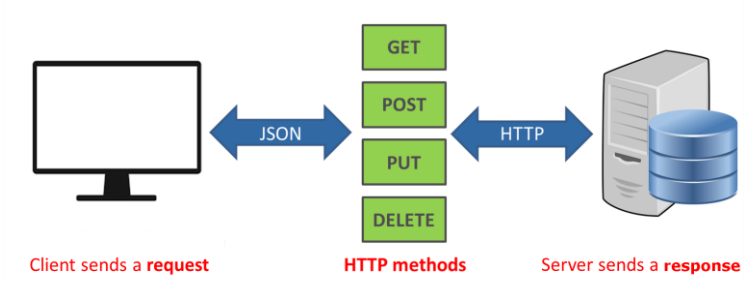
API

Web API



- GET /task/ เป็นการเรียกข้อมูลจากเซิร์ฟเวอร์
- POST /task/ การเพิ่มข้อมูลไปยังเซิร์ฟเวอร์
- DELETE /task/<item id> ลบรายการที่ไม่ต้องการออก
- PUT /task/<item id> แก้ไขข้อมูลเดิม

JavaScript Object Notation (JSON)

```
{
 "IP": "192.168.1.1",
 "Servername": "myName"
}
```

- ใช้กับเว็ปแอปพลิเคชันเพื่อสื่อสารกับเซิร์ฟเวอร์ด้วยการสื่อสารข้อมูลขนาดเล็ก (lightweight)
- รูปแบบการจัดเก็บแบบ JSON จะใช้เครื่องหมายจุลภาค (,) คั่นระหว่างข้อมูล และข้อมูลแต่ละตัวจะเก็บเป็นแบบจับคู่ Name: Value ภายใต้เครื่องหมายปีกกาเปิด { และปีกการปิด }

เปรียบเทียบข้อมูลภาษาไพธอนและเจสัน

ไพธอน (Python)	เจสัน (JSON)
dict	Object
list	Array
tuple	Array
str	String
int	Number
float	Number
True	true
False	false
None	null

- ออปเจ็ค (Object) การจัดเก็บข้อมูลในรูปแบบคู่ลำดับ
  - key: value
  - (คีย์จะต้องเป็นตัวแปรแบบสตริงเสมอ)

แปลงไพธอนเป็น Json

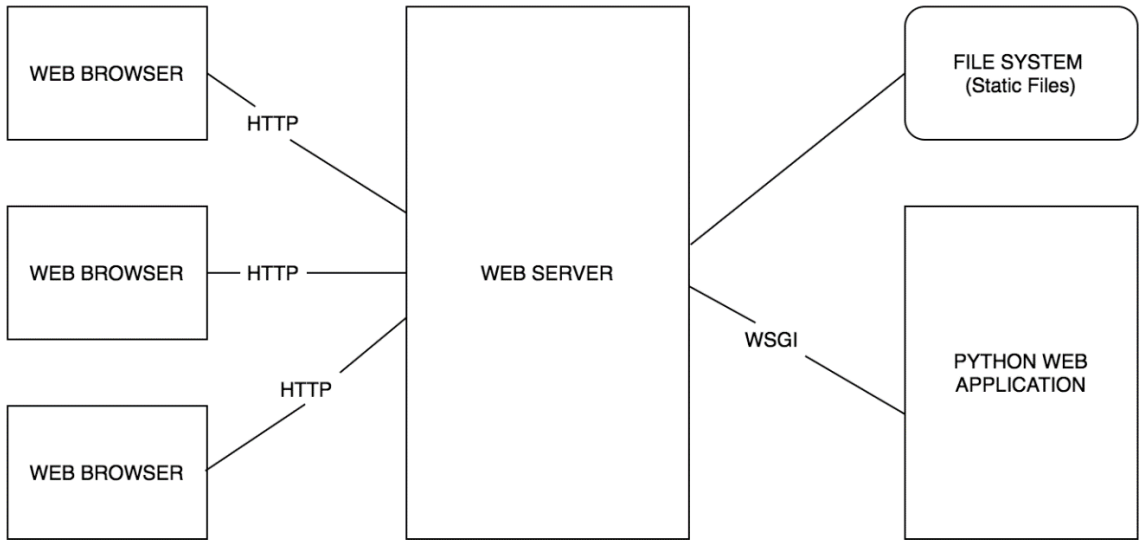
```python
import json

# a Python object (dict):
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
}

# convert into JSON:
y = json.dumps(x)

# the result is a JSON string:
print(y)
```

```python
import utime, machine
import random
import urequests as requests
import ujson
import random
import network

url_bkk = "http://worldtimeapi.org/api/timezone/Asia/Bangkok"
# internal real time clock
rtc = RTC()
# connect wifi
def connect():
    ssid = "ssid"
    password =  "password"
    station = network.WLAN(network.STA_IF)
    if station.isconnected() == True:
      print("Already connected")
      return
    station.active(True)
    station.connect(ssid, password)

# get Bangkok time
def get_bangkok_time():
    response = requests.get(url_bkk)

    if response.status_code == 200:
        parsed = response.json()
        datetime_str = str(parsed["datetime"])
        year = int(datetime_str[0:4])
        month = int(datetime_str[5:7])
        day = int(datetime_str[8:10])
        hour = int(datetime_str[11:13])
        minute = int(datetime_str[14:16])
        second = int(datetime_str[17:19])
        subsecond = int(round(int(datetime_str[20:26]) / 10000))
        rtc.datetime((year, month, day, 0, hour, minute, second, subsecond))

# get time
def get_current_time():
    current_time = "{0:4d}-{1:02d}-{2:02d}T{4:02d}:{5:02d}:{6:02d}".format(*rtc.datetime())
    return current_time

connect()
get_bangkok_time()
while True:
    print(get_current_time())
    utime.sleep(300)
```

