**IoT 시스템 설계**

설계 과제 #1. IoT 서버 설계



|  |  |
| --- | --- |
| 학부: | 전산전자공학부 |
| 과목코드: | ECE30003-01 |
| 이름: | 정지우(21800677),  최준혁(21900764),  한나린(22100768) |
| 제출 날짜: | 2023년 11월 4일 |

**목차**

1. **서론**
   1. **설계 과제 내용**
   2. **Requirements**
   3. **Specifications**
   4. **설계 과제 목적**
2. **본론**
   1. **Function Flow Chart**
   2. **UI 구성**
   3. **소스코드 및 알고리즘 설명**
   4. **최종 구현 및 데모 결과**
3. **결론**
   1. **작업 중 발생한 문제점**
   2. **팀워크**
   3. **결과 및 고찰**
4. **참고문헌**

**1. 서론**

**1.1. 설계 과제 내용**

RaspberryPi에서 웹 프레임워크인 Flask로 웹 서버를 구축하고, 이를 통해 NodeMCU의 센서에서 얻은 데이터 값을 전송받아 브라우저에서 확인하고, 버튼 클릭을 통해 액추에이터를 제어하도록 한다. 웹 서버와 NodeMCU 간의 통신은 MQTT 프로토콜을 사용한다.

설계 과제 수행 GitHub Repository URL : <https://github.com/DolmaengC/IoT_server>

에서 모든 작업을 확인할 수 있다.

**1.2. Requirements**

* NodeMCU를 사용하고, 센서 및 액추에이터로 LED, DHT22, CDS, Relay/USBLED, 0.96” OLED 디스플레이를 사용한다.
* NodeMCU의 기본 핀 할당은 LED-D0, DHT-D3, USBLED(RELAY)-D4, CDS-A0 로 한다.
* NodeMCU에서 DHTesp for DHT22, Adafruit\_SSD1306 and Adafruit\_gfx for OLED Display, EspMQTTClient for MQTT 라이브러리를 사용한다.
* NodeMCU의 경우, non-blocking 형태로 코드를 작성하며, 불가피한 경우를 제외하고는 delay() 함수를 사용하지 않는다.
* RaspberryPi에서 Mosquitto MQTT broker and client, Flask-Mqtt, Render\_template 패키지를 사용한다.

**1.3. Specifications**

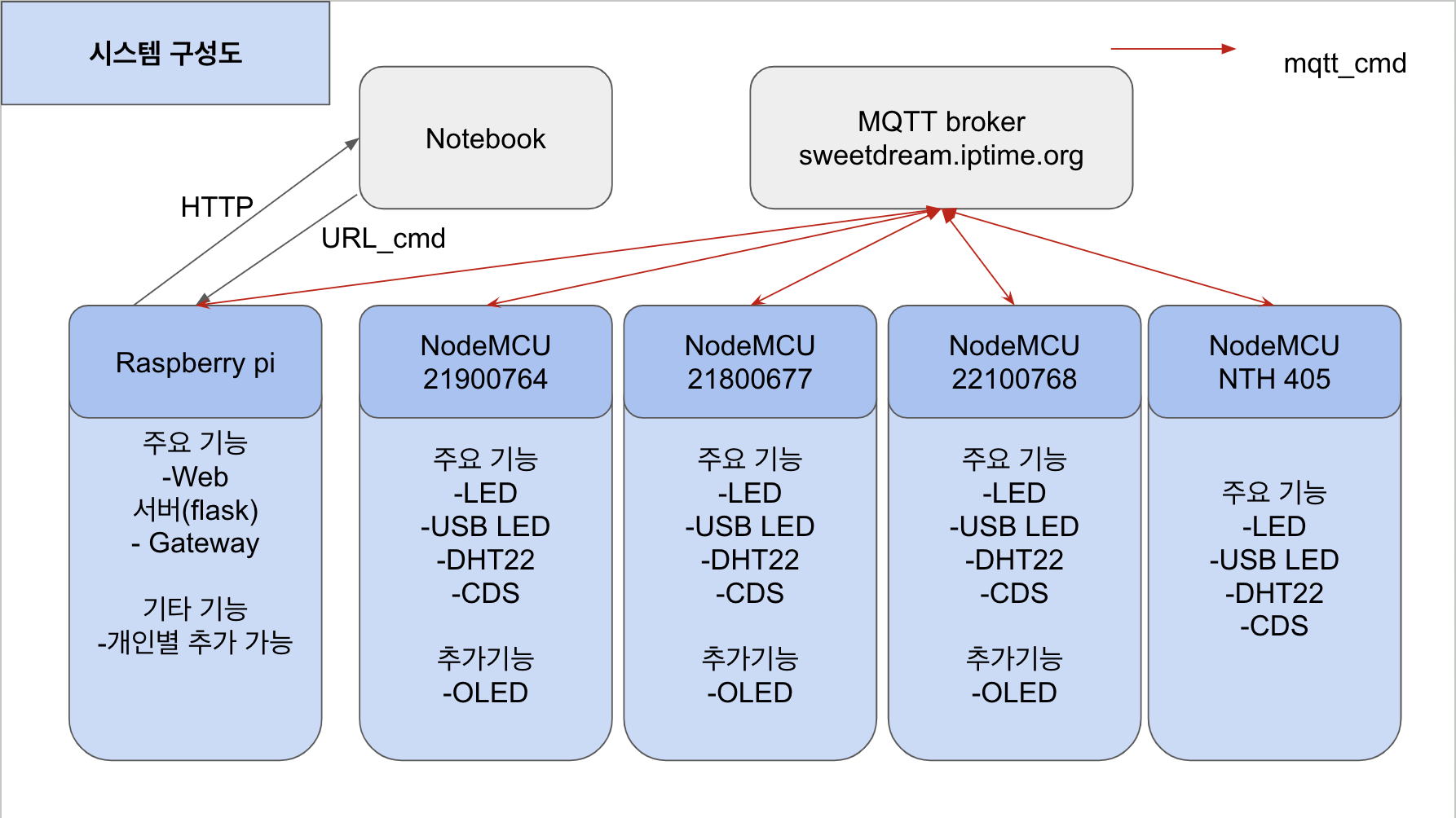
* NodeMCU에서 기본 기능
  + LED/USBLED 수동 제어(On, Off, Toggle)
    - LED/USBLED가 OFF 상태이면, 우선제어 상태가 해제된 상태이다.
  + USBLED 제어
    - 웹 브라우저에서 USBLED ON 버튼을 누를 경우, 조도값에 따른 제어보다 우선한다.
    - 웹 브라우저에서 USBLED OFF 버튼을 누를 경우, 강제로 OFF 되고, 이후 이벤트가 발생하면, 이벤트에 따라 동작한다.
    - 밝은 상태에서 어두운 상태로 바뀌는 이벤트가 발생할 경우, 10초 동안 USBLED가 ON 상태가 된다. 계속 어두운 상태이거나, USBLED가 ON인데 밝은 상태에서 어두운 상태로 바뀌는 이벤트가 발생한다면, USBLED 상태에 영향을 주지 않는다.
    - 이벤트 동작 상태에서는 USBLED TOGGLE 버튼을 눌러도 동작하지 않는다.
  + 온도/습도/조도 데이터
    - NodeMCU에서 온도/습도/조도 데이터는 10초에 한 번씩 측정하고 publish 한다.
    - 측정되는 온도/습도/조도 데이터는 브라우저에서 버튼을 누를 경우 해당 값을 표시해 브라우저에서 확인할 수 있도록 한다.
* 웹 서버에서의 기본 기능
  + LED/USBLED 수동 제어를 위한 버튼으로 On, Off, Toggle이 있다.
  + NodeMCU에서 측정되는 온도/습도/조도 데이터는 브라우저에서 버튼을 누를 경우 해당 값을 브라우저에서 확인할 수 있도록 한다.
* 추가 기능
  + NTH 405호에 설치된 NodeMCU의 데이터와 제어를 추가한다.
  + 모바일로 웹페이지에 접속하는 경우를 고려해, 네 개(팀원 3명과 NTH 405호)의 데이터가 한 개씩 나열될 수 있도록 Easy View를 추가한다.
* URL 커맨드
  + LED/USBLED를 제어하기 위한 URL 커맨드는 다음과 같다.
    - http://{자신의 Rpi IP 주소}/iot/학번/{command}
      * {led, ledon, ledoff} : LED {toggle, on, off}
      * {usbled, usbledon, usbledoff} : USBLED {toggle, on, off}
  + 온도/습도/조도 데이터를 확인하기 위한 URL 커맨드는 다음과 같다.
    - http://{자신의 Rpi IP 주소}/iot/학번/{command}
      * {dht22} : 온도/습도 데이터, JSON 형태
      * {cds} : 조도 데이터
  + 에러 페이지 처리
    - IP 주소 및 포트 번호가 올바르게 입력되었을 경우, URL 커맨드를 따르는 URL은 모두 정상적인 웹페이지로 동작하도록 한다.
    - Page Not Found 와 같은 결과는 출력되지 않도록 한다.
* MQTT Topic
  + NodeMCU에서의 MQTT Topic
    - LED/USBLED 제어를 위해 subscribe 되는 topic 과 message
      * Topic : iot/{학번}
      * Message : {led, ledon, ledoff, usbled, usbledon, usbledoff}
    - DHT22/CDS 센서 측정 값을 publish 하는 topic 과 message
      * Topic : iot/{학번}/{dht22, cds}
      * Message : JSON Format
  + Rpi, 브라우저 등에서의 MQTT Topic
    - LED/USBLED 제어를 위한 topic
      * Topic : iot/{학번}
      * Message : {led, ledon, ledoff, usbled, usbledon, usbledoff}
    - DHT22/CDS 센서 측정 값을 얻기 위한 topic 과 message
      * Topic : iot/{학번}/{dht22, cds}
      * Message : JSON Format

