

디지털 시계 제작 프로젝트 보고서

- 사용 부품
 - 1. NODE-MCU
 - 2. CDS
 - 3. ACTIVE-BUZZER
 - 4. DHT11
 - 5. 7-SEGMENT
 - 6. LED(RED)
 - 7. BUTTON
- 기능
 - 1. 시간, 날짜 출력

시간과 날짜를 출력한다.
날짜와 시간 중 버튼 혹은 웹을 통해 선택 가능하다.
웹을 통해 AM/PM과 24시간을 변경 할 수 있다.
 - 2. NTP를 이용한 자동 시간 설정

WiFi를 통해 시간을 국내 NTP서버로부터 받아온다.
 - 3. 시간대 변경

웹을 통해 시간대를 GMT기준으로 변경 가능하다. 기본값 +9
 - 4. 분 단위 시간 변경

웹을 통해 시간을 분단위로 변경 가능하다.
 - 5. 특정일 알람

원하는 날짜의 원하는 시간에 알람을 울린다.
웹으로 설정 가능하다.
버저와 LED로 표현한다. 버튼으로 끌 수 있다.
 - 6. 특정 요일 반복 알람

특정 요일의 원하는 시간에 알람을 울린다.
웹으로 설정 가능하다.
버저와 LED로 표현한다. 버튼으로 끌 수 있다.

7. 자동/수동 밝기 변경

CDS를 이용한 자동 밝기와 수동 밝기중 선택 가능하다.

기본 설정은 수동 밝기 7단계이며 웹으로 설정 가능하다.

8. 웹을 통한 온,습도 확인

웹을 통해 DHT11에서 측정된 온,습도와 체감온도를 확인 가능하다.

9. 장치의 상태 확인

장치가 켜져 있는지 꺼져 있는지 확인이 가능하다.

장치의 설정과 시간을 확인 가능하다.

