Internet of Things (IoT)

목차

1. IoT 개요

이상적 IoT 시스템	
Cloud Storage	Cloud에서의 저장
Cloud Processing	*Cloud Computing
Internet	*HTTP 등
Network	IoT의 게이트웨이
Local storage	Local에서의 저장
Local Processing	*Fog Computing
Sensors	값 측정, raw 데이터 전송, 저전력

1.1. Fog Computing / Cloud Computing

- 포그 컴퓨팅과 클라우드 컴퓨팅은 상호 보완적인 관계
- 포그 컴퓨팅에선 소용량 처리
- 클라우드 컴퓨팅에선 중앙식 대용량 처리
- IoT에서는 두 기술의 융합이 중요

측면	포그 컴퓨팅	클라우드 컴퓨팅
데이터 처리속도	Real-time에 가까운 처리	Non-real-time 처리
데이터 처리량	소규모 Local 데이터	대규모 Big 데이터
주요기능	분석/필터링	수집/저장/처리
분포범위	지리적으로 광범위한 분포	중앙 데이터 센터에 집중 구성
고성능처리	고성능 처리 어려움	고성능 처리 가능

1.2. Microcontroller (1)

- 많은 수의 핀으로 구성
- 아날로그 혹은 디지털 입출력 가능
- GPIO (General Purpose Input Output: 범용 입출력) :
  - 개발자 의도에 따라 사용할 수 있는 마이크로 콘트롤러 핀

- 주로 특수 목적으로 사용
- 한 칩 안에 처리장치, 기억장치, 입출력장치 내장
- 높은 성능과 빠른 처리속도가 필요 없고 항상 정확하고 똑같은 제어에 사용

MIPS	• Million Instructions Per Second의 약자 • 초당 처리되는 명령어 수를 백만단위로 나타낸 것
FLOPS	• Floating-point Operations Per Second의 약자 • 초당 처리되는 부동 소수점 연산 명령의 실행 횟수
MHz	CPU의 데이터 처리 속도인 클록(Clock)을 표시하는 단위

1.2. Microcontroller (2)

- 주로 RISC(Reduced Instruction Set Computer) 구조를 사용.

	CISC (Complex Instruction Set Computer)	**RISC**
명령어 처리	하드웨어에서 처리	소프트웨어에서 처리
하드웨어	크고 복잡	간단 (Chip 사이즈)
시스템 규모	크다	작다
사용	X86 기반 컴퓨터 (PC)	마이크로프로세서 / ARM기반 프로세서
기타사항	고전력, 대량의 데이터 처리, 고가	저전력, 소형, 저렴

- 프로그래밍 가능한 소형 디바이스
- 쉽게 연결 가능
- 오픈소스
- 사용하기 쉬운 소프트웨어로 구성됨

1.3. Microcontrollers and Computers

- Simple systems
- Control hardware
- Low speeds
- Small memory
- GPIO
- PWM
- ADC
- Run single software

Firmware

Good for sensors (8비트)

Good for some sensors and processing(32비트)

비트? -> 한 명령어로 처리할 수 있는 데이터의 크기

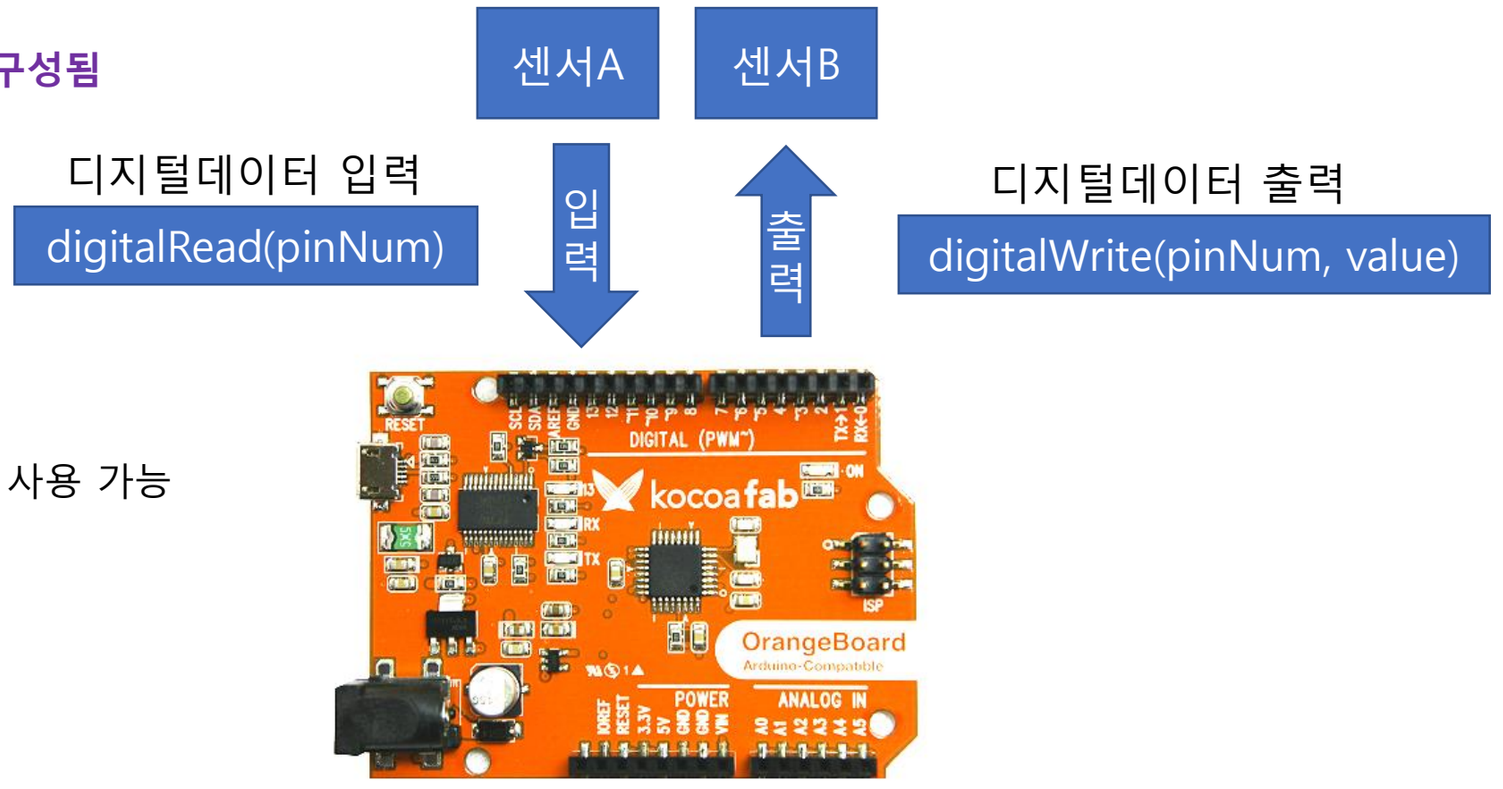
- Full CPUs
- High speeds
- Large memory
- GPIO
- Run OS

Software

Good for processing and network (x86기반)

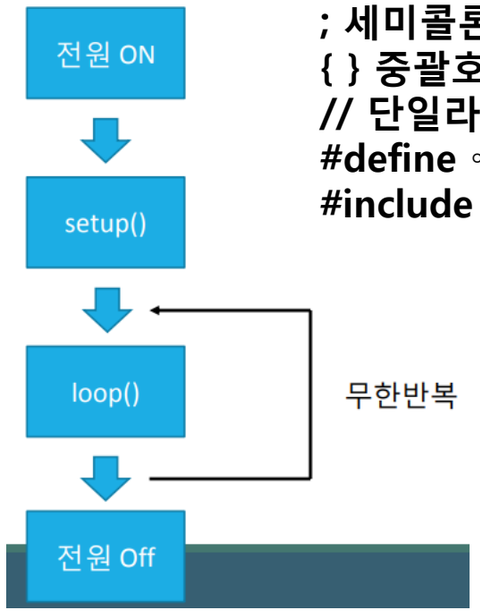
2. Arduino - 개요

- 프로그래밍 가능한 소형 디바이스
- 쉽게 연결 가능
- 오픈소스
- 사용하기 쉬운 소프트웨어로 구성됨



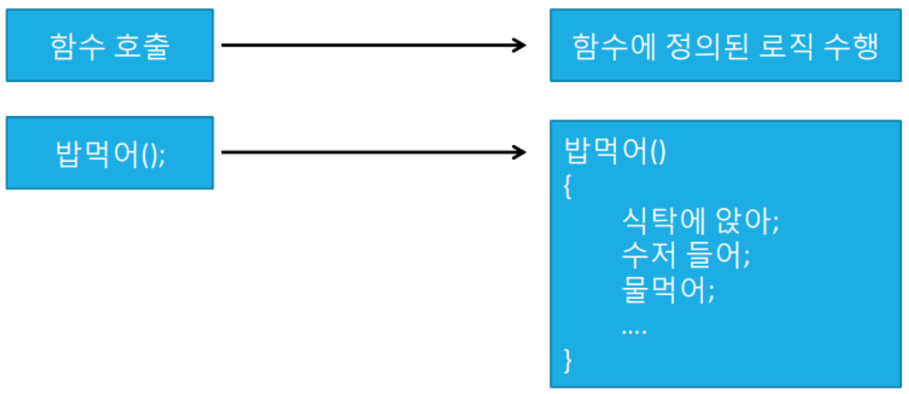
크로스 컴파일 플랫폼
: 다른 환경의 OS에서 공통으로 사용 가능

2.1. 프로그래밍 기본구조 (1)



; 세미콜론◦문의 종료를 알리는 구문
{ } 중괄호◦여러 개의 명령문이 모여서 하나의 기능을 수행할 때 명령문을 묶어주는 블록
// 단일라인 주석 /* … */ 다중라인주석
#define◦프로그래머가 컴파일 전에 값에 이름을 부여하는데 사용
#include◦스케치 프로그램에 라이브러리를 포함하는데 사용

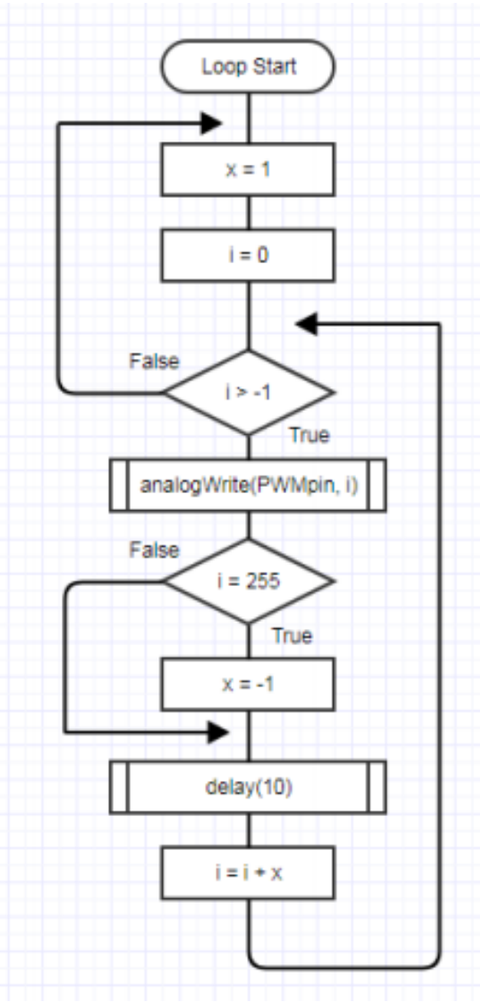
함수(Function) -> 특정 기능을 구현하도록 명령들을 묶어 놓은 것



2.1. 프로그래밍 기본구조 (2)

- 제어문과 반복문
- *다음 loop()함수의 결과를 예상/서술하시고 이를 순서도로 표현하시오.

```
void loop()
{
   int x = 1;
   for (int i = 0; i > -1; i = i + x){
      analogWrite(PWMpin, i);
      if (i == 255) x = -1;            // switch direction at peak
      delay(10);
   }
}
```



<순서도>

2.2. Arduino – 기본함수(1)

- pinMode(pin,mode)

Example Code

The code makes the digital pin 13 `OUTPUT` and Togles it `HIGH` and `LOW`

```
void setup()
{
  pinMode(13, OUTPUT);           // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH);        // sets the digital pin 13 on
  delay(1000);                   // waits for a second
  digitalWrite(13, LOW);         // sets the digital pin 13 off
  delay(1000);                   // waits for a second
}
```

Notes and Warnings

The analog input pins can be used as digital pins, referred to as A0, A1, etc.

2.2. Arduino – 기본함수(2)

digitalRead(pin)

Example Code

Sets pin 13 to the same value as pin 7, declared as an input.

```
int ledPin = 13;    // LED connected to digital pin 13
int inPin = 7;      // pushbutton connected to digital pin 7
int val = 0;        // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin 13 as output
  pinMode(inPin, INPUT);        // sets the digital pin 7 as input
}

void loop()
{
  val = digitalRead(inPin);     // read the input pin
  digitalWrite(ledPin, val);    // sets the LED to the button's value
}
```

Notes and Warnings

If the pin isn't connected to anything, digitalRead() can return either HIGH or LOW (and this can change randomly).

The analog input pins can be used as digital pins, referred to as A0, A1, etc.

2.2. Arduino – 기본함수(3)

digitalWrite(pin, val)

Example Code

The code makes the digital pin 13 an `OUTPUT` and toggles it by alternating between `HIGH` and `LOW` at one second pace.

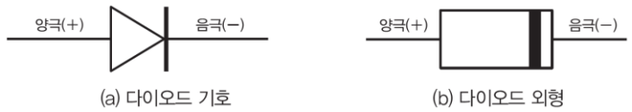
```
void setup()
{
  pinMode(13, OUTPUT);           // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH);        // sets the digital pin 13 on
  delay(1000);                   // waits for a second
  digitalWrite(13, LOW);         // sets the digital pin 13 off
  delay(1000);                   // waits for a second
}
```

Notes and Warnings

The analog input pins can be used as digital pins, referred to as A0, A1, etc.

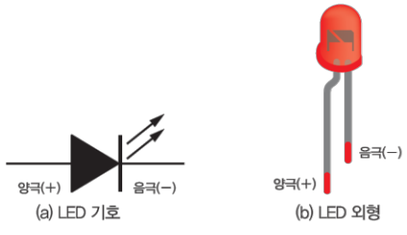
2.3. Arduino - LED

- **다이오드**
  - 양극에서 음극으로 순방향으로만 전류가 흐름



(a) 다이오드 기호 (b) 다이오드 외형

- **LED : Light Emitting Diode, 발광 다이오드**
  - 순방향 연결에서 빛을 냄
  - 화학물질에 따라 다양한 색상의 빛을 냄
  - 리모컨의 적외선 LED, 살균 소독용 자외선 LED 등도 존재
  - 데이터 핀에 연결하여 비트 단위 데이터 확인



(a) LED 기호 (b) LED 외형

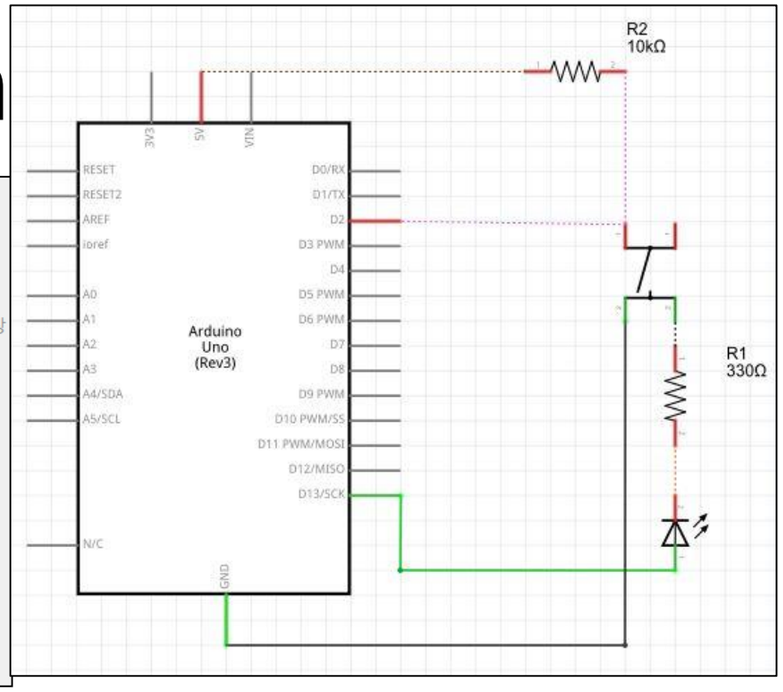
```
// LED를 디지털 13번 핀에 연결합니다.
int  led = 13;

// 실행시 가장 먼저 호출되는 함수이며, 최초 1회만 실행됩니다.
void setup() {
      // 디지털 13번핀을 출력 핀으로 설정합니다.
      pinMode(led, OUTPUT);
}

// setup() 함수가 호출된 이후, loop() 함수가 호출되며,
// 블록 안의 코드를 무한히 반복 실행합니다.
void loop() {
      // LED를 ON 합니다.
      digitalWrite(led, HIGH);
      // 1초간 대기합니다.
      delay(1000);
      // LED를 OFF 합니다.
      digitalWrite(led,LOW);
      // 1초간 대기합니다.
      delay(1000);
}
```

<LED 한 개, 깜박이기>

2.4. Arduino – Push button

```
/*
 제목          : Push button으로 LED 켜기
 내용          : push button으로 LED를 제어해 봅시다.
 */

// push button을 디지털 2번 핀에 연결합니다.
const int button1Pin = 2;
// led를 디지털 13번 핀에 연결합니다.
const int ledPin =  13;

// 실행시 가장 먼저 호출되는 함수이며, 최초 1회만 실행됩니다.
void setup() {
      //2번 핀을 입력 핀으로 설정합니다.
      pinMode(button1Pin, INPUT);
      //13번 핀을 출력 핀으로 설정합니다.
      pinMode(ledPin, OUTPUT);
}
```

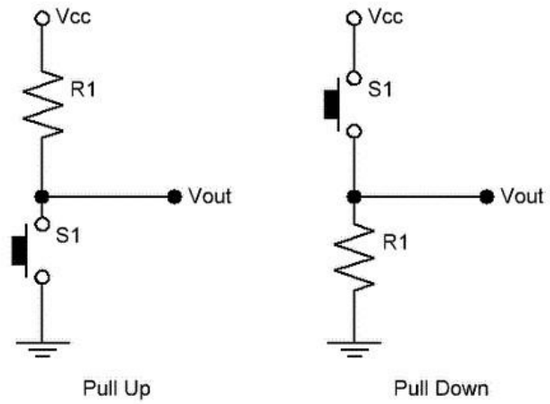
```
// setup() 함수가 호출된 이후, loop() 함수가 호출되며,
// 블록 안의 코드를 무한히 반복 실행합니다.
void loop() {
      // 버튼에서 읽어 올 디지털 값을 저장할 변수를 선언합니다.
      int button1State;
      // 버튼이 눌렸는 지 아닌지 버튼의 상태를 읽어와서 앞에 선언한 변수에 저장
      button1State = digitalRead(button1Pin);

   // 버튼이 눌렸다면
      if(button1State == LOW){
            // LED를 ON 시킵니다.
            digitalWrite (ledPin, HIGH);
      }
      // 버튼이 눌리지 않았다면
      else{
            //LED를 OFF 시킵니다.
            digitalWrite(ledPin, LOW);
      }
}
```



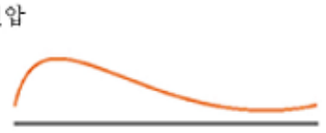
* 핀에 어떤 값이 입력이 되었는지 모르는 경우: **플로팅(floating)**

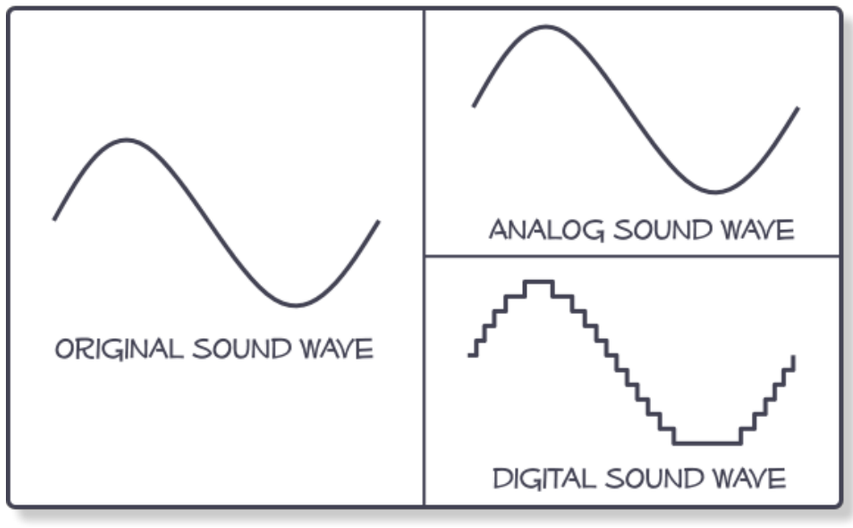
플로팅 상태가 되면 소자의 특성과 주위 환경에 따라
High, Low 상태가 수시로 변할 수 있고, 여러 노이즈를 유발하여
오작동을 일으킬 수 있음.

