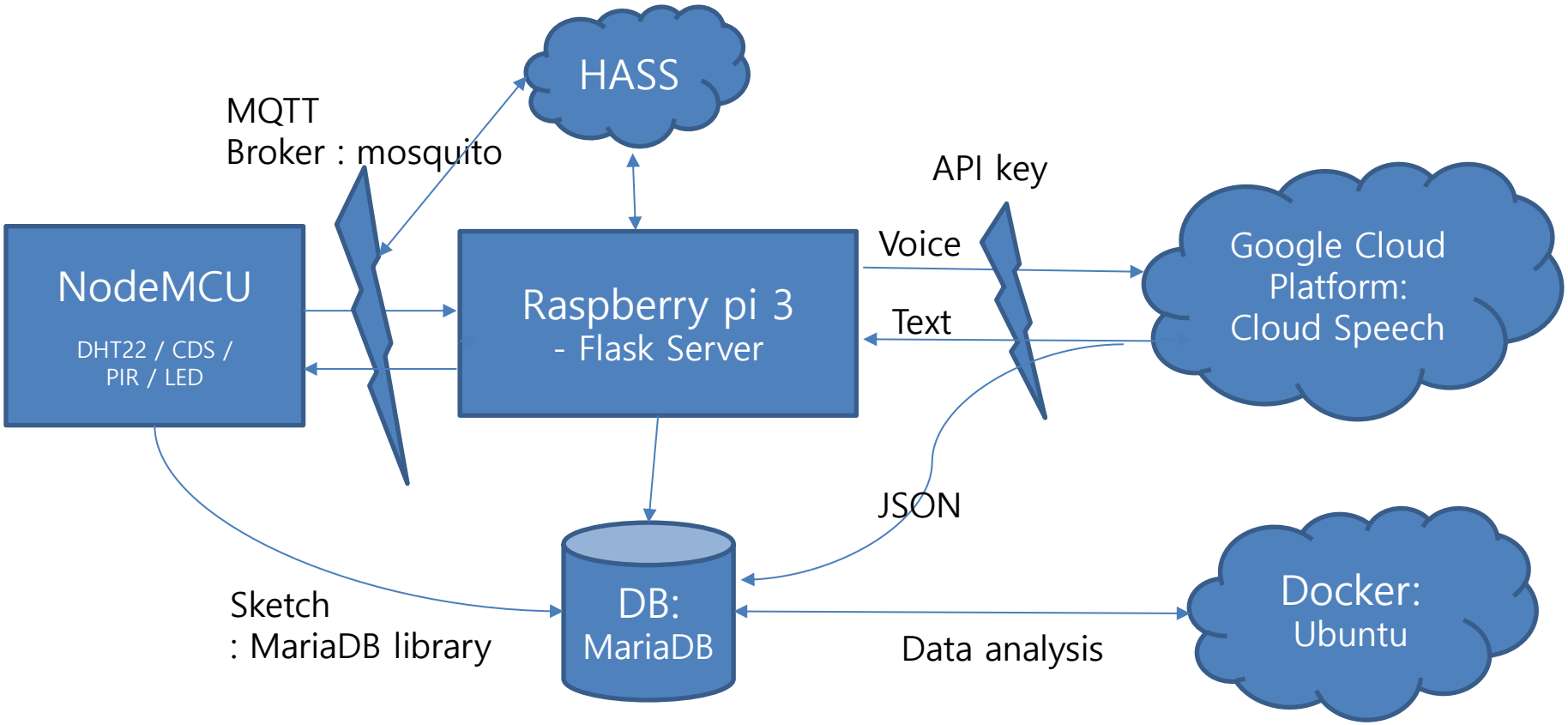
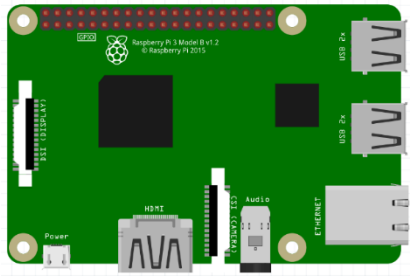
Overview