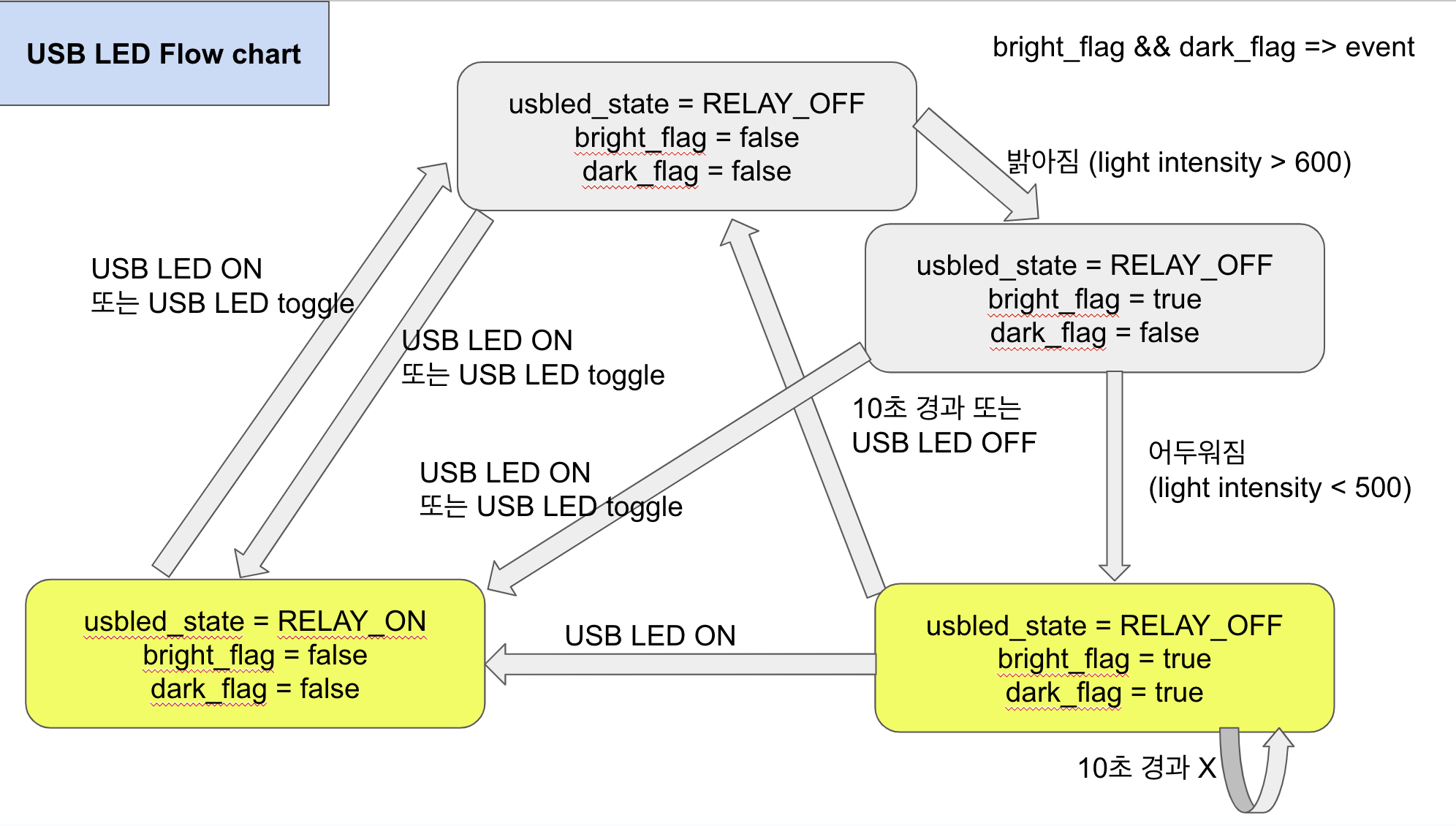
**1.4. 설계 과제 목적**

웹 서버를 설계하고 구축해 웹 브라우저에서 NodeMCU의 센서들이 측정한 데이터를 확인하고, NodeMCU의 액추에이터를 제어하는 과정을 통해 웹 프레임워크, 라이브러리, 패키지, 프로토콜 등을 사용해보고, IoT 서버 설계 및 구축하는 과정과 서버와 디바이스(NodeMCU)의 통신 과정을 이해한다.

**2. 본론**

**2.1. Function Flow Chart**

****



**2.2. UI 구성**

표의 형태로 팀원 3명과 NTH 405호의 데이터가 온도, 습도, 조도 행에 각각 표시되고, 제어 행에 LED TOGGLE, LED ON, LED OFF, USBLED TOGGLE, USBLED ON, USBLED OFF 버튼이 차례대로 나열된다. 표 상단에는 Refresh 버튼이 위치한다.

**2.3. 소스코드 및 알고리즘 설명**

**2.3.1 Arduino Source Code**

|  |
| --- |
| void loop() {  mqtt\_client.loop();  unsigned long currentMillis = millis();  // 안정적으로 Nan이 아닌 온습도 값을 얻기 위해 10초가 아닌 1초마다 온습도 측정  // 안정적인 auto light 기능을 위해 매초마다 조도 측정  if (currentMillis - previousMillis\_sensor >= interval\_OneSecond) {  previousMillis\_sensor = currentMillis;  lightValue = analogRead(CDS\_PIN);  Serial.print("lightValue: ");  Serial.println(lightValue);  // read temperature and humidity  float tmp\_temperature = dht.getTemperature();  float tmp\_humidity = dht.getHumidity();  // print temperature and humidity to serial monitor  if ( !isnan(tmp\_humidity) && !isnan(tmp\_temperature) ) {  Serial.print("Temperature = ");  Serial.print(temperature);  Serial.print(" Humidity = ");  Serial.println(humidity);  humidity = tmp\_humidity;  temperature = tmp\_temperature;  OLED\_Print\_text(humidity, temperature);  }  // USBLED turn on 상태일 때는 밝아진 상태에서 어두워져도 USBLED 상태에 영향을 주지 않음  if (usbled\_state == RELAY\_ON) {  // USBLED turn on 상태이면 밝아짐 확인 플래그와 어두워짐 확인 플래그 모두 초기화  dark\_flag = false;  bright\_flag = false;  }  else {  // 밝은 상태에서 어두워지는 이벤트 발생 여부 확인  if (dark\_flag && bright\_flag) {  // 10초 경과시 USB LED 불을 끄고 event를 위한 flag 초기화  if (currentMillis - light\_Timer >= 10000) {  digitalWrite(USBLED\_PIN, RELAY\_OFF);  dark\_flag = false;  bright\_flag = false;  }  }  else {  if (lightValue < 500 && bright\_flag) { // 밝은 상태에서 어두워지는 이벤트 발생  digitalWrite(USBLED\_PIN, RELAY\_ON);  light\_Timer = currentMillis;  dark\_flag = true;  } else if (lightValue > 600) { // 밝아지는 이벤트 발생  bright\_flag = true;  }  }  }  }  // Publish every 10 second  if (currentMillis - previousMillis\_publish >= interval\_TenSecond) {  previousMillis\_publish = currentMillis;    // 온습도 데이터를 json 형식으로 가공하여 publish 진행  sprintf(pub\_data, "{\"temperature\" : %3.2f, \"humidity\" : %3.2f}", temperature, humidity);  Serial.println(pub\_data);  mqtt\_client.publish("iot/21900764/dht22", pub\_data);  sprintf(pub\_data, "%3.2f", temperature);  mqtt\_client.publish("iot/21900764/dht22\_t", pub\_data);  sprintf(pub\_data, "%3.2f", humidity);  mqtt\_client.publish("iot/21900764/dht22\_h", pub\_data);  sprintf(pub\_data, "%d", lightValue);  mqtt\_client.publish("iot/21900764/cds", pub\_data);  }  } |

