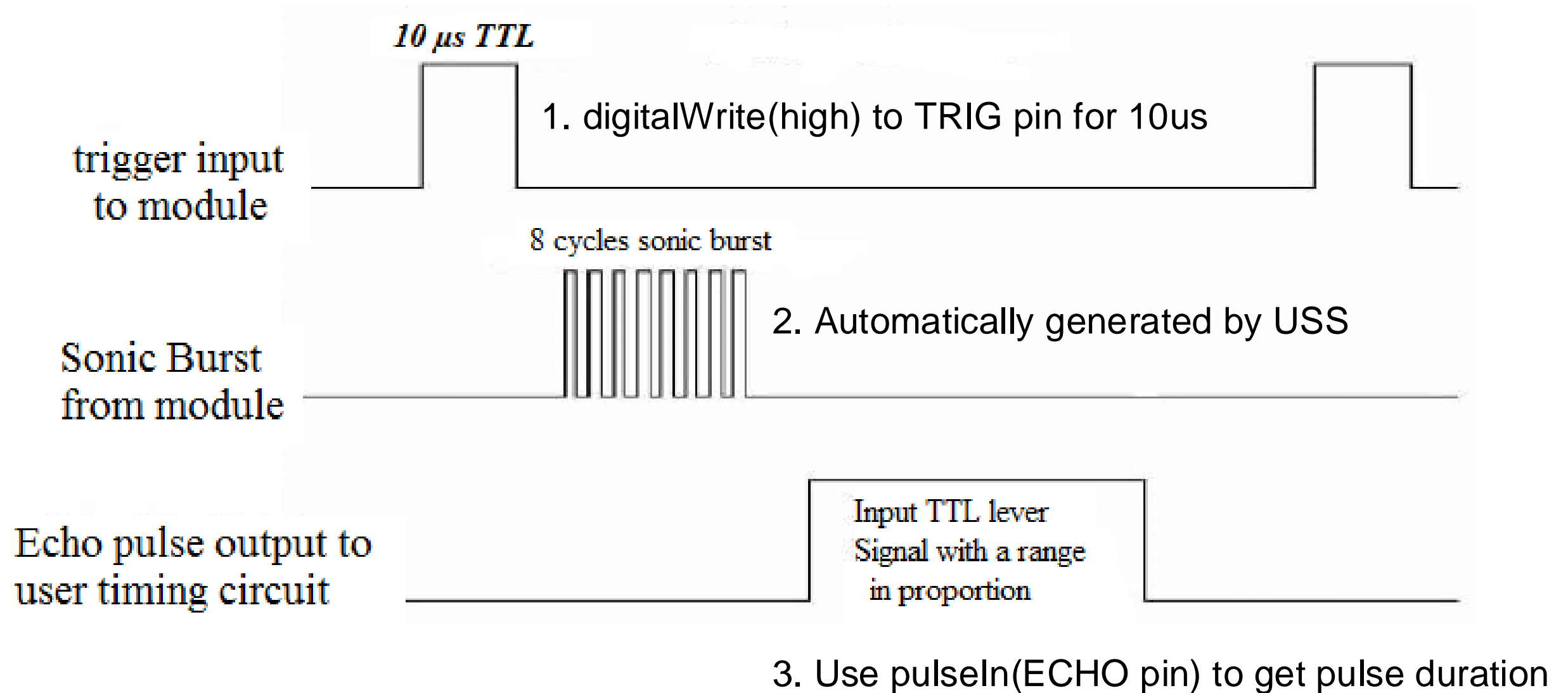
- 동작 원리
 - 특정 파형의 초음파 발신(ping) 후 수신될 때까지의 시간을 측정
 - HC-SR04, SRF05 등
 - Trigger: 초음파 발신 명령 입력
 - Echo: 초음파 발신 직후 0->1로, 수신 직후 1->0으로 출력
 - 시간-거리 변환
 - $\text{Distance (m)} = \text{Time (sec)} \times \text{Speed (m/sec)}$
 - 상온(24°C)에서 음속: 346m/s
 - 거리 $d \text{ (m)} = 346 * t / 2$ (왕복거리 / 2)
 - $d \text{ (mm)} = 173 * t \text{ (ms)}$



SRF05의 핀 배치 (전면)

