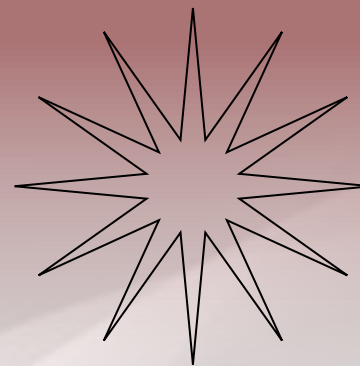
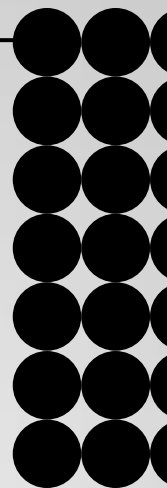


# 라즈베리 파이 기반 미세먼지 탐지기 개발



고려대학교 인공지능사이버보안학과 2018271333 박신영



# Table of contents

## Introduction

프로젝트의 필요성  
타겟과 목표 설정

01

## Design

선행 연구 조사  
연구 방법

02

## Development

구현환경 구축  
시각화 적용

03

## Evaluation

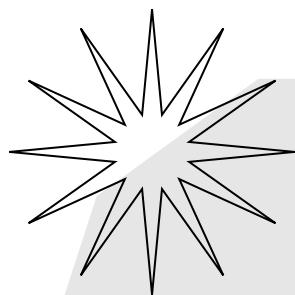
시행착오와 수정  
향후 보완 계획

04

# 01

# Introduction

: 프로젝트의 시작



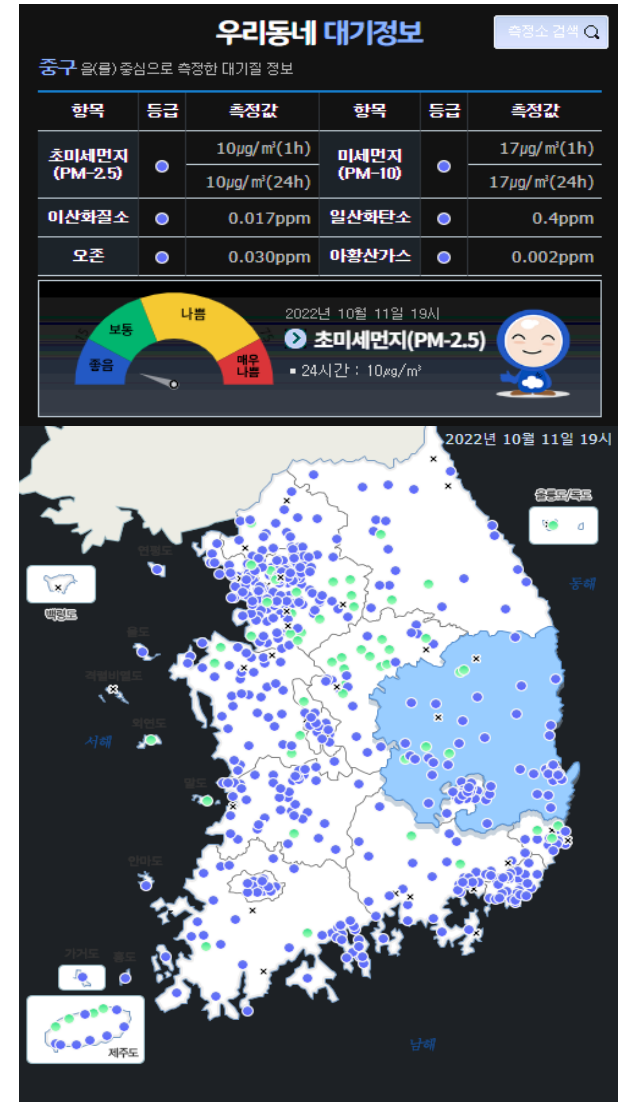
무엇을 위한 프로젝트인가?

누구를 대상으로 하는가?

# 연구 목적

## ✓ 과제의 필요성

- 최근 심각성이 대두되는 **미세먼지 피해 사례** 및 **농도 예보 관심 증가**
  - 개인이 측정한 데이터를 손쉽게 다양한 방법으로 분석할 수 있는 기능 제공
- > 사용자 편의성 확보



AirKorea에서 제공하는 다양한 정보와 API 활용 가능

## 연구 목적

### ✓ 타겟 페르소나 설정



Target. 일반 사용자

장소 제약 없이 **주변 환경 상태**를 측정  
내가 있는 지역의 **예보, 경보** 알기

요구사항

시스템 및 프로그래밍 **배경 지식 부족**

특징

### ✓ 과제 목표

**라즈베리파이** 보드와 **미세먼지 센서**를 연동하여 미세먼지 수치 측정

OpenAPI 활용한 주변 지역 대기질 및 예보 경보 획득

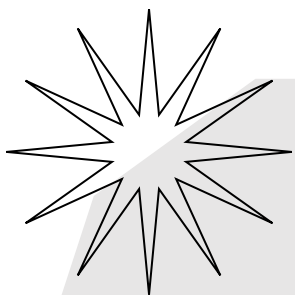
측정 수치 + 예보 정보 -> 사용자에게 편리한 분석/경보 기능 제공

WebUI 설계 후 구현을 통해 쉬운 작동 방식과 오류회복성이 있는 프로그램 작성

# 02

# Design

: 새로운 시스템의 설계



기존의 탐지기는 어떤 방식으로 **데이터**를 출력할까?  
어떤 데이터들을 **표현**할까?

## Air Quality Monitoring System Based on IoT using Raspberry Pi(IEEE, 2017)

- 라즈베리 파이(Raspberry pi 2 model B 사용)로 시스템 제어 + 아두이노에 연결한 센서로 다양한 환경 매개변수 감지(CO, CO2, 온도, 습도, 압력) -> 파이를 통해 클라우드로 지속적 전송
- Node Red
  - Node.js 위에서 사용되는 IoT용 비주얼 프로그래밍 도구, IBM 개발
  - 내장 라이브러리 이용해 시스템 흐름도 시각화
- MQTT 초경량 연결 프로토콜

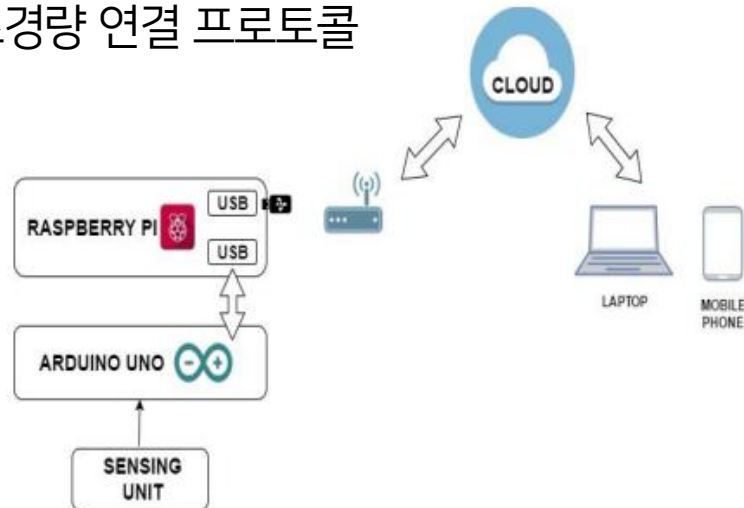


Fig. 1. Simplified diagram of Proposed System

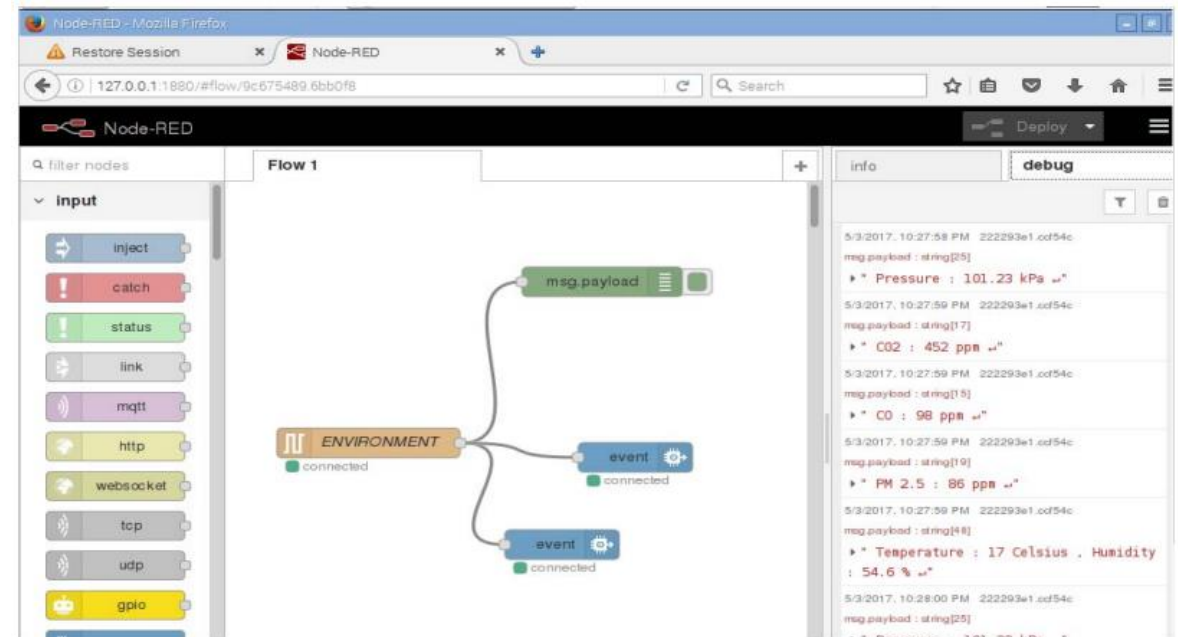


Fig. 3. Node-RED Flow of System

# 연구 방법

## ✓ 과제 수행 방법

- 라즈베리파이와 미세먼지 센서 연결하여 수치 측정, Python 프로그래밍으로 측정 결과 분석 제공
- Github 이용, 라즈베리파이에서 git을 통해 파이썬 코드를 받아 실행
- 프로젝트 환경 구축

라즈베리파이, PMS 7003M 먼지센서, Micro SD카드, 연결케이블

## ✓ 측정장치 제작을 위한 사전 조사

- **라즈베리파이 Raspberry Pi**
  - 초소형/초저가의 싱글 보드 컴퓨터이자 오픈 하드웨어
  - 아두이노와 달리 운영체제 설치가 가능하며 직접 프로그래밍을 통해 외부기기 제어 가능(Microprocessor)
  - 단순 외부기기 제어가 아닌 데이터 처리와 분석 중심인 과제의 특성상 라즈베리파이 사용이 적합





## 연구 방법

### ✓ 측정장치 제작을 위한 사전 조사

- PMS 7003M 먼지센서

- 과제에서 사용할 미세먼지 측정센서(샤오미 사의 공기청정기 미에어프로 내 센서와 호환 가능)
- 메인보드인 라즈베리파이와 연결 후 측정데이터 입출력이 가능해야 함

- GPIO(General Input/Output) UART

- UART란 초기에 주로 사용된 두 장치 간 직렬 데이터를 교환하기 위한 프로토콜
- 단순하고 비용 저렴하며 구현 간편
- 먼지 센서의 데이터 프로토콜이 UART를 통해 전송되므로 Tx, Rx 포트 연결해 데이터 수신 필요
- 포트 활성화 설정이나 USB to UART 제품으로 간단하고 빠른 연결 가능

=> 라즈베리파이 운영체제(라즈비안) 설치 -> 미세먼지 센서 연결 -> 구현한 Python 코드 불러오기

- 측정 및 분석 정보 선별 (날짜/시간정보, 미세먼지 농도 및 수치, 로그)



## ✓ 측정 센서 데이터 시트 확보

### • PMS7003 미세먼지 센서

- 단위부피당 크기가 다른 각 입자의 수로 출력되며,  
입자 수 단위부피는 0.1L이고 질량농도의 단위는  $\mu\text{g}/\text{m}^3$