풀업/풀다운 저항 사용



3. Analog & Digital

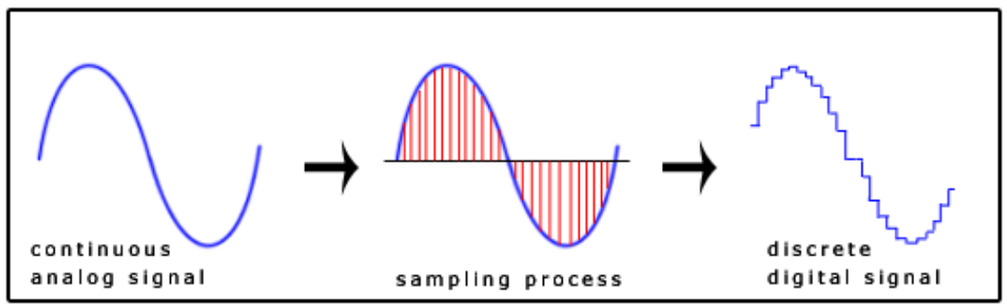
구분	디지털 신호	아날로그 신호
정의	특정 값을 단위로 불연속적으로 변하는 신호	빛, 소리와 같이 연속적으로 변하는 신호
그래프	전압  시간 막대 모양의 불연속적인 형	전압  시간 곡선 형태의 연속적인 형태
예	컴퓨터, 휴대폰 등의 현대 문명에서 사용되는 대부분의 신호	온도 변화에 따른 온도계의 눈금 변화나 아날로그 손목 시계 등
장점	정보의 저장과 전달이 쉽다 변형 없이 전달이 가능하다.	세밀한 표현이 가능하다.
단점	원래의 정보를 그대로 기록할 수 없다.	신호를 전할 때마다 신호가 변형될 수 있다.

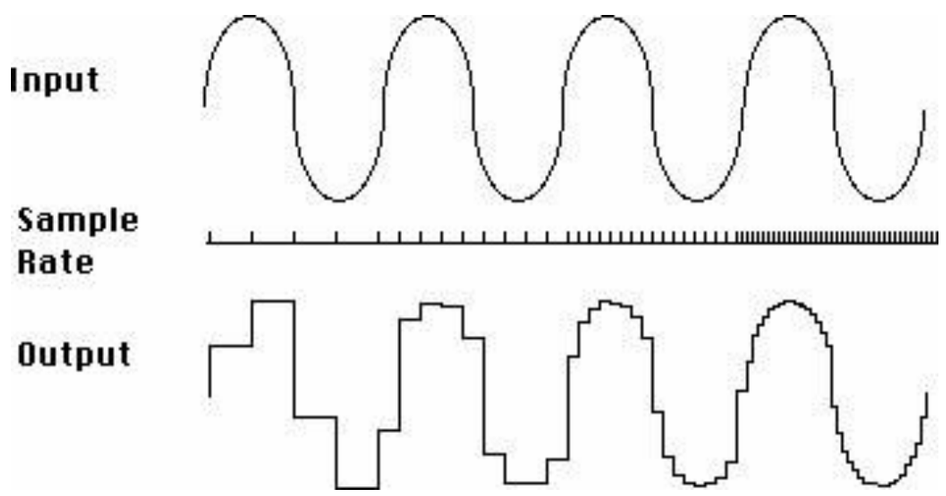


Parameters

Bits per sample **(진폭축)**
Sampling rate **(시간축)**

3.1. Sampling





- 얼마나 자주 데이터를 얻는지.

* **Nyquist Theorem**

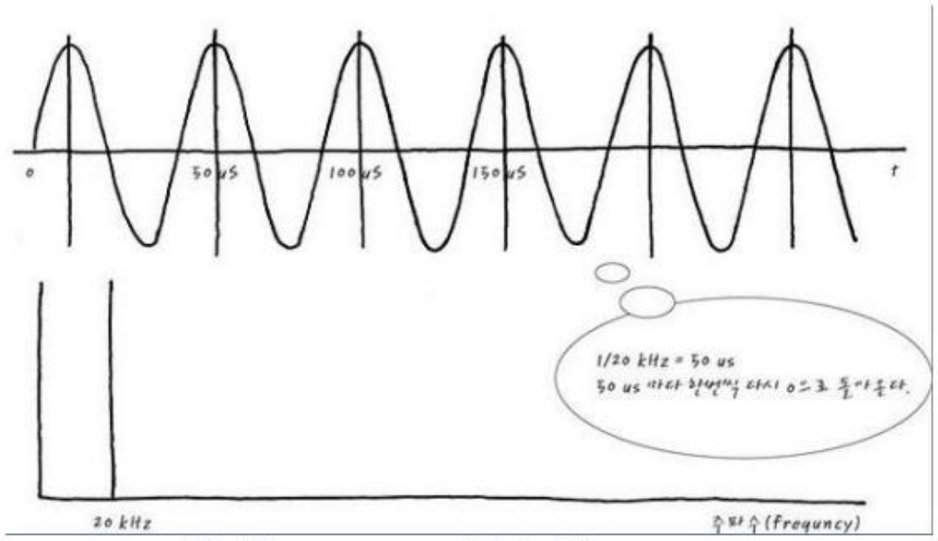
Sampling frequency >= 2 x maximum frequency

→Nyquist frequency 이상 주파수 일시, 신호 원본가능

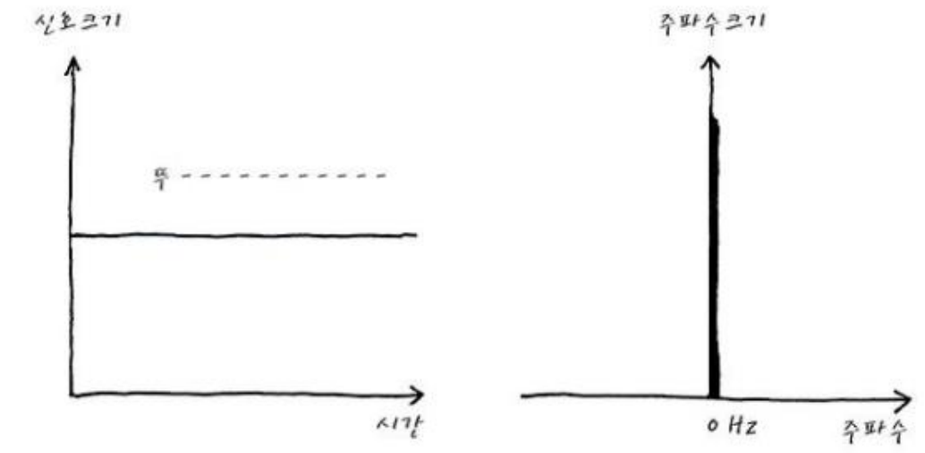
3.2. ADC

특징	설명
* **표본화 (Sampling)**	• 연속적인 신호 파형을 일정 시간 간격으로 검출 • 표본화에 의해 검출된 신호 : 아날로그 형태 • 표본화 횟수 = 최고 주파수 X 2 (<-**Nyquist Theorem)** • 표본화 간격 = 1 / 표본화 횟수(주파수)
* **양자화 (Quantizing)**	• 아날로그 신호를 유한 개의 부호에 대한 값으로 조정하는 과정 • 양자화 레벨을 세밀하게 하면 양자화 잡음이 줄지만, 데이터의 양이 많아지고 전송 효율이 낮아짐 • 양자화 레벨 : 신호를 부호화할 때 2진수로 표현할 수 있는 레벨

3.3. 신호와 주파수(1)

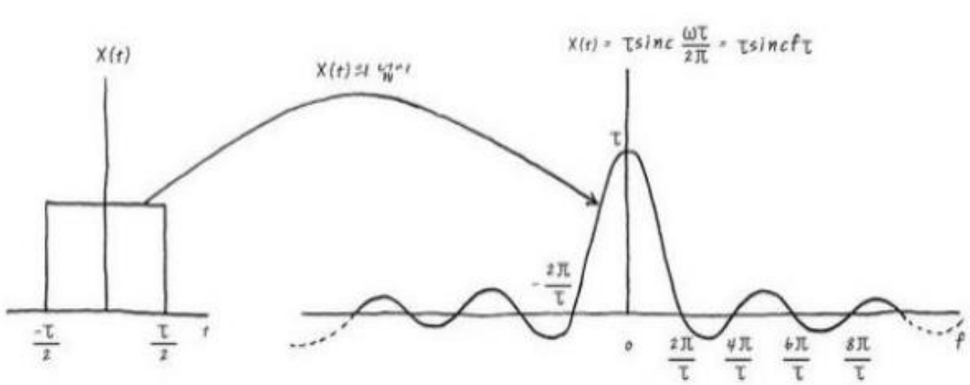


f=20kHz 일 때 cos(2πft) 함수의
시간 영역과 주파수 영역



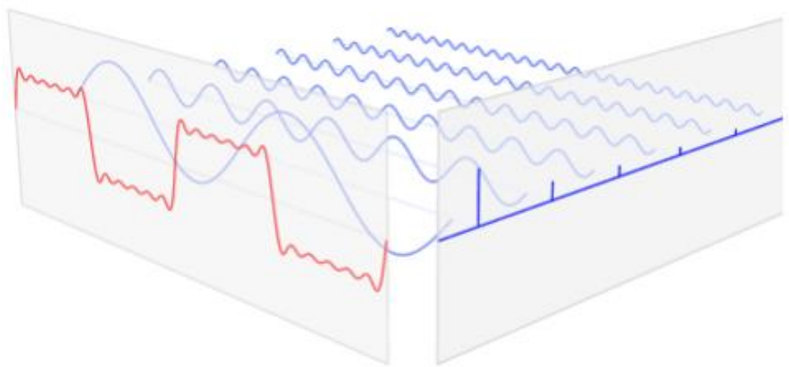
DC 신호의 시간 영역과 주파수 영역

3.3. 신호와 주파수(2)



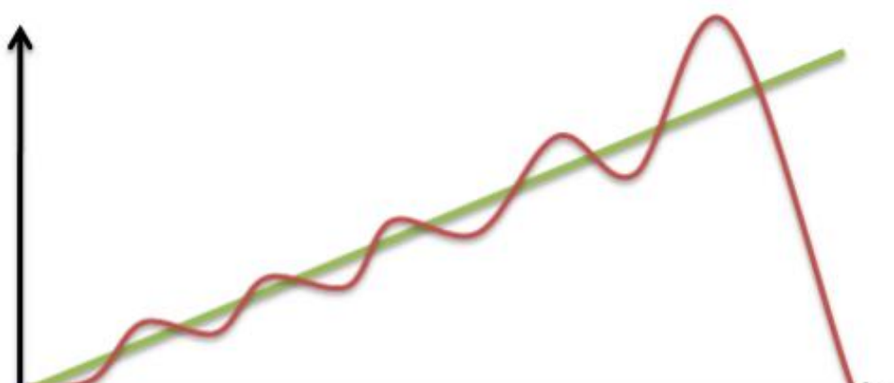
Rectangular 함수와

그 Fourier Transform인 sinc 함수



맨 앞의 붉은 색 신호는 입력 신호이고 뒤의 파란색 신호들은 푸리에 변환(Fourier transform)을 통해 얻어진 (원본 신호를 구성하는) 주기함수 성분들이다. 각각의 주기함수 성분들은 고유의 주파수(frequency)와 강도(amplitude)를 가지고 있으며 이들을 모두 합치면 원본 붉은색 신호가 된다.

3.4. Fourier transform (1)



y=x

$y=\frac{2}{1}\sin x-\frac{2}{2}\sin 2x+\frac{2}{3}\sin 3x-\frac{2}{4}\sin 4x+\frac{2}{5}\sin 5x-\frac{2}{6}\sin 6x+\ldots\sin 10x$

* 시간 도메인에서 어떤 연속적인 신호도 엄선된 사인 곡선의 합으로 나타낼 수 있다

3.4. Fourier transform (2)

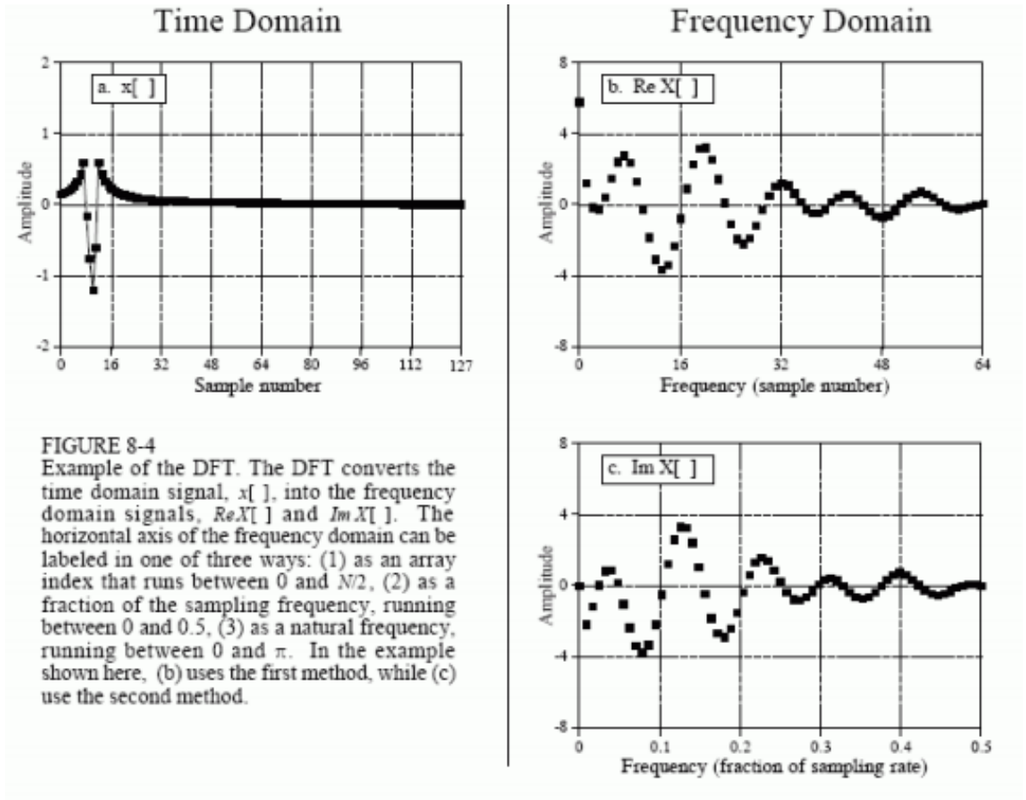
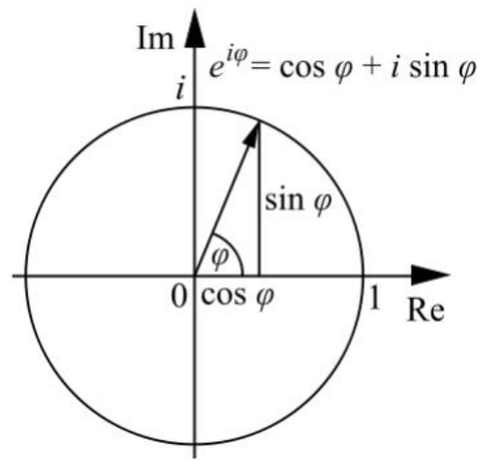


FIGURE 8-4
Example of the DFT. The DFT converts the time domain signal, $x[\ ]$, into the frequency domain signals, $Re X[\ ]$ and $Im X[\ ]$. The horizontal axis of the frequency domain can be labeled in one of three ways: (1) as an array index that runs between 0 and $N/2$, (2) as a fraction of the sampling frequency, running between 0 and 0.5, (3) as a natural frequency, running between 0 and π. In the example shown here, (b) uses the first method, while (c) use the second method.

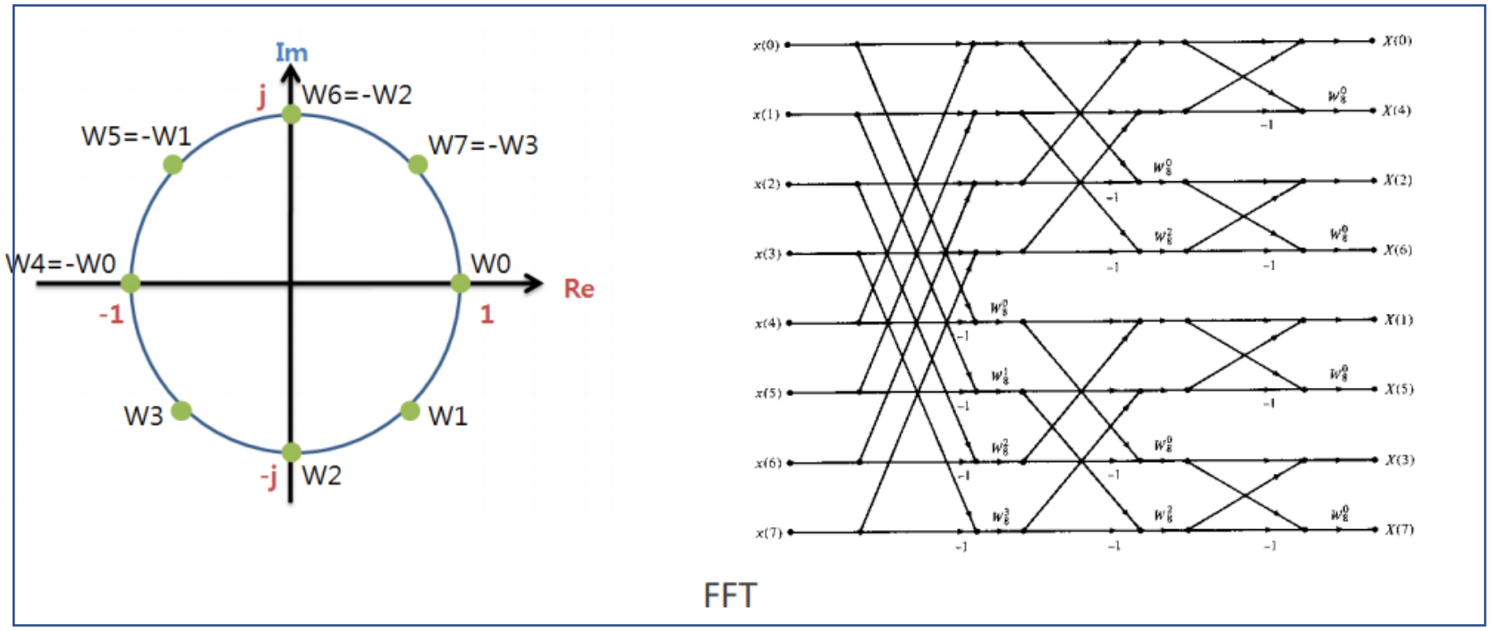
$$X(jw) = \int_{-\infty}^{\infty} x(t) e^{-jwt} dt$$



$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$

오일러 공식을 이용하면 $e^{j2\pi ux}$는 실수부가 $\cos(2\pi ux)$, 허수부가 $\sin(2\pi ux)$인 주기함수임을 알 수 있음.