VCC	TRIG	ECHO	OUT	GND
-----	------	------	-----	-----

초음파센서

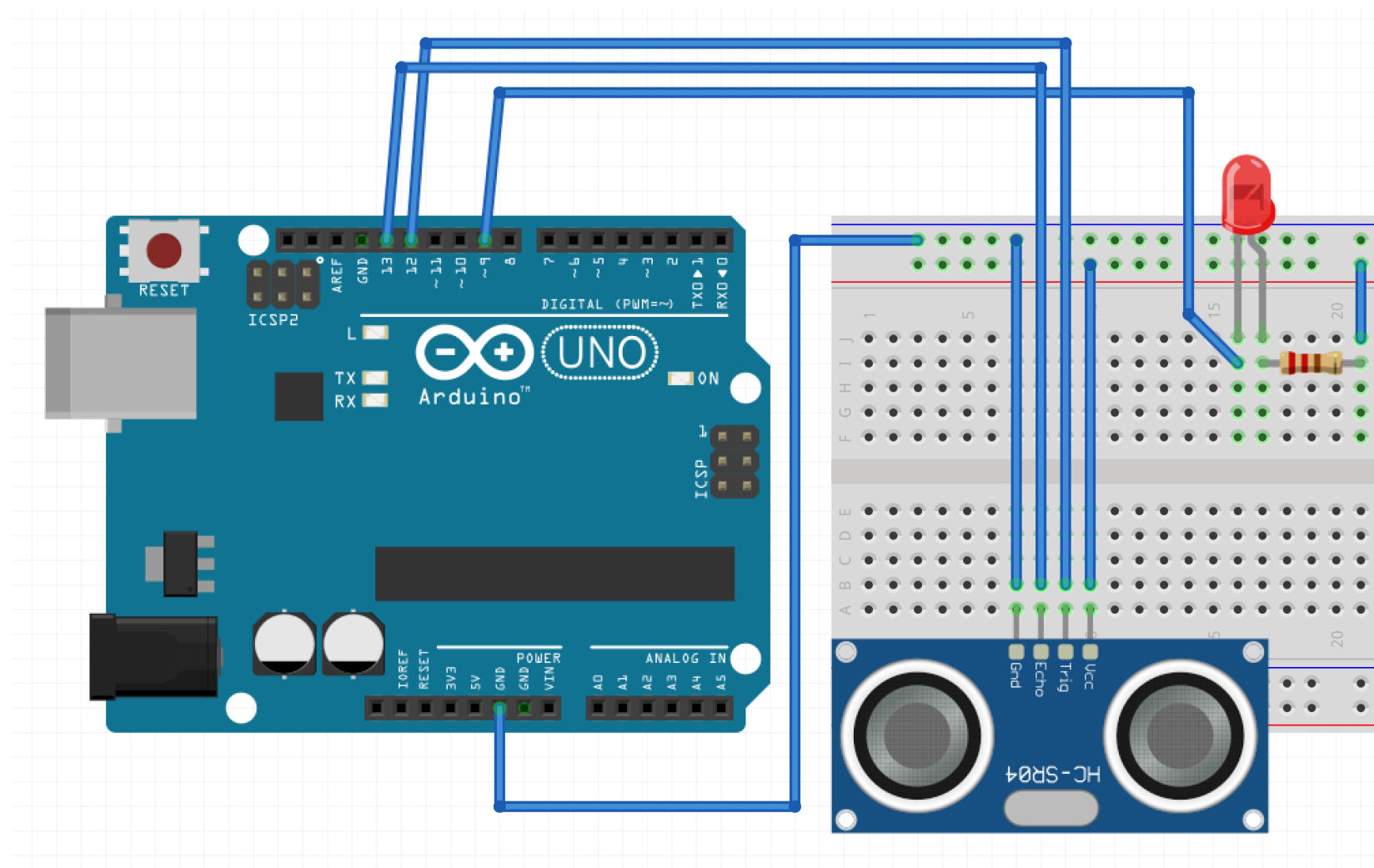


초음파센서

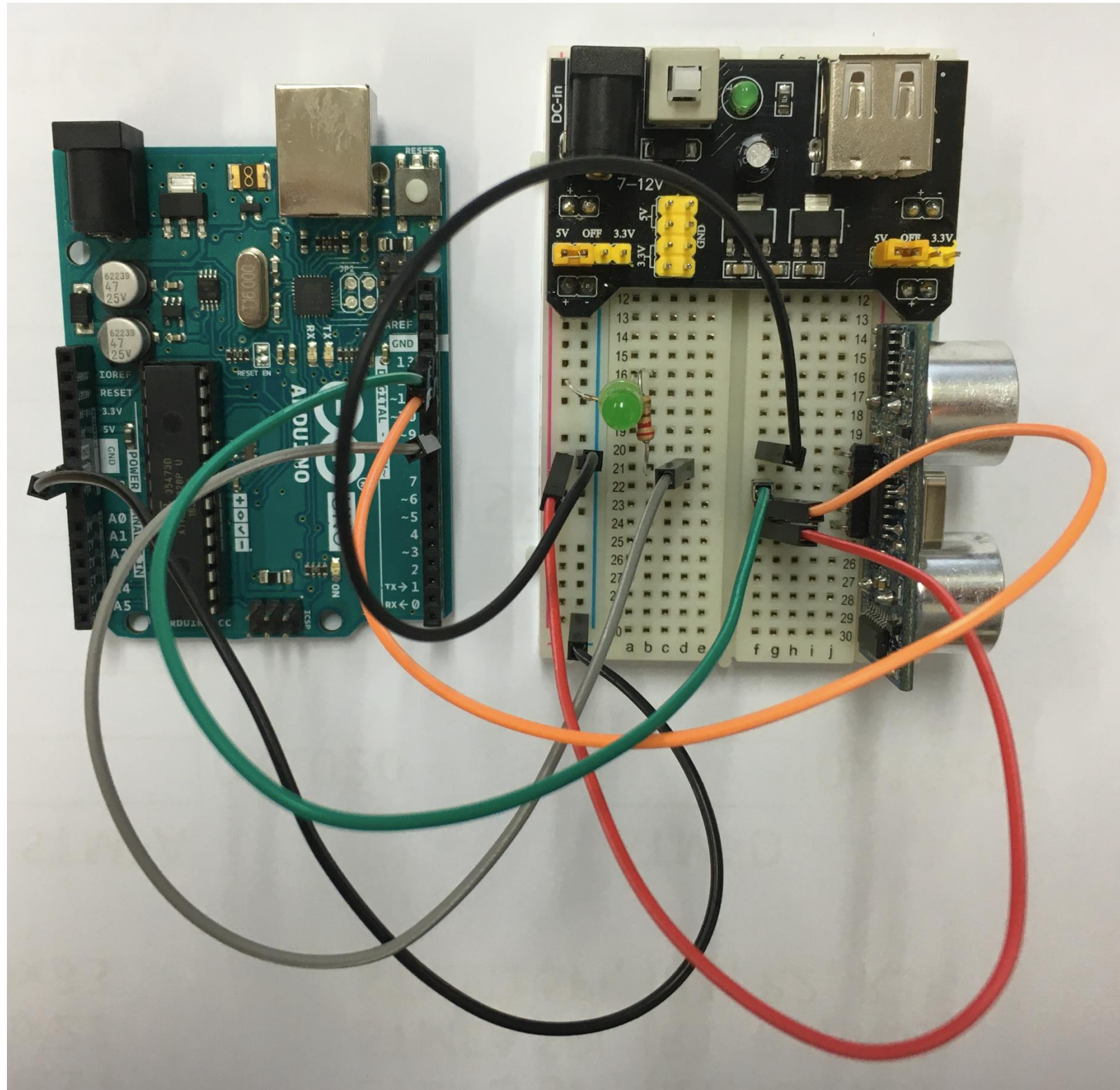
- Timeout 고려
 - 초음파 발신 후 수신이 되지 않는 경우 timeout 발생
 - Echo 핀이 수백 ms에서 수 초까지도 high로 고정됨
 - Timeout 발생시 해당 시간 동안 거리측정 불가
- 측정대상의 모양 및 재질에 민감
 - 단단한 평면에 대해 안정적인 측정 가능
- 최대 측정 주파수
 - 초당 최대 측정 회수가 40회로 제한됨
 - 직전에 발신한 초음파가 사라질 때까지 기다려야 함
 - 측정간격이 지나치게 짧을 경우
 - 이전에 발신된 초음파 수신 -> 실제 거리보다 훨씬 짧게 거리를 인식

실습 1: 초음파센서 동작 실험 (회로구성)