• 회로 구성

1. CDS

A0

2. BUZZER

D6

3. DHT11

D5

4. 7-SEGMENT

CLK : D3

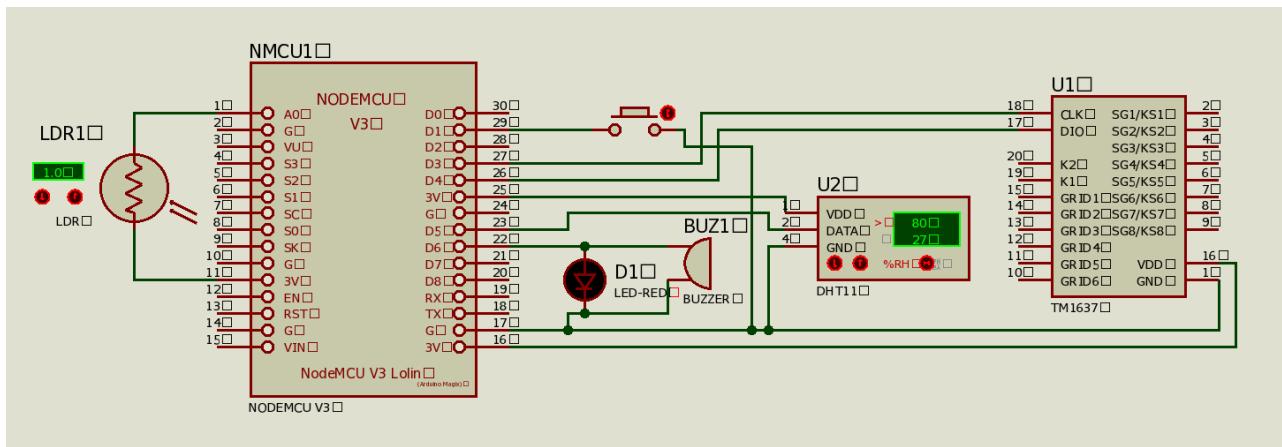
DIO : D4

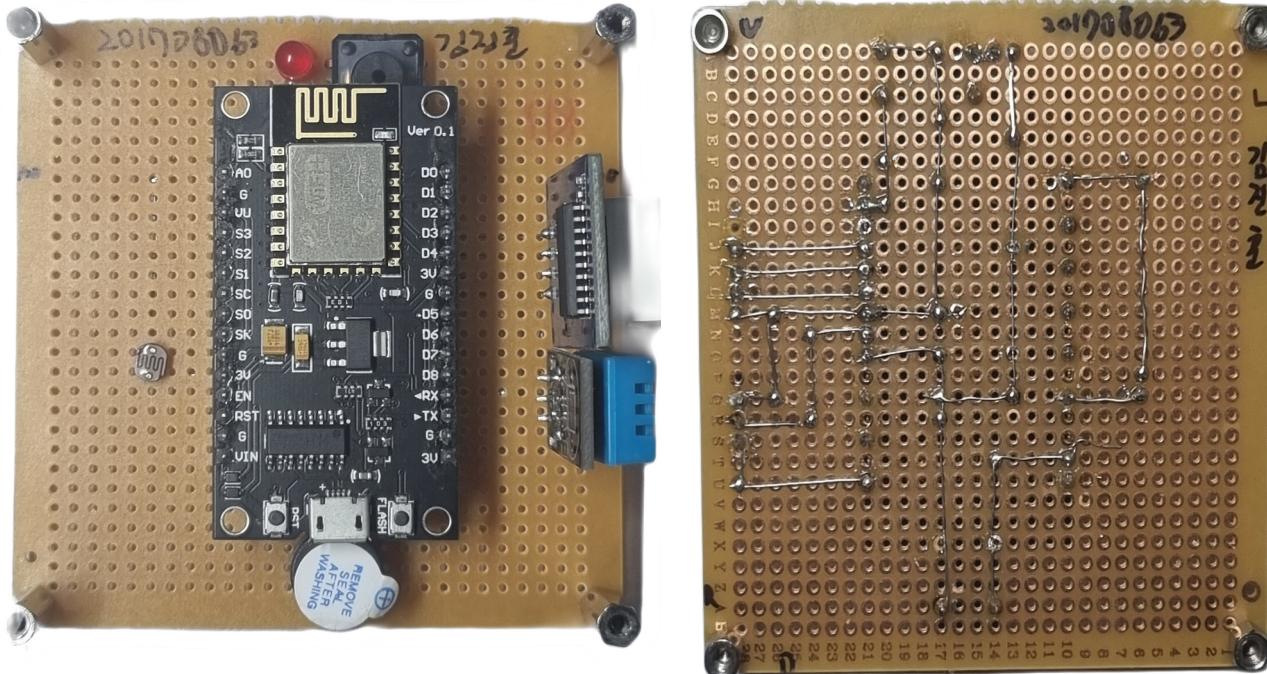
5. LED

D6

6. BUTTON

D1(내부 풀업)





• 시스템 구성

1. 측정 데이터

1. DHT11으로 온습도 측정을 하여 체감온도로 계산 후 온습도 값과 함께 NODE-RED로 전송합니다.
2. CDS를 이용 주변의 밝기를 측정합니다. 초기 계획에서는 풀업 저항을 이용하여 해상도를 조절할 계획이었으나 사용할 저항값을 선택하기 위하여 브레드보드에서 구현 할 때 저항 없이도 실내기준 한낮의 밝은 환경과 밤의 어두운 환경을 구분 할 수 있어 사용하지 않았습니다.

낮, 스마트폰에서 측정한 50~60lux의 환경에서는 analogRead(A0) 기준 1024, 최대 밝기로 출력 되었습니다.

어두운 방안, 스마트폰에서 측정한 10~12lux의 환경에서는 약 600 정도의 값으로 4~5단계의 밝기가 출력 되었습니다.

밤, 스마트폰에서 측정한 1~2lux의 환경에서는 121또는 그 이하의 값이 측정이 되어 최저 밝기가 세그먼트에 출력 되었습니다.