3.4. Fourier transform (3)



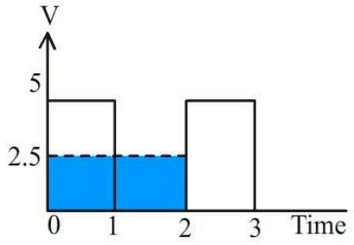
FFT

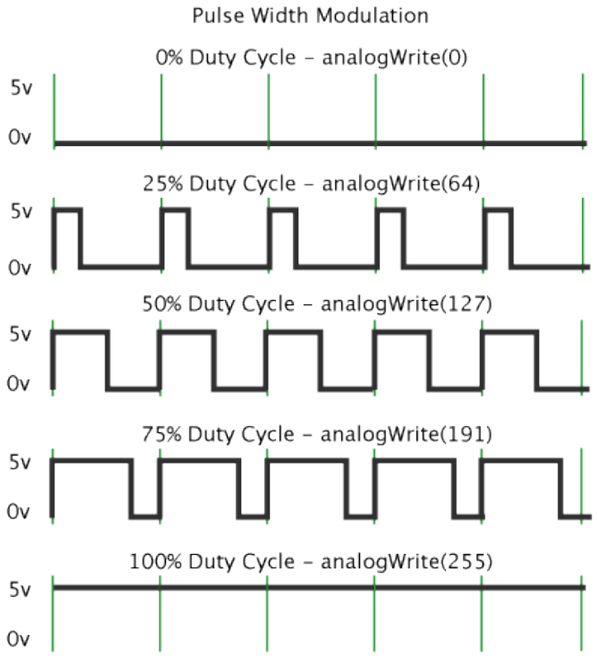
FFT: 연산횟수를 줄일 수 있도록 고안된 알고리즘. FFT는 1960년대 중반 J.W.콜리와 J.W.터키에 의해 일반적으로 알려지게 되었다.

N-point DFT의 복잡도: N^2
***FFT의 복잡도: N*Log(N)**

3.5. PWM (Pulse Width Modulation)

- We set the % of "high" cycle
  - 0 – 0%
  - 255 – 100%
    - Depends on the library
- Implementation
  - Hardware
  - Software
- Usage
  - LED dimming
  - Servo Motors





- ◦ ***듀티비 (Duty cycle)**
- ◦ ***주파수**

아두이노

디지털 출력: High or Low => 5V 또는 0V 한 가지만 출력

◦ DAC (Digital-analog converter) 지원 하지 않음

PWM (Pulse Width Modulation)

◦ 반복되는 펄스의 1주기 내에서 5V(High) 전압의 시간 폭과 0V(Low) 전압의 시간 폭 비율을 변경

◦ 평균 직류 전압의 크기를 변화시키는 방법으로 아날로그 출력 전압을 만드는 효과를 이용

3.6. ADC /PWM 아두이노에서의 사용

```
// setup() 함수가 호출된 이후, loop() 함수가 호출되며,
// 블록 안의 코드를 무한히 반복 실행됩니다.
void loop() {
        // 가변저항으로 측정된 값을 읽습니다.
        // 가변저항으로부터 입력되어지는 전압의 크기(0~5V)에 따라 0~1023 범위의 값으로 변환되어 반환합니다.
        int pmValue = analogRead(potentiometer);

        // 측정된 밝기 값를 시리얼 모니터에 출력합니다.
        Serial.print("potentiometer = ");
        Serial.println(pmValue);

        // 가변저항으로부터 측정된 값 (0~1024)을 아날로그 출력 값 (0~255) 범위로 변환합니다.
        pmValue = map(pmValue, 0, 1023, 0, 255);
        // LED가 연결된 핀으로, 변환된 값 만큼의 밝기로 켜지도록 설정합니다.
        // 만일 측정된 값이 500 이라면, 디지털 핀으로 출력 할 수 있는 최대값 255의 절반에 해당되므로,
        // LED가 5V 전류로 낼 수 있는 최대 밝기의 절반으로 해석 할 수 있습니다.
        // 이는 디지털로 아날로그 신호를 보내는 펄스 폭 모듈레이션(PWM)에서 duty-cycle이 50%인 것으로 설정됩니다.
        // 오렌지보드 디지털 핀의 PWM 주파수를 약 500Hz로 가정하면, 1초에 255번만 HIGH 신호를 보내는 것과 같습니다.
        analogWrite(led, pmValue);
        // 0.1초 동안 대기합니다.
        delay(100);
}
```

- 아두이노는 10비트의 ADC를 사용

- map 함수의 사용

- analogWrite -> PWM 출력 (8비트)

4. Electricity Equations

- **옴의 법칙**: 전류의 세기는 두 점 사이의 전위차에 비례하고, 전기 저항에 반비례한다

$$I = \frac{V}{R}$$

- **KCL (키르히호프의 전류 법칙):** 회로에 있는 어떤 마디에서도 모든 전류 대수 합은 0이다.

$$\sum_k i_k = 0$$

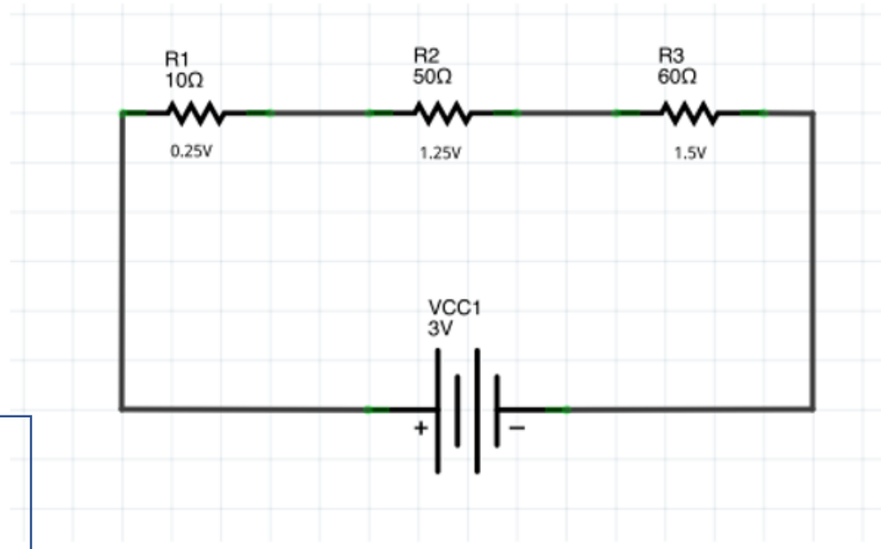
- **KVL (키르히호프의 전압 법칙):** 회로에 있는 어떤 폐경로 주위의 모든 전압의 대수 합은 0이다.

$$\sum_k V_k = \sum_k I_k R_k$$

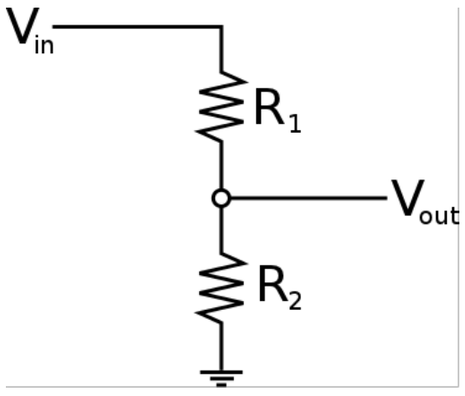
4.1. Ohm's Law

$$I = \frac{V}{R}$$

$$I = \frac{3V}{10\Omega + 50\Omega + 60\Omega} = 0.025A = 25mA$$



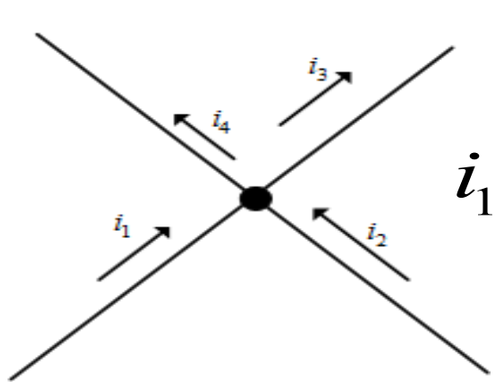
Voltage Divider



4.2. KCL

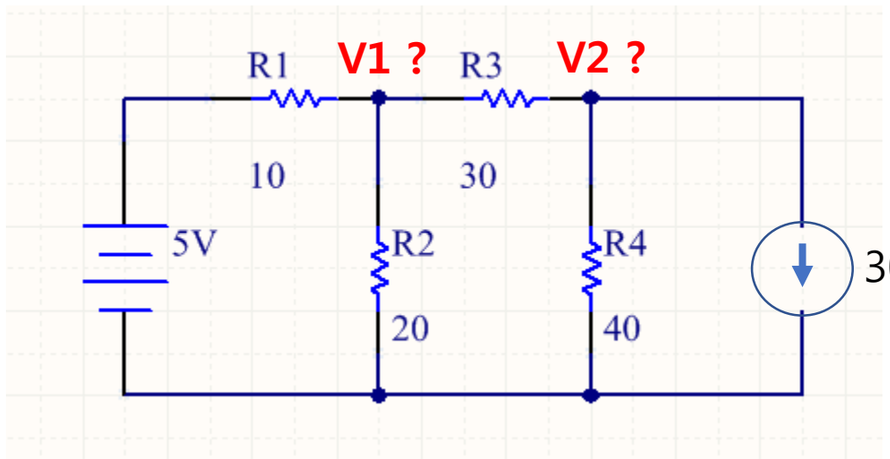
- **Kirrchhoff's current law (KCL)**
  - 회로에 있는 어떤 마디에서 모든 전류의 대수 합은 0이다.

$$\sum_{n=1}^{N} i_n = 0 \qquad (KCL)$$



$$i_1 + i_2 - i_3 - i_4 = 0$$

Q



4.3. KVL

- Kirrchhoff's voltage law (KVL)
  - 회로에 있는 어떤 폐경로 주위의 모든 전압의 대수 합은 0이다.

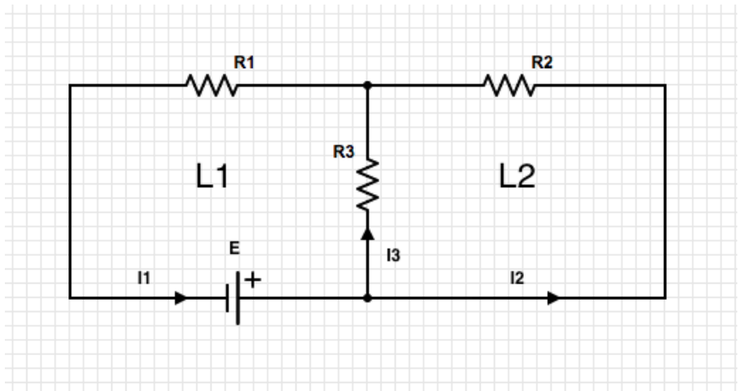
$$\sum_{n=1}^{N} v_n = 0 \qquad (KVL)$$

  - 전압 상승과 전압 강하는 서로 다른 부호 (일관된 규칙 필요)

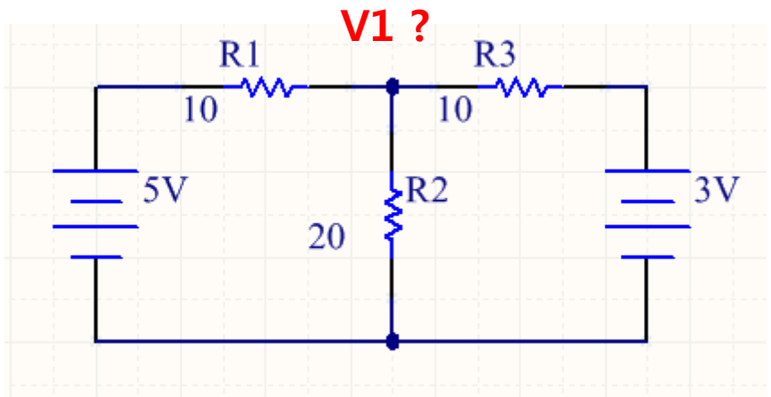
$$\sum_k V_k = \sum_k R_k I_k$$

$$L_1 : V = I_1 R_1 + I_3 R_3$$

$$L_2 : 0 = I_2 R_2 - I_3 R_3$$



Q

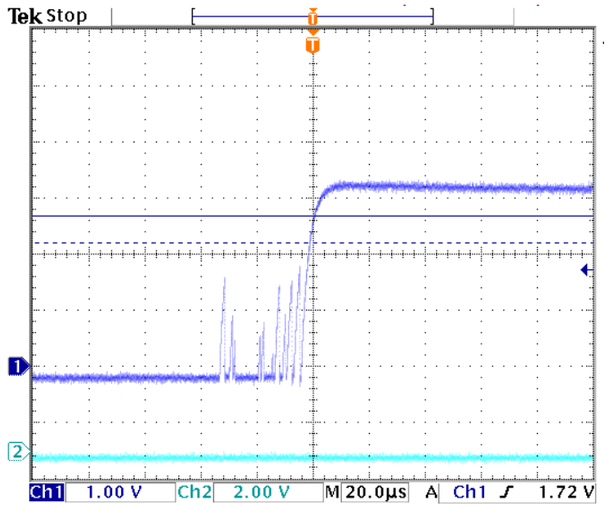


5. 센서와 직렬통신

- Analog 센서
  - 전압으로 출력
  - 데이터 신뢰도 낮음

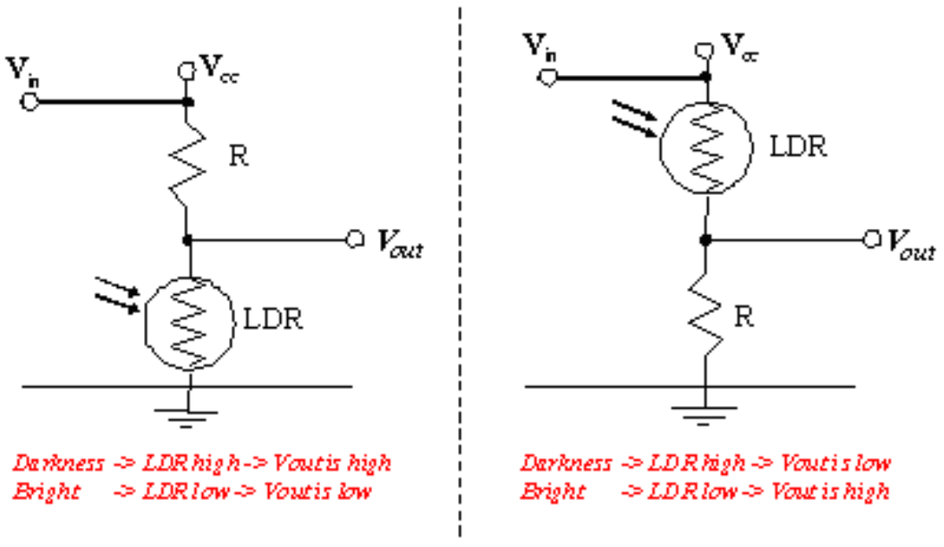
- Digital 센서
  - 직렬통신으로 출력
  - 데이터 신뢰도 높음

- Button
  - Analog 센서의 일종
  - 풀업/풀다운 구성하여 사용
  - Debounce noise (Chattering noise) 해결 -> **Low Pass Filter** 사용

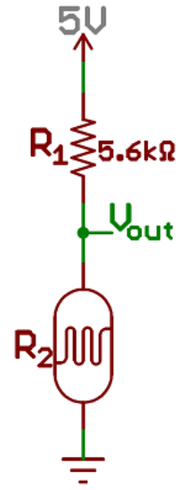


5.1. 조도센서

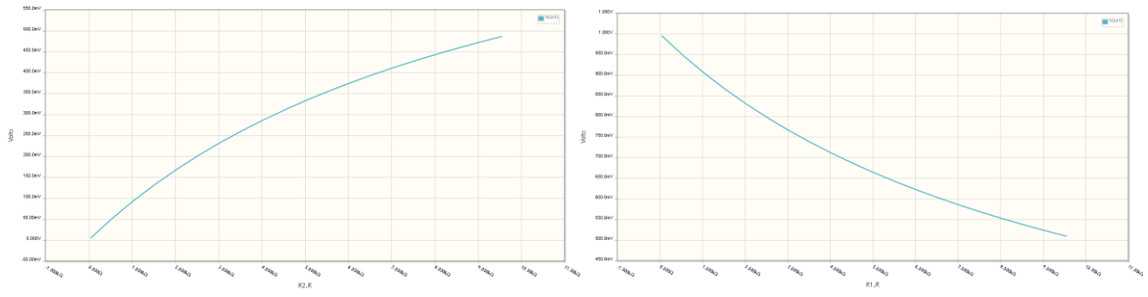
- 조도센서



두 구성의 차이점?



CdS센서(조도센서)의 데이터시트를 읽고, 전압->조도 변환하기



5.2. 온도센서

- **Thermistor**
  - Resistance

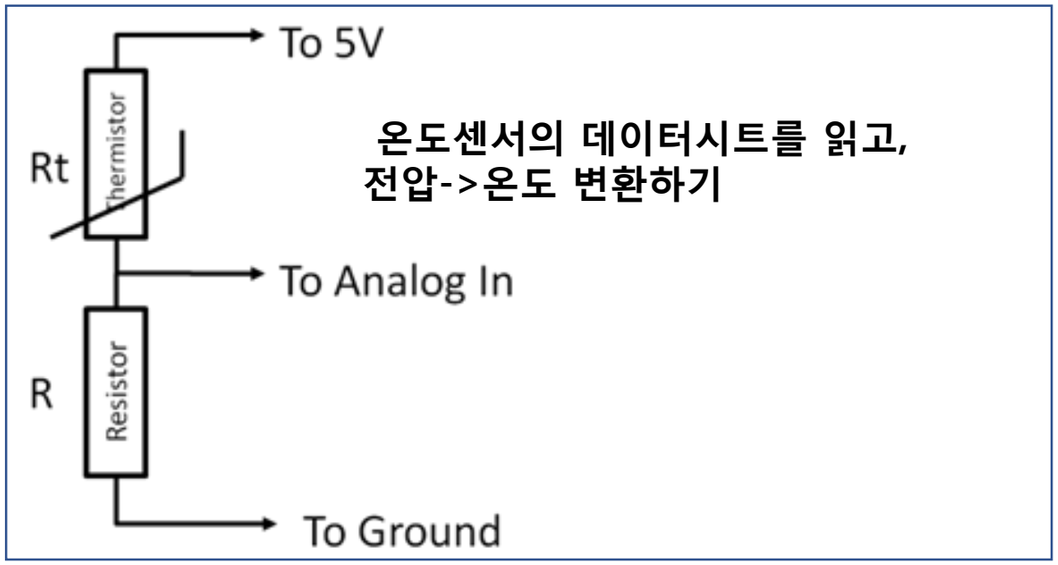
- Parameters
  - R_{25} resistance - 25 degrees
  - B – constant

$$V_{measured} = pin\frac{5}{1023}$$

$$R_T = \frac{5}{V_{measured}}R$$

$$ratio = \frac{1}{B}\ln(\frac{R_T}{R_{25}})$$

$$Temperature = \frac{1}{\frac{1}{298} + ratio}$$



온도센서의 데이터시트를 읽고,
전압->온도 변환하기

- **LM35**

$$V_{measured} = pin\frac{5}{1023}$$

$$Temperature = V_{mesured} * 100$$

5.3. 거리센서

초음파센서의 Timing Diagram을 읽고, 거리를 유추

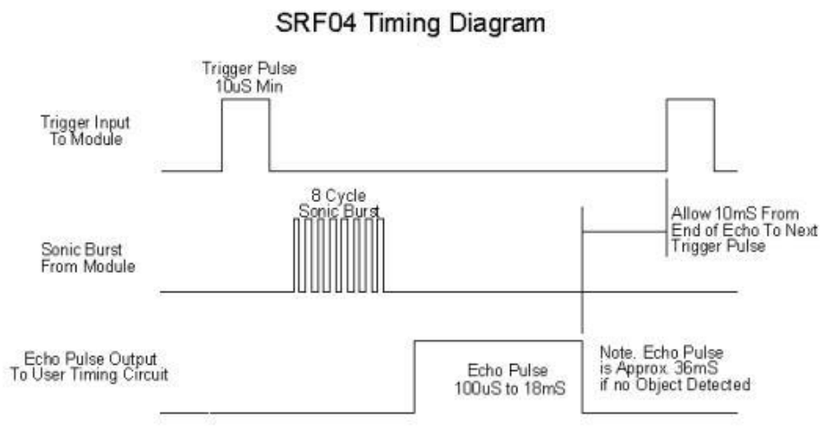
- 거리는 Echo Pulse의 길이

PSD센서의 데이터시트를 읽고, 거리를 유추

- 거리는 전압

• 초음파 센서

- Ultrasonic
- Sends a pulse
- Real time system
- Works on microcontrollers



• PSD 센서(적외선 거리센서)

