인하공업전문대학
INHA TECHNICAL COLLEGE

무선 센서 네트워크
12주차

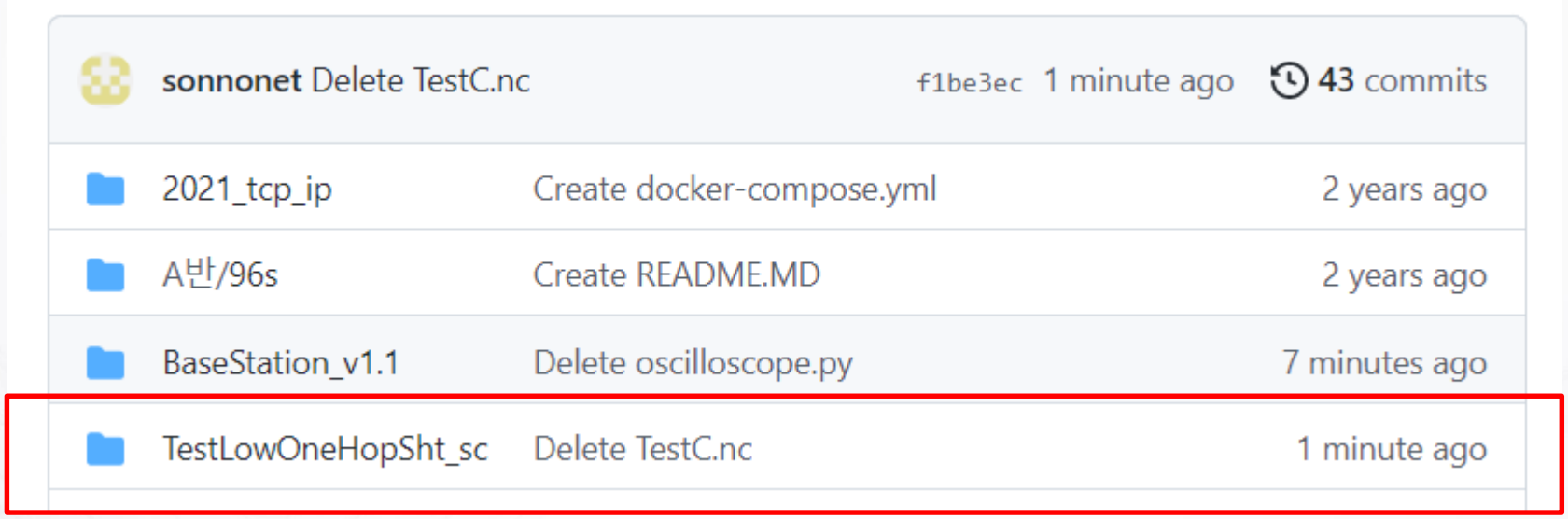
인하공업전문대학 컴퓨터 정보과

김한결 강사

목차

- 요약
  I. Zigbee TinyOS 기반 온습도 센서
  II. NesC기반 미들웨어
  III. GUI 그라파나 표현

- Github 소스코드 다운
- https://github.com/sonnonet/inhatc



TinyOS 기반 NesC 프로그래밍 (온습도센서)

TestAppC.nc

~/sonnonet_tinyos/Tinyos/TestLowOneHopSht_sc

```
includes Test;
configuration TestAppC
{
}
implementation
{
  components TestC, MainC;
  components LedsC, new TimerMilliC();

  components ActiveMessageC as AMC;
  components new AMSenderC(AM_TEST_DATA_MSG) as AMSC;

  TestC.Boot -> MainC;
  TestC.Leds -> LedsC;
  TestC.MilliTimer -> TimerMilliC;

  TestC.RadioControl -> AMC;
  TestC.RadioSend -> AMSC;

  components new SensirionSht11C() as Sht11ChOC;
  TestC.Temp -> Sht11ChOC.Temperature;
  TestC.Humi -> Sht11ChOC.Humidity;

  components new IlluAdcC() as Illu;
  TestC.Illu -> Illu;

  components BatteryC;
  TestC.Battery -> BatteryC;
}
                                                    16,0-1        All
```

TinyOS 기반 NesC 프로그래밍 (온습도센서)

Test.h

~/sonnonet_tinyos/Tinyos/TestLowOneHopSht_sc

```
#ifndef TEST_H
#define TEST_H
#include "message.h"
enum {
  TEST_PERIOD = 10240LU,
};
enum {
  DFLT_VAL = 0x11,
};
enum {
  TEST_DATA_LENGTH = TOSH_DATA_LENGTH - 6,
};
enum {
  AM_TEST_DATA_MSG = 0xA4,
};

typedef nx_struct test_data_msg {
  nx_am_addr_t srcID;
  nx_uint32_t seqNo;
  nx_uint16_t type;
  nx_uint16_t Temp;
  nx_uint16_t Humi;
  nx_uint16_t Illu;
  nx_uint16_t battery;
  //nx_uint8_t testData[TEST_DATA_LENGTH];
} test_data_msg_t;

#endif // TEST_H
~
```

TestC.nc

~/sonnonet_tinyos/Tinyos/TestLowOneHopSht_sc

```
module TestC
{
  uses {
    interface Boot;
    interface Leds;
    interface Timer<TMilli> as MilliTimer;

    interface SplitControl as RadioControl;
    interface AMSend as RadioSend;

    interface Read<uint16_t> as Temp;
    interface Read<uint16_t> as Humi;
    interface Read<uint16_t> as Illu;

    interface Battery;
  }
}

```

TinyOS 기반 NesC 프로그래밍 (온습도센서)

TestC.nc

~/sonnonet_tinyos/Tinyos/TestLowOneHopSht_sc

```
implementation
{
  message_t testMsgBffr;
  test_data_msg_t *testMsg;

  uint32_t seqNo;
  uint8_t step;


  task void startTimer();
  event void Boot.booted() {
    testMsg = (test_data_msg_t *)call RadioSend.getPayload(
      &testMsgBffr, sizeof(test_data_msg_t));
    testMsg->srcID = TOS_NODE_ID;

    seqNo = 0;

    post startTimer();
  }

  task void startTimer() {
    call MilliTimer.startPeriodic(TEST_PERIOD);
  }

  task void radioOn();
  event void MilliTimer.fired() {
    post radioOn();
  }
```