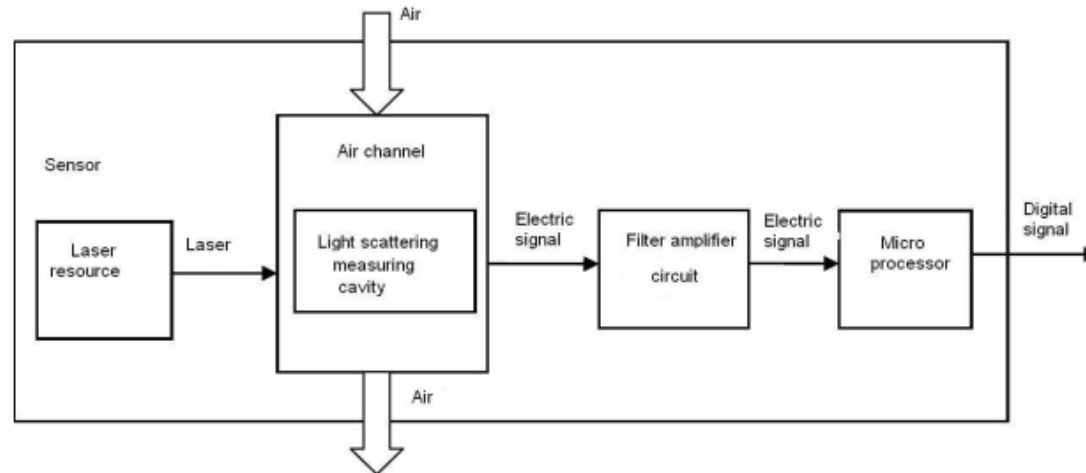


Figure 1 Functional block diagram of sensor

파라미터	인덱스	단위
측정 범위	0.3~1.0 1.0~2.5 2.5~10	$\mu\text{m}$ (마이크로미터)
계산 효율성	50%@0.3 $\mu\text{m}$ 98%@>=0.5 $\mu\text{m}$	
유효 범위 (PM 2.5 기준)	0~500	$\mu\text{g}/\text{m}^3$
최대 범위 (PM 2.5 기준)	1000 이하	$\mu\text{g}/\text{m}^3$
분해능	1	$\mu\text{g}/\text{m}^3$
최대 일관성 오류 (PM 2.5 기준)	$\pm 10\% @ 100 \sim 500 \mu\text{g}/\text{m}^3$ $\pm 10 \mu\text{g}/\text{m}^3 @ 0 \sim 100 \mu\text{g}/\text{m}^3$	
표준 용량	0.1	L 리터
단일 응답 시간	1 이내	second (s)
전체 응답 시간	10 이내	second (s)
전원 공급	Typ: 5.0 Min: 4.5 Max: 5.5	volt (v)
활성 전류	100 이하	Milliampere ( mA )
대기 전류	200 이하	Microampere ( $\mu\text{A}$ )
신호 레벨	L < 0.8 @ 3.3 H > 2.7 @ 3.3	volt (v)
작동 온도 범위	-10 ~ +60	$^{\circ}\text{C}$
작동 습도 범위	0~99%	
보관 온도 범위	-40 ~ +80	$^{\circ}\text{C}$
MTTF (Mean Time to Failure)	3 이하	year (y)
물리적 크기	48×37×12	Millimeter ( mm )

### ✓ 측정 센서 데이터 시트 확보

- **AM2302 온습도 센서**

- 1-wire 버스를 통한 디지털 신호를 출력
- 최대 100m 장거리 신호 전송 가능

전원 공급	3.3 - 5.5V DC
출력 신호	<b>1-wire 버스</b> 를 통한 디지털 신호
감지 요소	고분자 습도 capacitor
작동 범위	습도 0 ~ 100% 온도 -40 ~ 80 °C
정확도	습도 +-2% RH (MAX +-5%) 온도 +-0.5 °C
분해능(해상도)	습도 0.1% RH 온도 0.1 °C
반복성	습도 +-1% RH 온도 +-0.2 °C
습도 이력?	+-0.3 RH
long-term 안정성	+-0.5% RH/year
호환성	완전 호환

# 연구 방법

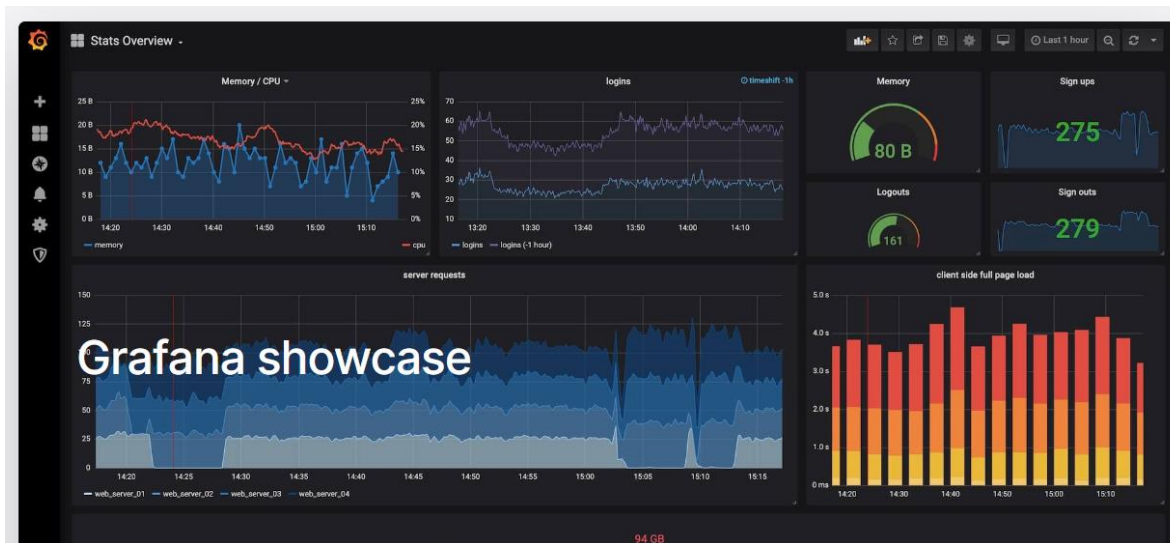
## ✓ 화면 시각화 기획

### • InfluxDB

- 실시간으로 수집된 데이터를 시간 순서에 따라 저장, 조회하는 시계열 DB
- 모니터링 시스템을 위한 UI와 Data store 영역 간편한 설정 가능