NodeMCU Overview

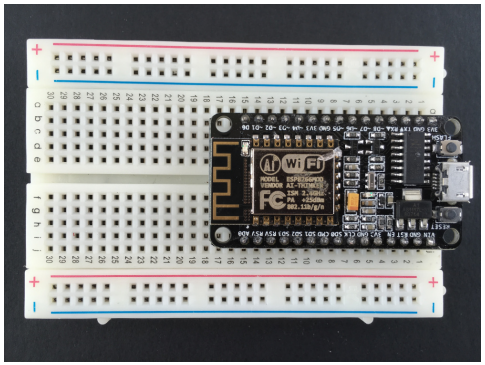
<Flask 서버, MQTT Broker>



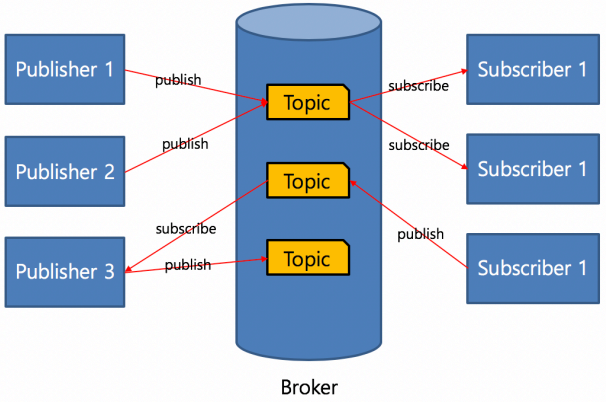
Publish/Subscriber
센서 값 전달

mosquitto

Publish/Subscriber
명령



<HASS>



MariaDB

<Database>

센서 값 입력 받기

❑ Receive Sensor data

```
#include <DHT.h>;

//pin 지정
#define DHTPIN D2
#define DHTTYPE DHT22
#define D0 16
#define D1 5
#define LIGHT_PIN A0
#define LED_PIN D0
#define RELAY1_PIN D1
#define RELAY_OFF HIGH  // HIGH 신호이면 릴레이 동작하지 않음
#define RELAY_ON LOW    // LOW 신호이면 릴레이 동작함

#define PIR_PIN D3

//센서 초기상태 지정
int relay_state = RELAY_OFF; // REALY 초기설정 : OFF
int pirState = LOW; // PIR 초기 설정
```

❑ Receive Sensor data

```
// Relay ON & OFF Flags
boolean body = false;
boolean tem_hum = false;
boolean light = false;

DHT dht(DHTPIN, DHTTYPE);

//변수 설정
float temperature;
float humidity;
int lightValue;
int pir_value;

void setup() {
        Serial.begin(9600);

        //Initialized DHT sensor
        dht.begin();
```

❑ Receive Sensor data

```
        pinMode(PIR_PIN,INPUT); //인체감지 센서 입력 단자
        pinMode(LIGHT_PIN, INPUT); // 조도센서 입력단자
        pinMode(LED_PIN, OUTPUT);  // Initialize the LED pin as an output
        pinMode(RELAY1_PIN, OUTPUT); // 릴레이 signal 단자

        digitalWrite(LED_PIN, LOW);  //LED off
        digitalWrite(RELAY1_PIN, RELAY_OFF); // 릴레이 OFF
}

void loop() {
delay(1000);

//RELAY CONTROL CONDITION
// 1. Temperature & Humidity
temperature = dht.readTemperature();
humidity = dht.readHumidity();
```

```
// 2. Body Detect
pir_value = digitalRead(PIR_PIN);
if(pir_value == HIGH) { //인체감지 시
        body = true;
        Serial.println("Motion detected!");
        if(pirState = LOW) {
                pirState = HIGH;
                client.publish(PIRSensor_State, "ON", true);
                delay(500);
        }
}

else {
        body = false;
        Serial.println("Motion Ended!");
        if(pirState == HIGH) {
                pirState = LOW;
                client.publish(PIRSensor_State, "OFF", true);
                delay(500);
        }
}
```

```
// 3. Light detect
        lightValue = analogRead(LIGHT_PIN);   // CDS 값 읽어 오기
        if( lightValue > 650 ) {   // 빛의 밝기가 어두워지면
                light = true;
        }

        else if (lightValue < 500) {
                light = false;
        }
         digitalWrite(RELAY1_PIN, relay_state);   // relay OFF
         Serial.println(lightValue);
```

