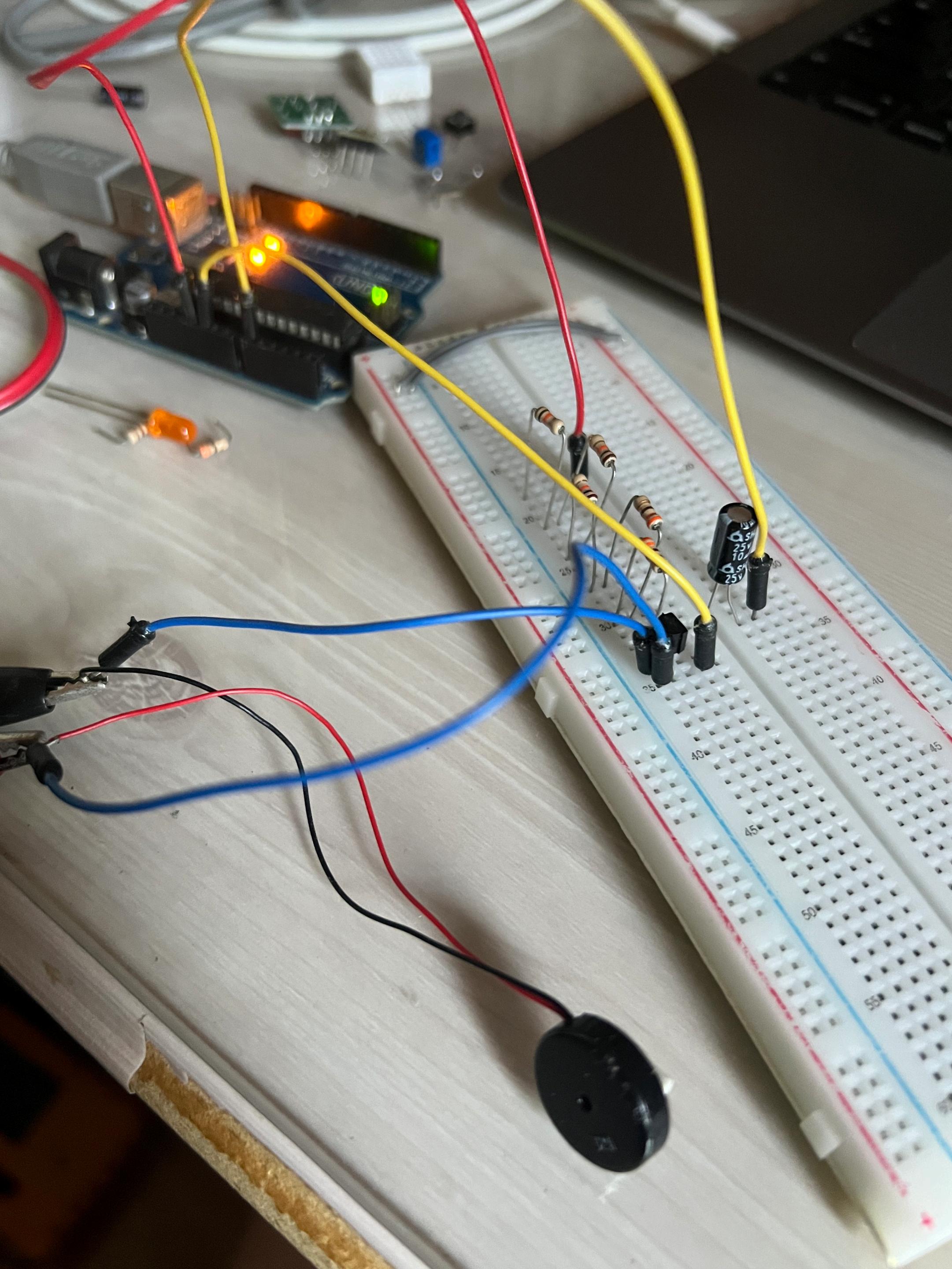
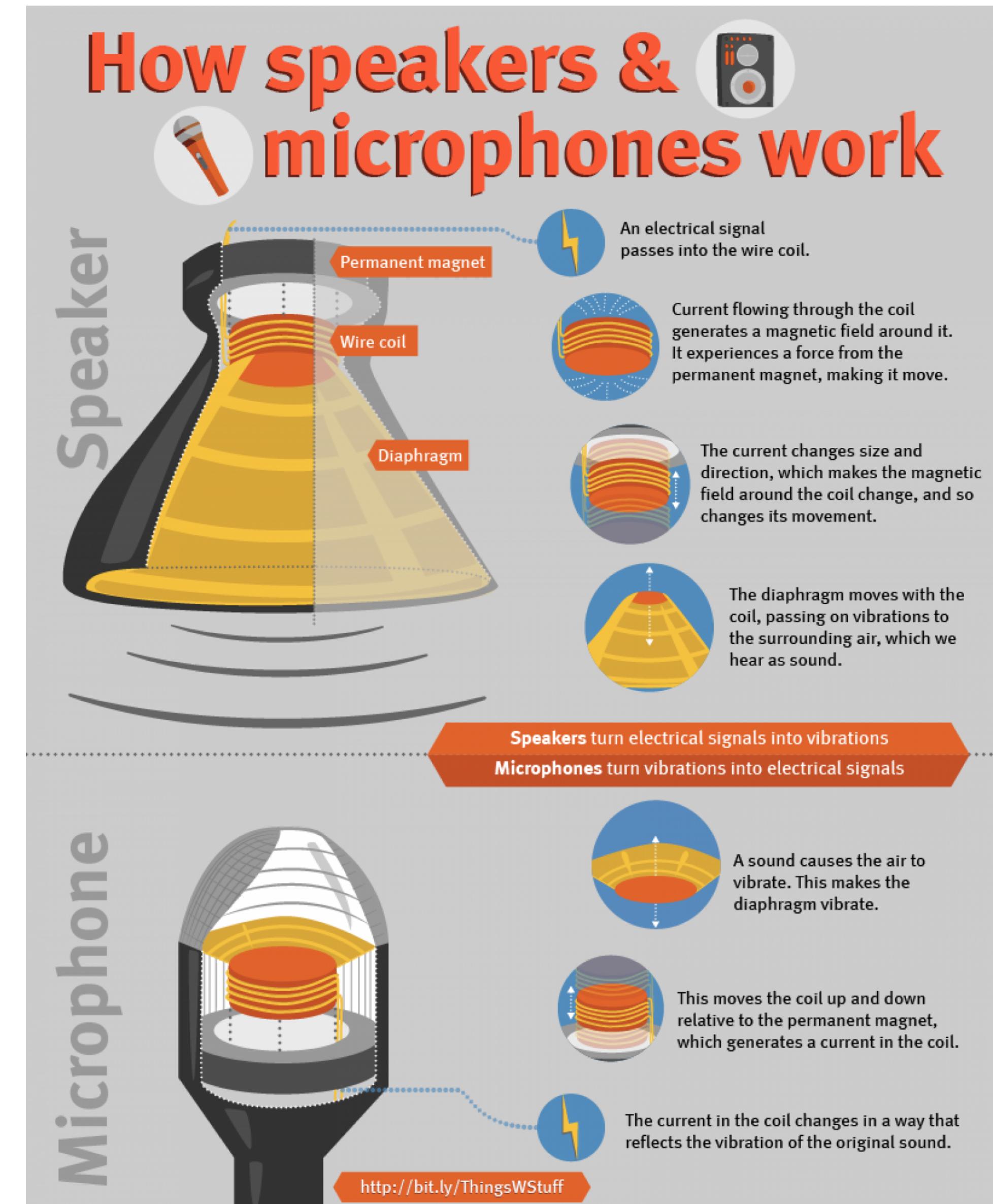