2. LED동작 점등 기능

1. ActiveBuzzer와 같이 연결하여 동시에 작동하게 하였습니다. 이를 통해 알람이 울릴때 소리와 동시에 깜빡이며 장치가 켜질때 짧은 버저소리와 함께 깜빡입니다.

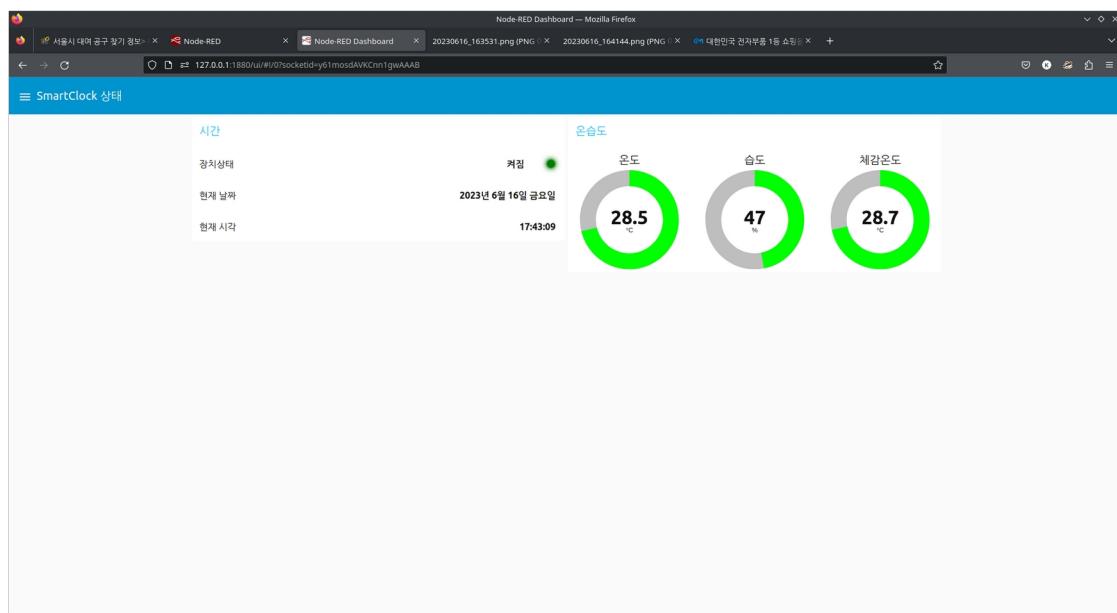
3. 사용한 토픽

1. SmartClock/mode24
2. SmartClock/DisplayMode
3. SmartClock/Brightness
4. SmartClock/Brightness/Auto
5. SmartClock/TimeZone
6. SmartClock/TimeOffset

AM,PM모드 혹은 24시간 중 선택
날짜,시간 중 출력할 정보 선택
밝기 수동 설정
밝기 자동 설정 여부 선택
TimeZone 설정
분 단위 오프셋설정