### • Grafana

- 오픈소스 대시보드 서비스, 주로 시계열 DB를 데이터소스로 사용



InfluxDB와 Grafana를 연동해  
대시보드에 측정 수치를 시각화하도록 설계

# 연구 방법

## ✓ 화면 시각화 기획

- 공공데이터포털 > 한국환경공단\_에어코리아\_대기오염정보에서 제공하는 활용 가능한 정보 목록

대기질 예보통보 조회

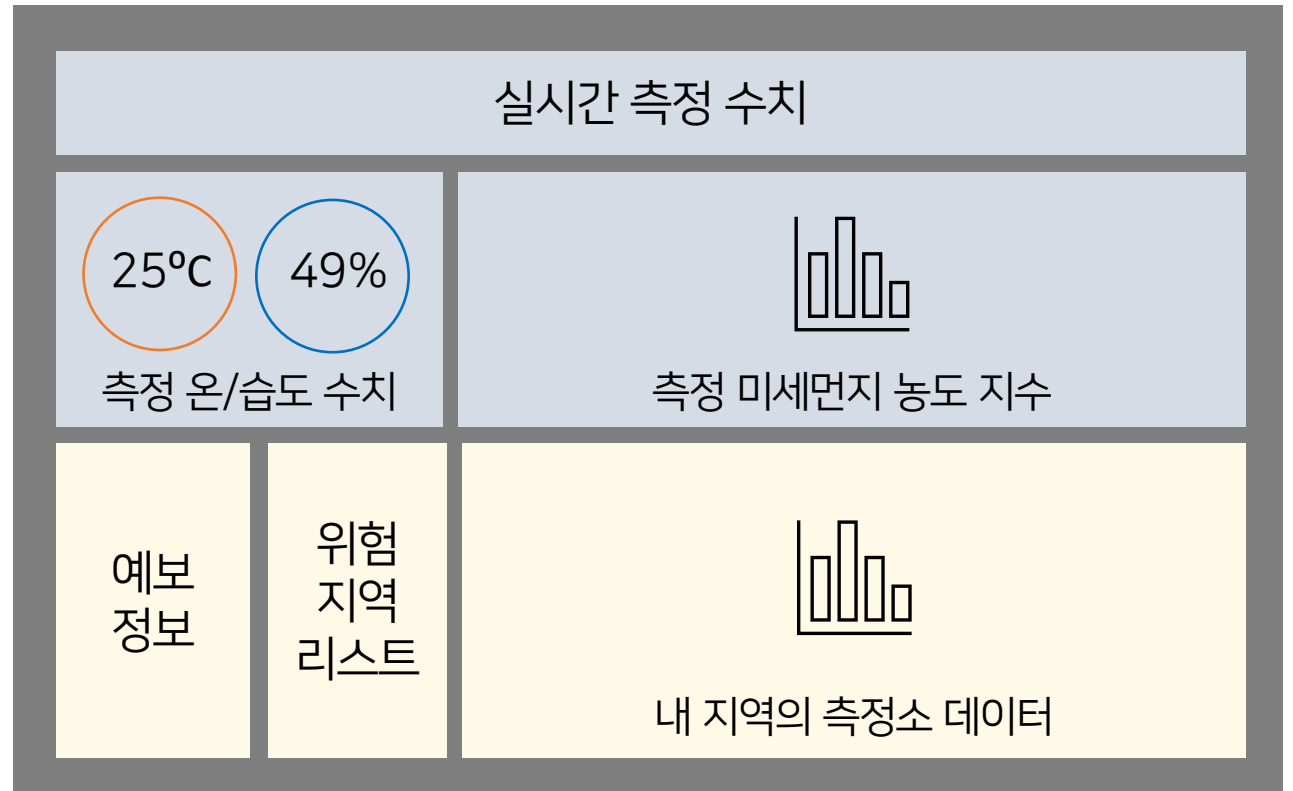
초미세먼지 주간예보 조회

측정소별 실시간 측정정보 조회

통합대기환경지수 나쁨 이상 측정소 목록 조회

시도별 실시간 측정정보 조회

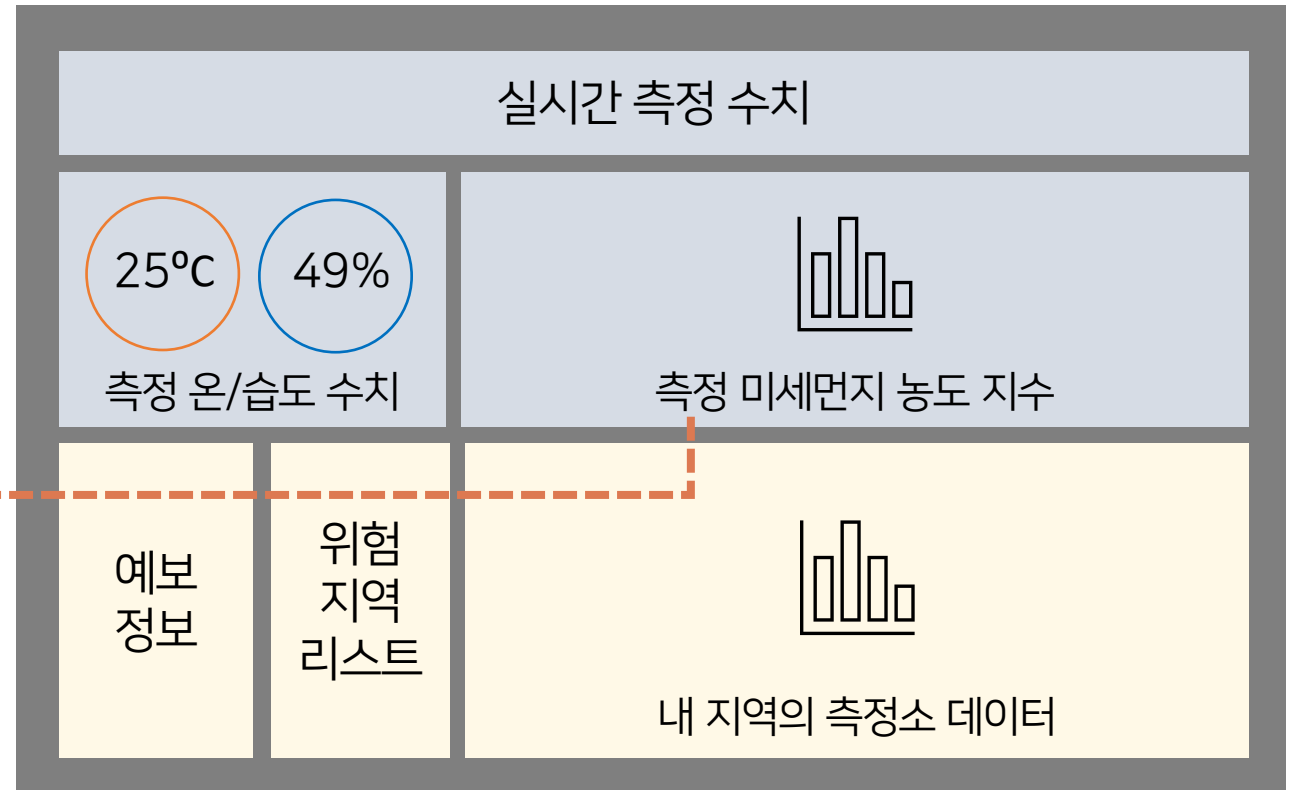
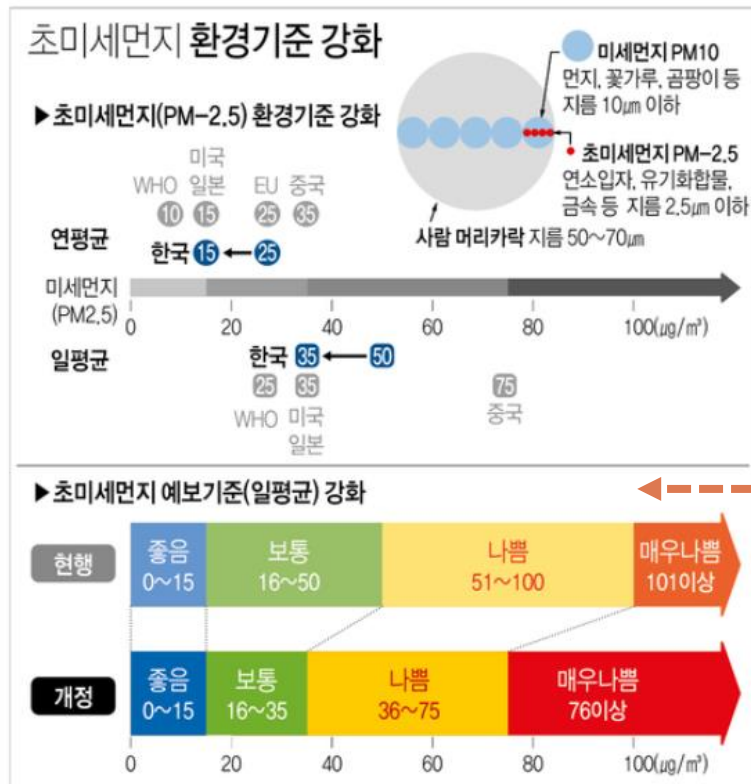
- 최상단에 실시간 센서 측정 정보를 노출
- 화면 전체를 반으로 분할해 하단부터 각 지역 측정소에서 가져온 정보 출력



# 연구 방법

## ✓ 화면 시각화 기획

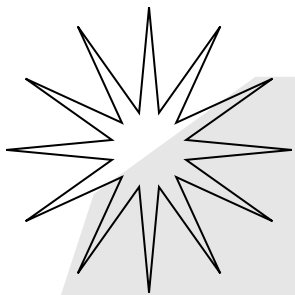
- 미세먼지 환경기준에 따라 농도 지수의 **임계값 (15/35/75)** 설정
- **색상 차이**를 두어 한 눈에 현재 상태를 확인 가능하도록 기획



# 03

## Development

: 미세먼지 탐지기 제안

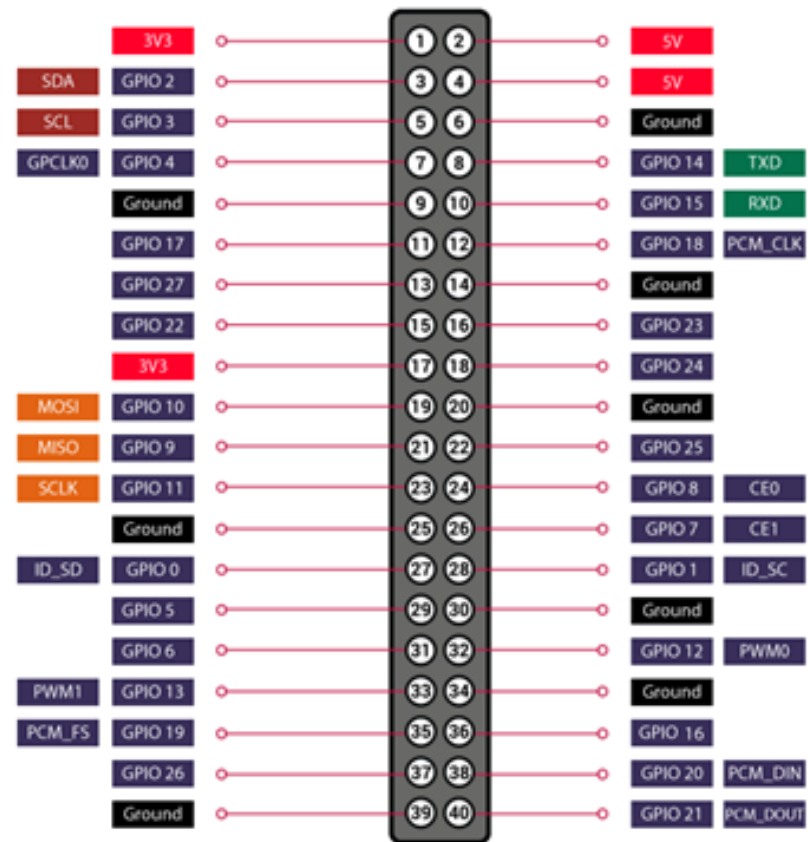
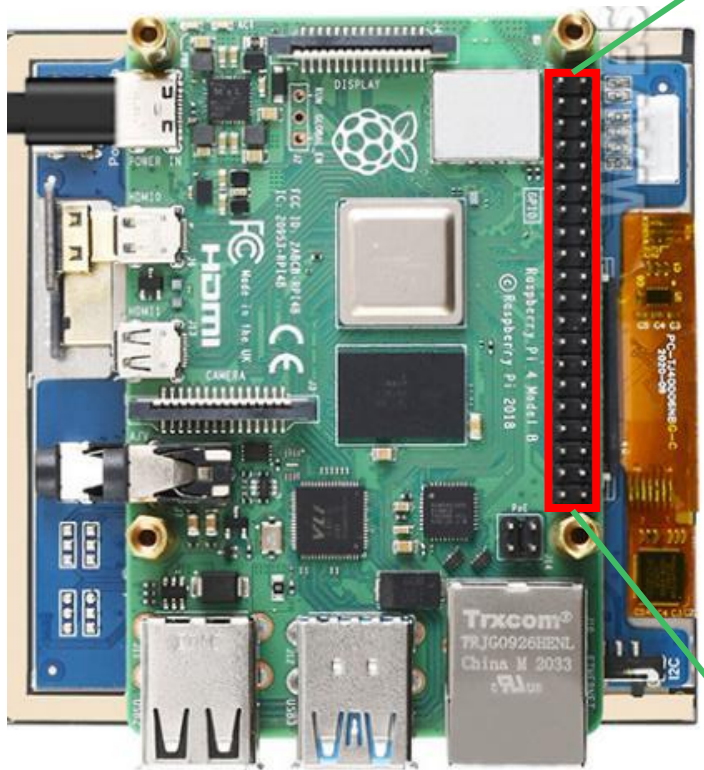


