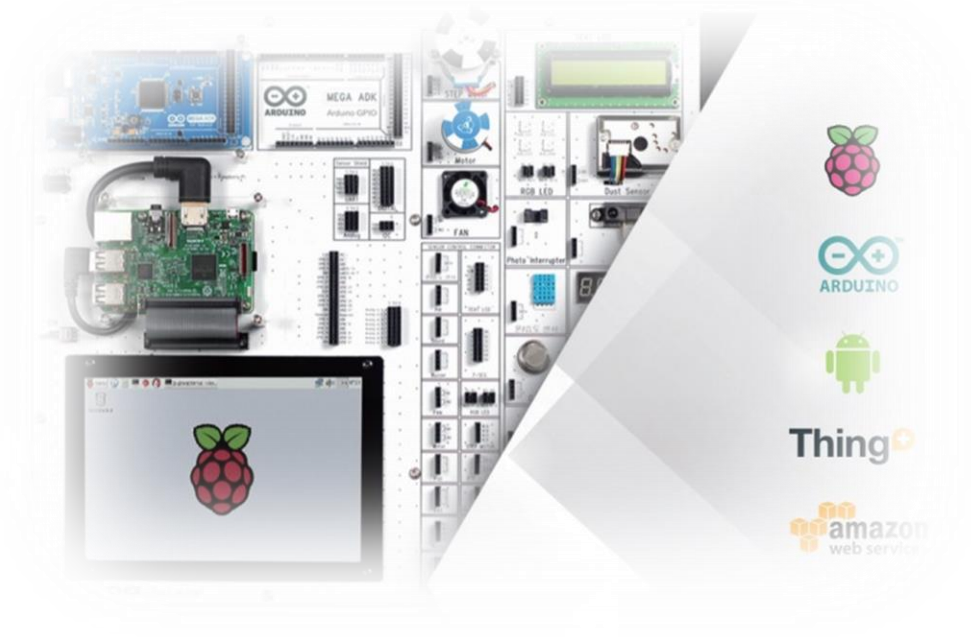
2021년도 2학기
휴먼스마트기기설계



교과목명 : 휴먼스마트기기설계 (01)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

교재명 : 스마트기기 개발을 위한 사물인터넷, 이수형. (LINC+ 사업단 배포)

3부 : 네트워크

6. 무선인터넷 기초 #1

네트워크

- 네트워크 (network, 통신망)
  - Net + Work : 그물망 - 컴퓨터들이 통신 기술을 이용하여 연결된 통신 방식
  - 네트워크 구성 요소 : 기기 ↔ 통신선로 ↔ 기기
- 규모에 따른 네트워크의 종류
  - PAN(Personal) < LAN(Local) < MAN(metropolitian) < WAN (Wide)
    AN (Area Network)
  - 간단한 구분 : LAN < WAN

인터넷

- 인터넷 (Internet)
  - 컴퓨터로 연결하는 TCP/IP 통신프로토콜을 이용해 정보를 주고 받는 컴퓨터 네트워크
  - 1960~1970년도 군사적인 목적으로 시작 : 1969년 ALPANET
  - 1973년 : International Network / Internetwork ⇒ Internet

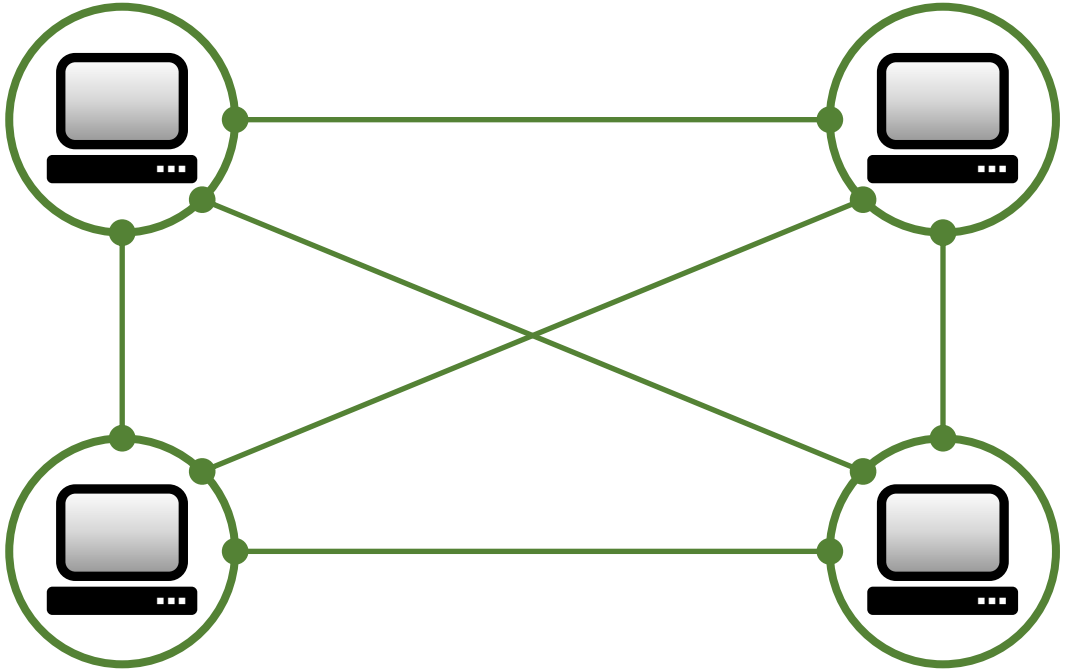
  - 참고 : 인터넷은 어떻게 동작하는가?
    - https://developer.mozilla.org/ko/docs/Learn/Common_questions/How_does_the_Internet_work

인터넷 연결 구조 #1

- 두 개의 컴퓨터의 통신
  - 통신선 연결이 필요

- 여러 개의 컴퓨터 통신
  - 4대 : 6개의 연결 필요
  - 5대 : 10개의 연결 필요
  - 10대 : 45개의 연결 필요
  - n대 : $C(n,2) = \binom{n}{2}$개 필요

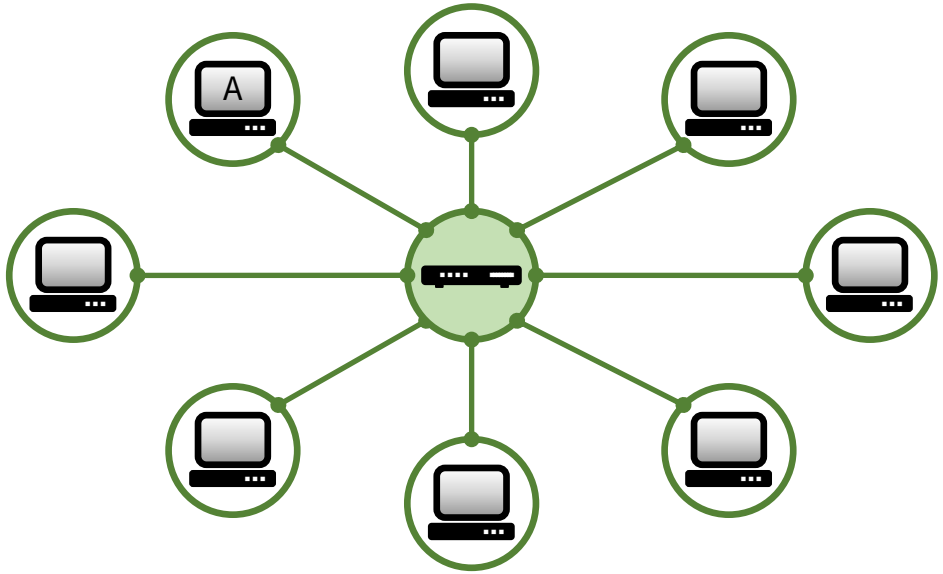
⇒ 연결을 줄이려면?



참고 : https://developer.mozilla.org/ko/docs/Learn/Common_questions/How_does_the_Internet_work

인터넷 연결 구조 #2

- 허브(hub)/라우터(router)를 이용
  - 연결의 수가 감소 (8대 : 28 → 8)
    - ➢ 허브 : 컴퓨터끼리의 통신을 연결 (무작정 연결)
    - ➢ 스위치 : 효율적인 통신 (스위칭 허브)
    - ➢ 라우터 : 내부 통신 (LAN) 및 외부 연결
    - ➢ 공유기 : NAT에 특화된 라우터

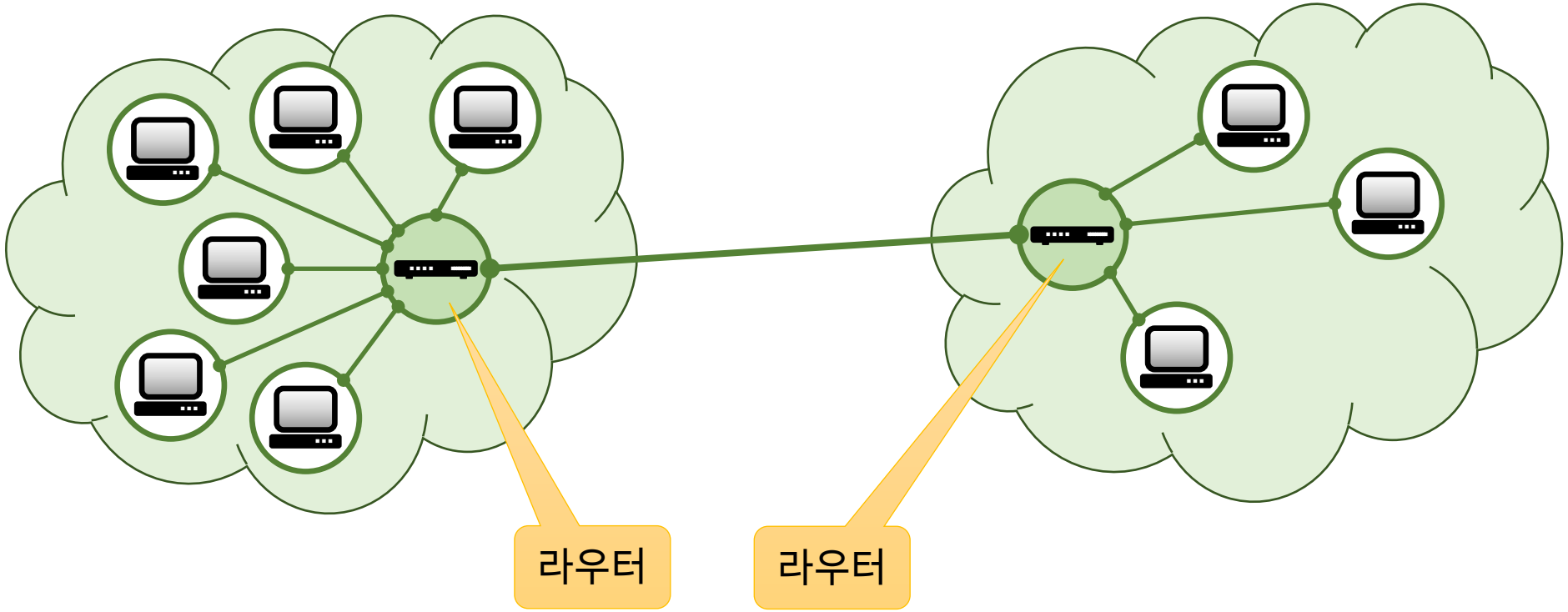


- 기기들의 주소 지정
  - MAC (Media Access Control) Address
  - 네트워크 어댑터(랜카드)의 물리적인 주소 : 16진수 6자리로 표현
    예] 00-11-22-11-AC-EF
  - 스위치에서 통신하는 경우 Mac Address를 토대로 전송
    vs. 허브에서는 연결된 모든 어댑터에 전송

참고 : https://developer.mozilla.org/ko/docs/Learn/Common_questions/How_does_the_Internet_work