TestC.nc

```
void startDone();
task void radioOn() {
  if (call RadioControl.start() != SUCCESS) startDone();
}

event void RadioControl.startDone(error_t error) {
  startDone();
}

task void readTask();
void startDone() {
  step = 0;
  post readTask();
  call Leds.led0Toggle();
}

void sendDone();
task void sendTask() {
  testMsg->seqNo = seqNo++;
  testMsg->type = 2; //THL type 2

  if (call RadioSend.send(AM_BROADCAST_ADDR, &testMsgBffr,
    sizeof(test_data_msg_t)) != SUCCESS) sendDone();
  call Leds.led2Toggle();
}

```

TestC.nc

```
  event void RadioSend.sendDone(message_t* msg, error_t error) {
    sendDone();
  }

  task void radioOff();
  void sendDone() {
    call Leds.led0Off();
    call Leds.led1Off();
    call Leds.led2Off();
    post radioOff();
  }

  void stopDone();
  task void radioOff() {
    if (call RadioControl.stop() != SUCCESS) stopDone();
  }

  event void RadioControl.stopDone(error_t error) {
    stopDone();
  }

  void stopDone() {
  }

```

TestC.nc

```
97   task void readTask() {
98     switch(step) {
99       case 0:
100        call Temp.read(); break;
101      case 1:
102        call Humi.read(); break;
103      case 2:
104        call Illu.read(); break;
105      default:
106        testMsg->battery = call Battery.getVoltage();
107        post sendTask();
108        break;
109    }
110    step += 1;
111  }
112
113  event void Temp.readDone(error_t error, uint16_t val) {
114    //if (error != SUCCESS) call Leds.led00n();
115    testMsg->Temp = error == SUCCESS ? val : 0xFFFA;
116    post readTask();
117  }
118  event void Humi.readDone(error_t error, uint16_t val) {
119    //if (error != SUCCESS) call Leds.led10n();
120    testMsg->Humi = error == SUCCESS ? val : 0xFFFB;
121    post readTask();
122  }
123  event void Illu.readDone(error_t error, uint16_t val){
```

TestC.nc

```
  event void Illu.readDone(error_t error, uint16_t val){
    testMsg->Illu = error == SUCCESS ? val : 0xFFFC;
    post readTask();
  }
}
```

oscilloscope.py

~/sono_raspberrypi/BaseStation_v1.1

```
# Data Format
# CO2  Data0
# THL Temperature Data0, Humidity Data1, Illumination Data2, Battery Data3

import sys
import tos
import datetime
import threading

AM_OSCILLOSCOPE = 0x93

class OscilloscopeMsg(tos.Packet):
    def __init__(self, packet = None):
        tos.Packet.__init__(self,
                            [('srcID',  'int', 2),
                             ('seqNo', 'int', 4),
                             ('type', 'int', 2),
                             ('Data0', 'int', 2),
                             ('Data1', 'int', 2),
                             ('Data2', 'int', 1),
                             ('Data3', 'int', 1),
                             ('Data4', 'int', 2),
                             ],
                            packet)
if '-h' in sys.argv:
    print "Usage:", sys.argv[0], "serial@/dev/ttyUSB0:57600"
    sys.exit()

```

oscilloscope.py

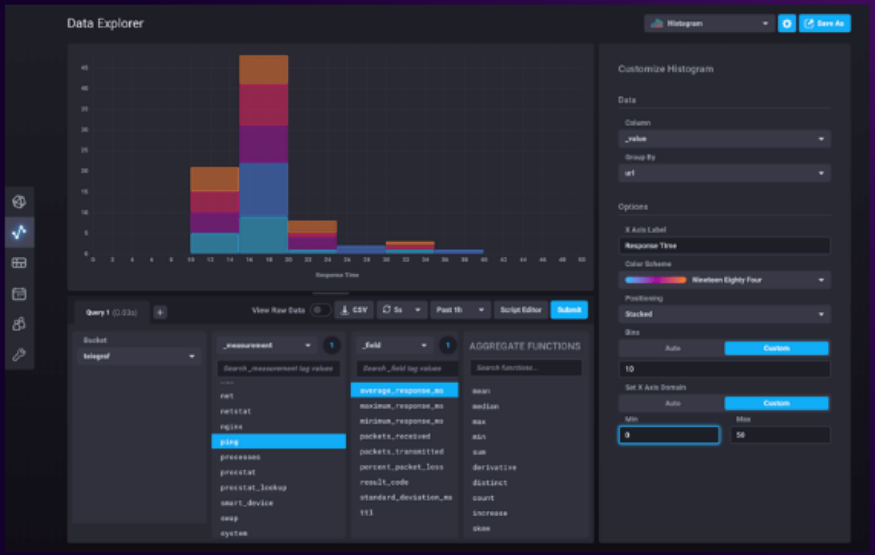
```
am = tos.AM()

while True:
    p = am.read()
    msg = OscilloscopeMsg(p.data)
    print p

####### THL Logic ############
    if msg.type == 2:
        battery = msg.Data4

        Illumi = int(msg.Data2)+ int(msg.Data3*256)
        Illumi = Illumi
        humi = -2.0468 + (0.0367*msg.Data1) + (-1.5955*0.000001)*msg.Data1*msg.Data1
        temp = -(39.6) + (msg.Data0 * 0.01)
        try:
            with conn.cursor() as curs:
                                Now = datetime.datetime.now()
                                sql = """insert into JB_Sensor_THL(NODE_ID,SEQ,TEMPERATURE,HUMIDITY,ILLUMINATION,REGDATE)
₩
                                    values(%s, %s, %s, %s, %s, %s)"""
                                curs.execute(sql,(msg.srcID,msg.seqNo,temp,humi,Illumi,Now))
                                conn.commit()
        except all,e:
            print e.args
            conn.close()
        print "id:" , msg.srcID, " Count : ", msg.seqNo, ₩
                "Temperature: ",temp, "Humidity: ",humi, "Illumination: ",Illumi, "Battery : ", battery
```

InfluxDB

InfluxDB

InfluxDB is a time series database designed to handle high write and query loads.

Get InfluxDB



InfluxDB 1.x

InfluxDB 1.x is the open source time series database component of the TICK Stack (Telegraf, InfluxDB, Chronograf, Kapacitor).

InfluxDB 2.0

Currently in beta, InfluxDB 2.0 incorporates everything you need in a time series platform into a single binary.