**2.3.2 Python Source Code**

1. 라이브러리 추가 및 변수 선언

|  |
| --- |
| from flask import Flask, render\_template  from flask\_mqtt import Mqtt  import time # library for time delay  import json  app = Flask(\_\_name\_\_)  app.config['MQTT\_BROKER\_URL'] = 'sweetdream.iptime.org'  app.config['MQTT\_BROKER\_PORT'] = 1883  app.config['MQTT\_USERNAME'] = 'iot'  app.config['MQTT\_PASSWORD'] = 'csee1414' |

1. MQTT topic 설정

|  |
| --- |
| mqtt = Mqtt(app)  # 21900764  pub\_topic\_21900764 = 'iot/21900764' # iot/{student id}  sub\_topic\_dht22\_21900764 = 'iot/21900764/dht22'  sub\_topic\_dht22\_h\_21900764 = 'iot/21900764/dht22\_h'  sub\_topic\_dht22\_t\_21900764 = 'iot/21900764/dht22\_t'  sub\_topic\_cds\_21900764 = 'iot/21900764/cds'  # 22100768  pub\_topic\_22100768 = 'iot/22100768' # iot/{student id}  sub\_topic\_dht22\_22100768 = 'iot/22100768/dht22'  sub\_topic\_dht22\_h\_22100768 = 'iot/22100768/dht22\_h'  sub\_topic\_dht22\_t\_22100768 = 'iot/22100768/dht22\_t'  sub\_topic\_cds\_22100768 = 'iot/22100768/cds'  #21800677  pub\_topic\_21800677 = 'iot/21800677' # iot/{student id}  sub\_topic\_dht22\_21800677 = 'iot/21800677/dht22'  sub\_topic\_dht22\_h\_21800677 = 'iot/21800677/dht22\_h'  sub\_topic\_dht22\_t\_21800677 = 'iot/21800677/dht22\_t'  sub\_topic\_cds\_21800677 = 'iot/21800677/cds'  # nth405  pub\_topic\_nth405 = 'iot/nth405' # iot/{student id}  sub\_topic\_dht22\_nth405 = 'iot/nth405/dht22'  sub\_topic\_dht22\_h\_nth405 = 'iot/nth405/dht22\_h'  sub\_topic\_dht22\_t\_nth405 = 'iot/nth405/dht22\_t'  sub\_topic\_cds\_nth405 = 'iot/nth405/cds' |

1. MQTT payload 데이터 저장을 위한 글로벌 변수 선언 (json form)

|  |
| --- |
| # global variable for message payload  mqtt\_message = ''  json\_data = {  "temperature\_21900764" : '',  "humidity\_21900764" : '',  "light\_intensity\_21900764" : '',  "temperature\_22100768" : '',  "humidity\_22100768" : '',  "light\_intensity\_22100768" : '',  "temperature\_21800677" : '',  "humidity\_21800677" : '',  "light\_intensity\_21800677" : '' ,  "temperature\_nth405" : '',  "humidity\_nth405" : '',  "light\_intensity\_nth405" : ''  }  print('@@ Use URL:/iot/21900764/{led,ledon,ledoff,dht22,cds}') |

1. 웹상에서 LED 제어

|  |
| --- |
| view\_mode = 0  # cmd = {led, ledon, ledoff, usbled, usbledon, usbledoff}  @app.route('/iot/21900764/<cmd>')  def get\_command\_21900764(cmd):  global mqtt\_message  global view\_mode  if cmd == 'led':  mqtt.publish(pub\_topic\_21900764, 'led')  elif cmd == 'ledon':  mqtt.publish(pub\_topic\_21900764, 'ledon')  elif cmd == 'ledoff':  mqtt.publish(pub\_topic\_21900764, 'ledoff')  elif cmd == 'usbled':  mqtt.publish(pub\_topic\_21900764, 'usbled')  elif cmd == 'usbledon':  mqtt.publish(pub\_topic\_21900764, 'usbledon')  elif cmd == 'usbledoff':  mqtt.publish(pub\_topic\_21900764, 'usbledoff')  elif cmd == 'change\_mode':  if view\_mode == 1:  view\_mode = 0  else:  view\_mode = 1  if view\_mode == 0:  return render\_template('index.html', json\_data=json\_data)  else :  return render\_template('index\_easy\_view.html', json\_data=json\_data)  # cmd = {led, ledon, ledoff, usbled, usbledon, usbledoff}  @app.route('/iot/22100768/<cmd>')  def get\_command\_22100768(cmd):  global mqtt\_message  if cmd == 'led':  mqtt.publish(pub\_topic\_22100768, 'led')  elif cmd == 'ledon':  mqtt.publish(pub\_topic\_22100768, 'ledon')  elif cmd == 'ledoff':  mqtt.publish(pub\_topic\_22100768, 'ledoff')  elif cmd == 'usbled':  mqtt.publish(pub\_topic\_22100768, 'usbled')  elif cmd == 'usbledon':  mqtt.publish(pub\_topic\_22100768, 'usbledon')  elif cmd == 'usbledoff':  mqtt.publish(pub\_topic\_22100768, 'usbledoff')  if view\_mode == 0:  return render\_template('index.html', json\_data=json\_data)  else :  return render\_template('index\_easy\_view.html', json\_data=json\_data)  # cmd = {led, ledon, ledoff, usbled, usbledon, usbledoff}  @app.route('/iot/21800677/<cmd>')  def get\_command\_21800677(cmd):  global mqtt\_message  if cmd == 'led':  mqtt.publish(pub\_topic\_21800677, 'led')  elif cmd == 'ledon':  mqtt.publish(pub\_topic\_21800677, 'ledon')  elif cmd == 'ledoff':  mqtt.publish(pub\_topic\_21800677, 'ledoff')    elif cmd == 'usbled':  mqtt.publish(pub\_topic\_21800677, 'usbled')  elif cmd == 'usbledon':  mqtt.publish(pub\_topic\_21800677, 'usbledon')  elif cmd == 'usbledoff':  mqtt.publish(pub\_topic\_21800677, 'usbledoff')  if view\_mode == 0:  return render\_template('index.html', json\_data=json\_data)  else :  return render\_template('index\_easy\_view.html', json\_data=json\_data)  # cmd = {led, ledon, ledoff, usbled, usbledon, usbledoff}  @app.route('/iot/nth405/<cmd>')  def get\_command\_nth405(cmd):  global mqtt\_message    if cmd == 'led':  mqtt.publish(pub\_topic\_nth405, 'led')  elif cmd == 'ledon':  mqtt.publish(pub\_topic\_nth405, 'ledon')  elif cmd == 'ledoff':  mqtt.publish(pub\_topic\_nth405, 'ledoff')  elif cmd == 'usbled':  mqtt.publish(pub\_topic\_nth405, 'usbled')  elif cmd == 'usbledon':  mqtt.publish(pub\_topic\_nth405, 'usbledon')  elif cmd == 'usbledoff':  mqtt.publish(pub\_topic\_nth405, 'usbledoff')  if view\_mode == 0:  return render\_template('index.html', json\_data=json\_data)  else :  return render\_template('index\_easy\_view.html', json\_data=json\_data) |