인터넷 연결 구조 #2

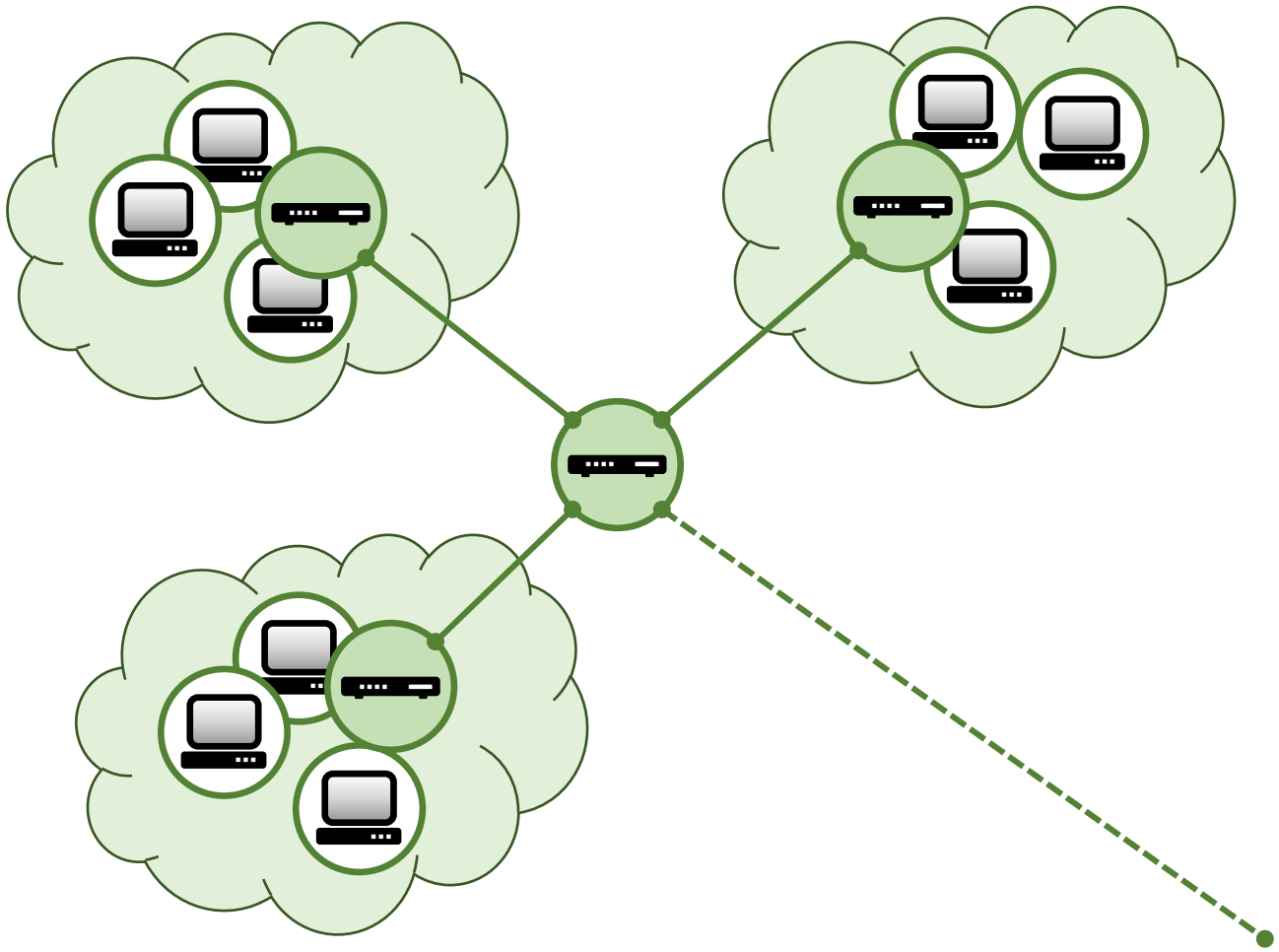
- 라우터끼리 연결 → 확장



참고 : https://developer.mozilla.org/ko/docs/Learn/Common_questions/How_does_the_Internet_work

인터넷 연결 구조 #3

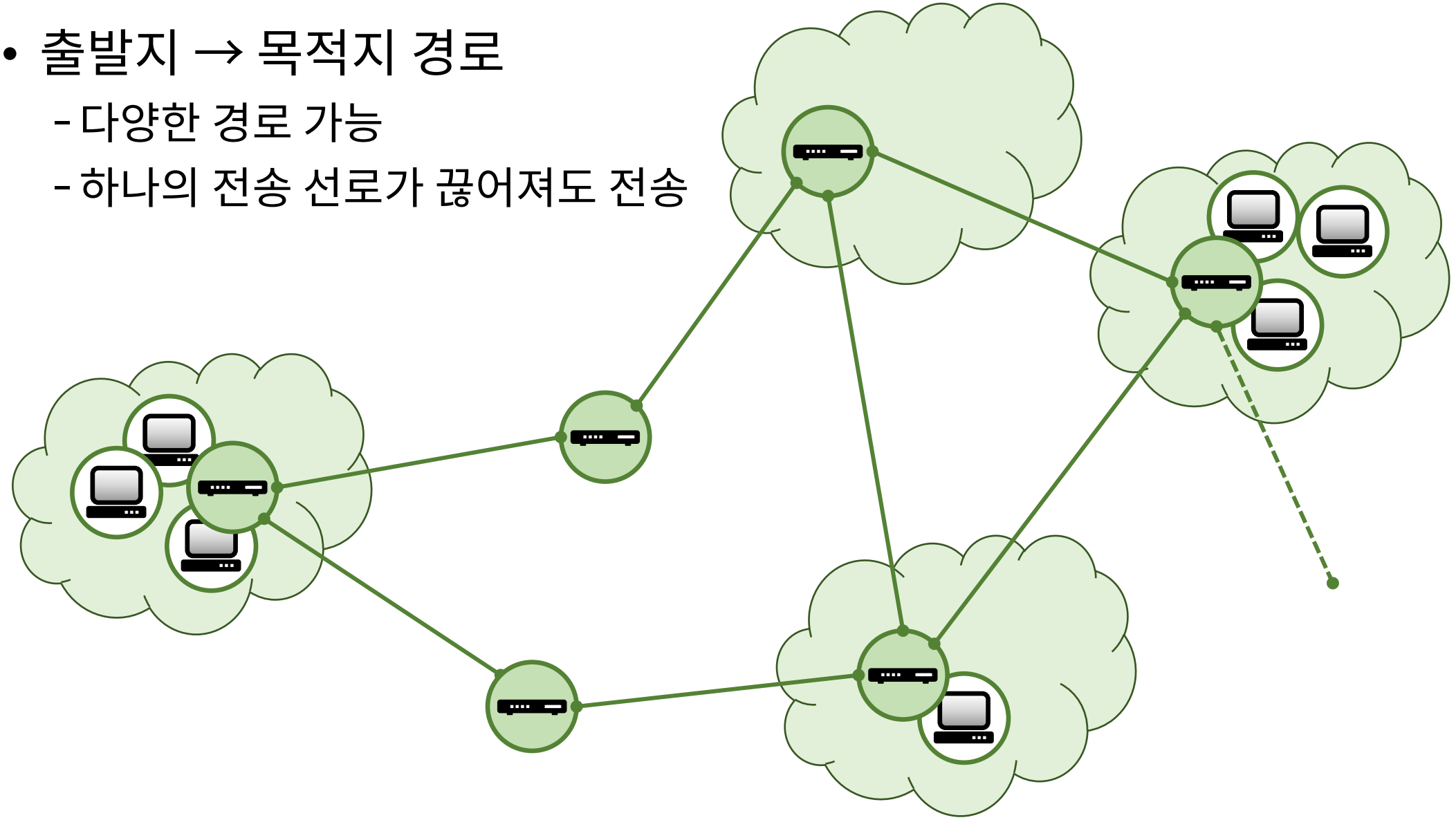
- 라우터들의 라우터
  - 무한히 확장 가능



참고 : https://developer.mozilla.org/ko/docs/Learn/Common_questions/How_does_the_Internet_work

인터넷 연결 구조 #4

- 출발지 → 목적지 경로
  - 다양한 경로 가능
  - 하나의 전송 선로가 끊어져도 전송



OSI 7 Layers

- OSI 7 Layers
  - Open Systems Interconnection Reference Model
  - 국제표준화기구(ISO, International Standard Organization) 1984년 발표
  - 개방화된 데이터 통신 환경에서의 표준 통신 모델

- 비교] TCP/IP Layers
  - TCP/IP : 현재 인터넷의 통신규약
  - 4개의 계층으로 이루어짐

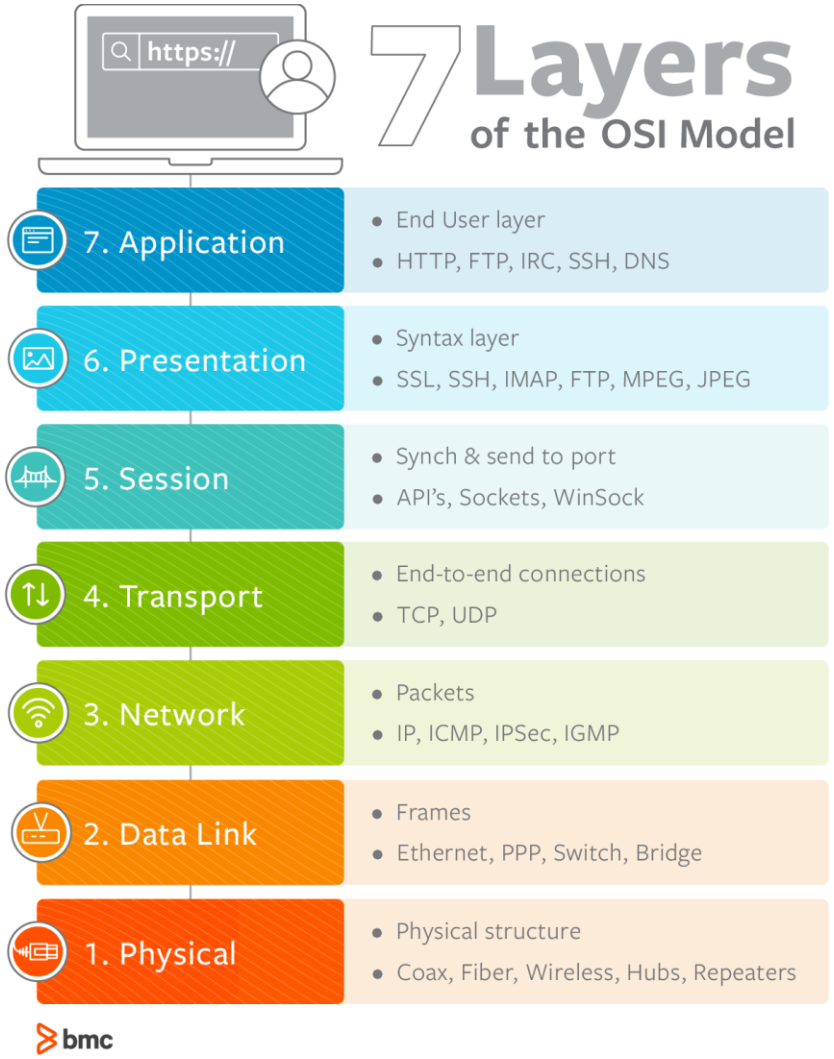
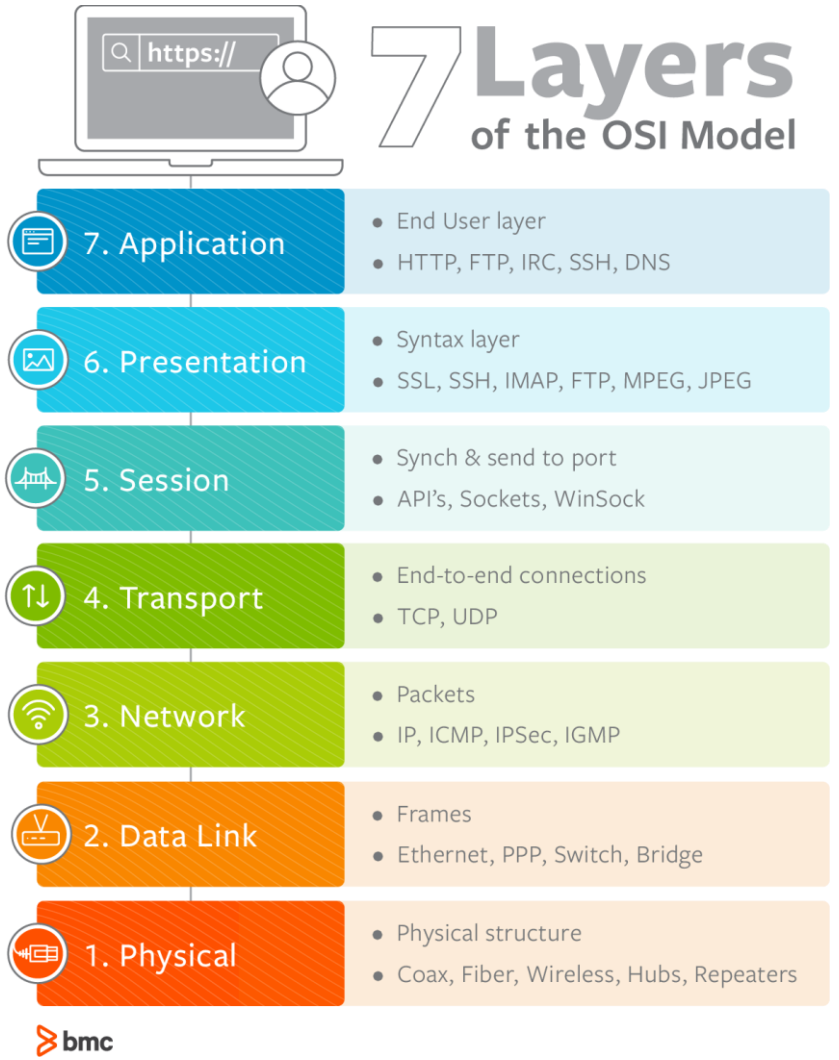


그림 출처 : https://velog.io/@devzunky/TIL-no.89-Network-The-7-Layers-of-the-OSI-Model