InfluxDB Cloud

InfluxDB Cloud is a fast, elastic, serverless time series platform as a service – easy to use with usage-based pricing

시계열데이터베이스 설치

InfluxDB 설치

- https://github.com/sonnonet/2024_inhatc

InfluxDB 설치

- InfluxDB download key using wget

```
wget -q https://repos.influxdata.com/influxdata-archive_compat.key
echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c influxdata-archive_compat.key' |
echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com
```

- Packages are up to date && install Influxdb

```
sudo apt-get update && sudo apt-get install influxdb -y
```

- InfluxDB as a background service on startup

```
sudo service influxdb start
```

- InfluxDB is status (service)

```
sudo service influxdb status
```

InfluxDB 데이터베이스 만들기

```
$ influx

>create database <데이터베이스이름>
확인 : show databases
```

시계열데이터베이스 설치

Python InfluxDB lib 설치

- https://github.com/sonnonet/2024_inhatc

influxdb import with python

```
pip install influxdb
```

오픈 소스 GUI 설치

Grafana 설치

- https://github.com/sonnonet/2024_inhatc

Grafana Installation

1. Install the prerequisite packages

```
sudo apt-get install -y apt-transport-https software-properties-common wget
```

2. Import the GPG key:

```
sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /etc/apt/keyrings/grafana.gpg >
```

3. To add a repository for stable releases, run the following command:

```
echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | sudo tee -a
```

4. Run the following command to update the list of available packages:

```
sudo apt-get update && sudo apt-get install grafana -y
```

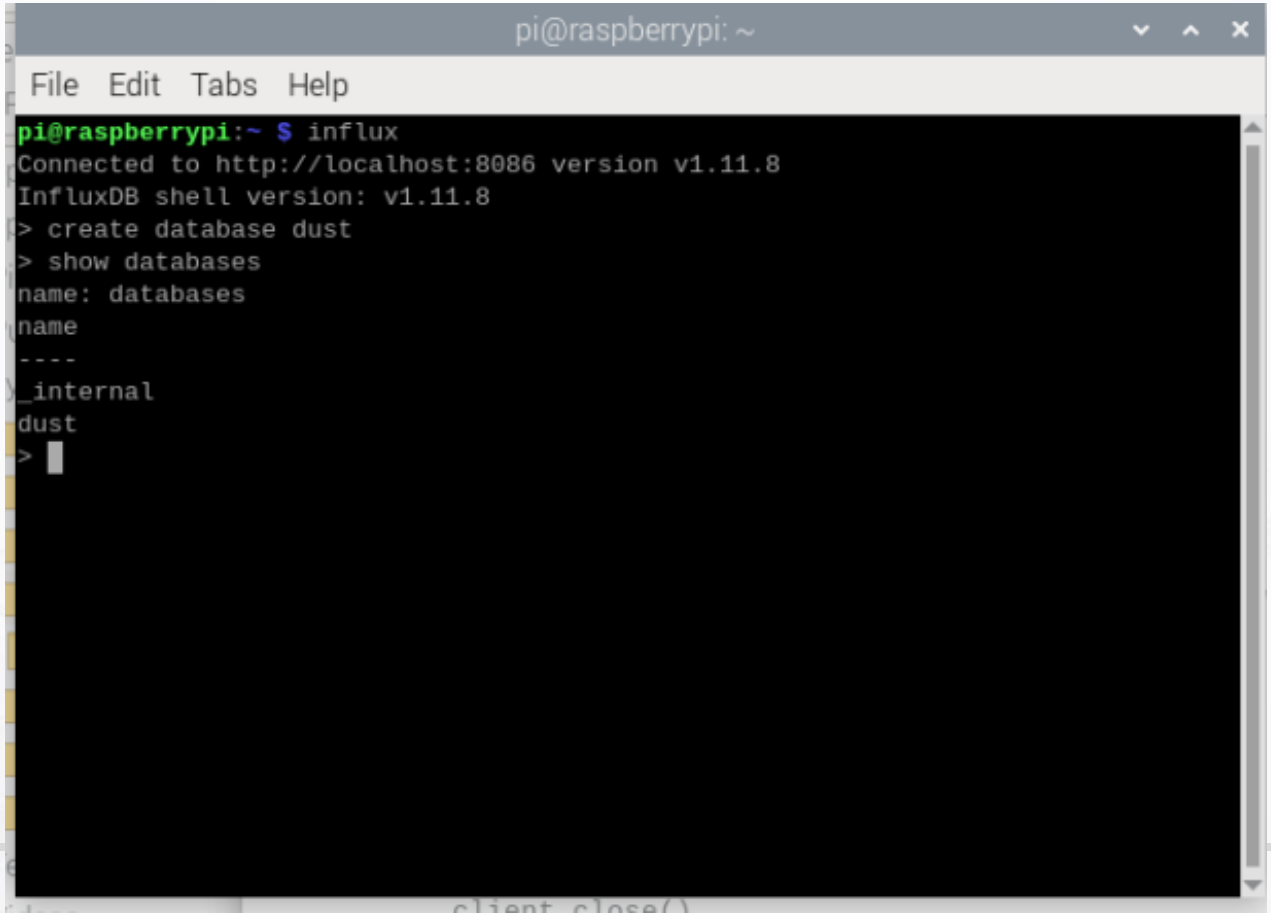
5. Run the following command to server start

```
sudo systemctl start grafana-server
```

오픈 소스 GUI 설치

influxDB Database 생성

$ Influx
$ create database dust
$ show databases
$ exit



오픈 소스 GUI 설치

Arduino to Python Serial Middleware

$ vim dustInfluxdb.py

```python


import time
import requests, json
from influxdb import InfluxDBClient as influxdb
import serial

seri = serial.Serial('/dev/ttyACM0', baudrate = 9600, timeout = None)


while(True):
    time.sleep(1)
    if seri.in_waiting !=0:
      content = seri.readline()
      a = float(content.decode())
      data = [{
          'measurement' : 'dust',
          'tags':{
              'InhaUni' : '2222',
          },
          'fields':{
              'dust' : a,
          }
      }]
```

