아두이노 와이파이 통신 및 제어

와이파이(Wi-Fi, Wireless Fidelity)

- 근거리 무선망으로 전자기기들이 무선랜(WLAN)에 연결할 수 있게 하는 기술
- 주로 2.4 기가헤르츠 UHF 및 5 기가헤르츠 SHF ISM 무선 대역을 사용

  - 2.4GHz의 범위

    - 2.4~2.462 GHz이고, 그 사이의 5MHz마다 하나의 채널이므로 총 11개의 채널이 있음
    - 2.4GHz에 사용할 수 있는 채널은 11개이지만, 예를 들어 802.11b는 RF 대역폭에 22MHz를 필요로 하므로 실제로 서로 겹치거나 간섭하지 않는 채널은 3개뿐임

  - 5GHz의 범위

    - 5.180 ~ 5.850 GHz이며 역시 5MHz마다 각 채널이 분리되어 있으므로 사용할 수 있는 채널은 36~165개임
    - 그러므로 802.11ac의 160MHz 대역폭 요건을 충족할 수 있음
    - 그러나 주파수가 높을수록 파장이 짧아져 회절 능력이 저하됨
    - 장애물을 통과하기 어렵고 동일한 전력률에서 2.4GHz에 비해 실효 전송 범위가 작은 이유가 이 때문임

  - 범위 측면에서 2.4GHz는 5GHz보다 더 멀리 도달할 수 있지만, 속도는 5GHz보다 느림
  - 5Ghz는 전송 범위가 더 작을 수 있지만 속도는 더 빠름
  - 그렇기 때문에 장치와 라우터 간의 전송이 거리의 영향을 받지 않도록 장치가 라우터에 더 잘 연결되는 최적의 지점에 라우터를 배치하는 것이 중요함
  - 라우터가 실내에 있고 2.4GHz 802.11b/g/n만 지원하는 경우, 최대 3개의 라우터를 배치하고 서로 간섭하지 않도록 채널을 1/6/11로 설정하는 것을 권장함
  - 무선랜은 일반적으로는 암호로 보호되어 있지만, 대역 내에 위치한 어느 장치라도 무선랜 네트워크의 자원에 접근할 수 있도록 개방도 가능함

ESP8266 와이파이 모듈(Wifi module)

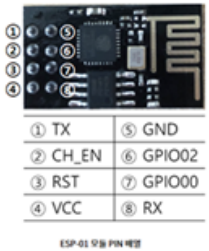
- ESP8266 칩셋<컴퓨터나 기타 전자 시스템에서 특정 기능을 수행하는 하나 이상의 집적 회로(IC)로 구성된 그룹>을 사용하기 때문에 ESP8266 와이파이 모듈이라고 함
  - 가장 범용적으로 사용되는 칩셋은 ESP8266 칩셋인데, ESP8266 칩셋으로 나온 보드는 여러가지가 있음
    - ESP-01 제품을 가장 많이 찾을 수 있으나 ESP 시리즈는ESP-14까지 출시되어 있음
    - 조금 더 사용이 편한 모듈로는 nodemcu와 같은 모듈이 있음
    - 아두이노용으로 나온 와이파이 실드의 경우에는 가격이 비쌈

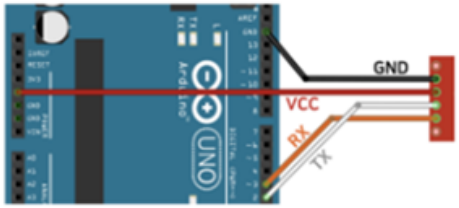
ESP-01 모듈

- 가장 저렴한 모듈로써 아두이노에서 사용하기에는 몇가지 해결해야 함
  - ESP 01은 3V로 동작을 하는데, 전류 소모량이 많음
    - ESP-01 모듈은 전류 소모량이 많고, TX와 RX 신호가 5V 출력이기 때문에 아두이노 보드 pin에 바로 연결하면 정상적으로 작동이 안되고 모듈이 고장날 수도 있음
- 아두이노와 9600 bps 속도로 시리얼 통신(Serial communication)을 주고 받아야 하는데 ESP-01 모듈은 115200 bps로 초기 설정되어 있음(때문에 펌웨어 업그레이드를 통해서 디폴트 통신 속도를 9600 bps로 변경해줘야 함)
  - 여러 배선 설계를 해야 하고 별도의 펌웨어 업그레이드 프로그램을 설치해야하기 때문에 그 과정이 까다로움
    - USB to ESP01 연결 아답터를 이용 - 아두이노 보드를 거치지 않고 바로 PC USB에서 ESP01로 쉽게 펌웨어를 업데이트할 수 있게 해줌
      - 펌웨어 업그레이드 방법
      - ESP8266 esp-01 활용하기-펌웨어 업데이트
      - ESP8266(ESP-01)펌웨어 업데이터

- ESP-01 어댑터라는 모듈을 사용

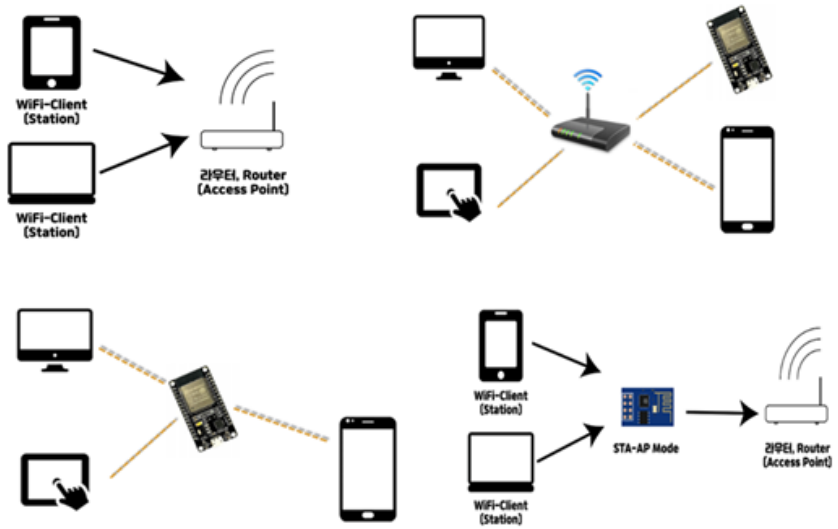
- 어댑터의 4 Pin(VCC, GND, RX, TX)만으로 훨씬 편하게 배선을할 수 있음
- 아답터 위에 ESP01 보드를 설치하면 되는 형태 - https://m.blog.naver.com/roboholic84/221261124179
- 레귤레이터가 내장되어 있기 때문에 아두이노 5V에 VCC를 연결하면 되고, RX와 TX를 각각 3번과 2번 포트에 연결
- 어댑터를 사용하면 펌웨이 업데이트도 안해도 됨
    - 아두이노에서 WIFI 연결하는 방법 - http://bcho.tistory.com/1280 자료 참고





