2023-2 IoT System Design

HW #04

Smart Home

Due 11.18(Sat) 22:00

전산전자공학부

22100579 이진주

1. **과제 설명**

이번 과제는 지난 설계과제#1의 내용에 기반하여 온습도 데이터 측정과 MQTT publishing이 이루어지되, 세부 내용을 바꾸고 몇 가지 기능을 추가한 형태이다. 주요 변경점은 OLED와 무선 AP 연결 기능의 추가, 제어를 위한 인터페이스가 Flask 웹 페이지가 아닌 Smart Home Platform: HASS(Home Assistant)로 이루어진다는 것이다.

요구 기능/동작

* NodeMCU
  + 온습도 데이터는 3초 단위로 측정된다
  + sensor에서 측정된 데이터는 지난 설계 과제의 MQTT format을 따라 publishing 되며, actuator는 같은 topic을 subscribe하여 payload에 따라 동작한다.
    - led -> ON if LED was OFF, OFF if LED was ON
    - ledon -> ON LED
    - ledoff -> OFF LED
    - usbled -> ON if USBLED was OFF, OFF if USBLED was ON
    - usbledon -> ON USBLED
    - usbledoff -> OFF USBLED
  + NodeMCU와 연결된 OLED에 온습도 데이터를 display한다.
  + WiFiManager를 사용하여 임의의 AP에 접속하도록 설정한다.
* Smart Home Platform인 HASS(Home Assistant) 인터페이스를 사용하여 LED/USBLED 액츄에이터가 조작 가능하도록 한다.
  + 버튼: LED/USBLED 제어 버튼. 조작에 따라 mqtt의 iot/[학번]으로 정해진 keyword가 publishing된다.
    - Switch.LED -> led
    - Button.LED\_ON -> ledon
    - Button.LED\_OFF -> ledoff
    - Switch.USBLED -> usbled
    - Button. USBLED\_ON -> usbledon
    - Button. USBLED\_OFF -> usbledoff
  + 온습도, 조도값 display: NodeMCU 측정값, iot/nth405/MQTT/{dht22\_t, dht22\_h, cds} 토픽을 통해 전달된 값
  + 자신의 위치가 표시된 map card
  + Event에 의한 NodeMCU의 동작 제어
    - 어두워지는 이벤트가 발생하면 10초간 USBLED의 불을 켠다.
    - 계속 어두운 상태는 USBLED 상태에 영향을 주지 않는다.
    - 어두운 상태에서 밝은 상태로 변할 경우 USBLED 상태에 영향을 주지 않음
    - State diagram:

텍스트, 그림, 아동 미술, 친필이(가) 표시된 사진

자동 생성된 설명

1. **nodeMCU code**
2. Global scope

|  |
| --- |
| /\*hw4\_nodeMCU\_DHT\_OLED\_HASS.ino\*/  #include "EspMQTTClient.h"  #include "DHTesp.h"  #include <Adafruit\_SSD1306.h>  #include <WiFiManager.h>  #define LED\_PIN D0  #define USBLED\_PIN D4  #define DHT\_PIN D3  #define CDS\_PIN A0  #define DHTTYPE DHTesp::DHT22  #define LED\_ON HIGH  #define LED\_OFF LOW  #define USBLED\_ON LOW  #define USBLED\_OFF HIGH  #define SCREEN\_WIDTH 128 // OLED display width, in pixels  #define SCREEN\_HEIGHT 32 // OLED display height, in pixels  #define OLED\_RESET -1 // Reset pin # (or -1 if sharing Arduino reset pin)  #define SCREEN\_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32 |

Mqtt 통신, sensor 및 actuator 사용, WiFi manager 사용을 위한 라이브러리를 include한다.

이후 코드들의 가독성을 위해 pin 번호와 state 등의 constant 정보들을 define해준다.

|  |
| --- |
| Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);  DHTesp dht;  float temperature = 0, humidity = 0;  int lightValue = 0, prevLightValue = 0;  int led\_state, usbled\_state;  String mqtt\_payload;  long long timer = 0;  #define interval 3000 |

OLED, DHT22, cds 사용을 위한 설정, non-blocking을 위한 요소들을 정의한다.

|  |
| --- |
| WiFiManager wm;  WiFiManagerParameter wifi\_ssid("SSID", "default\_ssid", "Enter SSID", 20);  WiFiManagerParameter wifi\_psk("Password", "default\_password", "Enter Password", 20);  bool res;  char wifissid\_char[128];  char wifipassword\_char[128];  String wifissid = "....";  String wifipassword = "....";  #define mqtt\_broker "sweetdream.iptime.org"  #define MQTTUsername "iot"  #define MQTTPassword "csee1414"  #define mqtt\_clienname "JJLee"  EspMQTTClient mqtt\_client(    wifissid\_char,    wifipassword\_char,    mqtt\_broker,  // MQTT Broker server ip    MQTTUsername,   // Can be omitted if not needed    MQTTPassword,   // Can be omitted if not needed    mqtt\_clienname,     // Client name that uniquely identify your device    1883              // The MQTT port, default to 1883. this line can be omitted  );  const char \*pub\_dht = "iot/22100579/dht22";  const char \*pub\_cds = "iot/22100579/cds";  const char \*sub\_topic = "iot/22100579";  int check\_wifi = 1;  int check\_mqtt = 1; |

WiFi manager를 위한 설정을 해준다.

