사물인터넷 실습 :

Raspberry Pi 기초 II

Electronics Everywhere
Network Everything
Cloud Intelligence

허윤석, 공학박사

email: yoonseok@gmail.com

Agenda

- 중간고사 문제 풀이
- Physical Computing
  - Python
  - NodeRED
- Docker & influxdb/grafana
- 학기말 고사 - 팀 프로젝트

중간고사 문제 해석 및 풀이

Lab 2 - Python

1.sudo apt install python-gpiozero python3-gpiozero

2.pinout

3.RPi.GPIO sample(switch and relay)

4.gpiozero sample

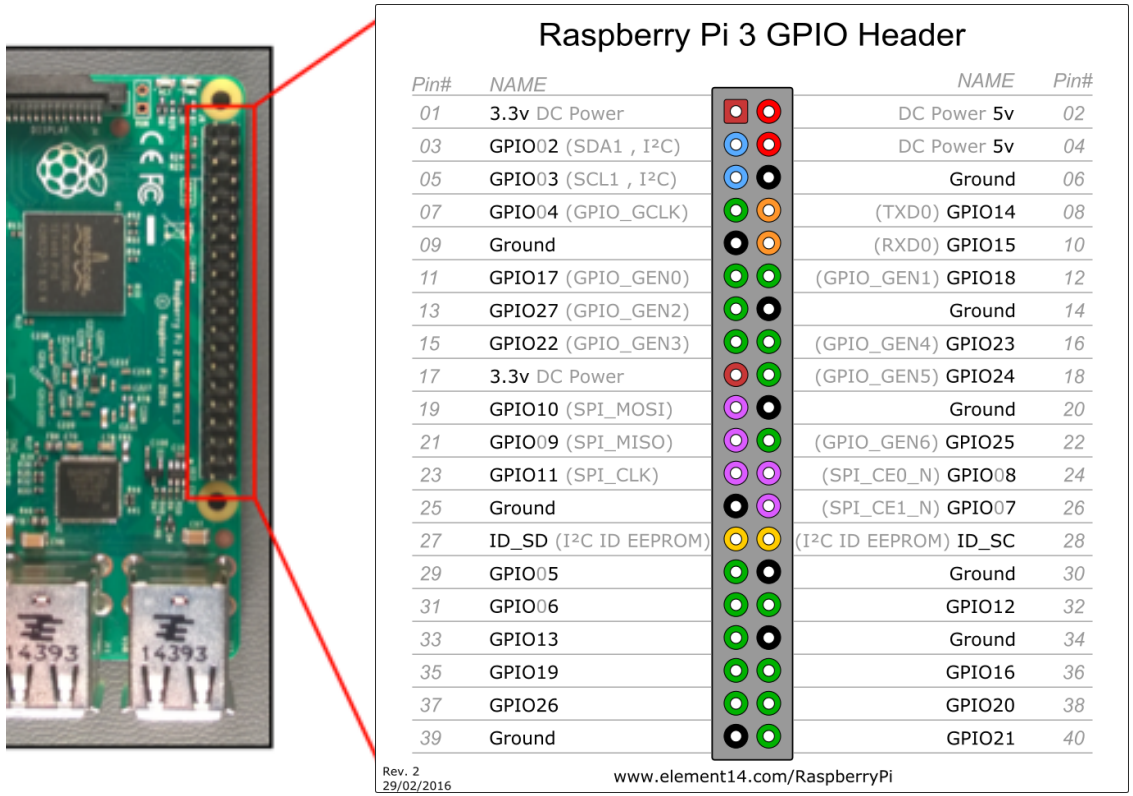
5.sudo apt install python-pip

6.pip install wiotp-sdk ibmiotf

7.Tour the GitHub page for iotf

8.Sample iot program

Hardware - Raspberry Pi Beginner's Guide



Raspberry Pi 3 GPIO Header

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

At the right-hand edge of the board you'll find 40 metal pins, split into two rows of 20 pins (Figure 1-14). This is the GPIO (general-purpose input/output) header, a feature of the Raspberry Pi used to talk to additional hardware from LEDs and buttons all the way to temperature sensors, joysticks, and pulse-rate monitors.

There are two ways to address the pins. Board number and BCM(Broadcom) number. BCM is the GPIO number.