ESP8266의 'AP모드'와 '스테이션(Station) 모드'의 개념

- AP(액세스 포인트) - 다른 기기들이 네트워크에 접속할 수 있도록 wifi 무선 접속환경을 제공하는 기기(wifi 공유기)
    - 다른 장치(디바이스, 휴대폰, 태블릿, 노트북 등)가 접속(Access)할 수 있는 곳(Point)
- 스테이션 - wifi 공유기에 접속을 하려는 기기
    - 공유기(AP)에 접속할 수 있게, AP에서 뿌리는 데이터 신호들을 받을 수 있는 모드를 뜻함
    - 따라서 STA 모드를 하게 되면 공유기와 같은 역할을 수행하지 못함



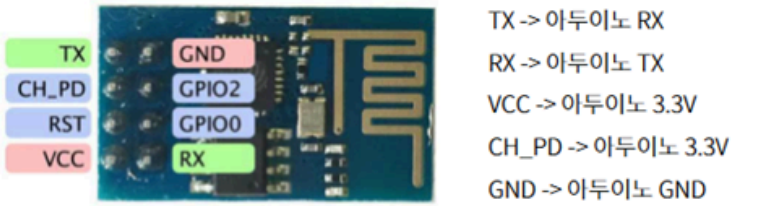
- 전용 AP 공유기의 경우 내부 IP주소뿐 아니라, 공인 IP주소를 가지고 있는데, 스테이션의 연결을 받아주면서 스테이션들이 직접 인터넷과 같은 외부 네트워크에 연결하도록 지원을 함
    - 하지만, 이런 스테이션들의 직접적인 외부 네트워크(인터넷) 연결을 지원하지 못하고, AP 자신만 직접적으로 외부 네트워크(인터넷) 연결이 가능한 AP를 소프트 AP(Soft AP)라고 함
        - 일반적인 AP(Access Point) 공유기는 무선 WiFi 네트워크 지원과 함께 유선 네트워크를 함께 지원하는데, ESP8266은 유선 인터페이스를 지원하지 않기 때문에, 그래서 유사한 기능만 지원한다는 의미로, 소프트 AP라고 불리기도 함
- ESP8266은 3가지 모드로 설정하여 사용할 수 있는데 3가지 형태중 하나로 선택하여 사용할 수 있음
    - 1 : Stand alone(스테이션)
        - 클라이언트로 작동하는 모드로 AP에 붙어서 네트워크 통신을 할 수 있음
    - 2 : AP
        - ESP8266이 서버가 되는 모드로 다른 단말이 ESP8266에 Wifi로 연결될 수 있게 함
    - 3 : AP + Standalone(스테이션)
        - 서버와 클라이언트를 동시에 지원하는 모드

하드웨어 시리얼과 소프트웨어 시리얼

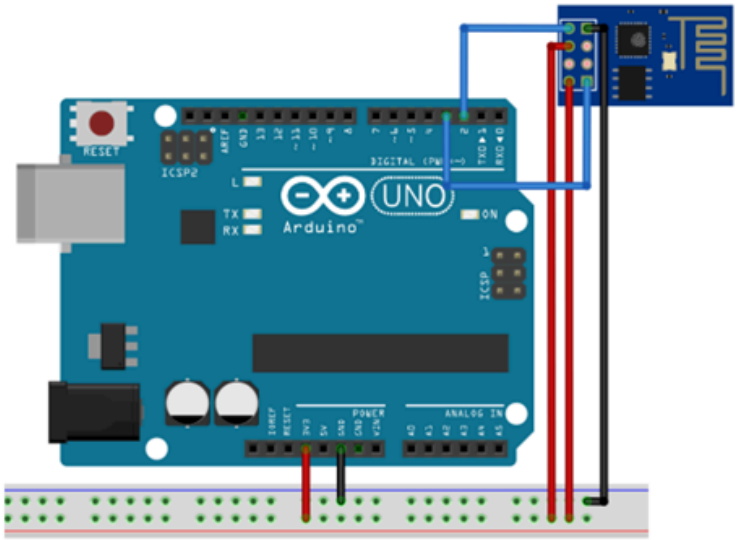
- Hardware Serial
  - 아두이노 우노의 0,1 번 포트는 시리얼 통신을 위한 RX,TX 포트
  - 하드웨어 시리얼 포트는 PC와 연결되어 있을때 PC와 통신을 목적으로 사용됨
  - 따라서 하드웨어 시리얼을 사용하지 않은 것인데, ESP8266 관련 라이브러리들을 살펴보면, 대부분 하드웨어 시리얼을 사용하는 것으로 보임
- 소프트웨어 시리얼
  - ESP-01 연결시에 포트를 2,3번 포트를 사용하고, 코드는 SoftSerial 라이브러리를 이용하였음
  - 그러면 하드웨어 시리얼 포트를 사용하지 않고 2,3번 포트를 사용한 후에 소프트웨어 시리얼 처리를 한 이유는 무엇일까?
- 그러면 이 문제를 어떻게 해결할 것일까?
  - 아두이오 우노 기판을 보면 TX와 RX라고 써있는 LED가 있는데, 하드웨어 시리얼 통신을 할때만 깜빡 거림
  - PC에 연결이 되어 있을 때, AT 명령을 실행할 때는 이 LED가 빛나지 않고, 코드를 업로드 할 때만 빛이 남, 즉 코드 업로드 시에만 하드웨어 시리얼을 PC가 사용한다는 것이기 때문에, 코드를 업로드한 후에 ESP01 보드를 TX,RX 단자에