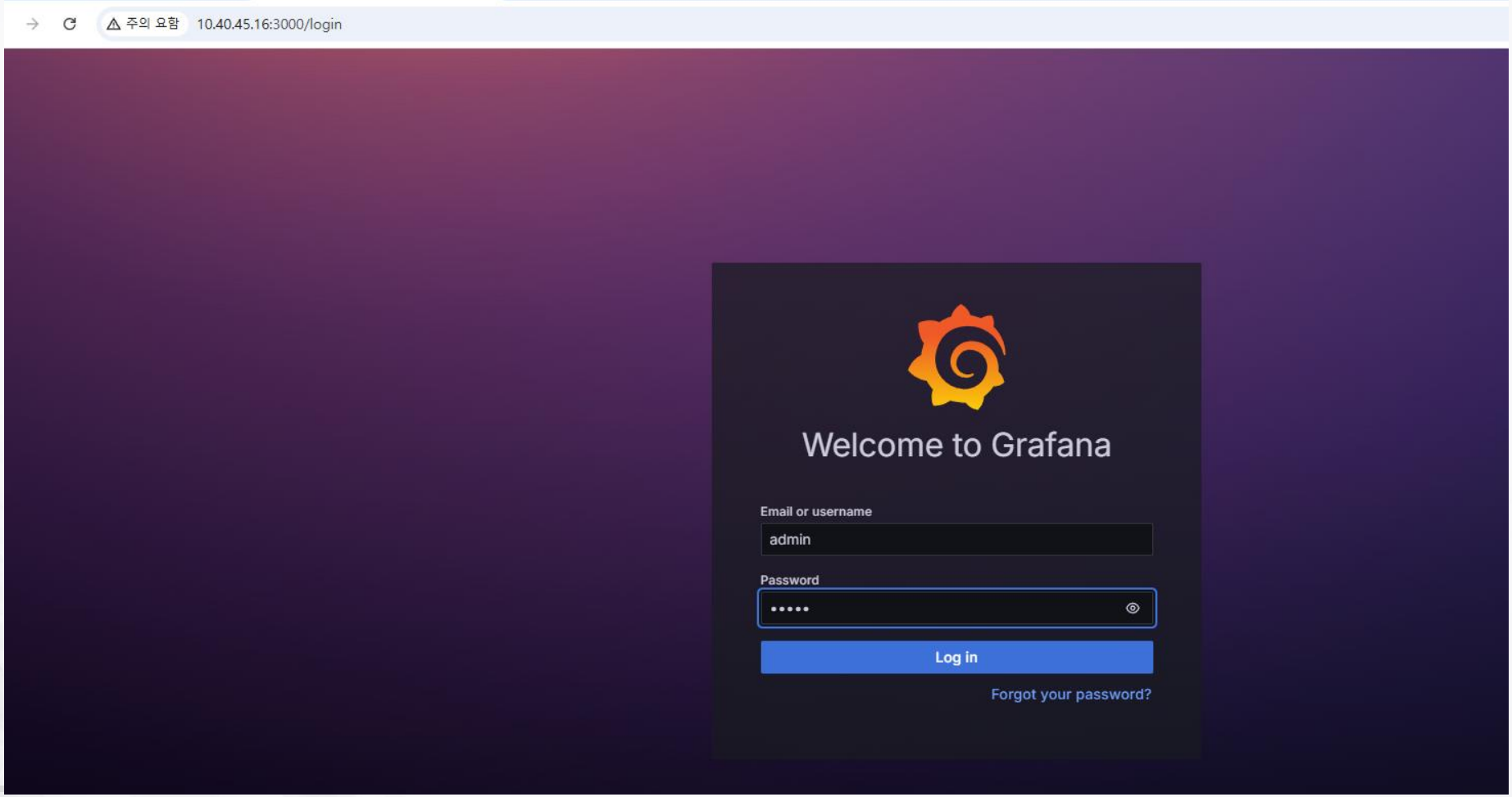
Arduino to Python Serial Middleware - 1

```
25        client = None
26        try:
27            client = influxdb('localhost',8086,'root','root','dust')
28        except Exception as e:
29          print ("Exception" + str(e))
30        if client is not None:
31            try:
32                client.write_points(data)
33            except Exception as e:
34                print("Exception write " + str(e))
35            finally:
36                client.close()
37        print(a)
38        print("running influxdb OK")
~
~
```

오픈 소스 GUI 설치

Grafana 접속

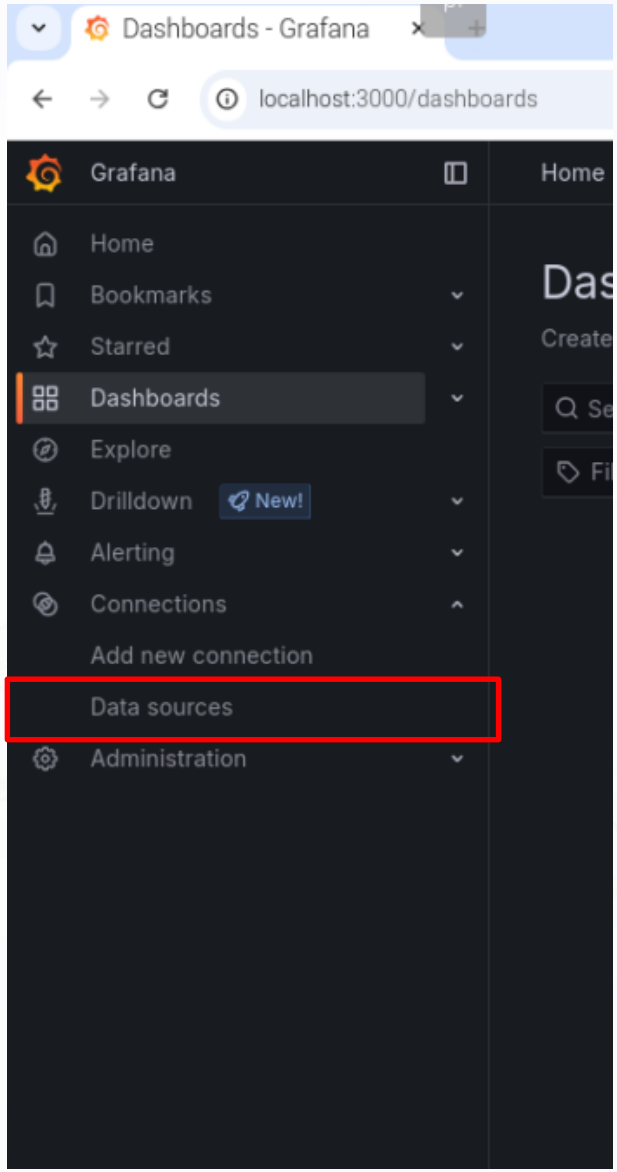
- 크롬미니 -> localhost:3000
- 기본 -> username : admin , password : admin



