[설계과제 1] IoT 서버 설계

School Of Computer Science And Electrical Engineering 21900050 권은혁

School Of Computer Science And Electrical Engineering 21900780 하정원

목차

1. 서론

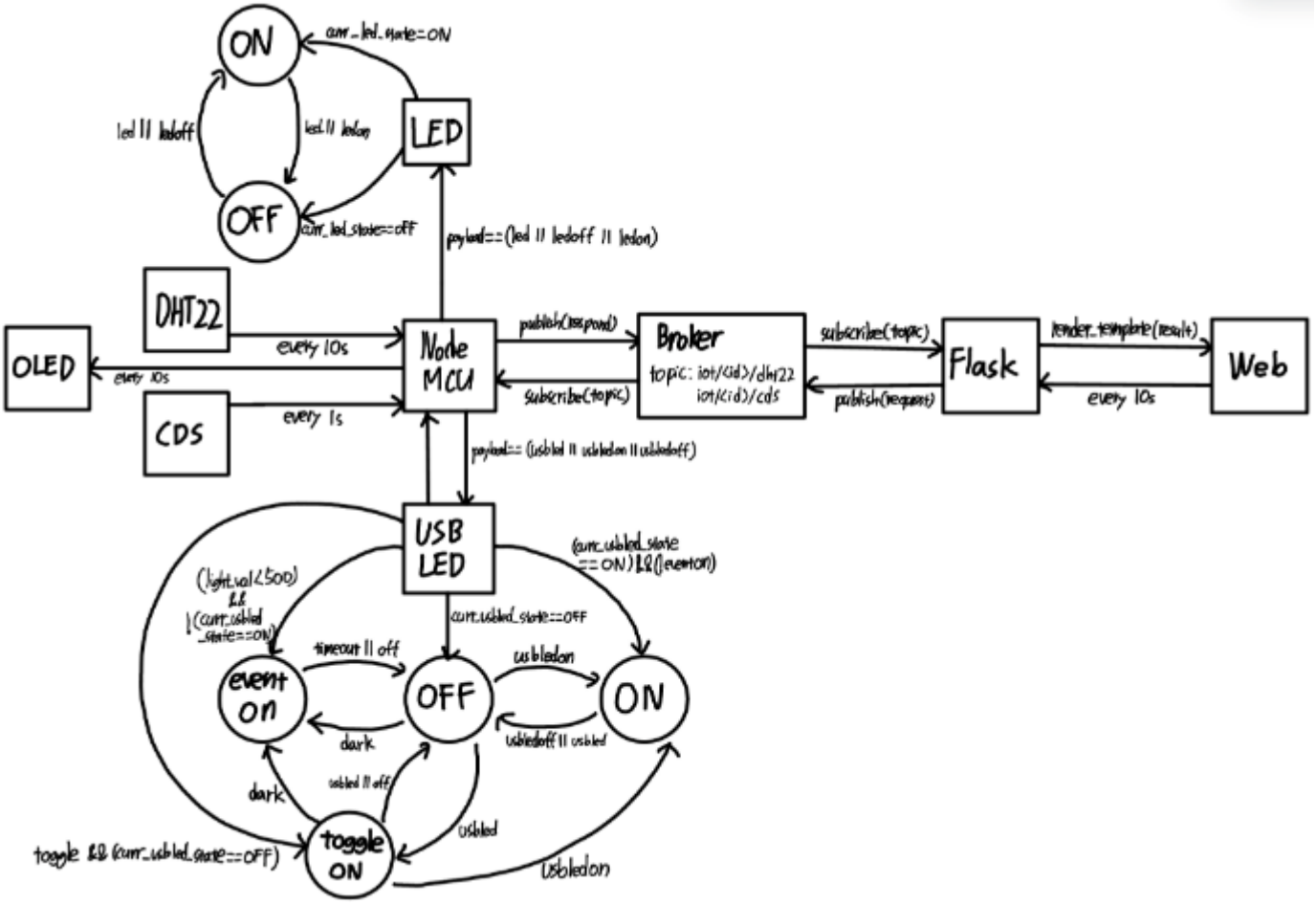
Internet of Things(IoT)의 구성요소는 크게 IoT Device, Server(or Cloud), 그리고 Application 이 있습니다. IoT 는 위 구성요소들이 모두 상호작용 해야하며, 다른 것들은 오작동하고 하나의 요소만 정상 작동한다면 User 가 원하는 기능을 수행하고 제공할 수 없습니다. 그렇기 때문에 IoT 에 관한 개발을 하기 위해서는 전체적인

흐름을 이해할 수 있어야 하고 각 구성요소가 상호작용하는 방식을 자세히 이해해야합니다.

본 프로젝트는 언제 어디서든 User 는 IoT 가 설치되어 있는 곳의 온습도를 알 수 있고, IoT 와 연결된 LED 를 통제할 수 있도록 서비스를 제공합니다. 해당 서비스를 제공하기 위해서는 IoT Device 와 Server 그리고 Application 측면에 존재하는 많은 요구 사항과 고려 사항을 충족 시켜야하지만 서비스를 제공할 수 있습니다. 간단한 기능을 제공하는 서비스임에도 불구하고 IoT 의 개괄을 모두 다루는 해당 프로젝트를 통해 IoT 동작원리와 기술 스택에 대해서 자세히 기술하고자합니다.