연결하면 하드웨어 시리얼을 사용할 수 있음

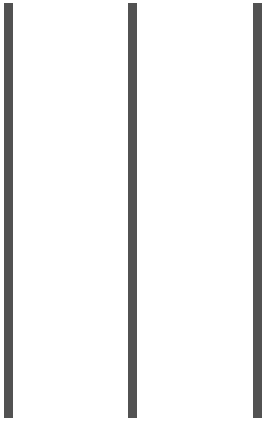
아두이노와 연결하기



TX -> 아두이노 RX
RX -> 아두이노 TX
VCC -> 아두이노 3.3V
CH_PD -> 아두이노 3.3V
GND -> 아두이노 GND



- TX와 RX pin은 각각 아두이노의 2번 Pin과 3번 Pin에 연결
  - 왜 아두이노의 RX-pin0, TX-pin1에 맞춰서 연결하지 않나요?
    - 아두이노가 스케치를 통해 신호를 받을 때나 PC로 코딩된 프로그램을 업로드받을 때 시리얼 통신을 사용



- TX와 RX는 각각 Transmit(전송), Receive(수신)이란 뜻으로 시리얼 통신을 할 때 데이터를 전송하고 수신하는 통로가 됨
- 이미 아두이노에 USB가 연결되어 있는 동안은 아두이노의 TX와 RX는 PC와 연결되어 있어서 아두이노와 ESP-01 와이파이 모듈 사이의 시리얼 통신을 위해 새로운 RX, TX를 디지털 Pin2번과 3번에 각각 지정을 해야 함
- 실제로 아두이노의 RX-pin0, TX-pin1에 연결하고 프로그램을 업로드하면 정상적으로 업로드되지 않음
- 주의 : 전송(TX) ---> 수신(RX) 짝으로 연결해야 함 - ESP-01 모듈에서 전송(TX)한 데이터를 아두이노에서 수신(RX)하고 또 반대로도 이뤄지기 때문임

AT 커맨드란?

- 미국 헤이즈 마이크로컴퓨터(Hayes Microcomputer products)사의 스마트 모뎀(Smart modem) 및 그 호환 모뎀을 제어하기 위하여 사용되는 명령
  - 현재 사실상의 표준으로 되어 있는 거의 모든 모뎀이 대응하고 있음
- 본래의 명칭은 헤이즈 명령어(Hayes command)인데, 통상 AT로 명령어가 시작되기 때문에 AT 명령어라고 불림
- 여기에서 AT는 주의 또는 주목이라는 의미의 attention의 약자
  - AT 명령어의 예를 들면, ATZ라고 입력하면 모뎀이 초기화되어 아무 명령도 입력되지 않은 상태로 되돌아가고 ATX1이라고 하면 회선 접속 시에 통신 속도를 표시함

AT 명령 실행

- 연결이 제대로 되었으면 AT를 입력할 경우 OK 응답이 오는 것을 볼 수 있음
- ESP01 보드의 통신 속도를 변경

- AT+CIOBAUD=9600 명령으로 변경하도록 되어 있는데, 이 명령은 최신 펌웨어에서는 동작하지 않고 AT+IPR이라는 명령을 사용해야 함
- 이 명령은 연결되어 있는 동안만 통신 속도를 변경하고 다시 ESP-01 디바이스를 뺐다가 다시 끼면 원래 통신속도로 돌아감

- ESP-01의 통신속도를 영구적으로 바꿔줄 수 있는 명령

  - AT_UART_DEF라는 명령을 사용 - https://www.esp8266.com/viewtopic.php?f=13&t=718
  - AT+ UART_DEF=,,,
  - 통신속도를 9600으로 영구적으로 변경하기 위해서는 아래 명령을 수행

    - AT+UART_DEF=9600,8,1,0,0

AT COMMAND LISTING

	COMMAND	FUNCTION
1	AT	Test UART Connection
2	AT+RESET	Reset Device
3	AT+VERSION	Querry firmware version
4	AT+ORGL	Restore settings to Factory Defaults
5	AT+ADDR	Query Device Bluetooth Address
6	AT+NAME	Query/Set Device Name
7	AT+RNAME	Query Remote Bluetooth Device's Name
8	AT+ROLE	Query/Set Device Role
9	AT+CLASS	Query/Set Class of Device CoD
10	AT+IAC	Query/Set Inquire Access Code
11	AT+INQM	Query/Set Inquire Access Mode
12	AT+PSWD	Query/Set Pairing Passkey
13	AT+UART	Query/Set UART parameter
14	AT+CMODE	Query/Set Connection Mode
15	AT+BIND	Query/Set Binding Bluetooth Address
16	AT+POLAR	Query/Set LED Output Polarity
17	AT+PIO	Set/Reset a User I/O pin
18	AT+MPIO	Set/Reset multiple User I/O pin
19	AT+MPIO?	Query User I/O pin
20	AT+IPSCAN	Query/Set Scanning Parameters
21	AT+SNIFF	Query/Set SNIFF Energy Savings Parameters
22	AT+SENM	Query/Set Security & Encryption Modes
23	AT+RMSAD	Delete Authenticated Device from List
24	AT+FSAD	Find Device from Authenticated Device List
25	AT+ADCN	Query Total Number of Device from Authenticated Device List
26	AT+MRAD	Query Most Recently Used Authenticated Device
27	AT+STATE	Query Current Status of the Device
28	AT+INIT	Initialize SPP Profile
29	AT+INQ	Query Nearby Discoverable Devices
30	AT+INQC	Cancel Search for Discoverable Devices