아두이노로 マイ크 샘플링해서 녹음하기

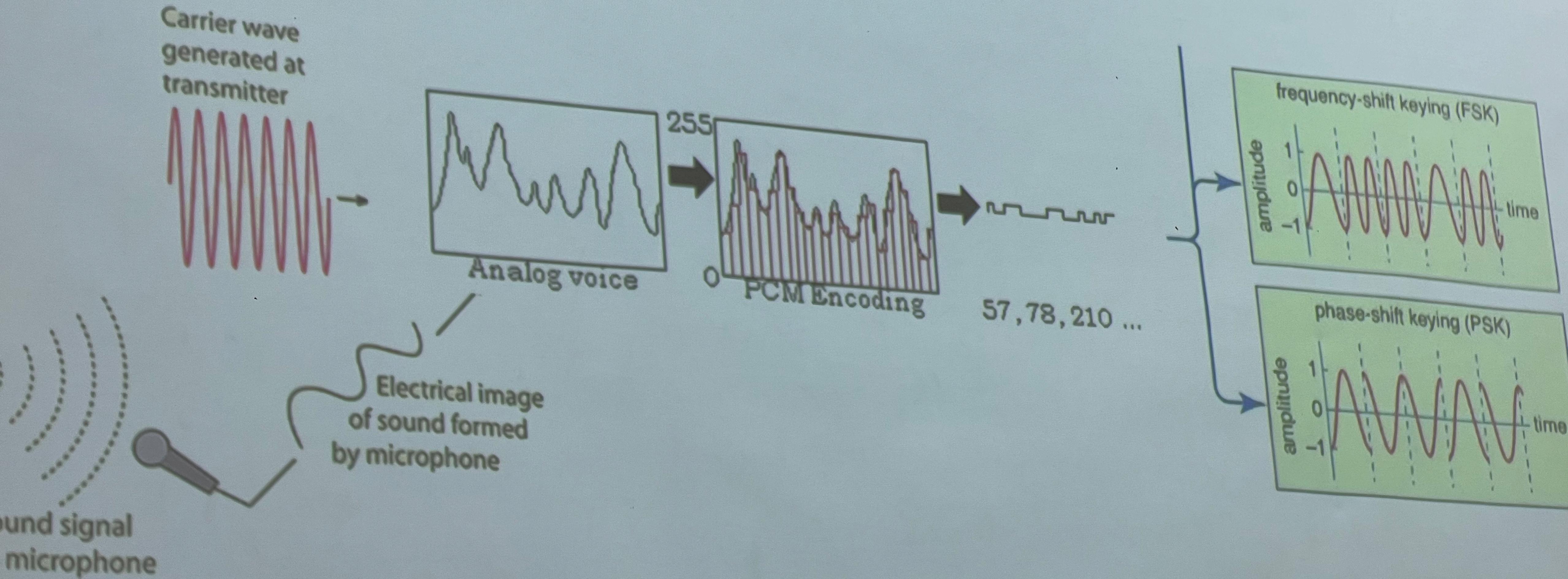
조영호



왜?