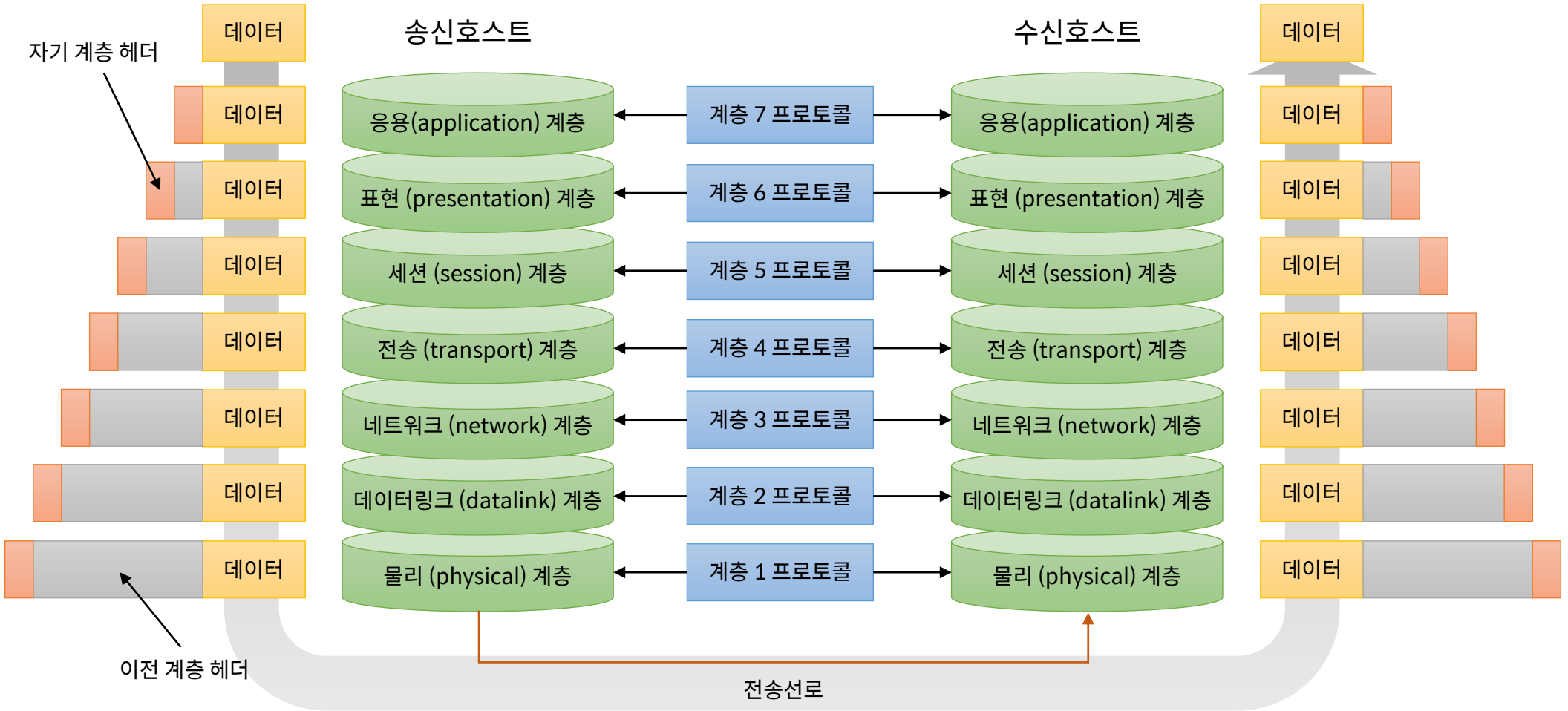
OSI 7 Layers

- 응용 계층 (Application layer)
  - 응용프로그램의 프로세스와 직접 관계하여 일반적인 응용 서비스 수행
  - 프로토콜 : DNS, FTP, HTTP
- 표현 계층 (Presentation layer)
  - 응용 계층 → 세션 계층으로 데이터 전달시에 적당한 형태로 변환, 코드 변환, 암호화, 압축
  - 프로토콜 : JPG, MPEG, SMB
- 세션 계층 (Session layer)
  - 양 끝단의 프로세스가 통신을 관리하기 위한 방법 제공
  - 프로토콜 : NetBIOS, SSH, TLS
- 전송 계층 (Transport layer)
  - 전체 메시지의 종단-종단(end to end)간 제어와 오류 관리, 패킷 전송 유효성 검사 (패킷 재전송)
  - 전송단위 : 세그먼트(segment), 프로토콜 : TCP, UDP 등, 장비 : 게이트웨이, L4 스위치
- 네트워크 계층 (Network layer) : 발신지에서 목적지까지 성공적으로 전달하는 기능
  - 전송단위 : 패킷(packet), 프로토콜 : IP, ICMP 등, 장비 : 라우터, L3 스위치
- 데이터 링크 계층 (Data link layer)
  - Point to Point 간 신뢰성 있는 전송 보장 (오류 제어, 흐름 제어)
  - 전송단위 : 프레임, 프로토콜 : 이더넷, MAC, PPP, 장비 : 브릿지, 스위치
- 물리 계층 (Physical layer)
  - 기본 네트워크의 하드웨어 전송 기술
  - 전송단위 : 비트(bit), 프로토콜 : RS-232 등, 장비 : 허브, 리피터



참고 : https://madplay.github.io/post/network-osi-7-layer

OSI 7 layer의 통신 구조



참고 : https://madplay.github.io/post/network-osi-7-layer

TCP/IP

- TCP/IP
  - 인터넷 통신용 프로토콜 중에서 가장 많이 사용되는 TCP 와 IP를 나타냄
  - IP (Internet Protocol) : 패킷 통신 방식의 프로토콜
  - TCP(Transmission Control Protocol) : 연결 지향(유지)의 통신 프로토콜
  - 패킷 통신 방식의 프로토콜 IP + 전동 제어 프로토콜 TCP

<table>
<tr><th>OSI Model</th><th>TCP/IP Model</th><th>Protocol</th></tr>
<tr><td>Application Layer</td><td rowspan="3">Application Layer</td><td rowspan="3">HTTP, SMTP, FTP,
TELNET, SNMP, POP3,
DNS</td></tr>
<tr><td>Presentation Layer</td></tr>
<tr><td>Session Layer</td></tr>
<tr><td>Transport Layer</td><td>Transport Layer</td><td>TCP, UDP</td></tr>
<tr><td>Network Layer</td><td>Internet layer</td><td>IP, ICMP, ARP</td></tr>
<tr><td>Data link Layer</td><td rowspan="2">Link Layer</td><td rowspan="2">MAC</td></tr>
<tr><td>Physical Layer</td></tr>
</table>

그림출처 : http://mhshohag.com/compare-and-contrast-osi-and-tcp-ip-models/

TCP/IP

- IPv4의 패킷 구조

VER 4 bits	HLEN 4 bits	Service Type 8 bits	Total Length 16 bits
Identification 16 bits		Flags 3 bits	Fragmentation offset 13 bits
TTL(Time to live) 8 bits	Protocol 8 bits	Header checksum 16 bits	
Source IP Address 4 bytes			
Destination IP Address 4 bytes			
Option (+Padding) 0~40 bytes			
Data 0~65516 bits			

- 인터넷상에 연결되어 있는 두 개의 기기(출발지→목적지)의 통신을 수행하는 프로토콜
- 시작/목적지 IP 주소 + 추가 정보 + 데이터

TCP 헤더 구조

Source Port 16 bits									Destination Port 16 bits
Sequence Number 32 bits									
Acknowledgement Number 32 bits									
Header Len. 4bits	Reserve 4 bits	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size 16 bits
Checksum 16 bits									Urgent Point 16 bits
Option (+Padding) 0~40 bytes									
Data									

- 출발지에서 목적지까지 전송되었을때 연결을 유지하면서 데이터를 전송하기 위한 프로토콜
- 시작/목적지 포트 번호 + 연결 번호 + 응답 번호 + 데이터

그림출처 : http://mhshohag.com/compare-and-contrast-osi-and-tcp-ip-models/

무선인터넷(WiFi) 규격

- 무선인터넷 규격 (IEEE 802.11)
  - 유선 LAN의 단점을 보완하기 위한 무선 근거리 통신망을 위한 무선 네트워크 기술

Wi-Fi(무선 LAN) 규격	책정 시기	최대 통신 속도	주파수대
IEEE 802.11a	1999년 10월	54Mbps	5GHz
IEEE 802.11b	1999년 10월	11Mbps	2.4GHz
IEEE 802.11g	2003년 6월	54Mbps	2.4GHz
IEEE 802.11n	2009년 9월	600Mbps	2.4GHz/5GHz
IEEE 802.11ac	2013년 12월	6.9Gbps	5GHz

출처 : https://www2.elecom.co.jp.k.gj.hp.transer.com/network/wireless-lan/column/wifi_column/00061/

6.1 NodeMCU의 무선인터넷 기능

- NodeMCU의 무선네트워크 기능
  - TCP/IP의 전체 기능 지원
  - 웹 서버와 같은 서버의 기능 및 클라이언트로 동작 가능
  - 외부 센서 모듈 및 다양한 출력장치 제어 가능

  ⇒ 사물인터넷 기기 구성
- 동작 모드
  - Station 모드
  - AP(Access point) 모드
  - Station + AP 모드

ESP8266의 무선 인터넷 모드

- Station Mode
  - 보통의 모바일 기기처럼 동작하는 모드
  - 공유기 등의 무선인터넷 장치를 통해서 인터넷에 접속
  - 기본 모드

그림 출처 : https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

ESP8266의 무선 인터넷 모드

- Soft AP(Access Point) Mode
  - 공유기와 같이 무선 네트워크에 연결해주는 장치
  - 스마트폰을 AP에 연결할 수 있음
  - 단, AP 단독 모드로는 인터넷 연결이 불가능

참고 : AP - 무선장치들을 유선장치에 연결할 수 있게 하는 장치

그림 출처 : https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

ESP8266의 무선 인터넷 모드

- AP + Station Mode
  - AP 모드와 Station모드를 동시에 사용하는 기능

그림 출처 : https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

WIFI 라이브러리

- ESP8266 Arduino Core Project
  - 아두이노 IDE에서 아두이노 스타일로 ESP8266용 프로그램 작성이 가능하게 하는 라이브러리 모음
  - WiFi 라이브러리 클래스 목록

B
BufferDataSource
BufferedStreamDataSource

C
ClientContext

D
DataSource

E
ESP8266WiFiAPClass
ESP8266WiFiClass
ESP8266WiFiGenericClass
ESP8266WiFiMulti
ESP8266WiFiScanClass
ESP8266WiFiSTAClass

P
ProgmemStream

S
SList
SSLContext

U
UdpContext

W
WifiAPlist_t
WiFiClient
WiFiClientSecure
WiFiEventHandlerOpaque
WiFiEventModeChange
WiFiEventSoftAPModeProbeRequestReceived
WiFiEventSoftAPModeStationConnected
WiFiEventSoftAPModeStationDisconnected
WiFiEventStationModeAuthModeChanged
WiFiEventStationModeConnected
WiFiEventStationModeDisconnected
WiFiEventStationModeGotIP
WiFiServer
WiFiUDP

  - 많이 사용하는 클래스 : WiFiClient(일반적인 클라이언트), WiFiServer(서버용)
  - https://arduino-esp8266.readthedocs.io/en/latest/index.html

