CHAPTER 01: 디지털 신호 처리의 개요 (Introduction of Digital Signal Processing)

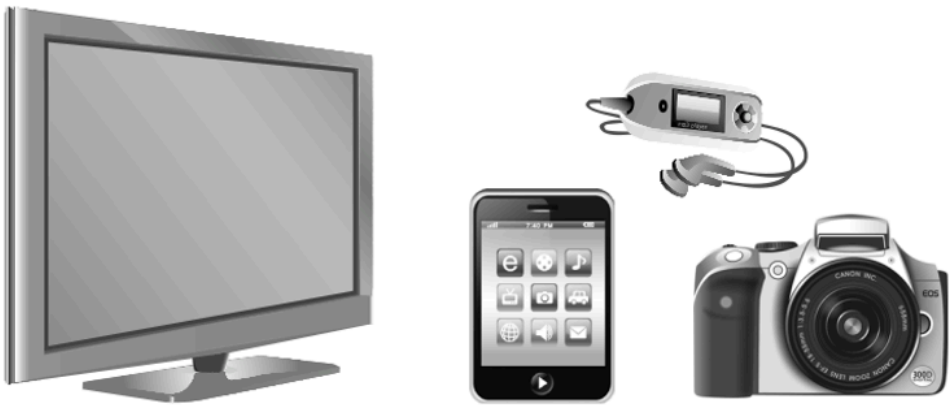
1.1 서 론

디지털 시대의 도래

디지털은 21세기의 시대적 아이콘

- 디지털 기술과 재품의 보편화 <- **다기능, 고성능, 편리성, 경제성**

  (예) 컴퓨터, 휴대전화, 디지털 카메라, MP3, PMP, DVD, HDTV 등
- 디지털 신호 처리의 필요성과 중요성은 더욱 커짐



[그림 1-1] 생활에서 사용되는 디지털 제품들

그림1-1 생활에서 사용되는 디지털 제품들

디지털의 특징

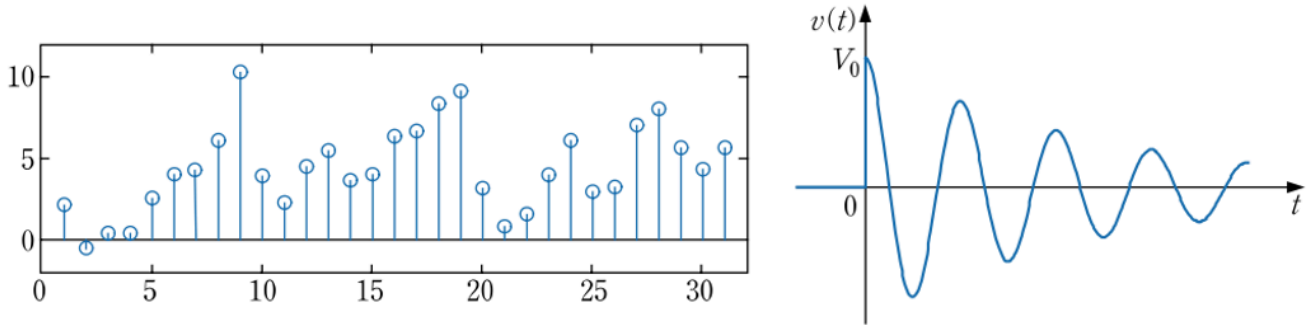
- 모든 대상(신호)이 단순한 숫자의 나열(수열)로 변환
  - 적용 대상을 가릴 필요 없이 여러 신호를 하나의 틀에서 통합적으로 다룰 수 있음

> (예)
> 아날로그 : 문서=타자기, 음악=오디오, 동영상=TV
> 디지털 : 컴퓨터, 스마트폰

- 신호의 조작이 보다 단순하고 쉬우며 훨씬 다양한 처리가 가능함
- S/W의 힘을 빌어 H/W 단독으로는 불가능한 처리를 제공할 수 있음

1.2.1 신호 (Signal)

- 물리량의 변화 형태를 담은 일련의 정보/자료의 집합
  (예) 월별 평균 기온, 전압, 전류, 심전도(ECG), 뇌전도(EEG), 주식가격 등
  - 신호는 다양한 형태 또는 표현 방식으로 나타낼 수 있음
    음성신호 → 마이크(전기신호), 테이프(자기신호), CD(숫자열)
  - 신호는 수학적으로 한 개 이상의 독립 변수의 함수로 표현됨
  - 정보는 신호가 변화하는 양상 속에 담겨 있음