오픈 소스 GUI 설치

Grafana 설정

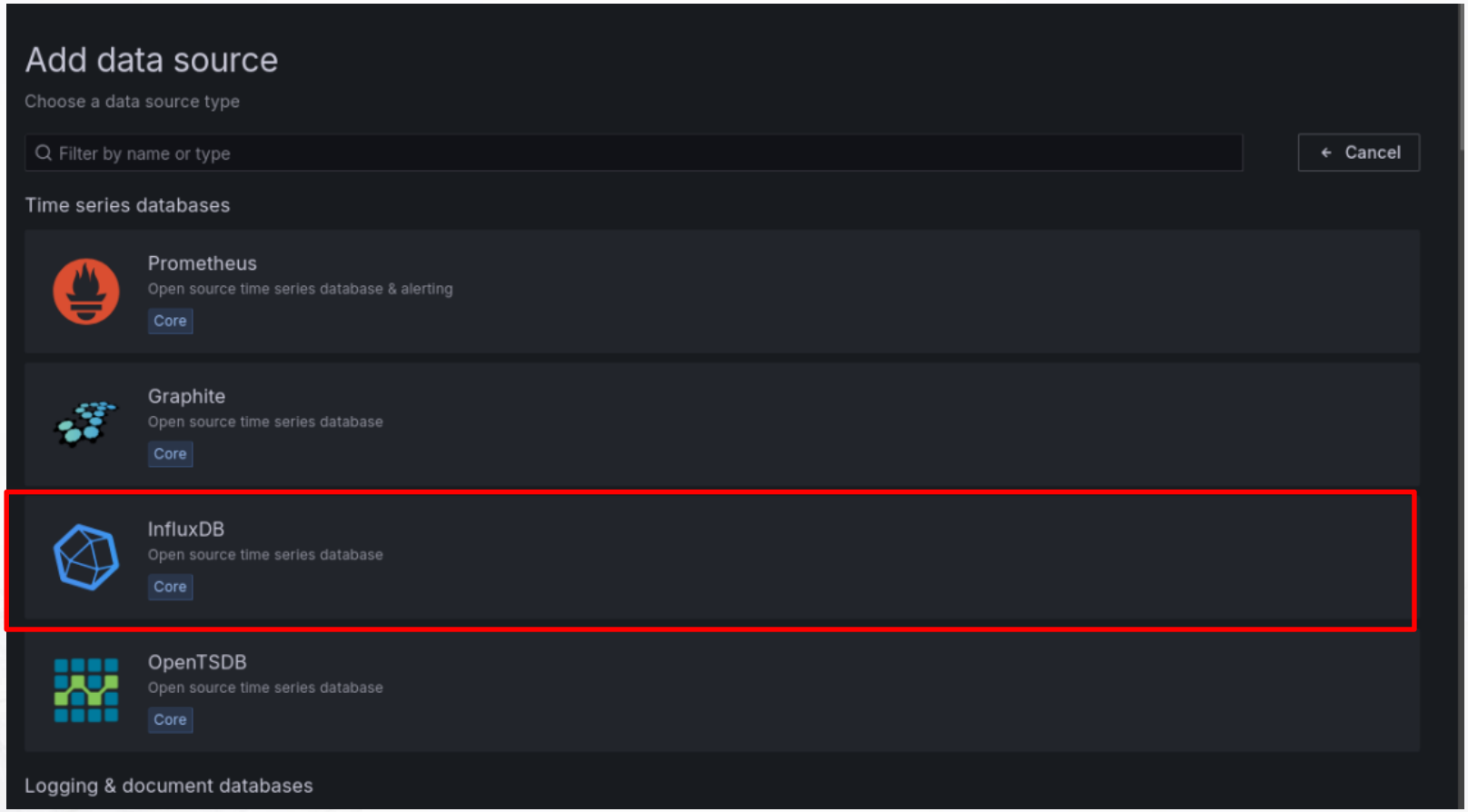
- Connections -> Data sources



오픈 소스 GUI 설치

Grafana 설정

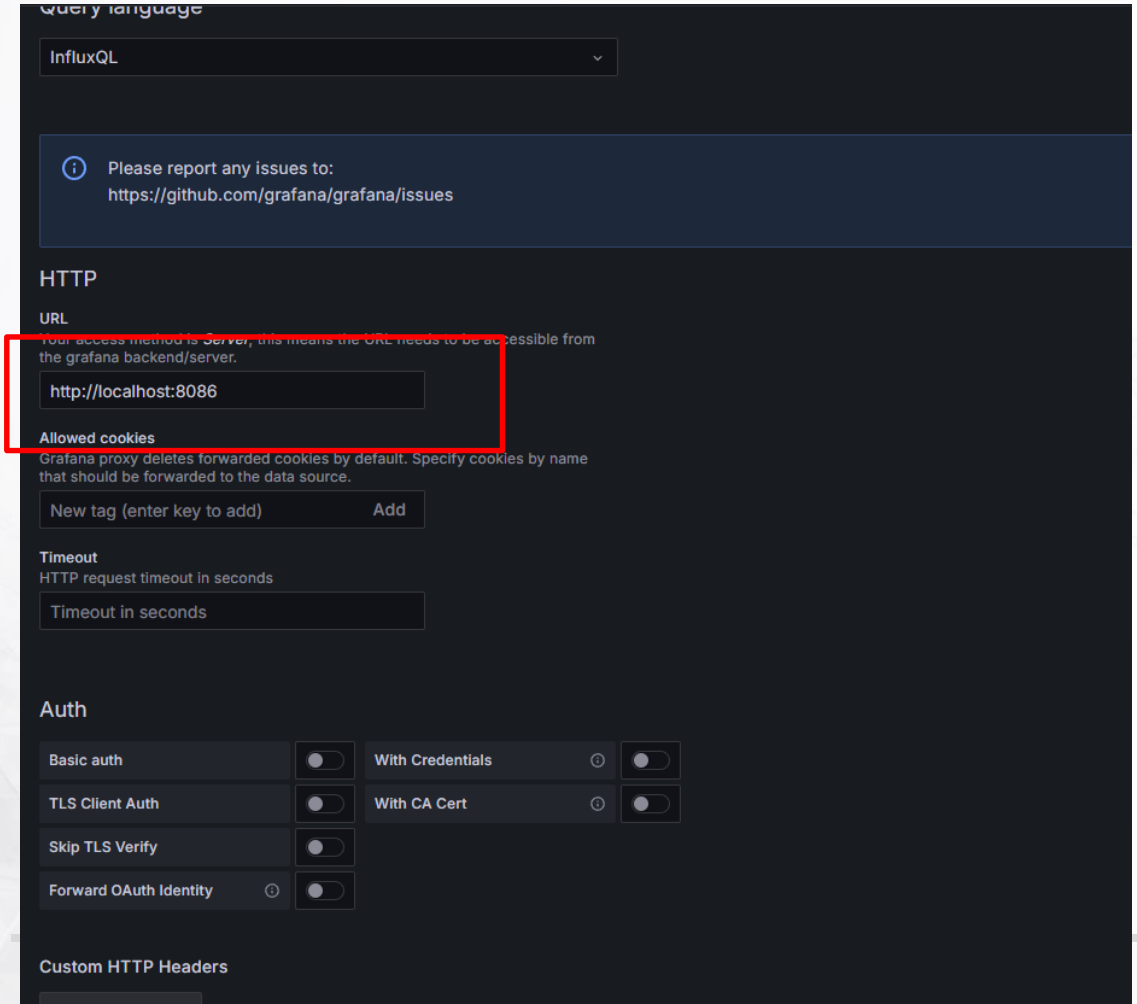
- InfluxDB



오픈 소스 GUI 설치

Grafana 설정