ERROR CODES

ERROR CODE	VERBOSE
0	Command Error/Invalid Command
1	Results in default value
2	PSKEY write error
3	Device name is too long (>32 characters)
4	No device name specified (0 lenght)
5	Bluetooth address NAP is too long
6	Bluetooth address UAP is too long
7	Bluetooth address LAP is too long
8	PIO map not specified (0 lenght)
9	Invalid PIO port Number entered
A	Device Class not specified (0 lenght)
B	Device Class too long
C	Inquire Access Code not Specified (0 lenght)
D	Inquire Access Code too long
E	Invalid Iquire Access Code entered
F	Pairing Password not specified (0 lenght)
10	Pairing Password too long (> 16 characters)
11	Invalid Role entered
12	Invalid Baud Rate entered
13	Invalid Stop Bit entered
14	Invalid Parity Bit entered
15	No device in the Pairing List
16	SPP not initialized
17	SPP already initialized
18	Invalid Inquiry Mode
19	Inquiry Timeout occured
1A	Invalid/zero lenght address entered
1B	Invalid Security Mode entered
1C	Invalid Encryption Mode entered

예제 1) - 아두이노와 와이파이 연결 테스트

- 1) ESP-01의 기본 baud rate는 115200

- 2) 현재 상태의 작동 여부를 확인하기 위해 하드웨어와 소프트웨어 시리얼 baud rate를 둘 다 115200으로 맞추어 주었음
- 3) 115200으로 해놓아도 AT command의 사용에는 지장이 없음
- 4) 아두이노 시리얼 모니터의 전송 옵션을 Both NL&CR(시리얼 모니터에서 줄 입력 방식을 바꾸어야함

    - 보통은 line ending 없음이 선택되어있는데, 안되면 이것을 Both NL&CR로 변경)

- 5) 보드 레이트를 115200으로 맞추어 주고 아두이노 시리얼 모니터 입력창에 AT를 입력하고 엔터를 침
- 6) 응답으로 OK가 들어온다면 기기의 정상작동과 연결이 잘 되었음을 확인할 수 있음
- 7) 수신된 데이터에 읽을 수 없는 문자가 섞여 있을 수 있지만 OK만 확인된다면 문제는 없음
- 아두이노에서 softwareSerial을 통해 안정적으로 사용하기 위해서는 baud rate를 9600으로 변경해야 함

    - ESP-01의 baud rate를 변경하는 방법으로는 기본 baud rate가 9600으로 설정된 펌웨어를 모듈에 업데이트하는 방법
    - AT command(AT+UART_DEF=9600,8,1,0,0)를 사용하여 변경해주는 방법이 있음

```
//WIFI 연결 테스트 - Serial_basic_esp01(#https://postpop.tistory.com/23)
//코드 내용 - 아두이노 콘솔에서 받은 명령을 ESP01 시리얼 포트로 전송하고 ESP-01에서
//나온 결과값을 아두이노 콘솔에 출력하도록 하는 코드

#include <SoftwareSerial.h>
#define rxPin 3
#define txPin 2
SoftwareSerial esp01(txPin, rxPin);

void setup() {
  Serial.begin(115200); //시리얼 모니터
  esp01.begin(115200);  //와이파이 시리얼

  //AT+UART_DEF=9600,8,1,0,0를 이용하여 변경
  //Serial.begin(9600);   //시리얼 모니터
  //esp01.begin(9600);    //와이파이 시리얼
}

void loop() {
  if (esp01.available()) {
    Serial.write(esp01.read());  //와이파이 측 내용을 시리얼 모니터에 출력
  }
  if (Serial.available()) {
    esp01.write(Serial.read());   //시리얼 모니터 내용을 와이파이 측에 쓰기
  }
}
```

```
28 //https://m.blog.naver.com/withmymentor/221895371214
29 //https://rasino.tistory.com/297
```

Wifi-Ex.ino

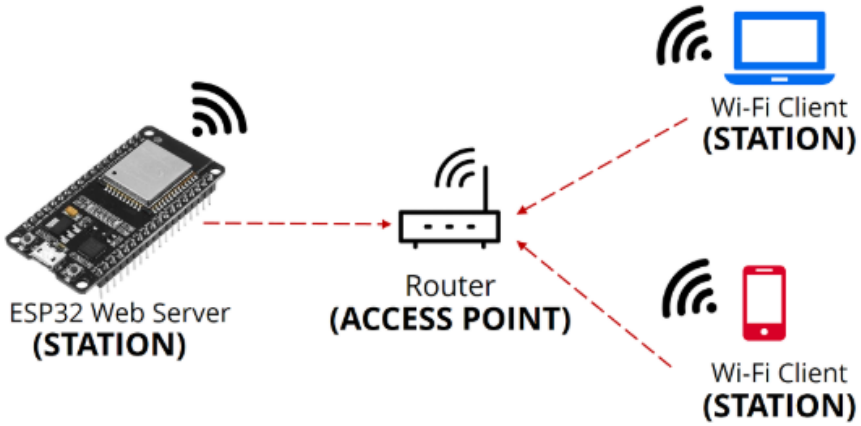
```
1  //https://m.blog.naver.com/withmymentor/221895371214
2  //https://rasino.tistory.com/297
3  #include <SoftwareSerial.h>
4  SoftwareSerial wifi(2, 3);
5
6  void setup() {
7   wifi.begin(9600);
8   Serial.begin(9600);
9  }
10
11 void loop() {
12  if (Serial.available() > 0)
13    wifi.write(Serial.read());
14  if (wifi.available() > 0)
15    Serial.write(wifi.read());
16 }
```

Output Serial Monitor ×

Message (Enter to send message to 'Arduino Uno' on 'COM4') Both NL & CR 9600 baud