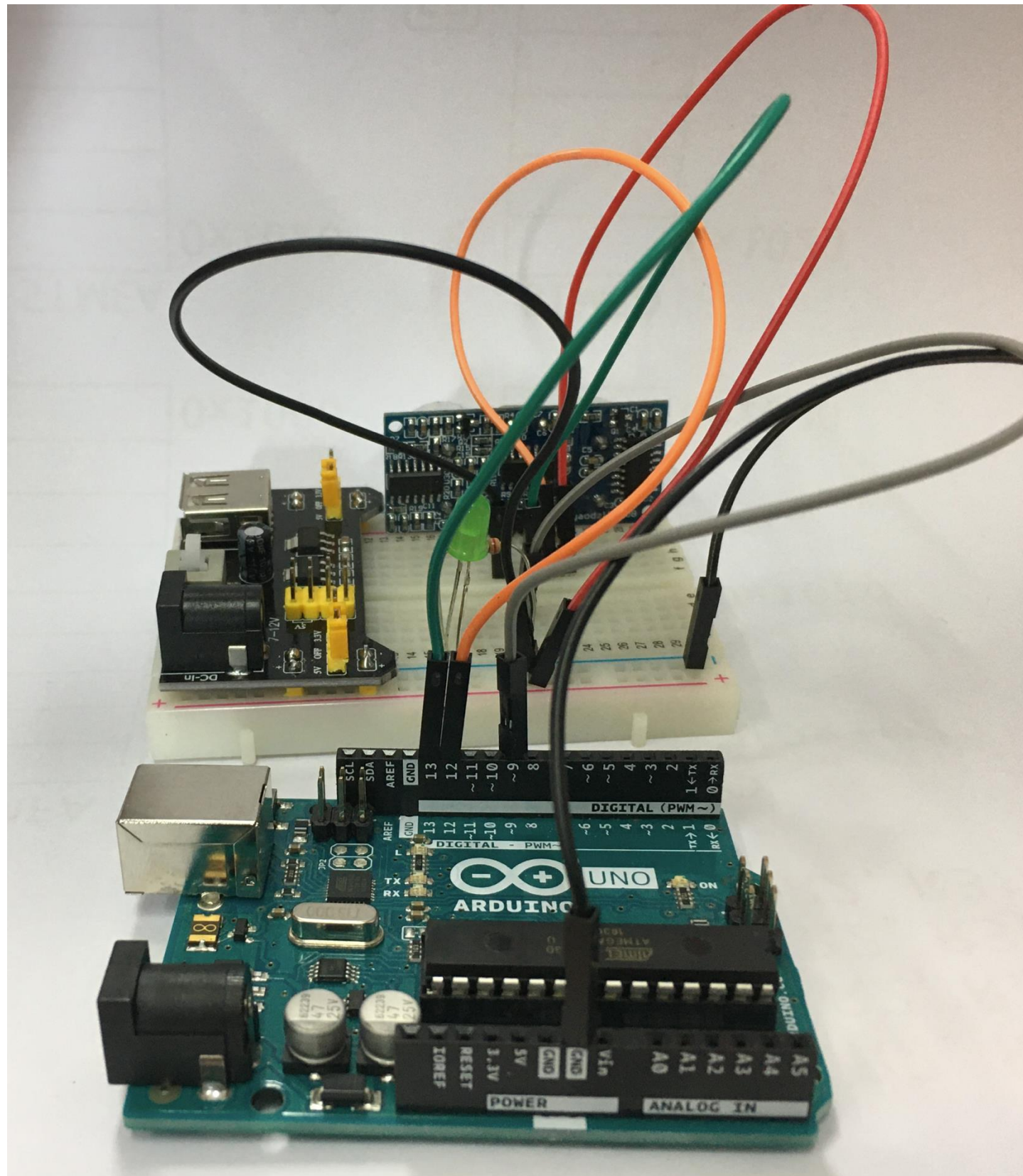
- 회로 구성
 - Pin 연결 : Trig=12, Echo=13, LED=9,
 - 그리고 전원 Vcc(5V), GND연결 (전원 모듈 사용)
- OUT 단자는 비워둠 - 아래 그림은 OUT 단자가 원래 없는 SRF04의 연결 예
- 초음파센서 발신부가 브레드보드 바깥 쪽을 향하도록 설치