Hands On - Python GPIO Handling

gpiozero module

```
from gpiozero import Button, LED
from signal import pause
led = LED(14)
button = Button(21)
button.when_pressed = led.on
button.when_released = led.off
pause()
```

```
from signal import pause
import gpiozero
r = gpiozero.OutputDevice(14, active_high=True, initial_value=False)
button = gpiozero.Button(21)
button.when_pressed = r.on
button.when_released = r.off
pause()
```

RPi.GPIO module

BCM(GPIO) Pin Numbering

```
import RPi.GPIO as g
g.setmode(g.BCM) # well know GPIO port
g.setup(14, g.OUT)
g.output(14, g.HIGH)
g.output(14, g.LOW)
```

Board Pin Numbering

```
import RPi.GPIO as g
g.setmode(g.BOARD) # the board's pin number
g.setup(8,g.OUT)
g.output(8,g.HIGH)
g.output(8,g.LOW)
```

Lab 2 - IBM IOTF Python Module

```
import wiotp.sdk
import RPi.GPIO as g
from signal import pause

deviceOptions = {
    "identity": {"orgId": "ooo", "typeId": "RPi", "deviceId": "iotDev1"},
    "auth": {"token": “rpi11111"},
}
data = {
    "d" : {
    }
}

def commandProcessor(cmd):
    print(cmd.data["d"])
    if cmd.data["d"]["lamp"]:
        if cmd.data["d"]["lamp"] == "on":
            g.output(14, g.HIGH)
            data["d"]["lamp"] = "on"
        else:
            g.output(14, g.LOW)
            data["d"]["lamp"] = "off"
        deviceCli.publishEvent("status", "json", data, qos=0)

g.setmode(g.BCM)
g.setup(14, g.OUT)
deviceCli = wiotp.sdk.device.DeviceClient(deviceOptions)
deviceCli.commandCallback = commandProcessor
deviceCli.connect()
pause()
```

Hands On - IBM IOTF Python Module

Modify the previous IBM IOT Device python program to report the GPIO port 14 every 10 seconds.

Hint :

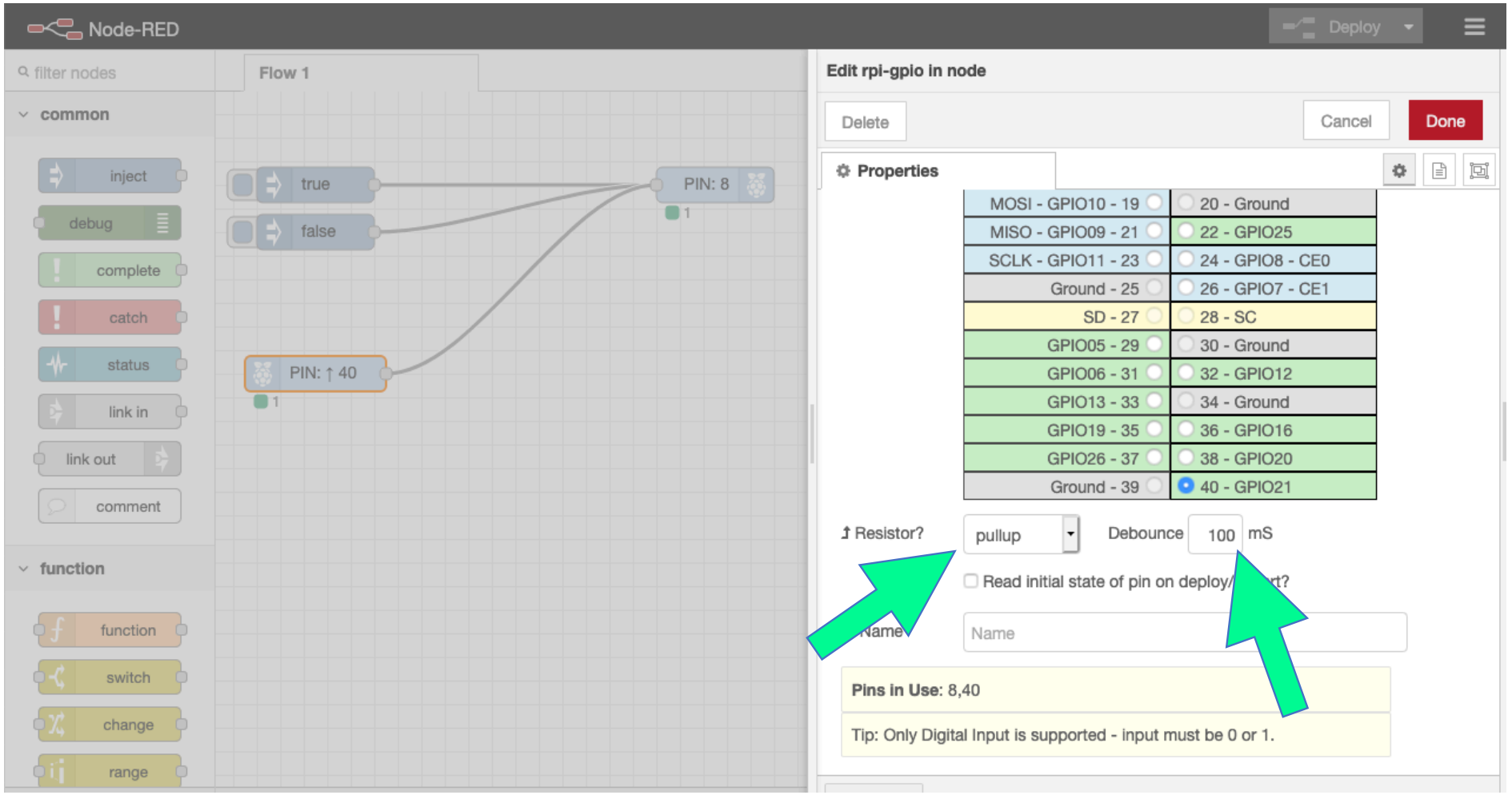
Replace `pause()` with `while True:` loop and implement the loop