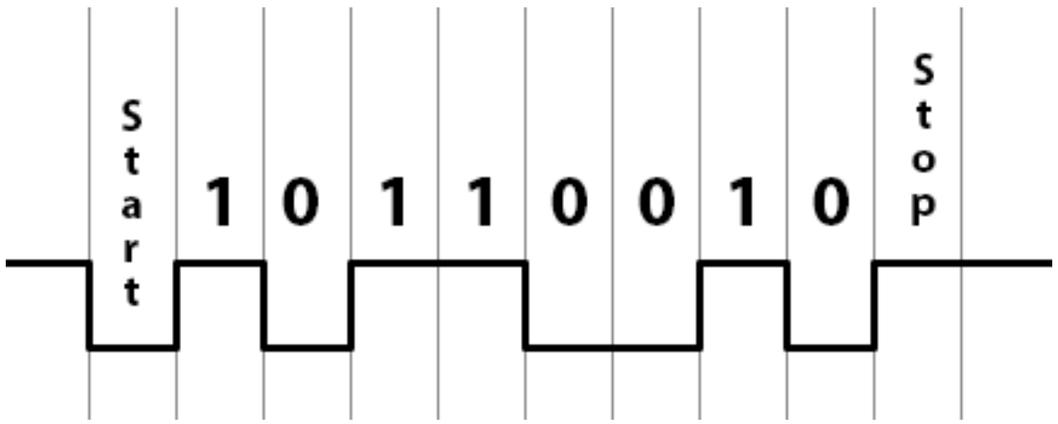
- Analog
- Measure voltage

5.4. UART

- Universal Asynchronous Receiver/Transmitter
  - Universal : 여러 통신 규약과 함께 사용됨
    - RS-232C, RS-422, RS-485 등
  - **Asynchronous** : 동기화를 위한 별도의 클록을 사용하지 않음

- **저수준**의 통신 방법
  - 하드웨어 수준에서 지원
    - 통신을 위한 전용의 하드웨어 핀 제공
- 마이크로컨트롤러의 동작 전압을 기준으로 하는 TTL (Transistor-Transistor Logic) 레벨 사용

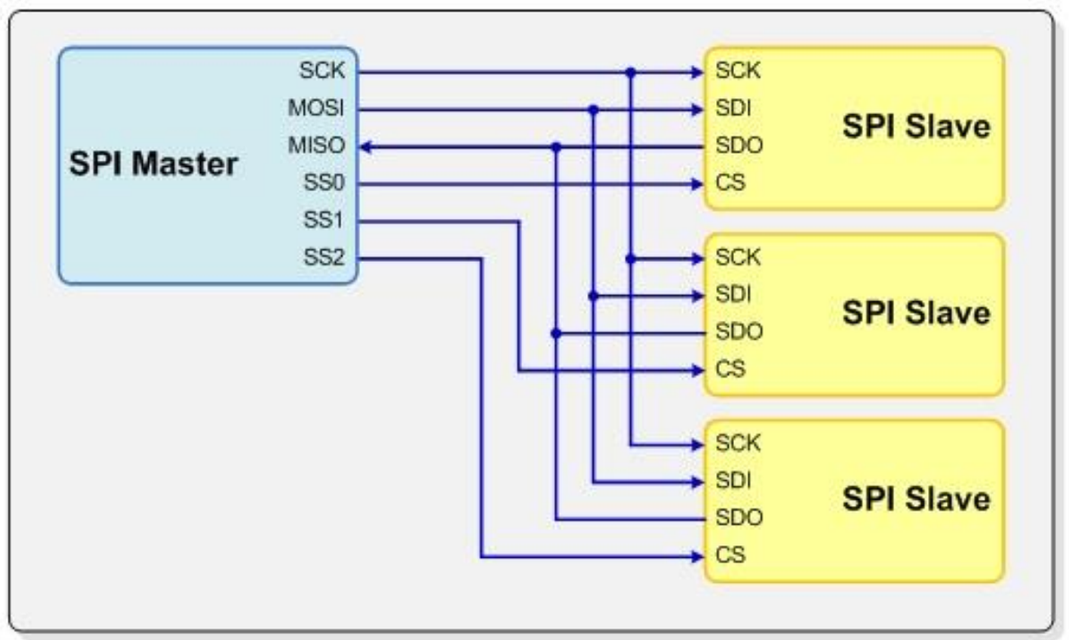
TTL(Transistor-Transistor Logic)
바이폴라 트랜지스터를 사용하여 만든 디지털 로직.
대부분 +5V 전원전압에서 동작하며, 속도가 빠른 반면에 소비전력이 크다는 특징을 가짐



***Parameters**
- bps
- Data bit
- Start bit
- Stop bit
- Parity bit

5.5. SPI 통신

- SPI(Serial Peripheral Interface) 통신
  - SPI 인터페이스는 두 개 이상의 주변장치 간에 데이터 통신을 목적으로 모토롤라에서 개발한 직렬 통신 방식
  - 1:N 연결을 지원하고 Master-slave 모드로 통신
  - SPI 통신을 위해서는 MISO와 MOSI, SCK, SS를 사용 (4-wire)
  - SPI 통신은 SS를 이용하기 때문에 같은 주소 충돌이 발생하지 않지만, 연결하는 장치수가 늘어날 수록 연결선의 수가 늘어남
- 아두이노 SPI 통신
  - SPI통신을 위한 라이브러리를 제공
  - 아두이노 SPI 라이브러리는 마스터 모드만 지원
  - 슬레이브를 설정하실 경우는 직접 레지스터를 통해 제어



- **MOSI (Master Out Slave In)**
- **MISO (Master In Slave Out)**
- **SCK (Serial Clock)**
- **SS (Slave select)**

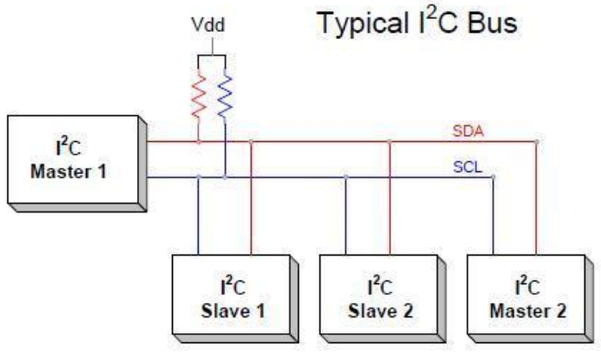
5.6. I2C (1)

- **I2C** 인터페이스는 **IC(집적 회로)** 사이에 데이터 통신을 목적으로 필립스에서 개발한 직렬 통신 방식

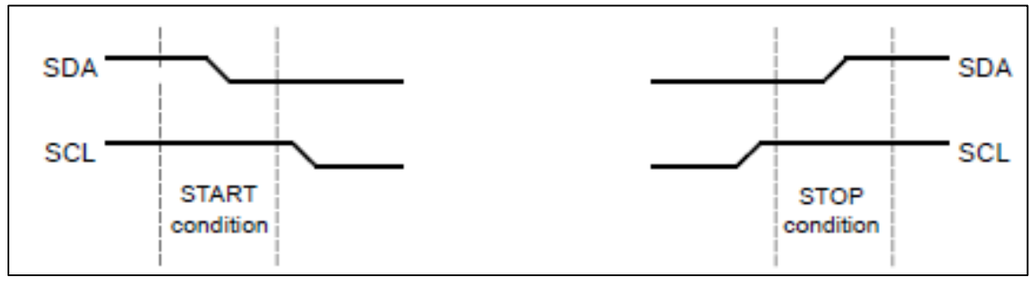
  - I2C 통신은 1:N연결을 지원하고 Master-Slave 모드로 통신
  - I2C 통신은 SPI와 다르게 슬레이브 선택을 위해 소프트웨어적인 주소를 사용하므로 연결하는 장치가 늘어나도 사용하는 핀이 증가하지 않음
  - I2C 통신은 데이터 송수신을 위한 SDA와 동기화 클록을 위한 SCL 핀 2개만 사용 (2-Wire)
  - UART나 SPI와 비교할 때 속도의 한계가 있으므로 사용에는 제한
  - 그러나, 아두이노에서는 SPI와 달리 Wire라이브러리에서 슬레이브 모드를 지원하고, UART와 달리 1:N통신이 지원되므로 비교적 간단하게 다양한 기능 구현이 가능

5.6. I2C (2)

- I2C 통신 인터페이스
  - 두 회선에 풀업저항 연결



- I2C 통신 동작
- 스타트 조건
  - 마스터가 슬레이브에게 '통신을 시작하겠다'라는 의미의 상태로 SCL의 상태가 High일 때, SDA의 상태가 High에서 Low로 바뀜으로써 슬레이브는 통신 시작을 감지
- 스톱조건
  - 마스터가 슬레이브에게 '통신을 종료하겠다'라는 의미의 상태로 SCL의 상태가 High일 때, SDA의 상태가 Low에서 High로 바뀜으로써 슬레이브는 통신 종료를 감지



5.7. 직렬통신방식 비교

	UART	SPI	I2C
동기 / 비동기	비동기	동기	동기
전이중 / 반이중	전이중	전이중	반이중
통신 회선 (n개 슬레이브 연결)	2n	3+n	2
연결 방식	1:1	1:n (마스터-슬레이브)	1:n (마스터-슬레이브)
슬레이브 선택		SS 신호 선 (하드웨어)	SDA선으로 전송되는 슬레이브 주소 (소프트웨어)

6. 무선 네트워크

2세대 디지털 전환 이후 빠르게 진화

- 전송속도, 커버리지, 무선대역폭활용, 저전력 등 비약적 발전

- LTE (3GPP-Long Term Evolution)
  - 가장 대표적 무선시스템으로 마켓을 석권함
  - 통신사업자가 주파수를 획득하여 유료로 서비스함
- WiMAX (IEEE802.16e/m)
  - IEEE를 중심으로한 LTE의 경쟁기술로 마켓의 열세에 의해 도태됨
- WiFi (IEEE802.11)
  - Hot Spot의 무선인터넷 환경을 위한 기술로 LTE의 보완제로 자리매김함
  - 개인사용자가 AP를 설치하며 비면허대역을 이용하여 무료 서비스함
- Bluetooth (IEEE802.15)
  - 근거리 무선기기들의 상호 접속 용도로 사용
  - 무선 헤드셋, 키보드, 마우스 등

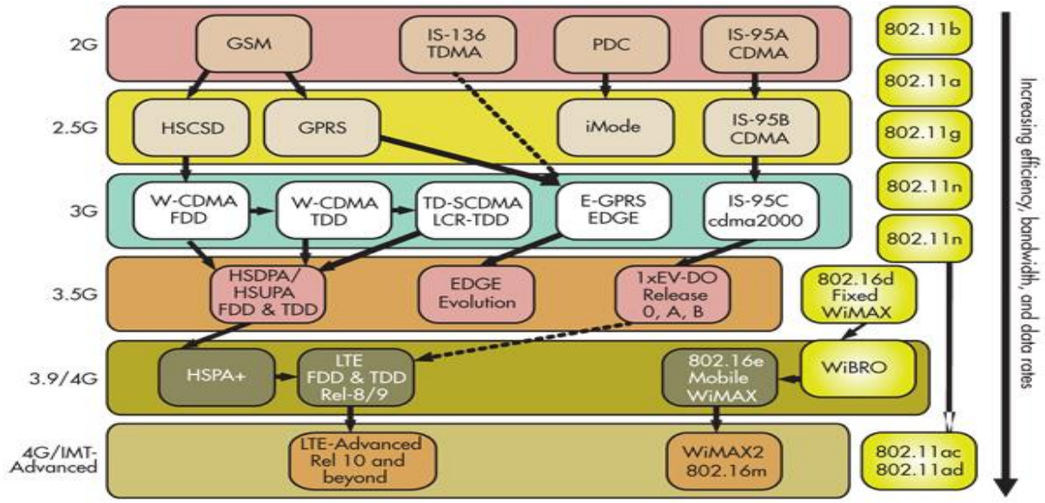


Fig1. In the evolution of wireless cellular standards, all paths eventually lead to some form of LTE or a WiMAX alternative. (courtesy of Agilent Technologies)

6.1. LTE vs WiFi

- **LTE**
  - 가격
    - LTE와 같은 셀룰라 통신은 통신사업자가 제공하는 유료 서비스임
    - 국가로부터 전용 주파수를 경매등의 절차로 비싸게 임대하여 사용함
    - 기지국 등 통신망을 구축하는 비용이 비쌈
  - 속도
    - 무선통신의 속도는 사용하는 주파수 대역폭에 비례함
    - 전용 면허대역은 주파수 대역폭이 적기 때문에 높은 속도 달성이 어려움
- **WiFi**
  - 가격
    - 비면허 **ISM** (industrial/scientific/medical) 밴드를 활용하여 주파수 사용이 무료
    - 기지국 역할의 WiFi AP는 개인이 자유롭게 설치할 수 있으며 가격이 저렴함
  - 속도
    - 면허대역에 비해 상대적으로 넓은 주파수 대역폭 확보가 가능하여 속도 빠름
    - 고정환경이라는 특수성으로 속도를 높일 수 있는 다양한 응용기술 활용가능함

6.2. WiFi

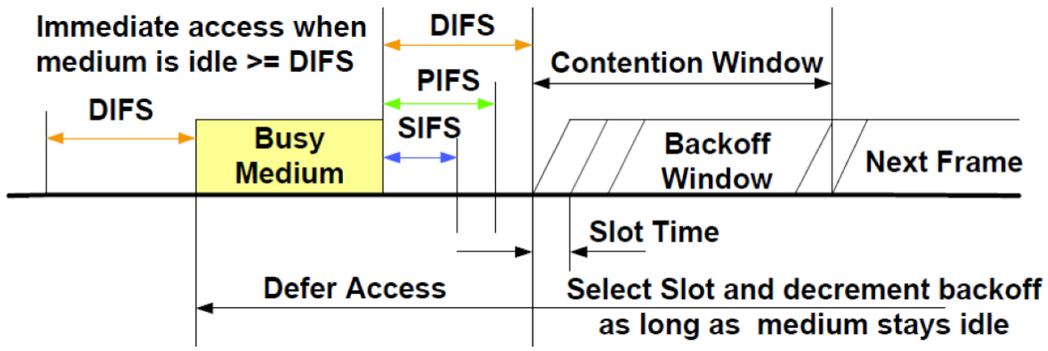
- WiFi
  - 무선랜 (Wireless LAN)으로 개발됨
  - Wireless Fidelity에서 약자가 브랜드명이 됨

- IEEE
  - Institute of Electrical and Electronics Engineers
  - 전세계 전기전자 전문가들이 활동하고 있으며 다양한 표준을 정의함
  - 대표적으로 IEEE802.3 (Ethernet 표준), IEEE802.11 (WiFi 표준)이 있음

- WiFi Alliance
  - WiFi의 인증/마케팅을 담당하는 단체
  - IEEE에서 개발된 기술표준을 상용화/제품화를 담당함

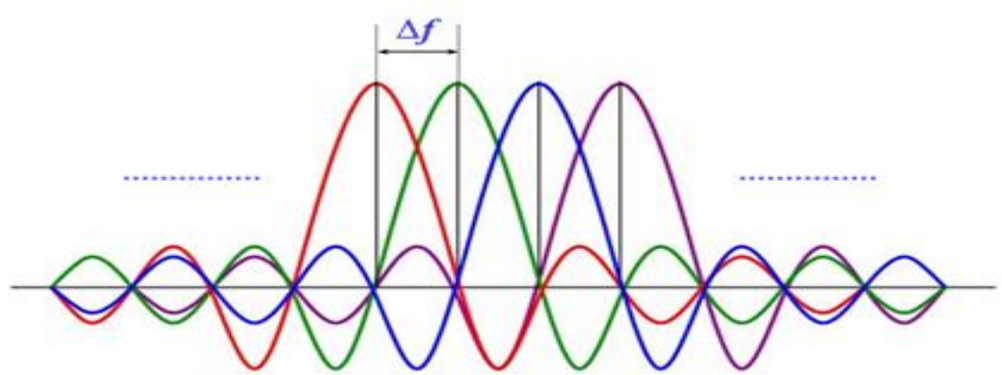
6.3. WiFi 기초기술

- **CSMA/CA**
  - Carrier sensing multiple access collision avoidance
  - 시분할 방식의 packet 전송방식
  - **Listen-before-talk**의 상호 충돌 회피 방식이 특징

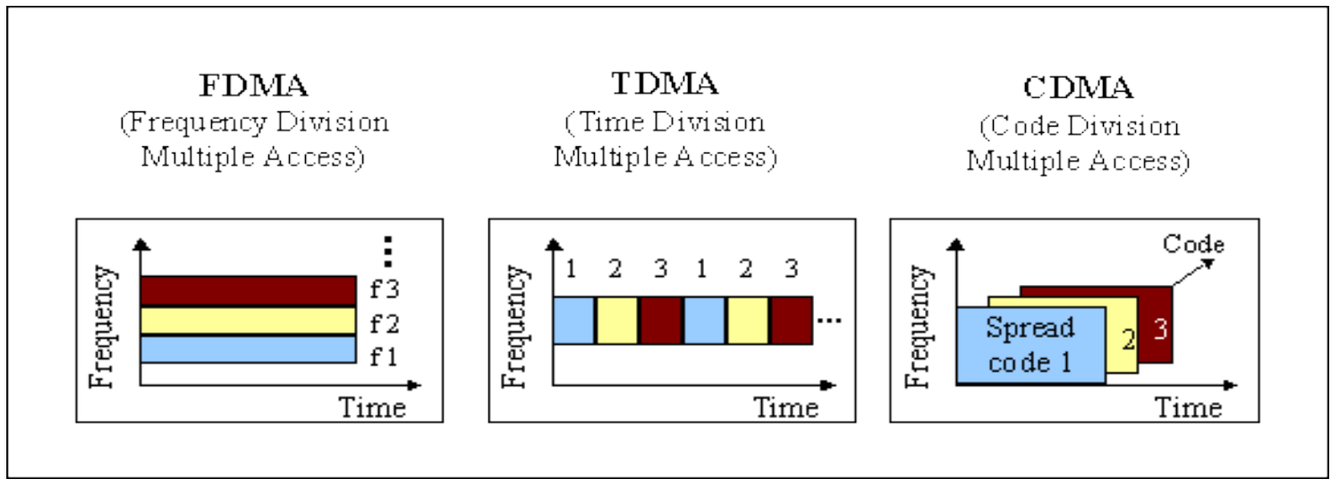


OFDM

- Orthogonal frequency division multiplexing (직교분할다중화)
- 넓은 주파수 대역을 다수의 반송파로 나누는 효율적 전송기법
- 주파수 선택적(frequency-selective) 페이딩에 강함
- FFT/IFFT를 이용한 송수신 신호처리의 단순화



6.4. 다중 접속 방식



•Frequency-Division Multiple Access
할당된 주파수를 여러 개의 채널로 분할

•Time-Division Multiple Access
할당된 주파수를 작은 시간단위로 분할하는 시분할 방식으로 사용자는 자신에게 주어진 시간에 주파수를 독점