개발 환경의 구축과 구현의 시작

## 환경 구축

### ✓ 측정 환경 구축 및 센서 연결

- 라즈베리파이 핀맵





## 환경 구축

### ✓ 측정 환경 구축 및 센서 연결

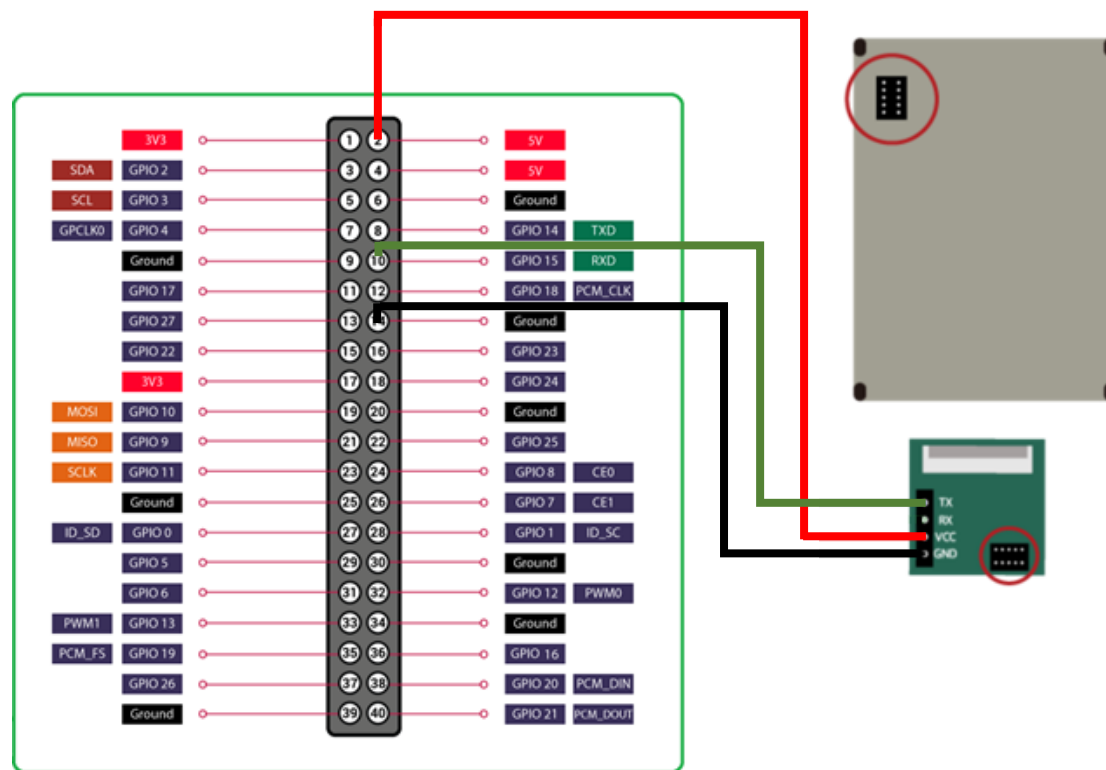
- 미세먼지 센서(PMS7003) 연결도

**VCC** : 트랜지스터에 공급되는 전원을 이야기하는 용어  
모든 회로는 (+)극에서 시작해 (-)극으로 끝남.  
(+)극을 VCC, (-)극을 GND라고 부름.

**GND** : 전압의 크기를 나타내는 기준 전압. 0V이며,  
우리가 평소에 사용하는 110V나 220V는  
기준전압인 GND보다 110V, 220V 높다는 뜻.

**TX** : 데이터를 **송신하는** 핀

**RX** : 데이터를 **수신받는** 핀



## 환경 구축

### ✓ 측정 환경 구축 및 센서 연결

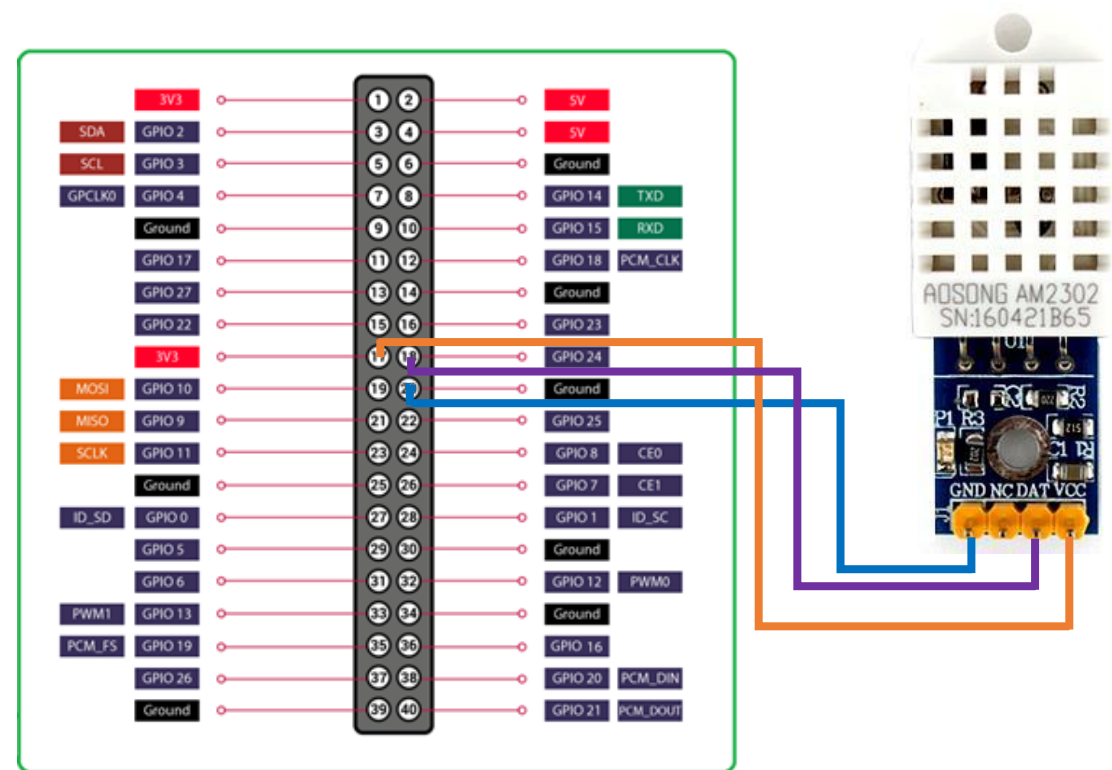
- 온습도 센서(AM2302) 연결도

**VCC** : 트랜지스터에 공급되는 전원을 이야기하는 용어  
모든 회로는 (+)극에서 시작해 (-)극으로 끝남.  
(+)극을 VCC, (-)극을 GND라고 부름.

**GND** : 전압의 크기를 나타내는 기준 전압. 0V이며,  
우리가 평소에 사용하는 110V나 220V는  
기준전압인 GND보다 110V, 220V 높다는 뜻.

**TX** : 데이터를 **송신하는** 핀

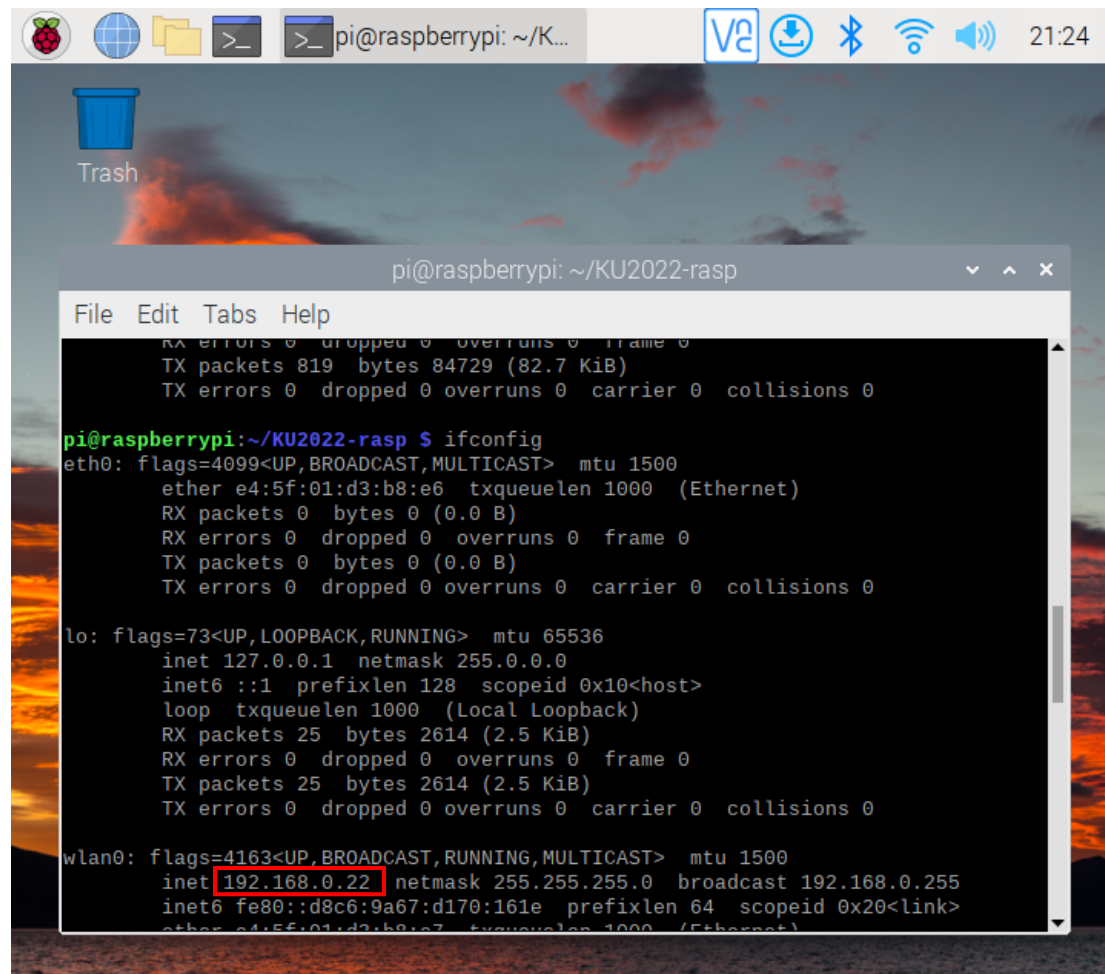
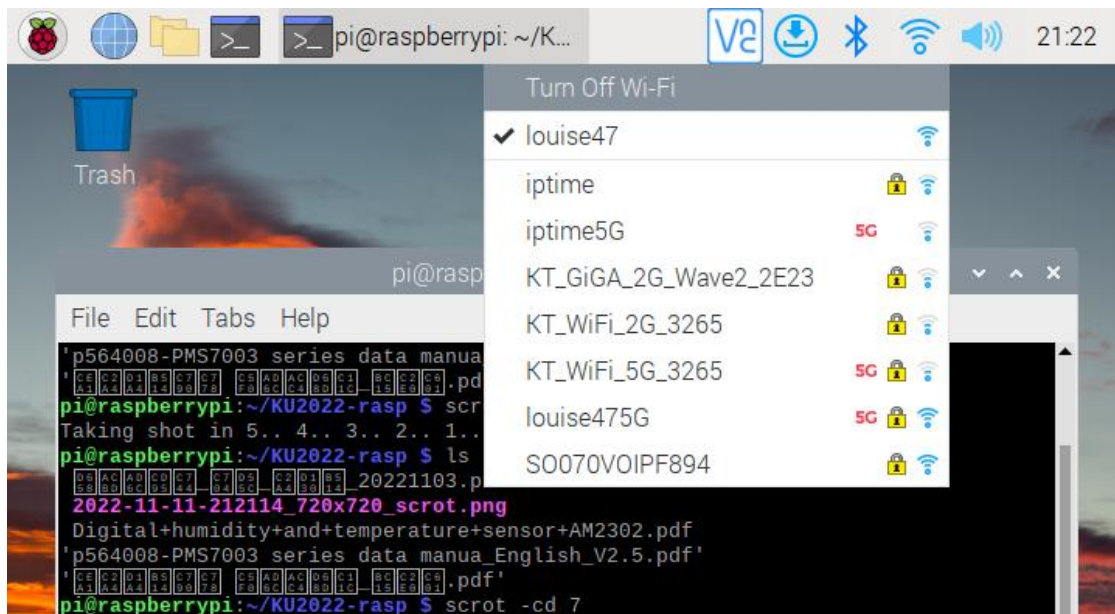
**RX** : 데이터를 **수신받는** 핀



# 환경 구축

## ✓ SSH 접속을 통한 원격 제어

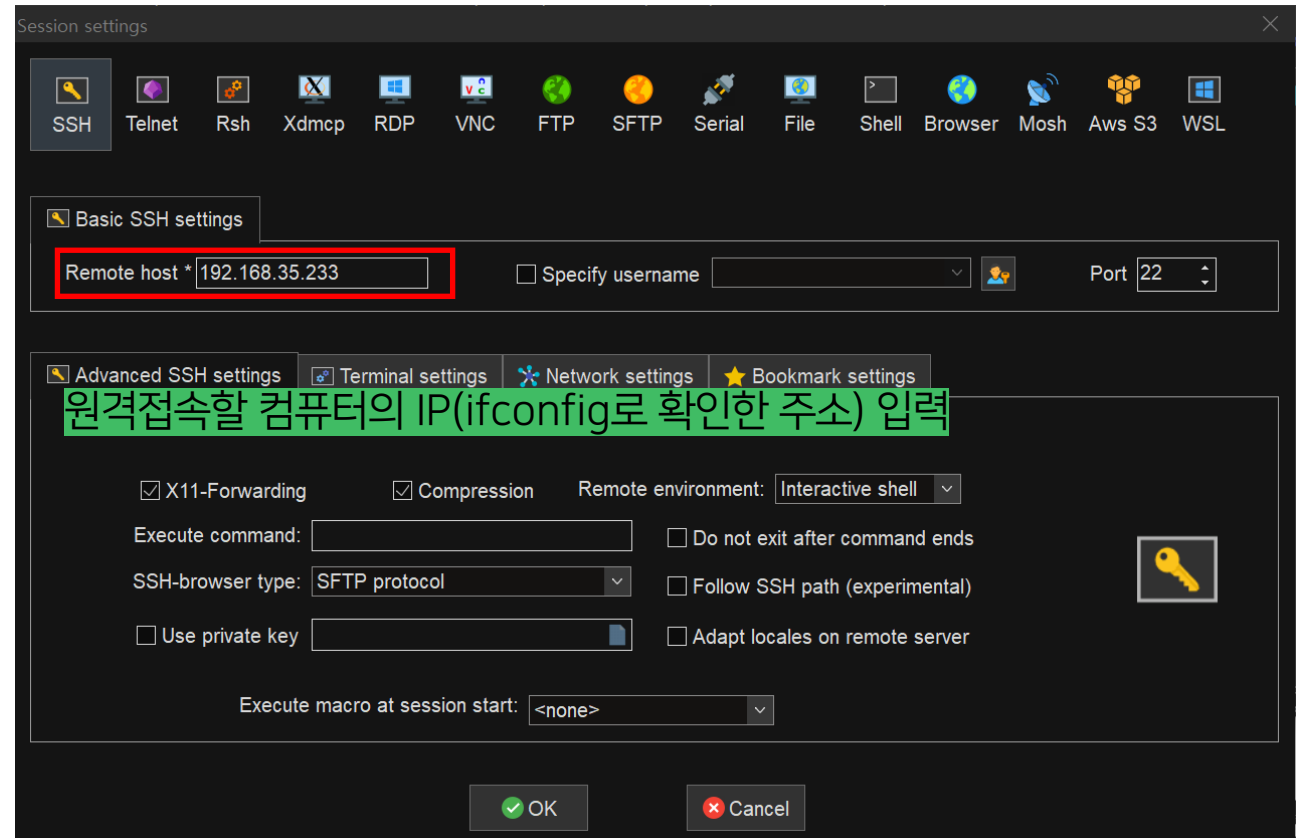
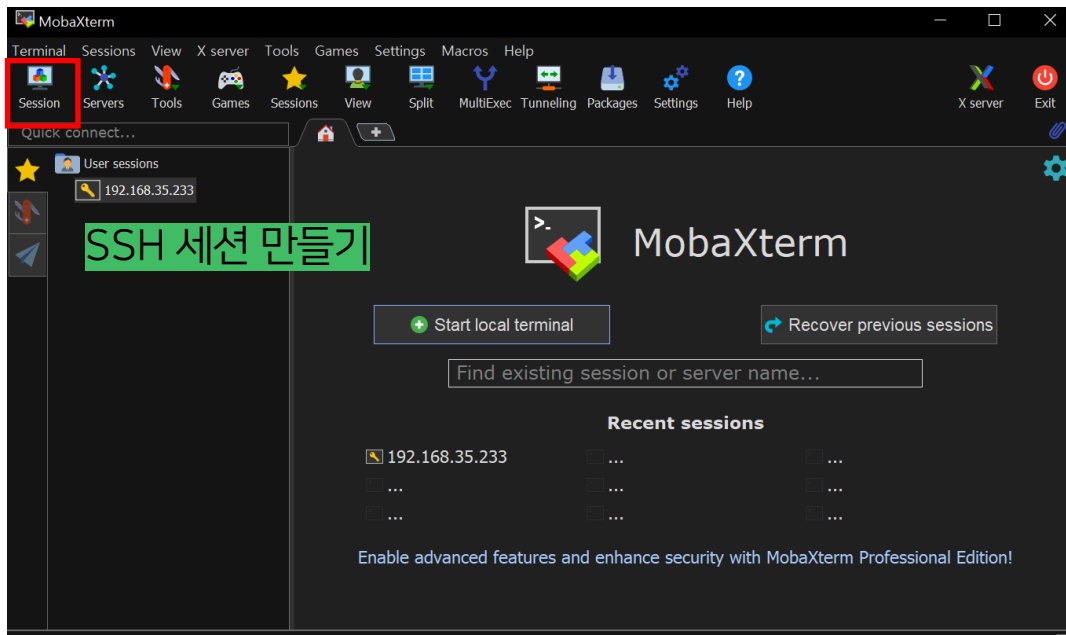
- 라즈베리파이 네트워크 접속
  - Raspbian UI로 손쉽게 와이파이 연결 가능
  - 터미널에서 ifconfig로 현재 IP주소 확인