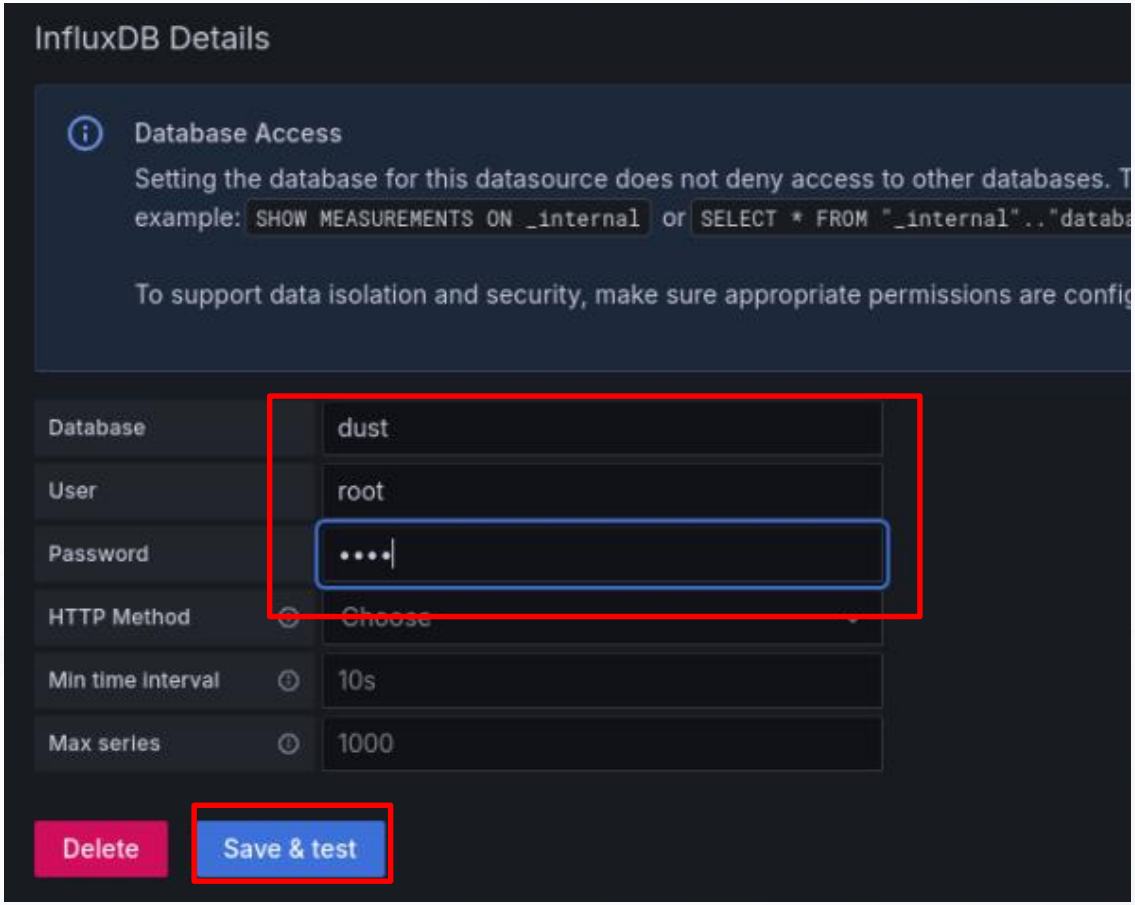
- InfluxDB 연결 설정
- http://localhost:8086



오픈 소스 GUI 설치

Grafana 설정

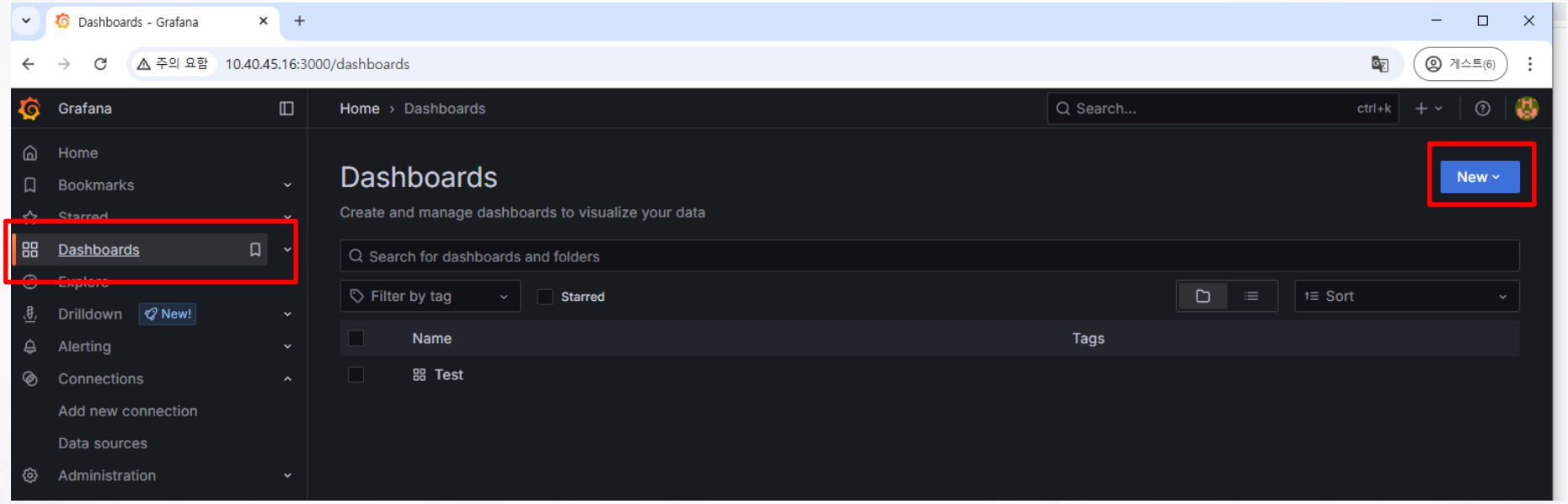
- InfluxDB 연결 설정
- Database : dust
- User : root, password: root