예제 6-1. 무선인터넷 연결

- 하드웨어 연결
  - NodeMCU 보드만 사용

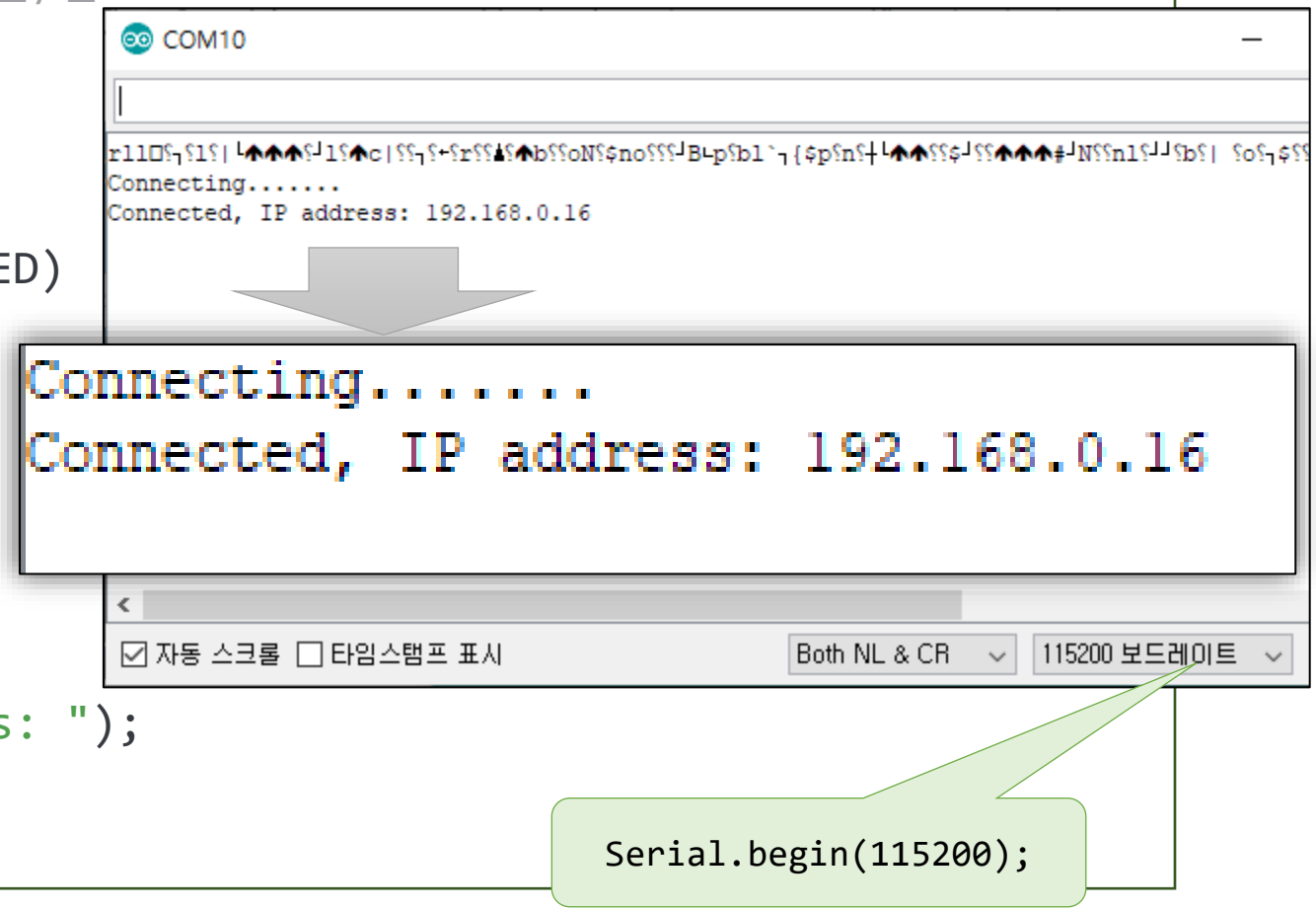
예제 6-1. NodeMCU 보드의 무선인터넷 연결

```
// 무선인터넷 연결
// 필요한 라이브러리 헤더 포함
#include <ESP8266WiFi.h>

void setup()
{
  // 시리얼 통신 초기화
  // 아두이노에서는 9600bps를 사용하였으나 ESP8266에서는 115200bps를 많이 사용한다.
  Serial.begin(115200);
  Serial.println();
```

예제 6-1. 무선인터넷 연결

예제 6-1. NodeMCU 보드의 무선인터넷 연결

```
  // 무선인터넷(Wifi) 시작 : 네트워크 이름/암호
  WiFi.begin("UIDUK_CLASS", "");

  // 네트워크(공유기)에 연결 시도
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  // 연결된 후에 할당받은 IP 주소 출력
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
}
```



예제 6-1. 무선인터넷 연결

- 무선인터넷 연결
  - 초기에 연결 : setup() 함수 내부에서 연결

```
// 무선인터넷(Wifi) 시작 : 네트워크 이름/암호
WiFi.begin("UIDUK_CLASS", "");
```

  - 접속과정 : 시간이 걸리므로, 주기적으로 WiFi.status() 함수로 검사하면서 연결되기 전까지 반복

```
// 네트워크(공유기)에 연결 시도
Serial.print("Connecting");
while (WiFi.status() != WL_CONNECTED)
{
  delay(500);
  Serial.print(".");
}
Serial.println();
```

- 0 : **WL_IDLE_STATUS**
  - Wi-Fi가 다른 상태들 사이에 있는 경우
- 1 : **WL_NO_SSID_AVAIL**
  - 주어진 네트워크(공유기) 이름이 없는 경우
- 3 : **WL_CONNECTED**
  - 연결이 성공한 경우
- 4 : **WL_CONNECT_FAILED**
  - 비밀번호가 틀린 경우
- 6 : **WL_DISCONNECTED**
  - 기기가 Station 모드로 지정되어 있지 않는 경우

예제 6-1. 무선인터넷 연결

예제 6-1. NodeMCU 보드의 무선인터넷 연결

```
void loop() {
  Serial.println();
  WiFi.printDiag(Serial);
  delay(2000);
}
```

- 초기 연결 후 loop()함수에서 무선 인터넷 환경 진단
  - WiFi.printDiag(Serial);

⇒ WiFi진단 ⇒

```
Mode: STA+AP
PHY mode: N
Channel: 1
AP id: 0
Status: 5
Auto connect: 1
SSID (11): UIDUK_CLASS
Passphrase (0):
BSSID set: 0
```

Station + Soft AP 모드

IEEE 802.11n

6.2 간단한 웹 서버 구현

- 서버-클라이언트 모델
  - 인터넷 통신에서 대부분의 기기들이 통신하는 방식
  - 서버(server) : 서비스를 지원해주는 기기 – 고정된 IP 주소가 필요
  - 클라이언트(client) : 서버에 접속해서 서비스를 받는 고객

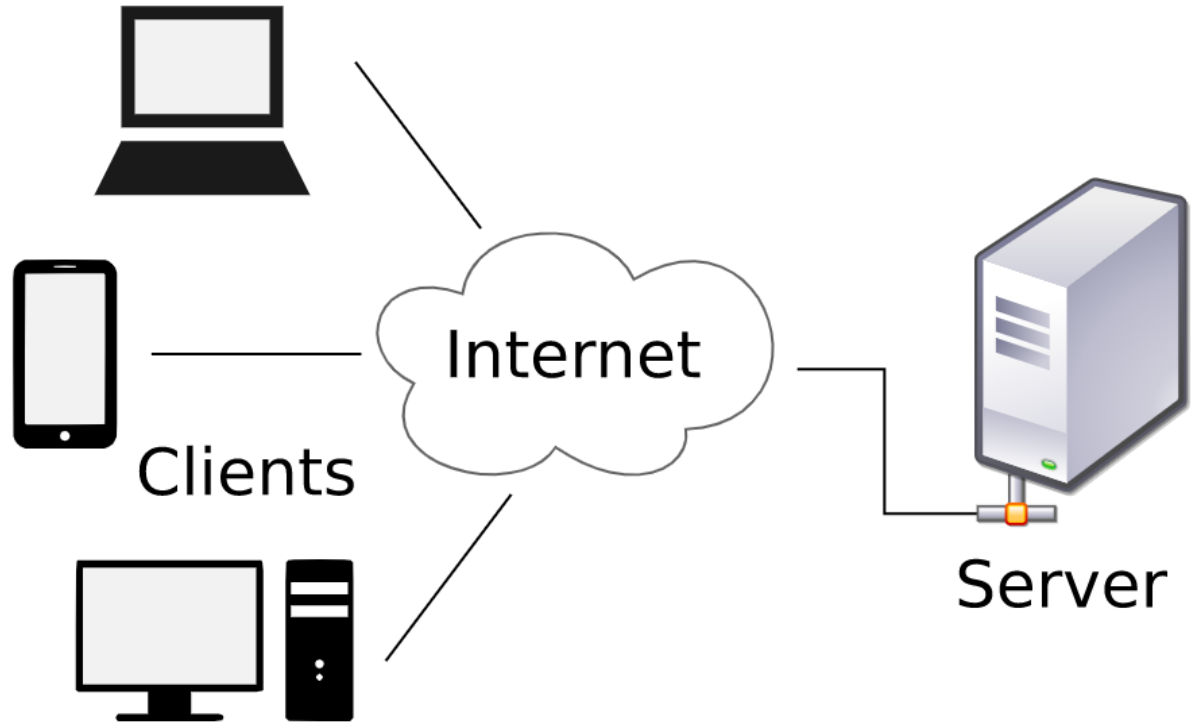


그림 출처 : https://ko.wikipedia.org/wiki/클라이언트_서버_모델

서버

- 서버의 특징
  - 일반 클라이언트에서 접속이 가능해야 함 ⇒ 인터넷 상에서 고정된 IP 주소 필요
  - 일반적으로 각 서비스 마다 정해진 포트 번호가 존재함
    - 포트(port: 항구) : 각 컴퓨터에서 네트워크를 통해 통신하는 경우의 접속 통로
    - 네트워크 통신을 하는 경우 여러 개의 어플리케이션에서 여러 개의 통신을 동시에 수행할 수 있으며 이때 각각의 통신 내용은 고유의 포트를 통해서 이루어짐
    - 널리 알려진 서비스들은 포트 번호가 정해져 있음 (변경 가능)

서버

- 인터넷 서비스 종류
  - SSH : 암호화된 통신을 이용하는 터미널 서비스 (포트 22)
  - FTP (File Transfer Protocol) : 파일 전송 서비스 (포트 21)
  - Email : 전자메일 서비스 (POP3, IMAP, SMTP)등의 서비스가 있음
    - SMTP (Simple Mail Transfer Protocol) : 전자 메일 송신 서비스 (포트 25, 465, 587…)
    - POP3 (Post Office Protocol) : 전자 메일 수신 서비스 (포트 110)
  - HTTP (Hyper Text Transfer Protocol) : 웹 서비스 (포트 80)
    - HTTPS (Secured HTTP) : 암호화된 웹통신 서비스 (포트 443)

IP 주소 체계

- IP 주소
  - TCP/IP 네트워크의 하나의 기기(컴퓨터, 라우터 등)를 고유하게 식별하는 숫자
  - IPv4 : 32비트 – 8bit 단위로 끊어서 표시, 예: 192.168.10.34
    - ➢ 8 bit : 0000 0000 ~ 1111 1111 = 0 ~ 255
  - IP 주소 할당 : InterNIC 에서 관리

클래스	주소범위	설명
A 클래스	1.0.0.1 ~ 126.255.255.254	2진수 0으로 시작
B 클래스	128.0.0.1 ~ 191.255.255.254	2진수 10으로 시작
C 클래스	192.0.0.1 ~ 223.255.255.254	2진수 110으로 시작
D 클래스	224.0.0.0 ~ 239.255.255.255	멀티캐스트용도로 예약됨
E 클래스	240.0.0.0 ~ 254.255.255.254	예약됨

- ➢ 위 범위에 들어가지 않는 주소들은 특수 용도로 사용됨
  - ✓ 예] 127.0.0.1 → loopback(자신), 192.168.xxx.xxx → 사설 IP 등

IP 주소 체계

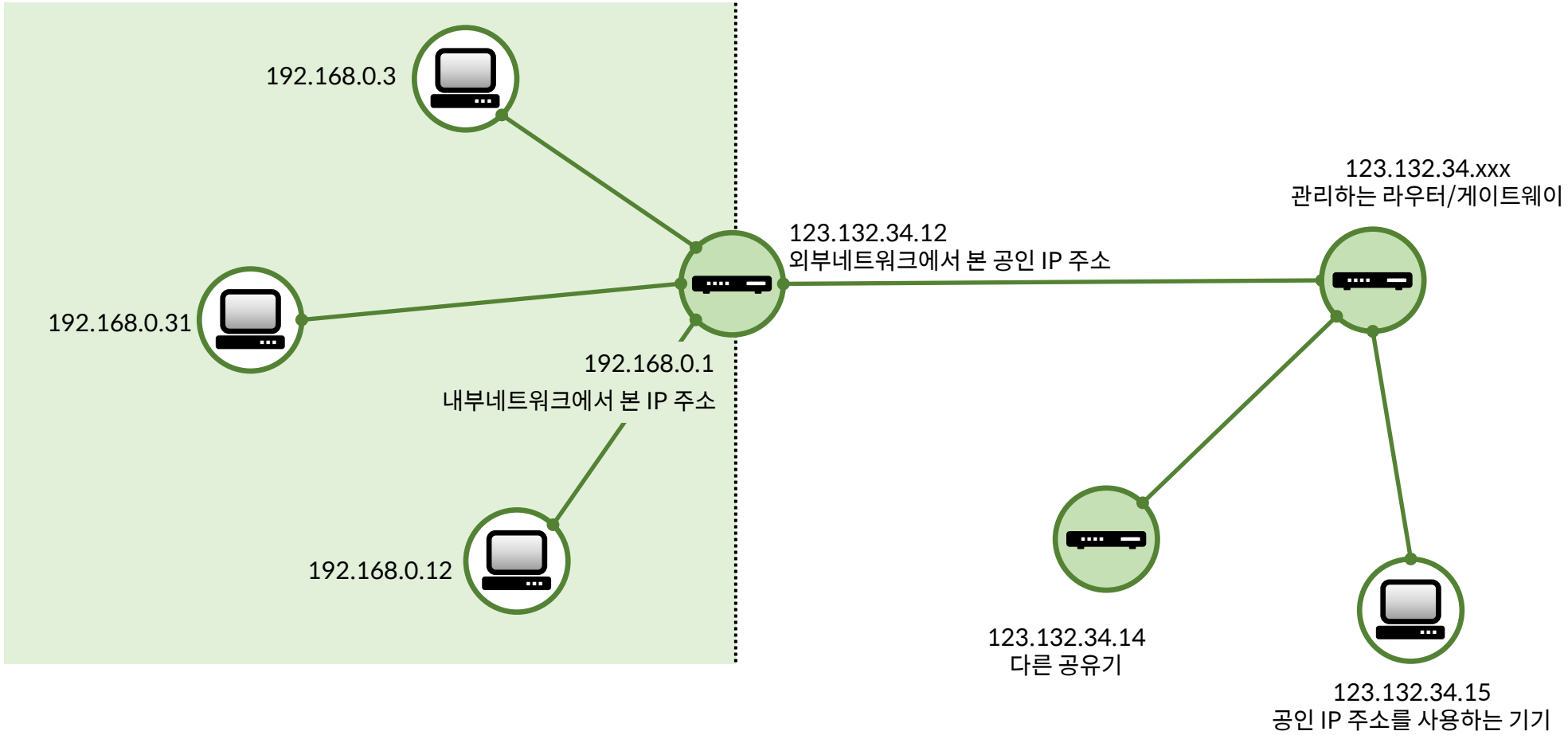
- 고정 IP vs 유동 IP
  - 고정 IP : 특정 기기에 IP 주소가 정해진 경우
    - ➢ 기기가 많아지면 주소가 부족해짐
  - 유동 IP : 여러 개의 IP 주소를 여러 개의 기기가 나누어서 사용
    - ➢ 모든 컴퓨터가 켜져 있지 않다는 가정에서 출발함.
      예] 100개의 주소를 150대가 돌아가면서 사용
    - ➢ ISP(Internet Service Provider, 인터넷 제공업체) 에서 주로 사용
    - ➢ IP 할당 (DHCP)
      - ✓ Dynamic Host Configuration Protocol 서비스
      - ✓ 공유기에서 가상 IP 주소를 할당하여 사용하는 경우에도 사용
        - NAT (Network Address Translation) : IP 패킷의 주소/포트를 재기록(변형)하여 통신하는 기술
        - 공유기를 통하여 나가는 경우에는 공유기의 주소를 사용 ⇒ IP 주소 공유가 가능하게 함