2. 개괄 및 재료

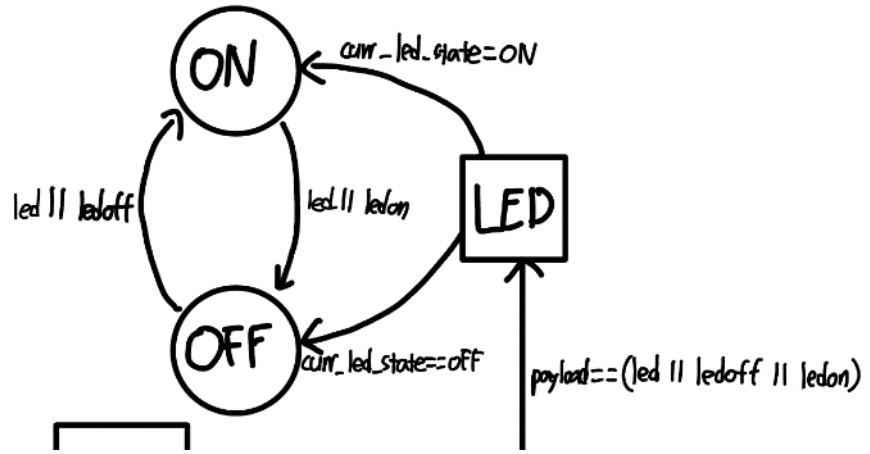
2.1. 프로젝트 개괄



<전체 개요도>

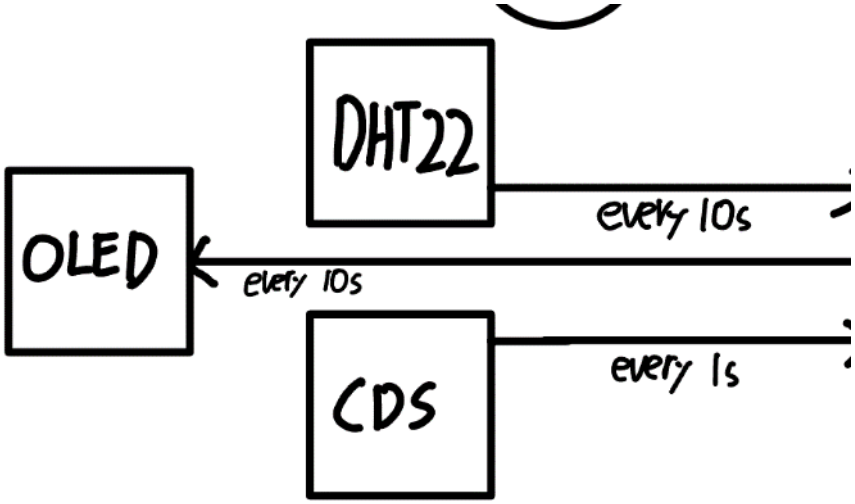
기능적으로 보았을 때 NodeMCU 는 USBLED, LED 그리고 온습도 센서와 작동하고, Flask(Server)는 Web 과 작동합니다. 그리고 NodeMCU 와 Flask 는 MQTT 라는 메세지 프로토콜을 사용해서 데이터 송수신합니다. MQTT 는 Broker 라는 시스템을 사용하는데 이는 Host 간의 직접적인 통신이 아닌 NodeMCU 의 Broker 와 Flask 의 Broker 를 통해 통신을 할 수 있게 합니다. 이러한 기술 스택은 End System 끼리의 request/respond 를 가능케 합니다.

NodeMCU 는 Flask 에서 보낸 요청에 따라 기능을 수행하거나 데이터를 보내주고, Flask 는 받은 데이터를 웹을 통해 User 에게 보여주는 역할을 합니다.



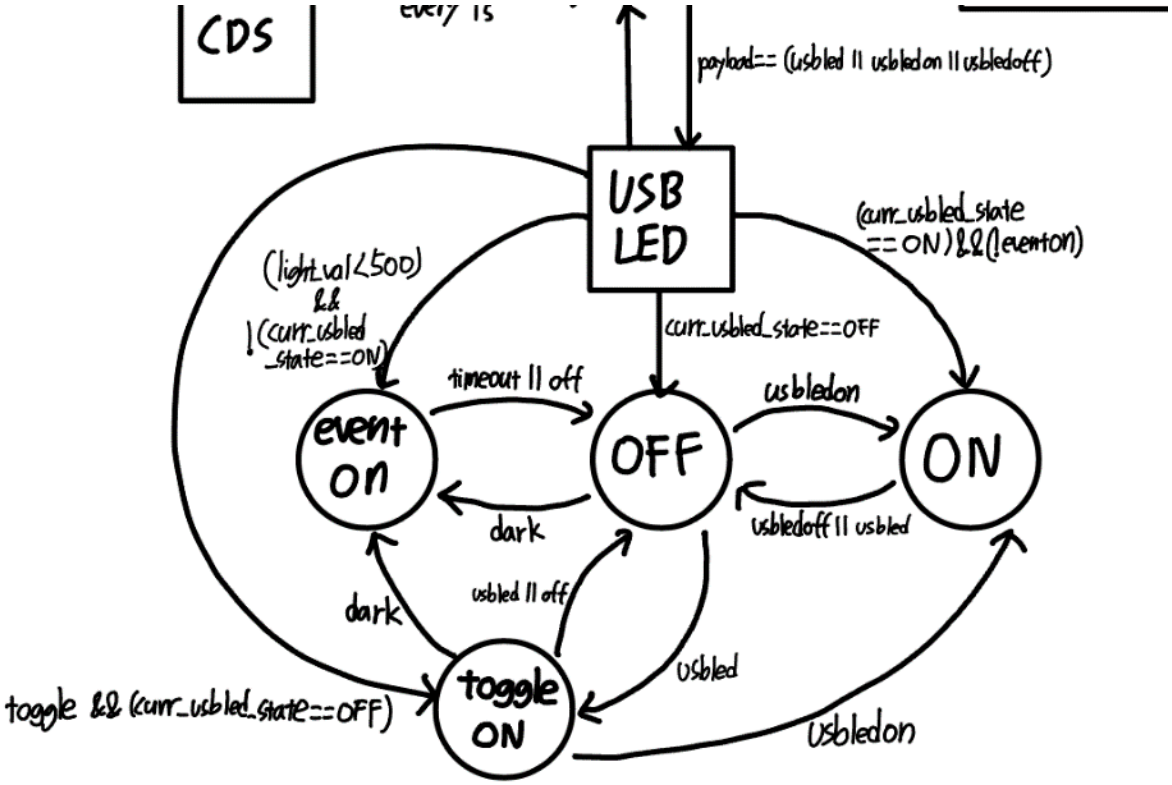
<LED 시나리오>

각 기능별로 시나리오를 설명하자면, 먼저 LED 시나리오는 Server 로부터 NodeMCU 가 받은 payload 가 led 이거나 ledoff 이거나 ledon 이면 시작됩니다. 그리고 현재 상태가 ON 상태이거나 OFF 상태일 것이 때문에 현재 상태에서 payload 를 처리해야하도록 위와 같이 조건을 나타냈습니다. 그리고 현재 상태에서 payload 에 따라 ledon, ledoff, led 를 처리하면 LED 시나리오가 완성됩니다.



<DHT22, CDS, OLED 시나리오>

DHT22, CDS, OLED 의 시나리오는 유사합니다. 먼저 DHT22 는 10 초마다 온습도를 측정해서 저장하면 됩니다. 그리고 동일한 과정으로 CDS 는 1 초마다 조도를 측정해서 저장하면 되는데, 10 초가 아니고 1 초인 이유는 CDS 는 USBLED 의 event 에 사용되는 값이기 때문에 조도값을 매번 측정하고 있어야하기 때문입니다. 그리고 OLED 는 DHT22 가 값을 측정할 때마다 그 값을 보여주는 역할을 합니다. 그렇기에 DHT22 가 동작할 때마다 같이 동작합니다.



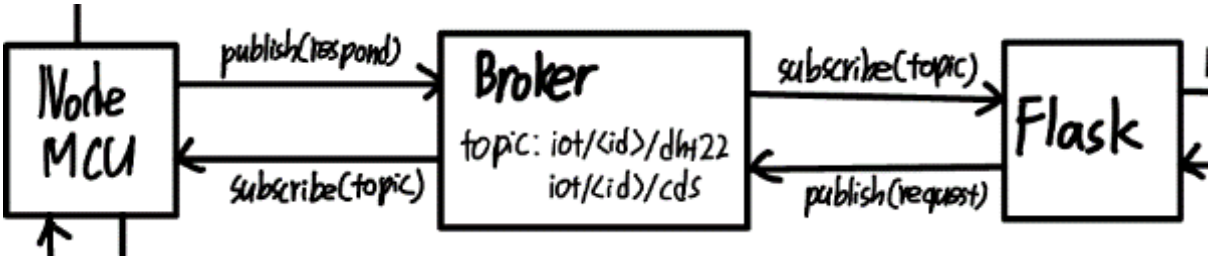
<USBLED 의 시나리오>

USBLED 의 시나리오는 기본적인 조건외에도 많은 복잡한 조건들을 가지고 있습니다. 복잡한 조건은 아래와 같습니다.