Ref: <https://microdigisoft.com/wifimanager-with-esp8266-autoconnect-without-having-to-hard-code/>

WiFiManager 인스턴스를 하나 선언하고, 연결 정보를 받아오기 위해 WiFiManagerParameter도 설정해둔다. SSID와 Password 정보를 받아와 mqtt client의 연결에 사용할 것이다.

연결 시도의 결과값을 받아오기 위한 Boolean 변수도 설정해주고, 연결 정보(SSID, Password)를 저장할 String 형태의 변수들, 이를 mqtt client에서 사용 가능하도록 형변환 시켜줄 char array 타입의 변수들을 선언해준다.

그리고 mqtt 연결을 위한 정보들을 define하고, EspMQTTClient 구조체에 정보들을 담아준다. 이후 사용하기 위한 topic들, 연결 정보 확인을 위한 변수 등도 여기에 함께 선언해준다.

1. connectWifi

|  |
| --- |
| bool connectWifi(){    res = wm.autoConnect(); // password protected ap    if(!res) Serial.println("Failed to connect");    else Serial.println("connected...yeey :)");    return res;  } |

WiFiManager 라이브러리의 autoConnect()를 사용하여 주변 임의의 와이파이를 연결하는 함수이다.

일반적으로는 최초로 연결을 시도할 SSID와 password를 인자로 주나, 코드 내부의 하드코딩을 최소화하기 위하여 인자를 주지 않았다.

autoFonnect()는 연결 시도의 결과를 True/false로 반환하는데, 이를 res라는 Boolean 변수에 담아 결과에 따른 Serial 출력이 가능하도록 하였다.

1. setup

|  |
| --- |
| void setup(){    Serial.begin(115200);    Serial.println("... setup ...");    wm.addParameter(&wifi\_ssid);    wm.addParameter(&wifi\_psk);    pinMode(LED\_PIN, OUTPUT);  // LED    pinMode(USBLED\_PIN, OUTPUT);  // USBLED    pinMode(CDS\_PIN, INPUT);    dht.setup(DHT\_PIN, DHTTYPE);    digitalWrite(LED\_PIN, LED\_OFF);    digitalWrite(USBLED\_PIN, USBLED\_OFF);    // SSD1306\_SWITCHCAPVCC = generate display voltage from 3.3V internally    if(!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS)) {      Serial.println(F("SSD1306 allocation failed"));      for(;;); // Don't proceed, loop forever    }    display.setTextSize(1); // Draw 2X-scale text    display.setTextColor(SSD1306\_WHITE);    display.cp437(true);    // Show initial display buffer contents on the screen --    display.display();    while(!connectWifi()){      Serial.println("Cannot connect wifi... retrying....");    }    wifissid = WiFi.SSID();    wifipassword = WiFi.psk();    wifissid.toCharArray(wifissid\_char, wifissid.length()+1);    wifipassword.toCharArray(wifipassword\_char, wifipassword.length()+1);    led\_state = LED\_OFF;    usbled\_state = USBLED\_OFF;  } |

시리얼 모니터와 nodeMCU의 pin 초기설정을 해준 뒤, OLED display 또한 세팅해준다.

WiFi manager에 global scope에 정의해둔 파라미터를 추가한다. 이후 연결된 wifi의 정보를 받아오기 위함이다.

그리고 위에서 정의한 connectWiFi()를 call해 wifi manager를 통한 AP 연결을 시도한다. 성공 여부를 리턴값으로 받아 실패시 성공할 재시도하도록 하였다.

그리고 연결이 성공하여 while문을 빠져나온 후에는 연결 정보를 미리 준비해둔 변수에 담아주고, state들을 초기화해준다.

