

學士學位論文

Raspberry Pi와 NodeMCU를 이용한 와이파이 스마트 멀티플러그 설계 및 구현

Implementation of WiFi Smart Multi Plug with Raspberry Pi and NodeMCU

김도희

安養大學校
情報通信工學科

2018年

Raspberry Pi와 NodeMCU를 이용한 와이파이 스마트 멀티플러그 설계 및 구현

Implementation of WiFi Smart Multi Plug with Raspberry Pi and NodeMCU

김도희

위 논문은 안양대학교 학생學位 논문으로
學位 논문 審査委員會에서 審査 通過하였음.

2018年 11月 28日

審査 委員長 : 김영기 (印)
審査 委員 : 홍진표 (印)
審査 委員 : 최선완 (印)

Raspberry Pi와 NodeMCU를 이용한 와이파이 스마트 멀티플러그 설계 및 구현

Implementation of WiFi Smart Multi Plug with Raspberry Pi and NodeMCU

김도희

위 논문은 안양대학교 학생學位 논문으로
學位 논문 審査委員會에서 審査 通過하였음.

2018年 11月 28日

指導教授： 최선완 (印)

국 문 요 약

각 가정은 물론 기업, 학교 할 것 없이 멀티탭을 사용하지 않는 곳은 거의 없다. 멀티탭은 하나의 콘센트에서 동시에 수많은 전기기구를 사용할 수 있는 구조의 접속기이다. 그런 만큼 멀티탭의 허용전력을 초과하여 사용하는 경우도 많으며, 그로 인한 화재발생도 매해 꾸준히 발생하고 있다. 따라서 보급률이 높은 스마트폰과 PC로 언제든지 접속할 수 있도록 홈페이지를 통해 원격으로 멀티탭의 현재상황을 확인하고 제어할 수 있는 멀티탭의 구현은 생활에 유익할 것이다.

본 논문에서는 WiFi 기반의 MQTT 통신 프로토콜을 이용한 와이파이 스마트 멀티플러그를 구현한다. 멀티탭의 기본 기능은 전체 전원 제어, 구별 전원 제어 기능 외에도 사용자가 원하는 시간에 언제든지 전원을 on, off 할 수 있는 예약 타이머 기능, 일정온도 이상이 될 경우 자동으로 전원을 차단하는 기능, 현재 집안 및 멀티탭의 실시간 온도 측정 기능을 추가한다.

목 차

제 1 장 서론	5
제 2 장 관련 연구	
제 1 절 관련 연구 및 분석	7
제 3 장 MQTT(Message Queuing Telemetry Transport)	
제 1 절 MQTT	10
제 2 절 MQTT 작동 원리	10
제 4 장 설계	
제 1 절 설계 환경	12
제 2 절 설계 모델	14
제 3 절 설계 고려사항	16
제 5 장 구현	
제 1 절 구현 환경	17
제 2 절 구현 구조	18
제 3 절 구현 화면	20
제 6 장 결론 및 향후 계획	27
참고문헌	28
부록1. API(Application Programming Interface)	30

부록 2. Raspberry Pi 원격접속 설정	----- 33
부록 3. Arduino Sketch 설정	----- 40

그림 목차

<그림 1-1> 세계 전기 보급률	-----5
<그림 1-2> 우리나라 전기 보급률	-----5-
<그림 1-3> 우리나라 전체 화재 원인	-----5-
<그림 1-4> 발화요인이 전기적 요인인 화재의 전기적 원인	-----6--
<그림 3-1> MQTT 프로토콜 작동원리	-----0 1
<그림 4-1> NodeMCU	----- 2
<그림 4-2> 스마플러그 설계 모델	-----4 1
<그림 4-3> 자체 구현 멀티탭 설계도	-----5 1
<그림 5-1> 스마플러그의 Web Page 모듈	-----81
<그림 5-2> 스마플러그의 Web Page 구조	-----02
<그림 5-3> Android상에서의 화면	-----22
<그림 5-4> iOS상에서의 화면	-----22
<그림 5-5> PC상에서의 화면	-----22
<그림 5-6> 전체 구 화면	-----32
<그림 5-7> 스위치를 on 했을 때의 화면	-----3 2
<그림 5-8> 스위치를 off 했을 때의 화면	-----3 2
<그림 5-9> 현재상황 및 구별 제어 화면	-----4 2
<그림 5-10> 예약제어 화면	-----52
<그림 5-11> 입력 필수 경고 화면	-----6 2
<그림 5-12> 예약 성공 알림창	-----6 2
<그림 5-13> 그래프로 표현한 실시간 온도	-----7 2
<그림 5-14> 게이지로 표현한 실시간 온도	-----7 2
<그림 5-15> 온도 구간별 게이지 색깔 ②	-----7 2
<그림 5-16> 온도 구간별 게이지 색깔 ③	-----7 2
<그림 5-17> 그래프의 상세한 온도 수치 항목	-----8 2

표 목차

<표 2-1> 스마트 멀티플러그 기능 분석	-----7-
<표 4-1> ESP8266 사양	----- 31
<표 5-1> 스마트 멀티플러그 개발환경	-----7 1
<표 5-2> 자체 구현 멀티탭 부품 목록	-----7 1

제 1 장 서론

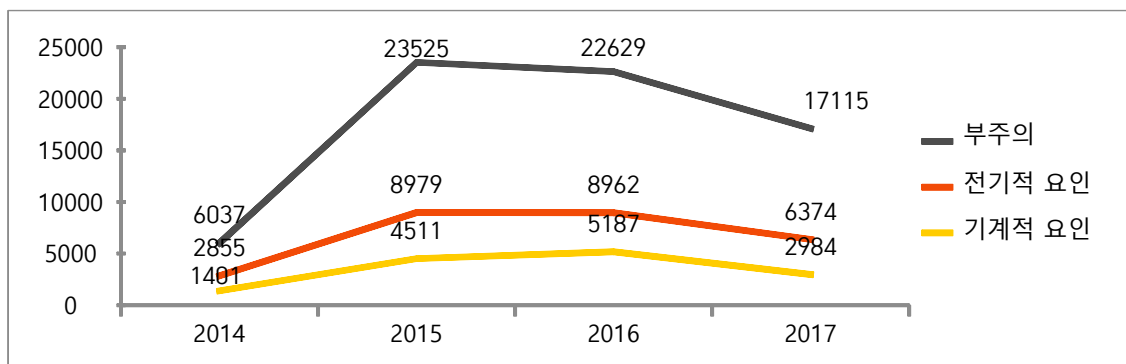
세계 어디서든 전기의 보급은 활발히 이루어지고 있음에 따라 하나의 콘센트에 여러 전자기기를 이용할 수 있는 멀티탭 또한 일상에서 가장 손쉽게 볼 수 있는 제품 중 하나 일 것이다. <그림 1-1>은 세계 전기 보급률을 나타낸다. 일부 아프리카 지역을 제외하고는 세계 어디에서든 전기를 사용하고 있다. 그 중에서도 우리나라는 대중적으로 전기를 보급하고 있다.



<그림 1-1> 세계 전기 보급률 [1]



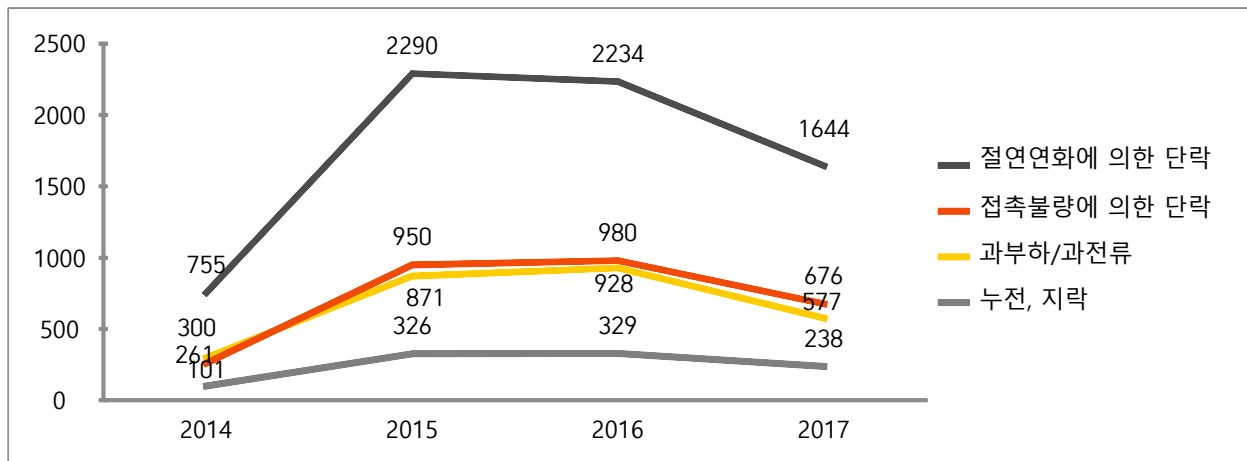
<그림 1-2> 우리나라 전기 보급률 [1]



<그림 1-3> 우리나라 전체 화재 원인 [2]

<그림 1-3>은 우리나라의 화재 발생원인 중 전기적 요인이 부주의로 인한 화재발생을 뒤이은 가장 큰 원인이라는 것을 볼 수 있다. <그림 1-4>에서는 그런 전기적 요인 중에서도 차례로 절연연화에 의한 단락,

접촉불량에 의한 단락, 과부하/과전력 등이 화재의 원인이 되고 있음을 나타낸다. 최근에는 사물인터넷, IoT(Internet of Things)를 이용한 스마트 멀티플러그의 제품의 구현이 활발히 이루어지고 있다. 그 중에서 가장 흔하게 접할 수 있는 스마트 멀티플러그에는 ‘샤오미(Xiaomi)’ 사의 제품, ‘스토리링크’ 사의 제품, ‘와트드림’ 사의 제품과 설계 및 구현예정인 스마트 멀티플러그에 대해서는 제 2 장에서 자세히 기술한다.