(a) 춘천의 평균 기온(2013년 3월) (b) RLC 병렬회로의 전압

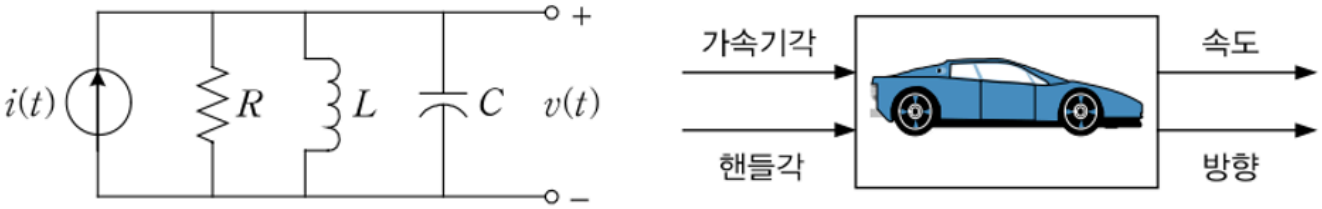
[그림 1-2] 신호의 예

그림1-2 신호의 예

1.2.2 시스템(System)

> - 시스템(System)은 특정한 목적에 맞도록 주어진 신호를 조작하고 처리해내는 장치이다.
>   - 시스템은 들어오는 신호(입력)에 대한 반응으로 다른 신호(출력)을 만들어 낸다.
>   - 시스템은 입력, 출력, 그리고 동작 규칙에 의해 명확하게 규정되며, 수학적으로 하나 또는 여러개의 방정식으로 표현된다.

- 일련의 신호를 처리(교환, 변환, 가공, 추출, 전송)해 다른 일련의 신호를 만드는 실체
  (예) 전기회로, 통신 시스템, 생체 시스템, 생산 시스템 등
  - 입력, 출력(응답), 동작 규칙에 의해 기술됨 → ***시스템 모델링(System Modeling)***
  - 수학적으로 일련의 방정식으로 표현됨
  - 물리적 요소(하드웨어) 또는 알고리즘(소프트웨어)으로 구성
    (예) 오디오의 등화기(equalizer) : 전자회로 구현(H/W) , DSP chip + 알고리즘(S/W)



(a) RLC 병렬회로 (b) 자동차

[그림 1-4] 시스템의 예

그림1-4 시스템의 예

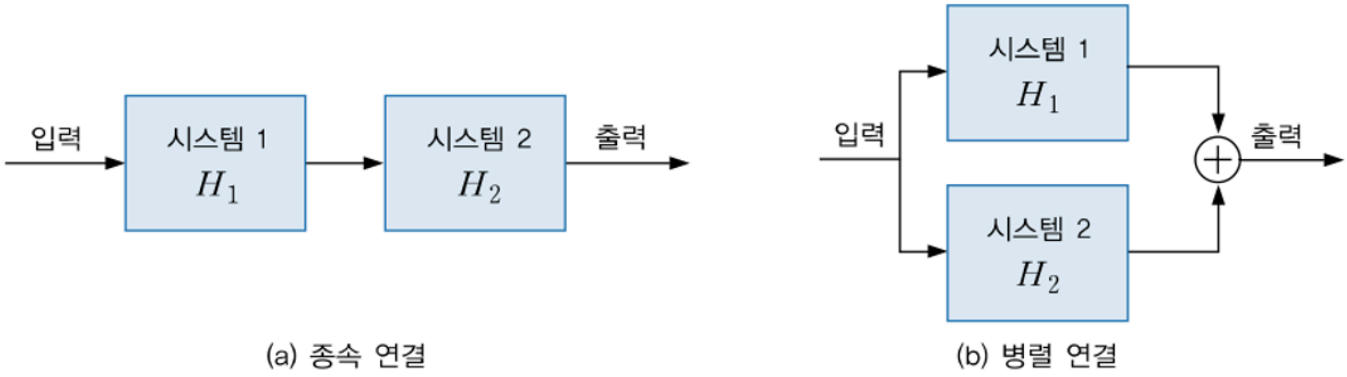
1.2.3 신호와 시스템의 표현

시각적 표현

- 파형은 시간에 따른 신호의 값의 변화를 그래프로 나타낸 것이다.
- 신호의 파형으로부터 신호의 특성과 관련한 기초적인 정보들을 파악할 수 있다.