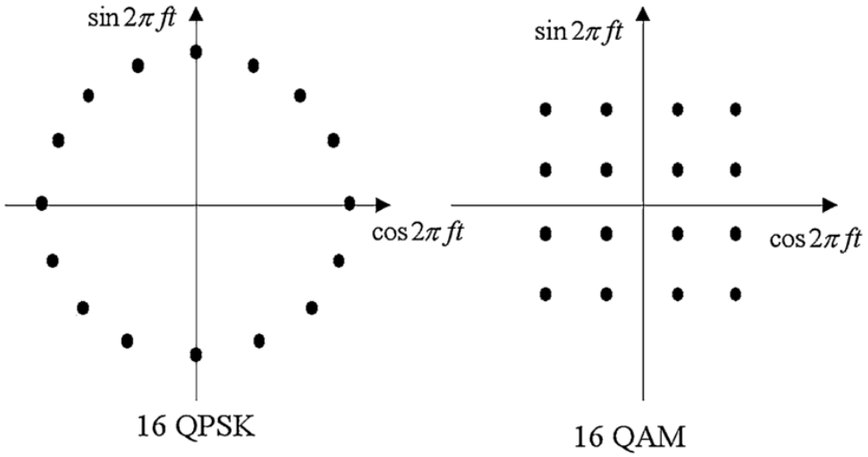
•Code-Division Multiple Access
내용을 암호화해서 가용할 수 있는 주파수 대역 전체로 확산

6.5. 변조

BPSK (Binary phase-shift keying)
- 한 신호에 0 또는 1을 표현

QPSK (Binary phase-shift keying)
- 위상을 등분하여 정보를 표현

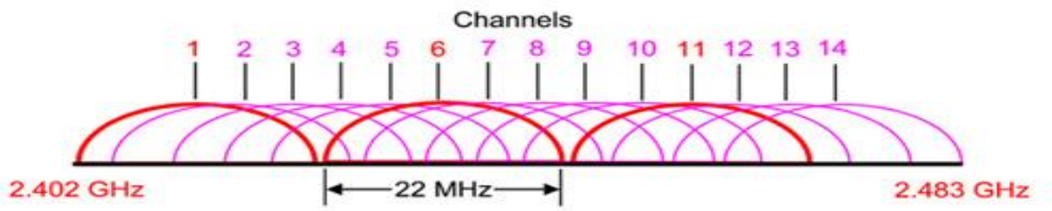
QAM (Quadrature Amplitude Modulation)
- 위상 + 진폭으로 데이터를 표현



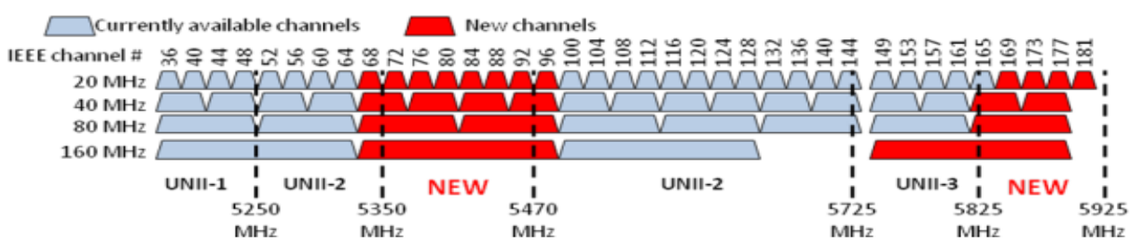
*

6.6. Channel

- 2.4GHz ISM band
  - 14개의 overlapping 채널 (Ch 1, 6, 11 사용시 간섭 없음)
  - 인접 AP 간 간섭, Bluetooth 기기, 전자레인지 등 간섭 문제

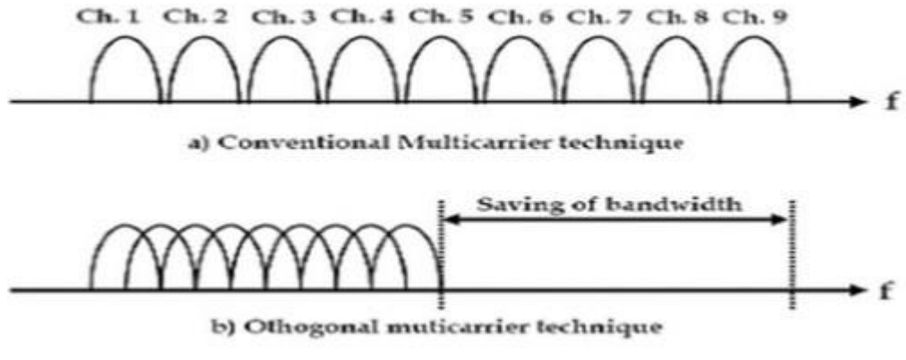


- 5GHz ISM band
  - 사용 기기가 적어 상대적으로 쾌적한 무선환경
  - 넓은 대역폭으로 인한 고속전송에 유리함 (전파투과성은 불리)



6.7. OFDM

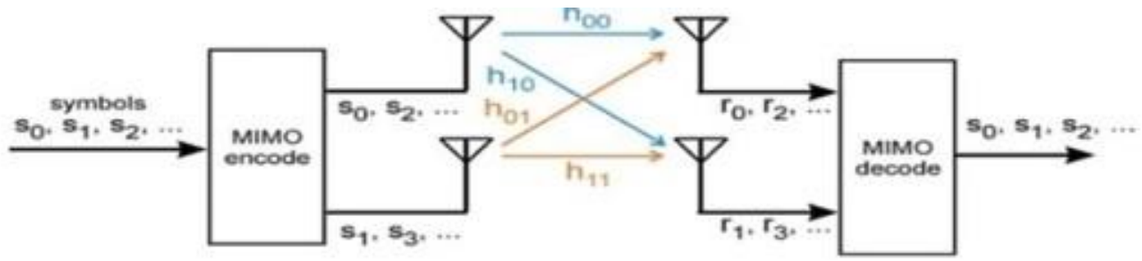
- 주파수의 직교성을 유지하면서 시스템은 효율적으로 데이터 전송가능



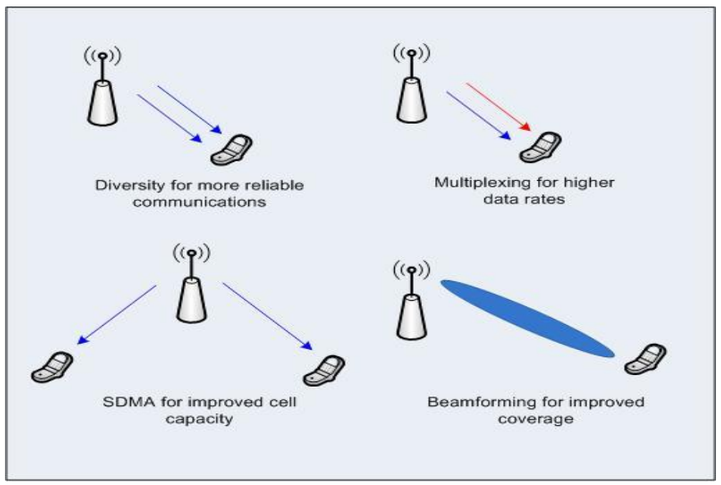
전송 방식	시스템 측면	사용자 측면
OFDM고속 대용량화 과정	서브캐리어(Ch1, Ch2..)가 여러 개 됨으로 대용량이 됨	사용자는 서브캐리어를 동시에 여러 개를 받음으로 고속 대용량화함
OFDM 장점	주파수 효율이 좋음 Multipath fading 강함 도플러 효과 좋음 협대역 간섭에 좋음 고속 구현이 용이하다 임펄스 잡음에 강함 셀 간섭 적음	부채널 주파수 중첩 데이터 병렬 전송 도플러 천이 효과발생 주파수 선택적 페이딩 효과 FFT이용 고속의 신호처리 복잡도 크게 감소됨 부반송파간 직교성 유지

6.8. MIMO / OFDM (1)

- **MIMO**
  - Multiple input multiple output (다중송수신 안테나)
  - 다수개의 안테나를 이용한 공간다중화로 전송율/신호대잡음비 향상
  - 시스템(송신)측면에서는 직렬로 데이터를 보내고, 수신(단말)측에서는 병렬로 하여 수신



전송 방식	시스템 측면	사용자측면
MIMO 고속 대용량화 과정	데이터를 S0, S1, S2를 차례대로 전송 송신측에서 S0, S2 전송 송신측에서 S1, S3 전송하여 고속 대용량화 함	수신측에서 r0, r2를 수신 수신측에서 r1,r3를 수신 병렬로 수신함으로써 고속 대용량화 함
MIMO 장점	안테나 병렬 전송	안테나별 데이터 다중화



6.8. MIMO / OFDM (2)

- LTE의 핵심 기술요소
- MIMO와 OFDM을 같이 사용하여 주파수간 간섭을 제거하여 고속 대용량화 가능

다중접속기술 MIMO	직교주파수 OFDM
다중 입력 다중 출력 다중 접속 가능 스마트 안테나 기술 LTE의 핵심기술	부반송파간의 직교성 유지 넓은 대역이 필요 대역 중첩 전송 대역폭 효율 고속전송 가능
4G, LTE, LTE-A	HDTV, DMB, 와이브로

6.9. 네트워크 형태

- **인프라 방식**
  - AP(access point)나 무선랜 공유기에 여러 대의 클라이언트가 접속해 네트워크를 구성하여 통신하는 방법
- **애드혹(Ad-Hoc) 방식 또는 피어 투 피어(Peer-to-Peer) 방식**
  - 각 클라이언트가 AP나 무선랜 공유기 없이 서로 접속해 네트워크를 구성해 통신하는 방법

- STA(Station)
  - WiFi 단말(WiFi 인터페이스를 가진 단말)을 의미
- AP(Access Point)
  - 802.11 무선랜 인터페이스와 802.3 이더넷 인터페이스를 가지는 장비로, STA가 보낸 데이터를 무선으로 받아 이더넷 포트를 통해 유선망으로 보내주는 장비

6.10. OSI 7계층, TCP/IP 4계층

1. 물리계층
- 전기적 성질과 시간 간격, 비트들이 매체를 통화하는 방법, 물리적인 구조화, 커넥터의 크기와 모양 등

2. 데이터 링크 계층
- 물리계층에 연결된 소프트웨어 인터페이스
- 네트워크 계층을 위해 "신뢰성 있는 통신"을 제공
- 링크에 대한 데이터 구조화, 손상여부 판단하는 체크섬

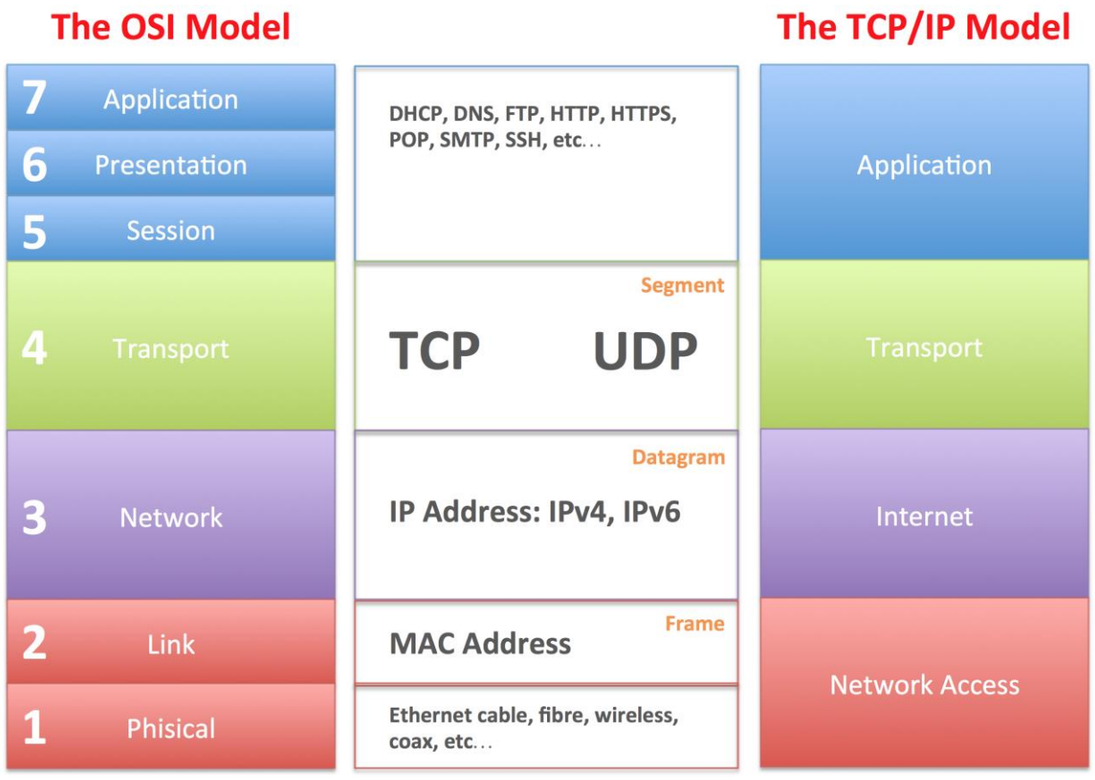
3. 네트워크 계층
- 노드간 패킷 전달
- 주소 할당, 경로 선택, 단편화와 조립, 혼잡과 흐름 제어

4. 전송 계층
- 신뢰성 있는 종단 대 종단간 통신
- 손상, 패킷 유실, 순서가 바뀌는 등 오류를 다루어 하위 계층의 신뢰성을 보완

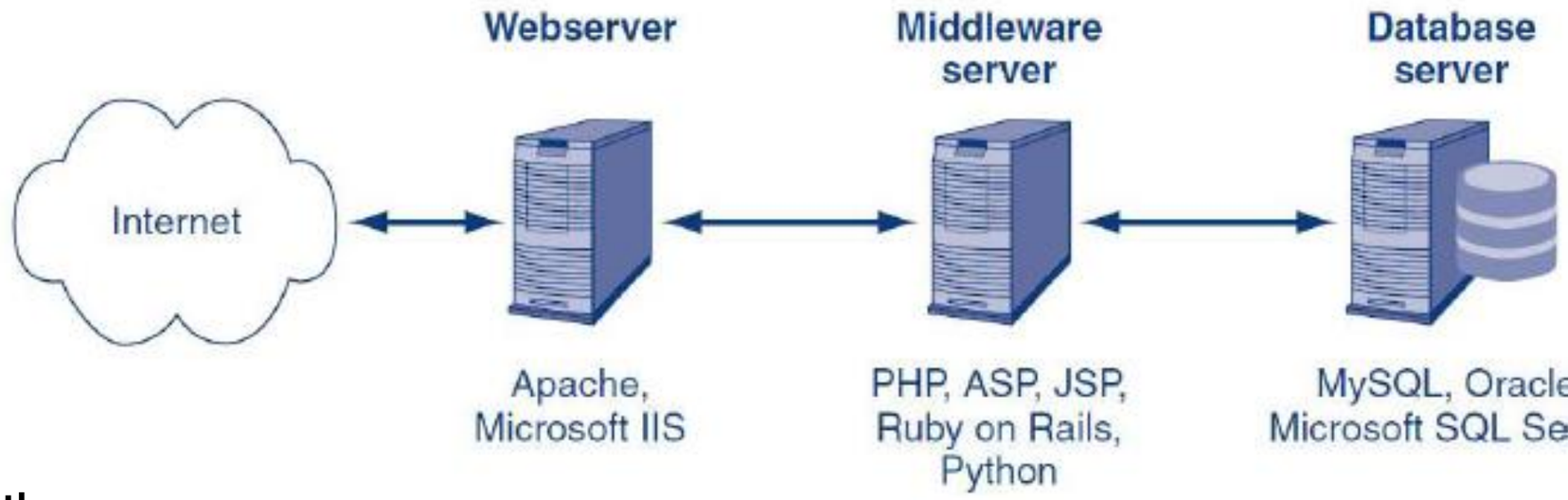
5. 세션 계층
- 컴퓨터들 간의 전이중의 신뢰성 있는 통신 스트림 제공

6. 표현 계층
- 데이터 해석
- 데이터 표현, 압축들 간의 변화 조절

7. 응용 계층
- 아래 6계층을 이용하는 사용자 프로그램



7. Web

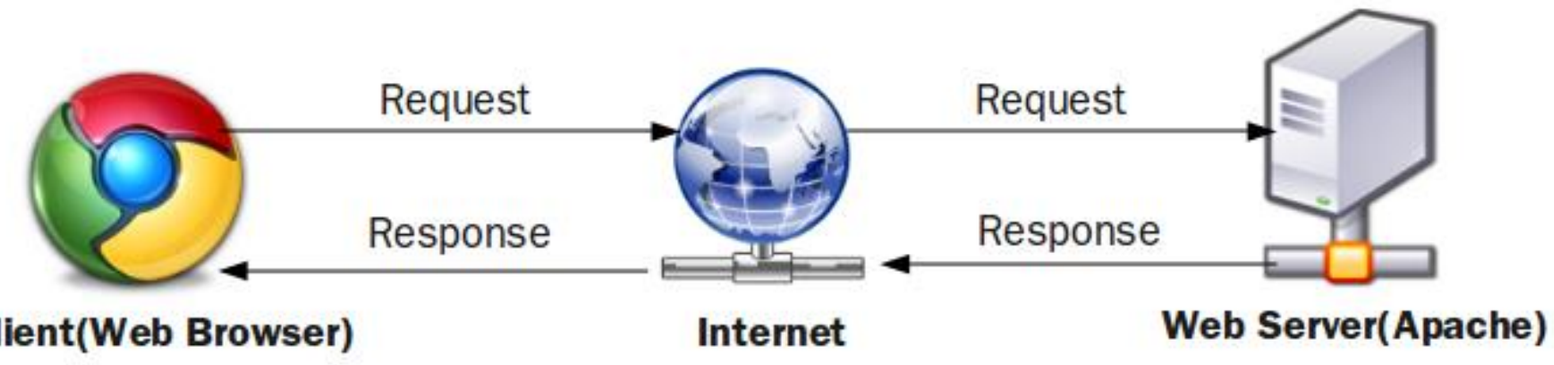


웹서버

- HTML로 구성된 웹 페이지를 제공해주는 서버
- **웹 브라우저** 간의 데이터통신을 위해 필요한 기술

HTML : Hypertext Markup Language

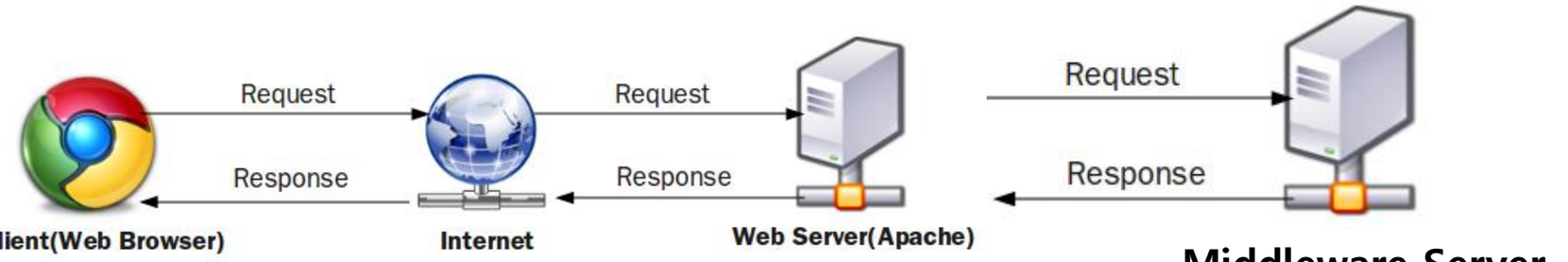
7.1. Web server



웹 서버

- 웹 브라우저의 요청이 있는 경우 가장 처음 이를 받아들여 요청한 페이지를 응답
- **HTML**로 구성된 웹 페이지를 제공
- **정적인** 웹 페이지 제공
- Apache HTTP서버(무료) / 마이크로소프트 ISS