<그림 1-4> 발화요인이 전기적 요인인 화재의 전기적 원인 [2]

따라서 본 논문에서는 ‘제 2 장’에서 다뤄질 스마트 멀티플러그의 관련연구를 기반으로 사물인터넷, IoT(Internet of Things)를 이용하여 화재방지 및 안전에 중점을 두어 웹페이지로 원격으로 접속하여 제어할 수 있는 과열, 과전력으로 인한 화재방지를 기본으로 한 스마트 멀티플러그 설계 및 구현에 대해 기술한다. 이에 대한 설계 고려사항은 ‘제 4 장’에서 상세히 기술한다. 이어질 ‘제 3 장’에서는 본 논문에서 설계 및 구현에 대해 서술하고 있는 스마트 멀티플러그에서 사용하고 있는 주된 프로토콜에 대해 상세히 기술한다.

제 2 장 관련연구

제 1 절 관련연구 및 분석

<표 2-1>은 기존의 스마트 멀티플러그와 우리 모델(스마플러그)의 기능별 비교표이다.

<표 2-1> 스마트 멀티플러그 기능 분석

기능 \ 이름	샤오미(Xiaomi) [3]	스토리링크 [4]	와트드림 [5]	스마플러그
전체 구의 구성	3구, 6구	1구	1구	4구
구별 제어	X	X	X	O
전체 제어	O	O	O	O
과열제어	X	X	X	O
실시간 온도 측정	X	X	X	O
실시간 전력 측정	O	X	X	X
예약 타이머	O	O	O	O
사용 가능한 기기	Android, iOS (애플리케이션)	Android, iOS (애플리케이션)	Android, iOS (애플리케이션)	PC, Android, iOS (웹)
과전압 차단	X	X	X	O
사용하는 통신 방식	WiFi	WiFi	Bluetooth	WiFi

1. 샤오미(Xiaomi) [3]

1.1. 전체 구의 구성

3구와 6구를 선택할 수 있으며 110V전원을 사용한다.

1.2. 전체 제어

3구든 6구든 구의 개수와는 상관없이 한 번에 모든 구를 on, off 할 수 있다.

1.3. 실시간 전력 측정

현재 멀티탭에서 사용 중인 실시간 전력을 측정하고 누계 시킨 후, 중국식으로 계산된 예상 전기 요

금을 산출한다.

1.4. 예약 타이머

원하는 시간대 별로 전원을 on, off 할 수 있다.

1.5. 사용 가능 기기

Android, iOS로 전용 애플리케이션을 통해 사용할 수 있다.

2. 스토리링크(STORY LiNK) [4]

2.1. 전체 구의 구성

1구로 이루어져 있다.

2.2. 전체 제어

구의 개수와는 상관없이 한 번에 모든 구를 on, off 할 수 있다.

2.3. 예약타이머

원하는 시간대 별로 전원을 on, off 할 수 있다.

2.4. 사용 가능 기기

Android, iOS로 전용 애플리케이션을 통해 사용할 수 있다.

3. 와트드림(Watt Dream) [5]

3.1. 전체 구의 구성

1구로 이루어져 있다.

3.2. 전체 제어

구의 개수와는 상관없이 한 번에 모든 구를 on, off 할 수 있다.

3.3. 예약타이머

원하는 시간대 별로 전원을 on, off 할 수 있다.

3.4. 사용 가능 기기

Android, iOS로 전용 애플리케이션을 통해 사용할 수 있다.

4. 스마플러그

4.1. 전체 구의 구성

총 4구로 구성되어 있다.

4.2. 전체 제어

전체 전원을 한번에 on, off 할 수 있다.

4.3. 개별 제어

4개의 구를 각각 하나씩 제어할 수 있다.

4.4. 과열제어

일정온도 이상으로 온도센서를 통해 온도가 측정되었을 경우, 자동으로 모든 전원을 차단한다. 전원이 차단된 이후로도 일정온도 이하로 온도가 내려가지 않으면, 전원을 켤 수 없다.

4.5. 실시간 온도 측정

실시간으로 멀티탭 내부 온도를 측정하여, 그래프와 게이지로 표현한다.

4.6. 과전압 차단

과전압 및 누전 차단기를 내부에 배치하여 일정 전력이상 사용될 경우, 자동으로 모든 전원이 차단된다.

4.7. 사용 가능 기기

웹페이지를 통해 접근이 가능하므로 스마트폰, PC 등 기기에 관계없이 url을 통해 접근이 가능한 기기라면 충분히 사용할 수 있다.

본 논문에서 구현하는 스마트 멀티플러그는 WiFi 통신을 기반으로 하여, 총 4구의 전체 제어 및 개별 제어가 가능하며, 실시간 온도 측정으로 과열제어와 과전압 차단을 통해 화재방지를 미연에 방지하고자 하며, 웹페이지로 접근이 가능해 기존에 있는 모델들과 달리 접근성을 더 용이하게 하는 것에 초점을 맞추고자 한다.

제 3 장 MQTT

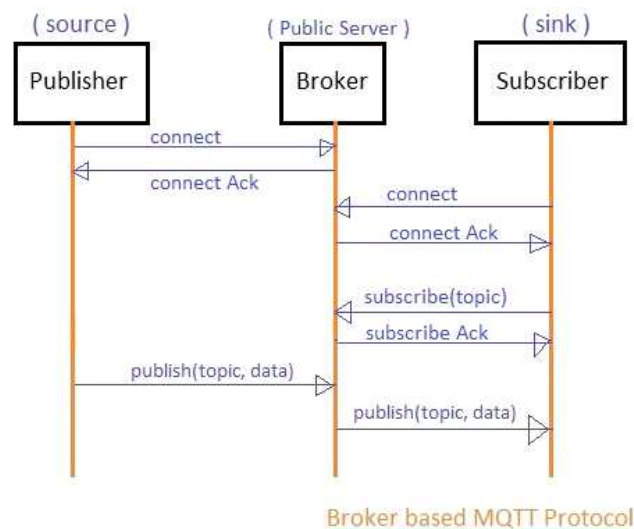
(Message Queuing Telemetry Transport) [6]

제 1 절 MQTT

제 3 장에서는 스마트 멀티플러그에서 사용될 주된 통신 프로토콜인 MQTT 프로토콜에 대해 기술한다. MQTT 프로토콜은 Message Queuing Telemetry Transport의 약어로서, 사물 통신(M2M: Machine to Machine), 사물 인터넷(IoT: Internet of Things)과 같이 대역폭이 제한된 통신 환경에 최적화하여 개발된 푸시 기술(push technology) 기반의 경량 메시지 전송 프로토콜이다[6] .

모바일 기기나 낮은 대역폭의 소형 디바이스들에 최적화되었다. Publish/Subscribe 형태를 취하여 세 가지의 QoS(Quality of Service)레벨을 제공한다. IBM이 주도하여 개발하였고 OASIS란 민간 표준화 기구에서 표준화가 되었으며 ISO 표준으로 ISO/IEC PRF 20922로 등록 되어있는 프로토콜이다. 또한 MQTT 프로토콜은 TCP/UDP 환경 모두에서 사용할 수 있으며, 포트번호로는 1883을 대표적으로 사용한다[7][8] . MQTT 프로토콜의 작동원리를 간단하게 그림으로 표현하면 아래의 <그림 3-1>와 같다.

제 2 절 MQTT 작동 원리



<그림 3-1> MQTT 프로토콜 작동원리 [9]

MQTT 프로토콜은 <그림 3-1>에서 볼 수 있다시피 푸시 기술(push technology)에서 일반적으로 사용되는 클라이언트/서버 방식 대신, 메시지 매개자(broker)를 통해 송신자가 특정 메시지를 발행(publish)하고 수신자가 메시지를 구독(subscribe)하는 방식을 사용한다. 이때 발행(publish)은 클라이언트의 callback() 함수 안에서 작동한다. 즉, 매개자(broker)를 통해 서버와 클라이언트가 자유롭게 각자의 topic에 맞게 원하는 메시지가 송수신 된다. 메시지 길이가 가장 작게는 2 바이트까지 가능하고, 초당 1,000 단위의 메시지 전송이 될 수 있어 가볍고 빠르다는 장점을 갖는다. 따라서 원격 검침, 원격 의료 등 다양한 분야에 효율적으로 사용될 수 있다[6] .

제 4 장 설계

제 1 절 설계환경

본 논문에서는 웹페이지와 Raspberry를 통해 클라이언트(NodeMCU)로부터 다양한 종류의 메시지를 발행(publish), 구독(subscribe) 하는 방법으로 스마트 멀티플러그를 구현한다. MQTT 프로토콜을 이용한 실제 설계와 구현은 제 4 장과 제 5 장에서 기술한다.

그에 앞서 WiFi 기반의 MQTT 프로토콜 방식을 이용한 스마트 멀티플러그에서 멀티탭과 직접 접속하여 Raspberry Pi 서버와 MQTT 통신을 하는 보드인 NodeMCU에 대해 기술한다.