1. Event 상황에서 toggle 은 발생할 수 없다. 그렇지만 OFF 라는 우선상태는 발생할 수 있다.
2. Toggle on 인 상태에서 조도가 500 이하로 내려간다면 event on 으로 간주하게 한다. 그러나 우선상태인 ON 일 때는 이러한 과정이 발생하지 않는다.
3. Toggle on 에서는 2 번과 같은 조건이 있기 때문에 사용자가 원한다면 우선상태 ON 으로 갈 수 있어야한다.

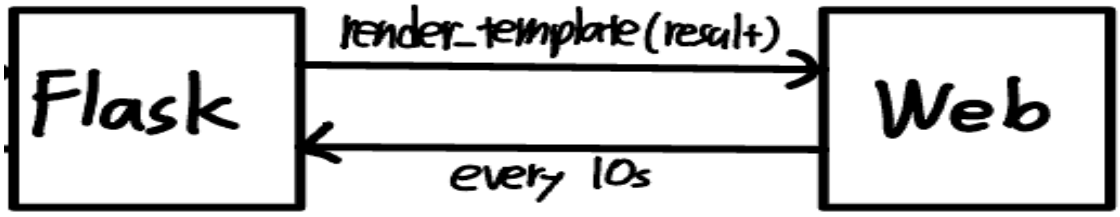
4. Event on 상태에서 timeout 되어 OFF 가 되었을 때 여전히 어두운 상태라면 event on 이 발생하지 않아야한다. 그러나 밝아졌다가 다시 어두워지면 event on 이 발생해야한다.

이러한 복잡한 조건과 기본적으로 제안되었던 조건들을 구현했을 때 USBLED 는 조도에 따라 자동으로 10 초 동안 켜지는 기능, 강제로 ON&OFF 하는 기능, Toggle 하는 기능으로 구현됩니다.



<NodeMCU 와 Flask 의 MQTT 통신 시나리오>

NodeMCU 와 Flask 는 MQTT 로 통신을 합니다. 그렇기 때문에 Broker 가 필요한데, 이 프로젝트 같은 경우에는 Public ip 를 가진 sweetdream.iptime.org 에 username 과 password 로 접속하여 통신합니다. 이는 NodeMCU 와 Flask 둘 다 동일하게 위 Broker 를 통해 통신합니다. Flask 가 request 를 보낼 때 목적에 맞는 topic 에 publish 하면 NodeMCU 는 그 topic 에 subscribe 하고 있다가 request 가 들어오면 그 request 에 대해 적절한 respond 를 Broker 에게 보냅니다. 이때 Flask 는 그 respond 를 받기 위해 해당 topic 에 subscribe 하고 있기 때문에 respond 를 받을 수 있게 됩니다. 이러한 시나리오를 통해 NodeMCU 와 Flask 는 통신합니다.



<Flask 와 Web 시나리오>

Flask 는 Web 에서 사용자의 요구를 route 로 처리합니다. 그리고 한가지 더 기능하는 것은 자동으로 10 초마다 refresh 하는 것입니다. 이때 통신 형태는 json 구조를 따르고 있습니다. NodeMCU 에서 Flask 로 전송한 데이터는 Brower 로 바로 전송되지 않습니다. Browser 는 개별적으로 10 초마다 Flask 서버에 정보를 요청하고 반환 받은 정보를 화면에 업데이트 합니다. 또한 버튼을 누르는 경우 이것은 Flask 서버를 거쳐서 Node MCU 까지 전달됩니다. 마지막으로 refresh 버튼을 누르면 Flask 서버에 정보를 즉시 요청하고 받은 값을 화면에 업데이트 합니다.

2.2. 프로젝트 구성 재료

IoT Device(HW)	NodeMCU	DHT22 센서	LED/USBLED	USB 커넥터	Relay	CDS 센서
IoT Device(SW)	EspMQTTClient.h (1.13.3)	DHTesp.h (1.19)	ArduinoJson.h (6.21.3)	C/C++		
Server	Raspberrypi 4	Python (3.9.2)	Flask (1.1.2)	MQTT (2.0.11)		
Application	Any Web brower					

3. 결과

3.1. 사용한 라이브러리

3.1.1. NodeMCU - ArduinoJson 라이브러리

Benoit Blanchon 이라는 사람이 만든 ArduinoJson 이라는 라이브러리를 활용해서 NodeMCU 에서 데이터를 JSON 형식으로 만들어 flask 에 전송할 수 있었습니다. 해당 라이브러리에 대해서는 개발자의 깃헙 repo 에서 더 자세히 확인 가능합니다 (https://github.com/bblanchon/ArduinoJson). JSON 은 개방된 표준 파일 형식으로 Javascript 에서 시작되었지만 프로그래밍 언어의 제약을 받지 않는 text 형식을 가지고 있어서 다른 언어나 플랫폼에서도 데이터를 송수신 하는 곳에 주로 사용되고 있습니다.

JSON 의 기본 형태는 property 와 value 가 짝을 이루고 있습니다. Property 는 문자열이어야 하지만 그 value 는 문자나 숫자도 가능합니다. 그리고 모든 object 는 중괄호 “{}”로 둘러싸여 있습니다. 예를 들어 {“apple”:1, “pear”:2} 는 하나의 object 로 “apple”과 “pear” 2 가지 property 가 있으며, 각각 1 과 2 라는 value 를 가지고 있습니다.

다음은 사용법 입니다.
해당 라이브러리를 사용하기 위해서는 먼저 라이브러리를 import 해 줍니다.

```
#include <ArduinoJson.h>
```

이후 StaticJsonDocument object 를 하나 정의하고 “<>” 안에 그 byte 크기를 정의 합니다. Dht22 센서의 값은 크지 않기 때문에 100byte 으로 정의 하였습니다. 그리고 object 에 대괄호 “[]”를 사용하여 그 안에 문자열 값을 넣으면 그것이 property 로 설정이 되고 이후 “=” operator 다음에 나오는 값이 해당 property 의 value 가 됩니다.

```
StaticJsonDocument<100> dht22_doc;
dht22_doc["dht22_h"] = humidity;
dht22_doc["dht22_t"] = temperature;
```