# 환경 구축

## ✓ SSH 접속을 통한 원격 제어

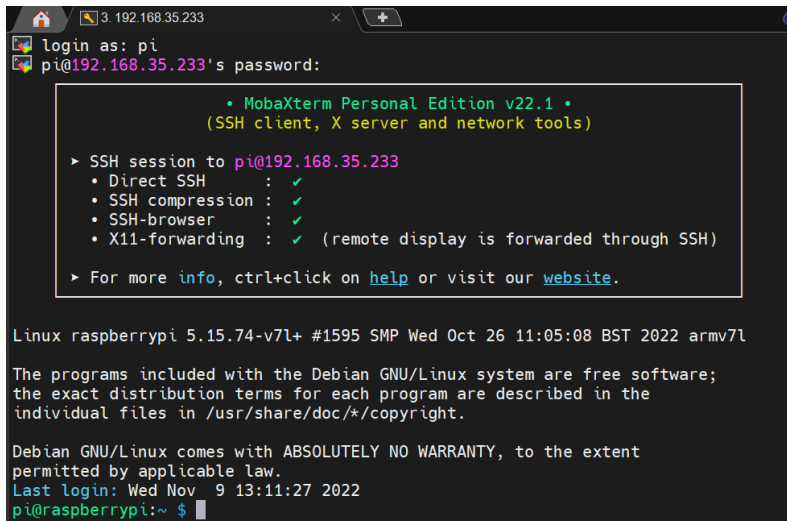
- MobaXterm 사용
  - SSH를 비롯한 여러 기능을 제공하는 원격 컴퓨팅 toolbox



# 환경 구축

## ✓ SSH 접속을 통한 원격 제어

- Login 시 원격 접속 가능



```
login as: pi
pi@192.168.35.233's password:
• MobaXterm Personal Edition v22.1 •
  (SSH client, X server and network tools)

> SSH session to pi@192.168.35.233
  • Direct SSH : ✓
  • SSH compression : ✓
  • SSH-browser : ✓
  • X11-forwarding : ✓ (remote display is forwarded through SSH)

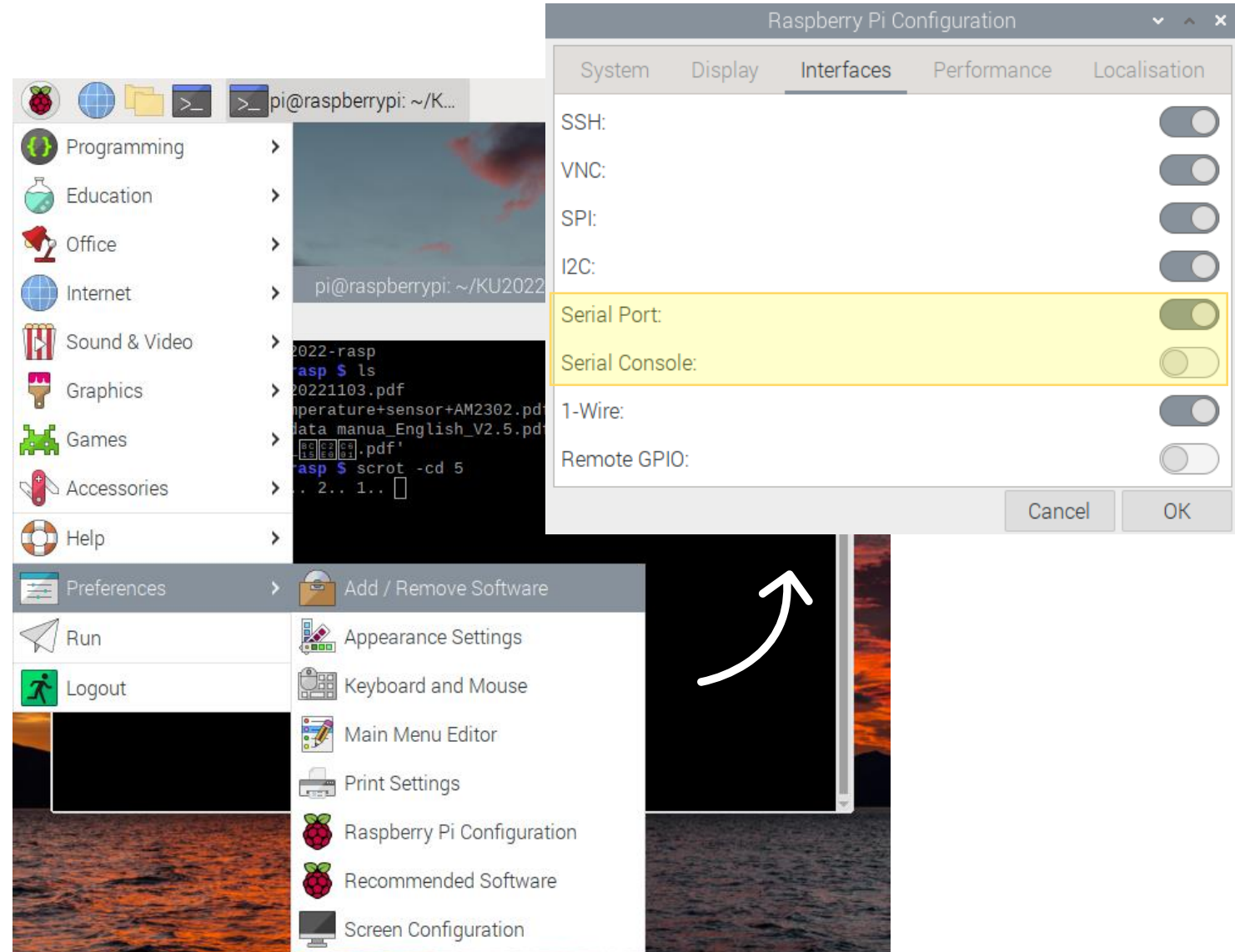
  > For more info, ctrl+click on help or visit our website.

Linux raspberrypi 5.15.74-v7l+ #1595 SMP Wed Oct 26 11:05:08 BST 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 9 13:11:27 2022
pi@raspberrypi:~$
```

- 라즈베리파이와 타 보드(센서) 간 통신을 위한 GPIO UART 활성화  
- config 설정 메뉴에서 Serial port 활성화





# 측정 데이터 추출

## ✓ 센서 데이터 획득

- 먼지센서 데이터 추출

① sudo nano /boot/config.txt로 파일의 맨 아래에

dtoverlay=pi3-disable-bt 명령 입력

② sudo reboot 로 재시작

③ dmesg | grep tty로 확인

-> ttyAMA0 가 사용 가능한 UART임을 확인

```
pi@raspberrypi:~$ ls /dev
autofs      loop1      ram13      tty14      tty40      ttyAMA0    vcsu2
block       loop2      ram14      tty15      tty41      ttyprintk  vcsu3
btrfs-control loop3      ram15      tty16      tty42      ttyS0      vcsu4
bus         loop4      ram2       tty17      tty43      uhid       vcsu5
cachefiles  loop5      ram3       tty18      tty44      uinput     vcsu6
cec0        loop6      ram4       tty19      tty45      urandom    vcsu7
cec1        loop7      ram5       tty2       tty46      v4l        vga_arbiter
char        loop-control ram6       tty20      tty47      vchiq      vhci
console     mapper     ram7       tty21      tty48      vcio       video10
cuse        media0     ram8       tty22      tty49      vc-mem     video11
disk        media1     ram9       tty23      tty5       vcs        video12
dma_heap    media2     random     tty24      tty50      vcs1       video13
dri         media3     rfkill     tty25      tty51      vcs2       video14
fb0         mem        serial0    tty26      tty52      vcs3       video15
fd          mmcblk0    serial1    tty27      tty53      vcs4       video16
full       mmcblk0p1  STT        tty28      tty54      vcs5       video18
```

```
GNU nano 5.4 /boot/config.txt *

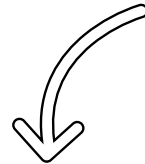
# Disable compensation for displays with overscan
disable_overscan=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[all]

[pi4]
# Run as fast as firmware / board allows
arm_boost=1

[all]
enable_uart=1
dtoverlay=w1-gpio
dtoverlay=pi3-disable-bt
```



```
pi@raspberrypi:~$ dmesg | grep tty
[ 0.000000] Kernel command line: coherent_pool=1M 8250.nr_uarts=1 snd_bcm2835
.enable_compat_alsa=0 snd_bcm2835.enable_hdmi=1 video=HDMI-A-1:720x720M@60 smsc9
5xx.macaddr=E4:5F:01:D3:B8:E6 vc_mem.mem_base=0x3ec00000 vc_mem.mem_size=0x40000
000 console=tty1 root=PARTUUID=c0dd7138-02 rootfstype=ext4 fsck.repair=yes root
wait quiet splash plymouth.ignore-serial-consoles
[ 0.000446] printk: console [tty1] enabled
[ 1.699756] fe201000.serial: ttyAMA0 at MMIO 0xfe201000 (irq = 34, base_baud
= 0) is a PL011 rev2
[ 3.469399] systemd[1]: Created slice system-getty.slice.
```

```
pi@raspberrypi:~/PMS7003$ sudo python3 dust_chk.py
PMS 7003 dust data
PM 1.0 : 16
PM 2.5 : 25
PM 10.0 : 29
```

측정 결과가 잘 출력되는 것을 확인

# 측정 데이터 추출

## ✓ 센서 데이터 획득

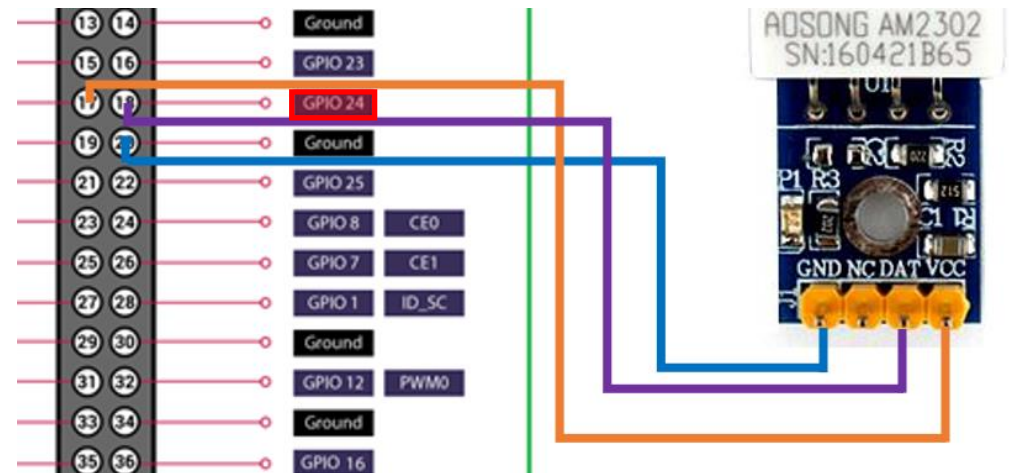
- 온습도 센서 데이터 추출
  - Adafruit\_DHT는 pi3까지만 호환 가능하며 pi4부터는 adafruit-circuitpython-dht를 사용하거나 직접 로컬에 칩셋코드를 추가해야 함.

```
pi@raspberrypi:~ $ sudo pip3 install adafruit-circuitpython-dht
```

```
pi@raspberrypi:~ $ sudo apt-get install libgpiod2
```

```
1 import board
2 import adafruit_dht
3
4 dht_device = adafruit_dht.DHT22(board.D24)
5
6 t = dht.temperature
7 h = dht.humidity
```