IP 주소 체계

- 공인 IP vs 사설 IP
  - 공인 IP : 인터넷 상에서 고유하게 부여된 (공인된) IP 주소
    - ➢ 전 세계에서 유일한 주소, 어느 곳에서나 접근이 가능함
    - ➢ 32비트의 체계로는 전 세계의 모든 기기가 사용하기에 주소의 숫자가 부족함
  - 사설 IP : 내부적으로 사용하는 IP 주소 (= 가상 IP)
    - ➢ 사용 주소 : 192.168.xxx.xxx, 172.16.xxx.xxx ~ 172.31.xxx.xxx, 10.xxx.xxx.xxx
    - ➢ 공유기에서 주로 사용 : 주로 하나의 대표 공인 IP를 여러 개의 기기가 공유하여 사용
    - ➢ IPv4의 주소 부족 문제 해결
      - ✓ 하나의 대표 공인 IP를 여러 개의 기기가 공유
    - ➢ 보안 문제가 해결
      - ✓ 외부에서 접근이 불가능

IP 주소 체계

- 공유기, 가상(사설) IP



예제 6-2. 간단한 웹 서버 구현 #1

예제 6-2. 간단한 웹 서버 구현

```
// 서버 구동 시험
#include <ESP8266WiFi.h>

// 무선네트워크 접속용 이름/비밀번호
#define SSID "UIDUK_CLASS"
#define PASS ""

// 서버용 클래스 객체 (웹서버에 사용되는 80번 포트 지정)
WiFiServer server(80);
```

예제 6-2. 간단한 웹 서버 구현 #2

```
void setup()
{
  // 시리얼 통신 초기화
  Serial.begin(115200);
  Serial.println();

  // 무선인터넷(Wifi) 시작 : 네트워크 이름/암호
  WiFi.begin(SSID, PASS);

  // 네트워크(공유기)에 연결 시도
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  // 할당받은 IP 주소 출력
  Serial.println("Connected");
  Serial.println(WiFi.localIP());
```

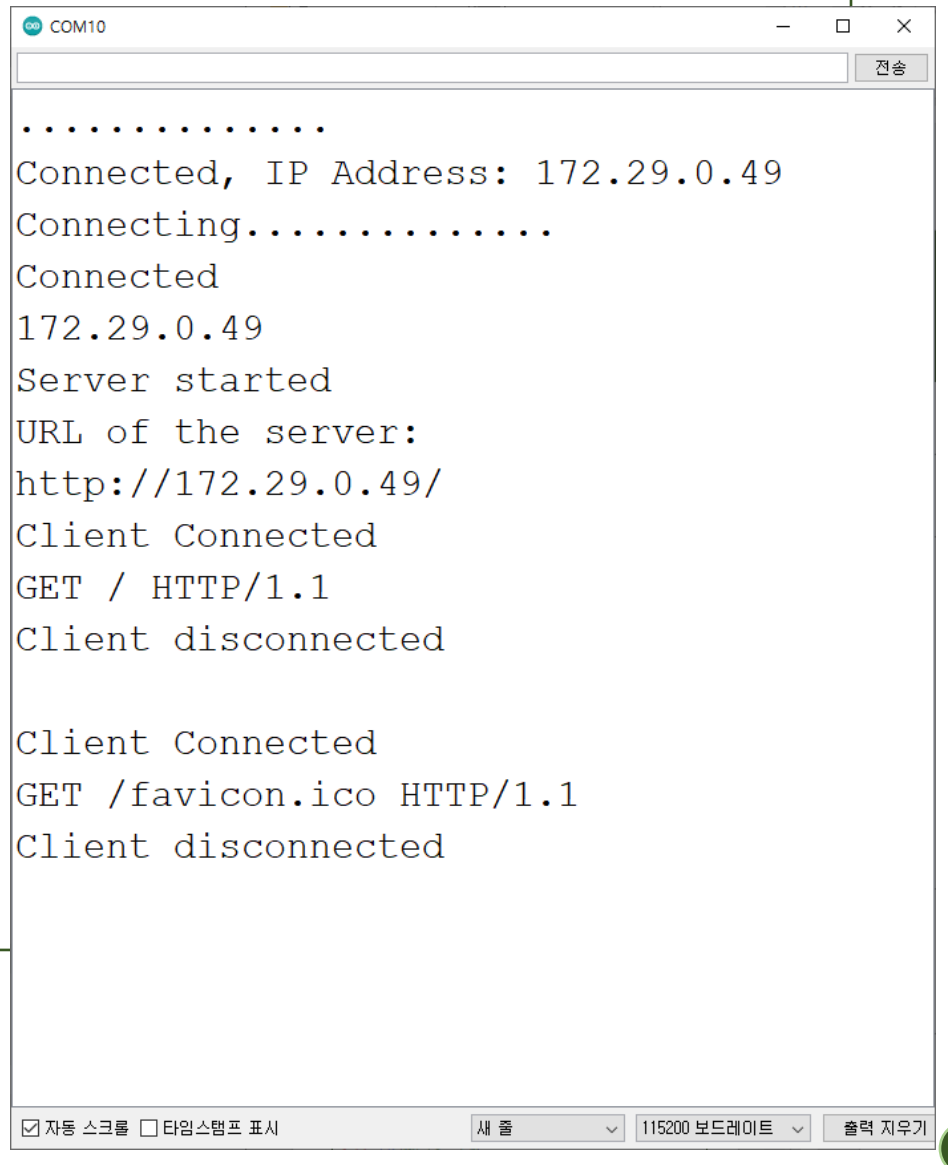
예제 6.1과 동일한 과정

예제 6-2. 간단한 웹 서버 구현 #3

```
  // 서버 시작
  server.begin();
  Serial.println("Server started");

  // 서버 정보 출력
  Serial.println("URL of the server: ");
  Serial.print("http://");   Serial.print(WiFi.localIP());  Serial.println("/");
}

void loop() {
  // 서버에 접속한 클라이언트가 있는지 검사
  WiFiClient client = server.available();
  // 접속한 클라이언트가 없으면 함수 종료
  if(!client) {
    return;
  }

  // 클라이언트(웹 브라우저)가 추가 정보를 보낼 때까지 대기
  Serial.println("Client Connected");
  while(!client.available()) {
    delay(1);
  }
```

예제 6-2. 간단한 웹 서버 구현 #4

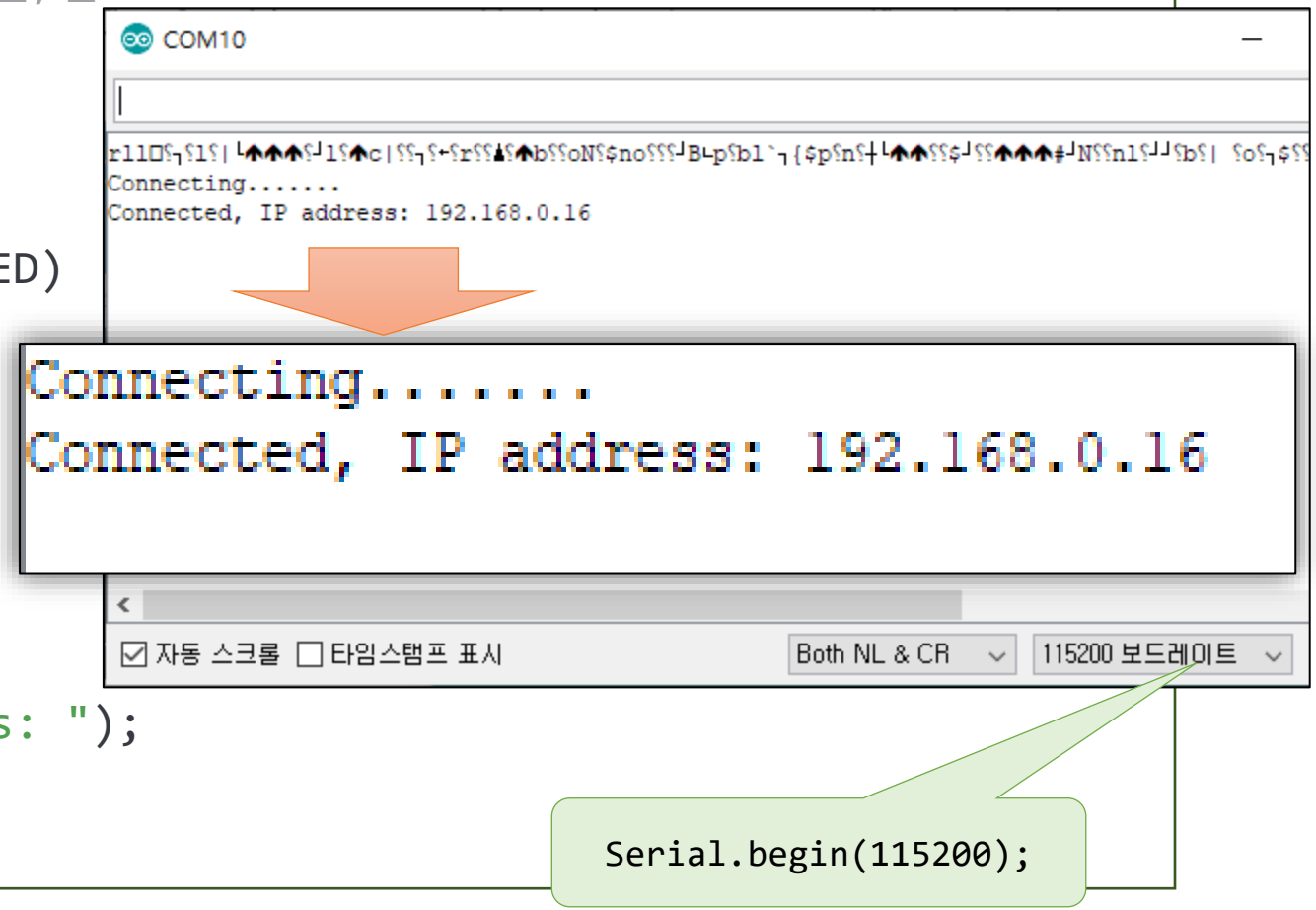
```
  // 클라이언트가 보낸 요구사항을 읽어들이기
  String request = client.readStringUntil('\r');
  Serial.println(request); // 시리얼 모니터로 출력
  client.flush();

  // 클라이언트에게 응답을 보내기
  client.println("HTTP/1.1 200 OK");  // 응답 헤더
  client.println("Content-Type: text/html");
  client.println(""); // 헤더와 본문의 구분
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.println("Connected to NodeMCU device !!!");
  client.println("</html>");

  // 출력을 다 보낸 후에는 연결이 종료됨
  delay(1);
  Serial.println("Client disconnected");
  Serial.println("");
}
```



예제 6-1. 무선인터넷 연결

예제 6-1. NodeMCU 보드의 무선인터넷 연결

```
  // 무선인터넷(Wifi) 시작 : 네트워크 이름/암호
  WiFi.begin("UIDUK_CLASS", "");

  // 네트워크(공유기)에 연결 시도
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  // 연결된 후에 할당받은 IP 주소 출력
  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
}
```