❑ Flag control

```
        if(body == true && light == true){
                relay_state = RELAY_ON;   //어둡고 사람이 있으면
        }

        else if(body == true && light == false){
                relay_state = RELAY_OFF;   //밝고 사람이 있으면
        }

        else if(body ==false && light ==false){
                 relay_state = RELAY_OFF; // 밝고 사람도 없으면
        }

        else {
                relay_state = RELAY_OFF;
        }
}
```

HASS

Home Assistant

❑ 1. HASS @ Raspberry Pi

- RPI3에 HASS 설치
- HASS와 NodeMCU MQTT를 활용한 연결 (각 센서 값 subscribe)
- Dashboard에 센서 값 받아와서 보여주기
- Dashboard에 외부 IP 사용하기

Foreign API

- 그리고, sensors.yaml 이라는 파일을 만들어 설정을 해주면 그대로 configuration.yaml에 반영이 되어 나타난다.
- Sensors.yaml에서 설정 해줄 것은,

```
- platform: darksky
  api_key:
  monitored_conditions:
    - summary
    - temperature
    - humidity
    - pressure
    - visibility
    - ozone
```

위와 같다.

Home Assistant Configuration

- 1. Configuration.yaml 수정
  - Mqtt 설정
  - Sensor에 대한 내용은 sensors.yaml의 내용을 참고한다고 설정
- 2. Sensors.yaml 수정
  - 온도, 습도, 조도 센서 값을 받아오는 모듈 코딩

Configuration.yaml MQTT 설정

- Home assistant가 설치가 되어 있다면, ~/.homeassistant 경로에서 configuration을 수정 할 수 있다.
- $cd ~/.homeassistant
- $vim configuration.yaml에서

```
# mqtt
mqtt:
  broker: 203.252.106.154
  username: iot
  password: !secret mqtt_password
  protocol: 3.1.1
```

- Mqtt 설정을 해준다.

- 또한, sensor에 대한 항목은 sensors.yaml에 대한 부분을 사용하겠다고 적어준다.

```
sensor: !include sensors.yaml
```