<그림 4-1> NodeMCU [10]

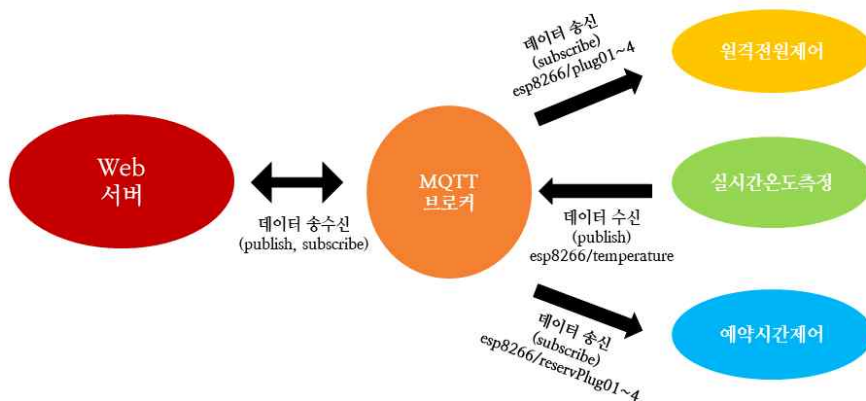
<표 4-1> ESP8266 사양 [11]

Module	Model	ESP8266-12
	IC	ESP8266
Wireless Parameter	Wireless standard	IEEE 802.11b/g/n
	frequency range	2.412GHz-2.484GHz
	Transmitted power	802.11b: +16 +/-2dBm (@11Mbps)
		802.11g: +14 +/-2dBm (@54Mbps)
		802.11n: +13 +/-2dBm (@HT20, MCS7)
	Receive sensitivity	802.11b: -93 dBm (@11Mbps ,CCK)
		802.11g: -85dBm (@54Mbps, OFDM)
		802.11n: -82dBm (@HT20, MCS7)
Hardware Parameter	Wireless form	Stamp hole
		I-PEX connector , SMA connector
		Onboard PCB antenna
	Hardware connector	UART, IIC, PWM, GPIO, ADC
	Working voltage	3.3V
	GPIO driver capability	Max: 15ma
	Working current	Continue to send=> AVRG: ~70mA,MAX: 200mA
		Normal mode=> AVRG: ~12mA,MAX: 200mA standby: <200uA,
Serial transmission	Working temperature	-40℃~125℃
	Ambient temperature	temperature: <40℃, RH: <90%R.H.
	Size	24.0mm*16.0mm*1mm
Soft parameter	Transmission rate	110-921600bps
	TCP Client	5
Soft parameter	Wireless network type	STA/AP/STA+AP
	Security mechanism	WEP/WPA-PSK/WPA2-PSK
	Encryption Type	WEP64/WEP128/TKIP/AES
	Firmware Upgrade	Local serial port, OTA Remote upgrade
	Network protocol	IPv4, TCP/UDP/FTP/HTTP
	User Configuration	AT+order set, Web Android/iOS , Smart Link APP

NodeMCU는 ESP8266이라는 칩을 장착해서 WiFi기반의 통신환경을 사용할 수 있게 해주며 아두이노 스케치를 이용해 프로그램을 업로딩 할 수 있는 보드이다. 네트워크 프로토콜로는 TCP, UDP, FTP, HTTP를 사용한다. 본 논문에서는 NodeMCU를 자체 구현 멀티탭에 연결해서 Raspberry Pi 서버와 원격제어, 예약제어, 실시간 온도 측정 등의 각 topic과 함께 전송된 메시지를 서로 주고받게 된다.

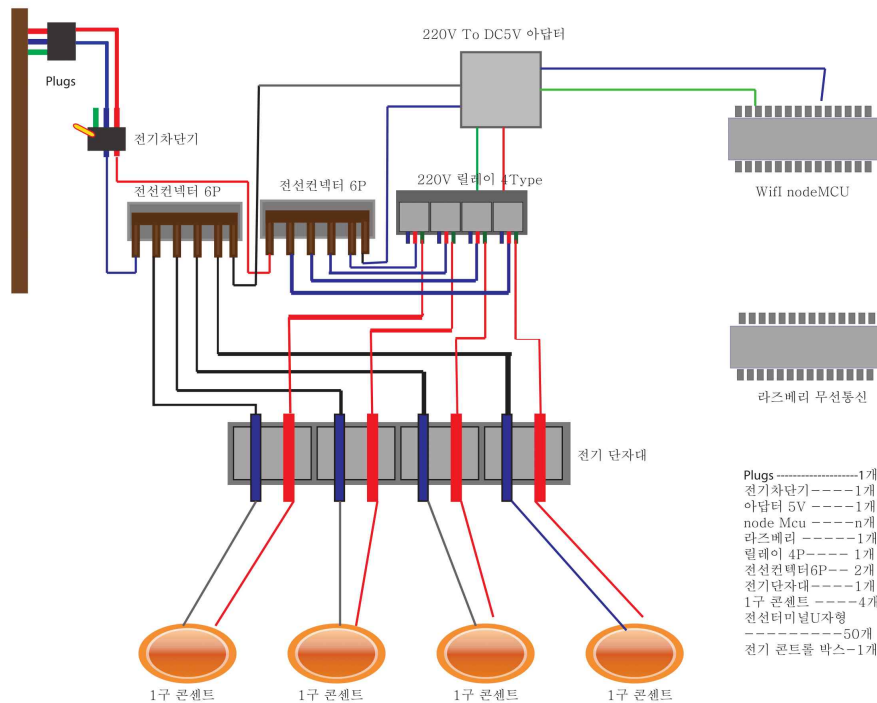
제 2 절 설계모델

스마플러그는 <그림 4-2>의 모델처럼 실행되며 자체 구현 멀티탭 설계도는 <그림 4-3>와 같다.



<그림 4-2> 스마플러그 설계 모델

<그림 4-2>는 스마플러그의 설계 모델을 간단히 표현한 그림이다. 스마플러그는 웹 서버와 NodeMCU로 구성된 클라이언트 사이에 MQTT 브로커라는 중개자를 사이에 둔다. 스마플러그의 원격전원제어, 실시간 온도 측정, 예약시간제어 등의 기능에 알맞은 센서를 이용해 센싱한 데이터를 MQTT 브로커의 중개자를 통해 웹 서버와 송수신한다. 주고받는 데이터의 특성에 알맞게 클라이언트 혹은 웹 서버는 데이터를 송신할 것인지, 수신할 것인지 결정한다. 그렇게 주고받은 데이터는 웹 서버 혹은 클라이언트에게 전송되어 데이터에 해당되는 기능을 처리하는 방식의 모델이다.



<그림 4-3> 자체 구현 멀티탭 설계도

스마플러그의 자체 구현 멀티탭 설계도는 <그림 4-3>과 같다. 그리고 클라이언트로서 직접 웹 서버로부터 다양한 데이터를 넘겨받아 직접 스마플러그를 제어하는 NodeMCU는 신호를 받아 전원을 직접 토글 시켜주는 4채널 릴레이와 연결한다. NodeMCU는 220V to 5V 어댑터로부터 5V 전압을 인가받아 전원을 켜다. 여기서 NodeMCU는 데이터를 업로딩할 때 사용하는 마이크로 5핀 케이블을 통해 전압을 인가받는다. 또한 NodeMCU와 직접 연결하는 4채널 릴레이는 NodeMCU로부터 5V 전력을 인가받고, 최대 250V까지 전원을 제어할 수 있게 해준다. 마지막으로 NodeMCU에 너무 많은 모듈이 배치될 경우 제대로 작동이 되지 않을 가능성이 있으므로 그에 대한 안정성을 위해 NodeMCU에 꼭 연결해야하는 RTC(Real Time Clock) 모듈을 제외하고는 센서를 부착하지 않는다. 또한 각각 1구 콘센트 끼리 연결되어 있는 전선들은 전기적으로 차단기와 연결해 NodeMCU에 안전한 전력공급을 위해 따로 차단기를 배치해 일정 전력이상 흐르게 되면 자동으로 전원이 내려가게 된다.

제 3 절 설계 고려사항

본 논문에서는 NodeMCU를 이용해 실제로 Raspberry Pi와 WiFi 기반의 MQTT 통신을 구현한다. 또한 WiFi 기반의 MQTT 프로토콜 방식을 이용한 스마트 멀티플러그의 구현을 위한 설계 고려사항은 다음과 같다. 자체 구현 멀티탭을 위한 자세한 부품목록은 제 5 장 구현 부분에 기술한다.

1. 기존에 나와 있던 일반 멀티탭을 개조하는 것이 아니라 실제로 멀티탭부터 구현한다.
2. 애플리케이션으로 구현할 경우 Android와 iOS 두 버전으로 모두 구현해야 하는 어려움이 있으므로 스마트폰과 PC로도 접근이 가능한 웹페이지로 스마트 멀티플러그를 제어할 수 있게 한다.
3. 사용자 위주의 GUI를 기본으로 한다.
4. 스마트 멀티플러그를 설치한 곳 외에서도 접근할 수 있도록 웹페이지는 반드시 외부에서 접속할 수 있도록 DDNS 서버를 기반으로 한다.
5. 멀티탭의 기본 규격은 220V로 한다.
6. NodeMCU에 기본적으로 공급되는 전압은 5V로 한다.
7. NodeMCU의 모델은 현재 가장 안정적인 NodeMCU Lua 0.9 버전을 사용한다.
8. NodeMCU는 마이크로 5핀 케이블을 통해 어댑터를 통해 변환된 5V를 인가받는다.

제 5 장 구현

제 1 절. 구현 환경

스마플러그가 구현된 환경은 다음 아래의 <표 5-1>, <표 5-2>와 같다.