The loop needs to check the GPIO port, build the message, and publish

Hands On - NodeRED

1. bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
2. sudo systemctl start nodered.service
3. sudo systemctl enable nodered.service
4. NodeRED GPIO program
5. Install node-red-contrib-scx-ibmiotapp on the NodeRED

Hands On - NodeRED GPIO Handling



Lab 3 - NodeRED GPIO Handling

Run the modified IBM IOT Python program

Install node-red-contrib-scx-ibmiotapp on the NodeRED

Send commands to turn on/off the Python device

Lab 4 - Docker & influxdb/grafana

influxDB/Grafana Review & Lab Steps

1. `curl -fsSL get.docker.com -o get-docker.sh && sh get-docker.sh`
2. `sudo apt install docker-compose`
3. `sudo usermod -aG docker pi && sudo systemctl restart containerd`
4. logoff and login again
5. `unzip influx.zip && cd influx`
6. run `docker-compose up`
7. `docker exec -it influxdb influx`

   `create database sensorDB`
8. `nohup sys.monitor &`

InfluxDB

InfluxDB is an open-source time series database (TSDB) developed by InfluxData. It is written in Go and optimized for fast, high-availability storage and retrieval of time series data in fields such as operations monitoring, application metrics, Internet of Things sensor data, and real-time analytics. It also has support for processing data from Graphite.

https://en.wikipedia.org/wiki/InfluxDB

InfluxDB key concepts

https://docs.influxdata.com/influxdb/v1.7/concepts/key_concepts/

database	**field key**	**field set**
field value	**measurement**	**point**
retention policy	**series**	**tag key**
tag set	**tag value**	**timestamp**

InfluxDB & Grafana

Influxdb insertion uses line protocol

https://docs.influxdata.com/influxdb/v1.5/write_protocols/line_protocol_tutorial/