1. onConnectionEstablished

|  |
| --- |
| void onConnectionEstablished(){    Serial.println("... onConnectEstablished ...");    // about LED and USBLED control...    mqtt\_client.subscribe(sub\_topic, [](const String & payload) {      Serial.print("mqtt subscribe: ");      Serial.println(payload);        if(payload == "led"){        if(led\_state == LED\_ON) led\_state == LED\_OFF;        else led\_state = LED\_ON;      }      if(payload == "ledon") led\_state = LED\_ON ;      if(payload == "ledoff") led\_state = LED\_OFF ;      if(payload == "usbled"){        if(usbled\_state == USBLED\_ON) usbled\_state == USBLED\_OFF;        else usbled\_state = USBLED\_ON;      }      if(payload == "usbledon") usbled\_state = USBLED\_ON ;      if(payload == "usbledoff") usbled\_state = USBLED\_OFF ;    });    mqtt\_client.publish(sub\_topic, "Hello from nodeMCU"); //send it once    } |

Mqtt client를 통해 토픽을 subscribe해주고, 받은 payload를 판별하여 LED와 USBLED의 적절한 state를 결정한다.

1. Loop

|  |
| --- |
| void loop(){    if(check\_wifi == 1 && mqtt\_client.isWifiConnected()) {    Serial.println("Wifi connected!");    Serial.print("SSID: ");    Serial.println(wifissid\_char);    Serial.print("PW: ");    Serial.println(wifipassword\_char);    check\_wifi = 0;    }    if(check\_wifi = 0 && !mqtt\_client.isWifiConnected()) {      Serial.println("Wifi connection lost...");      connectWifi();    }    if(check\_mqtt == 1 && mqtt\_client.isMqttConnected()) {    Serial.println("MQTT connected");    check\_mqtt = 0;    }    mqtt\_client.loop(); //on    display.clearDisplay();    //sensing dht22 and pub dht22, cds    if(timer < millis()){      humidity = dht.getHumidity();      temperature = dht.getTemperature();      lightValue = analogRead(CDS\_PIN);      if(isnan(temperature) || isnan(humidity)) Serial.println("loop: cannot read from DHT22");      mqtt\_payload = String("{\"Temp\" : ") + String(temperature) + String(", \"Humi\" : ") + String(humidity) + String("}");      mqtt\_client.publish(pub\_dht, mqtt\_payload);      mqtt\_payload = String("{\"Light\" : ") + String(lightValue) + String("}");      // Serial.println("loop:" + mqtt\_payload);      mqtt\_client.publish(pub\_cds, mqtt\_payload);      display.setCursor(10, 5);      display.print(F("T: "));      display.print(temperature);      display.print(char(248)) ;      display.println("C");      display.display();      display.setCursor(10, 15);      display.print(F("H: "));      display.print(humidity);      display.println("%");      display.display();      // Show initial text      timer = millis() + interval;    }    digitalWrite(LED\_PIN, led\_state);    digitalWrite(USBLED\_PIN, usbled\_state);  } |

Wifi 연결과 mqtt 연결을 확인하고 연결 정보를 Serial에 출력한다. 만약 연결이 중간에 끊어진다면 다시 connectWifi()를 call하여 주변 AP와 연결을 시도한다.

Mqtt\_client를 실행시키고, non-blocking 형식으로 3초의 interval을 주어 온습도, 조도 측정 및 mqtt publishing을 실행한다. 이때 전송되는 payload는 하드코딩하여 JSON format으로 만들어주었다.

또한 측정된 온습도, 조도값을 OLED에 display해준다.

그리고 현재 state에 저장된 값을 nodeMCU에 써준다.

1. **yaml code**

hass의 설정을 위한 .yaml 파일이다.

1. Configuration.yaml

|  |
| --- |
|  |

기본 설정을 불러오고 테마 및 기타 설정 파일들을 include한다. Lovelace 설정은 yaml 모드로 하여 yaml파일에서의 변경점이 적용되도록 하였다.

|  |
| --- |
|  |

밝았다가 어두워지는 이벤트가 발생했을 때 10초간 USBLED에 점등 신호를 보내고 이후 소등되도록 trigger와 action을 설정하였다.

|  |
| --- |
|  |

기본 대시보드의 설정이다.

위치를 추적하도록 고정 위치 설정은 해제하고, 엔티티들이 표시될 아이콘을 설정해주었다.

|  |
| --- |
| 텍스트, 스크린샷, 메뉴이(가) 표시된 사진  자동 생성된 설명 |

Mqtt에 sensor, switch, button 엔티티를 두어 각 행동에 따라 정해진 topic으로 keyword를 publishing하고, subscribe된 payload에 따라 state를 설정하도록 했다.

|  |
| --- |
|  |

예쁘게 display되도록 그룹으로 묶어주었다.