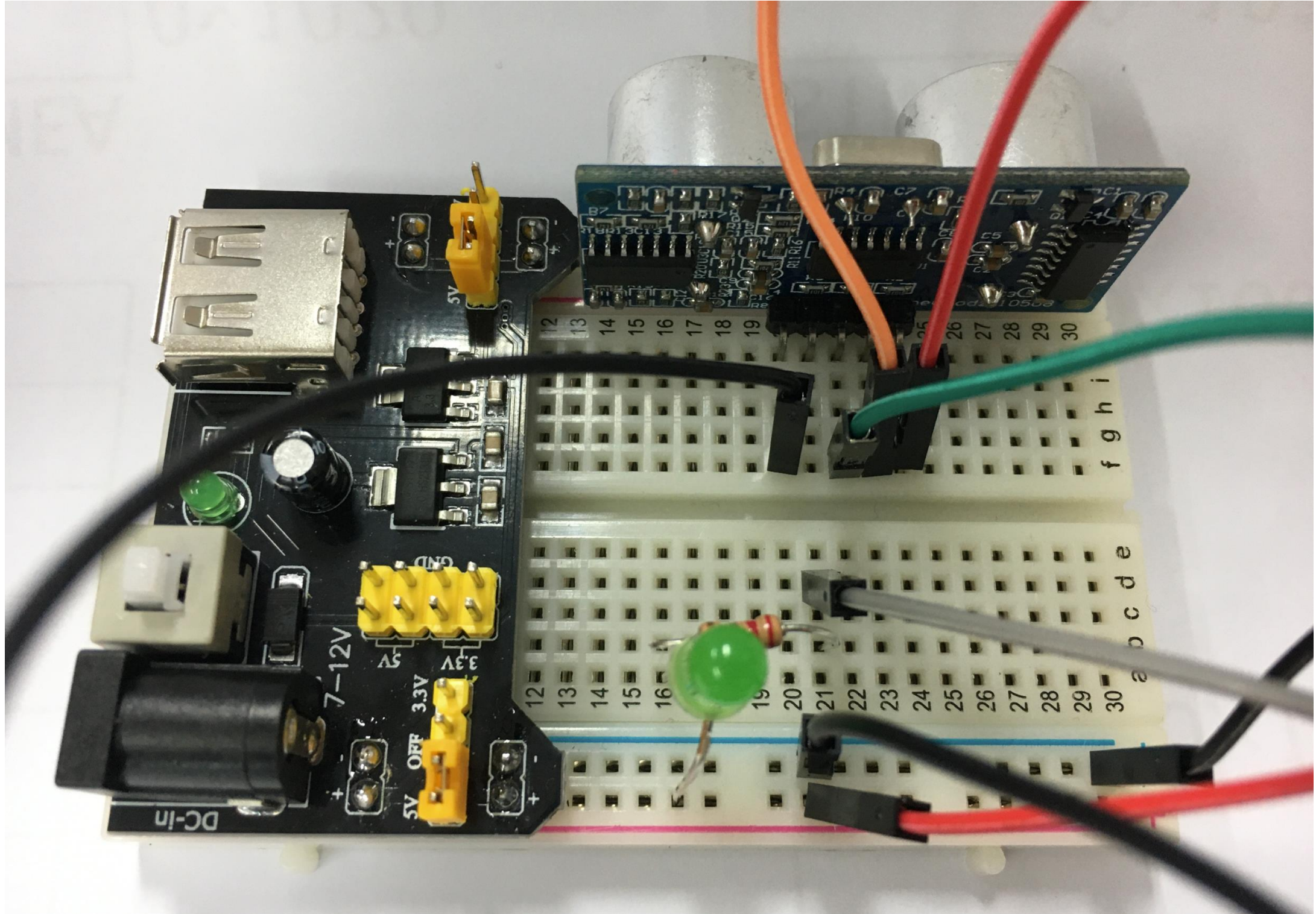
회로 구성 예



회로 구성 예



회로 구성 예

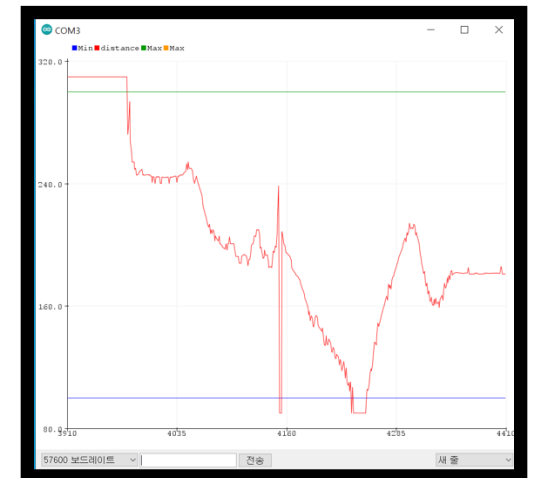


자주 하는 실수

- 전원 모듈에 어댑터를 연결하지 않은 경우
- LED 극성 반대로 연결
- 전원 모듈 점퍼를 3.3V로 연결
- TRIG, ECHO를 서로 반대로 연결
- 초음파센서의 VCC, GND를 브레드보드의 GND, VCC에 연결 (short로 전원모듈 damage)
- 전원 모듈 자체 불량 (교체 요청할 것)
- 아두이노와 브레드보드의 GND끼리 연결하지 않음
- 초음파센서 핀 5개를 브레드보드의 전원 라인에 꽂음
- 측정시 딱딱한 물체(스마트폰 추천) 사용할 것