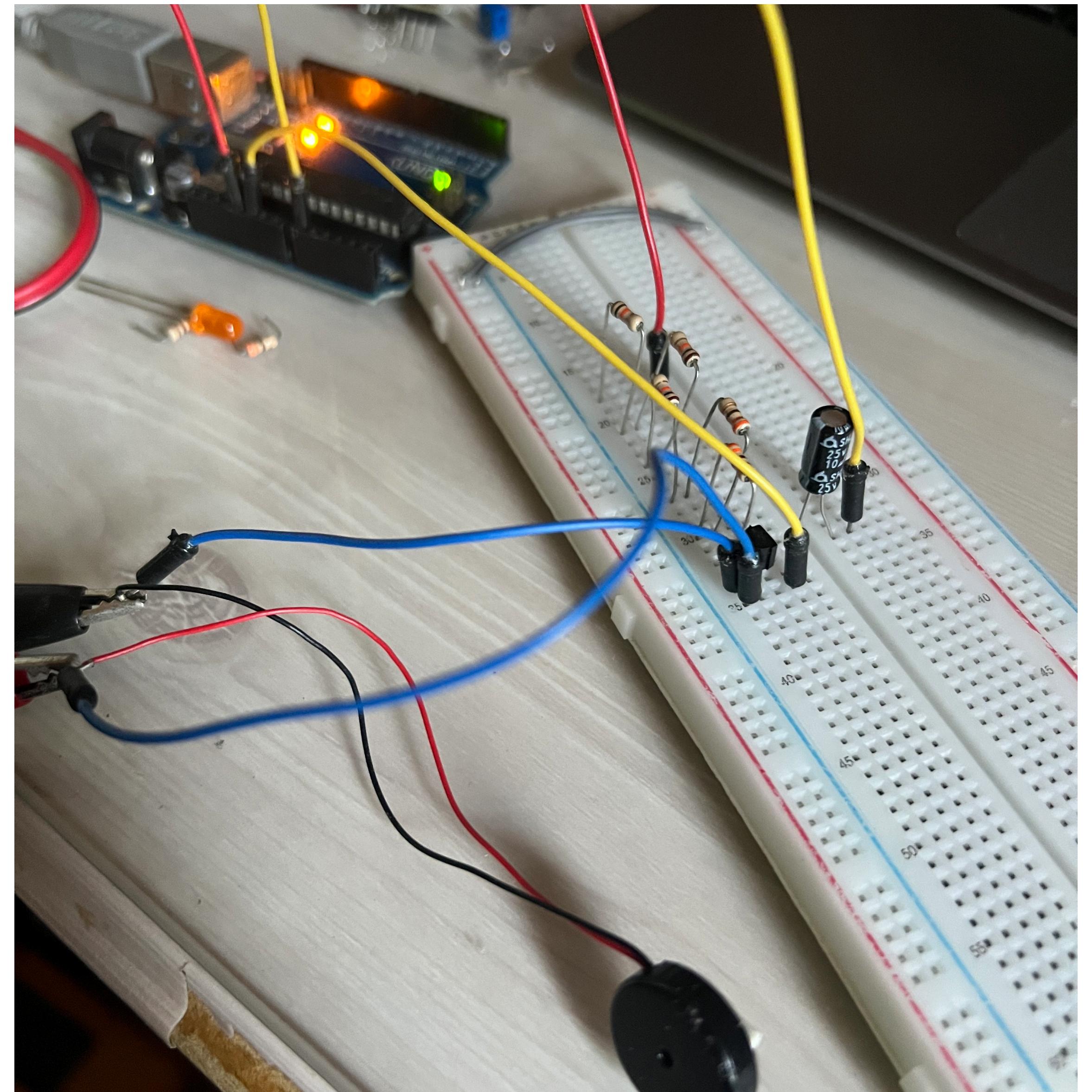


디지털 전송



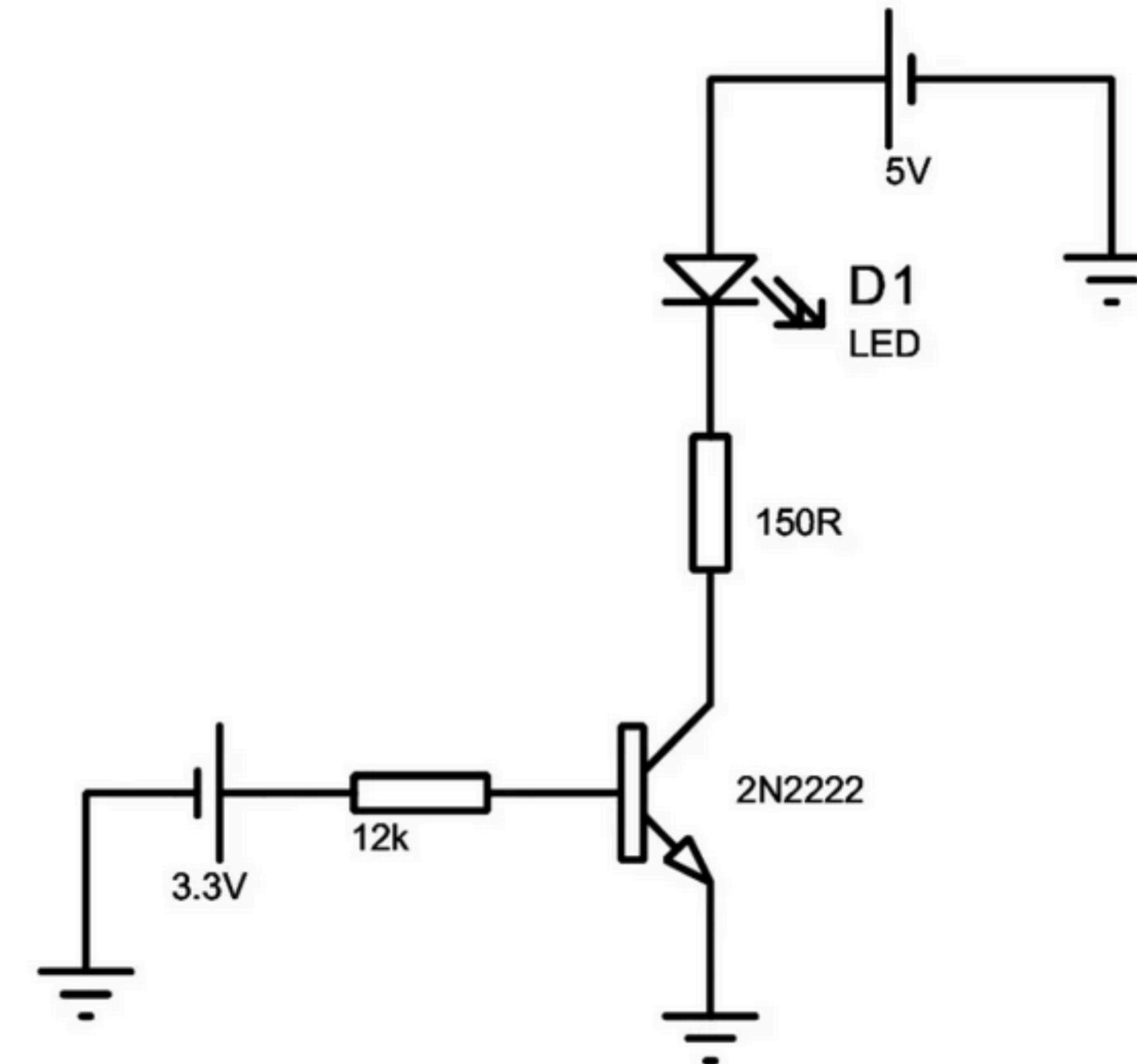
개발 회로

TMI: 전자전기 복전 신청했다가 접음



개발 회로

2n2222 transistor as a switch



2n2222 transistor as a switch

<https://www.npntransistors.com/2n2222-transistor-datasheet/>

개발

코드

~~delay(1);~~

~~analogRead(A0);~~

The screenshot shows a terminal window titled "nvim asdf.ino" displaying Arduino C++ code. The code initializes serial communication at 115200 baud, sets up Timer 0 for 8000Hz, initializes the ADC, and defines a function to read the ADC value. The code uses standard Arduino libraries and syntax.

```
1 void setup() {
1   Serial.begin(115200);
2
3   OCR0A = 0xF9; // 249
4   TCCR0B = (1 << WGM01) | (1 << CS01);
5   TIMSK0 = (1 << OCIE1A);
6
7   initADC();
8 }
9
10 void loop() {
11 }
12
13 ISR(TIMER0_COMPA_vect) { // 8000hz
14   Serial.write(readADC());
15 }
16
17 void initADC() {
18   ADMUX |= (1 << REFS0) | (1 << ADLAR); // ADLAR -> ADCH, ADCL 왼쪽 정렬
19 }
20
21 byte readADC() {
22   ADCSRA |= (1 << ADSC); // start conversion
23   while(ADCSRA & (1 << ADIF)); // interrupt flag <- when conversion completed
24
25   uint8_t high;
26   high = ADCH; // read only 1 byte
27
28   return high;
29 }
```

At the bottom of the terminal window, there are status indicators: "NORMAL" with a right arrow, "asdf.ino" with a left arrow, and a series of command-line navigation keys: "gk < ⌂ 20 ⌄ Top 1:1 ⌄ 12:4".