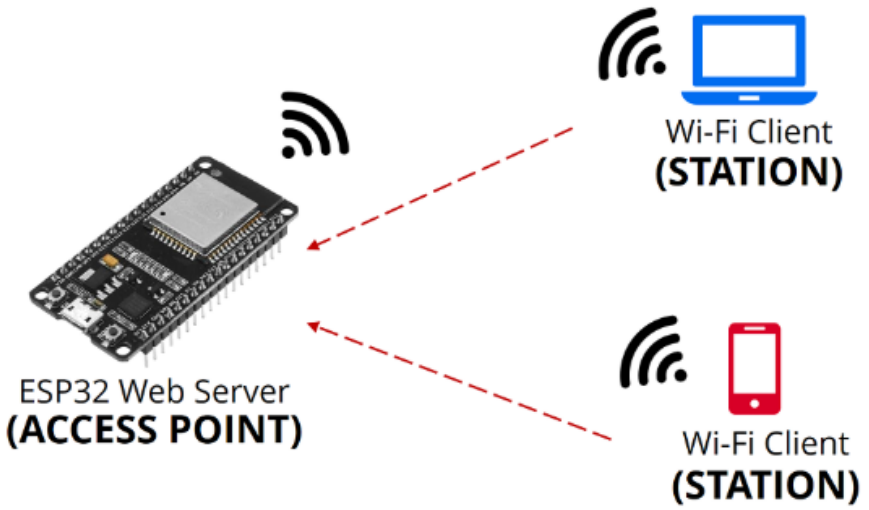
```
AT

OK
AT

OK
```

예제 2) - Station Mode




```
#include <WiFi.h>

const char *ssid_Router     = "******"; //Enter the router name - Wi-Fi 상태에서 무선 속성을 선택
const char *password_Router = "******"; //Enter the router password

void setup(){
  Serial.begin(115200);
  delay(2000);
  Serial.println("Setup start");

  //라우터의 SSID와 password를 인터넷 연결에 사용할 공유기의 이름과 암호로 입력
  WiFi.begin(ssid_Router, password_Router); //WiFi.begin()으로 ESP32 내부의 WiFi를 활성화

  Serial.println(String("Connecting to ")+ssid_Router);
  while (WiFi.status() != WL_CONNECTED){ //WiFi 상태가 WL_CONNECTED가 될 때 까지 대기, 0.5초마다 시리얼 포트로 점을 찍어줌
    delay(500);
    Serial.print(".");
  }
  Serial.println("WnConnected, IP address: ");
  Serial.println(WiFi.localIP()); //공유기(AP)에 연결이 되면?? - 연결이 되었다는 메시지와 공유기에서 받아온 IP 번호를 출력
  Serial.println("Setup End");
}

void loop() {
}
```

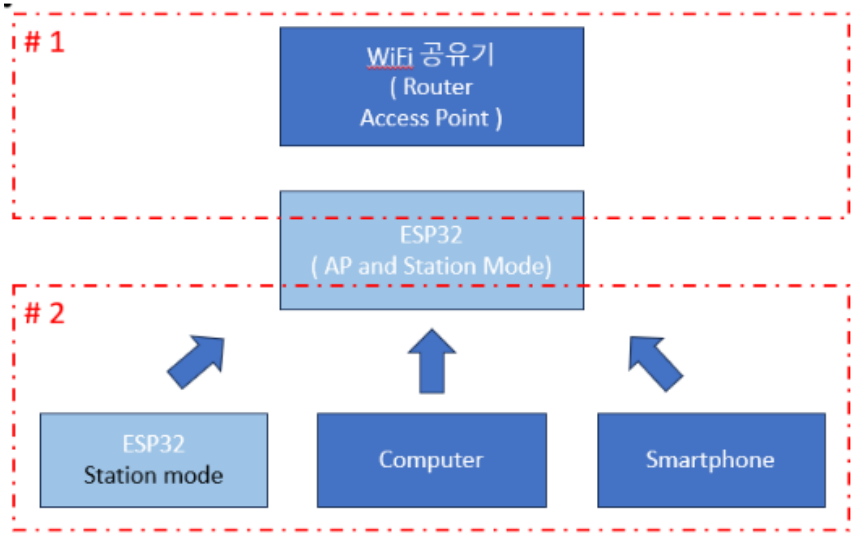
예제 3) - AP Mode




```
#include <WiFi.h>

const char *ssid_AP     = "WiFi_Name"; //Enter the router name
const char *password_AP = "12345678"; //Enter the router password

IPAddress local_IP(192,168,1,100); //Set the IP address of ESP32 itself
IPAddress gateway(192,168,1,10);   //Set the gateway of ESP32 itself
IPAddress subnet(255,255,255,0);  //Set the subnet mask for ESP32 itself

//컴퓨터에서 ESP32를 공유기(AP)로 인식하는 것을 볼 수 있음
//암호 12345678 을 입력하면 공유기에 연결이 되었다고 나오는데 "인터넷 없음, 보안" 이라고 현재 상태를 표시해 줌
//연결된 컴퓨터에서 ipconfig로 확인해 보면 ip를 받아옴

void setup(){
  Serial.begin(115200);
  delay(2000);
  Serial.println("Setting soft-AP configuration ... ");
  WiFi.disconnect();
  WiFi.mode(WIFI_AP);
  Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
```

```
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  }else{
    Serial.println("Failed!");
  }
  Serial.println("Setup End");
}

void loop() {
}
```

예제 4) - Station/AP Mode

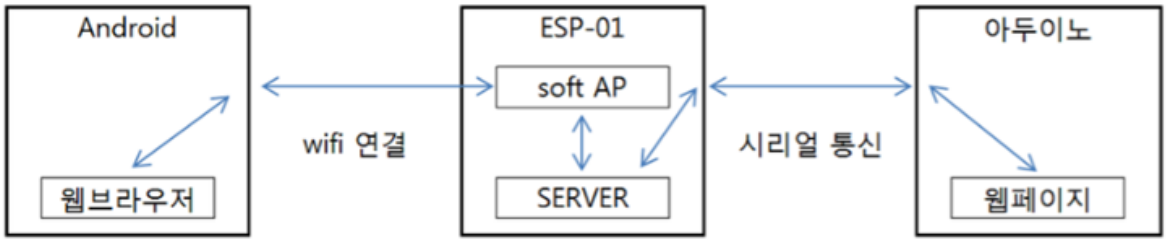


ESP-01 wifi 모듈을 이용한 아두이노 원격제어(스마트폰과 아두이노 간의 원격제어용 데이터 송신 및 수신)의 방법

- ESP-01(ESP8266 계열)은 TCP/UDP 스택이 이미 들어가 있기 때문에, TCP 통신이 가능함
- HTTP 통신을 하려면 TCP 프로토콜을 이용하여 HTTP 메세지를 보내주면 되는데, AT 명령어를 사용하면 됨

ESP-01 모듈을 soft AP로 설정(공유기 기능)하여 스마트폰의 wifi가 AP의 ip로 연결할 수 있게 하고 또한, server로 설정하여 soft AP를 통해 연결된 스마트폰이 server로 웹페이지를 요청할 수 있도록 해보자!!!