예제 6-2. 간단한 웹 서버 구현 (설명) #1

- 초기 설정 : setup() 함수
  - 서버 클래스 객체 생성 : 서버 포트 번호 지정

```
// 서버용 클래스 객체 (웹서버에 사용되는 80번 포트 지정)
WiFiServer server(80);
```

  - 접속과정 : 예제 6-1과 동일
  - 서버 시작 : WiFiServer::begin() 함수 사용

```
  // 서버 시작
  server.begin();
  Serial.println("Server started");
```

  - 서버 정보 출력

```
  // 서버 정보 출력
  Serial.println("URL of the server: ");
  Serial.print("http://"); Serial.print(WiFi.localIP()); Serial.println("/");
```

예제 6-2. 간단한 웹 서버 구현 (설명) #2

- loop() 함수
  - 접속한 클라이언트가 있는지 검사하고 없으면 함수 종료

```
// 서버에 접속한 클라이언트가 있는지 검사
WiFiClient client = server.available();
// 접속한 클라이언트가 없으면 함수 종료
if(!client) {
  return;
}
```

  - 웹 브라우저가 보내는 추가 정보 확인

```
// 클라이언트(웹 브라우저)가 추가 정보를 보낼 때까지 대기
Serial.println("Client Connected");
while(!client.available()) {
  delay(1);
}
```

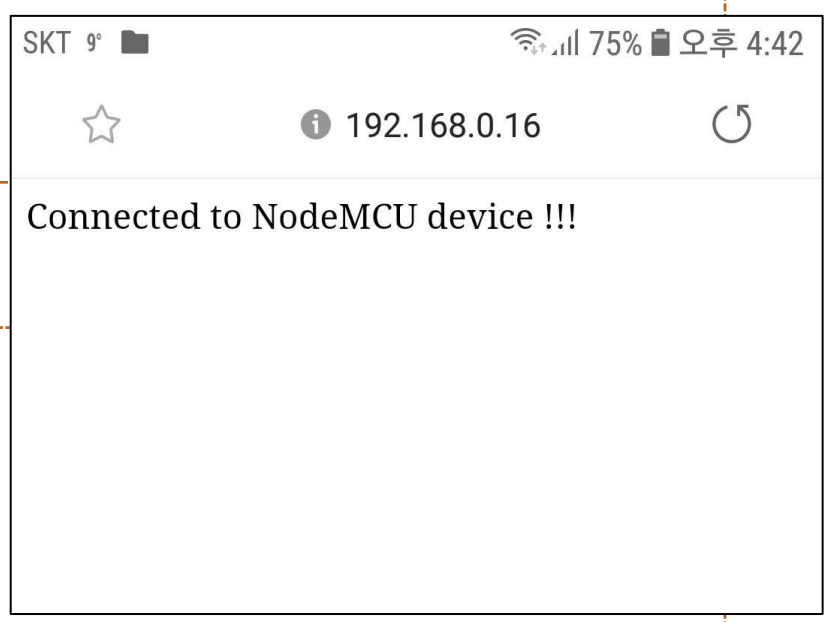
예제 6-2. 간단한 웹 서버 구현 (설명) #3

- loop() 함수
  - 웹 브라우저가 보내는 추가정보 확인

```
// 클라이언트가 보낸 요구사항을 읽어들이기
String request = client.readStringUntil('\r');
Serial.println(request); // 시리얼 모니터로 출력
client.flush();
```

  - 서버 응답 전송 : 서버 → 웹 브라우저 (클라이언트)

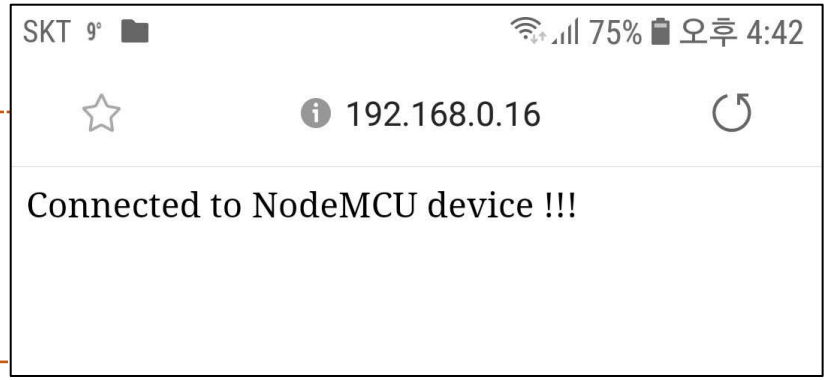
```
// 클라이언트에게 응답을 보내기
client.println("HTTP/1.1 200 OK");  // 응답 헤더
client.println("Content-Type: text/html");
client.println(""); // 헤더와 본문의 구분
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("Connected to NodeMCU device !!!");
client.println("</html>");

// 출력을 다 보낸 후에는 연결이 종료됨
```



예제 6-2. 간단한 웹 서버 구현 (설명) #4

- 브라우저로 전송된 정보
  - 헤더 : 서버가 보내는 정보의 종류 (웹페이지, jpeg 파일, …)
  - 본문 : 웹 페이지 소스

```
<!DOCTYPE HTML>
<html>
Connected to NodeMCU device !!!
</html>
```



  - 웹 브라우저 (클라이언트)가 보낸 요청정보

```
GET / HTTP/1.1
```

요청 방법 (GET, PUT, …)

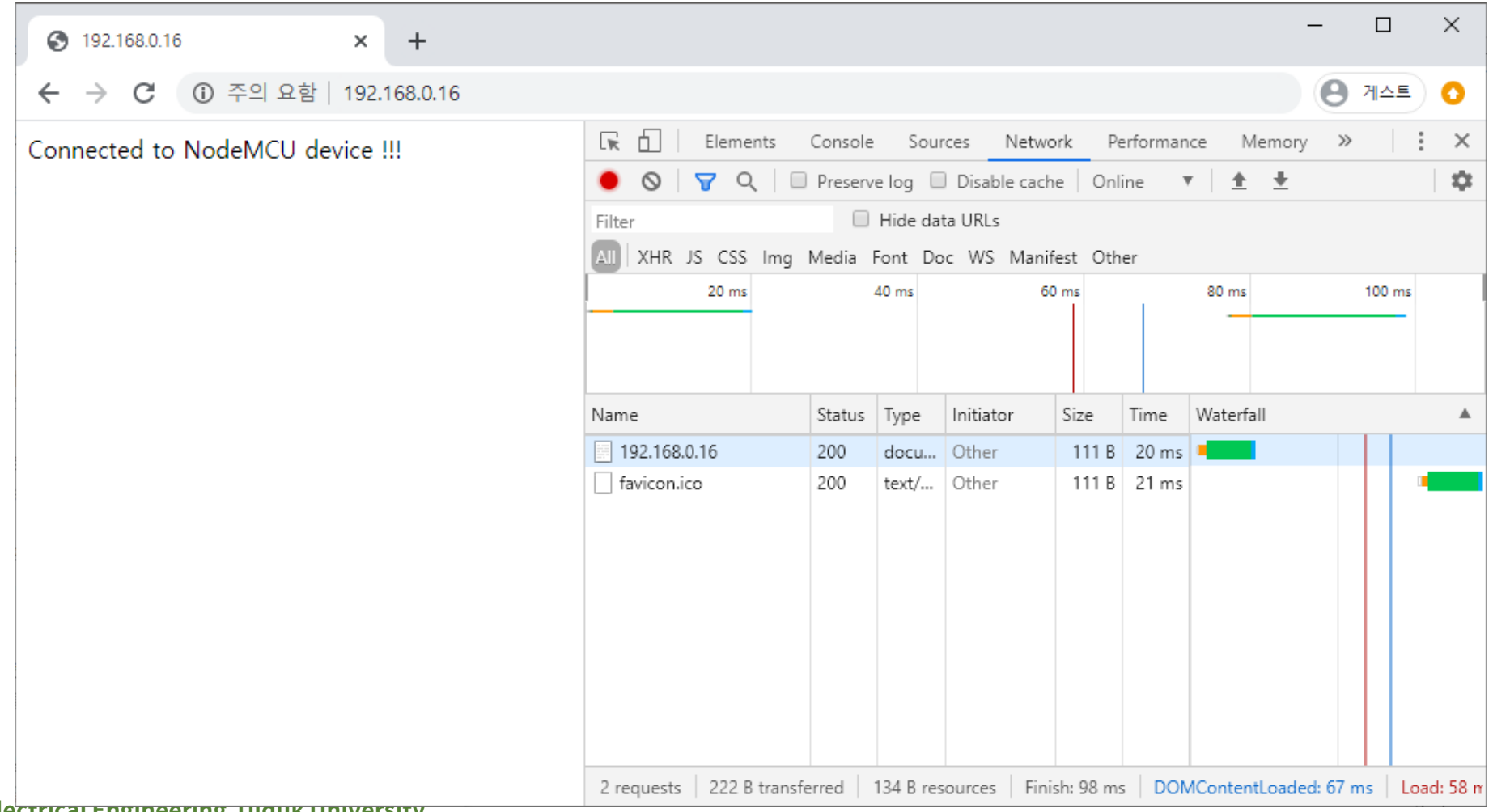
페이지 주소

프로토콜 : 'HTTP/1.1'

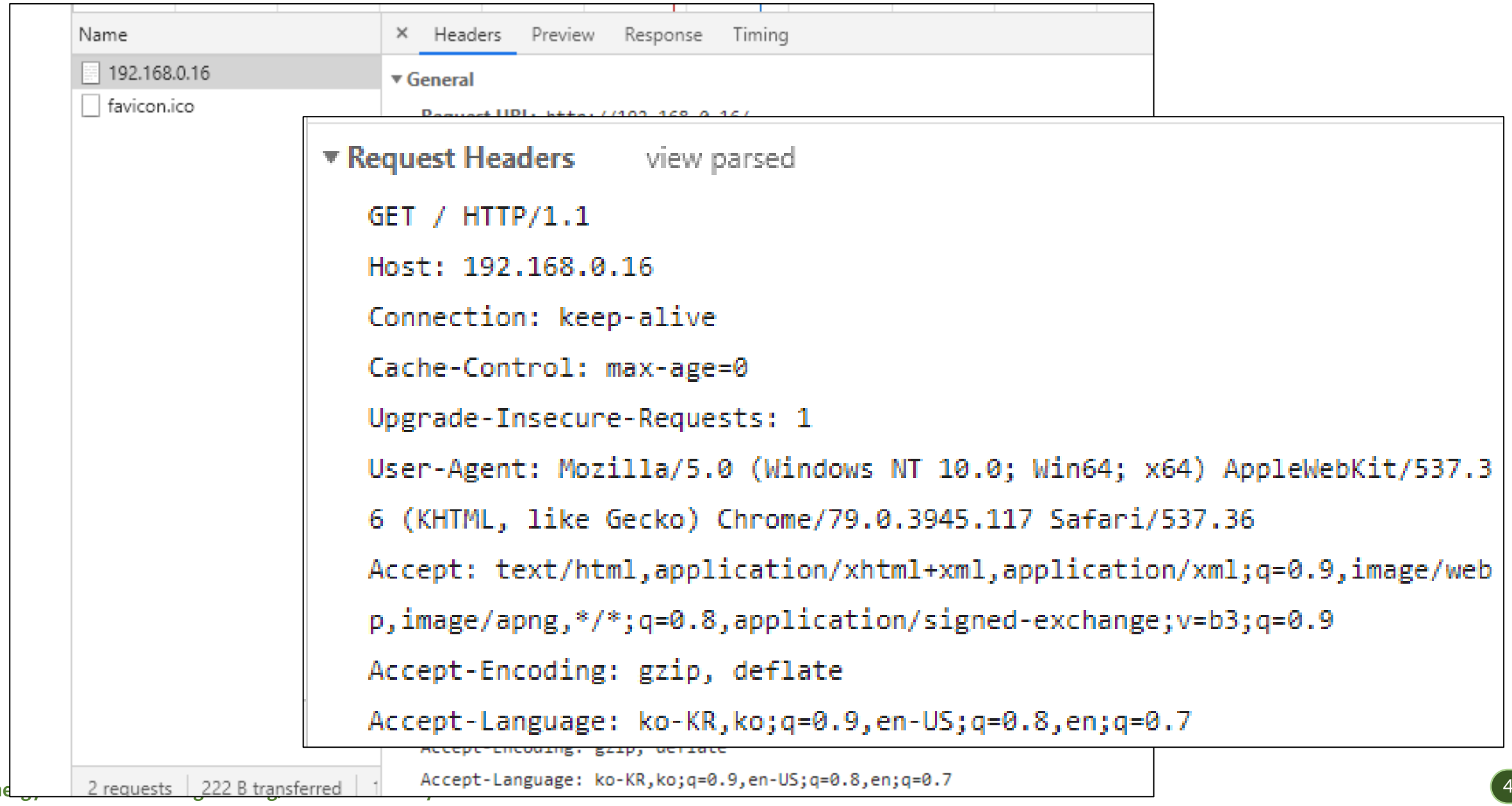
```
Connected
192.168.0.16
Server started
URL of the server:
http://192.168.0.16/
Client Connected
GET / HTTP/1.1
Client disconnected

Client Connected
GET /favicon.ico HTTP/1.1
Client disconnected
```