1. 404 에러(Page Not Found) 시에도 에러 페이지를 보내지 않고 기본 페이지를 보여주도록 함

|  |
| --- |
| @app.errorhandler(404)  def not\_found\_error(error):  if view\_mode == 0:  return render\_template('index.html', json\_data=json\_data)  else :  return render\_template('index\_easy\_view.html', json\_data=json\_data) |

1. 모든 topic에 대해 MQTT subscribe

|  |
| --- |
| # When mqtt is connected, subscribe to following topics  @mqtt.on\_connect()  def handle\_connect(client, userdata, flags, rc):  mqtt.subscribe(sub\_topic\_dht22\_21900764)  mqtt.subscribe(sub\_topic\_dht22\_t\_21900764)  mqtt.subscribe(sub\_topic\_dht22\_h\_21900764)  mqtt.subscribe(sub\_topic\_cds\_21900764)  mqtt.subscribe(pub\_topic\_22100768)  mqtt.subscribe(sub\_topic\_dht22\_22100768)  mqtt.subscribe(sub\_topic\_dht22\_t\_22100768)  mqtt.subscribe(sub\_topic\_dht22\_h\_22100768)  mqtt.subscribe(sub\_topic\_cds\_22100768)    mqtt.subscribe(pub\_topic\_21800677)  mqtt.subscribe(sub\_topic\_dht22\_21800677)  mqtt.subscribe(sub\_topic\_dht22\_t\_21800677)  mqtt.subscribe(sub\_topic\_dht22\_h\_21800677)  mqtt.subscribe(sub\_topic\_cds\_21800677)  mqtt.subscribe(pub\_topic\_nth405)  mqtt.subscribe(sub\_topic\_dht22\_nth405)  mqtt.subscribe(sub\_topic\_dht22\_t\_nth405)  mqtt.subscribe(sub\_topic\_dht22\_h\_nth405)  mqtt.subscribe(sub\_topic\_cds\_nth405) |

1. MQTT를 통해 메세지를 받을 경우, 토픽에 따라 payload 데이터 처리

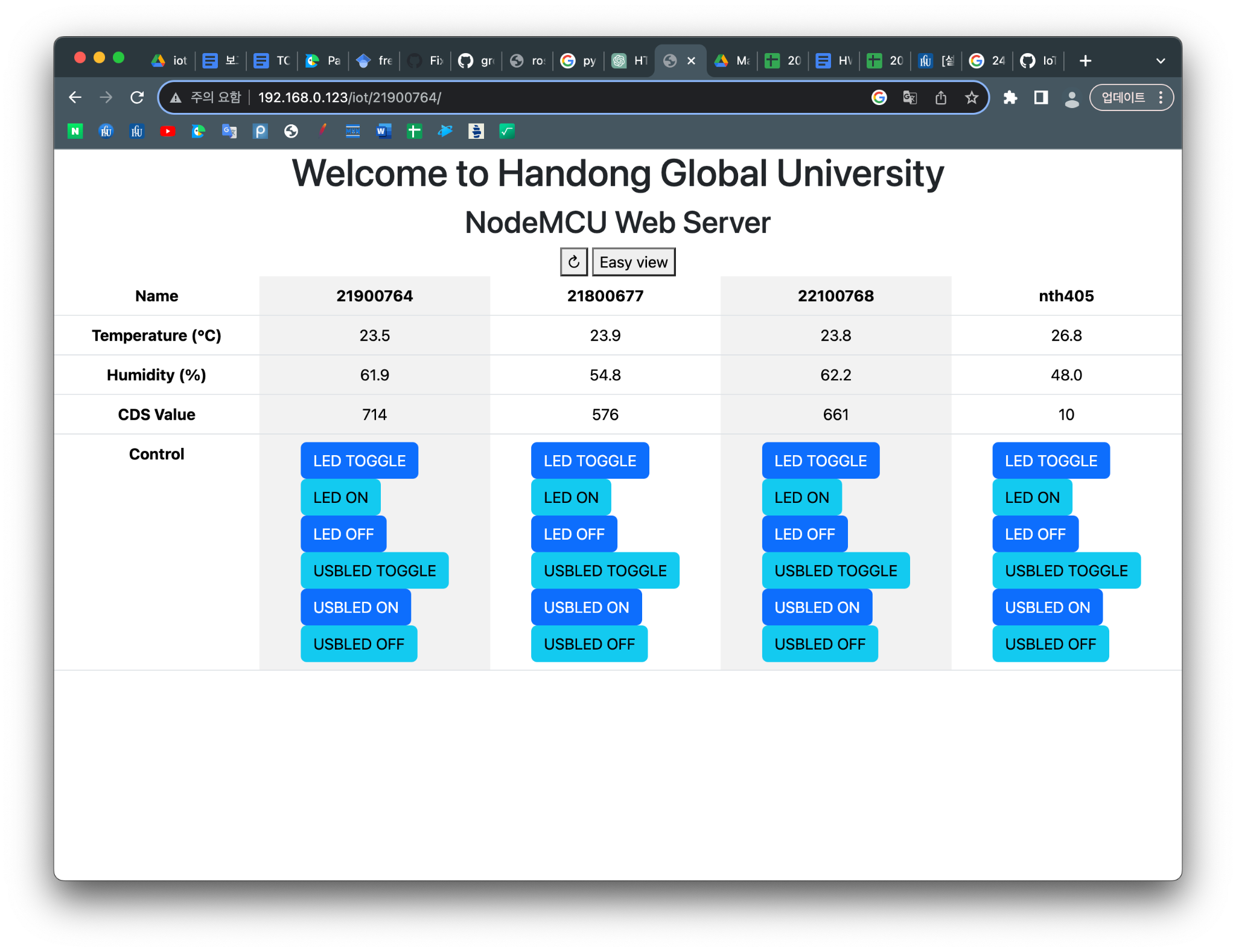
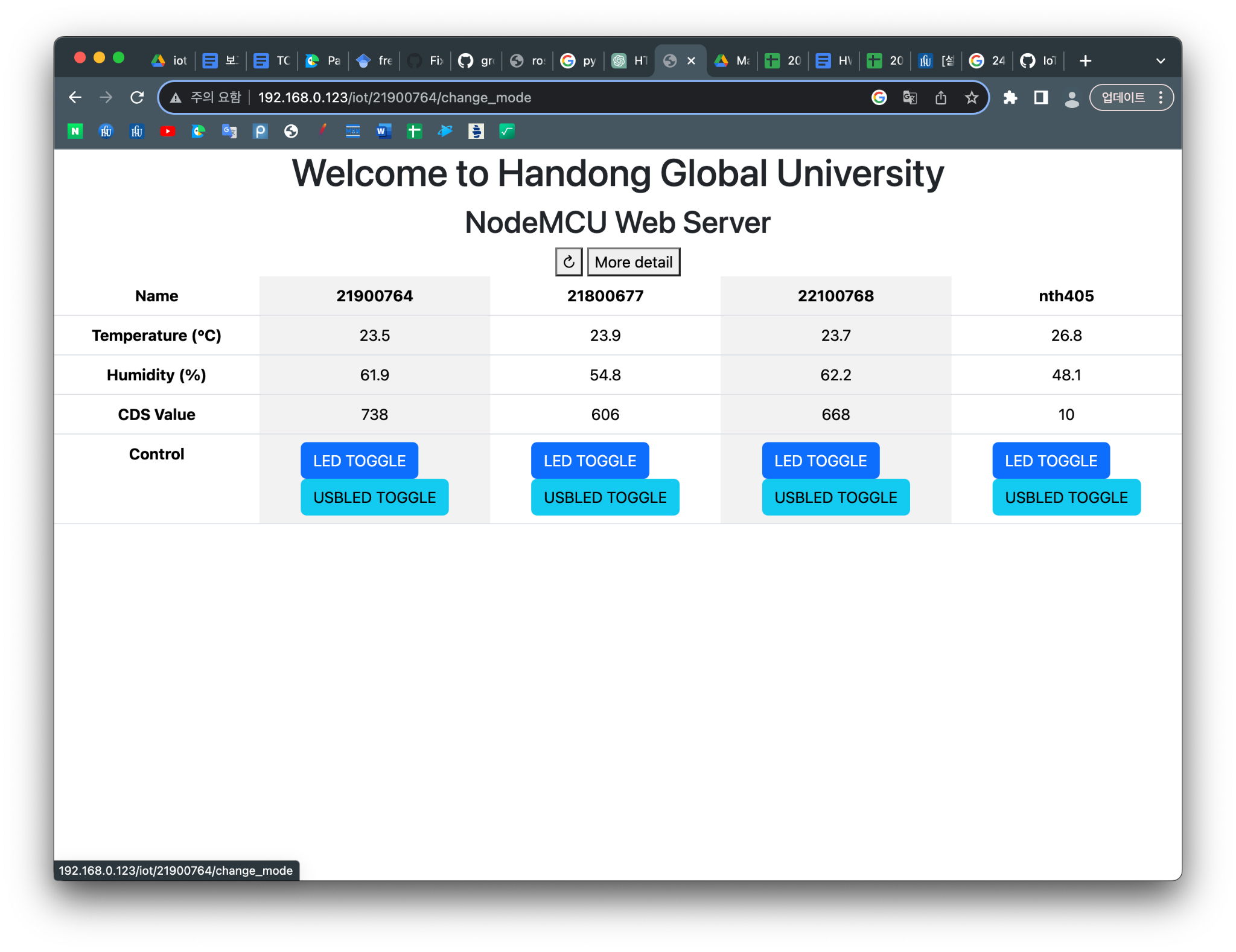
|  |
| --- |
| # When mqtt receives message from subscribed topic  @mqtt.on\_message()  def handle\_mqtt\_message(client, userdata, message):  global mqtt\_message    global json\_data  topic = message.topic  payload = message.payload  payload = payload.decode('utf-8')  topic = str(message.topic)  payload = str(message.payload)[2:-1]  if topic == sub\_topic\_dht22\_21900764:  payload = json.loads(payload)  json\_data['humidity\_21900764'] = payload["humidity"]  json\_data['temperature\_21900764'] = payload["temperature"]  elif topic == sub\_topic\_cds\_21900764:  json\_data['light\_intensity\_21900764'] = payload  elif topic == sub\_topic\_dht22\_22100768:  payload = json.loads(payload)  json\_data['humidity\_22100768'] = payload["humidity"]  json\_data['temperature\_22100768'] = payload["temperature"]  elif topic == sub\_topic\_cds\_22100768:  json\_data['light\_intensity\_22100768'] = payload  elif topic == sub\_topic\_dht22\_21800677:  payload = json.loads(payload)  json\_data['humidity\_21800677'] = payload["humidity"]  json\_data['temperature\_21800677'] = payload["temperature"]  elif topic == sub\_topic\_cds\_21800677:  json\_data['light\_intensity\_21800677'] = payload  elif topic == sub\_topic\_dht22\_t\_nth405:  json\_data['temperature\_nth405'] = payload  elif topic == sub\_topic\_dht22\_h\_nth405:  json\_data['humidity\_nth405'] = payload  elif topic == sub\_topic\_cds\_nth405:  json\_data['light\_intensity\_nth405'] = payload    mqtt\_message = payload  print("Topic: " + topic , end="")  print(" message: ", end="")  print(mqtt\_message) |