블럭선도(Block diagram)

- 신호 처리 시스템의 해석 및 설계
  ① 시스템을 우선 몇 개의 작은 시스템(부시스템)으로 분리
  ② 각 부시스템의 기능과 동작 특성을 파악
  ③ 각 부시스템 간의 관계를 파악
  ④ 전체 시스템 분석 및 설계
- 블록선도 : 시스템의 구성과 기능을 알기 쉽게 시각적으로 나타낸 그림
  ① 시스템을 각 부시스템 또는 구성요소로 나누고 이를 블록으로 대체
  ② 블록 안에 시스템의 특성을 말해주는 수식, 그래프, 명칭 등을 표시
  ③ 신호의 흐름을 따라 각 부시스템 또는 구성 요소를 연결
- 블록선도의 연결
  ▶ 종속(cascade)연결: 각 부시스템들을 직렬로 연결
  ▶ 병렬(parallel)연결: 각 부시스템들을 병렬로 연결
  ▶ 궤환(feedback): 출력을 입력으로 다시 '되먹이는' 연결



[그림 1-6] 시스템의 연결 방법


그림1-6 시스템의 연결 방법

수학적(이론적) 표현

- 수학적 표현을 사용하여 뉴턴의 운동 방정식이나 맥스웰의 전자기 방정식과 같이 신호와 시스템의 본질을 간결하게 수식으로 표현
- 수학적 모델은 실제 신호와 시스템을 오차 없이 정확하게 표현하는 것은 아니고 핵심적인 특성을 표현하는 것임
- 신호는 함수로 시스템은 방정식으로 표현함

> - 신호는 수학적으로 **함수**로 표현.
> - 신호의 시간영역(`time domain`) 표현은 시간에 따른 값의 변화(파형)를 함수로 나타냄.
> - 신호의 주파수영역(`frequency domain`) 표현은 신호에 포함된 주파수 성분들을 주차수의 함수로 나타내는 것으로, (주파수) **스펙트럼(`Spectrum`)**이라고 함.
> - 시스템은 수학적으로 입력과 출력의 관계를 규정짓는 방정식으로 표현.
> - 시스템의 시간영역 표현은 입출력 신호를 시간의 함수로 표현한 방정식으로, 미분/차분 방정식과 **컨볼류션(`convolution`)** 표현이 대표적임.
> - 시스템의 주파수영역 표현은 입출력 신호를 주파수의 함수로 표현한 수식으로, 주파수 응답와 전달함수가 있음.

예)

$$v_s(t) = v_R(t) + v_C(t) = Ri(t) + \frac{1}{C}\int i(t)dt$$

$$H(s) = \frac{V_C(s)}{V_s(s)} = \frac{1}{RCs+1}$$

1.2.4 신호처리 (`Signal Processing`)

- 원하는 목적에 알맞은 결과를 얻을 수 있도록, 신호에 대해 시스템을 이용하여 교환, 변환, 가공, 전송, 저장 등을 가하는 행위
  ▶ 해석 : 신호로부터 원하는 특정 정보를 빼내어 적절한 방법으로 표현
  ▶ 합성 : 조절 신호에 의해 원하는 출력 신호를 발생
  ▶ 변환 : 신호를 한 물리적인 형태로부터 다른 형태로 변환
  ▶ 필터링 : 불필요한 성분을 제거하거나 바람직한 형태로 신호를 변형

예제:1-1

'알리바바와 40인의 도둑' 이야기에서 "열려라, 참깨"라는 암호로 동굴 문을 여는 장면을 현대판 보안 시스템으로 바꾸어 설명하라.

> 풀이

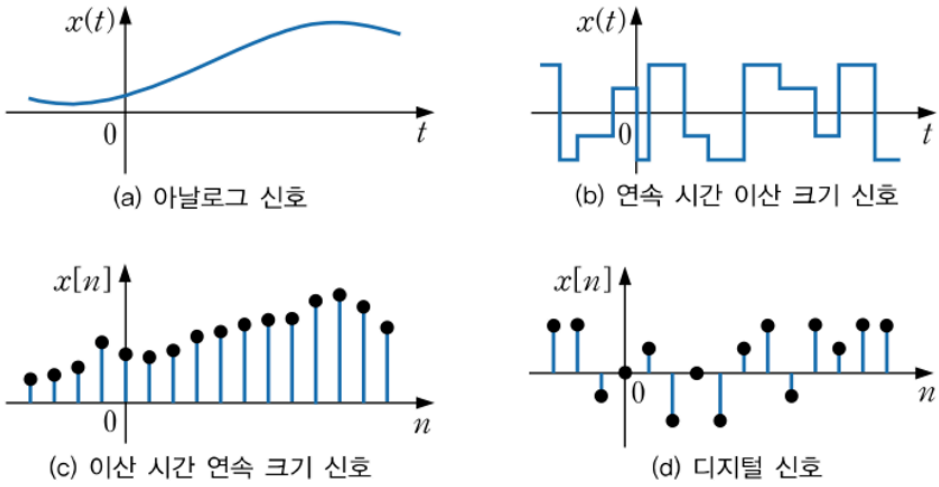
▶ 마이크(변환) : 음성 신호를 전기 신호로 변환
▶ 필터(필터링) : 심한 바람소리를 걸러내고 사람 음성만 깨끗하게 뽑아냄
▶ 암호해독기(해석) : 입력 신호가 "열려라 참깨"가 맞는지를 판별
▶ 음성 합성(합성) : "출입을 승인합니다"라는 기계음을 발생