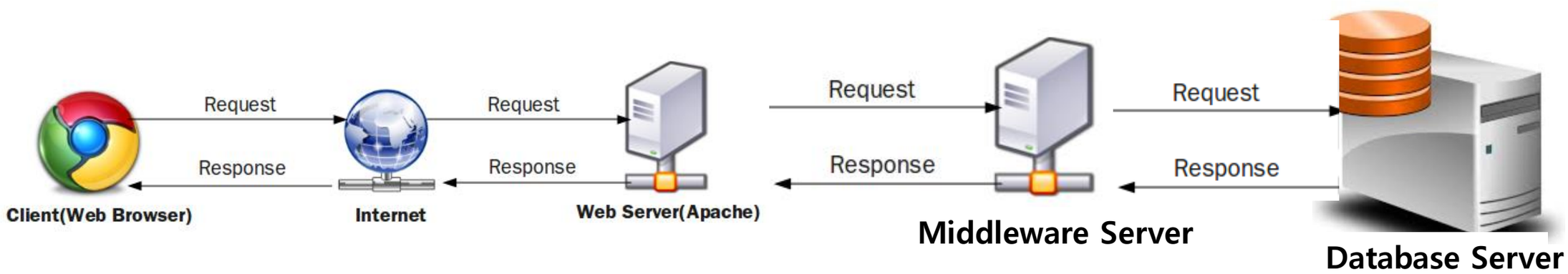
7.2. Middleware Server



미들웨어 서버 (Web Application Server)

- 웹서버와 웹 컨테이너의 결합으로 다양한 기능을 수행,
- **자바스크립트/PHP**등으로 구성된 웹 페이지를 제공
- **동적인 웹 페이지** 제공
- JSP/PHP/Python 등의 어플리케이션 사용
- DB와 연결되어 데이터 요청 및 전달 가능

7.3. Database Server



데이터베이스(DB) 서버

- Middleware Server의 명령 하에 요청된 정보를 제공
- 데이터의 **검색과 보관**
- Oracle/MySQL 등 사용

7.4. HTTP (Hypertext Transfer Protocol)

- 웹 서버와 클라이언트(브라우저) 사이의 문서를 전송하기 위한 규약
  - 손쉽게 사용 가능
  - Text based의 하이퍼텍스트 문서
    - 문자
    - 링크
    - 그림
  - HTML(Hypertext Markup Language)을 주로 사용
    - 하이퍼텍스트 문서를 만들기 위한 프로그래밍 언어의 일종
    - 간단히 설계 가능한 UI
    - 별도의 컴파일이 필요 없이 웹 브라우저에서 바로 해석 가능

Query 문자열은 URL에서 ? 뒤에 붙는 값을 의미

- http://test.com/projects?show=true
  - Address : test.com
  - Port : 80 (Default)
  - URL: /projects
  - Querystring : show=true

7.5. HTTP 요청과 응답 (1)

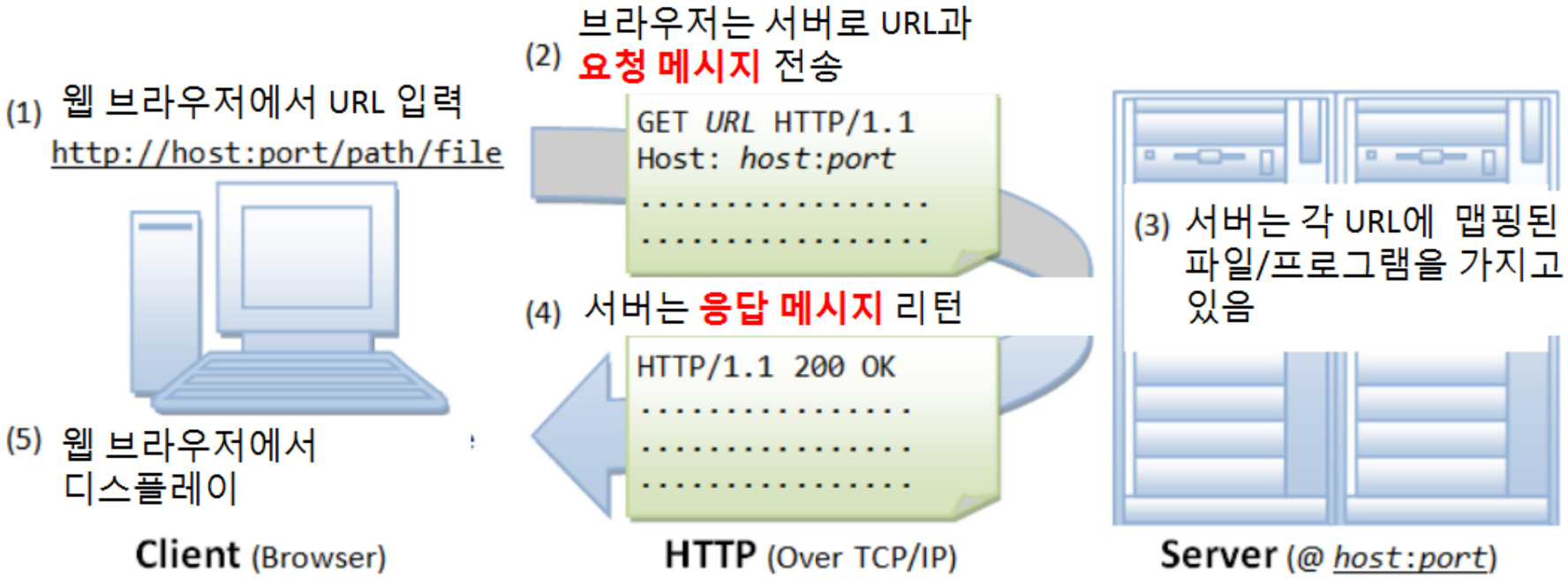


Image from https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

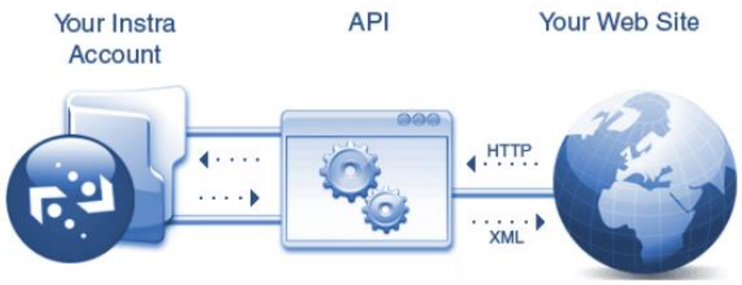
7.5. HTTP 요청과 응답 (2)

- Request (요청)
  - GET
    - 가장 일반적인 HTTP Request 형태
    - URL에 각 이름과 값을 &로 결합
  - POST
    - HTTP 헤더에 오브젝트를 전송
    - URL을 통해 인수를 전송하지 않음
  - PUT
    - 새로운 데이터를 저장
  - DELETE
    - 오브젝트를 제거

Response (응답)

- Status
  - 실행결과 코드
    - 200번대 -> 성공
    - 300번대 -> 리다이렉션 (추가 동작 필요)
    - 400번대 -> 클라이언트 측 에러
    - 500번대 -> 서버 측 에러
- Headers
  - 전달할 데이터의 형식과 길이
- Data
  - 실제 데이터(HTML)

7.6. Web service (API)(1)



- Key를 가지고 있는 사람만이 가져갈 수 있도록 정보를 제공

- Application Programming Interface
- 프로그램이나 어플리케이션이 정보처리를 위해 운영체제에 호출하는 함수나 서브루틴의 집합

7.6. Web service (API)(2)

- XML HTTP
- 일반적으로 API라고 지칭
- 메시지 지향
- XML 유니코드 사용
- Application을 위한 기능 수행
  - 모바일 App에 적용 가능
  - 여러 UI 사용 가능

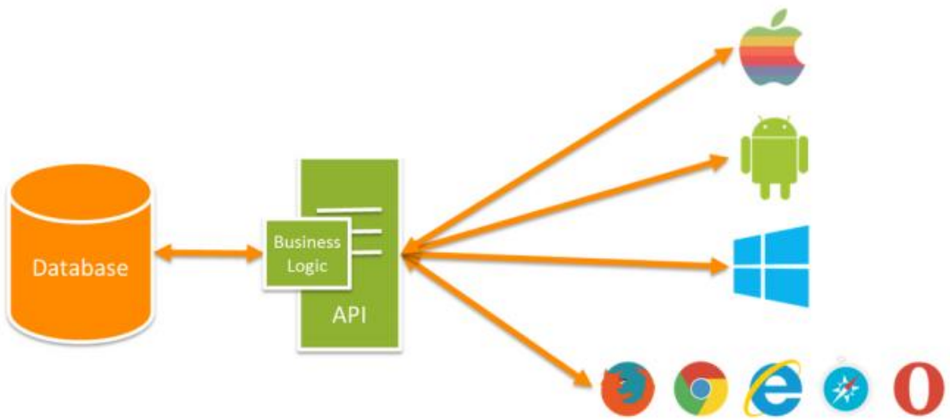


Image from http://blogs.msdn.com/b/martinkearn/archive/2015/01/05/introduction-to-rest-and-net-web-api.aspx

지역 변경

```
// send the HTTP PUT request
client.print("GET /data/2.5/weather?q=Seoul,kr&appid=");
client.print(VARID);
client.println(" HTTP/1.1");
client.println("Host: api.openweathermap.org");
client.println("Connection: close");
client.println();
```

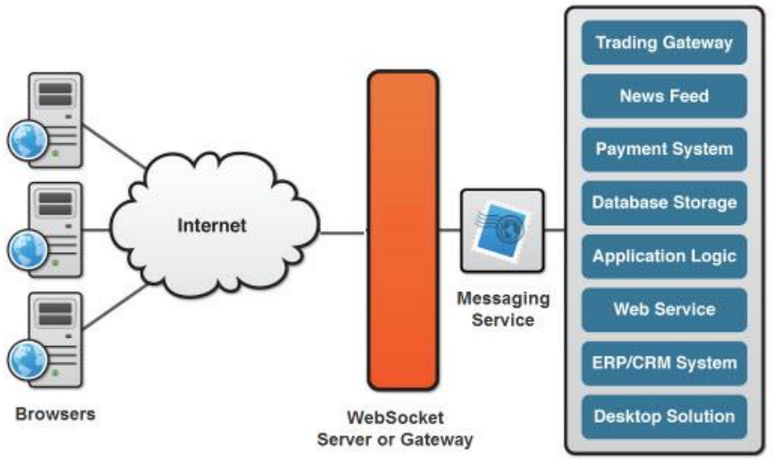
Example of current weather API respond:

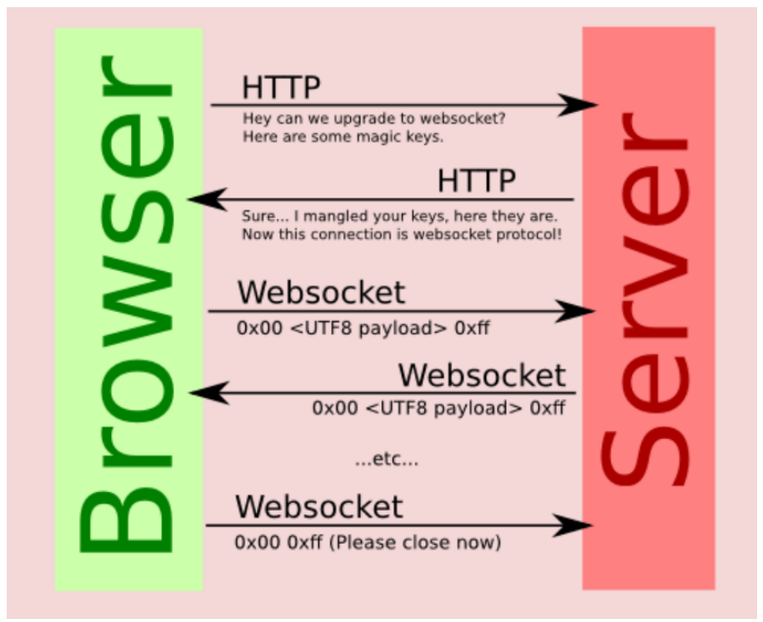
```
"main":{
"temp":306.15, //current temperature
"pressure":1013,
"humidity":44,
"temp_min":306, //min current temperature in the city
"temp_max":306 //max current temperature in the city
},
```

- Address : api.openweathermap.org
- Port : 80 (Default)
- URL: data/2.5/
- Querystring : q=Seroul,kr&appid=appid
- Request Message: GET

7.7. Websocket

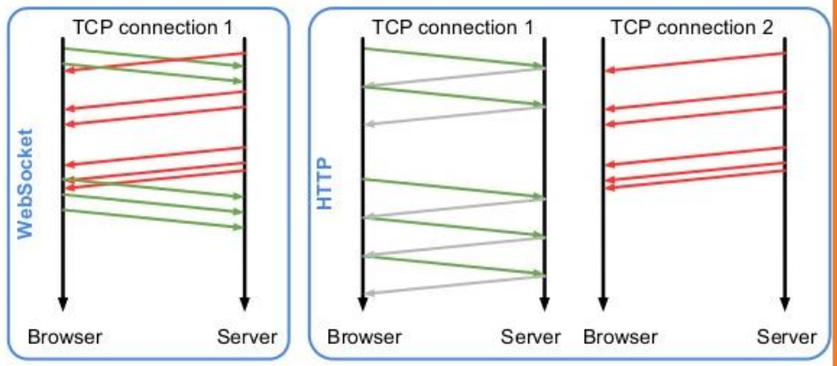
- 서버와 웹브라우저가 비동기 전이중(Full Duplex) 통신을 할 수 있는 방법
  - XML HTTP Request의 단점(단방향 통신)을 해결
  - HTTP나 HTTPS를 사용하여 한 번 연결한 후에는 통신을 직접적으로 수행
  - 일반 소켓(TCP) 연결





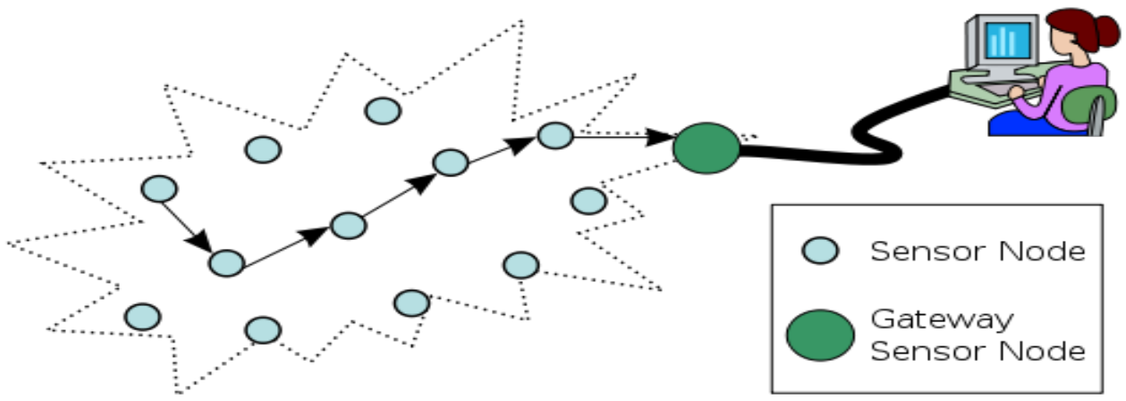
HTTP
- HTTP는 최소한 2개의 소켓 필요
- HTTP는 한 번에 한 개의 요청만 수행
- 매번 재 접속이 필요
- 메시지 순서제어 불가

Websocket
- 1개의 소켓으로 전이중 통신
- 한번 연결 후 지속적 통신
- 메시지 순서제어 가능

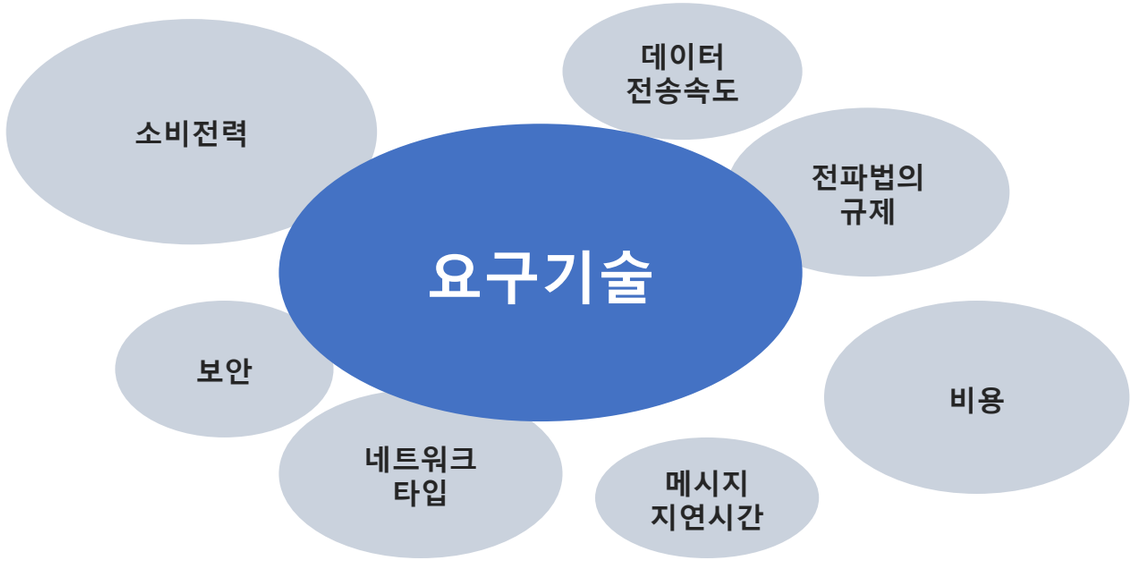


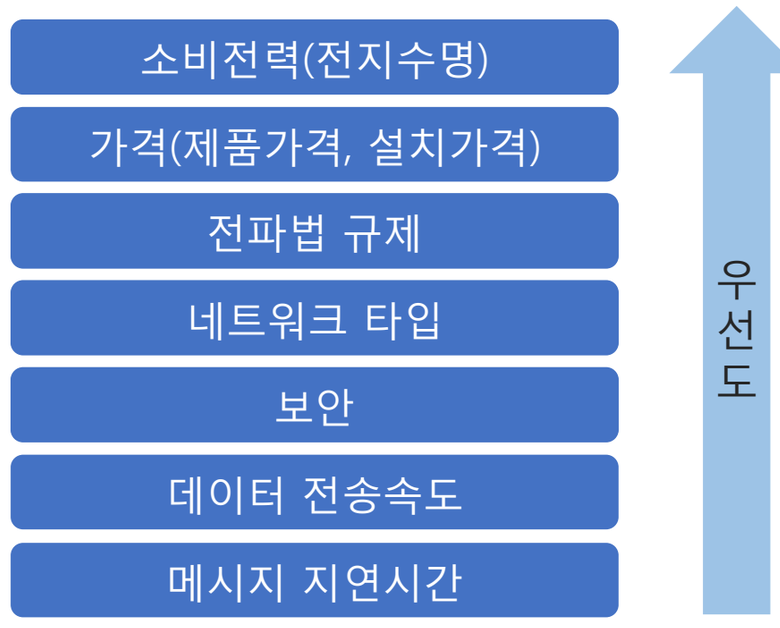
8. WSN (1)

- 무선센서 네트워크란
  - WSN(Wireless Sensor Networks).
  - [Wireless Sensor & Actuator Networks] : 센서 뿐만 아니라 Actuator도 포함한 제어계측용 무선네트워크를 의미.
  - 게이트 웨이를 경유하여 유선 네트워크와 접속하고 계측제어 데이터 액세스 단말 기능을 담당.
  - 주로 빌딩 오토메이션, 홈 오토메이션, 공장 오토메이션 등의 분야에 활용됨.