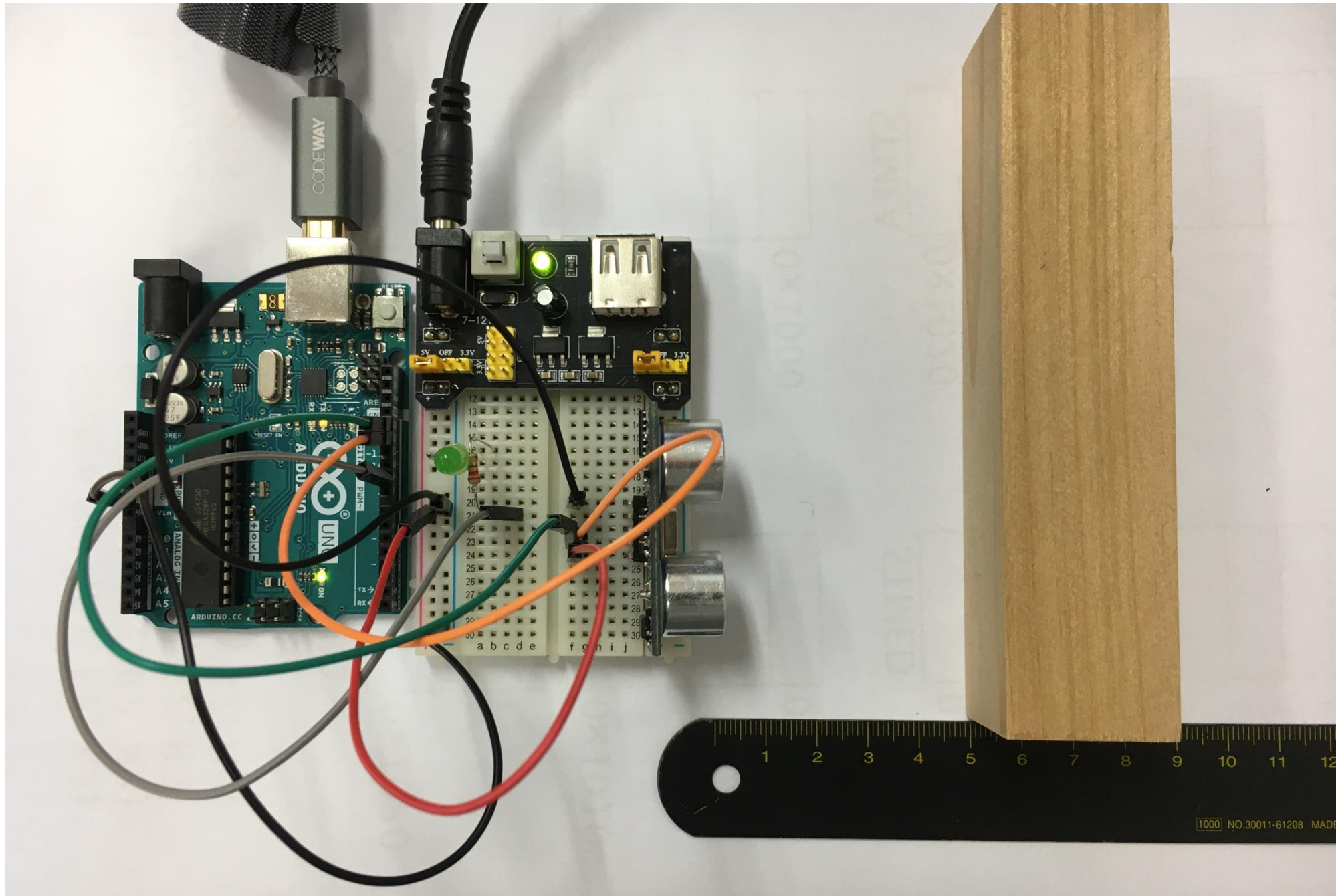
실습 1: 초음파 센서 동작 실험

- 회로 테스트용 코드
 - 07_example_1.ino
 - https://www.dropbox.com/scl/fi/w76h6a6ax0r275lkcl4kr/07_example_1.ino?rlkey=ulauwtga0uevrlsr4kww28592&dl=0
 - 점검 사항
 - 시리얼 모니터에서 거리가 출력되는 것을 확인
 - 측정 대상 물체와 초음파 센서 사이 거리가 10 ~ 30cm 구간에서 LED가 켜지는 것을 확인
- 시리얼 플로터 동작 확인
 - 측정 대상 물체의 재질/형상에 따른 거리 측정 성공률 확인
 - 실패하는 경우 0이 출력됨
 - 딱딱한/부드러운 재질
 - 평면/곡면
 - 손바닥, 안경 닦이 수건, 스마트폰 등