개발

ADC Datasheet

- ATmega328 ADC 분해능: 10bit

- 8bit / 8kHz

Youtube: 16bit, 24bit / 44.1 ~ 96kHz

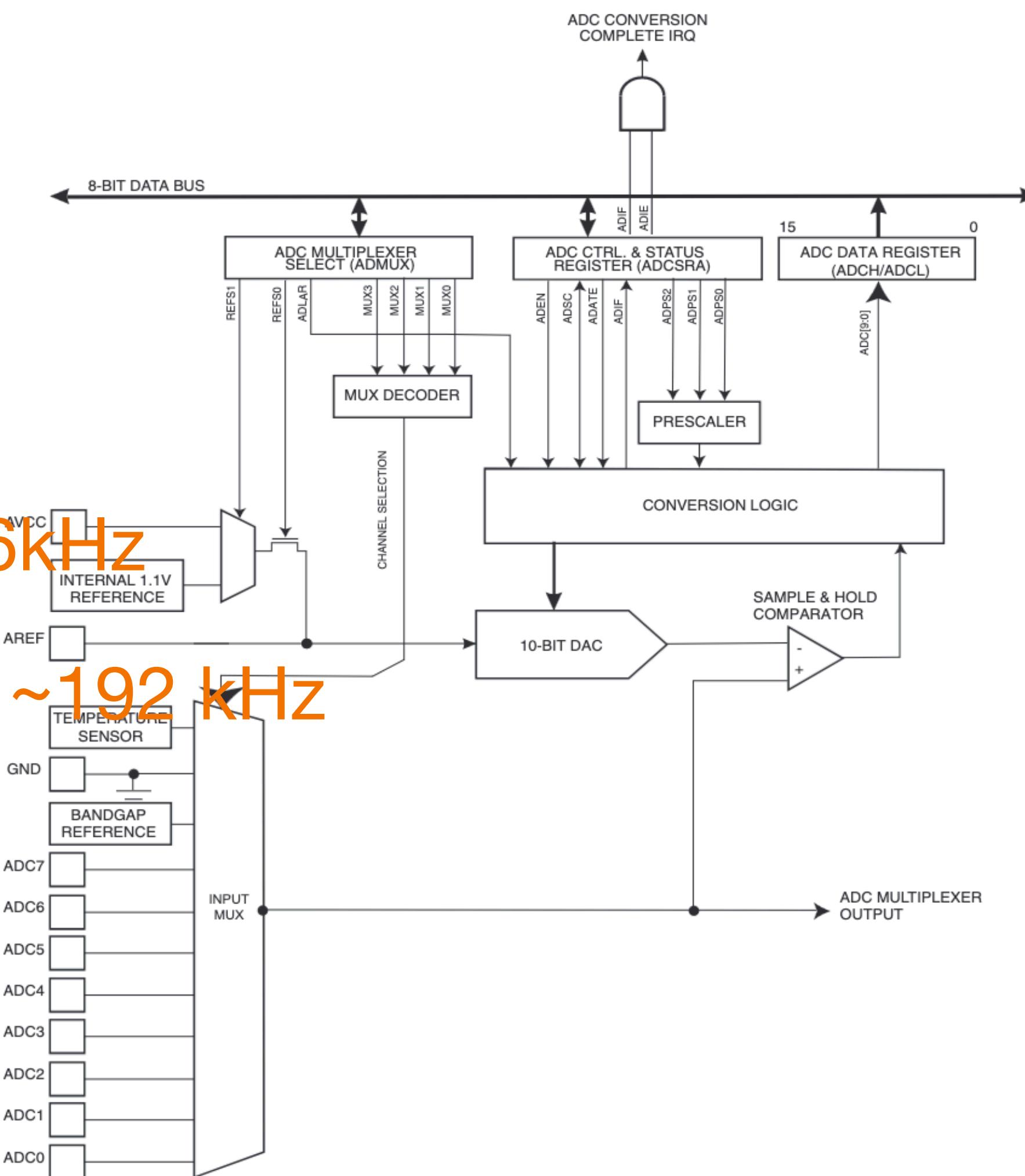
Apple music: 16bit, 24bit / 44.1 ~ 192 kHz