<표 5-1> 스마플러그 개발환경

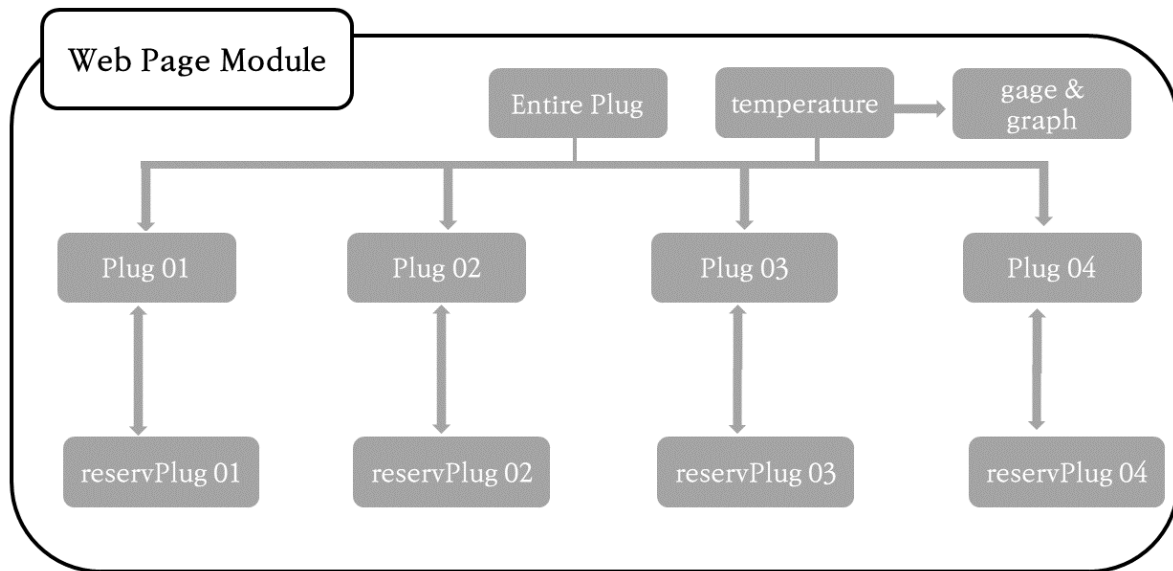
구분	개발환경
Equipment	Raspberry B 3.0, Intel(R) Core(TM) i7-4500U CPU@ 1.80GHz 2.40GHz
OS	Raspbian, Window 10
Platform	Web Browser
Language	HTML, C, JSON, Node.js
Tools	Android, iOS, PC
Location	안양대학교 수리관

<표 5-2> 자체 구현 멀티탭 부품목록

부품	수량
PVC박스(하이박스) 200*300*130	1
20A 배선 혹은 누전차단기	1
4P단자대	4
전원 플러그	1
2.5 Y자 터미널	50
1구 노출 콘센트	4
2.5 HIV 전선	2m
PVC 연선 전선	4m
PCB 기판 17*17 싱글	2
220V to DC 5V 2A 어댑터	1
핀헤더 소켓 2줄용(NodeMCU를 끼울 수 있는 것)	2
NodeMCU Lua 0.9 (ESP-12 Module)	1
RTC 모듈(DS3231)	1
아두이노 4채널 릴레이 모듈	1
라즈베리파이 B 3.0	1

제 2 절 구현 구조

스마플러그가 구현된 웹페이지 모듈은 다음 아래의 <그림 5-1>과 같다.



<그림 5-1> 스마플러그의 Web Page 모듈

따라서 위 Web Page 모듈은 다음과 같은 기능을 수행한다.

esp8266/plug01 ~ 04 : 4개의 구를 각각 on, off

esp8266/temperature : 멀티탭 내부 온도를 실시간으로 측정하여 전송

esp8266/EntirePlug : 모든 구를 on, off

esp8266/reservPlug01 ~ 04 : 4개의 구를 각각 on, off 할 수 있는 예약 타이머

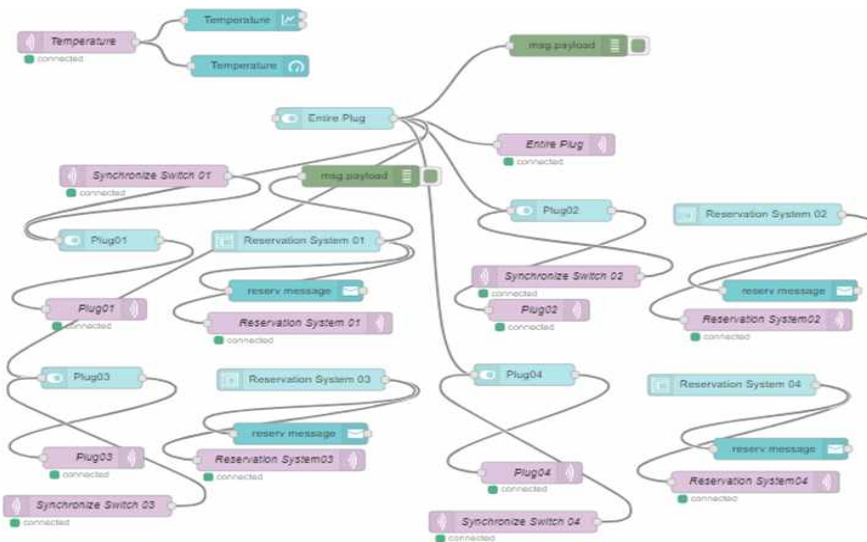
또한 웹페이지에서 각각 원격전원제어, 실시간온도측정, 예약시간제어 등의 항목에 대해서 사용자를 통해 웹페이지 상에서의 버튼을 누르거나 값을 입력하는 등의 이벤트가 발생하면, MQTT 브로커를 통해 이벤트에 맞는 topic과 함께 topic에 맞는 메시지가 NodeMCU에 전달된다. NodeMCU에 업로딩된 프로그램을 통해 NodeMCU는 해당 topic에 맞는 작업을 수행한다.

첫 번째로 원격전원제어는 웹페이지에서 스위치버튼을 이용해 사용자가 각 멀티탭의 구에 해당되는 스위치를 켜거나 끄는 이벤트를 발생시키면 'esp8266/plug01' 부터 'esp8266/plug04'까지의 topic을 'on' 과 'off' 라는 메시지를 Raspberry가 publish하게 되면 NodeMCU가 해당 topic을 subscribe하여 메시지와 함께 전달받는다. NodeMCU는 메시지에 따라 구별로 해당 전원을 켜거나 끄는 동작을 수행한다.

실시간온도측정은 웹페이지에서 게이지와 그래프를 이용해 매초마다 실시간으로 현재 멀티탭 내부의 온도를 표현한다. 'esp8266/temperature' 라는 topic을 이용해 NodeMCU는 RTC모듈인 DS3231 내부의 크리스탈 오실레이터를 통해 측정된 온도 값을 NodeMCU가 Raspberry Pi 서버에게 publish 하면 Raspberry Pi 서버는 해당 값을 subscribe 해서 게이지와 그래프를 통해 표현한다.

예약제어는 웹페이지에서 분단위로 예약이 가능한 버튼과 예약하기를 원하는 상태를 스위치로 표현 하여 해당버튼은 필수 값으로 지정해서 사용자가 값을 입력하지 않으면 예약버튼을 통해 값이 전달되지 않도록 한다. 또한 'esp8266/reservPlug01' 부터 'esp8266/reservPlug04' 까지의 topic을 이용해 모든 구를 사용자가 예약제어 할 수 있도록 한다. 분단위로 입력받은 값은 RTC 모듈인 DS3231을 통해 매순간마다 측정 되는 시간 값을 이용해 예약추정시간을 함수를 통해 계산한다. 그리고 예약추정시간을 DS3231을 통해 매순간마다 측정되는 시간 값과 비교하여 예약추정시간과 값이 동일해지는 순간 원격전원제어의 원리처럼 NodeMCU가 해당 메시지에 따라 동작을 수행한다. 따라서 각 기능에 대한 자세한 구현에 대해서는 아래에 기술한다.

1. 웹페이지에 접속하면 전체 구 제어, 개별 구 제어, 예약 제어, 실시간 온도 총 4개의 항목이 필요에 따라 접거나 펼쳐서 사용할 수 있도록 되어 있다.
2. 전체 구 제어를 누르면 웹페이지에 표시되어있는 개별 구 제어에 있는 스위치가 on으로 바뀐다.
3. 개별 구 제어에 있는 각 스위치들은 현재 멀티탭의 전원 상태와 동기화 되어있다. 예약전원제어 기능을 사용하거나 과열제어로 인해 자동으로 전원이 차단되었을 경우 전원이 차단됨과 동시에 웹페이지에서의 스위치의 상태도 해당상태에 맞게 변한다.
4. 예약 제어는 분단위로 가능하며, 최대 30일 미만까지 가능하다. 예약버튼을 누르면, 예약이 되었다는 알림창이 뜬다. 타이머로 설정한 시간이 지나면, 개별 구 제어에 있는 구별 스위치가 예약상태에 따라 자동으로 변한다.
5. 실시간 온도 측정에서는 서버가 켜진 시간을 기준으로 하여 그래프의 경우에는 X축은 1일, Y축은 0℃ ~ 70℃까지 측정한다. 그래프 하단에 있는 게이지바에는 0℃ ~ 70℃까지의 온도를 0℃ ~ 36.5℃까지는 초록색, 36.5℃ ~ 59℃까지는 노란색, 50℃ ~ 70℃까지는 붉은색으로 표시한다.
6. 만약 실시간 온도 측정의 결과 50℃이상이 된다면, 자동으로 모든 전원이 차단되며 차단된 순간의 온도가 바로 전송된다.



<그림 5-2> 스마플러그의 Web Page 구조

구별 제어 및 현재상황을 구현한 부분은 <그림 5-2>에서 Plug01부터 Plug04까지다. 웹페이지에서는 출력을 담당하는 MQTT 패킷을 이용해 on이나 off라는 메시지를 NodeMCU로부터 출력한다. NodeMCU는 웹페이지로부터 전달받은 메시지를 통해 전원을 끄거나 켜다. 이때 Plug01부터 Plug04까지는 Synchronize Switch01부터 Synchronize Switch04까지 입력을 담당하는 MQTT패킷과 각각 이름에 맞게 연결한다. 이는 예약제어와 과열제어와 연동되어 사용자가 시간에 맞춰 구별로 전원을 끄거나 켜기를 설정한다면 실제 스마플러그에서도 전원이 들어옴과 동시에 웹페이지의 스위치도 동기화 시켜주는 역할을 한다. 이는 사용자가 멀리 떨어져 있을 때도 실시간으로 스마플러그의 전원을 확인할 수 있게 한다.

전체 제어는 Entire Plug와 Plug01부터 Plug04까지 연결된다. 이는 Entire Plug에서 스위치를 움직이는 동시에 해당 상태를 Plug01부터 Plug04까지 전부 전달한다. 그 이후의 과정은 위에서 기술한 부분과 동일하다.