8. WSN (2)


요구기술
소비전력
데이터
전송속도
전파법의
규제
보안
비용
네트워크
타입
메시지
지연시간


소비전력(전지수명)
가격(제품가격, 설치가격)
전파법 규제
네트워크 타입
보안
데이터 전송속도
메시지 지연시간
우선도

8. WSN (3)

- 무선센서 네트워크의 장점
  - 비용절감
  - 생산 프로세서 변경의 유연성 : 생산설비의 증감과 레이아웃 변경 에 대해서 신속하고 용이하게 대응 가능.
  - 센싱 범위의 확대 : 회전체 배치와 수백 개 계측점의 대량 센서 배치가 가능.
- 종래의 무선센서 네트워크의 문제점
  - 표준화 프로토콜의 결여, 독자적인 프로토콜 사용
  - 부품의 안정 공급이 보장되지 않음
  - 상당히 고가(지적소유권의 라이선스 비용)로 보급되기 어려움
  - 상호 접속성이 거의 불가능

- 표준화된 무선센서 네트워크의 문제점
  - 사용자의 다양한 사양을 만족시킬 수 없다.
  - 표준화된 통신 프로토콜은 높은 보안이 요구 된다.
  - 제품의 차별화가 필요하다.

8.1. WSN 기술요구 (1)

- 소비전력
  - 1회의 전지교환으로 수년간 작동이 가능한 소비전력 성능이 요구 됨

- 주요 고려 사항
  - 데이터 송신주기.
    - 주기가 짧아질 수록 전지의 수명이 짧아짐.
  - 통신의 방향성
    - 송신 뿐인 단 방향 통신과는 달리 쌍 방향 통신의 경우 수신을 기다리는 동안 많은 전력이 소비됨.
  - 데이터 프레임의 길이
    - 송신에 필요한 시간은 프레임의 길이에 비례. 또한 길이가 긴 프레임은 전파공중 충돌에 따른 송신 실패 가능성도 높기에 재송신으로 인해 송신전력이 허비될 수 있음.
  - 데이터전송의 지연시간
    - 소비전력을 억제시키기 위해 슬립모드 사용. 다만 슬립기간이 길면 길 수록 전력소비는 적으나 데이터 전송에 필요한 지연시간 증가.

8.1. WSN 기술요구 (2)

- 비용
  - 실장 비용을 삭감하기 위해 고도집적화 IC 부품과 간단한 통신 프로토콜 사용.

- 전파법의 규제
  - 무선센서 네트워크에 채용되는 통신규격은 소재국의 규제에 허가되는 범위에서만 동작 가능.

- 네트워크 타입
  - 무선의 특징을 살리는 새로운 타입의 네트워크로서 [애드 혹][멀티 홉][메쉬 네트워크]사용.

- 메시지 지연시간
  - 무선 시스템에 있어서 메시지 충돌은 피할 수 없기 때문에 실시간성이 요구되는 어플리케이션을 목표로 하지 않음.

8.1. WSN 기술요구 (3)

- 보안
  - 표준화된 무선센서 네트워크가 확산됨에 따라 무선 센서 네트워크의 보안에 대한 요구도 높아질 것이다. 보안의 실장은 무선센서 네트워크의 성능에 다음과 같은 영향을 미친다.

- 프레임 길이

보안용 데이터가 증가함에 따라 프레임이 길어지므로 네트워크 트래픽, 전력소비 및 데이터 전송의 신뢰성에 영향을 줌.

- 자원

필요한 RAM과 플래시 메모리가 많아져 비용이 증가.

- 처리 능력

보안 데이터의 송신 및 암호화로 인한 처리시간 증가.

- 설치 비용

현장에서 네트워크에 참가 불가능한 문제점이 발생할 우려가 증가함에 따라이를 처리하기 위한 엔지니어링 비용 증가.

8.1. WSN 기술요구 (4)

데이터 전송속도

일반적으로 무선센서 네트워크에 요구되는 데이터 전송속도는 매우 낮다. 데이터 전송속도를 높임에 따라 같은 크기의 전송 데이터에 필요한 송신시간이 짧아지기 때문에 전력소비를 감소시킬 수 있지만, 다음과 같은 트레이드 오프관계가 있다.

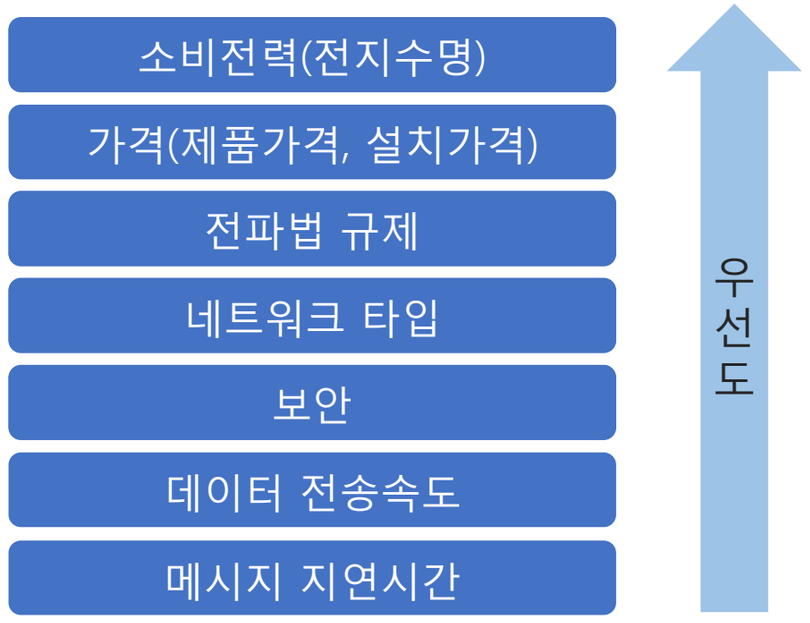
- 제품비용

고속 통신 프로토콜은 처리가 복잡하므로 제품비용 증가.

- 통신거리

이용 가능한 전파강도는 송신 파워 및 전송거리에 의해 결정.

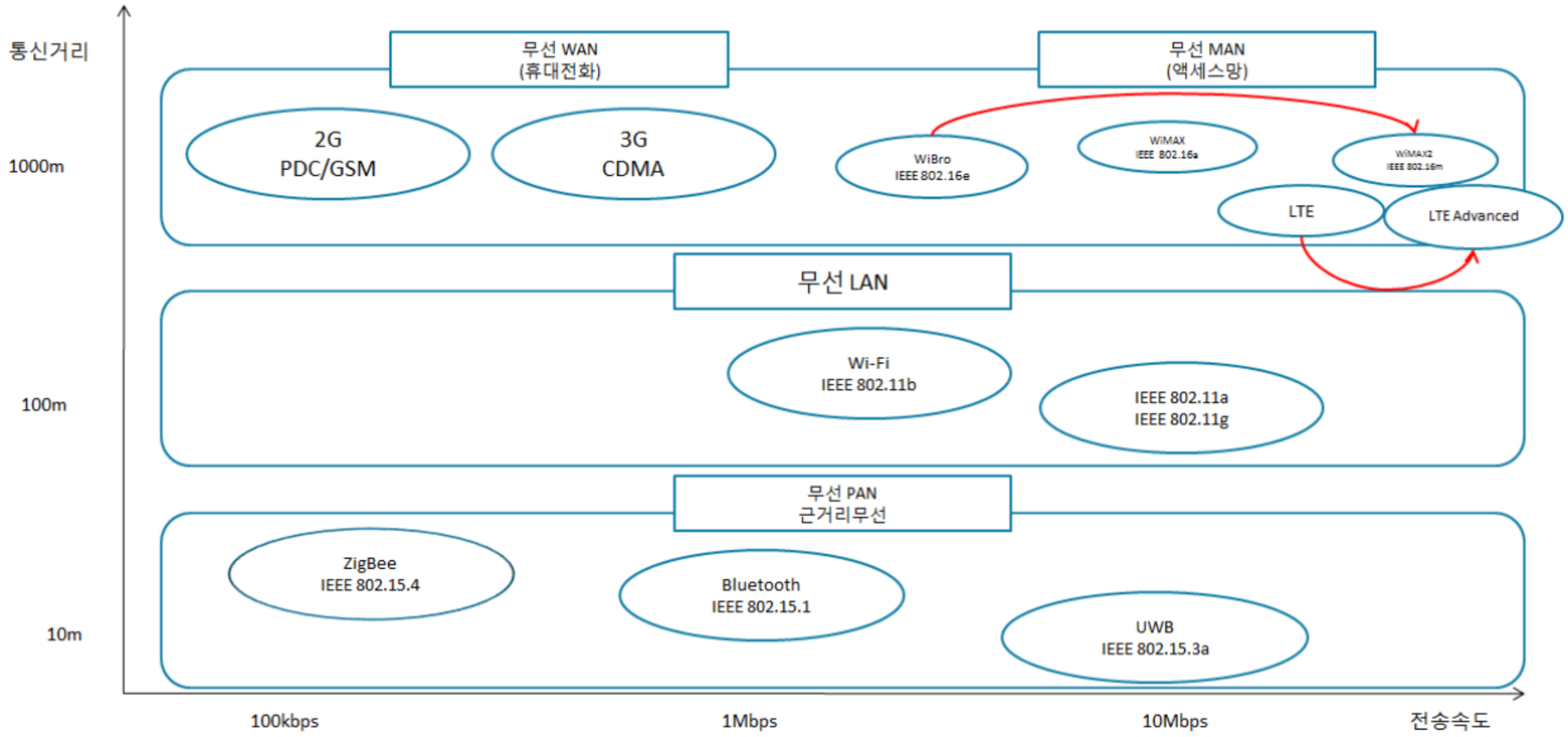
때문에 송신파워가 변하지 않는다는 조건하에 전송속도가 높아지면 통신거리가 짧아짐.



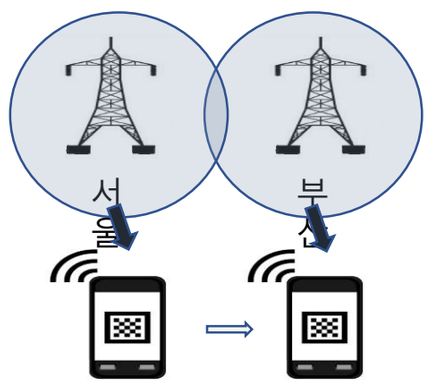
8.2. 무선통신과 표준화(1)

- 무선 광역통신망의 무선 WAN(Wide Area Network)
  - 휴대전화와 관련이 있음.
- 무선 MAN(Metropolitan Area Network)
  - 저렴한 가격으로 접속 서비스가 가능한 도시지역 무선 액세스망이다.
- 무선 구내 통신망인 무선 LAN(Local Area Network)
  - 사무실이나 가정에 보급됨. Notebook PC에서 필수 기능.
- 근거리 무선 네트워크의 무선 PAN(Private Area Network)
  - 개인의 행동범위를 커버하는 통신 네트워크이다.
- WiBro(**Wi**reless **Bro**adband) →WiMAX2
  - 이동하면서도 초고속 인터넷을 이용할 수 있는 무선 휴대 인터넷(Portable Internet)의 명칭.
- LTE(Long Term Evolution)→LTE Advanced
  - 일본 NTT가 중심이 된 컨소시엄에서 개발한 3.9세대, 혹은 4세대(4G) 무선 통신 기술이다.

8.2. 무선통신과 표준화(2)



8.3. 네트워크 타입

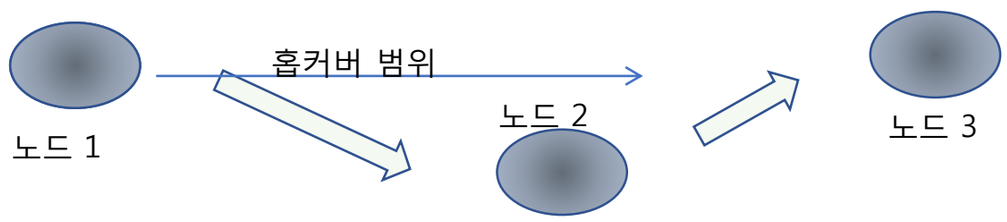


애드-혹 네트워크

특정 인프라에 의존하지 않고 그 장소에서 자동적으로 상호 접속하여 구축할 수 있는 네트워크. 구성변경 및 부분 고장에 대하여 네트워크 재구성이 용이함.

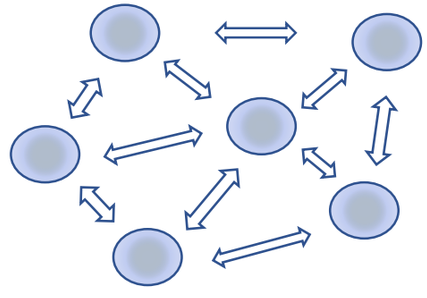
멀티 홉 네트워크

메시지가 송신 노드로부터 직접적으로 최종 목적 노드로 전송되는 것이 아닌 1개 이상의 중간 노드를 경유하는 전송 방식.



- **송신파워의 증가를 통한 통신거리의 연장에는 한계가 있지만 멀티 홉 기술을 이용하면 최대 송신파워를 증가시키지 않아도 통신거리의 연장이 가능**

- **메쉬 네트워크**
  - 전체를 제어하는 곳이 파괴되면 상호 연락이 전혀 되지 않았던 계층적 네트워크와 달리 하나가 파괴되어도 다른 루트를 경유하여 새로운 루트로 통신이 유지되는 시스템.
  - 유선 네트워크에서 여분의 네트워크를 구축하려는 경우에 다중화 통신모듈과 케이블 비용 등으로 비싸지지만 무선 메쉬 네트워크에서는 저비용으로 신뢰성이 높은 네트워크를 구축.



8.4. IEEE 근거리 무선 표준화

- IEEE 근거리 무선 통신 표준화 활동
  - IEEE 802.15.1 :
    - 1Mbs정도의 중간 전송속도를 가지는 PAN 기술의 블루투스 규격
  - IEEE 802.15.3 :
    - 고속의 전송속도 PAN인 WiMedia의 규격.
    - 멀티미디어 신호를 목적으로 하는 표준.
    - 제품화 실적은 없음.
    - 현재 그룹활동의 중심은 IEEE 802.15.3a로 이전됨.
  - IEEE 802.15.4 :
    - 낮은 전송속도의 PAN규격.
    - 주로 무선센서 네트워크를 목표로 하는 표준.

8.5. 블루투스

- 블루투스의 도전
  - 블루투스는 Ericsson, IBM, Intel, Nokia, 도시바의 5개사가 중심이 된 Bluetooth Special Issue Group에 의해 제안된 근거리 무선네트워크.
  - 2002년 4월 IEEE에 제안되어 IEEE 최초의 근거리 무선 규격 802.15.1 로서 승인됨.
- 블루투스의 사양 :
  - 2.4GHz 주파수 대역을 이용.
  - 통신거리는 수 미터에서 수십 미터.
  - 데이터 전송속도는 1Mbps.

- 변조방식: FHSS(Frequency Hopping Spectrums Spread)
  - FHSS는 이용하는 주파수가 고정되어 있지 않고, 극도로 짧은 시간마다 일정한 규칙으로 홉핑(변경)을 수행하는 방식
- 블루투스가 국내 무선센서 네트워크에 적용한 실적이 많지 않은 이유
  - 기동시간 : 전력소비가 심각함.
  - 네트워크 용량 : 1대의 마스터 노드에 7개의 슬레이브 노드밖에 접속할 수 없음.
  - 비용 : 블루투스의 통신 프로토콜이 상당히 복잡하고, 시장에서 기대하는 5달러 이하의 저렴한 칩 세트 비용은 거의 이루어지지 못했다.

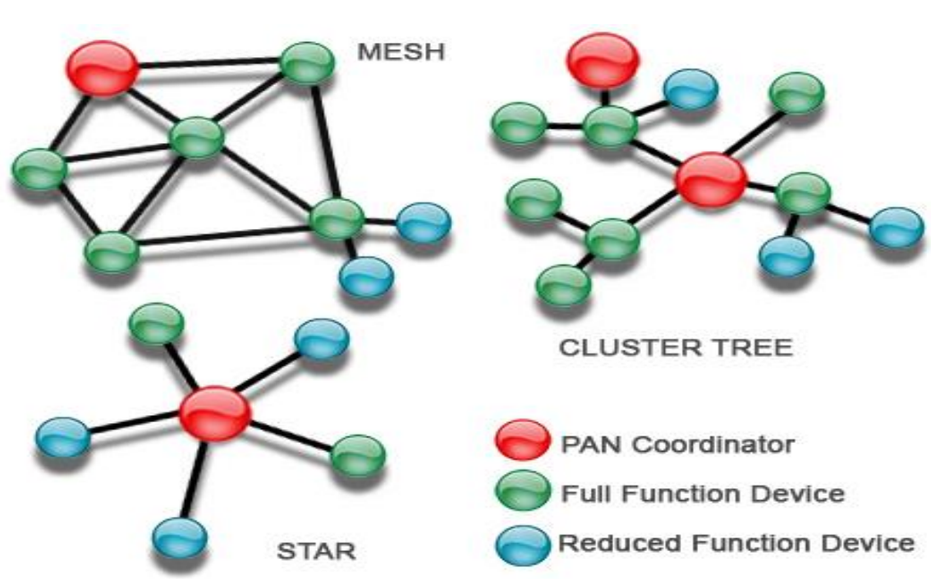
8.6. ZigBee (1)

- IEEE 802.15.4
  - 블루투스의 문제점을 인식하여 업계표준화 단체인 ZigBee 연합이 발족됨.
  - ZigBee 연합은 저속 통신속도의 무선 PAN을 제안.
  - 2003년 IEEE802.15.4-2003 규격으로 승인됨.
  - 규격 범위: 물리층과 미디어 액세스층(MAC층).
  - 상위 프로토콜은 ZigBee 연합에 의해 별도로 개발되고 있음.
  - ZigBee 규격 [3가지 Low]:
    - 낮은 통신속도
    - 낮은 전력소비
    - 저비용

- [ZigZag]와 [Bee] 의 합성어
- 꿀벌처럼 작고 민첩하며 서로 정보교환 한다는 뜻
- 저소비전력 , 저비용, 네트워크 용량, 상호접속성의 요구를 만족시킴.
- 2004년 12월에 ZigBee 최초의 사양인 버전 1.0이 발표됨.