Pi에서 Rx 들어오는 핀인 GPIO 24에 맞춰 입력



## 측정 데이터 추출

### ✓ 센서 데이터 획득

- 측정 수치(미세먼지 농도, 온도, 습도, 현재 날짜/시간) 화면 단순 출력 구현

```
DATA read success
=====
Header : B M          | Frame length : 28
PM 1.0 (CF=1) : 17     | PM 1.0 : 17
PM 2.5 (CF=1) : 20     | PM 2.5 : 20
PM 10.0 (CF=1) : 23    | PM 10.0 : 23
0.3um in 0.1L of air : 2835
0.5um in 0.1L of air : 812
1.0um in 0.1L of air : 119
2.5um in 0.1L of air : 10
5.0um in 0.1L of air : 2
10.0um in 0.1L of air : 0
Reserved F : 151 | Reserved B : 0
CHKSUM : 650 | read CHKSUM : 650 | CHKSUM result : True
=====
온 도 : 75.2 F / 24.0 C      습 도 : 51.7%
2022-11-15 17:01:42
```



# 데이터베이스 생성

## ✓ InfluxDB

- Continuous Query 연속적인 쿼리

시간에 따른 데이터의 처리가 핵심 목적

샘플링이 일정 주기마다 실행되도록 연속적인 쿼리 제공

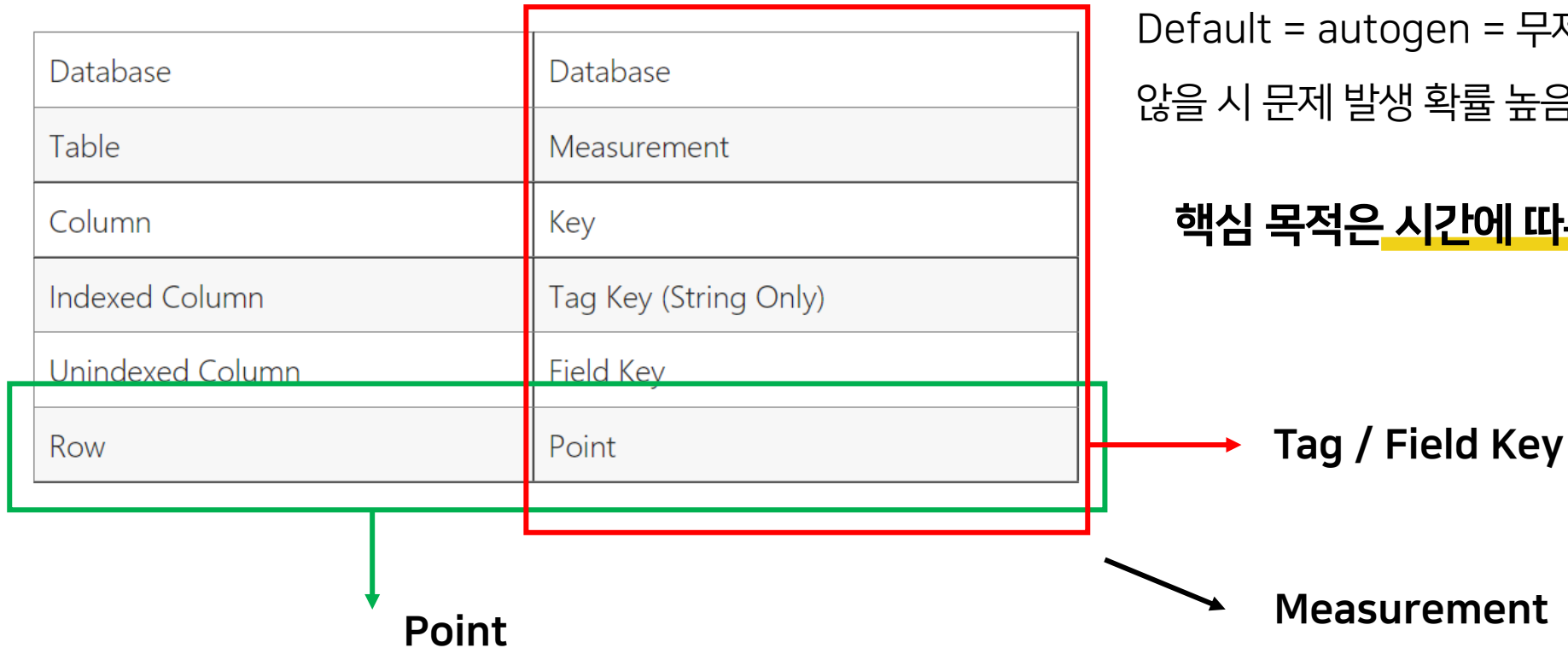
- Retention Policy 보존 정책

오래된 데이터를 자동으로 삭제하는 정책

DB 단위로 정의되며 여러 개의 보존 정책을 가짐

Default = autogen = 무제한이므로 별도 설정을 하지 않을 시 문제 발생 확률 높음

**핵심 목적은 시간에 따른 데이터의 삽입과 조회!**



# 데이터베이스 생성

## ✓ InfluxDB 설치

- InfluxDB에서 버전과 플랫폼 확인 후 설치 진행 (라즈비안은 Debian 기반의 OS)

### InfluxDB 2.x Open Source Time Series Database

InfluxDB is an open source time series database. It has everything you need from a time series platform in a single binary – a multi-tenanted time series database, UI and dashboarding tools, background processing and monitoring agent. All this makes deployment and setup a breeze and easier to secure.

The InfluxDB Platform also includes APIs, tools, and an ecosystem that includes 10 client and server libraries, Telegraf plugins, visualization integrations with Grafana, Google Data Studio, and data sources integrations with Google Bigtable, BigQuery, and more.

Version

Platform

InfluxDB v2.5.1

Ubuntu & Debian

```
# influxdb.key GPG Fingerprint: 05CE15085FC09D18E99EFB22684A14CF2582E0C5
wget -q https://repos.influxdata.com/influxdb.key
echo '23a1c8836f0afc5ed24e0486339d7cc8f6790b83886c4c96995b88a061c5bb5d influxdb.'
echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdb.gpg] https://repos.influxdata.com/debian buster stable'

sudo apt-get update && sudo apt-get install influxdb2
```

[Documentation](#) | [Release Notes](#) | [Register Download](#)

## InfluxDB 1.8 ver 설치

```
$ wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
$ echo "deb https://repos.influxdata.com/debian buster stable"
  | sudo tee /etc/apt/sources.list.d/influxdb.list
$ sudo apt update
$ sudo apt install influxdb
$ sudo systemctl unmask influxdb
$ sudo systemctl enable influxdb

$ sudo systemctl start influxdb
```

- 매뉴얼대로 정상 설치 실패
- 라즈베리 파이에서 **InfluxDB 2.x 이상 버전**은 설치가 **불가능**

# 데이터베이스 생성

## ✓ InfluxDB로 센서 데이터 가져오기

- 설치한 InfluxDB에 데이터를 연결, 저장

```
pi@raspberrypi:~ $ influxd
88888888      .d888 888      88888888b. 8888888b.
888      d88P" 888      888  "Y88b 888  "88b
888      888  888      888      888  888  .88P
888 888888b. 8888888 888 888 888 888 888 888 8888888K.
888 888 "88b 888 888 888 888 Y8bd8P" 888 888 888 "Y88b
888 888 888 888 888 888 888 X88K 888 888 888 888
888 888 888 888 888 Y88b 888 .d8""8b. 888 .d88P 888 d88P
88888888 888 888 888 888 "Y88888 888 888 8888888P" 88888888P"

2022-12-06T11:46:06.674734Z info InfluxDB starting {"log_id": "0eaoyA
.10", "branch": "1.8", "commit": "688e697c51fd"}
2022-12-06T11:46:06.674835Z info Go runtime {"log_id": "0eaoyApW000",
"maxprocs": 4}
run: open server: listen: listen tcp 127.0.0.1:8088: bind: address already in use

pi@raspberrypi:~ $ service influxdb start
==== AUTHENTICATING FOR org.freedesktop.systemd1.man
Authentication is required to start 'influxdb.servic
Authenticating as: ,, (pi)
Password:
==== AUTHENTICATION COMPLETE ====
pi@raspberrypi:~ $ influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> CREATE
```

- <http://localhost:8086> 의 InfluxDB  
에 연결이 잘 이루어졌는지 체크

```
# influxdb에 연결하는 함수
def get_ifdb(db, host = 'localhost', port = 8086, user = 'root', passwd = 'root'):
    client = InfluxDBClient(host, port, user, passwd, db)

    try:
        client.create_database(db)

        print('Connection Successful')
        print('=====')
        print('Connection Info')
        print('=====')
        print('host:', host)
        print('username :', user)
        print('database :', db)

    except:
        print('Connection Failed')
        pass

    return client
```

# 데이터베이스 생성

## ✓ InfluxDB로 센서 데이터 가져오기

# 연결된 데이터베이스에 데이터를 입력하는 함수

```
def my_test(ifdb):
    json_body = []
    tablename = 'sensor_data'
    fieldname = 'mydata'
    data = dust.unpack_data(buffer)
    point = {
        "measurement":tablename,
        "tags":{

        },
        "fields":{
            "PM 1.0": data[dust.DUST_PM1_0_CF1],
            "PM 2.5": data[dust.DUST_PM2_5_CF1],
            "PM 10.0": data[dust.DUST_PM10_0_CF1],
            "0.3um in 0.1L of air": data[dust.DUST_AIR_0_3],
```

```
def do_test(): # 코드 실행 시 가장 먼저 do_test 함수 실행
    mydb = get_ifdb(db='sensor_data')

    my_test(mydb)
```

```
        "temperature": dhtDevice.temperature,
        "humidity": dhtDevice.humidity
    },
    "time":None,
}

vals = list(range(1,11))

for v in vals:
    dt = datetime.now() - timedelta(hours=-9) # 한국 시간 UTC 9 고려

    np = deepcopy(point)
    np['fields'][fieldname] = v
    np['time'] = dt
    json_body.append(np)

    time.sleep(1)

ifdb.write_points(json_body)

result = ifdb.query('select * from %s'%tablename)
pprint.pprint(result.raw)
```

- Points의 Measurement, tags, fields를 이용해 데이터를 **sensor\_data**라는 새로 생성한 데이터베이스에 저장

# 데이터베이스 생성

## ✓ 데이터베이스 확인

```
pi@raspberrypi:~ $ influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> show databases
name: databases
name
----
_internal
PMS7003
myDB
DHT22_DB
sensor_data

> use sensor_data
Using database sensor_data
> show measurements
name: measurements
name
----
sensor_data
```

- InfluxDB의 기본 조작 명령어

Influx DB 접속	Influx
DB 생성	Create database sensor_data use sensor_data
measurement 조회	Show measurements select * from sensor_data
태그키 조회	Show tag keys show field keys

# 데이터베이스 생성

## ✓ 데이터베이스 확인

```
> select "temperature" from sensor_data
name: sensor_data
time                temperature
----                -
1670485182865441000 27.9
1670485183866671000 27.9
1670485184868118000 27.9
1670485185869571000 27.9
1670485186870967000 27.9
1670485187872377000 27.9
1670485188873605000 27.9
1670485189875011000 27.9
1670485190876407000 27.9
1670485191877813000 27.9
1670486198335528000 27.5
1670486199336811000 27.5
```

```
> show series
key
---
sensor_data
sensor_data,0.3um\ in\ 0.1L\ of\ air=4128,0.5um\ in\ 0.1L\ of\ air=1178,1.0um\ in\ 0.1L\ of\ air=184,10.0
um\ in\ 0.1L\ of\ air=0,2.5um\ in\ 0.1L\ of\ air=12,5.0um\ in\ 0.1L\ of\ air=4,PM\ 1.0=26,PM\ 10.0=34,PM\
2.5=30
sensor_data,0.3um\ in\ 0.1L\ of\ air=4788,0.5um\ in\ 0.1L\ of\ air=1358,1.0um\ in\ 0.1L\ of\ air=239,10.0
um\ in\ 0.1L\ of\ air=0,2.5um\ in\ 0.1L\ of\ air=18,5.0um\ in\ 0.1L\ of\ air=8,PM\ 1.0=28,PM\ 10.0=44,PM\
2.5=39
sensor_data,0.3um\ in\ 0.1L\ of\ air=4830,0.5um\ in\ 0.1L\ of\ air=1414,1.0um\ in\ 0.1L\ of\ air=250,10.0
um\ in\ 0.1L\ of\ air=0,2.5um\ in\ 0.1L\ of\ air=12,5.0um\ in\ 0.1L\ of\ air=2,PM\ 1.0=29,PM\ 10.0=40,PM\
2.5=39
```

```
> select "humidity" from sensor_data
name: sensor_data
time                humidity
----                -
1670485182865441000 48.4
1670485183866671000 48.4
1670485184868118000 48.4
1670485185869571000 48.4
1670485186870967000 48.4
1670485187872377000 48.4
1670485188873605000 48.4
1670485189875011000 48.4
1670485190876407000 48.4
1670485191877813000 48.4
1670486198335528000 48.4
1670486199336811000 48.4
```