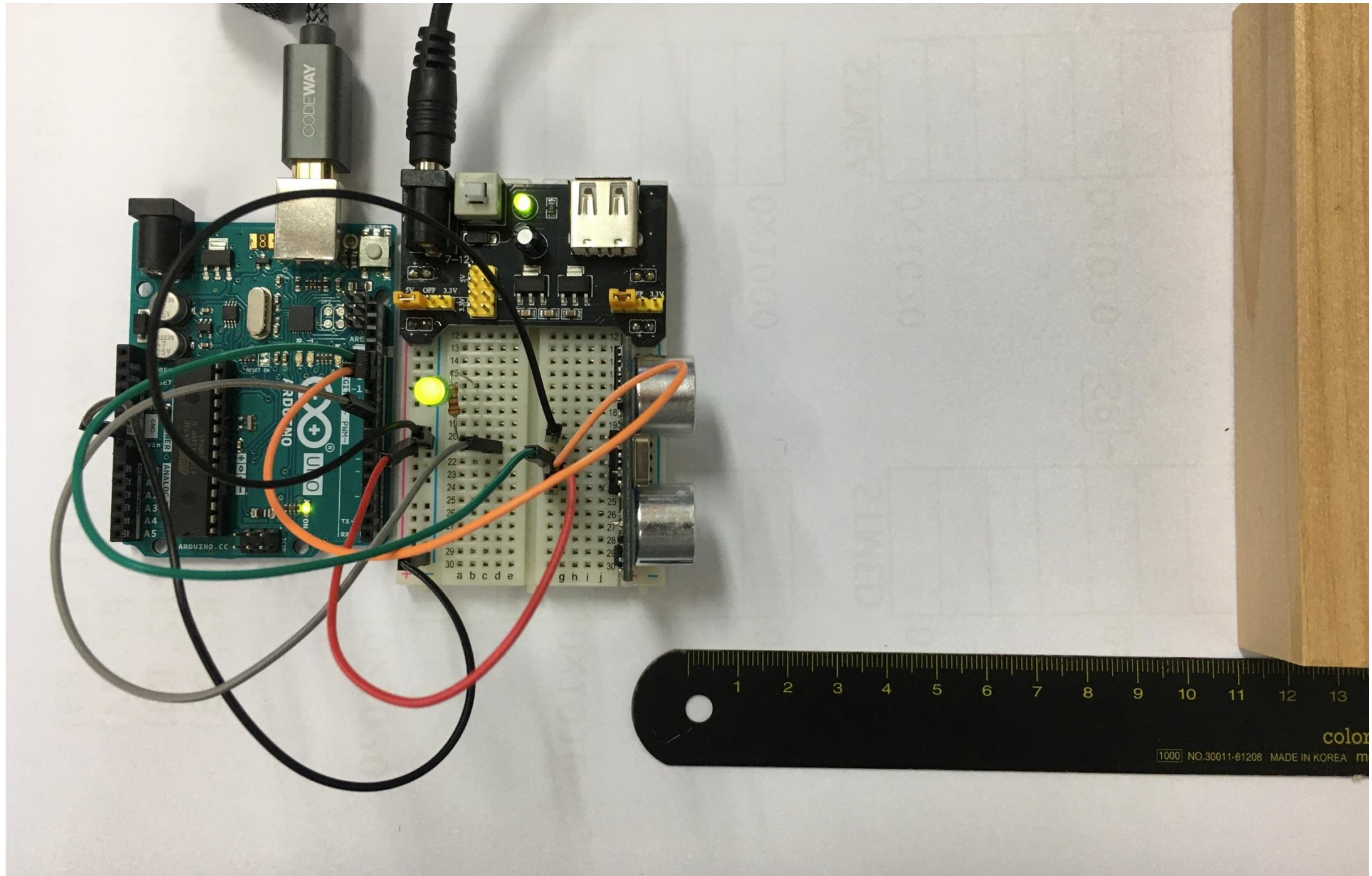
실습 1: 초음파 센서 동작 실험

- 물체와의 거리 5cm



실습 1: 초음파 센서 동작 실험

- 물체와의 거리 11cm



실습 코드 - 상수 설정 및 초기화

```
1 // Arduino pin assignment
2 #define PIN_LED 9
3 #define PIN_TRIG 12 // sonar sensor TRIGGER
4 #define PIN_ECHO 13 // sonar sensor ECHO
5
6 // configurable parameters
7 #define SND_VEL 346.0 // sound velocity at 24 celsius degree (unit: m/sec)
8 #define INTERVAL 100 // sampling interval (unit: msec)
9 #define PULSE_DURATION 10 // ultra-sound Pulse Duration (unit: usec)
10 #define _DIST_MIN 100.0 // minimum distance to be measured (unit: mm)
11 #define _DIST_MAX 300.0 // maximum distance to be measured (unit: mm)
12
13 #define TIMEOUT ((INTERVAL / 2) * 1000.0) // maximum echo waiting time (unit: usec)
14 #define SCALE (0.001 * 0.5 * SND_VEL) // coefficent to convert duration to distance
15
16 void setup() {
17     // initialize GPIO pins
18     pinMode(PIN_LED, OUTPUT);
19     pinMode(PIN_TRIG, OUTPUT); // sonar TRIGGER
20     pinMode(PIN_ECHO, INPUT); // sonar ECHO
21     digitalWrite(PIN_TRIG, LOW); // turn-off Sonar
22
23     // initialize serial port
24     Serial.begin(57600);
25 }
```


실습 코드 - 상수 설정 및 초기화

- 상수 선언 들
 - Line 2-4: 아두이노 핀 지정
 - Pin 9: LED 연결, Active low (0일 때 동작) 회로
 - Pin 12, 13: 초음파 센서의 TRIG, ECHO 단자에 연결
 - Line 7-11: 각종 설정값
 - **SND_VEL**: 음속, unit: m/sec
 - **INTERVAL**: 초음파 센서 측정 주기, unit: msec
 - **PULSE_DURATION**: 초음파 송출 시간, unit: usec
 - **_DIST_MIN, _DIST_MAX**: 측정 범위, unit: mm
 - Line 13-14: 측정 대기 시간 및 거리 계산 식
 - **TIMEOUT** : 초음파 펄스 echo 대기 최대 시간
 - **SCALE** : 시간 → 거리 환산 계수
- setup() 함수
 - Line 18-21 아두이노 입출력 핀 모드 설정
 - 초음파 센서 TRIG 출력은 0으로 초기화
 - Line 24: 시리얼포트 속도 설정 (57,600bps 임에 주의!)
 - 통신 에러가 발생하지 않는 한도 내에서 높은 또는 적당한 속도로 설정