위에서 만들어진 json object 는 “[]” 연산자를 사용해 특정 property 의 값을 쉽게 읽고 쓸 수

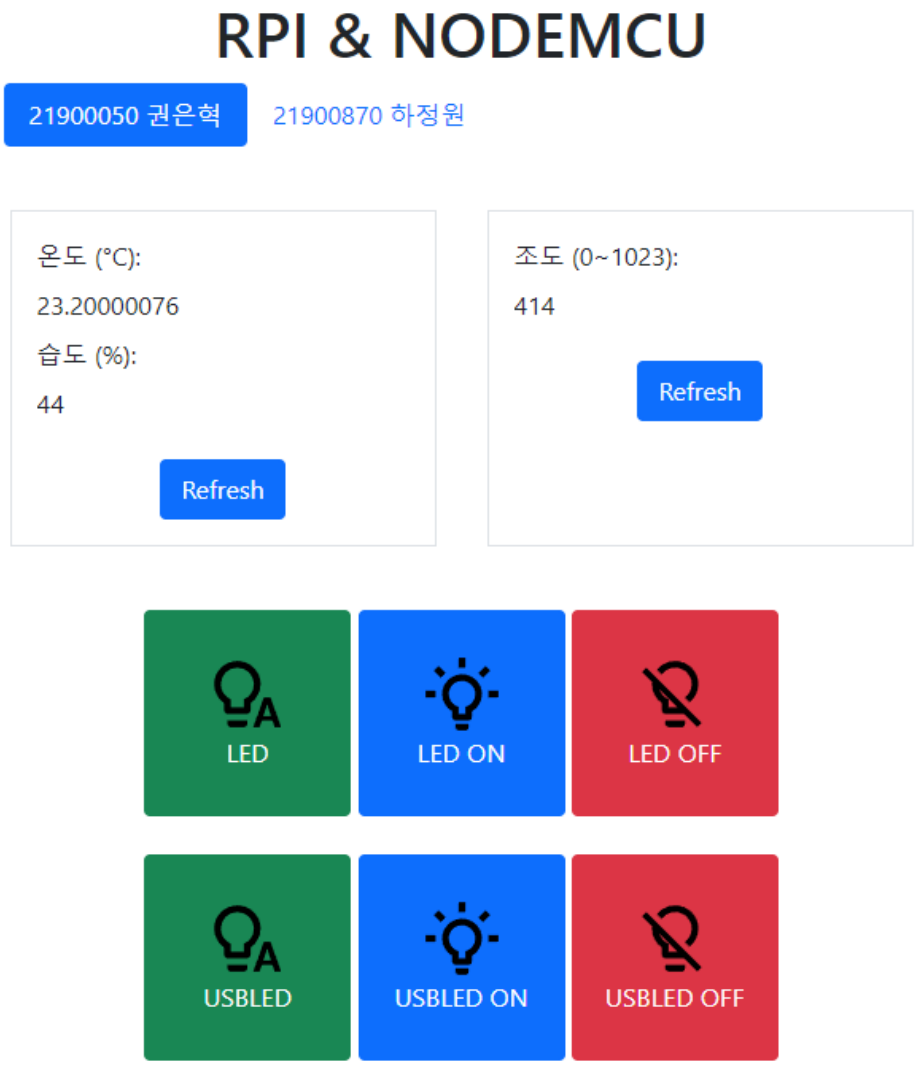
있지만, 문자열의 형태를 갖추고 있지 않습니다. 이것은 전송될 수 없는 형태로 메모리 속에 byte 형태로만 저장되어 있는 것입니다. 따라서 serialization 을 통해 전송 가능한 형태로 만들어 줘야 합니다. serializeJson() 함수는 JSON object 와 String 변수를 입력받아서 JSON object 를 String 변수 안에 문자열 형태로 저장합니다.

```
String output;
serializeJson(dht22_doc, output);
```

이제 output 이라는 변수에 문자열 형태로 JSON object 가 저장되었습니다. 이제 이 값을 출력하거나 MQTT 라이브러리를 사용해서 publish 할수 있게 되었습니다.

3.2. Web page

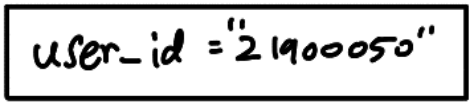
먼저 웹 페이지 구현은 ChatGPT 의 도움을 받아 제작한 것임을 밝힙니다. HTML 부분에서 ChatGPT 의 도움을 받았고, JavaScript 는 직접 제작했습니다.



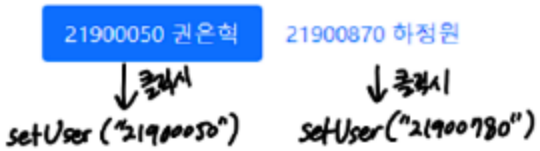
< 기본 페이지 화면 >

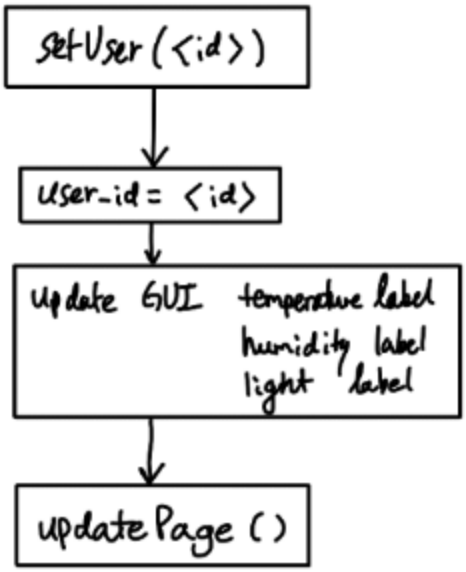
Flask 서버를 키면 접속할 수 있는 웹 페이지입니다. 상단의 제목 바로 아래 학번과 이름이 적힌 버튼을 클릭하면 화면에서 보여주는 온도, 습도, 조도 값과 버튼이 작동하는 사용자를 변경할 수 있습니다. 온도와 습도는 동일한 센서에서 측정하기 때문에 두개를 묶어서 Refresh 버튼이 있습니다. Refresh 버튼은 Flask 에서부터 값을 새로 받아와서 화면에 업데이트 합니다. 모든 센서의 값은 10 초에 한번씩 서버로부터 새로 받아와서 업데이트 합니다.

Javascript 부분에는 4 가지 함수가 존재합니다. setUser(), updatePage(), sendCommand(), setInterval(). 각각의 함수는 특정 event 가 발생했을 때 실행됩니다. 그리고 global 변수로 user_id 가 정의되어 있습니다. 이 변수는 초기에 “21900050” 값을 가집니다.



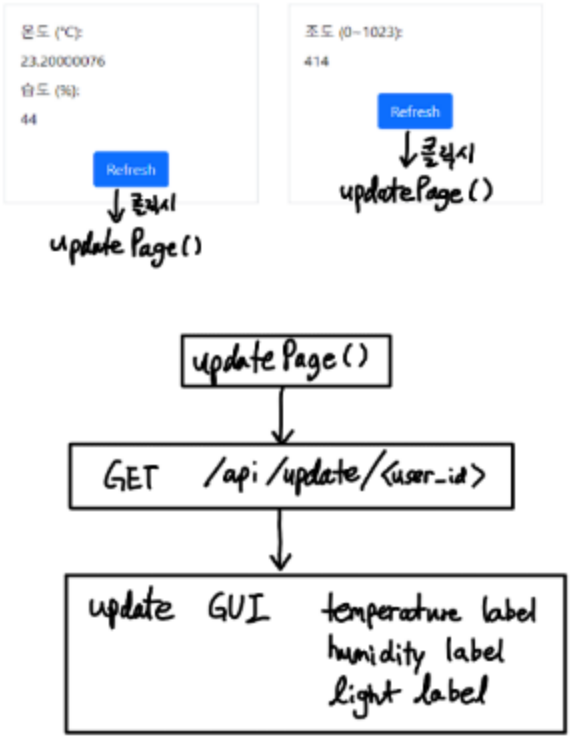
< global 변수: user_id >





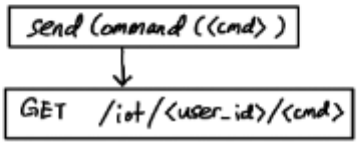
< 학번과 이름이 있는 버튼을 눌렀을 때 >

먼저 setUser() 함수는 페이지 제목 바로 아래에 있는 학번과 이름이 적혀 있는 버튼을 눌렀을 때 호출됩니다. “21900050 권은혁” 버튼이 눌리면 함수의 argument 로 “21900050”이 들어가고, “21900780 하정원” 버튼이 눌리면 함수의 argument 로 “21900780”이 들어가게 됩니다. 함수 내부에서는 크게 3 가지 기능을 수행하는데, 먼저 넘겨 받은 인자로 user_id 변수를 업데이트 하고, 다음으로 페이지 화면에 표시되는 문구를 “loading…”으로 수정하고 마지막으로 updatePage() 함수를 호출합니다. updatePage() 함수는 아래에서 설명 하고 있습니다.