- 측정데이터가 모두 데이터베이스에 잘 반영되는 것을 확인

## 시각화 과정

### ✓ Grafana 설치 및 데이터베이스 연동

#### • Docker

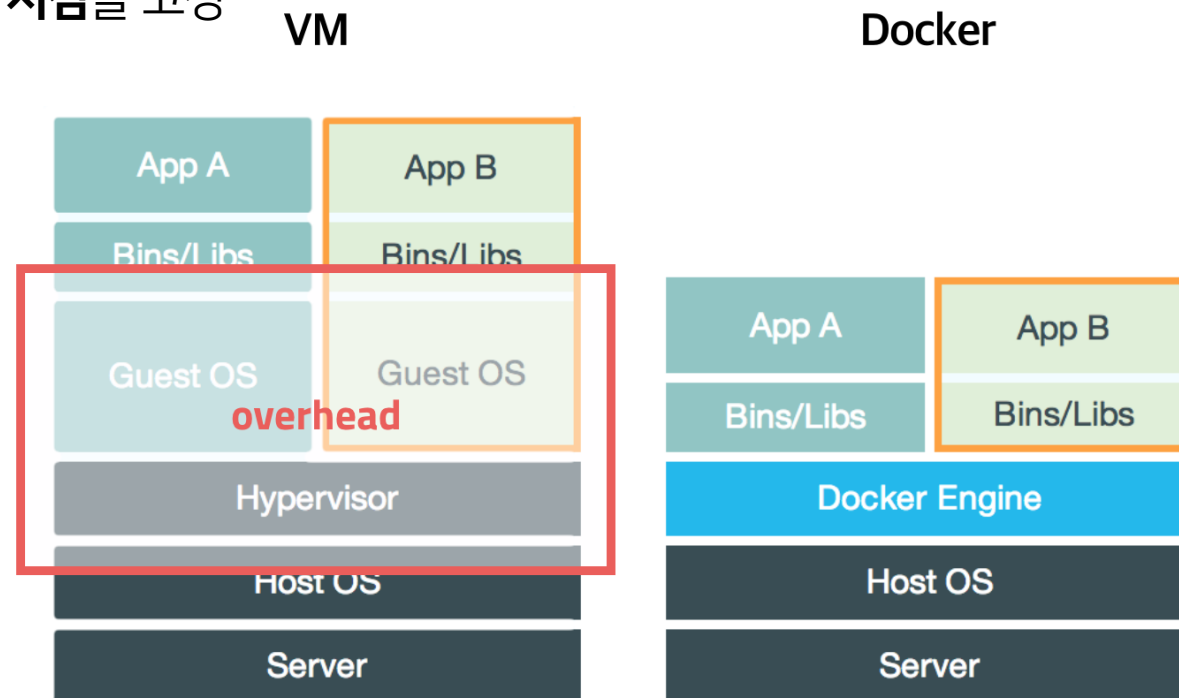
- 애플리케이션을 환경에 구매 받지 않고 실행하는 **컨테이너** 도구
- 서버를 코드로 구성하고 관리하는 방법으로 Docker의 장점 활용 -> Snowflake Servers 관리
- 도커 파일로 만든 **도커 이미지**를 통해 **서버 구성 시점**을 고정

#### • VM과의 차이

##### VM VS Docker

Host와 완벽하게 분리  
가상화된 HW 위 OS가  
올라가는 방식

Docker 엔진 위에 시스템  
실행에 **필요한 바이너리만**  
-> 속도, 성능 이점



# 시각화 과정

## ✓ Grafana 설치 및 데이터베이스 연동

- <http://localhost:3000/> 접속 시 Grafana 대시보드
- 초기 ID: admin / PW: admin

```
$ sudo curl -sSL https://get.docker.com | sh
$ sudo usermod -aG docker ${USER}
$ groups ${USER}
$ sudo apt-get install libffi-dev libssl-dev
$ sudo pip3 install docker-compose
$ sudo systemctl enable docker

$ sudo reboot
$ docker run -d -p 3000:3000 grafana/grafana
```

```
-----error failed building wheel for bcrypt-----
$ sudo pip3 install -U "bcrypt<4.0.0"로 해결-----
```

버전 충돌로 발생한 오류 해결

연동을 위해 [influxdb.conf](#) 파일의 **[http]** 부분 수정 필요

```
# Determines whether the Flux query endpoint is enabled.
flux-enabled = true
```

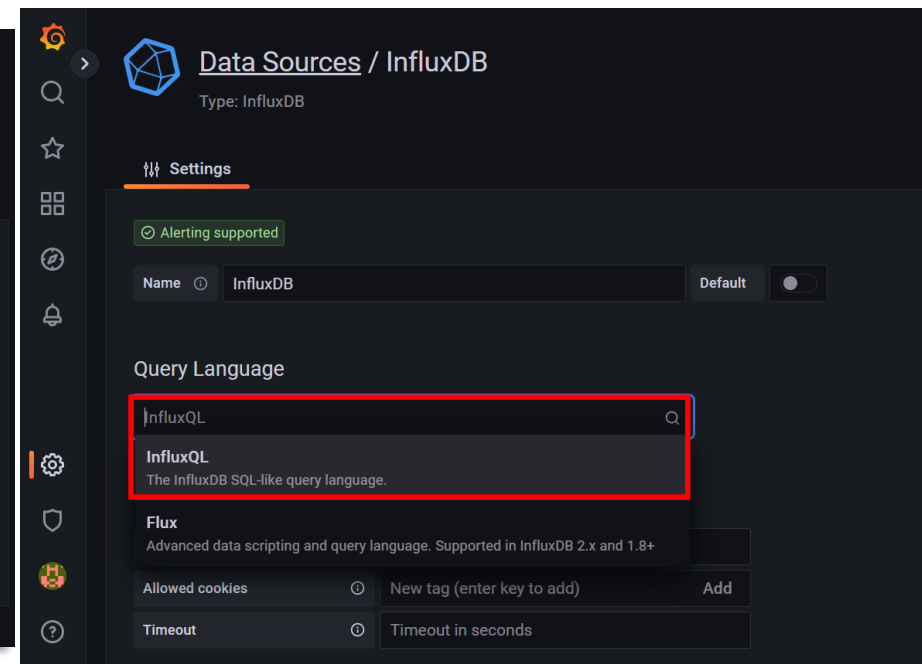
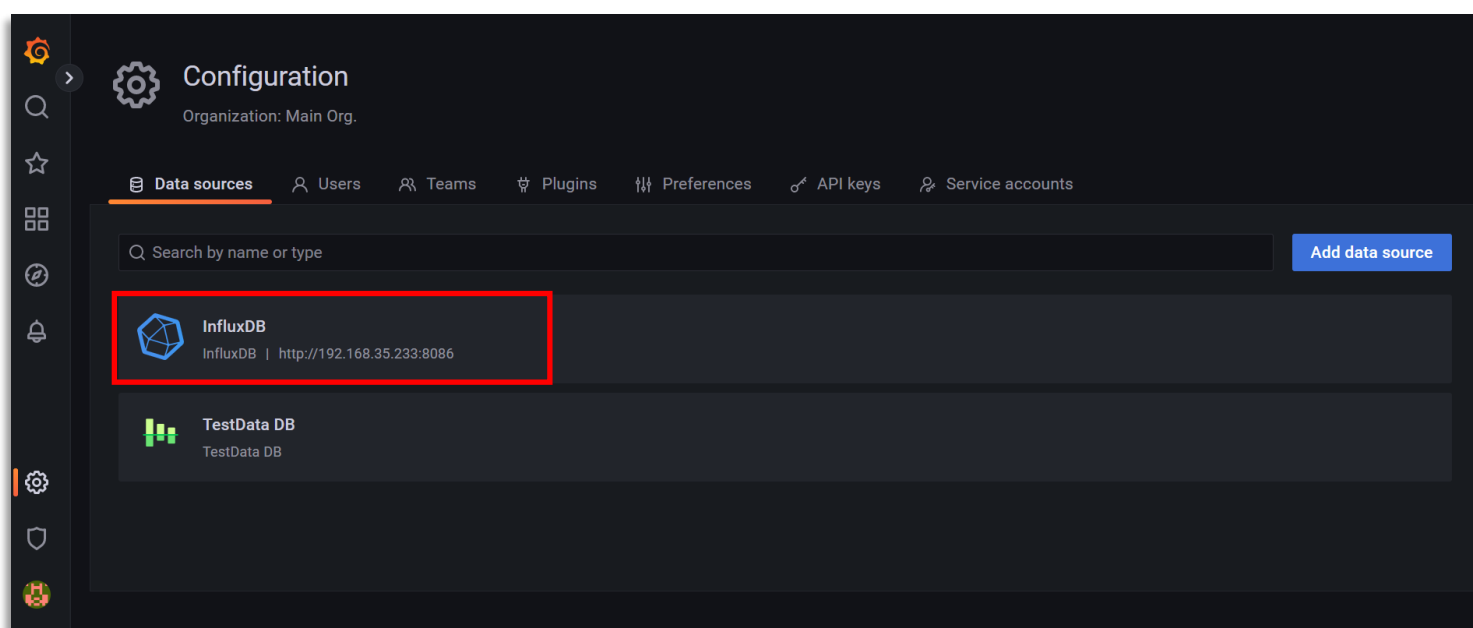
```
# Determines whether the Flux query logging is enabled.
# flux-log-enabled = false
```



# 시각화 과정

## ✓ Grafana 대시 보드 구성

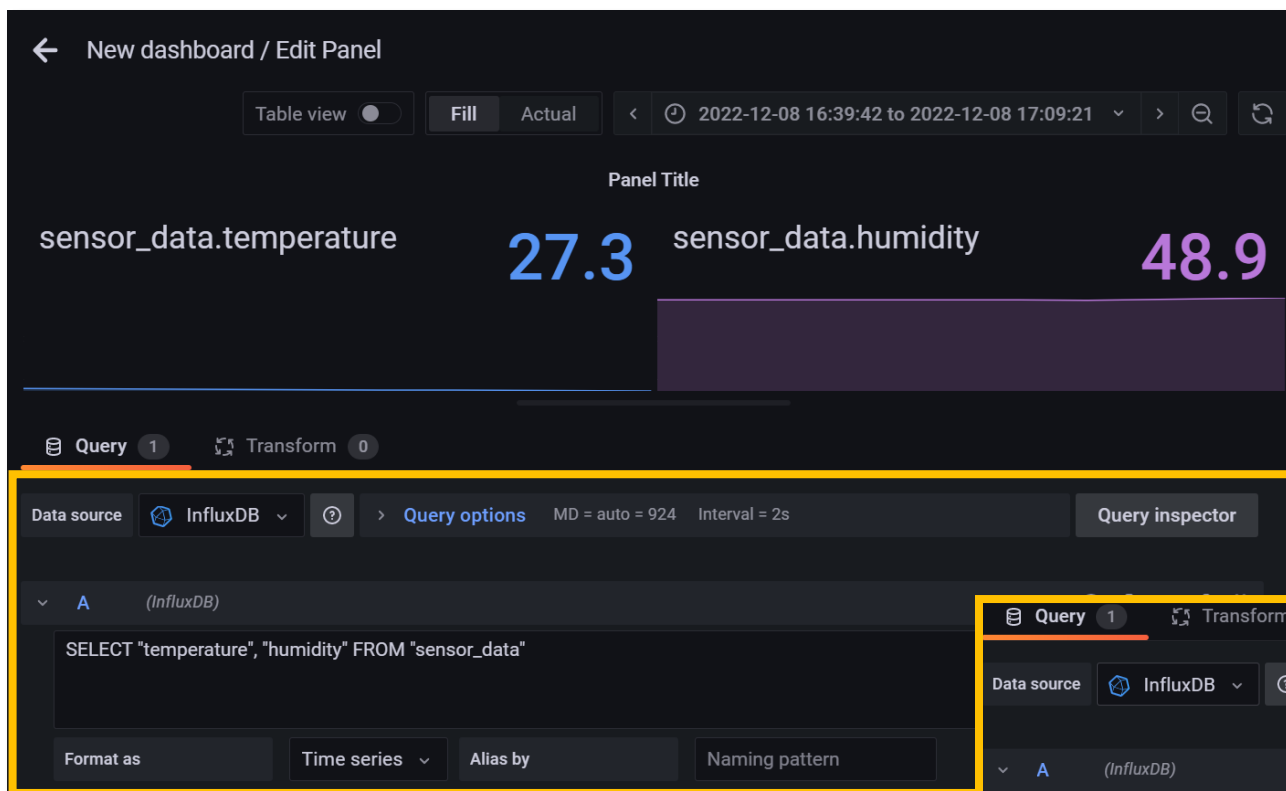
- InfluxDB 1.8 버전에서는 쿼리 언어 Flux가 지원되지 않고 **InfluxQL**만 가능
- 쿼리 언어를 InfluxQL로 설정해야 Grafana와 정상적으로 연동 가능