Chrome 웹 브라우저를 이용한 디버깅

- Chrome 디버깅 : 웹 개발용 기능
  - [메뉴 → 도구 더보기 → 개발자 도구], Ctrl+Shift+I



Chrome 웹 브라우저를 이용한 디버깅



HTTP 프로토콜

- 참고 : https://enai.tistory.com/26
- 주요 응답 코드

응답 코드	의미
200 OK	요청이 성공적으로 처리되었다.
201 Created	요청이 성공적으로 처리되었으며, 그 결과 새로운 리소스가 생성되었다.
400 Bad Request	잘못된 문법을 전송하는 경우. 없는 주소이거나 서버가 이해할 수 없는 경우 발생함.
404 Forbidden	컨텐츠에 접근 불가능한 경우에 발생. 주로 권한이 없는 경우에 발생함
500 Internal Server Error	서버 내부에서 처리가 불가능한 상황이 발생한 경우

6.3 ESP8266WebServer클래스

- 예제 6-2 : 범용 서버 클래스
  - 웹 서버만의 HTTP 프로토콜을 구현해야 함
  - 자세한 응답 방법 구현해야 함
- ESP8266WebServer 클래스
  - 웹 서버용 클래스 : HTTP 프로토콜을 지원함
  - 페이지에 맞는 응답을 응답 함수 형태로 지정해주면 됨 (`ESP8266Webser::on()` 함수 사용)

예제 6-3. ESP8266WebServer #1

예제 6-3. ESP8266WebServer 클래스를 이용한 서버

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

ESP8266WebServer server(80); // 서버객체를 정의 (포트번호 : 80)

void setup()
{
   < setup() 함수의 서버 접속까지 생략 : 예제 6-2와 동일 >

  // 서버 구현 부분
  // 각 세부주소에 대항 응답함수 설정
  server.on("/", procRoot);
  server.on("/test", procTest);
  server.on("/inline", []() { // C++ 람다 함수로 구현
    server.send(200, "text/plain", "inline function");
  });
  // 서버 시작
  server.begin();
}
```

예제 6-3. ESP8266WebServer #2

```
void loop() {
  // 클라이언트 접속 처리
  server.handleClient();
}

// "/" 주소에 대한 처리함수
void procRoot() {
  String ret = "<!DOCTYPE HTML>\r\n";
  ret += "<html>\r\n";
  ret += "Connected to NodeMCU device via ESP8266Webserver !!!\r\n";
  ret += "</html>\r\n";
  server.send(200, "text/html", ret);
}

// "/test" 주소에 대한 처리
void procTest() {
  String ret = "<!DOCTYPE HTML>\r\n";
  ret += "<html>\r\n";
  ret += "시험용 페이지입니다. !!!\r\n";
  ret += "</html>\r\n";
  server.send(200, "text/html", ret);
}
```

예제 6-3. ESP8266WebServer (설명) #1

- ESP8266WebServer::on() 함수
  - 요청한 주소에 대한 응답 처리 함수를 지정 (함수 이름 지정)

```
  server.on("/", procRoot);
  server.on("/test", procTest);
```

  - 응답 처리 함수 : 응답 내용을 문자열로 만들어서 전송

```
void procRoot() {
  String ret = "<!DOCTYPE HTML>\r\n";
  ret += "<html>\r\n";
  ret += "Connected to NodeMCU device via ESP8266Webserver !!!\r\n";
  ret += "</html>\r\n";
  server.send(200, "text/html", ret);
}
```

예제 6-3. ESP8266WebServer (설명) #2

- ESP8266WebServer::on() 함수
  - 요청한 주소에 대한 응답 처리 함수를 지정 (함수 이름 지정)

```
  server.on("/", procRoot);
  server.on("/test", procTest);
```

  - 응답 처리 함수 : 응답 내용을 문자열로 만들어서 전송

```
// "/test" 주소에 대한 처리
void procTest() {
  String ret = "<!DOCTYPE HTML>\r\n";
  ret += "<html>\r\n";
  ret += "시험용 페이지입니다. !!!\r\n";
  ret += "</html>\r\n";
  server.send(200, "text/html", ret);
}
```

예제 6-3. ESP8266WebServer (설명) #3

- ESP8266WebServer::on() 함수
  - 요청한 주소에 대한 응답 처리 함수를 지정 (함수 이름 지정)
  - C++ 람다함수 (lambda function)
    - C++11 에서부터 지원하며, 이름이 없는 익명함수를 바로 선언하여 사용
    - 다른 함수의 파라미터로 전달하는데 주로 사용
    - 형식 : `[capture] (param) -> return type { body }`
      - ✓ capture : 변수 전달
      - ✓ param : 인수, 인수가 없는 경우 `(param)`을 생략 가능
      - ✓ Return type : 리턴값의 데이터 형, 리턴값이 없는 경우 `-> return type`를 생략 가능
      - ✓ body : 함수 정의

```
server.on("/inline", []() { // C++ 람다 함수로 구현
  server.send(200, "text/plain", "inline function");
});
```

C++ Lambda Function

- C++ 람다 함수의 예

```
#include <iostream>

using namespace std;

int main(void)
{
    int a = 5;
    int b = 2;

    // 변수캡쳐를 사용하지 않고, 인자를 직접 받는 예시
    int result1 = [](int n, int m)->int{return n + m;}(a, b);    // 7

    // 변수캡쳐에 지역변수를 사용한 예시
    int result2 = [a]()->int{return a;}();                       // 5
    int result3 = [a, b]()->int{return a + b;}();                // 7
```

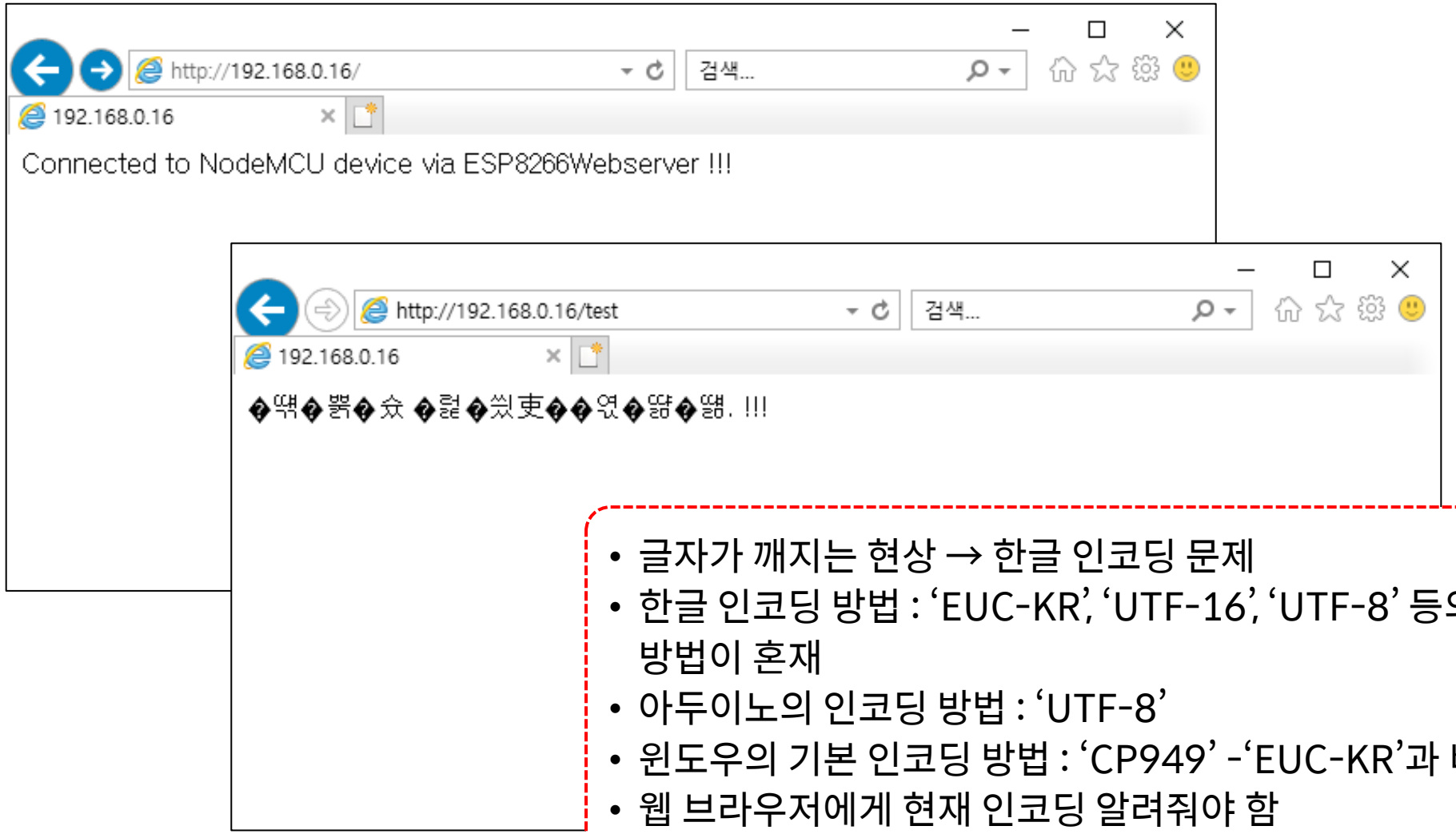
C++ Lambda Function

- C++ 람다 함수의 예

```
    int a = 5;
    int b = 2;

    // ...

    // 변수캡쳐에 =를 사용한 예시
    int result4 = [=]()->int{return a + b;}();                    // 7

    // 변수캡쳐에 &를 사용한 예시
    int result5 = [&]()->int{return a + b;}();                    // 7
    int result6 = [&]()->int{return a = a + b;}();                // 7, a = 7

    // 람다함수 내부에 람다함수를 포함시킨 예시
    int result7 = [&]()->int{                                     // 9
        return [&]()->int{return a + b;}();
    }();
}
```

예제 6-3. ESP8266WebServer 결과

- 웹 브라우저에서 호출한 결과



- 글자가 깨지는 현상 → 한글 인코딩 문제
- 한글 인코딩 방법 : 'EUC-KR', 'UTF-16', 'UTF-8' 등의 다양한 방법이 혼재
- 아두이노의 인코딩 방법 : 'UTF-8'
- 윈도우의 기본 인코딩 방법 : 'CP949' -'EUC-KR'과 비슷
- 웹 브라우저에게 현재 인코딩 알려줘야 함

예제 6-3. ESP8266WebServer 결과

- 웹 페이지의 인코딩 지정 방법

```
<!DOCTYPE HTML>
<html>
Connected to NodeMCU device !!!
</html>
```

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
Connected to NodeMCU device !!!
</body>
</html>
```

6.4 클라이언트

- 클라이언트 구현 방법
  - 범용 클라이언트 : WiFiClient 클래스 사용
  - 웹 클라이언트 : HTTPClient 클래스 사용
  - 웹 브라우저의 구현은 NodeMCU에서는 거의 불가능 (멀티미디어 표현이 거의 불가능)

예제 6-4. 웹 클라이언트의 구현 #1

예제 6-4. 웹 클라이언트의 구현

```
#include <ESP8266WiFi.h>

void setup()
{
   < WIFI 시작 부분 생략 : 예제 6-1과 동일 >
}

void loop() {
  // 클라이언트 객체 정의
  WiFiClient client;

  // 서버에 접속, 서버 주소와 포트 번호를 지정
  // 실패하면 메시지 표시하고 함수 종료
  if(!client.connect("energy.uu.ac.kr", 80)) {
    Serial.println("Connection failed !!");
    return;
  }
```

예제 6-4. 웹 클라이언트의 구현 #2

```
  // 서버에 요청할 request 생성
  String url = "/"; // 접속 주소
  // request 만들기 : "GET / HTTP/1.1"
  String request = "GET " + url + " HTTP/1.1\r\n";
  request += "Host: energy.uu.ac.kr\r\n";
  request += "Connection: close\r\n\r\n";
  Serial.print("request : ");
  Serial.println(request);

  client.print(request);   // 서버에 요청하기

  while(client.available() == 0) {  // 응답이 올때까지 대기하기
    // 아무것도 하지 않는다...
  }
  while(client.available()) {  // 서버로 부터 수신한 응답을 시리얼 모니터로 전송
    String line = client.readStringUntil('\r');      // 줄의 마지막까지 읽기
    Serial.print(line);
  }
  delay(10000);   // 10초동안 기다린다.
}
```