1. 80번 포트를 통해 출력

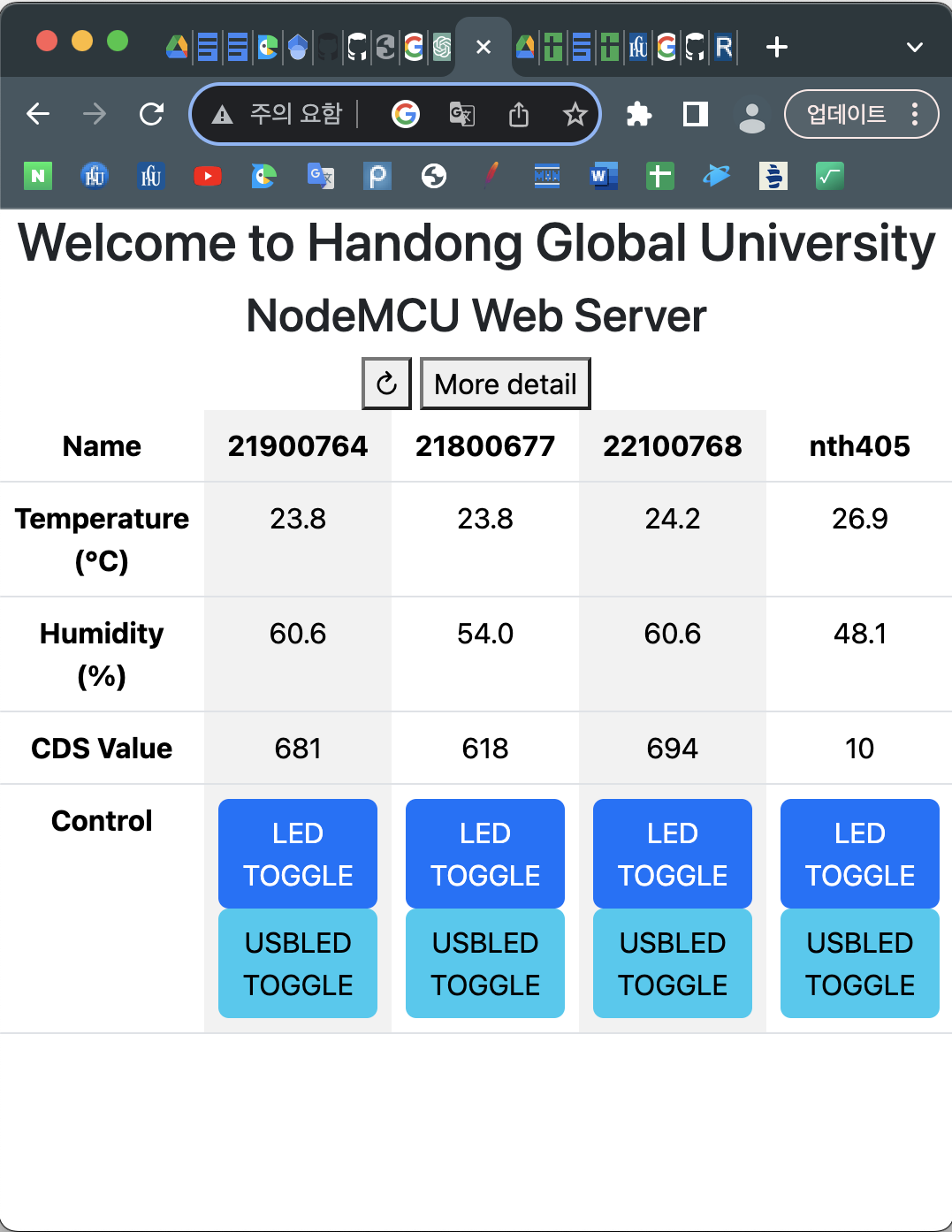
|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  app.run(host='0.0.0.0', port=80, debug=False) |