예약제어는 Reservation System 01부터 Reservation System 04까지로 구성되어 있다. 구의 전원을 어떻게 할지 결정하는 스위치와 몇 분 후에 해당 명령을 실행할지 결정하는 입력칸이 있다. 입력칸은 오직 숫자로만 입력할 수 있게 되어있다. 두 항목은 필수항목으로 작성되며, 각각의 형식에 맞지 않는 데이터가 입력되면 오류 메시지를 발생시킨다. 그리고 둘 중 하나라도 입력하지 않았을 경우 사용자에게 어떤 항목이 작성되지 않았는지 알려준다. 사용자가 올바르게 작성한 후 예약버튼을 누르면 해당 메시지가 NodeMCU로 전달되며 RTC모듈을 이용해 예약예상시간을 계산한다. 이를 실시간으로 현재시각을 갱신시켜 사용자가 지정한 예약설정에 의해 도출된 예약예상시간과 같은 지 비교한 후 같으면 예약하길 원하는 전원의 상태를 MQTT패킷을 통해 NodeMCU로 전달한다. 그와 동시에 Synchronize Switch에도 해당 상태를 전달해 실시간으로

웹페이지에서 구별 상황을 확인할 수 있다.

온도는 Temperature로 구성된다. 입력을 담당하는 MQTT 패킷으로부터 NodeMCU가 RTC모듈에 내장된 크리스탈 오실레이터에서 실시간으로 발생하는 데이터를 전달받아 이를 게이지와 그래프라는 웹페이지 상의 항목으로 표현한다. 그리고 50℃ 이상의 온도가 감지되면 Synchronize Switch01부터 Synchronize Switch04까지를 이용해 모든 전원을 차단시키고 실시간으로 온도를 감지하므로 50℃미만으로 온도가 내려가지 않으면 전원을 켤 수 없게 한다.

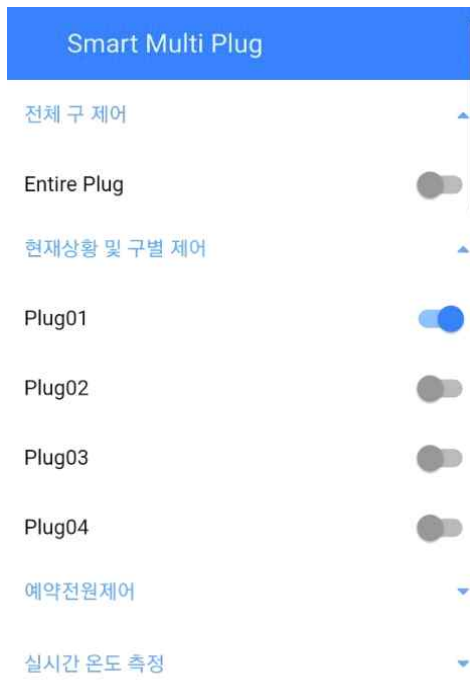
제 3 절. 구현 화면

1. 메인화면

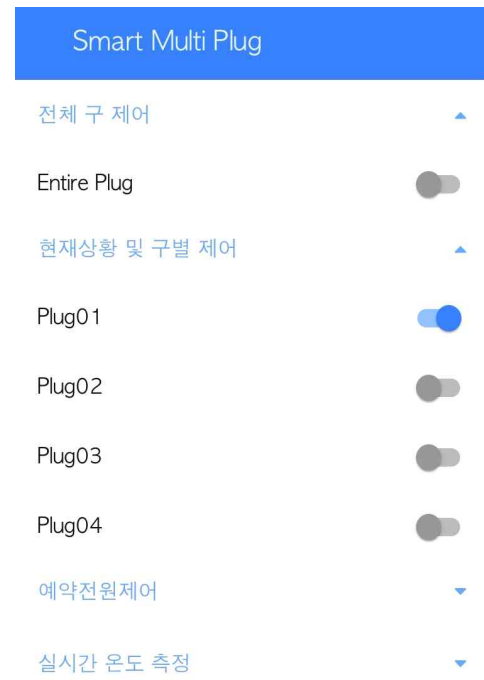
1.1. Android, 갤럭시 S8 기종에서의 화면이다.

1.2. iOS, 아이폰 8 기종에서의 화면이다.

1.3. PC에서의 화면이다.



<그림 5-3> Android상에서의 화면




<그림 5-4> iOS상에서의 화면



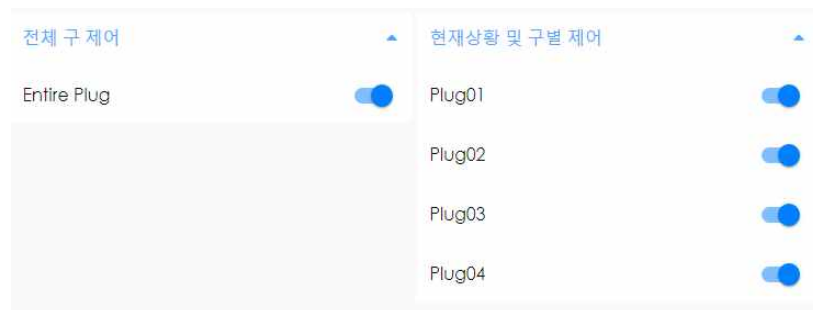
<그림 5-5> PC상에서의 화면

2. 전체제어화면

- 2.1. 전체제어화면은 다음 <그림 5-6>와 같다.
- 2.2.  버튼을 눌러 전체 구의 전원을 on, off 할 수 있다.
- 2.3. 전원을 on 했을 때의 화면은 <그림 5-7>, 전원을 off 했을 때의 화면은 <그림 5-8>과 같다.
- 2.4. ▲와 ▼ 버튼을 눌러 해당항목을 접거나 펼 수 있다.



<그림 5-6> 전체 구 화면




<그림 5-7> 스위치를 on 했을 때의 화면



<그림 5-8> 스위치를 off 했을 때의 화면






3. 현재 상황 및 구별 제어

- 3.1. 현재 상황 및 구별 제어화면은 다음 <그림 5-9>와 같다.
- 3.2.  버튼을 눌러 개별적으로 구의 전원을 on, off 할 수 있다.
- 3.3. 현재 스마트 멀티플러그의 상태와 동기화 된다.
- 3.4. ▲와 ▼ 버튼을 눌러 해당항목을 접거나 펼 수 있다.

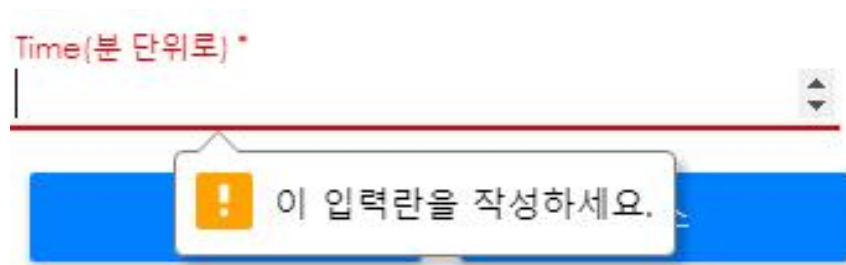


<그림 5-9> 현재상황 및 구별제어 화면

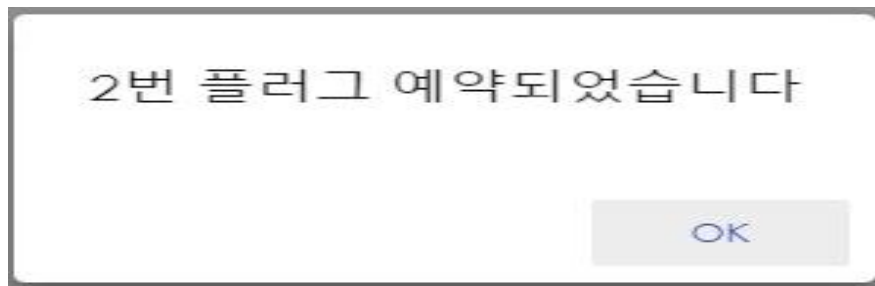
4. 예약제어

- 4.1. 예약제어화면은 다음 <그림 5-10>와 같다.
- 4.2.  버튼과 타이머 시간을 분 단위로 입력해서 개별적으로 구의 전원을 예약 on, off 할 수 있다.
- 4.3. 현재상황 및 구별 제어 화면과 동기화 된다.
- 4.4. ▲ 와 ▼ 버튼을 눌러 해당항목을 접거나 펼 수 있다.
- 4.5. <그림 5-11>과 같이 예약 시간을 입력하지 않으면, 경고창이 뜬다.
- 4.6.  버튼을 눌러서 예약을 설정할 수 있으며,  버튼을 눌러 예약하기 전의 입력 화면을 초기화 한다.
- 4.7. PC화면에서는  을 버튼을 눌러서 숫자를 조절할 수 있다.
- 4.8. 원하는 시간과 원하는 상태를 설정한 뒤  버튼을 누르면 <그림 5-12>처럼 알림창이 뜬다.





<그림 5-11> 필수 입력 경고 화면



<그림 5-12> 예약 성공 알림창

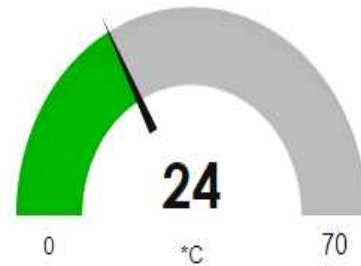
5. 실시간 온도 측정

- 5.1. ▲ 와 ▼ 버튼을 눌러 해당항목을 접거나 펼 수 있다.
- 5.2. <그림 5-13>과 같이 그래프와 <그림 5-14>와 같이 게이지로 온도를 나타낸다.
- 5.3. 센서에 의해 측정된 스마트 멀티플러그의 내부 온도가 0℃ ~ 36.5℃은 <그림 5-14>, 36.5℃ ~ 50℃은 <그림 5-15>, 50℃ ~ 70℃은 <그림 5-16>와 같이 게이지 색이 변한다.
- 5.4. 그래프의 경우 그래프에 커서를 가져다대면 <그림 5-17>과 같이 자세한 내용을 살펴볼 수 있다.