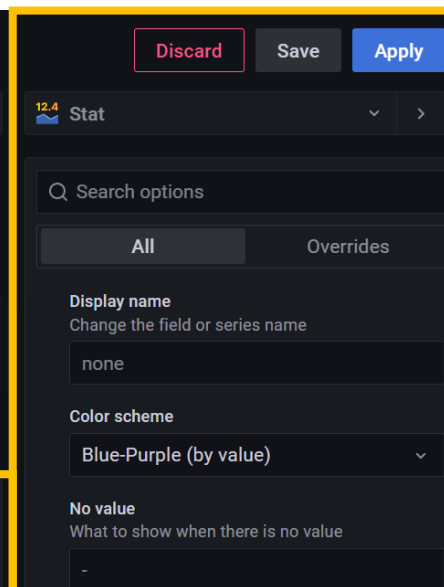
# 시각화 과정

## ✓ Grafana 대시 보드 구성

- 연동한 데이터베이스에 대해 쿼리문으로 패널을 만지고 세부 설정 조정

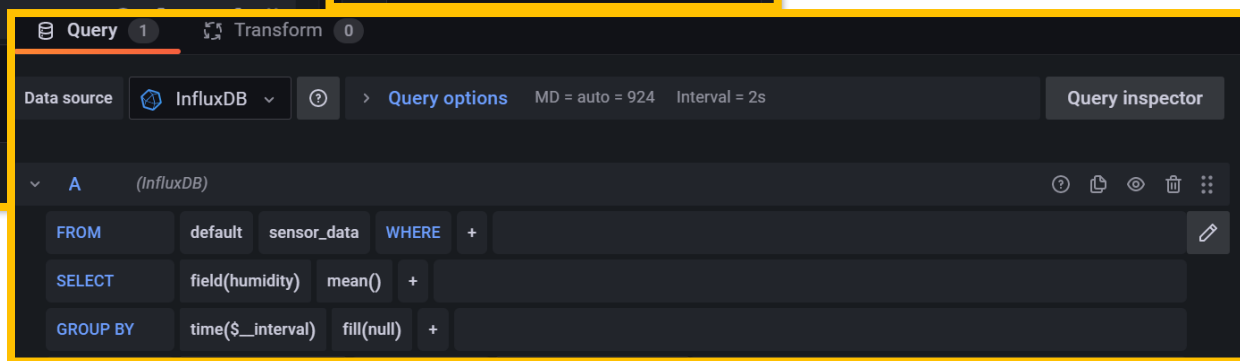


직접 쿼리문 작성



- 다양한 시각화 **그래프** 지원
- 색상 및 임계점, 축 설정 등 아주 구체적이고 세부적인 설정 가능

**에디터 모드** 변환 가능



# 시각화 과정

## ✓ Grafana 대시 보드 구성

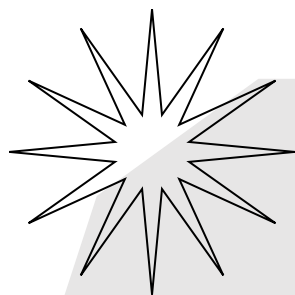
- 초기 시계열 데이터의 특징을 살린 그래프를 적용
- 수치 변화가 크지 않은 온도/습도의 경우 의미가 없다고 판단



# 04

## Evaluation

: 개선사항과 앞으로의 목표



목표한 **기능**이 잘 구현되었는가?

앞으로 **보완**할 점은?

# 개선사항

## ✓ Open API 크롤링

- 크롤링 오류로 초기 설계와 달리 API 정보들을 이용하지 못함

환경기상

한국환경공단

활용신청 [승인] 한국환경공단\_에어코리아\_대기오염정보

신청일 2022-11-15 만료예정일 2024-11-15

### 요청변수(Request Parameter)

항목명	샘플데이터	
serviceKey	<b>인증키</b>	공공데이터포털에서 받은 인증키
returnType	xml	xml 또는 json
numOfRows	1	한 페이지 결과 수
pageNo	1	페이지번호
stationName	신흥동	측정소 이름
dataTerm	DAILY	요청 데이터기간(1일: DAILY, 1시간: HOURLY, 1분: MINUTE)
ver	1.0	버전별 상세 결과 참고

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <header>
    <resultCode>00</resultCode>
    <resultMsg>NORMAL_CODE</resultMsg>
  </header>
  <body>
    <items>
      <item>
        <so2Grade>1</so2Grade>
        <coFlag>
          <khaiValue>55</khaiValue>
          <so2Value>0.003</so2Value>
          <coValue>0.6</coValue>
          <pm25Flag>
            <pm10Flag>
              <pm10Value>36</pm10Value>
              <o3Grade>2</o3Grade>
              <khaiGrade>2</khaiGrade>
              <pm25Value>26</pm25Value>
              <no2Flag>
                <no2Grade>1</no2Grade>
                <o3Flag>
                  <pm25Grade>2</pm25Grade>
                  <so2Flag>
                    <dateTime>2022-11-16 13:00</dateTime>
                    <coGrade>1</coGrade>
                    <no2Value>0.014</no2Value>
                    <pm10Grade>1</pm10Grade>
                    <o3Value>0.036</o3Value>
                  </so2Flag>
                </pm25Grade>
              </no2Flag>
            </pm10Flag>
          </coFlag>
        </item>
      </items>
      <numOfRows>1</numOfRows>
      <pageNo>1</pageNo>
      <totalCount>23</totalCount>
    </body>
  </response>
```

```
# AirKorea OpenAPI 사용
import requests
from bs4 import BeautifulSoup
import pandas
import urllib3 # 경고 무시

urllib3.disable_warnings()

M = '&numOfRows=1&pageNo=1&stationName=신흥동&dataTerm=DAILY&ver=1.3'
key = 'tsFgvelgFo8g9a12hc4f1YcN9z2S16kGxMe7FbBTAaPyEcR8gI2K8bFpegd02S4ngadYMTWn64d0MFzYH71w%3D%3D'
url = 'https://apis.data.go.kr/B552584/ArpltnInforInquireSvc/getMsrstnAcctoRltmMeasureDnsty?serviceKey=' + key + M
# 변수url의 구조: 'url주소'+ '?ServiceKey='+ '인증키를 넣기'+ 조건

# requests CERTIFICATE_VERIFY_FAILED 경고 무시
session = requests.Session()
session.verify = False
session.post(url='https://apis.data.go.kr/B552584/ArpltnInforInquireSvc/getMsrstnAcctoRltmMeasureDnsty?serviceKey=' + key + M, data={'bar': 'baz'})

response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")
ItemList = soup.findAll('item')
for item in ItemList:
    a = item.find('datetime').text
    g = item.find('pm10value').text
    i = item.find('pm25value').text
    s = item.find('pm10grade1h').text
    t = item.find('pm25grade1h').text
    print('측정소: 신흥동')
    print('측정시간: ' + a)
    print('미세먼지 농도: ' + g + '㎍/㎥' + s + '>')
    print('초미세먼지 농도: ' + i + '㎍/㎥' + s + '>')
    print('좋음 1 | 보통 2 | 나쁨 3 | 매우나쁨 4')
```

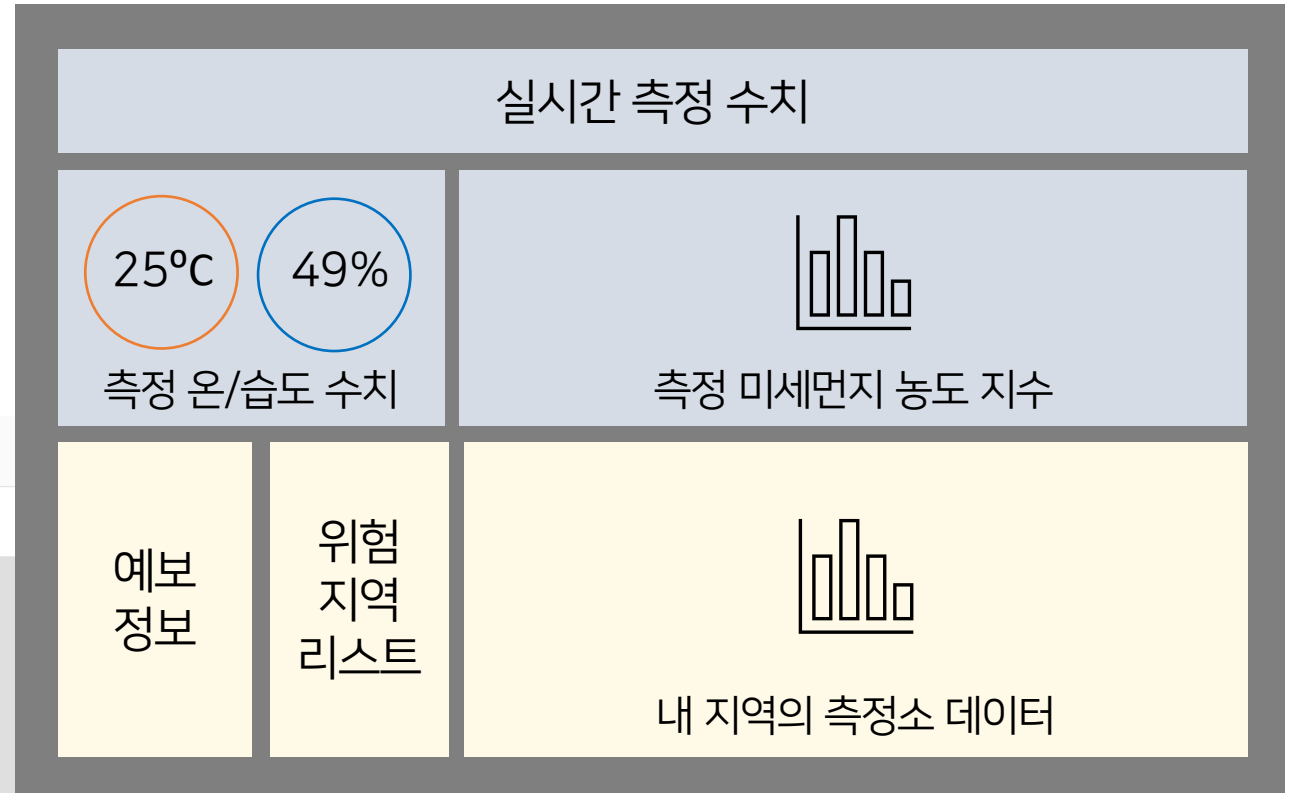
# 개선사항

## ✓ Geolocation API

- **내 주변 측정소**의 데이터를 가져와야 함
- 크롤링 시 위치 정보를 파악하는 API를 사용해 데이터를 가져올 측정소 판단 필요

요청변수(Request Parameter)

항목명	샘플데이터	설명
serviceKey	<b>인증키</b>	공공데이터포털에서 받은 인증키
returnType	xml	xml 또는 json
numOfRows	1	한 페이지 결과 수
pageNo	1	페이지번호
stationName	신흥동	측정소 이름
dataTerm	DAILY	요청 데이터기간(1일: DAILY, 1개월: MONTH, 3개월: 3MONTH)
ver	1.0	버전별 상세 결과 참고



# 개선사항

## ✓ 향후 보완할 점



## UI 보완

Cli를 사용하지 않고 바로 Grafana  
대시보드 화면을 **LCD에 출력**  
측정장치 및 데이터베이스 **연결 확인**  
**로그**가 나타나지 않도록 수정

## 정보 가공

단순한 origin 정보의 출력 뿐만 아니라 **데이터**  
**간 상관관계 분석** 등 의미 있는 정보로 가공  
측정데이터의 경우 모두 **숫자형**이었으나  
스트링 등의 **다양한 형태의 데이터**는 어떻게  
한 화면에 표현할 것인가?

*Thank you*

---



**KOREA**  
UNIVERSITY