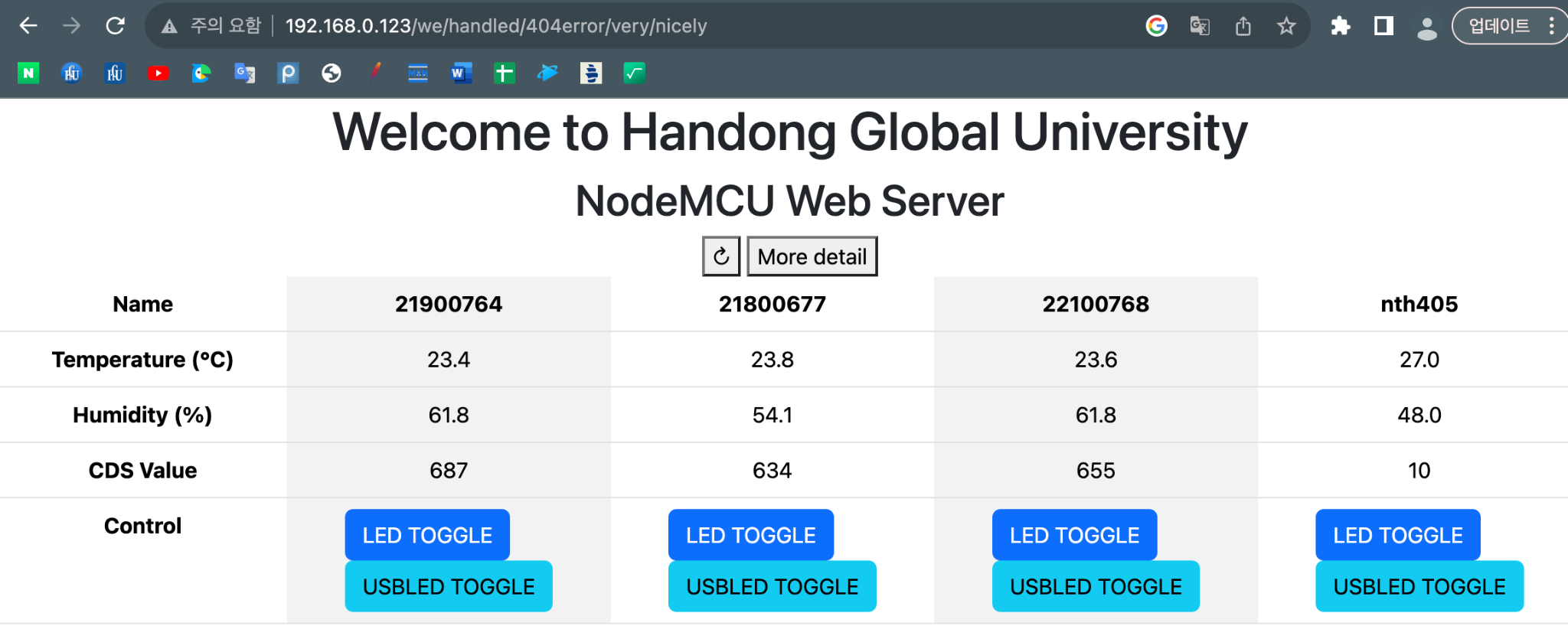
**2.4. 최종 구현 및 데모 결과**

**2.4.1 웹 페이지**

****

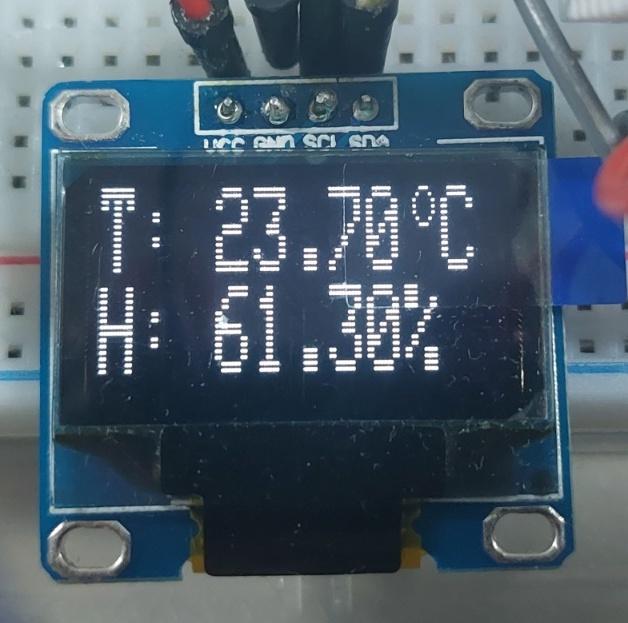
**[Main view] [Easy view]**

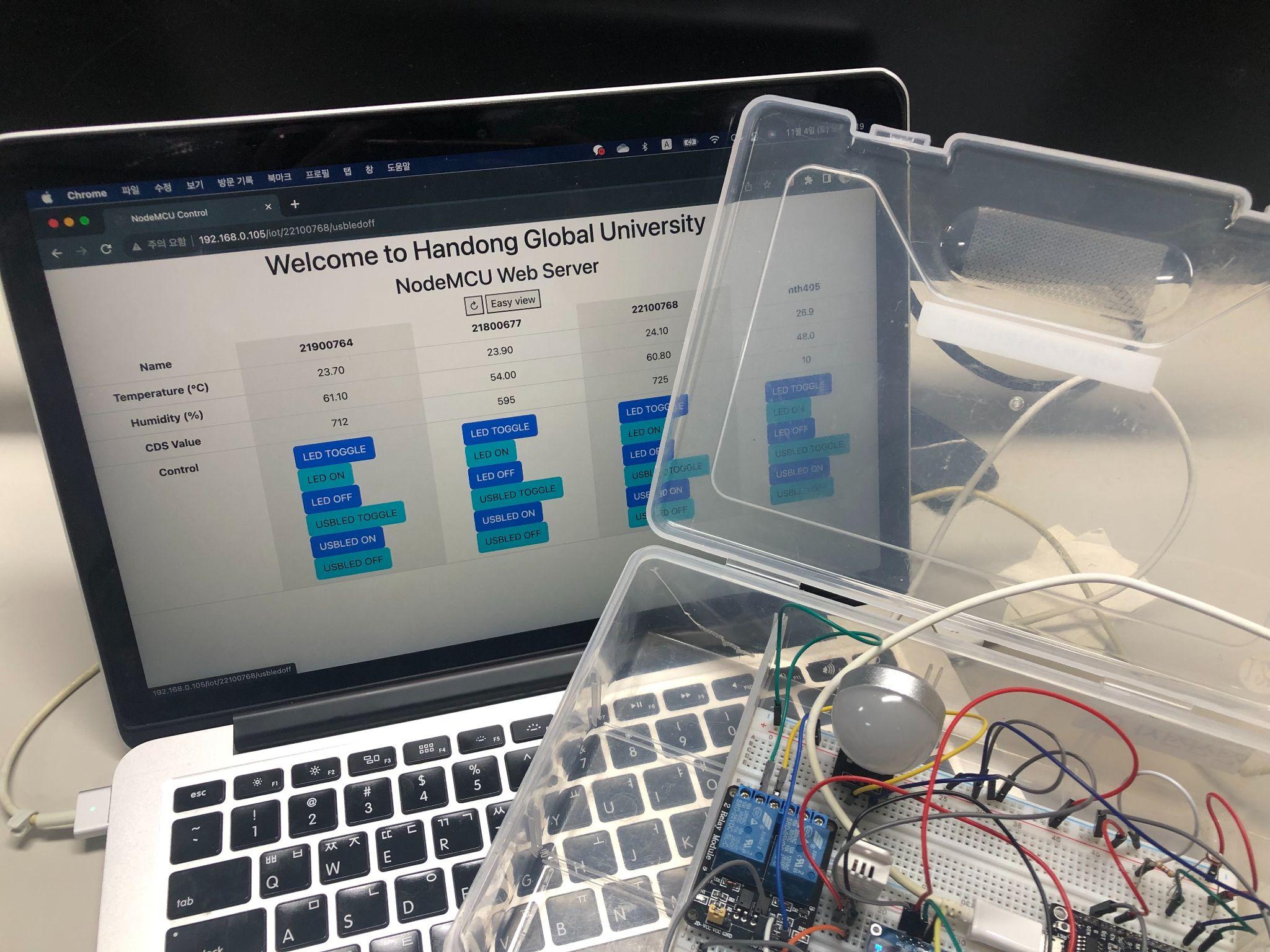
**[반응형 HTML]**

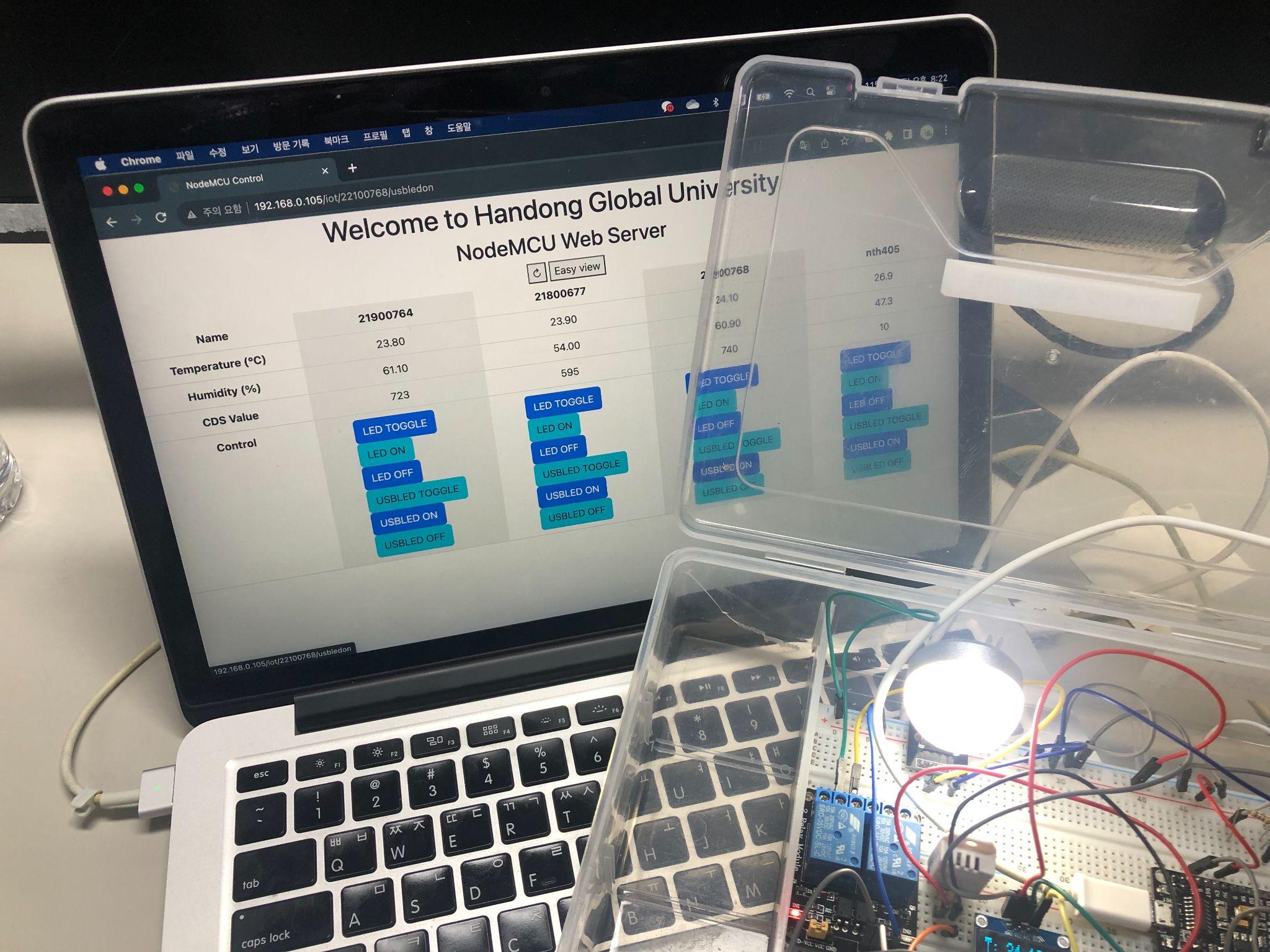
****

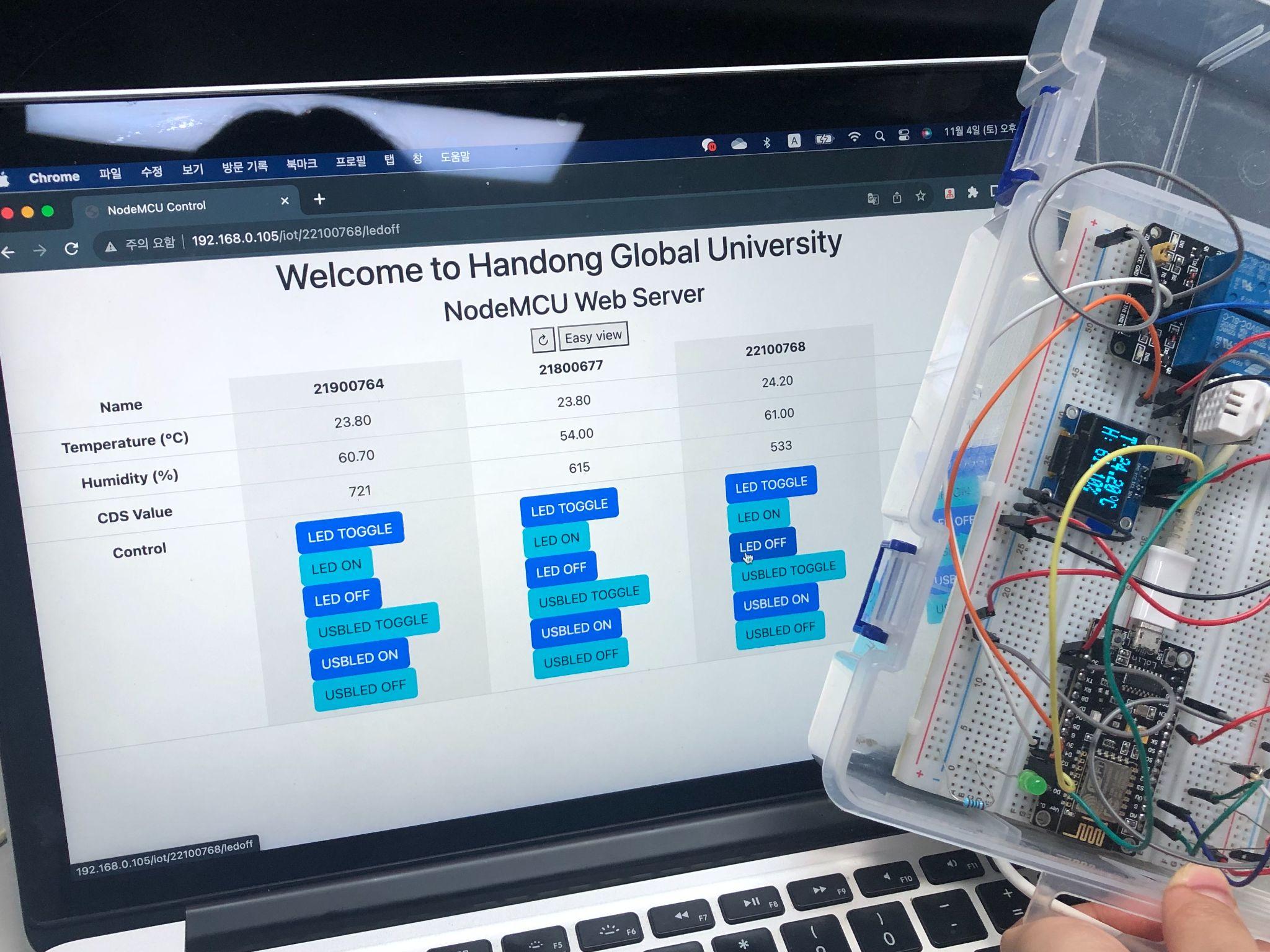
**[404 Handling]**

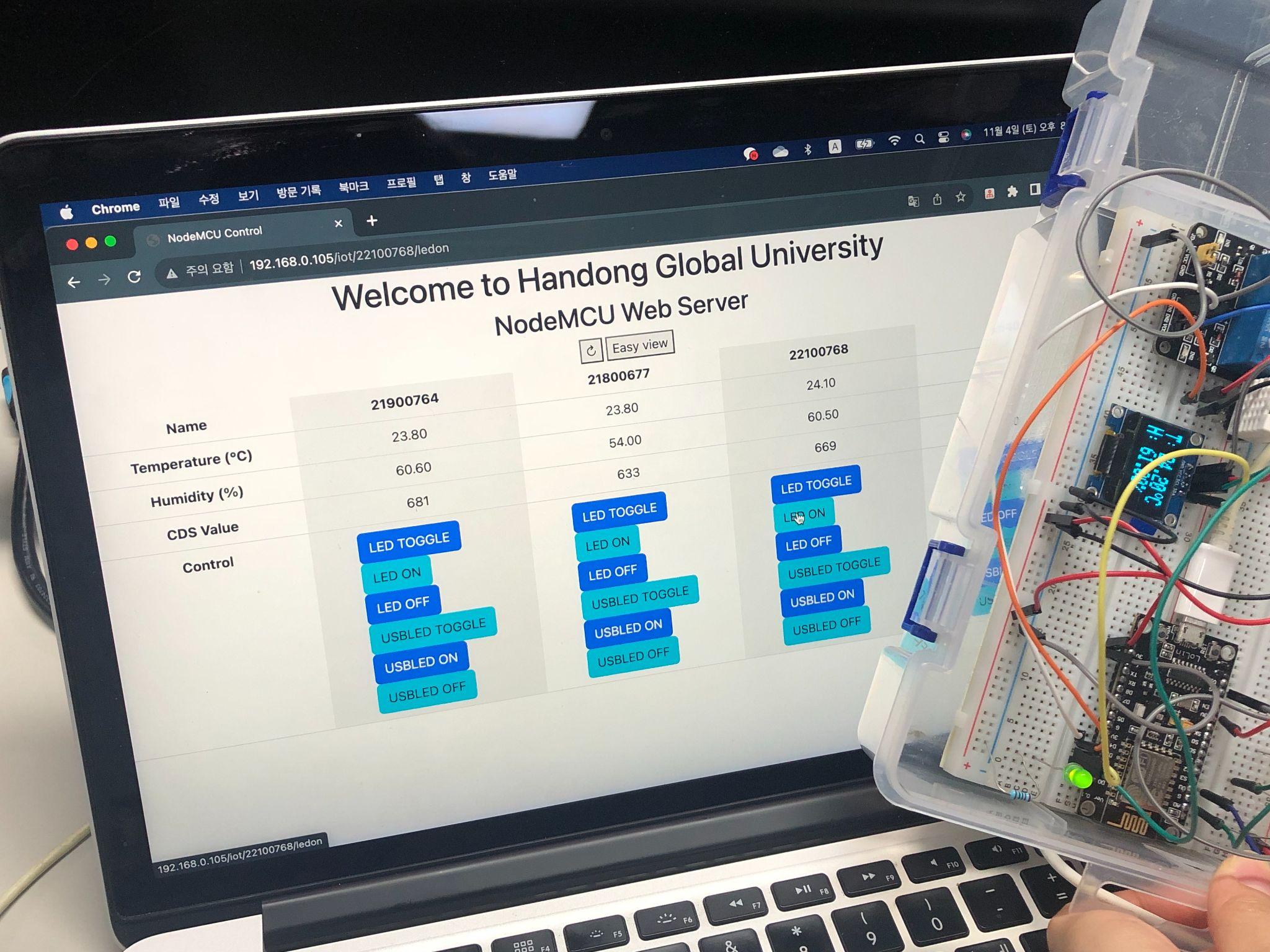
**2.4.2 OLED**

****

**2.4.3. USB LED ON/OFF**

****

**2.4.4. LED ON/OFF**

****

**3. 결론**

**3.1. 작업 중 발생한 문제점**

* NodeMCU에서 DHT22로 얻은 온습도를 json form으로 publish 하기 위해 참고 링크([Decoding and Encoding JSON Arduino | Random Nerd Tutorials](https://randomnerdtutorials.com/decoding-and-encoding-json-with-arduino-or-esp8266/))를 읽어보았지만, json 객체 생성의 필요성이 느껴지지 않았다. 따라서 json 객체를 만들어서 json string 형식으로 전환하는 과정을 생략하고 메뉴얼하게 json string 형식을 만들도록 설계했다.

|  |
| --- |
| sprintf(pub\_data, "{\"temperature\" : %3.2f, \"humidity\" : %3.2f}", temperature, humidity); |

* 온도/습도/조도 데이터를 별도로 저장해 브라우저에서 그래프의 형태로 확인하는 추가 기능을 구현하려고 했으나 실패했다.
  + 원인 : 소스 코드에서 사용한 json\_data의 형식이 배열이 아니라서 데이터를 누적시킬 수 없음
  + 시도

1. json\_data에서 저장하는 형식을 배열로 변경하여 데이터를 받아올 때마다 이를 배열에 저장하는 방식으로 변경하려 했으나, 기본 요구사항인 온도/습도/조도 데이터를 화면에 출력하는 것이 작동하지 않음. json\_data에 데이터가 제대로 저장되지 않는지 확인하기 위해 printf("Received MQTT message: topic={topic}, payload={payload}") 코드를 추가하여 콘솔상에 디버깅을 해보았으나, 문제 없이 저장되는 것으로 보아 json\_data가 HTML 상으로 전송되는 과정에서 문제가 발생하는 것으로 확인되었고, 이를 해결할 방법을 찾지 못해 해당 방식을 폐기함.