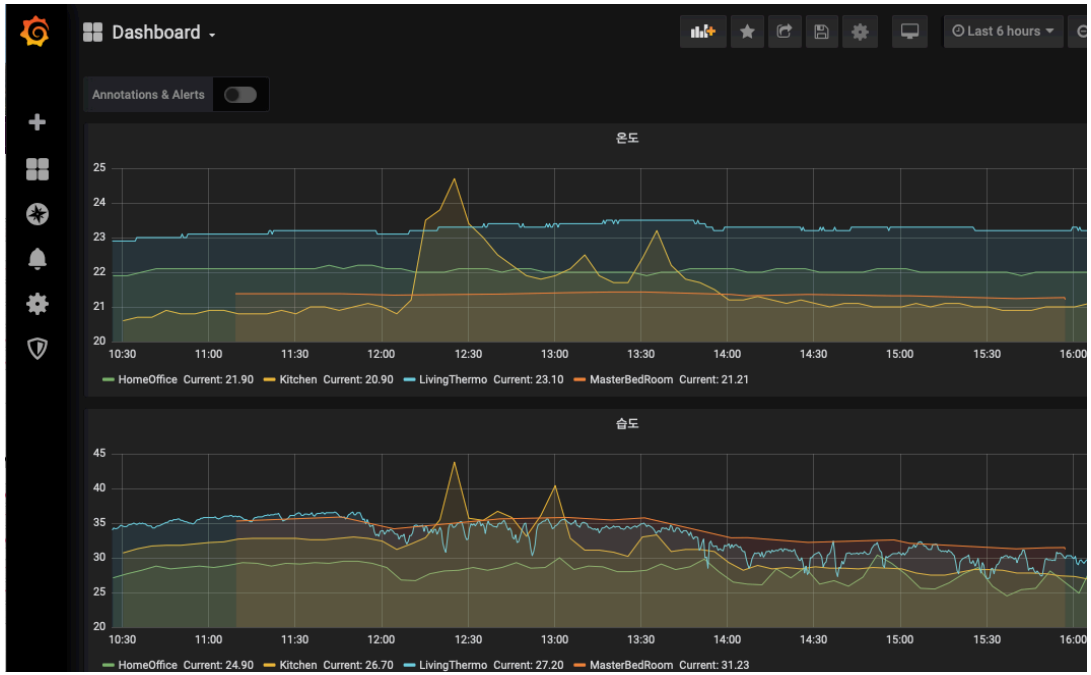
line protocol is the specification of the single-line data format for the data insertion to influxdb. One line consists of measurement, tag set, field set, and the timestamp. They are separated by a blank as follows.

```
weather,location=us-midwest<b>temperature=82<b>14658…

   Measurement name : weather
   Tag set          : location=us-midwest
   Field set        : temperature=82
   Time stamp       : 14658…..
```

InfluxDB & Grafana

Grafana is an open-source, general purpose dashboard and graph composer, which runs as a web application. It supports graphite, **InfluxDB** or opentsdb as backends.

From : https://wiki.archlinux.org/index.php/Grafana



Docker Compose

```
influxdb:
  image: influxdb:latest
  container_name: influxdb
  ports:
    #- "8083:8083"
    - "8086:8086"
    - "8090:8090"
  env_file:
    - 'env.influxdb'
  volumes:
    # Data persistency
    - ./influxdb/data:/root/.influxdb
    - ./influxdb/etc:/etc/influxdb/:ro

grafana:
  image: grafana/grafana:latest
  container_name: grafana
  ports:
    - "3000:3000"
  env_file:
    - 'env.grafana'
  links:
    - influxdb
```

Docker-Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

Docker Compose Hands on

Influxdb/Grafana Docker setup

```
mkdir influxlab
cd influxlab

copy & paste
        docker-compose.yml
        env.grafana
        env.influxdb
        influxdb.conf
mkdir -p influxdb/data
mkdir -p influxdb/etc
cp influxdb.conf influxdb/etc

docker-compose up
```