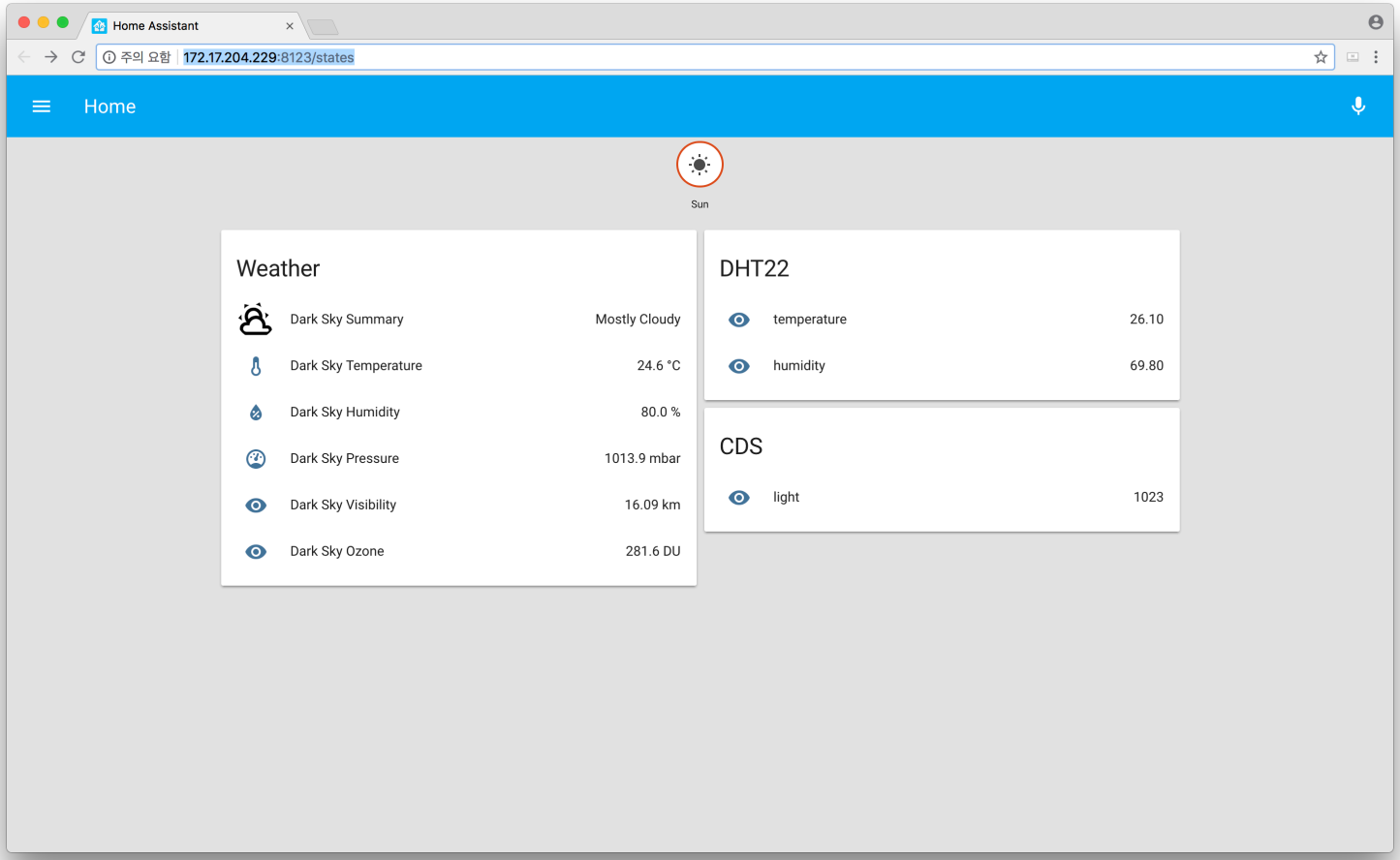
- sensors.yaml 에서 mqtt에서 센서 값을 받아오는 내용을 작성한다.
- $vim sensors.yaml

```
- platform: mqtt
  state_topic: "iot/21400357/dht22_t"
  name: temperature

- platform: mqtt
  state_topic: "iot/21400357/dht22_h"
  name: humidity

- platform: mqtt
  state_topic: "iot/21400357/light"
  name: light
```

- Groups.yaml에서 해당 내용을 그룹화하여 카드 섹션 형태로 나타나도록 코딩
- 그룹화 하지 않는다면, 버튼 형태로 나타난다.
- $vim groups.yaml

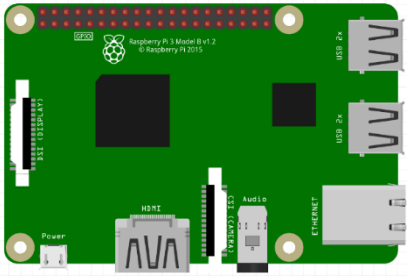
```
my_room:
  name: "DHT22"
  entities:
  - sensor.temperature
  - sensor.humidity

my_room2:
  name: "CDS"
  entities:
  - sensor.light
```