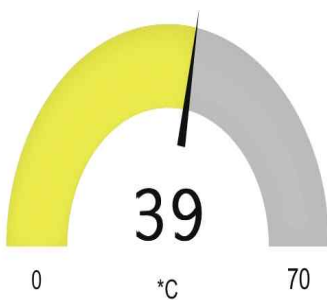
실시간 온도 측정



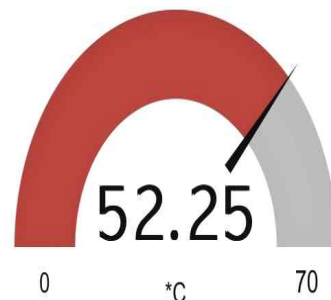
<그림 5-13> 그래프로 표현한 실시간 온도



<그림 5-14> 게이지로 표현한 실시간 온도



<그림 5-15> 온도 구간별 게이지 색깔 ②



<그림 5-16> 온도 구간별 게이지 색깔 ③



<그림 5-17> 그래프의 상세한 온도 수치 항목

제 6 장 결론 및 향후계획

본 논문에서는 Raspberry와 NodeMCU를 이용한 WiFi 기반의 스마트 멀티플러그를 구현하였다. 본 논문에서는 어떤 스마트 기기에도 대응할 수 있도록 Android와 iOS 기반의 애플리케이션이 아닌 웹페이지로 사용자가 해당 DDNS 서버의 주소를 알고 있다면, 언제든지 접근할 수 있는 스마트 멀티플러그 전용 제어 웹페이지를 구현하였다.

본 논문에서는 우리나라에서도 보급률이 높은 전기라는 에너지를 사용함에 따라 멀티탭이라는 기기가 길거리에서도 흔히 볼 수 있게 되었고, 그로 인해 부주의와 전기적 요인이라는 원인으로 나란히 우리나라의 화재발생원인 1, 2위를 다투고 있음을 확인할 수 있었다. 따라서 과열제어, 과전압제어를 기본으로 하는 스마트 멀티플러그의 구현이 우리나라의 화재발생률을 낮추고 혹여나 멀티탭의 전원을 끄지 않았을 경우의 불상사를 줄이는 데 큰 도움이 될 것으로 예상된다.

본 웹페이지는 일단 누구나 쉽게 접근할 수 있지만, 그만큼 해커들의 공격등에 취약하다는 것을 알 수 있다. 따라서 사용자의 아이디와 패스워드로 먼저 접근할 수 있도록 하여 해당 사용자가 아니라면 스마트 멀티플러그 전용 제어 웹 페이지에 접속할 수 없도록 함으로서 웹페이지의 보안성 측면이 충분히 보완될 수 있다.

또한 요즘 각광받고 있는 음성인식으로 음성인식으로도 스마트 멀티플러그를 제어할 수 있게 하고, 빅데이터를 활용해 사용자가 가장 많이 사용하는 시간대를 알아내 해당 시간에 자동으로 원하는 기기의 전원이 켜지게 함으로서 좀 더 편의성이 발전할 수 있을 것이다. 그리고 지금은 NodeMCU에 특정 WiFi의 SSID와 패스워드를 미리 입력함으로서 미리 입력한 네트워크 내에서 사용할 경우 다시 컴파일 해야 하는 불편함이 있지만, 후에 웹페이지에 사용자 로그인 화면과 함께 WiFi의 SSID와 패스워드를 입력하게 해서 어떤 곳이라도 컴파일의 불편함을 거치지 않고 바로 네트워크에 연결할 수 있게 한다면 보다 편의성을 보완할 수 있을 것이다.

참고 문헌

[1] 세계전기보급률

https://public.tableau.com/views/_4686/1_2?%3Aembed=y&%3AshowVizHome=no&%3AshowTabs=y&%3Adisplay_count=y&%3Adisplay_static_image=y

[2] 소방청 국가화재정보시스템

http://www.nfds.go.kr/fd_fire_anal_0001.jsf

[3] 네이버 스마트스토어 샤오미(Xiaomi) 멀티탭

<https://smartstore.naver.com/unicmnt/products/2315387342>

[4] 네이버 스마트스토어 스토리링크(STORY LiNK) 멀티탭

<https://smartstore.naver.com/storylink/products/675876619>

[5] 네이버 스마트스토어 와트드림(wattdream) 멀티탭

<https://smartstore.naver.com/insightpwr/products/655930225>

[6] 네이버 지식백과 MQTT

<https://terms.naver.com/entry.nhn?docId=3338580&cid=42346&categoryId=42346>

[7] MQTT 공식 홈페이지

<https://mqtt.org/>

[8] ISO 표준으로 지정된 MQTT

<https://www.iso.org/standard/69466.html>

[9] MQTT 프로토콜의 작동방식

<http://www.rfwireless-world.com/Terminology/MQTT-vs-HTTP.html>

[10] NodeMCU의 상세모습

http://mechasolution.com/shop/goods/goods_view.php?goodsno=539581&category=

[11] ESP8266 사용

http://mechasolution.com/shop/goods/goods_view.php?goodsno=539581&category=

부록 1. API

Smart Multi Plug + MQTT DDNS Server.ino

– Variable

ssid : NodeMCU와 연결할 WiFi의 SSID
password : NodeMCU와 연결할 WiFi의 패스워드
mqtt_server : NodeMCU와 연결할 MQTT 서버의 주소(IP주소나 DDNS 서버 주소)
espClient : MQTT 서버와 연결할 NodeMCU 클라이언트
ledGPIO14 : Plug01과 연결할 GPIO 핀번호
ledGPIO13 : Plug02와 연결할 GPIO 핀번호
ledGPIO12 : Plug03와 연결할 GPIO 핀번호
ledGPIO05 : Plug04와 연결할 GPIO 핀번호
seconds : 현재 시간 중 초
minutes : 현재 시간 중 분
hours : 현재 시간 중 시
day : 현재 시간 중 일
date : 현재 시간 중 날짜
month : 현재 시간 중 달
year : 현재 시간 중 년도
reserv_seconds[4] : 구별 최종 예상 예약시간 중 초
reserv_minutes[4] : 구별 최종 예상 예약시간 중 분
reserv_hours[4] : 구별 최종 예상 예약시간 중 시
resev_day[4] : 구별 최종 예상 예약시간 중 일
reserv_date[4] : 구별 최종 예상 예약시간 중 날짜
reserv_month[4] : 구별 최종 예상 예약시간 중 달
reserv_year[4] : 구별 최종 예상 예약시간 중 년도
weekDay[4] : 월 ~ 일까지의 요일
tMSB : 읽어낸 온도 값의 2의 보수의 정수 부분
tLSB : 읽어낸 온도 값의 정수가 아닌 분수나 수의 소수 아래의 값
temp3231 : 최종적으로 계산한 온도값
topic_reserv[4] : 구별 전송된 예약 topic 값

reservStatus[4] : 구별 전송된 예약 상태 값
now : NodeMCU가 전원이 들어오고 지난 시간
lastMeasure : 마지막으로 온도를 측정했던 시간
messageStatus : byte형의 Message를 String 형으로 변환
topic : 웹페이지가 publish한 topic
messageTemp : Message를 저장할 변수
strTime : Message를 통해서 분리해낸 예약시간
toIntTime : strTime을 int 형으로 변환
wantTime : 최종적으로 사용자가 상태를 변하기를 원하는 시간(분)
plug_num : 예약을 원하는 Plug 번호
topic_reserv[plug_num - 1] : 예약과 관련된 topic을 저장
temperatureTemp[7] : NodeMCU가 발행할 온도값을 저장

— Function

void setup_wifi() : NodeMCU와 WiFi 연결을 설정하는 함수

void callback(String topic, byte* message, unsigned int length) : 웹페이지로부터 publish한 topic을 subscribe하고 해당 topic에 대해 처리하는 함수

void makeMessageTemp(String topic, byte* message, unsigned int length) : 예약제어를 통해 publish된 topic을 나누고 해당 Message를 필요한 부분만 분리하는 함수

void reconnect() : NodeMCU와 MQTT 서버를 연결하는 함수

void setup() : NodeMCU의 전원이 들어오고 난 후 초기설정을 하는 함수

void loop() : NodeMCU의 전원이 계속 들어오는 한 무한적으로 반복하는 함수

void get3231Date() : 현재 RTC모듈의 시간을 구하는 함수

float get3231Temp() : RTC모듈에서 측정된 온도값을 구하는 함수

void watchConsole() : 스케치의 시리얼 모니터를 통해 RTC모듈의 현재 시간을 재설정하는 함수

void set3231Date() : 시리얼 모니터를 통해 얻은 값으로 재설정할 RTC모듈의 시간을 구하는 함수

byte decToBcd(byte val) : 10진수를 2진수로 변환하는 함수

Node-Red Setting

— Variable

id : 해당 flow의 id

type : 해당 flow의 type

label : 웹페이지에서 보여지게 될 해당 flow의 이름

name : Node-Red 설정 사이트 내에서 보여지게 될 해당 flow의 이름

icon : 해당 flow의 이름

collapse : flow의 접기, 펼치기 기능의 허용유무

order : flow의 순서

width : flow의 너비

disp : 웹페이지에서 해당 flow의 보여짐 유무

broker : MQTT 프로토콜 설정

qos : publish할 topic의 QoS 설정

topic : publish할 topic

position : 예약알림창 설정

displayTime : 예약알림창이 보여질 시간

xformat : 실시간 온도 측정 그래프 중 시간을 어떻게 나타낼 지에 대한 그래프의 x축 설정값

ymin : 온도를 나타낼 y축의 최솟값

ymax : 온도를 나타낼 y축의 최댓값

interpolate : 그래프를 어떤 모양으로 그릴지에 대한 값

onvalue : 스위치가 켜졌을 때 publish할 값

onvalueType : 스위치가 켜졌을 때 publish할 값의 자료형

offvalue : 스위치가 꺼졌을 때 publish할 값

offvalueType : 스위치가 꺼졌을 때 publish할 값의 자료형

gtype : 게이지타입

label : 게이지로 표현할 값의 이름

min : 게이지의 최솟값

max : 게이지의 최댓값

seg1 : 구간별로 색깔을 다르게 표현할 게이지의 첫 번째 값

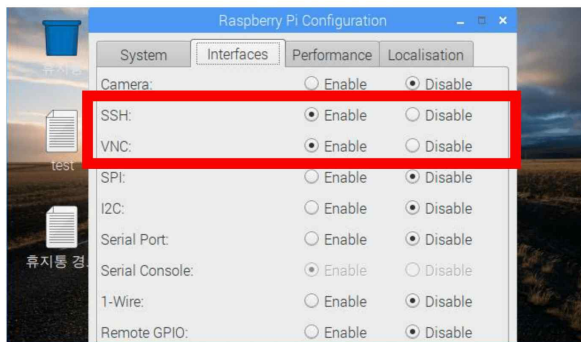
seg2 : 구간별로 색깔을 다르게 표현할 게이지의 두 번째 값

부록 2. Raspberry Pi 원격접속 설정

VNC 원격 서버 설정



설정 변경



1) `sudo apt-get install x11vnc xinetd`

2) `x11vnc`

기본으로 내장된 vnc 서버를 이용해 윈도우 환경에서도 라즈베리파이를 원격제어할 수 있도록 한다.
오른쪽에 있는 명령어를 실행한다.