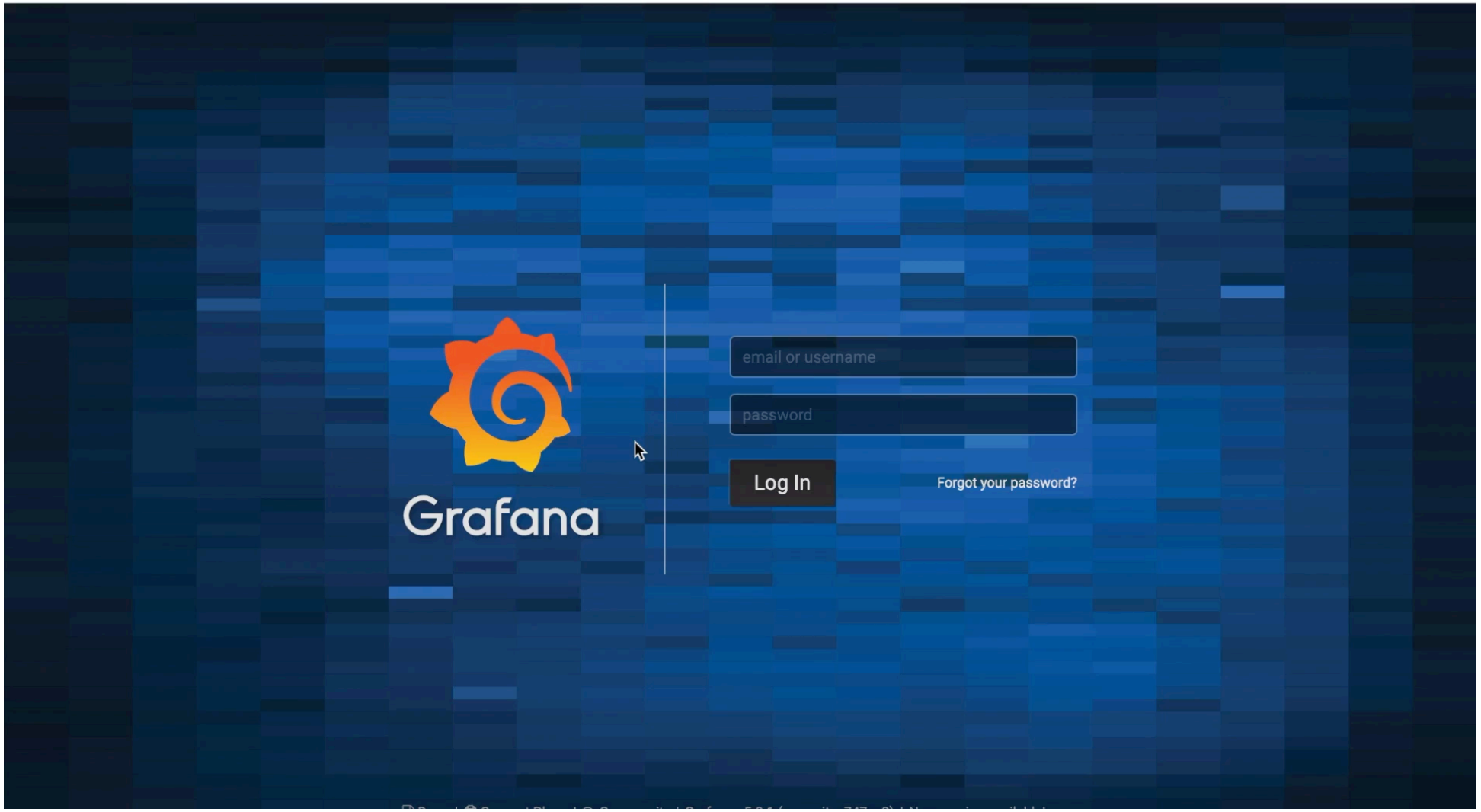
influxdb setup

```
docker exec -it influxdb /bin/bash
root@33452556 : influx
> CREATE USER admin WITH PASSWORD 'yourPassword' WITH ALL PRIVILEGES
> auth
> username: admin
> password:
> create database mydb
> CREATE USER iot WITH PASSWORD ‘iot'
> grant all on mydb to iot
```

Influxdb Hands On

```
pi@iotpi4:~ $ docker exec -it influxdb influx
Connected to http://localhost:8086 version 1.8.0
InfluxDB shell version: 1.8.0
> show databases
name: databases
name
----
_internal
sensorDB
> use sensorDB
Using database sensorDB
> show measurements
name: measurements
name
----
cpu
> select * from cpu
name: cpu
time                host   idle mem sys  user
----                ----   ---- --- ---  ----
1589115171615568622 influx 95.8 230 1.4  2.8
1589115177114919944 influx 94.7 231 4    1.3
1589115182448869285 influx 95.9 231 4.1  0
```