24.9.3.2 ADLAR = 1

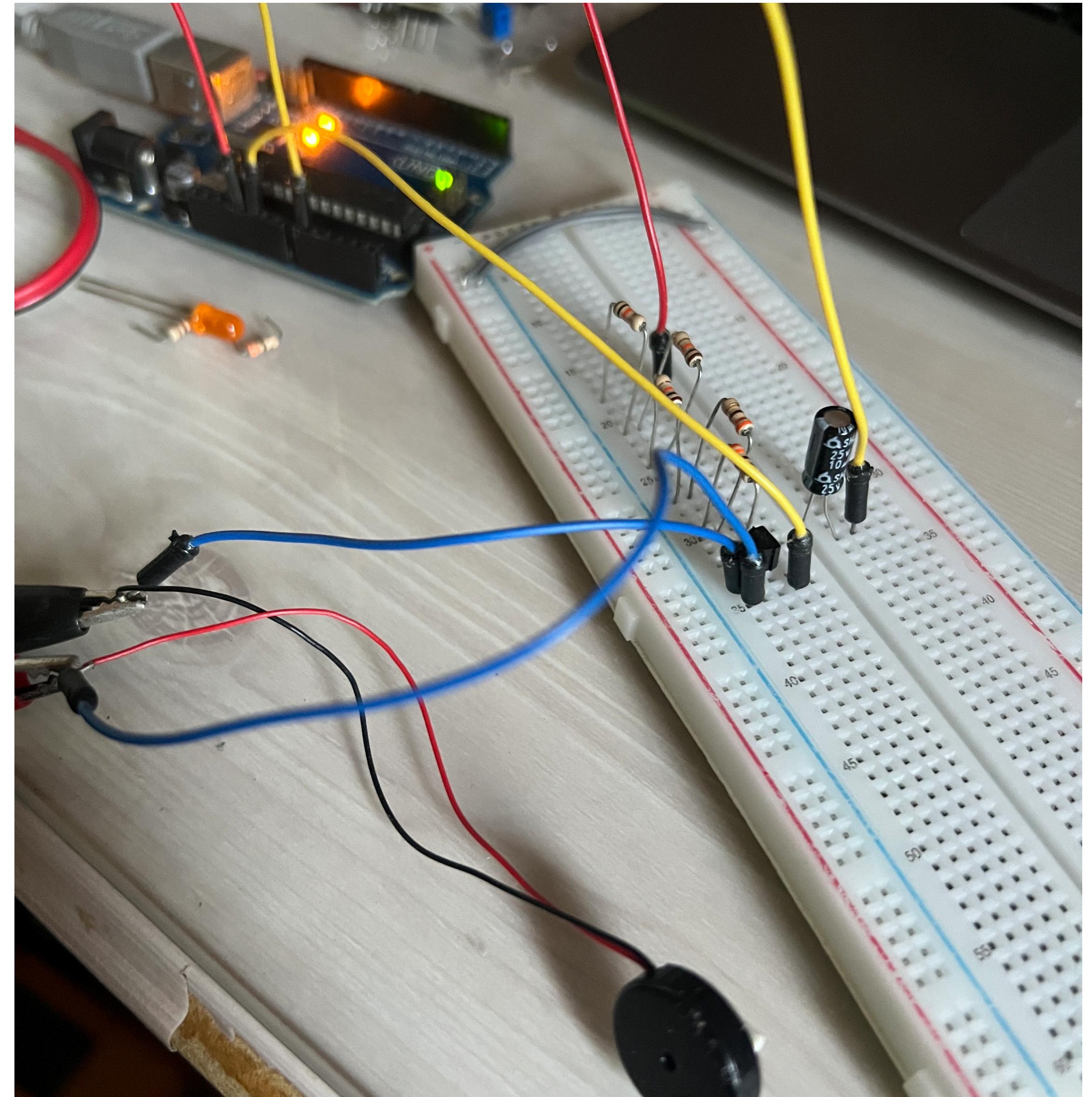
Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

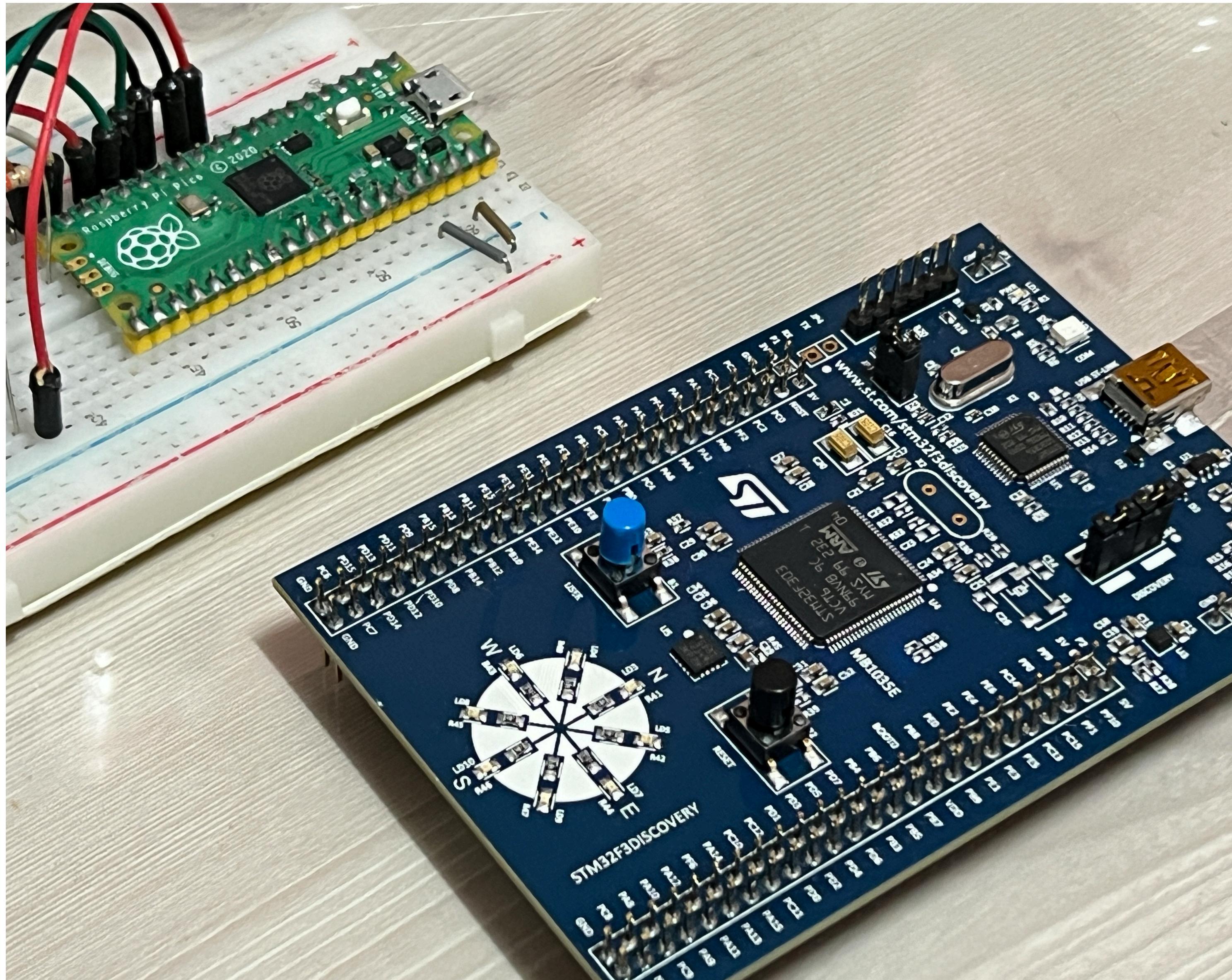
When an ADC conversion is complete, the result is found in these two registers.