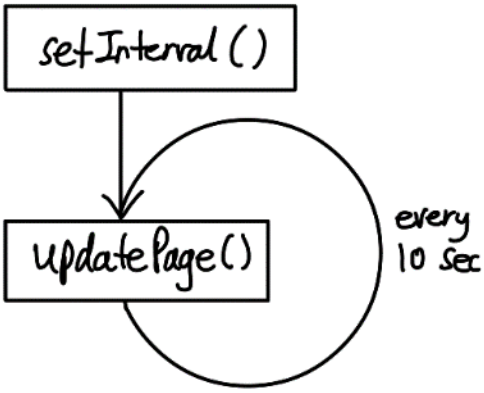
< 온도와 습도 Refresh and 조도 Refresh >

온도와 습도 label 아래에 있는 Refresh 버튼과 조도 label 아래에 있는 Refresh 버튼을 클릭하면 updatePage()함수가 호출됩니다. 해당 함수가 호출되면 “/api/update/<user_id>” URL 에 데이터를 요청하고 서버로부터 응답을 받아서 응답 받은 데이터를 사용해 페이지에 표시되는 온도, 습도, 조도 label 을 업데이트 합니다.



< LED 와 USBLED 버튼들 >

위에 버튼이 눌리게 되면, 버튼 아래에 적혀있는 것과 같이 버튼에 해당하는 명령 값이 argument 로 들어가는 sendCommand() 함수가 호출됩니다. 이 함수는 간단하게 “iot/<user_id>/<cmd>” URL 에 요청을 합니다.



< setInterval() >

마지막으로 setInterval() 함수입니다. 이 함수는 Javascript 내장 함수로 함수와, 시간을 입력 받아서 입력받은 시간마다 함수를 실행합니다. 이 프로젝트에서는 updatePage() 함수와 10 을 argument 로 사용하여, 매 10 초마다 updatePage()를 실행하여 사용자가 버튼을 클릭하지 않아도 페이지가 업데이트 되도록 하였습니다.

3.3. Flask Server

서버에 접속 할 수 있는 URL 의 종류는 크게 3 가지로 분류됩니다.

1. Command URL
2. API URL
3. Other URL

먼저 Command URL 은 NodeMCU 에 명령을 내리기 위해 사용됩니다.

<server addr> / iot / <sid> / <cmd>

'/' 앞뒤의 띄어쓰기는 단순히 가독성을 위한 것이고 "<>" 사이에 있는 값들은 치환 가능한 변수를 의미합니다. 다음은 각 변수에 들어갈 수 있는 값에 대한 설명입니다:

<server addr>는 서버의 주소를 나타냅니다. 예를 들어 192.168.0.95:5000 이 될 수 있습니다.

<sid>는 학생의 학번을 나타냅니다. 저희 프로젝트의 경우 21900050 또는 21900780 이 될 수 있습니다.

<cmd>는 NodeMCU 에게 보낼 명령을 나타냅니다. 다음 값중 하나가 될 수 있습니다: *led, ledon, ledoff, cds, dht22, usbledon, usbledoff, usbled*.

글로 된 설명을 테이블로 표시한 내용입니다:

<server addr>	**<sid>**	**<cmd>**	**설명**
<server addr> / iot	/ **21900050**	/ **led**	led 를 toggle 합니다
	/ **21900780**	/ **ledon**	led 를 킵니다
		/ **ledoff**	led 를 끕니다
		/ **usbled**	usbled 를 toggle 합니다
		/ **usbledon**	usbled 를 킵니다
		/ **usbledoff**	usbled 를 끕니다
		/ **cds**	조도 값을 새로 받아옵니다
		/ **dht22**	온습도 값을 새로 받아옵니다

API URL 은 Flask 서버로부터 정보를 받아오기 위해 사용됩니다. 받아오는 정보는 특정 학생의 NodeMCU 에서 Flask 서버로 전송하여 저장된 온도, 습도, 조도 값입니다. 브라우저에

API URL 을 통해 서버에 요청하면 서버에서 위에서 말한 3 가지 정보를 JSON 형태로 반환합니다.

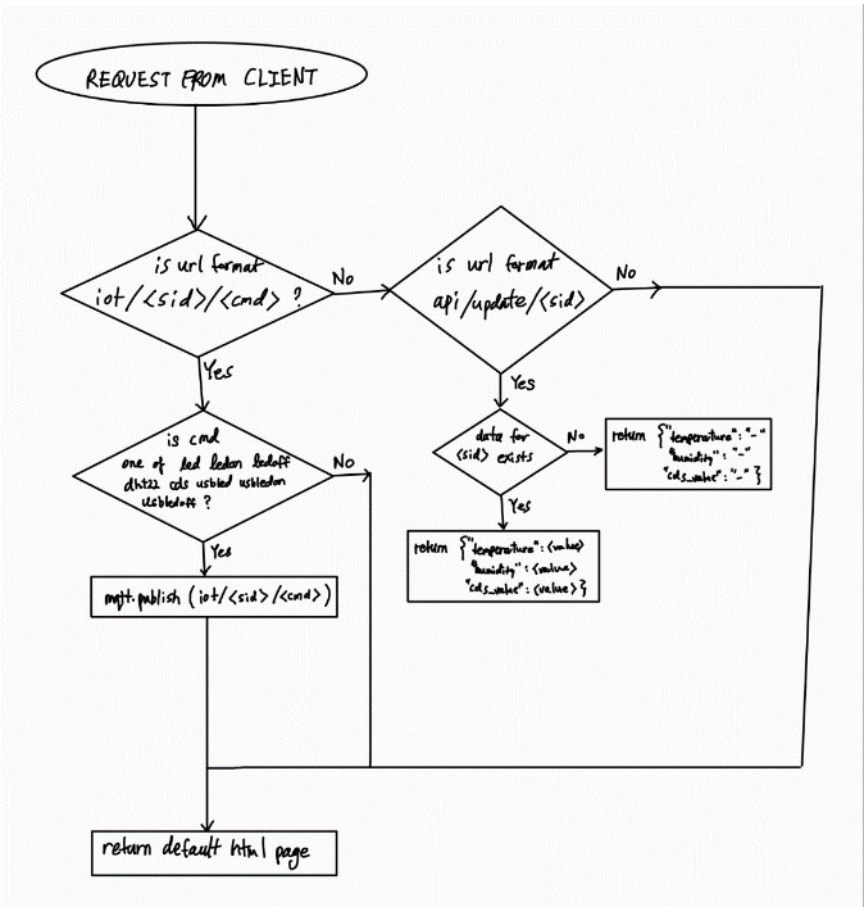
API URL 의 형태는 다음과 같습니다.