ทดสอบการเรียกใช้ API

- ให้ใช้โมดูลที่มีอยู่ ทดสอบการเขียนและอ่าน real-time clock ของไมโครคอนโทรเลอร์
- https://github.com/ckboa/AIoT/blob/main/reading_time.py

สร้าง Web-API ไว้ใช้เองด้วย Flask



- เว็ปแอปพลิเคชันเฟรมเวิร์กภาษาไพธอน พัฒนาขึ้นโดย Armin Ronacher
- การทำงานของแอปพลิเคชัน Flask จะต้องมีการสร้างอินสแทนส์ ภายใต้โปรโตคอล Web Server Gateway Interface (WSGI)

Hello Smart Farm

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def Hello():
    return '<h1>Hello Smart Farm!</h1>'

if __name__ == '__main__':
    app.run()
```

- app.run(host, port, debug, options)
  - **host** กำหนดชื่อของโฮสต์โดยดีฟอลท์จะใช้127.0.0.1 (localhost) เซิร์ฟเวอร์อื่นจะใช้ 0.0.0.0
  - **port** หมายเลยพอร์ต โดยดีฟอลท์จะอยู่ที่ 5000
  - **debug** โดยดีฟอลท์จะมีค่าเป็น False (True เพื่อแสดงค่าดีบัก)
  - **options** จะส่งไปยังเซิร์ฟเวอร์ Werkzeug (underlying)

สร้าง Environment สำหรับ Flask

- ติดตั้ง **virtualenv**

```
PS C:\Users\CK> pip install virtualenv
```

- สร้างโฟลเดอร์สำหรับเวอร์ชวล environment

```
PS C:\Users\CK> mkdir myapi
PS C:\Users\CK> cd myapi
PS C:\Users\CK\myapi> virtualenv venv
```

- เรียกใช้เวอร์ชวล environment และติดตั้ง Flask สังเกตคำว่า **(venv)** ที่เกิดขึ้นหน้าโฟลเดอร์

```
PS C:\Users\CK\myapi> venv/scripts\activate
(venv) PS C:\Users\CK\myapi> pip install Flask
```

```python
from flask import Flask
from flask_mysqldb import MySQL
from flask import jsonify
from flask import flash, request
from datetime import datetime

app = Flask(__name__)

#mySQL Config
app.config['MYSQL_HOST'] = '127.0.0.1'
app.config['MYSQL_USER'] = 'username'
app.config['MYSQL_PASSWORD'] = 'password'
app.config['MYSQL_DB'] = 'yourdatabase'