❑ 설정 완료 후, home assistant를 실행시키면, 아래와 같은 화면 실행



RPI3 Server

<Raspberry Pi>



Text (JSON)

API

Speech

Google Cloud Platform

Google Cloud Speech API

Convert
Speech-to-Text

Flask

MariaDB

<Node 서버>

MariaDB: Design Schema

- List all contributes

Time	Temperature	Humidity	CDS_value	PIR_value	Voice*
-	-	-	-	-	-

Table Sensors
(Time, temperature, humidity, CDS_value, PIR_Value, Voice)

* Voice attribute is saved in JSON type

불필요한 데이터와 비정규성을 제거하여 데이터베이스 최적화가 필요하다.

MariaDB: Normalization

- Normalized database
  - Table DTH22 (Time, temperature)
  - Table CDS (Time, CDS_value)
  - Table PIR (Time, PIR_value)
  - Table Voice (Voice)

- Set database constraint to each table
  - Data type
  - Data size
  - Primary Key
  - Null acceptance

❑ Tables and component attributes

Table DHT22

Time	Temperature	Humidity
Primary Key	float not null	float not null

Table CDS

Time	CDS_value
Primary Key	int not null

Table Voice

Voice (JSON)

Table PIR

Time	PIR_value
Primary Key	boolean not null

Node.js DB 설계 코드

❑ Create DB schema directly at MariaDB [1/2]

```
//Connect to MariaDB
sudo mysql –u root –p

//Create database named 'sensors'
MariaDB [(none)] : create database sensors;
MariaDB [(none)] : use sensors

//Select database and create tables with constraints under it
//Create table DHT22
MariaDB [(sensors)] : CREATE TABLE DTH22 (
time TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY,
temperature FLOAT NOT NULL,
humidity FLOAT NOT NULL);

//Create Table CDS
MariaDB [(sensors)] : CREATE TABLE CDS (
time TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY,
cds_value INT NOT NULL);
```

❑ Create DB schema directly at MariaDB [2/2]

```
//Create Table PIR
MariaDB [(sensors)] : CREATE TABLE PIR (
time TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY,
pir_value int NOT NULL);

//Create Table Voice
MariaDB [(sensors)] : CREATE TABLE VOICE (VOICE BLOB);
```

실행 결과 확인

- $show tables;
  Query로 생성된 테이블 확인

```
MariaDB [sensors]> show tables;
+------------------+
| Tables_in_sensors |
+------------------+
| CDS               |
| DHT22             |
| PIR               |
| VOICE             |
+------------------+
4 rows in set (0.00 sec)
```

Flask DB 연동

1. pi@raspberrypi: ~/flaskapp (ssh)

```
from flask import Flask, render_template, request
from flask_mysqldb import MySQL

app = Flask (__name__)

db = yaml.load(open('db.yaml'))
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'Rjsgh0614'
app.config['MYSQL_DB'] = 'flaskapp'

mysql = MySQL(app)

@app.route('/',methods = {'GET','POST'})
def index():
    if request.method == 'POST':
        userDetails = request.form
        name = userDetails['name']
        emails = userDetails['email']
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO users(name,email) VALUES(%s,%s)",(name,emails))

        mysql.connection.commit()
        cur.close
        return 'success'

    return render_template('index.html')

if __name__ == '__main__':
    app.run(host="0.0.0.0",debug=True)
    #app.run(debug=True)
~
~
~
~
~
"app.py" [readonly] 31L, 840C                                    1,1           All
```

```
<h1>FLASK</h1>

<form method = "POST" action"">
        Name <input type = "text" name ="name" />
        <br>
        Email <input type = "email" name = "email" />
        <br>
        <input type = "submit">
</form>
```

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| name  | varchar(20) | YES  |     | NULL    |       |
| email | varchar(40) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```