<server addr> / api / update / <sid>

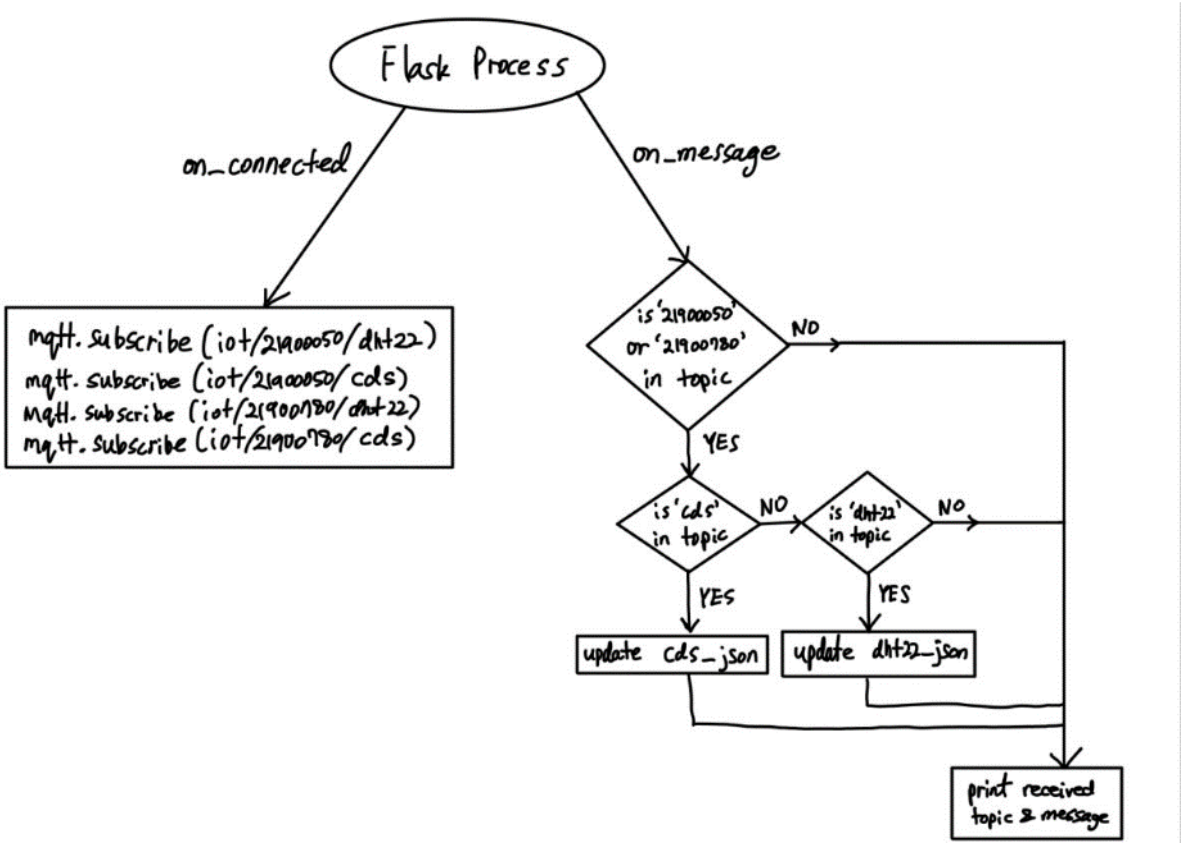
위 주소도 마찬가지로 '/' 앞뒤의 띄어쓰기는 단순히 가독성을 위한 것입니다. <server addr>은 서버의 주소를 나타내고 <sid>는 학번을 의미합니다. 서버에 저장되어 있는 값을 반환하고 만약에 저장된 값이 없다면 3 가지 property 에 대응되는 값에 '-'를 넣어서 반환합니다.

마지막으로 Other URL 은 위에 나오지 않은 다른 모든 URL 경로를 의미합니다. 주소창에 어떤 URL 을 입력하던지 기본 페이지를 보여주기 위해 Flask 에서 @app.errorhandler(404) 이라는 함수를 사용해서 찾을 수 없는 route 에 대해 전부 기본 html 페이지를 반환 하도록 만들었습니다.

아래는 Flask 에서 URL 의 경로를 처리하는 방법에 대한 flow chart 입니다.