Figure 24-1. Analog to Digital Converter Block Schematic Operation



결과





- 더 높은 분해능 (12bit)
- 더 높은 샘플링 레이트 (44kHz)
- 더 좋은 AMP
- 더 좋은 mic소자

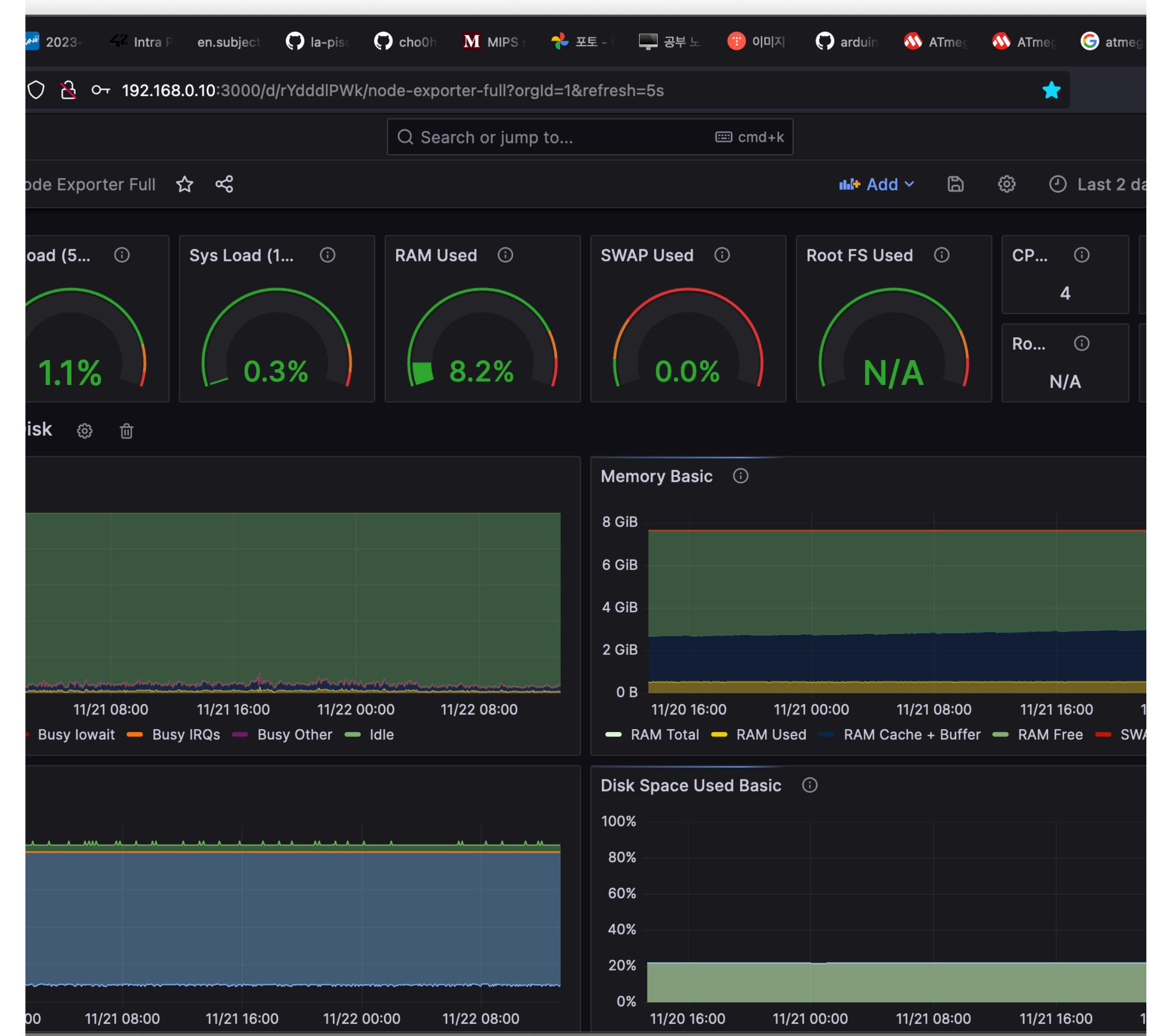


감사합니다..



개요

Grafana란?



개요

시계열 DB?

Add data source

Choose a data source type

Q Filter by name or type

Time series databases



Prometheus

Open source time series database & alerting

Core



Graphite

Open source time series database

Core



InfluxDB

Open source time series database

Core



OpenTSDB

Open source time series database

Core

InfluxDB

write

```
younghoc@joyeonghos-MacBook-Air-2:~/tmp/influx2
```

```
write.sh
```

```
:8087/api/v2/write?org=my-org&bucket=my-bucket" \
ion: Token mytokennnnnnnn" \
pe: text/plain; charset=utf-8" \
blication/json" \
ture=25.5,humidity=42.0
```

InfluxDB

query

```
younghoc@joyeonghos-MacBook-Air-2:~/tmp/influx2
```

```
query2.sh
```

```
http://192.168.0.10:8087/api/v2/query?org=my-org' \
  'Content-Type: application/vnd.flux' \
  'Content-Type: application/csv' \
  'Authorization: Token mytokennnnnnnnnn' \
  'Bucket: "my-bucket") \
  'TimeRange: -24h)
  (r) => r._measurement == "weather-test")'
```