[그림 1-5] '알리바바와 40인의 도둑'의 보안 시스템 블록선도

그림1-5 알리바바와 40인의 도적의 보안 시스템 블럭도

1.2.5 아나로그와 디지털 신호

- 아날로그 신호 : 연속 크기, 연속 시간
  (예) 심전도(ECG), 뇌전도(EEG) 등
- 디지털 신호 : 이산 크기, 이산 시간
  (예) 컴퓨터 이진 신호



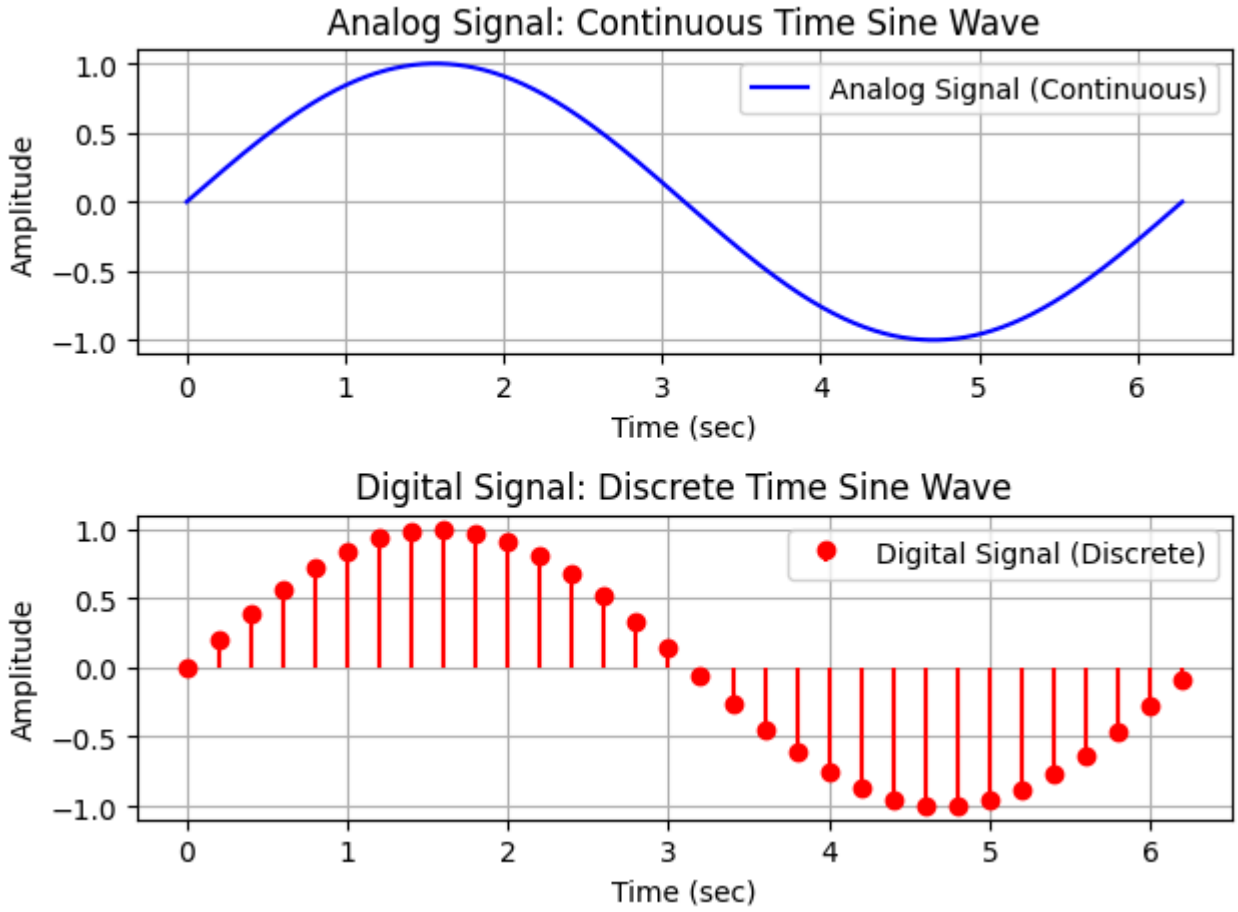
[그림 1–3] 시간과 크기에 따른 신호의 분류

그림1-3 시간과 크기에 따른 신호의 분류

- 다음의 코드는 연속적인 사인파(아날로그 신호)와 이산적인 샘플링 사인파(디지털 신호)를 생성하고 시각화하는 예제를 보여줌.

```
import numpy as np
import matplotlib.pyplot as plt

# 아날로그 신호: 연속 시간 사인파
t_analog = np.linspace(0, 2 * np.pi, 1000)  # 0부터 2π까지 1000개의 점 생성
amplitude_analog = np.sin(t_analog)  # 사인파 계산

# 디지털 신호: 이산 시간 사인파
t_digital = np.arange(0, 2 * np.pi, 0.2)  # 0부터 2π까지 0.2 간격으로 샘플링
amplitude_digital = np.sin(t_digital)  # 사인파 계산

# 그래프 그리기
#plt.figure(figsize=(12, 6))

# figsize=(width, height):
#     width: 그림의 너비. 인치 단위로 지정. 여기서는 12인치.
#     height: 그림의 높이. 인치 단위로 지정. 여기서는 6인치.

# 아날로그 신호 플롯
plt.subplot(2, 1, 1)  # 2행 1열의 첫 번째 서브플롯
plt.plot(t_analog, amplitude_analog, label='Analog Signal (Continuous)', color='b')
plt.title('Analog Signal: Continuous Time Sine Wave')
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.legend()

# 디지털 신호 플롯
plt.subplot(2, 1, 2)  # 2행 1열의 두 번째 서브플롯
plt.stem(t_digital, amplitude_digital, label='Digital Signal (Discrete)', basefmt="