2018-11-05

zam2695@naver.com

43

포트포워딩, DDNS 서버 설정



포트포워딩

현재 포트포워딩과 DDNS 서버 설정은

ipTIME 공유기를 사용했을 때를 가정으로 해서 진행합니다.



1) 192.168.0.1을 url창에 들어가서 관리자 페이지에 로그인한다.

2) 초기 id | pw = admin | admin

2018-11-05

zam2695@naver.com

45

포트포워딩, DDNS 서버 설정

포트포워딩



- 고급 설정
 - + 네트워크 관리
 - + 무선랜 관리
 - + NAT/라우터 관리
 - + 보안 기능
 - + 특수기능
 - + 트래픽 관리
 - + 시스템 관리

2018-11-05

zam2695@naver.com

46

포트포워딩, DDNS 서버 설정

포트포워딩

- 고급 설정
 - + 네트워크 관리
 - + 무선랜 관리
 - NAT/라우터 관리
 - 포트포워딩 설정
 - DMZ / Twin IP 설정
 - 포트트리거 설정
 - 기타기능 설정
 - 라우팅 테이블 관리
 - + 보안 기능
 - + 특수기능
 - + 트래픽 관리
 - + 시스템 관리

포트포워드 설정

정의된 리스트: 사용자정의 규칙이름

내부 IP주소: 192.168.0.0

현재 접속된 PC의 IP 주소로 설정(192.168.0.3)

프로토콜: TCP 외부 포트: ~ 내부 포트: ~

최대 60개의 규칙이 설정 가능합니다.

추가 취소

낮은 번호일수록 우선순위가 높습니다.
규칙이름을 클릭하시면, 해당 규칙을 수정할 수 있습니다.

동작	규칙이름	내부 IP	프로토콜	외부 포트	내부 포트	삭제
<input checked="" type="checkbox"/>	MQTT		tcp	1883-1883	1883-1883	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Node-Red		tcp	1880-1880	1880-1880	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Raspberry		tcp	80-80	80-80	<input type="checkbox"/>
<input checked="" type="checkbox"/>	SSH		tcp	2222-2222	22-22	<input type="checkbox"/>
<input checked="" type="checkbox"/>	VNC		tcp	5900-5900	5900-5900	<input type="checkbox"/>

2018-11-05

zam2695@naver.com

47

포트포워딩, DDNS 서버 설정

DDNS 설정

- 고급 설정
 - + 네트워크 관리
 - + 무선랜 관리
 - + NAT/라우터 관리
 - + 보안 기능
 - 특수기능
 - DDNS 설정**
 - WOL 기능
 - 호스트검색
 - 공지/광고 기능
 - IPTV 설정
 - + 트래픽 관리
 - + 시스템 관리

DDNS 설정

서비스 공급자: ipTIME DDNS
 호스트이름:
 사용자 ID:
 사용자 암호:
 보안문자(영문 소문자 5글자)를 입력하세요.

보안문자: fslay

새로고침

• ipTIME DDNS를 사용하실때는 별도의 등록과정이 필요 없습니다.
 • 호스트이름은 ipTIME.org 로 끝나야 합니다. (예> testtest.ipTIME.org)
 • 사용자ID는 E-mail주소를 입력해야 합니다.
 • 한 개의 호스트만 등록할 수 있습니다.

추가

호스트이름	접속상태	강신	삭제
kimjs0303.ipTIME.org	정상 등록	<input type="checkbox"/>	<input type="checkbox"/>

2018-11-05

zam2695@naver.com

48

포트포워딩, DDNS 서버 설정

라즈베리파이 DDNS 설정

Logon
 Registration

Welcome to myq-see.com

Create a user account or choose existing users below to begin.

DDNS account creation.

NEW USER REGISTRATION
 EMAIL ADDRESS:
 PASSWORD:
 PASSWORD CONFIRM:
 FIRST NAME:
 LAST NAME:
 SECURITY QUESTION: My first phone number
 ANSWER:
 CONFIRM YOU'RE HUMAN: 2+7=
 New Captcha
 Solve the problem above.
 Submit Reset

Already have an account? Click here to logon.

www.myq-see.com 접속 후 가입한 뒤
상세 사항 기입 후 DDNS 서버 생성

2018-11-05

zam2695@naver.com

49

포트포워딩, DDNS 서버 설정



라즈베리파이 DDNS 설정

```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo apt-get install ddclient  
패키지 목록을 읽는 중입니다... 완료  
의존성 트리를 만드는 중입니다  
상태 정보를 읽는 중입니다... 완료  
ddclient is already the newest version (3.8.3-1.1).  
0개 업그레이드, 0개 새로 설치, 0개 제거 및 0개 업그레이드 안 함.  
pi@raspberrypi:~$
```

- 1) sudo apt-get install ddclient
(입력하라는 내용이 나오면 공란에 엔터)
- 2) sudo /etc/init.d/ddclient stop

```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo /etc/init.d/ddclient stop  
[ ok ] Stopping ddclient (via systemctl): ddclient.service.  
pi@raspberrypi:~$
```

2018-11-05

zam2695@naver.com

50

포트포워딩, DDNS 서버 설정



라즈베리파이 DDNS 설정

```
----- ddclient.conf -----  
# /etc/ddclient/ddclient.conf  
#  
daemon=300  
syslog=yes  
pid=/var/run/ddclient.pid  
use=web  
login=xxxxxxx@gmail.com  
password=*****  
server=www.myq-see.com  
protocol=dyndns2  
xxx.myq-see.com  
-----
```

- 1) sudo nano /etc/ddclient.conf
좌측의 사항처럼 작성 후 저장
- 2) sudo /etc/init.d/ddclient start
- 3) sudo /usr/sbin/ddclient -daemon=0 -debug -verbose -noquiet
- 4) sudo systemctl start ddclient.service
- 5) login과 password는 가입시 만들었던 계정정보
- 6) 마지막은 생성한 DDNS 주소
- 7) 참고 :
<http://blog.cjbox.kr/entry/%EB%9D%BC%EC%A6%88%EB%B2%A0%8C%EC%9D%B43-DDNS-%ED%99%9C%EC%9A%A9>

2018-11-05

zam2695@naver.com

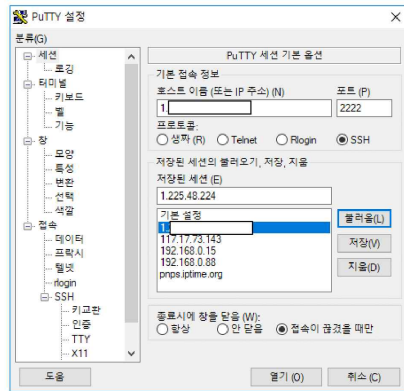
51

원격 접속



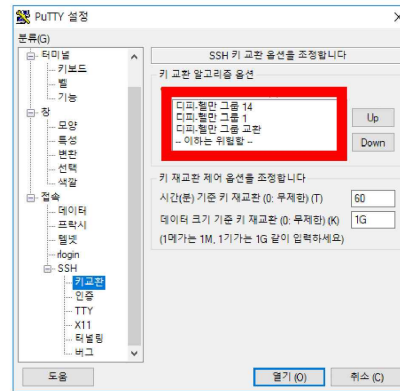
원격 접속 putty

1) 혹은 라즈베리파이의 DDNS 주소를 입력해도 된다.



인터넷 정보

인터넷 연결 상태	인터넷에 정상적으로 연결됨
인터넷 연결 방식	중적 IP 연결
인터넷 연결 시간	외부 IP 주소 1 1 시간 15 분 8 초



2018-11-05

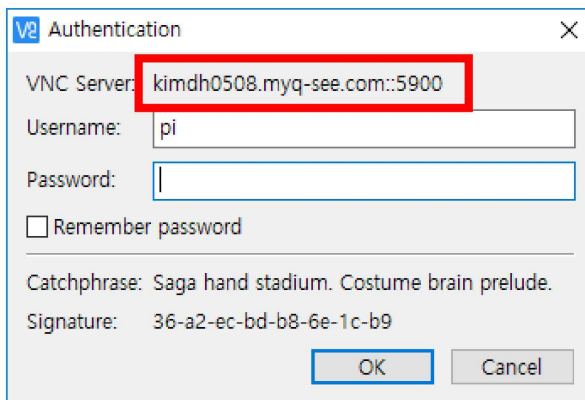
zam2695@naver.com

53

원격 접속



원격 접속 vnc



앞에서 라즈베리파이 DDNS 서버의 주소 뒤에 :5900 을 붙여준 뒤 연결을 설정한다.
안된다면 그냥 DDNS 주소를 넣어줘도 된다.