- 1) 만약 서버의 ip가 192.168.4.1로 설정되었을 경우, 스마트폰 웹브라우저 주소창에 http://192.168.4.1/on 입력하여 서버에 on 페이지에 대한 요청을 할 수 있게 됨
- 2) 이때 ESP-01 모듈은 들어온 웹페이지 요청 값을 시리얼 통신을 통해 아두이노에 전송하게 되고 아두이노는 수신된 웹페이지 요청 값을 코드에 따라서 원격제어에 필요한 on만을 분리해내고 그 값에 해당하는 코드를 실행시켜 아두이노 원격제어를 구현하게 됨
- 3) 만약 on에 대한 응답 코드가 아두이노에 설정되어 있다면 아두이노는 시리얼 통신을 통해 ESP-01 모듈에 웹페이지 요청에 대한 응답을 보내라고 명령하게 되고 ESP-01 모듈은 그 명령에 따라 스마트폰에 응답을 보내게 됨



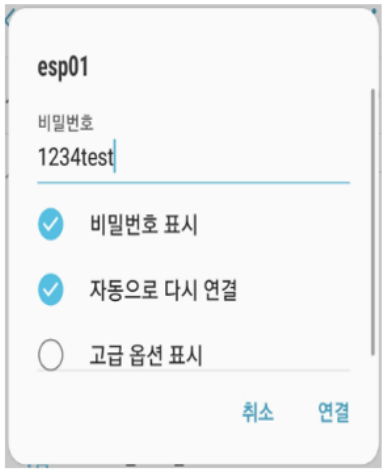
원격제어를 위해 ESP-01를 설정하는데 필요한 AT command

- 아두이노 시리얼 모니터 입력창에 붙여넣기 한 뒤 엔터를 쳐 ESP-01 모듈이 설정되는 것을 확인!!!
  - AT+RST // ESP-01 리셋
  - AT+CWMODE=2
    - 1 = Station mode (client), 2 = AP mode (host), 3 = AP + Station mode
  - AT+CWSAP="esp01", "1234test", 5, 3 // AT+CWSAP= softAP SSID, Password, channel id, ecn](https://)

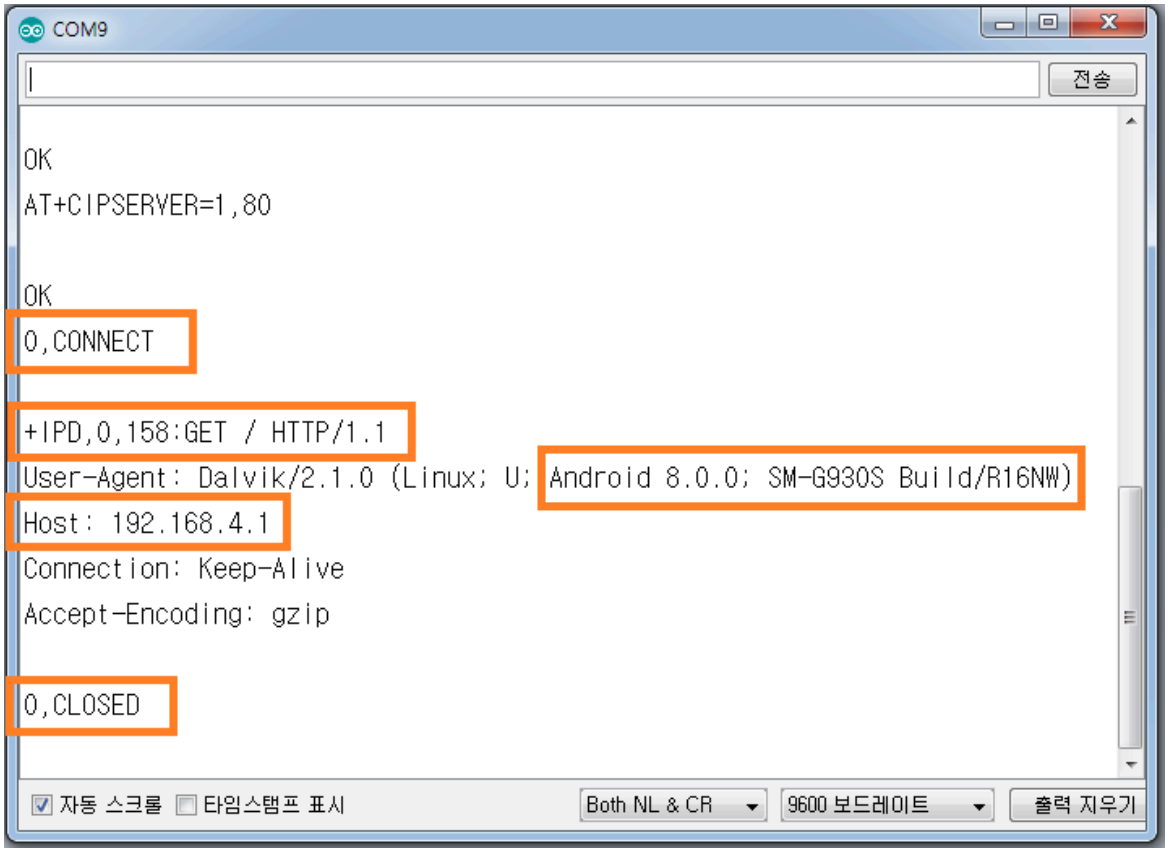
        - (보안설정) : 0 = OPEN, 2 = WPA_PSK, 3 = WPA2_PSK, 4 = WPA_WPA2_PSK
    - AT+CIFSR // AP IP 주소
    - AT+CIPMUX=1
        - 0: Single connection, 1: Multiple connections (0 ~ 4), 서버 설정 시 "1" 강제 사항
    - AT+CIPSERVER=1, 80
        - AT+CIPSERVER= mode, port
        - mode: 0: Delete server, 1: Create server, port number

---

스마트폰에서 wifi 장치를 켜게 되면 SSID "esp01"를 검색



- 스마트폰이 soft AP에 접속이 되면
  - 시리얼 모니터상에 connection id: 0이 연결됐다는 표시와 +IPD,0,459:GET /hi HTTP/1.1 수신 데이터가 표시되고
  - 접속장비와 soft AP의 ip 주소를 확인할 수 있고
  - 데이터 수신 후에 id 0번의 연결을 끊은걸 확인할 수 있음