plt.title('Digital Signal: Discrete Time Sine Wave')
plt.xlabel('Time (sec)')
plt.ylabel('Amplitude')
```

```
34 plt.grid(True)
35 plt.legend()
36
37 plt.tight_layout()
38 plt.show()
39
```



연속신호(Continuous Signal)와 이산신호(Discrete Signal)의 표기 방법에서 주의해야 할 점

- **연속신호 (Continuous Signal)** : 일반적으로 $x(t)$와 같은 형태로 표기. 여기서 t 는 시간의 연속적인 값을 나타냄.
  (아나로그) 주파수 f, (아나로그) 각주파수 ω
- **이산신호 (Discrete Signal)**: 보통 $x[n]$ 와 같은 형태로 표기. 이때 n은 이산적인 시간 인덱스를 나타냄(정수).
  (디지털) 주파수 F, (디지털) 각주파수 Ω

1.3 디지털 신호처리 (Digital Signal Processing, DSP)

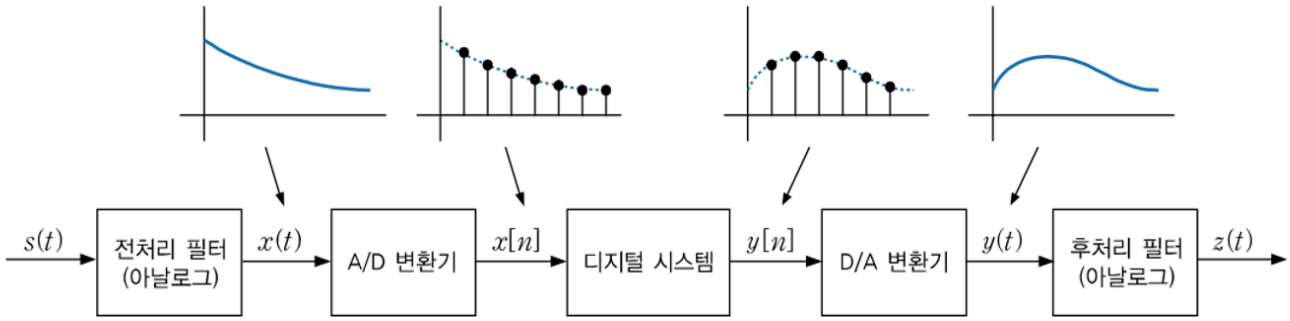
- 디지털 신호처리란 디지털 형태로 표현된 신호(디지털 신호)를 분석, 변환, 조작(manipulation)하는 기술을 의미 함.
- 디지털 시스템에 의한 이산 신호의 처리라고 볼 수 있음.
- 수치적인 처리 : 디지털 신호 = 수열(Sequence)

- 하드웨어적 and/or 소프트웨어적 처리

> ***주의: 수열(Sequence)와 순열(Permutation)은 다른 개념임.***
>
> - **수열 (Sequence)**: 특정한 순서로 나열된 값들의 집합을 의미함.
> 예를 들어, $x[n]$형태로 표현되는 디지털 신호는 시간 index n에 따라 각 샘플 값이 나열된 것을 의미함.
>   - **예시**: $(x[0], x[1], x[2], \cdots, x[N])$와 같이 시간에 따른 값들이 순서대로 배열.
> - **순열 (Permutation)** :주어진 집합의 원소들을 특정한 순서로 나열하는 방법을 의미함.
> 원소의 순서를 변경할 수 있으며, 원소의 개수에 따라 가능한 조합의 수가 결정됨.
>   - **예시**: 집합 {1, 2, 3}의 순열은
> (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)등 여러 가지가 있음.