예제 6-4. 웹 클라이언트의 구현 (설명) #1

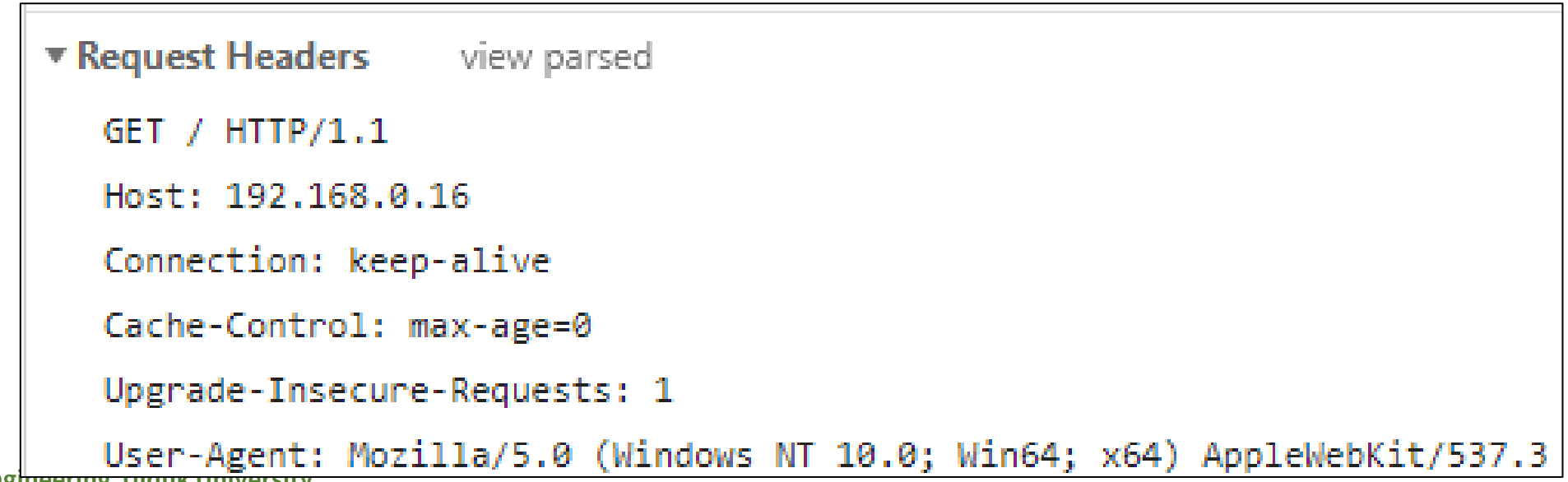
- WiFiClient 객체 정의 및 서버 접속

```
// 클라이언트 객체 정의
WiFiClient client;

// 서버에 접속, 서버 주소와 포트 번호를 지정
// 실패하면 메시지 표시하고 함수 종료
if(!client.connect("energy.uu.ac.kr", 80)) {
  Serial.println("Connection failed !!");
  return;
}
```

예제 6-4. 웹 클라이언트의 구현 (설명) #2

- 서버에 요청할 request 생성 후 요청

```
// 서버에 요청할 request 생성
String url = "/"; // 접속 주소
// request 만들기 : "GET / HTTP/1.1"
String request = "GET " + url + " HTTP/1.1\r\n";
request += "Host: energy.uu.ac.kr\r\n";
request += "Connection: close\r\n\r\n";

client.print(request);   // 서버에 요청하기
```



예제 6-4. 웹 클라이언트의 구현 (설명) #3

- 서버응답을 PC로 전송

```
while(client.available() == 0) {  // 응답이 올때까지 대기하기
  // 아무것도 하지 않는다...
}
while(client.available()) {  // 서버로 부터 수신한 응답을 시리얼 모니터로 전송
  String line = client.readStringUntil('\r');     // 줄의 마지막까지 읽기
  Serial.print(line);
}
```

```
HTTP/1.1 200 OK
Date: Tue, 28 Jan 2020 10:57:04 GMT
Server: Apache
Content-Length: 306
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <meta http-equiv="Refresh" content="0; URL=/xe/" />
  <title>== 위덕대학교 에너지전기공학부 ==</title>
</head>
<body>
 
</body>
</html>
```

예제 6-5. HTTPClient 클래스 사용 #1

예제 6-5. HTTPClient 클래스 사용

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

// 무선네트워크 접속용 이름/비밀번호
#define SSID "xxxxxxxxxxxxx"
#define PASS "xxxxxxxxxxxxx"

void setup()
{
 < WIFI 시작 부분 생략 : 예제 6-1과 유사 >
}
```

예제 6-5. HTTPClient 클래스 사용 #2

```
void loop() {
  // 클라이언트 객체 정의
  HTTPClient http;

  // 읽어들일 주소를 지정하여 접속을 시작함
  http.begin("http://energy.uu.ac.kr/");
  int retCode = http.GET();
  if(retCode > 0) {
    String line = http.getString();
    Serial.println(line);
  }
  else {
    Serial.println("Error");
  }

  http.end();
  // 1000초동안 기다린다.
  delay(1000000);
}
```

6.5 웹 클라이언트의 응용

- 기상청에서 제공하는 기상자료 받아와서 활용하기
  - 기상청 날씨 정보 : http://www.weather.go.kr/weather/lifenindustry/sevice_rss.jsp
  - 결과 : RSS(Rich Site Summary) 서비스로 지정한 지역의 날씨 예보 전송
  - RSS : 텍스트 형식으로 XML 구조를 가지는 기본적인 정보 (예:경주시/강동면)
    - http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=4713037000

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
<title>기상청 동네예보 웹서비스 - 경상북도 경주시 강동면 도표예보</title>
<link>http://www.kma.go.kr/weather/main.jsp</link>
<description>동네예보 웹서비스</description>
<language>ko</language>
<generator>동네예보</generator>
<pubDate>2020년 01월 28일 (화)요일 20:00</pubDate>
 <item>
<author>기상청</author>
<category>경상북도 경주시 강동면</category>
<title>동네예보(도표) : 경상북도 경주시 강동면 [X=101,Y=93]</title><link>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCode=4713037000</link>
<guid>http://www.kma.go.kr/weather/forecast/timeseries.jsp?searchType=INTEREST&amp;dongCode=4713037000</guid>
<description>
 <header>
  <tm>202001282000</tm>
  <ts>6</ts>
  <x>101</x>
  <y>93</y>
 </header>
 <body>
  <data seq="0">
```

기상청 날씨 정보 (RSS) 구조

한번의 날씨 정보

↓

XML 구조 해석

↓

활용

```
<header>
  <tm>202001282000</tm>   → 발표시각
  <ts>6</ts>
  <x>101</x>
  <y>93</y>
 </header>
 <body>
  <data seq="0">
   <hour>24</hour>        → 예보시각 (3시간 단위)
   <day>0</day>           → 날짜 (0: 오늘,1: 내일, 2: 모래)
   <temp>6.0</temp>       → 온도
   <tmx>-999.0</tmx>      → 최고온도 (음수이면 값이 없을 경우)
   <tmn>-999.0</tmn>      → 최저온도 (음수이면 값이 없을 경우)
   <sky>4</sky>           → 하늘상태 (1: 맑음, 3: 구름많음, 4: 흐림)
   <pty>0</pty>           → 강수상태 (0: 없음, 1: 비, 2: 비/눈, 3: 눈, 4: 소나기)
   <wfKor>흐림</wfKor>    → 날씨 한국어 설명
   <wfEn>Cloudy</wfEn>    → 날씨 영어 설명
   <pop>30</pop>          → 강수 확률 [%]
   <r12>0.0</r12>         → 12시간 예상 가우량
   <s12>0.0</s12>         → 12시간 예상 적설량
   <ws>2.6</ws>           → 풍속 [m/s]
   <wd>7</wd>             → 풍향 (0~7: 북, 북동, 동, 남동, ... 북서)
   <wdKor>북서</wdKor>    → 풍향 한국어
   <wdEn>NW</wdEn>        → 풍향 영어
   <reh>80</reh>          → 습도 [%]
   <r06>0.0</r06>         → 6시간 예상 강우량
   <s06>0.0</s06>         → 6시간 예상 적설량
  </data>
```

- XML 구조 : 태그(tag)들로 구성
- 예] <tag>내용</tag>

예제 6-6. 기상청 날씨정보 받아와서 출력하기 #1

예제 6-6. 기상청 날씨정보 받아와서 출력하기

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

// 무선네트워크 접속용 이름/비밀번호
#define SSID "xxxxxxxxxxxxx"
#define PASS "xxxxxxxxxxxxx"

void setup()
{
 < WIFI 시작 부분 생략 : 예제 6-5와 동일 >
}

// 서버에서 받아온 정보를 저장하는 변수
String xml;
```

예제 6-6. 기상청 날씨정보 받아와서 출력하기 #2

```
// 전체 문자열에서 하나의 태그 정보를 탐색하고 내용을 추출
String extractTag(String tag)
{
  // 시작 태그, tag == "hour"인경우 -> <hour>
  String st_tag = String("<") + tag + ">";
  // 종료 태그, tag == "hour"인경우 -> </hour>
  String ed_tag = String("</") + tag + ">";
  // 문자열 검색을 통해서 태그가 있는 위치 찾기
  int st_pos = xml.indexOf(st_tag);
  int ed_pos = xml.indexOf(ed_tag);

  // 태그가 없는 경우 빈 문자열을 반환
  if(st_pos < 0 || ed_pos < 0) return String("");

  // 태그 위치를 기반으로 내용물 추출
  String cont = xml.substring(st_pos + st_tag.length(), ed_pos);

  // 처음부터 추출한 태그가 들어있는 부분까지 삭제
  xml.remove(0, ed_pos + ed_tag.length());
  return cont;
}
```

- XML 구조 : 태그(tag)들로 구성
- 예] <tag>내용</tag>

예제 6-6. 기상청 날씨정보 받아와서 출력하기 #3

```
void loop() {
  // 클라이언트 객체 정의
  HTTPClient http;

  // 읽어들일 주소를 지정하여 접속을 시작함
  http.begin("http://www.kma.go.kr/wid/queryDFSRSS.jsp?zone=4713037000");
  int retCode = http.GET();
  if(retCode > 0) {
    xml = http.getString();
    Serial.print(xml.length());
    Serial.println(" bytes received");
    Serial.println(extractTag("tm"));
    // 5개 (3시간 단위이므로 총 15시간 후까지)의 정보를 읽어들인다.
    for(int i=0; i<5; i++) {
      Serial.print("시각 : ");
      Serial.print(extractTag("hour"));
      Serial.println("시");
      Serial.print("온도 : ");
      Serial.print(extractTag("temp"));
      Serial.println(" [*C]");
```

```
<data seq="0">
<hour>24</hour>      → 예보시각 (3시간 단위)
<day>0</day>         → 날짜 (0: 오늘,1: 내일, 2: 모래)
<temp>6.0</temp>     → 온도
<tmx>-999.0</tmx>    → 최고온도 (음수이면 값이 없을 경우)
<tmn>-999.0</tmn>    → 최저온도 (음수이면 값이 없을 경우)
```

예제 6-6. 기상청 날씨정보 받아와서 출력하기 #4

```
        Serial.print("날씨 : ");
        Serial.print(extractTag("wfKor"));
        Serial.println("");
        Serial.print("풍속 : ");
        Serial.print(extractTag("ws"));
        Serial.println(" [m/s]");
        Serial.print("풍향 : ");
        Serial.print(extractTag("wdKor"));
        Serial.println("");
        Serial.print("습도 : ");
        Serial.print(extractTag("reh"));
        Serial.println(" [%]");
        Serial.println();
      }
    }
    else { Serial.println("Error"); }
    http.end();
    // 1000초동안 기다린다.
    delay(1000000);
}
```

```
<wfKor>흐림</wfKor>   → 날씨 한국어 설명
<wfEn>Cloudy</wfEn>   → 날씨 영어 설명
<pop>30</pop>         → 강수 확률 [%]
<r12>0.0</r12>        → 12시간 예상 가우량
<s12>0.0</s12>        → 12시간 예상 적설량
<ws>2.6</ws>          → 풍속 [m/s]
<wd>7</wd>            → 풍향 (0~7: 북, 북동, 동,
                        남동, ... 북서)
<wdKor>북서</wdKor>   → 풍향 한국어
<wdEn>NW</wdEn>       → 풍향 영어
<reh>80</reh>         → 습도 [%]
<r06>0.0</r06>        → 6시간 예상 강우량
<s06>0.0</s06>        → 6시간 예상 적설량
```

예제 6-6. 기상청 날씨정보 받아와서 출력하기 결과

```
Connecting.......
Connected
7580 bytes received
202001282000
시각 : 24시
온도 : 6.0 [*C]
날씨 : 흐림
풍속 : 2.6 [m/s]
풍향 : 북서
습도 : 80 [%]

시각 : 3시
온도 : 4.0 [*C]
날씨 : 구름 많음
풍속 : 2.5 [m/s]
풍향 : 북서
습도 : 85 [%]

시각 : 6시
온도 : 4.0 [*C]
날씨 : 구름 많음
풍속 : 2.4000000000000004 [m/s]
풍향 : 북서
습도 : 80 [%]
```

추가 실습

- 예제 6-6의 결과를 OLED 장치를 통해서 출력하는 미니 일기 예보 기기를 제작하시오.