InfluxDB

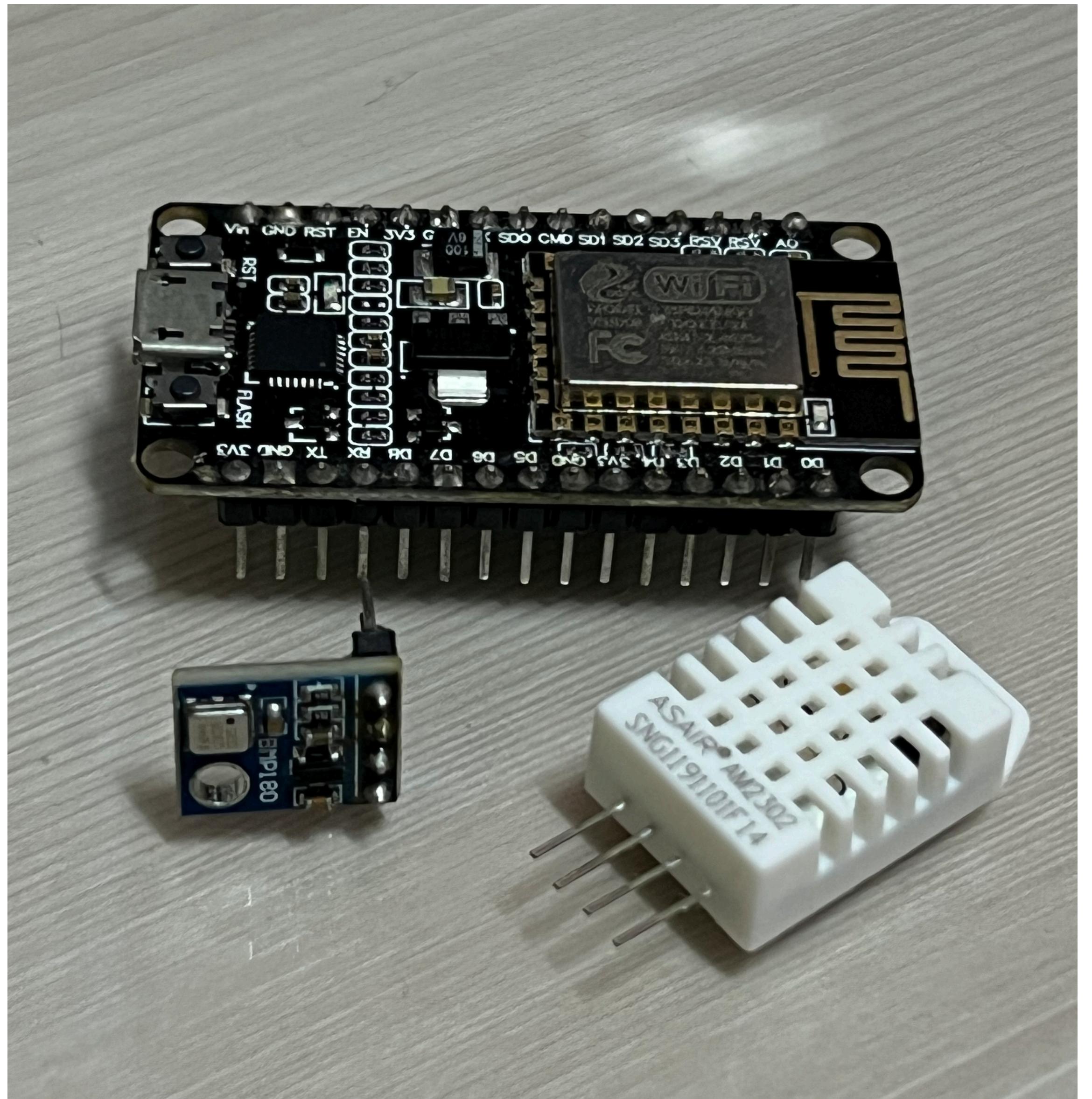
설치 (docker compose)



```
pi@raspberrypi: ~/tmp
@raspberrypi:~/tmp $ cat docker-compose.yml
version: "3.9"
services:
  weather-influx:
    image: influxdb:latest
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=my-userrrrrr
      - DOCKER_INFLUXDB_INIT_PASSWORD=my-passwordddddd
      - DOCKER_INFLUXDB_INIT_ORG=my-org
      - DOCKER_INFLUXDB_INIT_BUCKET=my-bucket
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=mytokennnnn
    volumes:
      - ./config:/etc/influxdb2
      - weather-influx:/var/lib/influxdb2
    ports:
      - 8086:8086
    restart: unless-stopped
volumes:
  weather-influx:
@raspberrypi:~/tmp $
```