1.3.1 디지털 신호처리 시스템의 구성

▶ **전처리 필터(`Pre-processing filter`)** : 신호 주파수 범위를 제한, 반주파수 중첩(anti-aliasing)
▶ **A/D 변환기** : 아날로그 신호 → 디지털 신호
▶ **디지털 시스템** : 디지털 신호 처리
▶ **D/A 변환기** : 디지털 신호 → 아날로그 신호
▶ **후처리 필터 (`Post-processing filter`)** : D/A 과정에서 발생하는 불필요한 과도 응답 제거



[그림 1-7] 전형적인 디지털 신호 처리 시스템의 구성

그림1-7 전형적인 디지털 신호 처리 시스템의 구성

1.3.2 디지털 신호처리의 장점

- 데이터/음성/영상 등 다양한 형태의 신호를 통합적으로 취급할 수 있다.
- 처리가 오로지 덧셈, 곱셈, 시간 지연에 기초하므로 신호 조작이 쉽다.
- 컴퓨터와 상용 S/W를 이용하여 편리하게 개발 & 검증할 수 있다.
- H/W와 S/W의 결합으로 다양한 처리 방식의 구현이 가능하다.

- 특히 아날로그 방식으로는 불가능한 작업도 구현할 수 있다.
- 프로그램의 교체만으로 전혀 다른 형태와 성능의 처리를 해낼 수 있다.
- 프로그램 및 디지털 소자의 특성상 동작 안정성과 신뢰성이 매우 높다.
- 데이터의 손상 없이 동일한 동작을 무한히 반복/재현할 수 있다.
- 수치적 처리로 잡음과 외란의 영향이나 유동(drift)을 감소시킬 수 있다.
- 정확도 및 감도를 특정 수준으로 보장할 수 있고, 수준의 조절이 쉽다.
- 시분할, 다중화 등에 의해 동시에 여러 신호들을 처리할 수 있다.
- 반도체 발달로 시스템을 값싸고, 작고, 가볍고, 간결하게 만들 수 있다.

디지털 신호처리의 단점

- A/D, D/A 변환 과정에서의 정보 소실
- 시간 지연 및 속도의 제한
- 시스템 안정도 저하 가능
- 시스템 설계 시 수학적 해석이 어려움 → ***더 많은 시간 소요***
- 양자화 오차에 의한 유한 어장 효과(`finite word length effect`)

예제: 디지털 필터에서의 유한 워드 길이 효과
상황: 16비트 고정 소수점 프로세서를 사용하여 디지털 저역 통과 필터를 구현한다고 가정
1) 계수 양자화 오차:

- 이상적인 필터 계수: 0.12345678
- 16비트로 표현 가능한 가장 가까운 값: 0.123456 (0.0000078의 오차)

> 이 작은 오차가 필터의 주파수 응답을 미세하게 변화시킬 수 있음. 특히 여러 계수에 걸쳐 이런 오차가 누적되면, 필터의 성능이 설계 사양에서 벗어날 수 있음.

2) 입력 양자화 오차:

- 실제 입력 신호 값: 0.7654321
- 16비트로 표현된 값: 0.765432 (0.0000001의 오차)

> 이러한 작은 오차들이 지속적으로 발생하면, 특히 낮은 진폭의 신호를 처리할 때 신호 대 잡음비(SNR)에 영향을 줄 수 있음.

디지털 신호처리(Digital Signal Processing, DSP)의 장점과 단점을 정리하면 다음과 같음.

장점	설명
정확성	아날로그 신호에 비해 정확한 처리가 가능하며, 노이즈와 왜곡에 대한 저항력이 높음.
유연성	다양한 알고리즘을 쉽게 적용할 수 있어 신호 처리 방법을 변경하기 용이함.
재현성	동일한 입력에 대해 항상 동일한 출력을 제공하여 일관성을 보장함.
저장 및 전송 용이성	디지털 데이터는 쉽게 저장하고 전송할 수 있으며, 압축 알고리즘을 통해 데이터 크기를 줄일 수 있음.

장점	설명
복잡한 처리 가능	고급 수학적 처리를 쉽게 수행할 수 있어 복잡한 신호 처리 작업이 가능함.
노이즈 저항	노이즈에 강하며, 오류 정정 기법을 통해 전송 중 발생할 수 있는 오류를 수정할 수 있음.
다양한 응용	음성 인식, 이미지 처리, 통신 시스템 등 다양한 분야에서 활용 가능함.
자동화 및 실시간 처리	컴퓨터와 프로세서를 사용하여 자동화된 처리가 가능하며, 실시간으로 신호를 분석하고 처리할 수 있음.