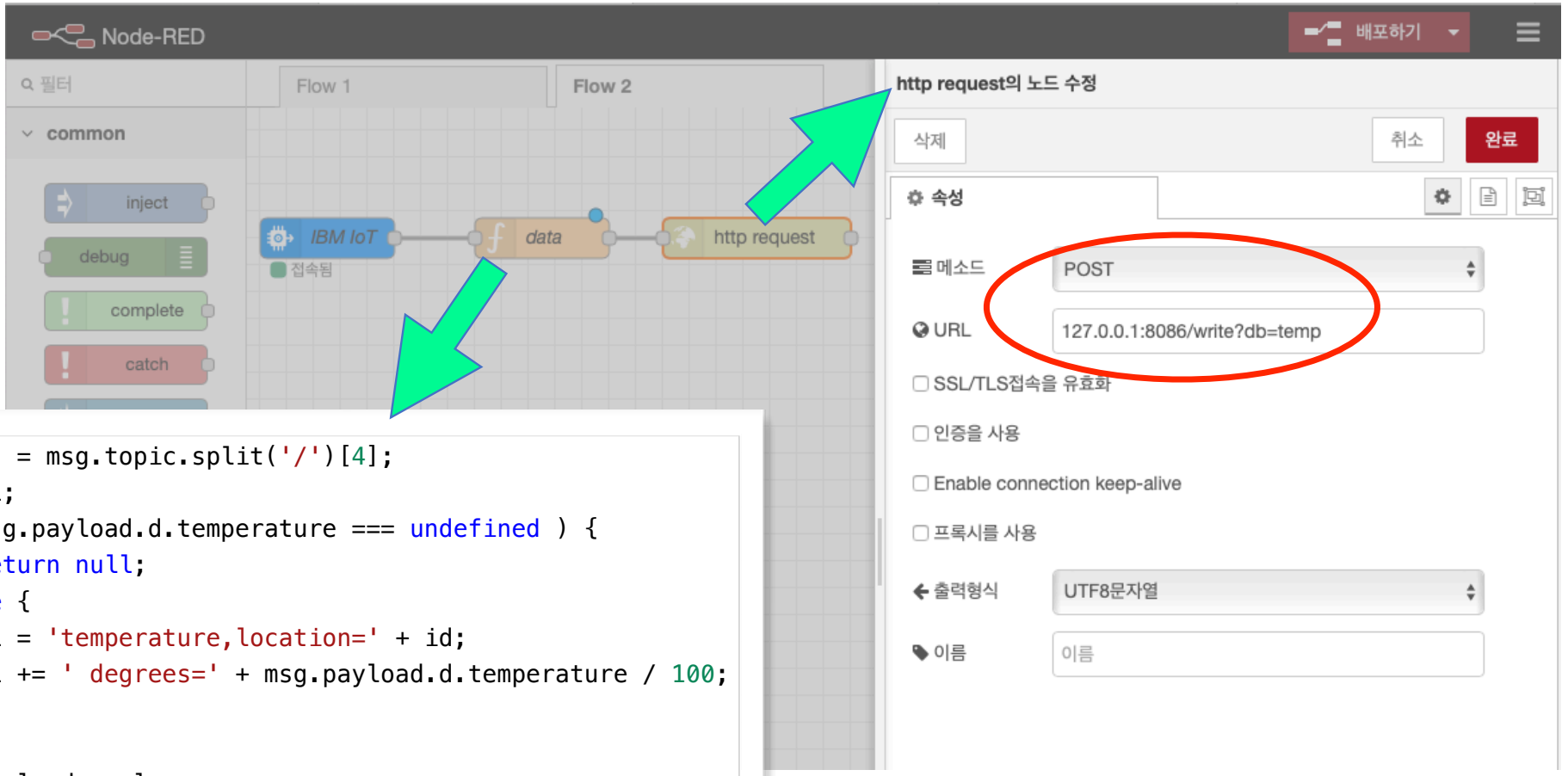
Configure Grafana Dashboard



Lab 5 - Docker & influxdb/grafana

- Run the Board A with IOT Thermometer
- Create a Flow that inputs the temperature to influxdb
- Create the Temperature Grafana Dashboard for