2. data.html이라는 파일을 새로 만들고, Flask 파이썬 파일에 @app.route('/data') 경로를 추가해 해당 페이지를 열 때 데이터를 전송함과 동시에 그래프를 생성하려 시도함. 전송 시에 Flask 내부의 jsonify를 사용하여 데이터를 전송하려 시도하였음. - jsonify를 사용 시 render-template를 사용할 수 없기 때문에 화면에 전송받은 json 데이터 내용만 출력됨.
3. data.html과 /data 경로는 데이터를 전송하여 저장하는 것으로 하고, graph.html을 새로 생성해 /graph 경로에서 그래프를 출력하고, HTML 내부에서 fetch('/data')를 사용하여 데이터를 주기적으로 받아오려 했으나, /data 경로를 접속하는 것이 프로젝트 진행에 매끄럽지 못하다고 여겨 이를 폐기함.
4. data.json을 개발 디렉토리 내부에 생성하여 이 json 파일에 데이터를 저장하는 방식을 시도해봄. Flask 안의 send\_from\_directory를 이용함.

|  |
| --- |
| @app.route('/data.json')  def serve\_data():  return send\_from\_directory(directory='.', filename='data.json') |

위의 코드를 적용하여 data.json 파일에 데이터를 저장하고, graph.html 내부에서 fetch('/data.json) 코드를 추가해 데이터를 받아오려 시도했으나, 해당 방식 실행 시 json 파일을 실행시마다 초기화 혹은 삭제시켜야 하는데, 이를 코드 내부에서 실행하기에 어렵다고 판단해 폐기함.

* + 결론 : 이 외에도 다양한 방식으로 시도해 보았지만, 이를 해결할 방식을 결국 찾지 못해 포기함.

**3.2. 팀워크**

* 일의 분배
  + 정지우 : 33%; 코드 작성, 추가 기능 구현, 깃허브 관리
  + 최준혁 : 33%; 코드 작성, 보고서 작성, 추가 기능 구현
  + 한나린 : 33%; 코드 작성, 보고서 작성
* 코드 및 문서 작성
  + 파이썬 코드 : 최준혁
  + 아두이노 코드 : 최준혁
  + HTML : 한나린
  + 보고서 : 정지우, 최준혁, 한나린
  + HowToRun.md : 정지우
  + README.md : 정지우