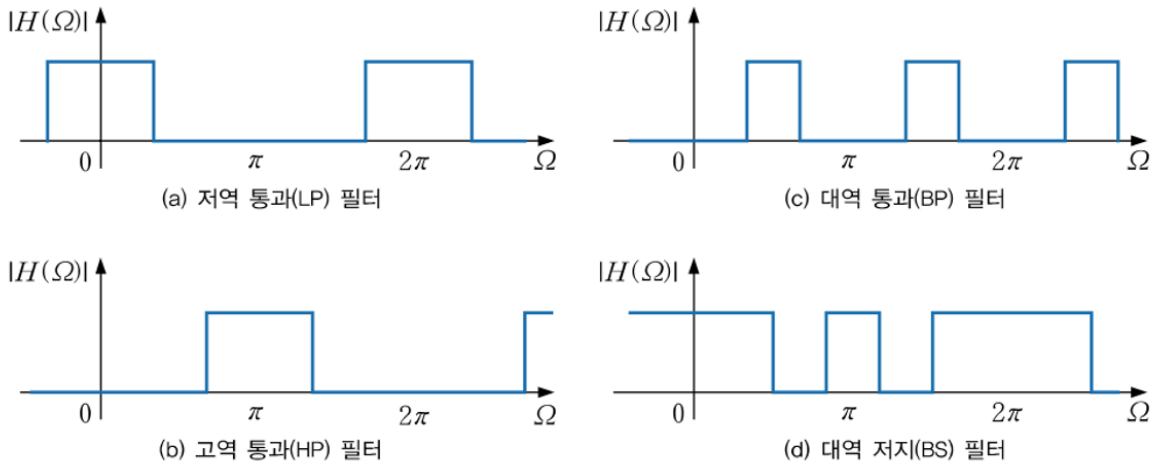
단점	설명
샘플링 손실	낮은 샘플링 주파수로 인해 정보 손실이 발생할 수 있으며, aliasing 현상이 발생할 수 있음.
연산 복잡성	고급 알고리즘 사용으로 인해 연산이 복잡하고 계산 자원이 많이 소모될 수 있음.
지연 시간	데이터 처리에 시간이 소요되어 실시간 응답이 필요한 애플리케이션에서 지연이 발생할 수 있음.
양자화 잡음	양자화 과정에서 발생하는 잡음이 신호의 품질을 저하시킬 수 있음.
비용	고급 하드웨어 및 소프트웨어의 비용이 많이 들 수 있으며, 초기 투자 비용이 증가할 수 있음.
복잡한 하드웨어 요구	DSP 구현을 위해 전용 하드웨어가 필요할 수 있으며, 설계와 유지 관리의 복잡성이 증가함.
데이터 손실 위험	디지털 데이터는 손상 시 복구가 어렵고, 저장 장치의 고장으로 인해 데이터 손실이 발생할 수 있음.
제한된 비트 수	비트 수에 의해 표현할 수 있는 신호의 범위가 줄어들며, 동적 범위가 제한될 수 있음.

1.3.3 디지털 신호처리의 목적

- **신호 해석 (`Signal Analysis`)**
  - 관측 신호로부터 그 신호의 특정한 성질을 해석
  - 스펙트럼 해석과 상관(correlation) 해석 등
- **정보 추출 (`Information Extraction`)**
  - (신호 해석/처리 편리하게) 관측 신호에 포함된 의미 있는 정보 추출
  - 확률 통계적 방법과 같은 수학적 기법 사용

> (예) 패턴 인식 특징 추출,
> 지문의 주요 특징 추출,
> 레이더 수집 신호의 특징 추출

- **필터링 (`Filtering`)**
  - 불필요한 성분을 제거하거나 바람직한 형태로 신호를 변형
  - 주파수 선택 필터 : 주파수에 따라 원하는 성분만 통과 (불필요한 성분 제거)



[그림 1-8] **주파수 선택 필터의 유형**

그림1-8 주파수 선택 필터의 유형

- 신호의 압축과 복원 (코딩)
  - 품질을 저하시키지 않으면서 방대한 데이터의 양을 줄임

```
(예) 음악 : wav 파일 1,411.2[kbps] (44.1[kbps]×16비트×2채널)
            MP3 파일은 192[kbps]

동영상 : 압축 않으면 700[MB] CD에 1분 전후 저장
         avi나 divx는 CD에 60분 정도 저장
```

- 신호의 예측과 시스템 식별
  - 적절한 처리에 의해 신호 발생 시스템의 특성이나 구조를 알아냄
- 신호의 합성
  - 음성, 영상 신호 합성