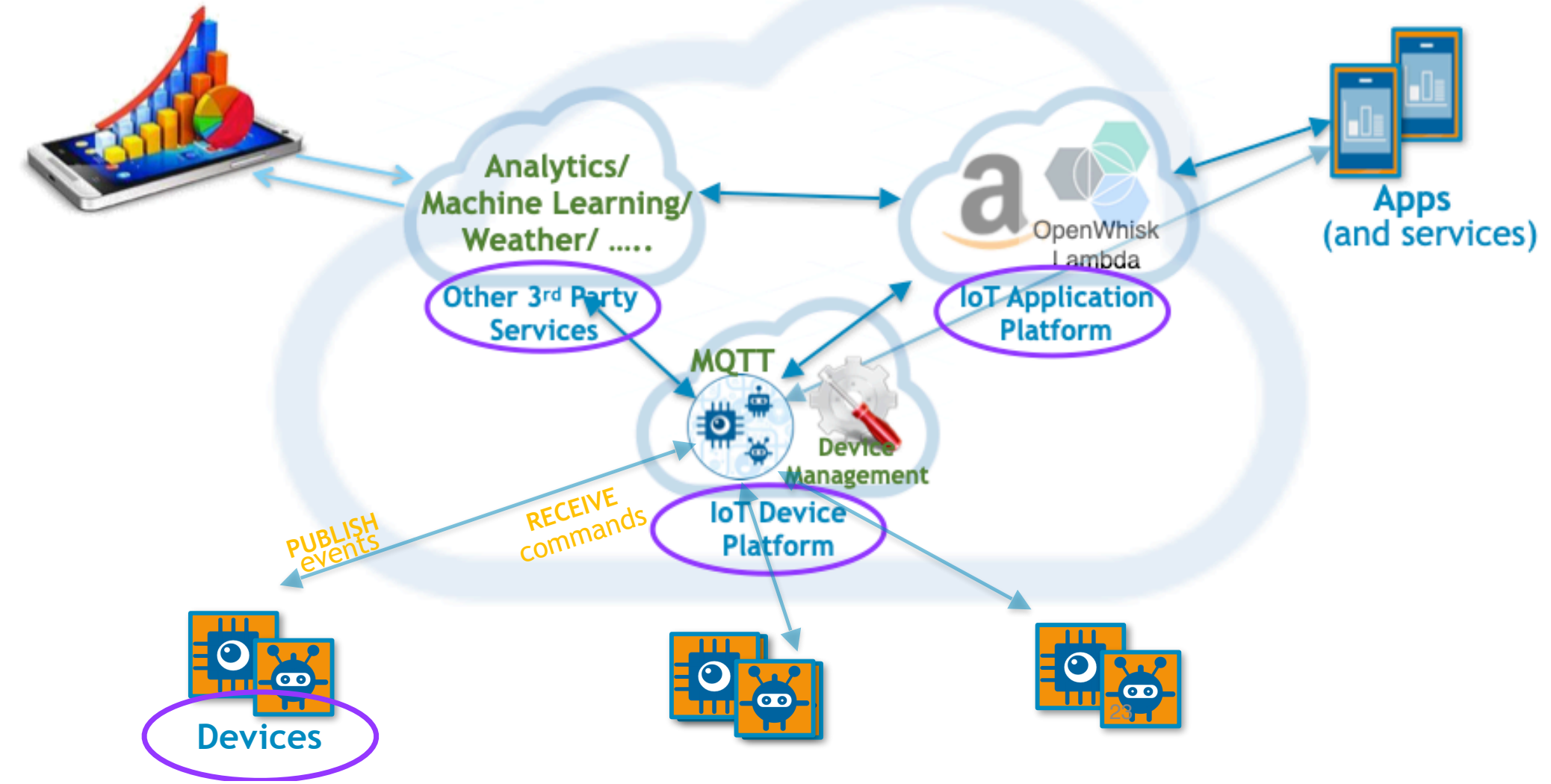
Lab 5 - Docker & influxdb/grafana



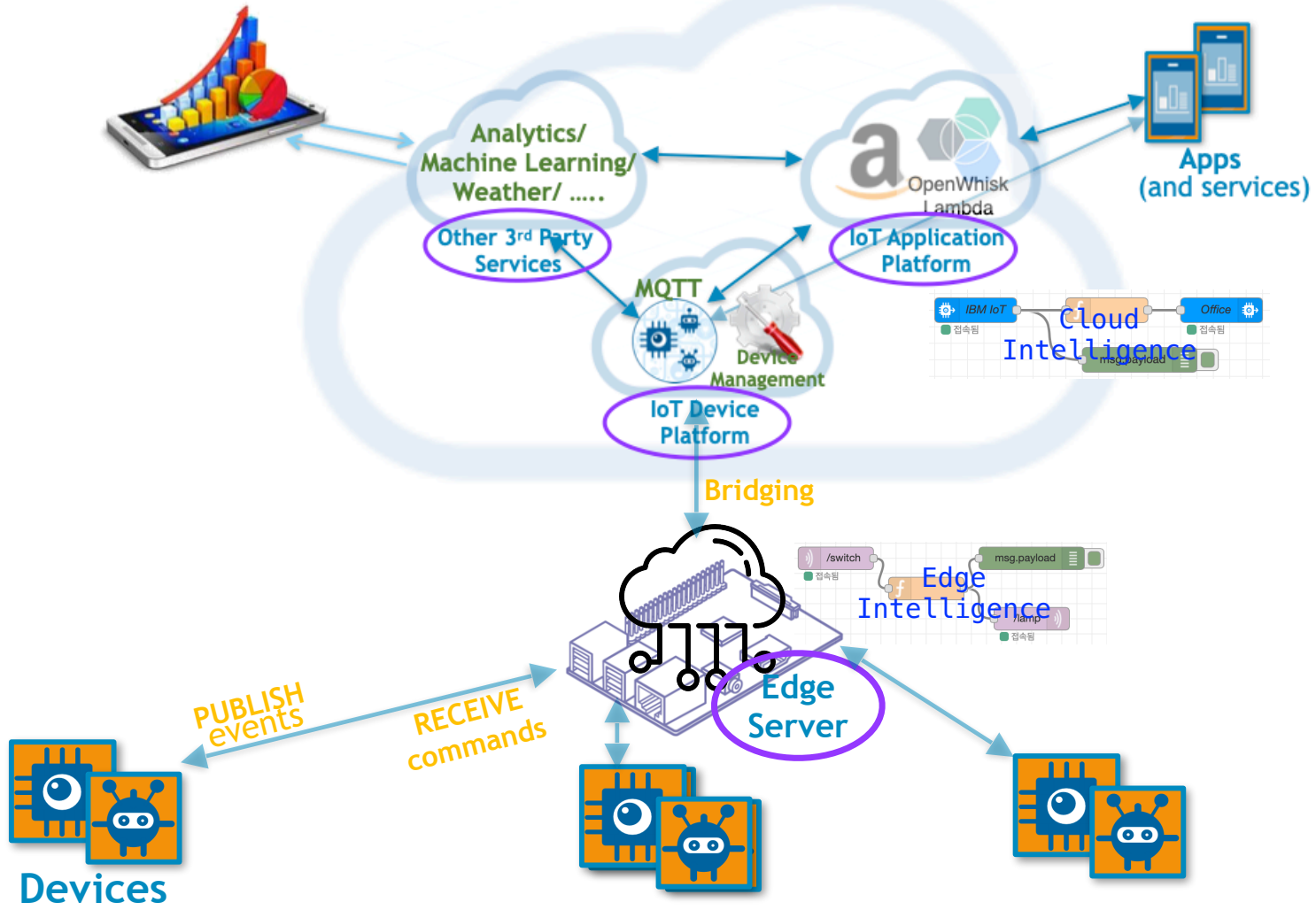

```
var id = msg.topic.split('/')[4];
var ql;
if (msg.payload.d.temperature === undefined ) {
    return null;
} else {
    ql = 'temperature,location=' + id;
    ql += ' degrees=' + msg.payload.d.temperature / 100;
}

msg.payload = ql;

return msg;
```