실습 코드 - loop()

```
27 void loop() {
28     float distance = USS_measure(PIN_TRIG, PIN_ECHO); // read distance
29
30     if ((distance == 0.0) || (distance > _DIST_MAX)) {
31         distance = _DIST_MAX + 10.0; // Set Higher Value
32         digitalWrite(PIN_LED, 1); // LED OFF
33     } else if (distance < _DIST_MIN) {
34         distance = _DIST_MIN - 10.0; // Set Lower Value
35         digitalWrite(PIN_LED, 1); // LED OFF
36     } else { // In desired Range
37         digitalWrite(PIN_LED, 0); // LED ON
38     }
39
40     // output the distance to the serial port
41     Serial.print("Min:"); Serial.print(_DIST_MIN);
42     Serial.print(",distance:"); Serial.print(distance);
43     Serial.print(",Max:"); Serial.print(_DIST_MAX);
44     Serial.println("");
45
46     // do something here
47     delay(50); // Assume that it takes 50ms to do something.
48
49     // wait until next sampling time.
50     delay(INTERVAL);
51 }
```

실습 코드 - loop()

- **loop()** 함수 - 반복 실행
 - Line 28: 거리 측정 함수로부터 측정된 거리를 로컬 변수로 읽어옴
 - Line 30-38: 측정 결과를 조정
 - 지정된 범위 밖이면 범위 밖의 값으로 강제 조정하고 LED OFF
 - 지정된 범위 안쪽이면 LED ON
 - Line 41-44: 측정 결과를 (시리얼 플로터 표시를 위해) 시리얼포트로 출력
 - 시리얼 플로터 도구의 입력 형식에 맞도록 min, max와 함께 출력
 - 시리얼 플로터 도구용 메시지 규칙:
 - 값들은 comma(,)로 구분해야하고, 공백이 없어야 함
 - Line 30-38에서 결과를 범위에 맞춰 조정하는 이유는 y축 auto-scale 방지용임
 - 시리얼 플로터 동작 (시리얼 모니터 꺼야함)
 - Line 47: 측정 결과로 다른 일을 한다고 가정하여 시간 때우기
 - Line 50: 다음 거리 측정까지 대기

실습 코드 - 거리측정

```
53 // get a distance reading from USS. return value is in millimeter.
54 float USS_measure(int TRIG, int ECHO)
55 {
56     digitalWrite(TRIG, HIGH);
57     delayMicroseconds(PULSE_DURATION);
58     digitalWrite(TRIG, LOW);
59
60     return pulseIn(ECHO, HIGH, TIMEOUT) * SCALE; // unit: mm
61
62     // Pulse duration to distance conversion example (target distance = 17.3m)
63     // - pulseIn(ECHO, HIGH, timeout) returns microseconds (음파의 왕복 시간)
64     // - 편도 거리 = (pulseIn() / 1,000,000) * SND_VEL / 2 (미터 단위)
65     //   mm 단위로 하려면 * 1,000이 필요 ==> SCALE = 0.001 * 0.5 * SND_VEL
66     //
67     // - 예, pusseIn()이 100,000 이면 (= 0.1초, 왕복 거리 34.6m)
68     //       = 100,000 micro*sec * 0.001 milli/micro * 0.5 * 346 meter/sec
69     //       = 100,000 * 0.001 * 0.5 * 346
70     //       = 17,300 mm ==> 17.3m
71 }
```

실습 코드 - 거리 측정

- Line 56: 초음파 센서에 거리 측정을 위한 초음파 발신 요청 전송
 - TRIG 신호에 10us 동안 1을 출력 후 다시 0으로 복귀
- Line 60: 초음파가 도달한 시간으로 물체와의 거리 계산
 - **pulseIn()** 함수
 - <https://www.arduino.cc/reference/ko/language/functions/advanced-io/pulsein/>
 - 지정하는 핀이 지정한 값을 유지하는 시간(즉, pulse 길이)을 usec 단위로 측정
 - 우리 경우 : **pulseIn(PIN_ECHO, HIGH, TIMEOUT)** 으로 호출
 - ECHO pin이 1이 될 때까지 기다린 후, 그때부터 다시 0이 될 때까지의 시간을 측정
 - 그 시간이 초음파가 출발해서 어딘가에 부딪혀 돌아오는 왕복 시간
 - **TIMEOUT** 만큼의 시간이 지나도 ECHO pin이 0으로 돌아오지 않으면, **pulseIn()** 은 0을 리턴하면서 강제 종료됨 (즉, 물체가 너무 멀리 있다는..)
- Line 62-71: 시간으로부터 거리를 계산하는 방법 설명
 - **#define SCALE (0.001 * 0.5 * SND_VEL)**
 - 0.001은 usec을 msec으로 바꾸고
 - 0.5는 측정된 시간은 초음파 왕복 시간이므로 반으로 나누어 편도 시간으로 변경하기 위한 값
 - 결국 SCALE를 곱하면 mm 단위의 편도 거리가 계산됨