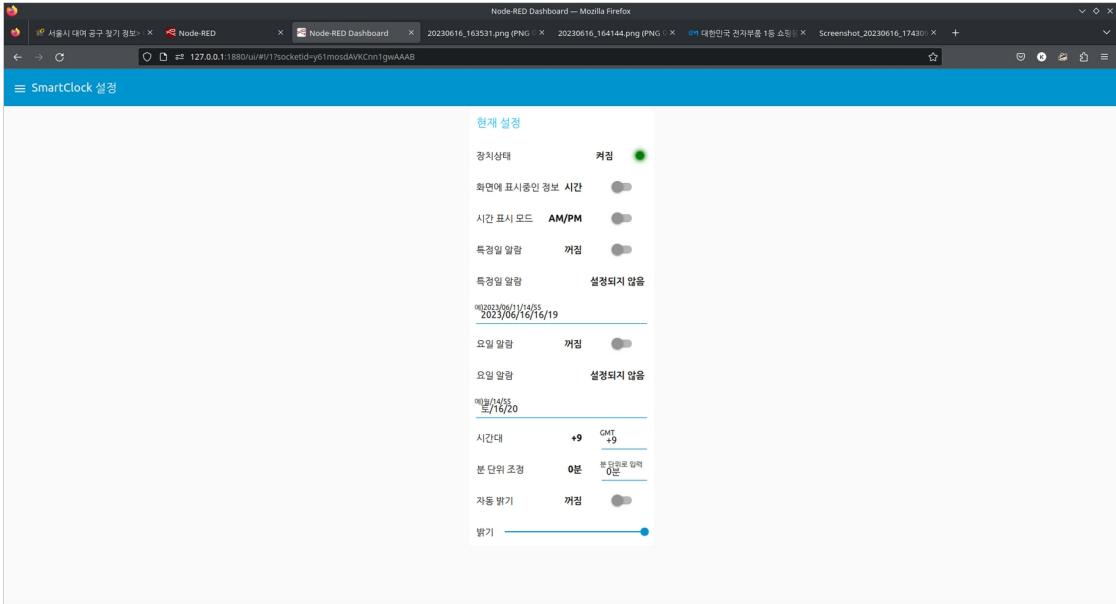
7. SmartClock/Alarm/Date	특정일 알람 on off
8. SmartClock/Alarm/Date/time	특정일 알람 시간 설정
9. SmartClock/Alarm/Week	특정 요일 알람 on off
10. SmartClock/Alarm/Week/time	특정 요일 알람 시간 설정
11. SmartClock/Hum	습도 정보
12. SmartClock/Temp	온도 정보
13. SmartClock/HeatIndex	체감온도 정보
14. SmartClock/Status	장치의 현재 설정 전송 요청
15. SmartClock/Status/DisplayMode	이하 장치 현재 설정 전송
16. SmartClock/Status/mode24	
17. SmartClock/Status/alarmdate	
18. SmartClock/Status/alarmdate/Status	
19. SmartClock/Status/alarmWeek	
20. SmartClock/Status/alarmWeek/Status	
21. SmartClock/Status/TimeZone	
22. SmartClock/Status/TimeOffset	
23. SmartClock/Status/autoBrightness	
24. SmartClock/Status/brightness	
25. SmartClock/Status>Date	
26. SmartClock/Status/Time	

4. DashBoard



SmartClock의 상태

장치 전원, 날짜, 시간, 온습도와 체감온도를 확인 가능함.



SmartClock의 설정

장치의 전원 상태와 각종 설정이 가능하며 모든 설정은 NodeMcu에서 실제로 바뀌지 않는다면 Node-Red에서도 변화하지 않음
 주기적으로 설정을 수신하여 DashBoard와 일치하는지 확인하며, 다르다면 NodeMcu에서 받아온 값으로 스위치나 텍스트 박스, 슬라이더가 바뀐 화면에 진입할 때마다 설정을 다시 수신함.

- 소스 코드

1. 사용한 헤더

```
ESP8266WiFi.h
PubSubClient.h
WiFiUdp.h
NTPClient.h
TM1637Display.h
string
DHTesp.h
```