학기말 고사 프로젝트

Internet of Things Architecture : Learned so far



Internet of Things Architecture with Edge Server : To Learn from now on



Final Term Exam Project

- **Team of 4 persons**
  - 조교 제외한 7 팀, 자발적 구성하여, 이번주 금요일(**5/15**)까지 제출
  - 다음주 금요일(**5/22**)까지 프로젝트 개요(팀명, 프로젝트명, 짧은 프로젝트 설명)
- **Mission : AnyIOT 2 with Edge Computing**
  - Must use esp8266 boards, Edge, Cloud
  - Use both Cloud and Edge Intelligence
- **Final Term Exam Presentation**
  - Problem Definition
  - Solution Definition
  - Architecture & Technologies
  - Reason of the Intelligence Placement
  - Live Demo
  - Limitation and Future Improvements
- **Evaluation Points**
  - Creativity & IOT Skills/Knowledge
  - Team Work
  - Completeness

Backup

Influxdb & Grafana Docker-compose configuration files

Docker-compose.yml

```
influxdb:
  image: influxdb:latest
  container_name: influxdb
  ports:
    #- "8083:8083"
    - "8086:8086"
    - "8090:8090"
  env_file:
    - 'env.influxdb'
  volumes:
    # Data persistency
    - ./influxdb/data:/root/.influxdb
    - ./influxdb/etc:/etc/influxdb/:ro

grafana:
  image: grafana/grafana:latest
  container_name: grafana
  ports:
    - "3000:3000"
  env_file:
    - 'env.grafana'
  links:
    - influxdb
```

influxdb.conf

```
[meta]
  dir = "/var/lib/influxdb/meta"

[data]
  dir = "/var/lib/influxdb/data"
  engine = "tsm1"
  wal-dir = "/var/lib/influxdb/wal"

[http]
  enabled = true
  bind-address = ":8086"
  auth-enabled = true
```

env.grafana

```
GF_INSTALL_PLUGINS=grafana-clock-panel,briangann-gauge-
panel,natel-plotly-panel,grafana-simple-json-datasource
```

env.influxdb

```
INFLUXDB_DATA_ENGINE=tsm1
INFLUXDB_REPORTING_DISABLED=false
```

Influxdb Client

Python sample

```
from influxdb import client as influxdb
db = influxdb.InfluxDBClient('127.0.0.1', 8086,'iot', 'iot', 'mydb')

db.write_points([{"measurement": "cpu", 'tags':{"host": "server01", "region":"us-north"}, 'fields':{'value':0.5}}])

result=db.query('select * from cpu')
print(result)
```

Web sample (curl)

```
curl -i -XPOST 'http://localhost:8086/write?db=mydb' --data-binary 'cpu,host=server01,region=us-west value=0.64'
```