- 데이터의 수신
  - +IPD,0,158:GET / HTTP/1.1 // +IPD, id, len:GET /data HTTP/1.1
    - id: id no. of connection, len: data length, data: data received
      - "+IPD,0,158:GET /" 이하가 수신된 데이터

- 수신 시에는 AT+CIPMUX=1 옵션 설정에 따라 멀티 커넥션 id 0부터 4까지 자동으로 할당되어 사용하게 되며 수신이 완료되면 자동으로 연결을 닫게 되는데 연결을 닫기 전에 수신되는 데이터는 다른 커넥션 id로 자동 할당 되어 수신되는 걸 테스트를 통해 확인할 수 있음

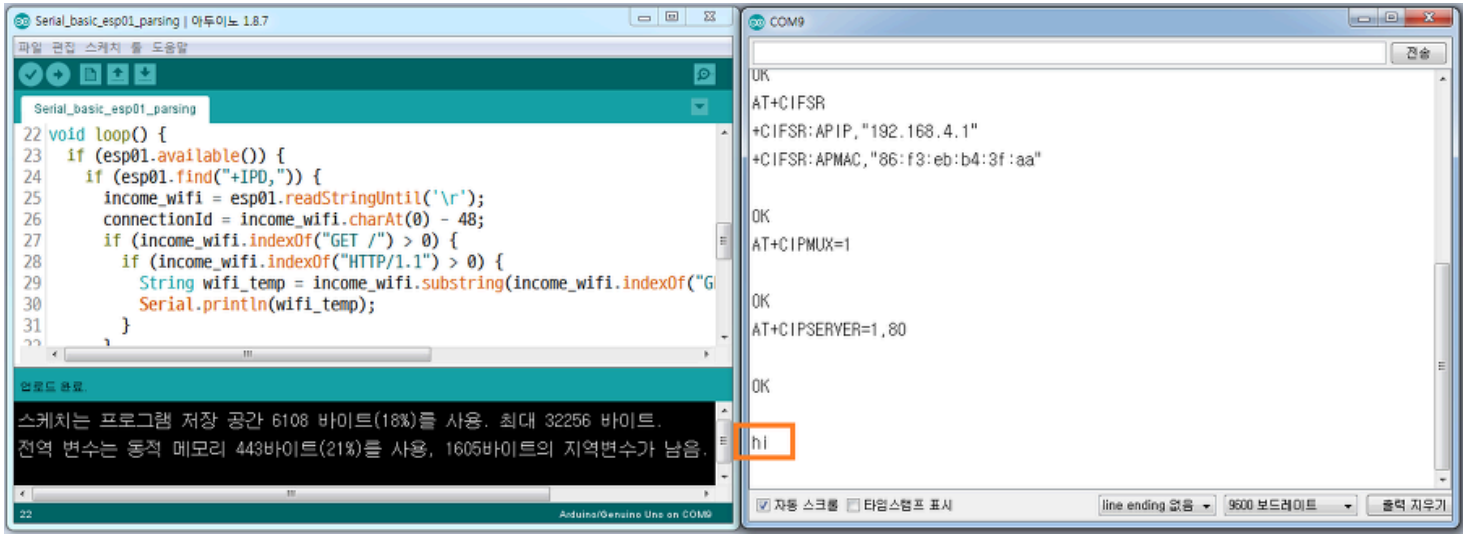
- 데이터의 전송
  - AT+CIPSEND=0,30
  - AT+CIPSEND=length // normal send (single connection).
  - AT+CIPSEND=id,length // normal send (multiple connection),
    - id: ID no. of connection, length: data length, MAX 048 bytes
  - AT+CIPCLOSE=0 // id: ID no. of connection to close,
    - Close TCP or UDP connection.For multiply connection mode
  - ESP-01을 통해 데이터 송신 시에는 아두이노에서 시리얼 통신을 통해 상기의 데이터 전송 AT command를 사용하여 전송 데이터와 함께 보내야만 데이터가 스마트폰으로 전송됨
  - 시리얼 통신으로 연결된다는 것 자체가 데이터의 전송 속도에 제한을 걸게 되는데, 이 상황에 더불어 모듈이 AT command를 수신하고 작동하는 시간이 소요되게 되어 아두이노에서 어떤 작동의 응답으로 ESP-01 모듈의 wifi를 통해 데이터 전송할 경우 약 1초 에서 2초의 시간이 걸리게 됨 - 짧은 시간 동안 연속되는 데이터의 전송하는 데 있어서 문제를 야기하게 된다고 할 수 있음