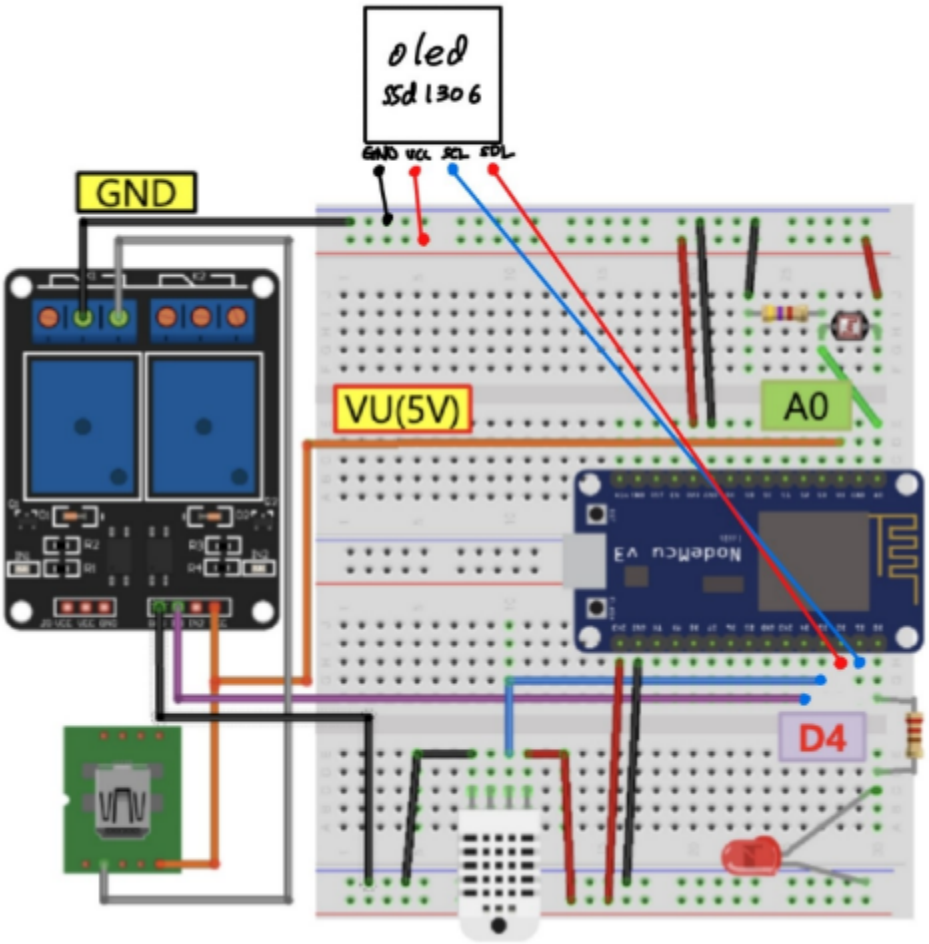
< URL 경로 처리 Flow Chart >



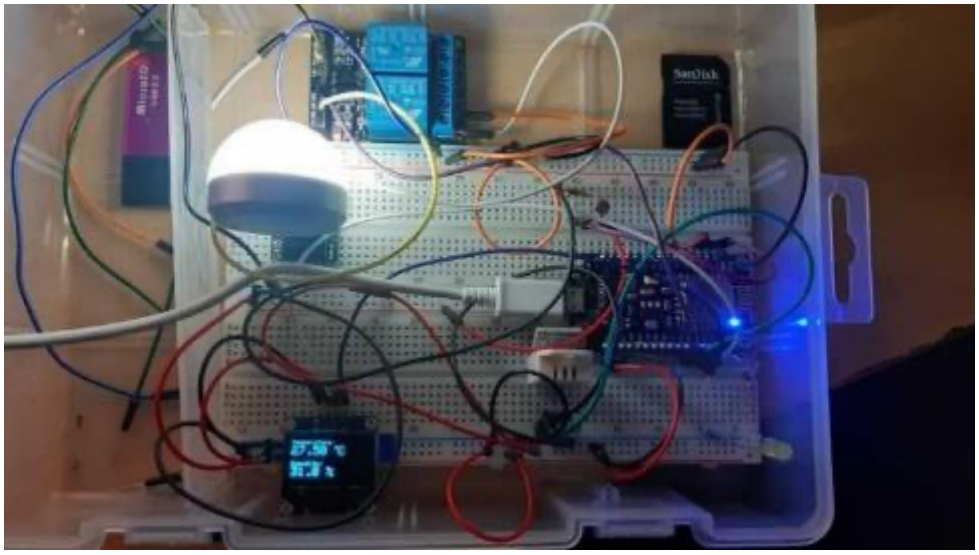
< MQTT 관련 flow chart>

서버에서는 URL 입력 외에도 MQTT 연결 관련해서 작업들을 수행하고 있습니다. 위는 Flask 에서 돌아가고 있는 MQTT 관련 함수들을 그린 것입니다. on_connected 는 MQTT 가 브로커에 처음 연결 되었을 때 실행 되는 것으로 필요한 topic 들을 subscribe 합니다. on_message 는 subscribe 한 topic 에서 메세지가 왔을 때 실행됩니다. 먼저 받은 toipc 의 학번을 확인을 한 다음 cds 의 값에 대한 것인지 dht22 에 대한 것인지 확인을 해서 flask 에 저장되어 있는 global 값을 업데이트 합니다. 그리고 마지막으로 디버깅을 위해 console 에 topic 과 message 값을 출력합니다.

3.4. NodeMCU



< NodeMCU 회로 구성도 >

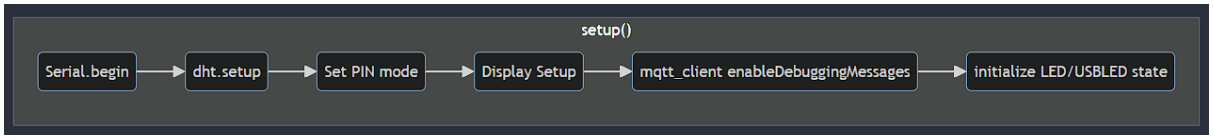


< NodeMCU 실제 모습 >

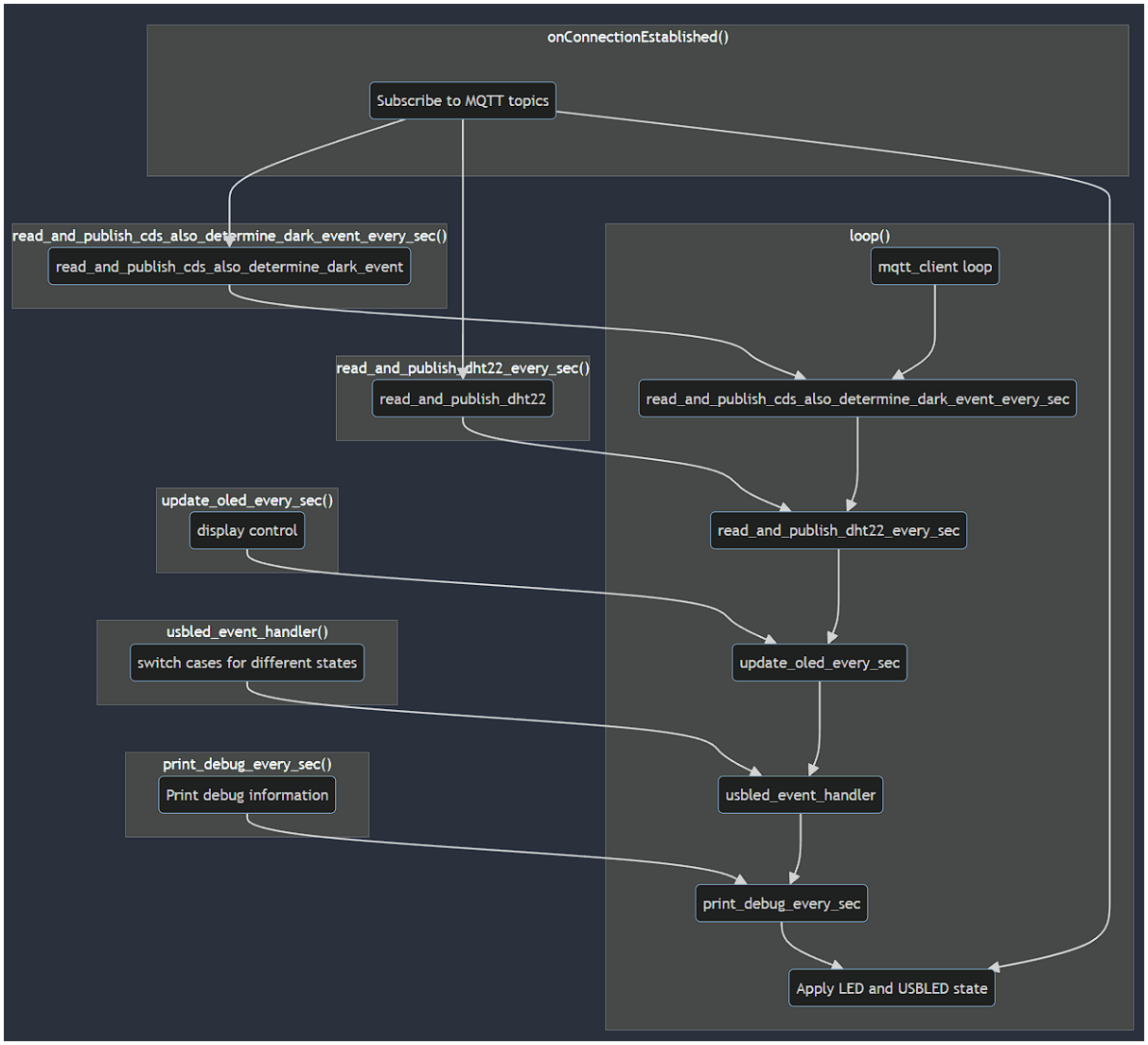
< oled 화면에 출력된 온습도 >

NodeMCU 의 최종 모습은 위와 같습니다.