2. 전역 변수

```
const char* ssid = "ce404";
//WiFi SSID
const char* password = "ce404lab";
//WiFi PW
const char* mqtt_server = "kjh0819.duckdns.org";
//Mosquitto Broker, 초당 천개 까지 테스트함
bool DisplayMode = 0;
//0:시간 1:날짜
bool mode24 = 0;
//0:24시간 1:am,pm
int alarmyear, alarmmonth, alarrray, alarmday, alarmhour, alarmminute = 0;
//특정일 알람 날짜,시간
int alarmWeekDay = 7;
//알람을 울릴 요일(0~6), 7은 존재하지 않는 요일, 설정안됨 상태
int alarmWeekHour, alarmWeekMinute = 0;
```

```

        //특정 요일 알람 시간
bool date_alarm_status = 0;
        //특정일 알람 켜짐 꺼짐
bool Week_alarm_status = 0;
        //특정 요일 알람 켜짐 꺼짐
bool Stop_Alarm_status = 0;
        //알람 중지 여부, false:중지 true:켜짐
int TimeZone = 9;
        //시간대 기본 9
int TimeOffset = 0;
        //시간 오프셋, 분단위
bool autoBrightness = 0;
        //자동 밝기 상태
int brightness = 7;
        //밝기, 기본 7
int Year, Month, day = 0;
        //날짜저장
uint8_t DisplayData [] = { 0xff, 0xff, 0xff, 0xff };
        //세그먼트에 전송할 데이터
int lastDate = 0;
        //마지막 출력 날짜, 바뀔때만 동작하기 위함
int lasthour = 25;
        // 00시 00분에 장치 시작시 미 동작 방지를 위해 기본값 25시로 설정
int lastmin = 25;
        // 시간에 변화 있을때만 동작하기 위해 마지막의 시간 기억
unsigned long long lastbuttonTime = 0;
        //마지막에 버튼을 누른 시간
unsigned long long lastTime1sec, lastTime2sec = 0;
        //각 초마다 작동할 코드들의 마지막 동작시간 기억

```

3. 작성한 함수

```

void sendSetting()
        //현재 설정값 전송
void gotlight()
        //밝기 측정 및 자동 밝기 설정
void SetDate()
        //현재 날짜 설정
void alarm()
        //알람 출력
void SetAlarmWeek(String Data)
        //요일 반복 알람 설정
void SetAlarmDate(String Data)
        //특정일 알람 설정
void DisplayDate()
        //날짜 출력
void DisplayTime()
        //시간 출력
ICACHE_RAM_ATTR void button()
        //버튼 동작, 인터럽트를 위해 램에 올려둠
void setup_display()
        //디스플레이 초기 설정

```

```

void setup_wifi()
    //와이파이 초기 설정
void callback(char* topic, byte* payload, unsigned int length)
    //MQTT데이터 수신,처리
void reconnect()
    //MQTT설정 및 연결 끊길시 재연결
void HumTemCheck()
    //온습도 측정, 전송

```

- 장점
 1. 시중의 제품과는 달리, 주기적으로 정확한 시간을 인터넷으로부터 받아 오차가 적다.
 2. 자동 밝기를 지원하여 낮에는 선명히, 밤에는 눈부심 없이 시간을 확인 가능하다.
- 단점
 1. SEGMENT의 문제로 4,5단계의 밝기 변화는 크지만 다른 밝기 변화는 미미하다.
 2. 부품 가격 총합 9000원 정도의 가격으로 인터넷에 연결되지 않고 알람이 없으며 백라이트가 존재하지 않는 시계의 소비자가 2000원보다 상당히 비싸다.
- 개선 할 점
 1. 브레드보드에서 테스트 후 만능 기판에 회로를 구성하였으나 부품 간의 간격을 고려하지 못한 회로 구성을 하였다. 이후에는 회로도를 작성 후 구현하여야 한다.
 2. 프로그램을 작성 전 구현할 기능은 확정하였으나 FlowChart등 상세한 계획을 세우지 않고 코딩을 하여 각종 변수와 사용하는 토픽이 규칙 없이 작성되었다. 이후에는 변수, 토픽 등을 작성 규칙을 정하여 준수하고 자세한 계획을 세운 후 코딩을 하여야 한다.
 3. 장치에 시간, 날짜 이외에도 현재 온도와 습도를 표시하는 기능을 추가하면 좋을 듯 하다.
 4. 디스플레이를 4칸 7SEGMENT가 아닌 더 많은 세그먼트 혹은 CLCD로 변경하여 웹을 통해 온습도, 수신한 날씨 등 더 많은 정보를 표시하면 좋을 듯 하다.
 5. 충전지를 포함하여 케이블 연결 없이 구동 가능하면 좋을 듯 하다.