2018-11-05

zam2695@naver.com

54

원격 접속



포트번호 확인하기

- 1) 라즈베리파이와 같은 데비안 종류의 os는 ufw라는 방화벽을 이용해 특정 포트를 열고 닫는다.
- 2) 혹여나 원격접속 오류가 날 경우에 포트가 열렸는지 확인해본다.
- 3) ufw 명령어 설치 : `sudo apt-get install ufw`
- 4) ufw 활성화 : `sudo ufw enable`
- 5) ufw 비활성화 : `sudo ufw disable`
- 6) 활성화 후 상태확인 : `sudo ufw status verbose`
- 7) 허용할 포트 번호 : 22, 5900, 80
- 8) 포트번호 허용 : `sudo ufw allow 포트번호`
- 9) 내부망 네트워크를 사용한다면,
telnet [(내부망 IP주소, 여기서는 라즈베리파이)ip] [포트번호] 를 통해 해당 포트가 열려 있는지 확인이 가능하다.

2018-11-05

zam2695@naver.com

55

원격 접속



포트번호 확인하기

방화벽이 열리지 않은 상태

```
Trying 123.123.123.123...
telnet: connect to address 123.123.123.123: Connection timed out
telnet: Unable to connect to remote host: Connection timed out
```

프로세스가 실행되고 있지 않은 상태

```
Trying 123.123.123.123...
telnet: connect to address 123.123.123.123: Connection refused
telnet: Unable to connect to remote host
```

정상적으로 연결되고 프로세스가 실행된 상태

```
Connected to 123.123.123.123
Escape character is '^['.
```

2018-11-05

zam2695@naver.com

56

```
pi@raspberrypi: ~  
pi@raspberrypi:~$ sudo ufw status verbose  
Status: active  
Logging: on (low)  
Default: deny (incoming), allow (outgoing), disabled (routed)  
New profiles: skip  
  
To Action From  
--  
22 ALLOW IN Anywhere  
8181 ALLOW IN Anywhere  
80 ALLOW IN Anywhere  
8080 ALLOW IN Anywhere  
5900 ALLOW IN Anywhere  
22 (v6) ALLOW IN Anywhere (v6)  
8181 (v6) ALLOW IN Anywhere (v6)  
80 (v6) ALLOW IN Anywhere (v6)  
8080 (v6) ALLOW IN Anywhere (v6)  
5900 (v6) ALLOW IN Anywhere (v6)  
  
pi@raspberrypi:~$
```

부록 3. Arduino Sketch 설정

Arduino Sketch



Arduino Sketch 설치



Download the Arduino IDE



- 1) <https://www.arduino.cc/>
- 2) SOFTWARE -> DOWNLOADS
- 3) ZIP파일이나 INSTALLER 중 선호하는 걸 선택한 후 다운로드 설치.

2018-11-05

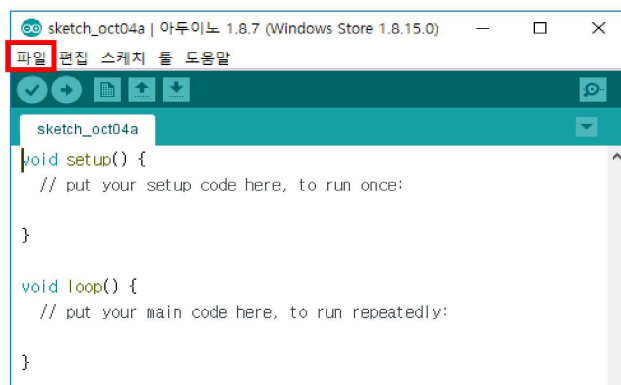
zam2695@naver.com

64

Arduino Sketch



NodeMCU 개발환경 구현하기



파일 -> 환경설정을 누른다.

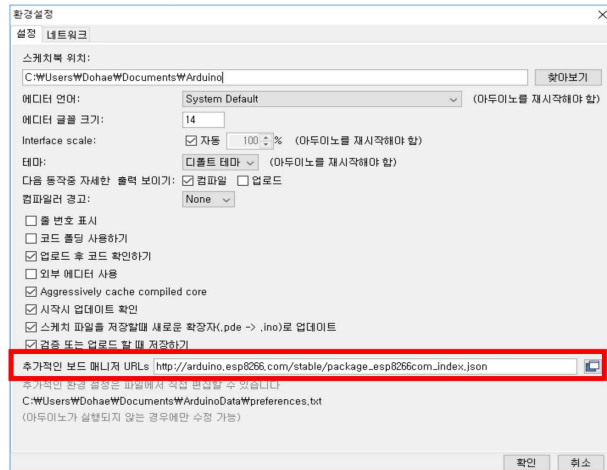
2018-11-05

zam2695@naver.com

65

Arduino Sketch

NodeMCU 개발환경 구현하기



- 1) 추가적인 보드 매니저 URLs에
http://arduino.esp8266.com/stable/package_esp8266com_index.json
- 2) 입력한 후 확인 버튼을 누른다.

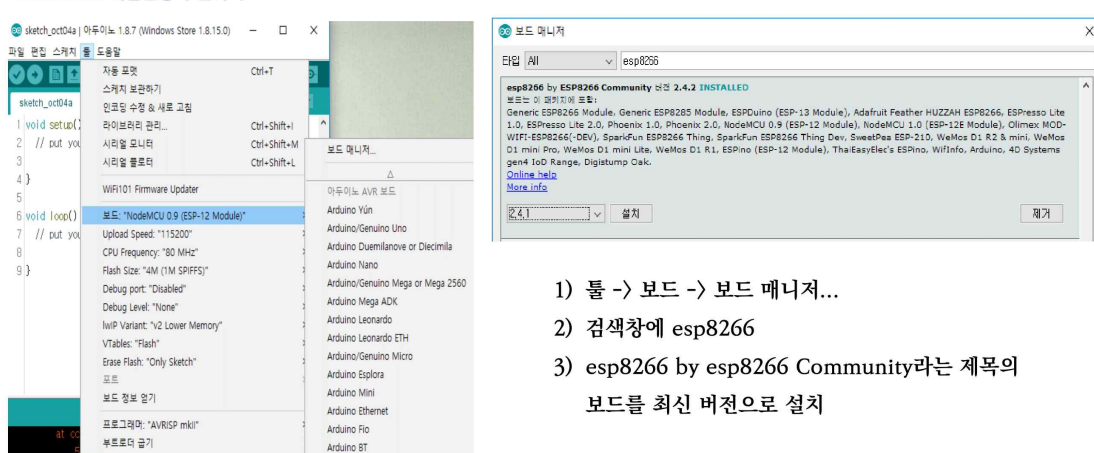
2018-11-05

zam2695@naver.com

66

Arduino Sketch

NodeMCU 개발환경 구현하기



- 1) 툴 -> 보드 -> 보드 매니저...
- 2) 검색창에 esp8266
- 3) esp8266 by esp8266 Community라는 제목의 보드를 최신 버전으로 설치

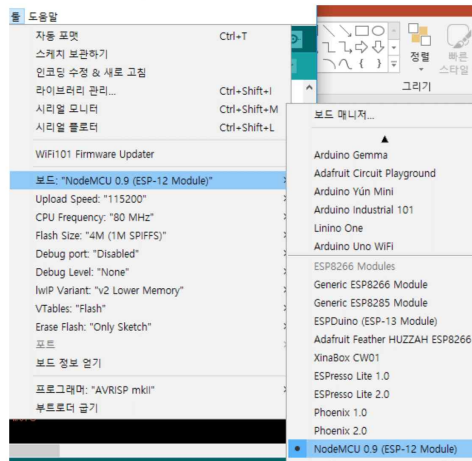
2018-11-05

zam2695@naver.com

67

Arduino Sketch

NodeMCU 개발환경 구현하기



- 1) 툴 -> 보드 -> NodeMCU 0.9 (ESP-12 Module) 선택
- 2) Upload Speed : 115200
- 3) CPU Frequency : 80 MHZ
- 4) Flash Size : "4M (1M SPIFFS)"
- 5) IwIP Variant : "V2 Lower Memory"
- 6) VTables : "Flash"
- 7) Erase Flash : "Only Sketch"
- 8) 프로그래머 : "ANRSIP mkII"

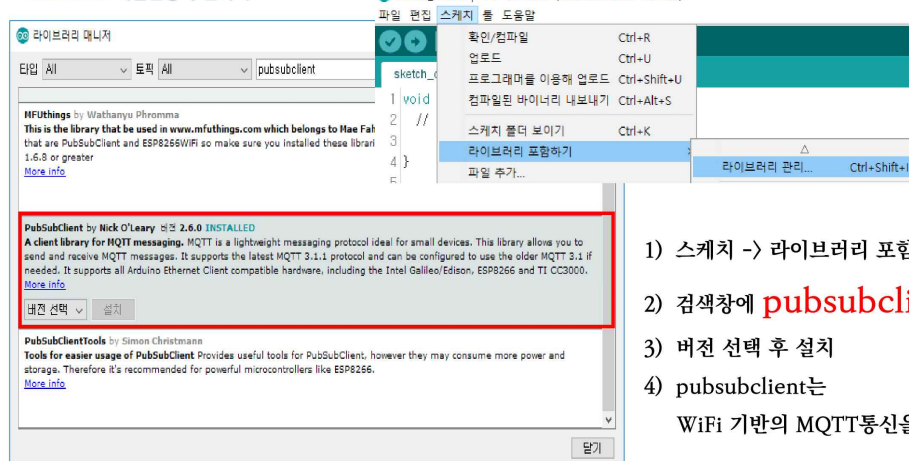
2018-11-05

zam2695@naver.com

68

Arduino Sketch

NodeMCU 개발환경 구현하기



- 1) 스케치 -> 라이브러리 포함하기 -> 라이브러리 관리
- 2) 검색창에 **pubsubclient** 입력
- 3) 버전 선택 후 설치
- 4) pubsubclient는
WiFi 기반의 MQTT통신을 위한 라이브러리

2018-11-05

zam2695@naver.com

69