# init MYSQL
mysql = MySQL(app)

# add dht data
@app.route('/sensor_dht', methods=['POST'])
def sensor_dht():
    try:
        details = request.json
        temperature = details['temp']
        huminity = details['humi']
        current = datetime.now()
        day = current.strftime("%Y-%m-%d")
        time = current.strftime("%H:%M:%S")
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO sensor_dht(temperature, huminity, Date, Time) VALUES (%s, %s, %s, %s)", \
            (temperature, huminity, day, time))
        mysql.connection.commit()
        cur.close()
        resp = jsonify("Data Added")
        resp.status_code = 200
        return resp
    except Exception as e:
        print(e)

if __name__ == '__main__':
    app.run(debug=True)
```

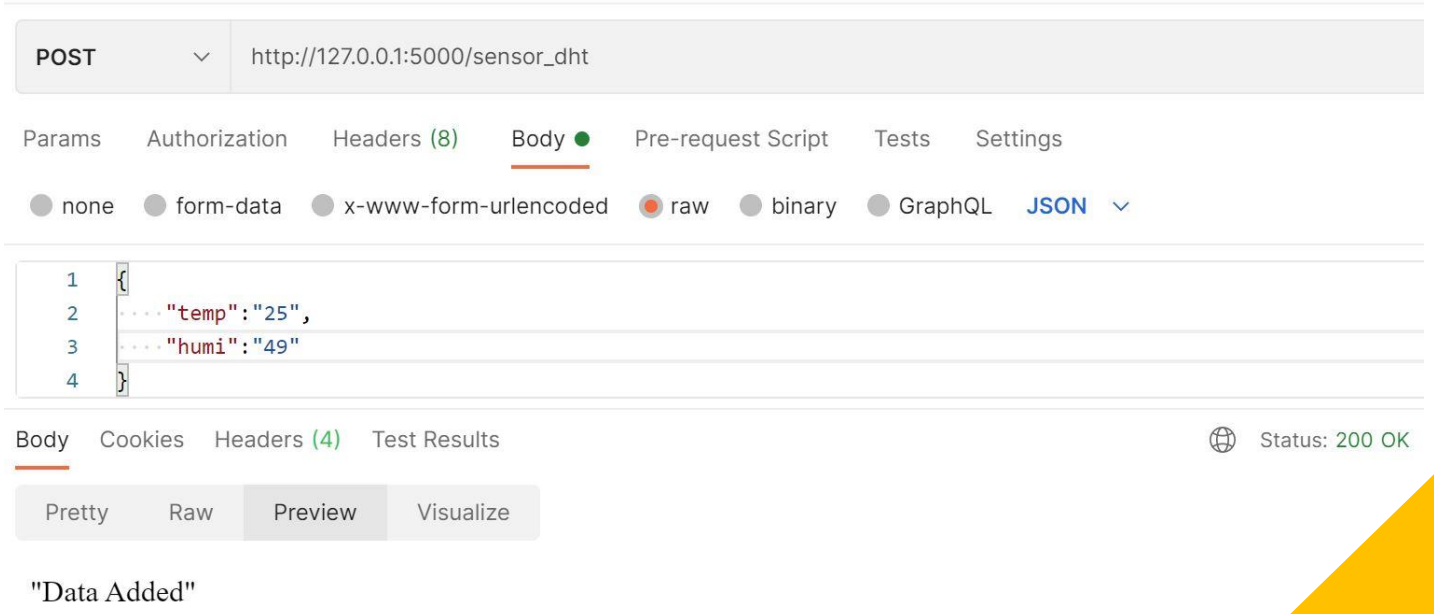
Flask to mySQL

- ทดสอบการเขียนข้อมูลลงฐานข้อมูลผ่าน API
- Table: sensor_dht

ติดตั้งโมดูล เพิ่มเติม

```
(venv) PS C:\Users\CK\myapi> pip install flask_mysqldb
```

ทดสอบด้วย Postman



ส่งค่าจาก Module (เพิ่มส่วน ... เท่าที่จำเป็น)

```
import ...
import urequests as requests
import ujson
import random


def read_sensor():
    ...


def insert_my_data(temp, humi):
    post_data = ujson.dumps({ "temp": temp , "humi": humi})
    url = " ... your url ... "
    res = requests.post(url, headers = {'content-type': 'application/json'},  data = post_data)
    text = res.text
    return text

....
```

- แก้ไข้โค้ดนี้ เพื่อเขียนข้อมูลไปยัง database โดยให้อ่านค่าจาก sensor อุณหภูมิและความชื้น