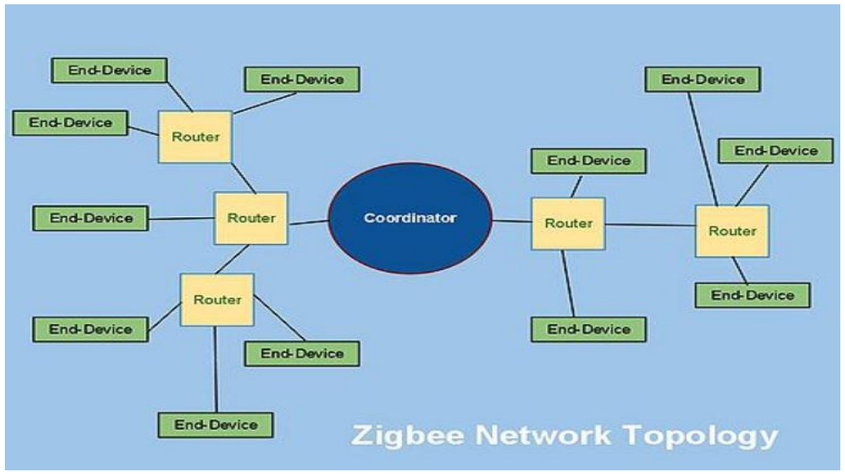
8.6. ZigBee (2)

- ZigBee 통신 정의 및 주요 특쟁
  - 저전력, 저가격, 사용의 용이성을 가진 WPAN(근거리무선통신 네트워크)의 대표적 기술 중 하나로 2003년 IEEE 802.15.4 표준의 PHY/MAC 층을 기반으로 한 상위 프로토콜 및 응용을 규격화한 기술
- 저전력 기반구조 : 수개월~수년 사용
- 최대 65,000개의 슬레이브 연결
- 다양한 네트워크 토폴로지 사용 가능 : 메쉬 네트워크
- 데이터 전송율: 250 kbps
- 적용거리 : 10~75m
- 전송 방식 : DSSS
- 사용주파수 : 868MHz, 915MHz, 2.4 GHz
- 활용: 센서간 통신



8.6. ZigBee (3)

- **논리 디바이스 타입 (ZigBee 버전)**
  - PAN 코디네이터 (ZigBee 코디네이터)
  - 코디네이터 (ZigBee 라우터)
  - 네트워크 디바이스 (ZigBee 엔드 디바이스)



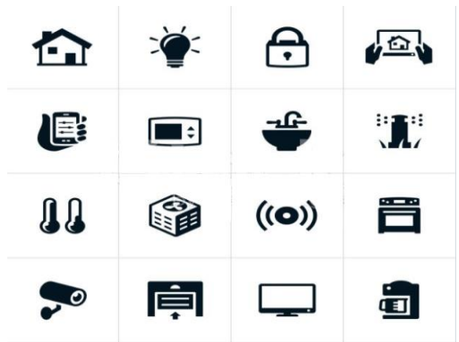
- **물리 디바이스 타입**
  - 풀 기능 디바이스 FFD(Full-Function Device)
  - 기능 한정 디바이스 RFD(Reduced-Function Device)

항목	FFD	RFD
라우터 기능	유	무
코디네이트	가능	불가능
라우터	가능	불가능
엔드 디바이스	가능	가능
가격	높음	낮음
MAC층 사이즈	16~20kB	12~16kB

8.6. ZigBee (4)

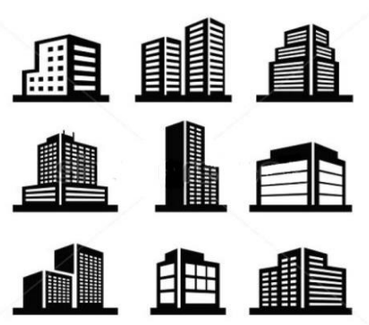
IEEE 802.15.4 명칭	PAN 코디네이터	코디네이터	네트워크 디바이스
ZigBee 명칭	ZigBee 코디네이터	ZigBee 라우터	ZigBee 엔드 디바이스
네트워크 기동 기능	유	무	무
라우터 기능	유	유	무
비콘 발행	가능	가능	불가
관리 범위	전체 노드	자기의 자식 노드	자신만
가능 노드 수	1 노드	복수 노드	복수 노드
전력 소비	대	대	소
요구되는 마이크로 프로세서 자원	대	중	소

8.7. 적용분야



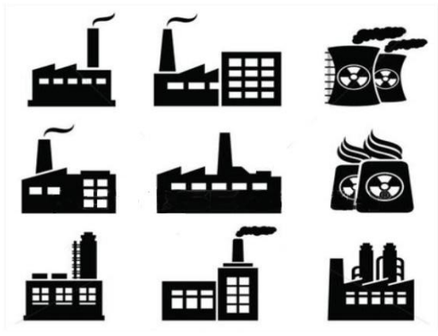
홈 오토메이션

가정 안의 각종 기기를 서로 유기적으로 결합하여 가정생활의 편리성 향상과 안전확보를 꾀하는 종합적인 시스템. 표준화된 무선센서 네트워크의 보급을 통해 네트워크와 가전의 연결과 서로 다른 제조사의 가전제품간의 상호 접속 을 가능하게 해줌.



빌딩 오토메이션

사무실이나 호텔 등 영업용 건물에서는 공조 시스템, 조명 시스템, 엘리베이터, 소방 시스템, 보안, 출입관리 등과 같이 각각의 네트워크가 이미 구축되어 있다. 이런 네트워크의 공사와 유지 보수에는 비용이 상당히 든다. 그러므로 도입비용과 관리비용이 저렴한 무선센서 네트워크의 발전이 크게 기대된다.



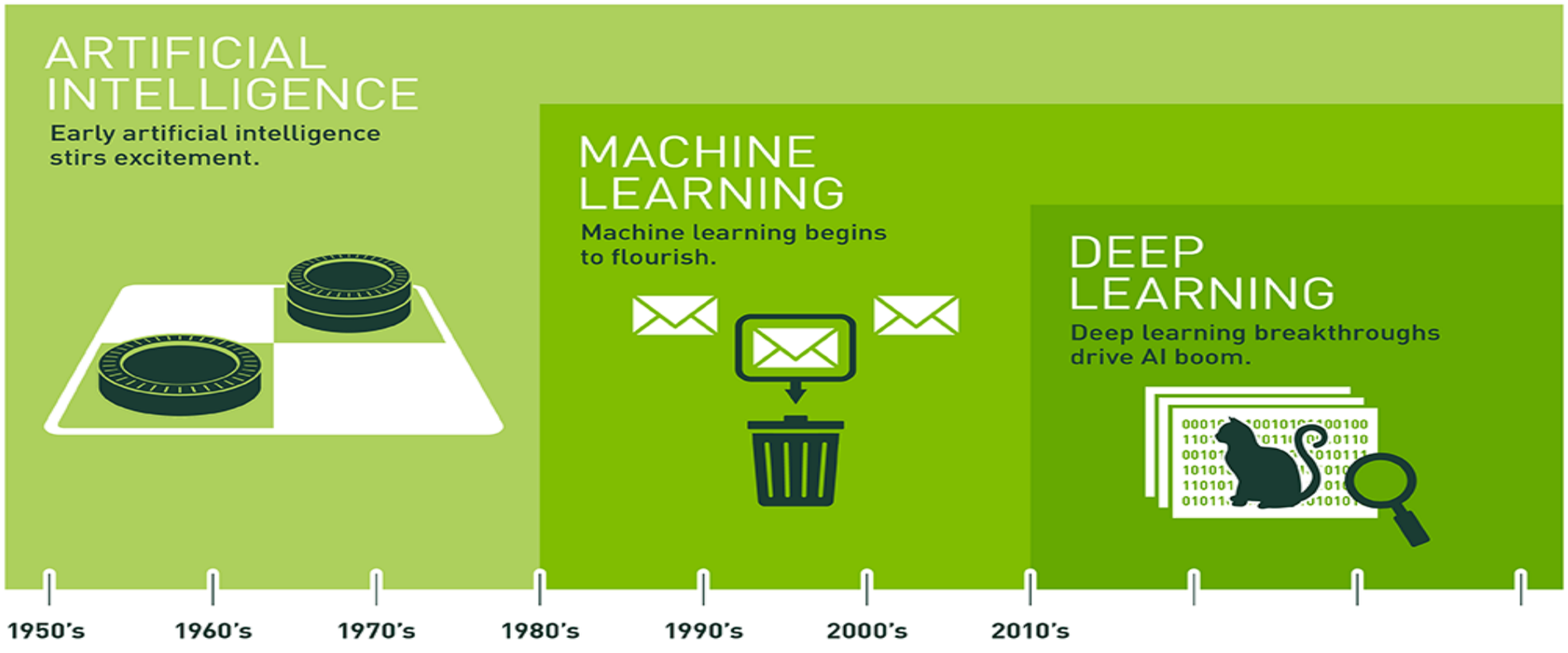
공업 오토메이션

사무실이나 공업오토메이션 분야에서는 각종 필드 버스나 공업용 이더넷 및 직렬 통신 네트워크가 계측과 제어에 사용되고 있다.

8.8. 무선규격 비교

IEEE 규격	802.11b	802.15.1	802.15.4
마케트 명	Wi-Fi	Bluetooth	ZigBee
주파수	2.4GHz	2.4GHz	2.4GHz
변조방식	CCK, PBCC	GFSK	O-QPSK
확산방식	DSSS	FHSS	DSSS
통신거리	100m	10m	30m
통신속도	11Mbps	1Mbps	250kbps
네트워크 용량	32 노드	7 노드	65,536 노드
스택 용량	1MB 이상	250kB 이상	16~128kB 이상
전지수명	수시간	수일	수년
어플리케이션	무선 LAN	무선 음성	제어계측

9. Machine Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

9.1. 지도학습 **(Supervised Learning)**

이미 **<u>유형(class)을 구분 짓는 속성(attribute)을 갖는 주어진 데이터 집합(Training set)</u>**으로부터 유형(class)를 구분하는 함수적 모델(model)을 찾아 유형(class)을 구분 짓는 속성(attribute)을 갖지 않는 새로운 데이터의 유형을 구분하는 기술

알고리즘

-Bayesian classification

-Decision tree

-Regression

-Neural Network

-hidden Markov model(HMM)

활용예시

- 이미지 인식
- OCR (이미지의 문자인식)
- 음성인식
- NLP(자연어인식)
- 추세예측(회귀분석)

9.2. 비지도학습 **(Unsupervised Learning)**

유형(class)을 구분 짓는 속성(attribute)을 갖지 않는 주어진 데이터 집합(Training set)으로부터 데이터 자체의 상호 유사성(likelihood or distance)을 통하여 유형(class)를 구분하는 함수적 모델(model)을 찾아 새로운 데이터의 유형을 구분하는 기술

알고리즘

-K-Means clustering

-Nearest Neighbor Clustering

-EM clustering

-Self-organizing feature map (SOM)

-Principal component analysis (PCA)

-Independent Component Analysis(ICA)

활용예시

- 마케팅의 고객세분화
- 개체의 분포 특성 분석
- News Summarizing

9.3. 강화학습 **(Reinforcement Learning)**

데이터의 상태(State)을 인식하고 이에 반응한 **<u>행위(Action)에 대하여 환경으로부터 받는 포상(Reward)을 학습</u>**하여 행위에 대한 포상을 최적화 하는 정책(model)을 찾는 기술

알고리즘

-Brute force
-Monte Carlo methods
-Markov Decision Processes
-Value Functions
-Q-Learning
-Dynamic Programming

활용예시

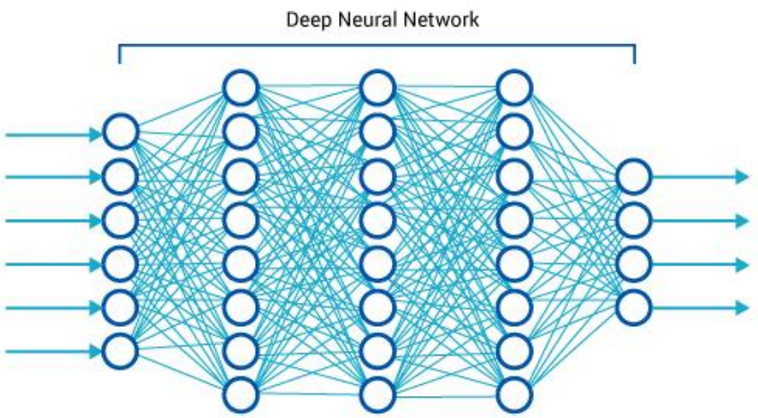
- 로봇제어
- 게임개인화
- 공정최적화

9.4. Deep Learning (1)

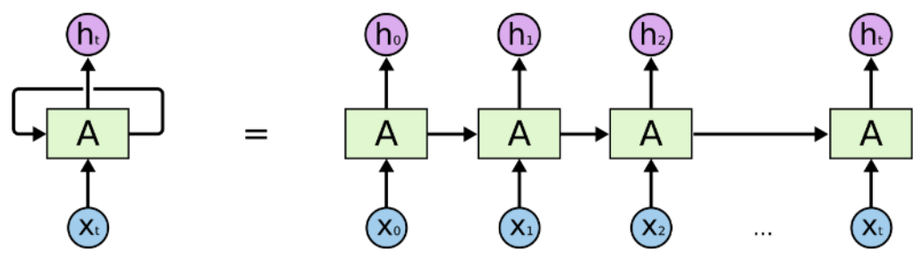
알고리즘	설명	주요활용분야
DNN (심층 신경망)	입력계층과 출력계층 사이에 복수개의 **은닉계층**들로 이뤄진 인공신경망	자연어처리, 음성인식
CNN (합성곱 신경망)	**최소한의 전처리**를 사용하도록 설계된 다계층 퍼셉트론의 한 종류	영상, 음성분석 분야
RNN (순환 신경망)	인공신경망을 구성하는 유닛 사이의 연결이 **Directed Cycle**을 구성하는 신경망	필기체 인식

활용분야	설명
자동음성인식	DNN 알고리즘을 사용한 음성 추출/분석 기술
영상인식	딥러닝 기술을 이용하여 자동영상 captioning 기술로 발전
자연어처리	재귀신경망(RNN)을 이용한 negative sampling, 단어표현
약물 발견과 독성학	다중 DNN 알고리즘을 이용한 약물 효험 예측
CRM(고객관계관리)	심층강화학습을 이용한 마케팅 활동의 값 예측

9.4. Deep Learning (2)

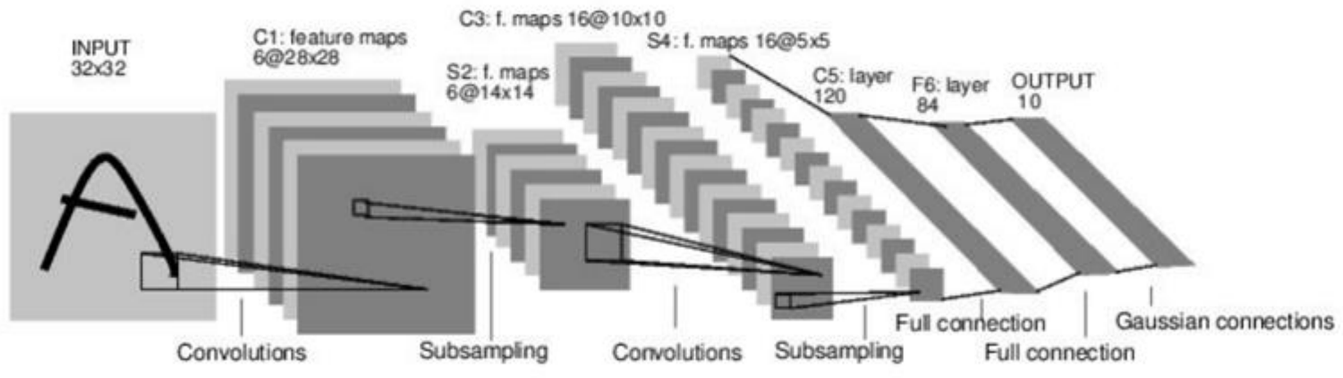


DNN : Deep Neural Network



An unrolled recurrent neural network.

RNN : Recurrent Neural Network



CNN : Convolution Neural Network

10. 디지털 필터 (1)

- 디지털 필터

FIR 필터(Finite Impulse Response Filter)
1. 입력신호의 유한한(Finite) 값들 만을 가지고 필터링 수행
2. 회귀성분을 갖지 않는다.
3. 동일한 특성을 구현할 때 차수가 IIR필터에 비해 높아서 구현비용, 실행시간 등이 많이 든다.
4. 위상변이(입력과 출력 간의 파형의 형태 유지)가 중요한 경우 반드시 FIR 필터를 사용해야 한다.

IIR 필터(Infinite Impulse Response Filter)
1. 입력신호의 값과 출력신호의 값이 재귀적으로(recursive) 적용되며 필터링이 수행
2. 구현식의 형태로 반복식이 되며 특성 함수인 임펄스 응답은 무한한 길이를 갖는다.
3. FIR 필터에 비해 차수가 적어져서 경제성이 있으나 위상특성의 측면에서는 비선형성을 가지므로 입력파형과 출력파형이 유사한 파형을 갖지 않는다.
4. 진폭이나 주파수 축의 왜곡이 발생할 가능성이 있으므로 원하는 필터링 대역보다 표본화 주파수를 크게 잡는 것이 좋다.

10. 디지털 필터 (2)

	FIR	IIR
회귀성분	없음	재귀적 적용
***임펄스 응답**	유한	무한
응답특성	선형	비선형
안정성	높다	낮다
연산량	크다	낮다
아날로그 필터와의 관계	전환 불가능	아날로그 필터에서 차용

10.1. 배치 방식과 재귀식 방식

- 배치 방식(batch process)
  : 연산 결과를 얻기 위하여 모든 데이터를 한꺼번에 계산하는 방식
  → 새로운 데이터가 들어오면 모든 데이터를 다시 더하여 $k+1$로 나누는 연산을 해야 하기 때문에 앞서 계산한 k개의 평균 데이터를 사용하지 않으므로 효율이 좋지 못함

- 재귀식 방식(recursive process)
  : 새로운 데이터를 반영하여 연산하기 위하여 기존의 결과를 활용하는 방식 **(IIR 필터의 일종)**
  → 앞선 결과를 이용하기 때문에 계산 효율이 좋음
  → 또한 모든 데이터들을 저장하고 있지 않고 이전 연산 결과와 추가된 데이터만 저장하면 되기 때문에 적은 메모리만으로도 연산이 가능함

10.2. 평균 필터 함수의 재귀식 구현

```
function [ avg ] = AvgFilter(x)

persistent prevAvg k
persistent firstRun

if isempty(firstRun)
  k = 1;
      prevAvg = 0;
  firstRun = 1;
end

alpha = (k - 1) / k;
avg   = alpha*prevAvg + (1 - alpha)*x;

prevAvg = avg;
k       = k + 1;

end
```

- 예제
  : 매틀랩을 이용하여 평균 필터 함수를 재귀 방식으로 구현하시오.

- 풀이
  : 재귀식으로 계산하기 위해서는 AvgFilter 함수 호출이 끝난 다음에도 직전 평균값과 데이터 개수는 계속 저장되어 있어야 함
  : 이를 위하여 평균값을 저장하는 변수 preAvf와 데이터 개수 k를 persistent 변수로 선언함
  : C/C++의 정적(static) 변수와 비슷함

  : AvgFilter 함수의 첫 부분에 if문 내부에서 persistent 변수로 선언되어 있는 변수를 체크하여 처음 실행되는 순간 초기화 작업을 수행함

10.3. 이동평균 필터

- 이동평균
  : 전체 데이터의 평균을 계산하기 보다는 가장 최근의 데이터 일부만으로 평균을 계산하는 방법
  : 새로운 데이터가 들어오면 가장 오래된 데이터는 버리는 방식으로 데이터 개수를 일정하게 유지하면서 평균을 구하는 방식

$$\bar{x}_k = \frac{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}{n}$$

  : 평균필터의 $\bar{x}_k$는 k개 데이터의 평균을 의미하나 이동평균의 $\bar{x}_k$는 $k-n+1$번째부터 k번째 데이터까지 총 n개 데이터의 평균임

- 이동 평균 필터의 재귀식 표현 → 재귀식 표현은 가능하나, 효율개선은 없음.
  이동평균 필터는 FIR필터의 일종.

$$\bar{x}_{k-1} = \frac{x_{k-n}+x_{k-n+1}+\cdots+x_{k-1}}{n}$$

→

$$\bar{x}_k - \bar{x}_{k-1} = \frac{x_{k-n+1}+x_{k-n+2}+\cdots+x_k}{n} - \frac{x_{k-n}+x_{k-n+1}+\cdots+x_{k-1}}{n}$$

$$= \frac{x_k - x_{k-n}}{n}$$

→

$$\bar{x}_k = \bar{x}_{k-1} + \frac{x_k - x_{k-n}}{n}$$

10.4. 1차 저주파 통과 필터

- 1차 저주파 통과 필터 (IIR 필터. 재귀식 연산 가능)
  : 저주파 통과 필터는 오래된 측정값일 수록 더 작은 가중치를 부여하는 필터임
  : 가중치 차별화 덕분에 저주파 통과 필터는 잡음 제거와 변화 민감성이라는 상충되는 요구를 이동평균 필터보다 더 잘 충족하며 계산식이 간단하다는 장점도 갖고 있음
  : 오래된 측정값일 수록 기하급수적으로 가중치를 낮게 주어 이동평균을 계산하기 때문에 지수가중 (exponential weighted) 이동평균 필터라고도 부름

평균필터	$\bar{x}_k = \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k$	$\bar{x}_k$: k개 데이터의 평균
이동 평균 필터	$\bar{x}_k = \frac{x_{k-n+1} + x_{k-n+2} + \cdots + x_k}{n}$ $\bar{x}_k = \bar{x}_{k-1} + \frac{\bar{x}_k - x_{k-n}}{n}$	$\bar{x}_k$: n개 데이터의 평균
1차 저주파 통과 필터 (지수가중 이동평균 필터)	$\bar{x}_k = \alpha\bar{x}_{k-1} + (1-\alpha)x_k$	$\bar{x}_k$: 가중치 평균 $(0 < \alpha < 1)$