코드상에서 아두이노 실행 시 (명령어 대신)자동으로 ESP-01을 설정되도록 하는 코드

```
#Serial_basic_esp01_parsing.ino
#include <SoftwareSerial.h>
#define DEBUG true

String income_wifi = "";
SoftwareSerial esp01(2,3);

void setup() {
    Serial.begin(9600);
```

```
    esp01.begin(9600); // your esp's baud rate might be different
    sendData("AT+RSTWrWn",2000,DEBUG); //리셋
    sendData("AT+CWMODE=2WrWn",1000,DEBUG); // configure as access point (working mode: AP+STA)

    sendData("AT+CWSAP=W"ESP-01W",W"1234testW",11,3WrWn",1000,DEBUG); // join the access

    //wifi 연결 시 비밀번호 연결을 비활성화하고자 한다면 3을 0으로 변경해주면 된다.
    //sendData("AT+CWSAP=W"ESP-01W",W"1234testW",11,0WrWn",1000,DEBUG);

    sendData("AT+CIFSRWrWn",1000,DEBUG); // get ip address
    sendData("AT+CIPMUX=1WrWn",1000,DEBUG); // configure for multiple connections
    sendData("AT+CIPSERVER=1,80WrWn",1000,DEBUG); // turn on server on port 80
}

void loop() {
    if (esp01.available()) {
    //수신 데이터를 필요한 값만 추출해내는 parsing
    //parsing을 하기 위해서는 String class를 활용해야만 함
    //상기 AT command 설정 테스트에서 데이터 수신이 +IPD,0,158:GET / HTTP/1.1 형식으로 들어온 걸 확인했었음
    //만약 웹브라우저에서 http://192.168.4.1/hi라고 입력하면 +IPD,0,459:GET /hi HTTP/1.1 데이터가 수신될 것임
    //즉, 모든 데이터 수신에는 +IPD,0,158:GET / 형식의 코드가 선행됨
    //데이터 수신의 확인을 위해 "+IPD,"를 이용

        if (esp01.find("+IPD,")) {

        //수신 데이터를 스트링 타입 income_wifi 변수에 저장해 주면 "0,158:GET / HTTP/1.1"라는 스트링이 저장됨
        income_wifi = esp01.readStringUntil('Wr'); // 만약 시리얼 버퍼에 "+IPD, "가 있으면, "+IPD,"가 확인될 경우 "+IPD,"는 삭제됨

        //필요한 data는 "GET /" 이후에 들어오고 종료 문자 역할을 하는 "HTTP/1.1"의 앞 공백 문자에서 끝나게 됨
        //String class에서는 스트링 중의 일부를 떼어내어 저장할 수 있는 string.substring(start index, end index); 함수를 제공하고 있음
        //저장할 스트링의 시작 인덱스 번호와 종료 인덱스 번호를 필요로 함

        //이 함수를 이용하기 위해 "GET /"와 "HTTP/1.1"의 인덱스 번호를 알아내 보자.
        //income_wifi 스트링에서 "GET /"의 시작 인덱스 위치를 확인하는 코드는 income_wifi.indexOf("GET /")가 되고
        //"HTTP/1.1"의 시작 인덱스 위치를 확인하는 코드는 income_wifi.indexOf("HTTP/1.1")가 됨

        String wifi_temp = income_wifi.substring(income_wifi.indexOf("GET /")+5, income_wifi.indexOf("HTTP/1.1")-1);
        //income_wifi.indexOf("GET /")는 "GET /"의 시작 인덱스 번호를 반환함
        //그러므로 필요한 위치의 인덱스 번호는  "GET /"의 글자 수 5를(공백 문자 포함) 더한 위치 값이 되게 됨
        //income_wifi.indexOf("HTTP/1.1")는 "HTTP/1.1"의 시작 인덱스 번호를 반환함
        //앞에 공백 문자(space)가 있으므로 -1을 해준 위치 값이 필요로 하는 위치 값이 되고, 상기의 코드에서는 data가 없으므로 두 위치 값은 같은 값일 것임
        Serial.println(wifi_temp);
        }
    }
}

```

```
//전송 실행 함수 - sendData(문자열 명령, 응답에 필요한 시간, 시리얼 모니터 출력 플래그)
String sendData(String command, const int timeout, boolean debug) { // 스트링 타입 AT command 전송 함수
    String response = ""; // 스트링 타입 지역변수 선언 및 초기화
    esp01.print(command); // ESP-01 모듈에 스트링 타입의 AT command 전송
    long int time = millis();
    while( (time+timeout) > millis()) { //ESP-01 모듈의 명령 실행 후 응답에 필요한 시간이 경과되면
        while(esp01.available()) { // 시리얼 버퍼에 ESP-01 모듈의 응답을 있을 경우
            char c = esp01.read(); // 1byte 데이터를 char타입 변수 c에 저장
            response+=c; // c에 저장된 문자를 문자열로 변경
        }
    }
    if(debug) Serial.print(response); // debug가 참이면 시리얼 모니터에 출력
    return response;
}
```



- 스마트폰 웹브라우저 주소창에서 192.168.4.1/hi를 입력하고 연결을 클릭하면 시리얼 모니터에 hi가 표시된 것을 볼 수 있음
    - AT command 설정 시에는 스마트폰 접속 시 관련 정보들이 시리얼 모니터에 출력되었지만 지금은 data를 제외한 모든 값은 출력이 안됨

추출된 스트링 데이터를 이용하여 LED의 제어

```
#Serial_basic_esp01_string_LED.ino

#include <SoftwareSerial.h>
#define DEBUG true
#define ledPin 13

String income_wifi = "";
SoftwareSerial esp01(2, 3);

void setup() {
    Serial.begin(9600);
    esp01.begin(9600); // your esp's baud rate might be different
    pinMode(ledPin, OUTPUT);
    sendData("AT+RSTWrWn", 2000, DEBUG); // reset module
    sendData("AT+CWMODE=2WrWn", 1000, DEBUG); // configure as access point (working mode: AP+STA)
    sendData("AT+CWSAP=W"ESP-01W",W"1234testW",11,0WrWn",1000,DEBUG); // join the access point
    sendData("AT+CIFSRWrWn", 1000, DEBUG); // get ip address
    sendData("AT+CIPMUX=1WrWn", 1000, DEBUG); // configure for multiple connections
    sendData("AT+CIPSERVER=1,80WrWn", 1000, DEBUG); // turn on server on port 80
}

void loop() {
    if (esp01.available()) {
        if (esp01.find("+IPD,")) {
            income_wifi = esp01.readStringUntil('Wr');
            String wifi_temp = income_wifi.substring(income_wifi.indexOf("GET /")+5, income_wifi.indexOf("HTTP/1.1")-1);
            if(wifi_temp == "on"){ // wifi_temp는 지역변수로서 "if (esp01.find("+IPD,")){} 함수를 나가면 초기화됨.
                digitalWrite(ledPin, HIGH);
            }
            else if(wifi_temp == "off"){
                digitalWrite(ledPin, LOW);
                }
                else {
                    Serial.println(wifi_temp);
            }
        }
     }
}

String sendData(String command, const int timeout, boolean debug) {
    String response = "";
    esp01.print(command); // send the read character to the esp01
```

```
    long int time = millis();
    while( (time+timeout) > millis()) {
        while(esp01.available()) { // The esp has data so display its output to the serial window
            char c = esp01.read(); // read the next character.
            response+=c;
        }
    }
    if(debug) Serial.print(response);
    return response;
}
```

안드로이드 앱을 이용하여 원격제어

<<<참조자료 사이트>>>