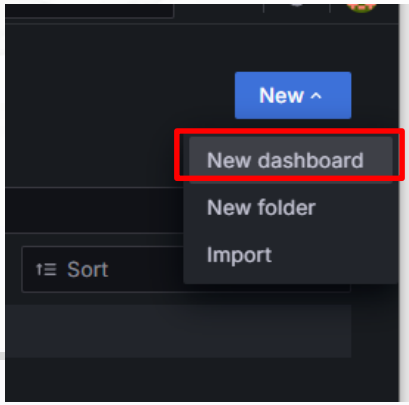


오픈 소스 GUI 설치

Grafana 설정

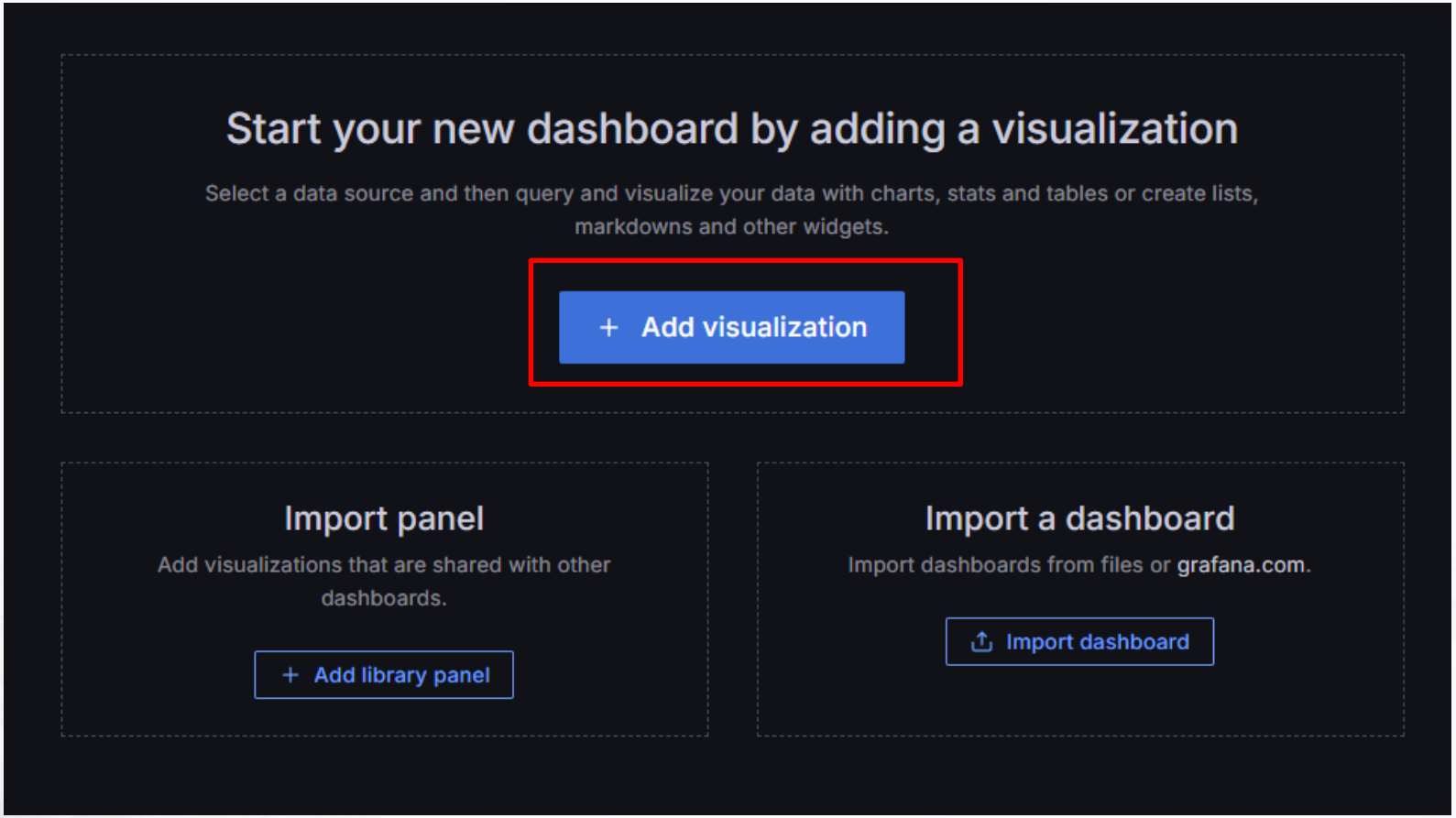
- Dashboards 설정





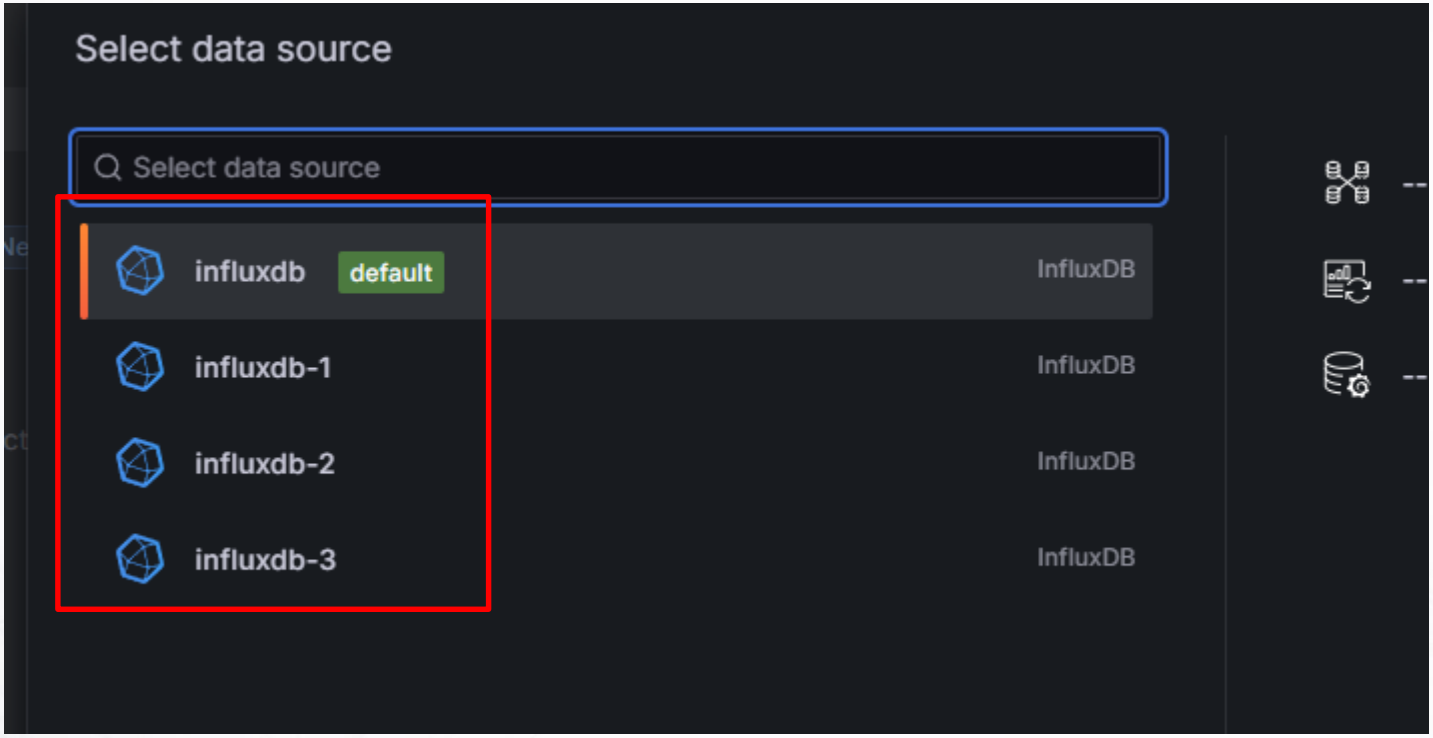
Grafana 설정

- Dashboards 설정 - 1



Grafana 설정

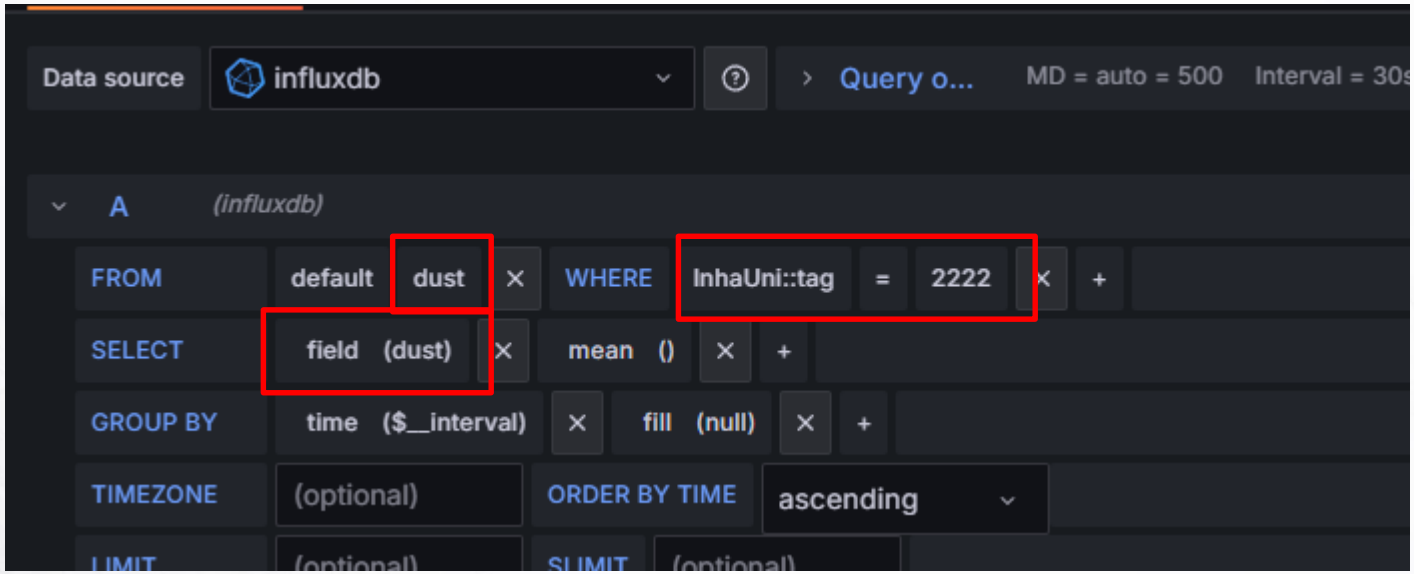
- Dashboards 설정 – 2
- 본인이 설정한 data source 지정



오픈 소스 GUI 설치

Grafana 설정

- Dashboards 설정 – 2
- select measurement -> dust
- tag -> inhaUni = 2222
- field(value) ->dust



Arduino to Python Serial using TelegramBot

- **handler 등록**

```
def main() -> None:
    """Run bot."""
    # Create the Application and pass it your bot's token.
    application = Application.builder().token("1331886552:

    # on different commands - answer in Telegram
    application.add_handler(CommandHandler(["start", "help"], start))
    application.add_handler(CommandHandler("set", set_timer))
    application.add_handler(CommandHandler("dust", dustToInfluxDB))
    application.add_handler(CommandHandler("unset", unset))
```

Arduino to Python Serial using TelegramBot

- 함수 구현

```
import cv2
import time
from telegram import Update
from telegram.ext import Application, CommandHandler, ContextTypes
from influxdb import InfluxDBClient as influxdb
import serial

seri = serial.Serial('/dev/ttyACM0', baudrate = 9600, timeout = None)

# Enable logging
logging.basicConfig(
    format="%(asctime)s - %(name)s - %(levelname)s - %(message)s", level=logging.INFO
)
async def dustToInfluxDB(context: ContextTypes.DEFAULT_TYPE) -> None:
    if seri.in_waiting !=0:
        content = seri.readline()
        a = float(content.decode())
    job = context.job
    await context.bot.send_message(job.chat_id, text=f"Dust Sensor Value! {a}")

# Define a few command handlers. These usually take the two arguments update and
# context.
# Best practice would be to replace context with an underscore,
```

Arduino to Python Serial using TelegramBot

- **set_timer 수정**

```
async def set_timer(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    """Add a job to the queue."""
    chat_id = update.effective_message.chat_id
    try:
        # args[0] should contain the time for the timer in seconds
        due = float(context.args[0])
        if due < 0:
            await update.effective_message.reply_text("Sorry we can not go back to future!")
            return

        job_removed = remove_job_if_exists(str(chat_id), context)
        context.job_queue.run_repeating(dustToInfluxDB, due, chat_id=chat_id, name=str(chat_id), data=due)

        text = "Timer successfully set!"
        if job_removed:
            text += " Old one was removed."
        await update.effective_message.reply_text(text)
```

12주차 수업이 끝났습니다

고생하셨습니다.