코드를 이 문서에서 전부 다 설명하기는 어렵기 때문에 ChatGPT 에게 제 코드를 주고 Mermaid 다이어그램을 그려달라는 prompt 를 써서 얻어낸 결과물입니다 (https://mermaid.live).



< setup() 함수 흐름 >



< onConnectionEstablished() 와 loop() 함수 흐름 >

위에 보시는 것과 같이 loop() 함수 내에서는 어떤 기능들이 실행되는지 최대한 알아보기 쉽게 하기 위해 모든 기능들을 다 함수 형태로 만들어서 사용했습니다,

onConnectionExtablished() 함수에서는 Flask 서버와 통신하기 위해 정해진 topic 에 subscribe 합니다. 그리고 받은 메세지에 따라서 read_and_publish_cds_also_determine_dark_event_every_sec() 함수와 read_and_publish_dht22_every_sec() 함수가 호출 될 수 있다는 것이 화살표로 표시되어 있습니다. 그 외에는 기본적으로 모두 loop() 함수에서 위 diagram 에 나와있는 순서대로 호출이 됩니다.

4. 토론

4.1. 해결한 문제점

4.1.1. 구현해야 될 기능이 조금 복잡했는데 Finite State Machine 이번 프로젝트를 진행하면서 해결한 여러 문제들 중 가장 큰 것은 NodeMCU 의 내부 코드 구현이었습니다. USB LED 를 요구사항 대로 정확하게 조절하는 코드를 만들기 위해서는 USB LED 의 상태와 event 에서 수행되는 기능이 명확하게 정의되어 있어야 했는데 FSM 을 그려서 쉽고 정확한 기능을 하는 코드를 구현 할 수 있었습니다.

4.1.2. 와이파이가 연결 안되는 문제가 있었습니다. 해당 문제는 loop 반복문의 복잡성을 줄여서 해결 할 수 있었습니다. 정확한 원인은 찾지 못했습니다.

4.1.3. 보고서를 어떻게 작성할지에 대한 의견 충돌이 있어서 의견을 주고받는 과정에서 여러 참고 서적이나 인터넷 블로그를 찾아 보면서 의견을 조율 할 수 있었습니다. 결과적으로 잘 갖추어진 보고서를 작성할 수 있었습니다.

4.2. 해결하지 못한 문제점

4.2.1. 네트워크가 끊겼을 때 컨트롤 안되는 것이 있습니다. 이것은 명령 실행에 대한 보장성이 없다는 뜻인데 프로젝트를 더 발전시키기 위해서는 해결하거나 최소한 메뉴얼을 정해야 된다고 생각했습니다.

4.2.2. 웹 페이지에서 NodeMCU 와 연결되었는지 확인이 안된다는 점입니다. 그리고 이것과 비슷하게 LED 불이 켜졌는지 꺼져 있는지 모른다는 문제점도 있습니다. 이것은 없으면 같은 공간 안에서는 사용자가 직접

확인을 할 수 있지만 외부에서 원격으로 LED 를 키고 끄는 기능에 대해서는 보장이 되지 않는다는 단점이 있습니다.

4.2.3. 퍼블릭 IP 주소의 부재로 동일한 와이파이 내에 있지 않으면 웹 서비스 제공이 불가능 하다는 점이 있습니다. 하지만 마지막 문제는 나중에 필요하다면 호스팅 서비스를 구매하는 것으로 해결 할 수 있을 것으로 보입니다.

5. 결론

본 프로젝트는 User 가 IoT 가 설치되어 있는 곳의 온습도를 알 수 있고, IoT 와 연결된 LED 를 통제할 수 있는 서비스를 제공합니다. 또한 IoT Device, Server, 그리고 Application 간의 Control Flow 와 각 Edge 의 기능과 구현 방식에 대해서 자세히 기술하고 있습니다. 그러므로 본 프로젝트는 IoT 동작원리와 서비스 제공 측면에서 기초적인 지식을 겸비하는데 유익한 프로젝트이며, 이러한 목적을 충족시키고자 하는 이들에게 좋은 Method 가 될 것입니다.

5.1. 배운 점

5.1.1. Client-Server 가 실질적으로 어떻게 작동하는 지에 대해서 알게 되었습니다. 네트워크 시간에 배웠던 Client-Server 구조를 IoT 라는 분야를 통해서 구현해봄으로써 해당 개념을 학습하는데 유익했습니다.

5.1.2. 왜 Flow Graph 를 먼저 그리는지 알게 되었습니다. 구현 해야하는 실제 제품이 얼마나 많은 경우 수와 예외사항을 가지고 있을 지 모르는데 이를 코딩하면서 머리로 처리하는 것은 굉장히 위험한 행위입니다.

그렇기 때문에 코드를 써내려가기 전 먼저 해당 Flow 와 필요 재료들을 정리 해놓는 것이 참 중요하다는 것을 알게 되었습니다.

5.1.3. 보고서를 갖춘 체계적인 프로젝트가 무엇인지 알게 되었습니다. 지금까지 프로젝트라는 이름의 과제를 진행했더라도 이런 정식적인 형태의 보고서를 갖추거나 README.txt, 소스코드 문서화 등을 이렇게 깊게 까지 해본 적은 없습니다. 이번 기회를 통해서 프로젝트란 무엇인지를 알게 되었습니다.

5.2. 향후 발전 방향

본 프로젝트는 아주 간단한 서비스를 제공하는 MVP 수준의 IoT 시스템입니다. 하지만 대부분의 IoT 시스템이 이러한 과정을 기반으로 만들어져 있기에 발전 시킬 수 있는 사항이 많습니다.

1. 단순한 ON/OFF/Toggle 이 아닌 조도에 따라 USBLED/LED 밝기를 조절하는 기능을 추가 할 수 있습니다. 이는 digital 함수가 아닌 analog 함수로 가능할 것으로 보입니다.
2. 스마트폰 Application 으로 서비스를 제공할 수 있습니다. 현재는 Flask 로 Web Browser 을 통해 서비스 받을 수 있는데, User 에게 보다 편한 서비스 제공을 위해서 Application 으로 서비스를 받을 수 있도록 구현할 수 있습니다.
3. 현재 없는 기능을 추가할 수 있습니다. 현재 상용화되고 있는 많은 IoT 기능들을 동일하게 수행할 수 있습니다. 왜냐하면 그 원리는 현재 프로젝트에서 크게 벗어나지 않기 때문입니다. 예를 들어 에어컨

ON/OFF, 보일러 ON/OFF, 가스 조절 등 외부에서도 본인의 집을 control 할 수 있는 기능들을 직접 구현할 수 있습니다.

본 프로젝트는 이 외에도 많은 발전 가능성을 가집니다. 왜냐하면 그만큼 기초개념으로 이뤄져있기 때문입니다. 이 프로젝트를 시작으로 하나씩 구현해나가며 결국에는 본인의 주거공간을 Smart home 으로 만드는 것이 가능해집니다.