**3.3. 결과 및 고찰**

RaspberryPi에서 웹 프레임워크인 Flask를 통해 웹 서버를 구축하고, RaspberryPi IP 주소와 지정한 포트 번호를 통해 웹 브라우저에 접속해 팀원 3명과 NTH 405호의 NodeMCU의 센서로 측정한 데이터를 확인하고, 버튼 클릭을 통해 LED와 USBLED를 수동 제어할 수 있었다. 또한, 조도 센서에서 측정한 값을 통해 밝은 상태에서 어두운 상태로 조도값의 변화가 일어날 경우, USBLED가 켜져 자동 제어되도록 할 수 있었다. 웹 서버와 NodeMCU 간의 통신은 MQTT 프로토콜을 사용해 Publish 와 Subscribe 의 개념을 이해하고 Topic 과 Message 로 값을 주고 받는 과정을 실습할 수 있었다.

이번 설계 과제를 수행하며, IoT 서버 설계와 구축, 디바이스(NodeMCU)에 설치된 센서 사용 및 측정 데이터 전송, 웹 서버에서의 디바이스 제어 등의 과정을 실습할 수 있었다. 이를 통해 웹 프레임워크인 Flask 및 각종 파이썬 패키지, DHT22, CDS, OLED 디스플레이 등을 사용하기 위한 아두이노 라이브러리, MQTT 프로토콜 등을 모두 사용해 볼 수 있었으며, 전체적인 IoT 시스템의 설계 및 구축을 이해할 수 있었다.

**4. 참고문헌**

1. Random Nerd Tutorial. n.d. Decoding and Encoding JSON with Arduino or ESP8266. Random Nerd Tutorial.  
   <https://randomnerdtutorials.com/decoding-and-encoding-json-with-arduino-or-esp8266/>
2. Datadog. n.d. JSON 출력에 json.dumps 대신 jsonify 사용하기. Datadog. <https://docs.datadoghq.com/ko/continuous_integration/static_analysis/rules/python-flask/use-jsonify/>
3. tedboy. n.d. flask.send\_from\_directory. tedboy.github.io.   
   https://tedboy.github.io/flask/generated/flask.send\_from\_directory.html
4. MDN. n.d Fetch API 사용하기. MDN.   
   https://developer.mozilla.org/ko/docs/Web/API/Fetch\_API/Using\_Fetch
5. ChatGPT3.5