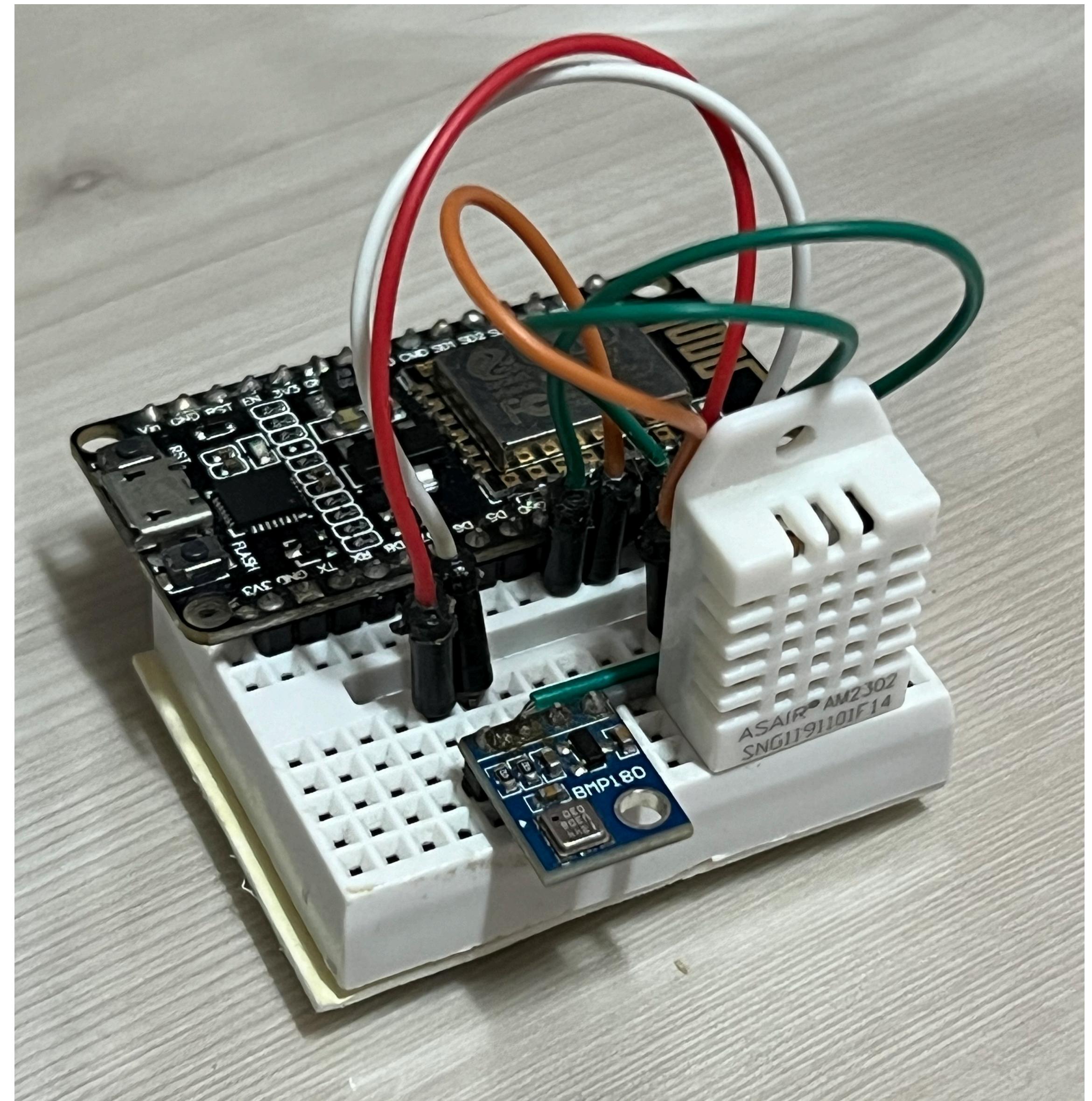
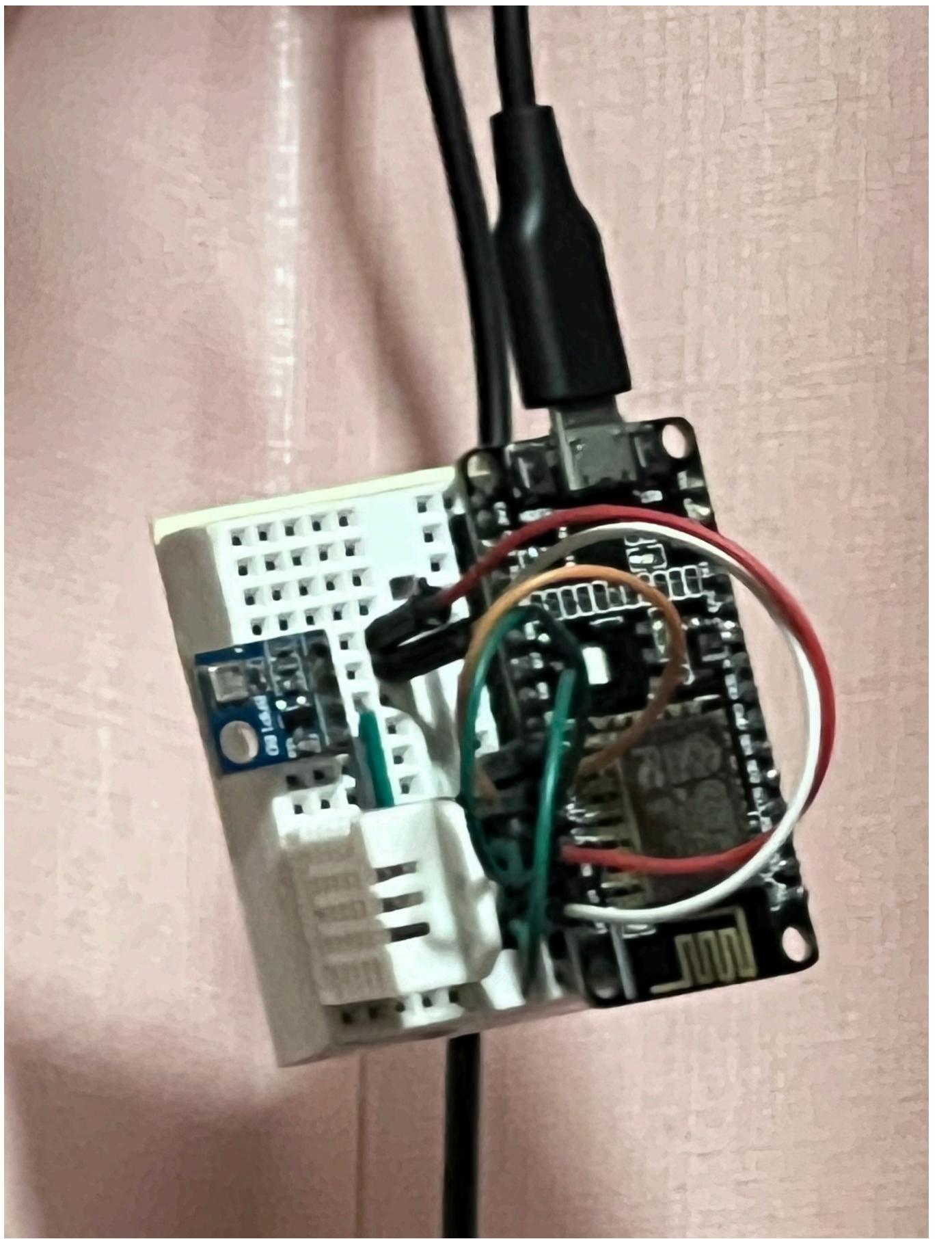
하드웨어 부품

- 온도, 습도 (DHT22)
- 온도, 기압 (BMP180)



하드웨어

조립, 설치



소프트웨어

arduino code (생략 버전)



The screenshot shows a terminal window titled "nvim main.ino" displaying Arduino C code. The code is organized into three main sections: sensor measurement functions, an HTTP upload function, and the main loop. The code uses the InfluxDB library for data transmission.

```
1 void measure_bmp180() {
1   data.bmp180_temperature = bmp180.getTemperature();
2   data.bmp180_pressure = bmp180.getPressure();
3 }
4
5 void measure_dht22() {
6   data.dht22_temperature = event.temperature;
7   data.dht22_humidity = event.relative_humidity;
8 }
9
10 void upload(String content) {
11   http.addHeader("Content-Type", "application/vnd.flux");
12   http.addHeader("Accept", "application/csv");
13   http.addHeader("Authorization", INFLUXDB_TOKEN);
14   http.POST(content);
15 }
16
17 void loop() {
18   delay(delayMS);
19
20   measure_bmp180();
21   measure_dht22();
22
23   String content;
24   content += "weather,sensor=bmp180 temperature=";
25   content += data.bmp180_temperature;
26   content += ",pressure=";
27   content += data.bmp180_pressure;
28   content += "\nweather,sensor=dht22 temperature=";
29   content += data.dht22_temperature;
30   content += ",humidity=";
31   content += data.dht22_humidity;
32   upload(content);
33 }
```

The status bar at the bottom indicates the mode is "NORMAL" and the file is "main.ino". Other status information includes line numbers 20, top, 1:1, and the time 13:50.

Demo

다음에..

백업상

- 장치의 소모 전력 측정하기
- sleep mode를 이용하여 더 오래 작동하도록
- 데이터를 더 드물게 전송하도록



조영호

감사합니다