1. Ui-lovelace.yaml

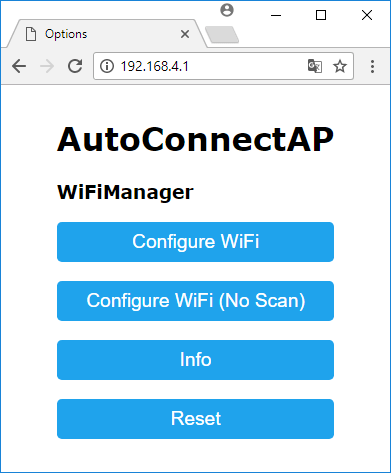
위의 configuration에서도 주요 기능들을 모두 포함한 ui를 완성할 수 있었으나, map card 기능은 버전 이슈로 추가할 수 없었기에 ui-lovelace를 사용하여 래핑하였다.

|  |
| --- |
|  |

위에서 정의해준 엔티티들을 display한다. group에서 서술한 것과 대부분 일치하도록 하였으며 map card를 추가하고 빈 공간에 인터넷 이미지 주소를 넣어 귀여운 햄스터 그림을 넣어주었다.

1. **결과 및 느낀점**
2. 결과

nodeMCU를 실행시키면 최초 1회 WiFi 연결을 위한 AP모드로 실행된다. Serial에 안내된 임시 AP로 연결한 기기에서 *192.168.4.1* 주소로 들어가면 AP 설정을 위한 웹 페이지를 이용 가능하다.



Configure WiFi button을 눌러 스캔된 와이파이들 중 임의의 것을 선택하고 정보를 입력하면, NodeMCU가 자동으로 재시작되며 연결된 WiFI와 함께 정상 동작하기 시작한다.

RPI에서 Hass를 활성화 한 후 동일 와이파이를 연결한 브라우저에서 *[rpi 주소]:8123* 접속시 확인 가능한 dashboard의 화면이다.

텍스트, 스크린샷, 포유류이(가) 표시된 사진

자동 생성된 설명

Map card를 통해 현재 위치가 표시된다.

My DHT22와 My Light intensity에는 nodeMCU의 sensor를 통해 측정된, mqtt로 전송된 값이 display된다.

NTH DHT22와 NTH Light intensity에는 mqtt의 특정 토픽 subscribe를 통해 받은 값이 display된다.

LED/USBLED card에는 LED를 조작하는 on/off 버튼과 USBLED를 조작하는 on/off 버튼이 위치한다.

Switch card에는 LED와 USBLED 값을 반전시킬 수 있는 토글이 위치한다.

그리고 햄스터 사진을 넣어 빈 공간을 채워주었다.

hass에서의 button, switch를 통한 조작에 따라 미리 정해진 topic에 keyword가 publish된다.

동일한 토픽을 subscribe 하고 있는 nodeMCU 또한 해당 keyword를 받아 payload에 저장하며, 그에 따라 적절한 제어를 수행한다.

|  |
| --- |
|  |

nodeMCU가 mqtt subscribe를 통해 입력된 payload와 각 keyword의 일치 여부를 판단하여 LED를 적절히 조작함을 확인하였다.

또한, 조도값이 500 이하로 내려갔을 때 USBLED가 10초간 점등하고 이후 점멸함을 확인하였다.

nodeMCU의 OLED에 hass에 display된 것과 같은 온습도 값이 display됨을 확인하였다.

1. 배운 점 및 느낀 점

Hass를 사용하니 커스텀할 수 있는 폭도 굉장히 넓고 깔끔하며 직관적인 인터페이스를 자유롭게 구성할 수 있어 아주 흥미롭고 재미있었다. 과제에서는 nodeMCU의 실습 키트를 제어해보았으나, 원리를 이해하고 응용한다면 더 다양한 범위에서, 실생활과도 밀접하게 사용할 수 있는 iot 환경을 직접 구성해볼 수 있을 것 같아 설레는 마음이 생겼다.

한국어 레퍼런스가 적고 버전이 계속해서 업데이트되며 변경점이 많아 구글링이 쉽지 않았지만, 결국 어디엔가는 같은 주제에서 고군분투하는 사람들이 있음을 확인할 수 있었고 동질감과 더불어 응원하는, 그리고 응원 받는 기분이 들렸다. 우여곡절 끝에 원하는 무언가를 실현시켰을 때의 뿌듯함도 한 몫 했다. 커뮤니티가 잘 활성화 되어있다는 것은 참 큰 이점인 것 같다. 여러 글들에 도움을 받았고, 나도 언젠가 이런 글들을 남겨 누군가의 도움이 되고 싶다는 생각을 했다.

라즈베리파이에서yaml 파일을 편집하며 vim의 다양한 기능을 사용해볼 수 있었다. vsCode 등의 IDE를 연결하는 방법도 있었으나, vim에 더 익숙해져 보고자 ssh와 CLI만을 사용해 보았는데, 옳은 선택이었던 것 같다. 한편으로는 indentation에 아주 예민한 yaml파일의 특성에 꽤나 고생을 했다. 눈으로 보기에 같아도 판정은 다른 경우가 종종 있는 것 같다. 시간이 된다면 이 문제를 확실히 확인하고 처리할 수 있는 방법을 알아보고 싶다.