디지털 신호 처리 연산

- 기본적인 디지털 신호 처리의 연산

▶ **컨벌루션(convolution)** : 디지털 필터링의 기본 연산

→ 선형 & 원형(주기) 컨벌루션

▶ **상관(correlation)** : 신호간 유사성 나타내는 지표

→ 자기 & 상호 상관

▶ **변조(modulation)** : 효율적인 통신을 위해서 반드시 필요한 조작

▶ **변환(transform)** : 수학적으로 신호 표현을 바꾸는 연산

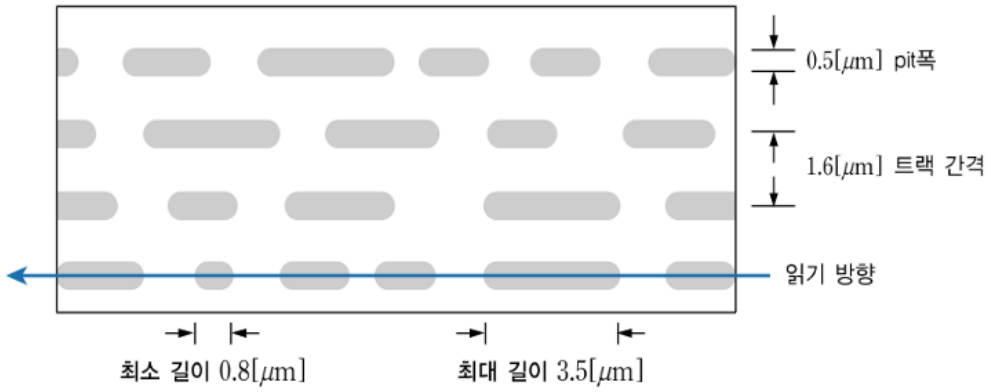
1.3.4 디지털 신호 처리의 응용

[그림 1-9] 디지털 신호 처리의 주요 응용

그림1-9 디지털 신호처리의 주요 응용

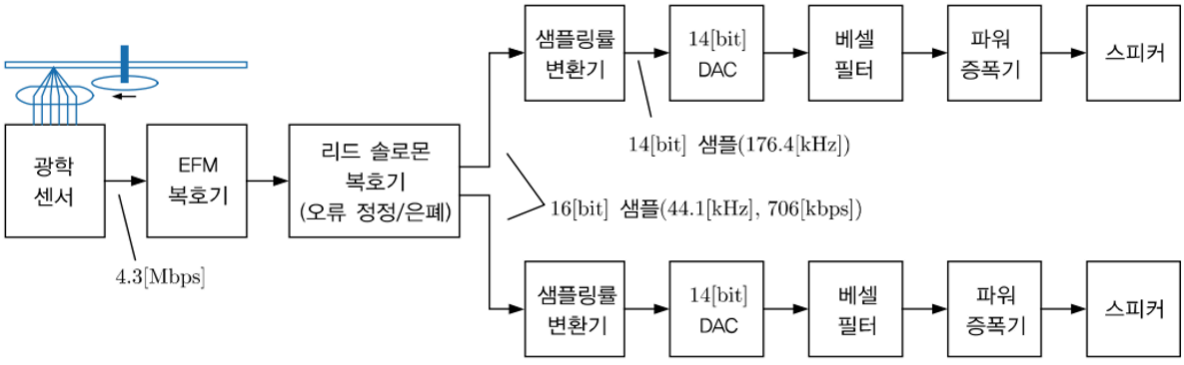
Ex: 오디오 CD 재생 시스템

- CD 기록 : 이진화된 정보 패턴에 맞추어 레이저로 표면을 태워 저장
- CD 재생 : 광학 pickup이 1.2[m/s] (210~480[rpm], 430만[bps])로 움직이며 이진정보 판독



[그림 1-10] CD의 표면

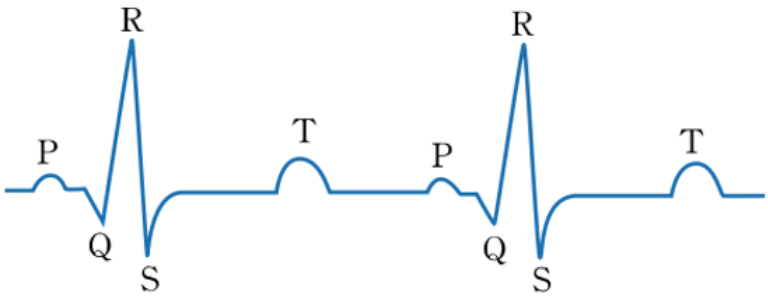
그림1-10 CD의 표현



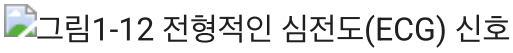
[그림 1-11] 오디오 CD 재생 시스템의 블록선도

Ex: 심전도(ECG) 신호의 처리

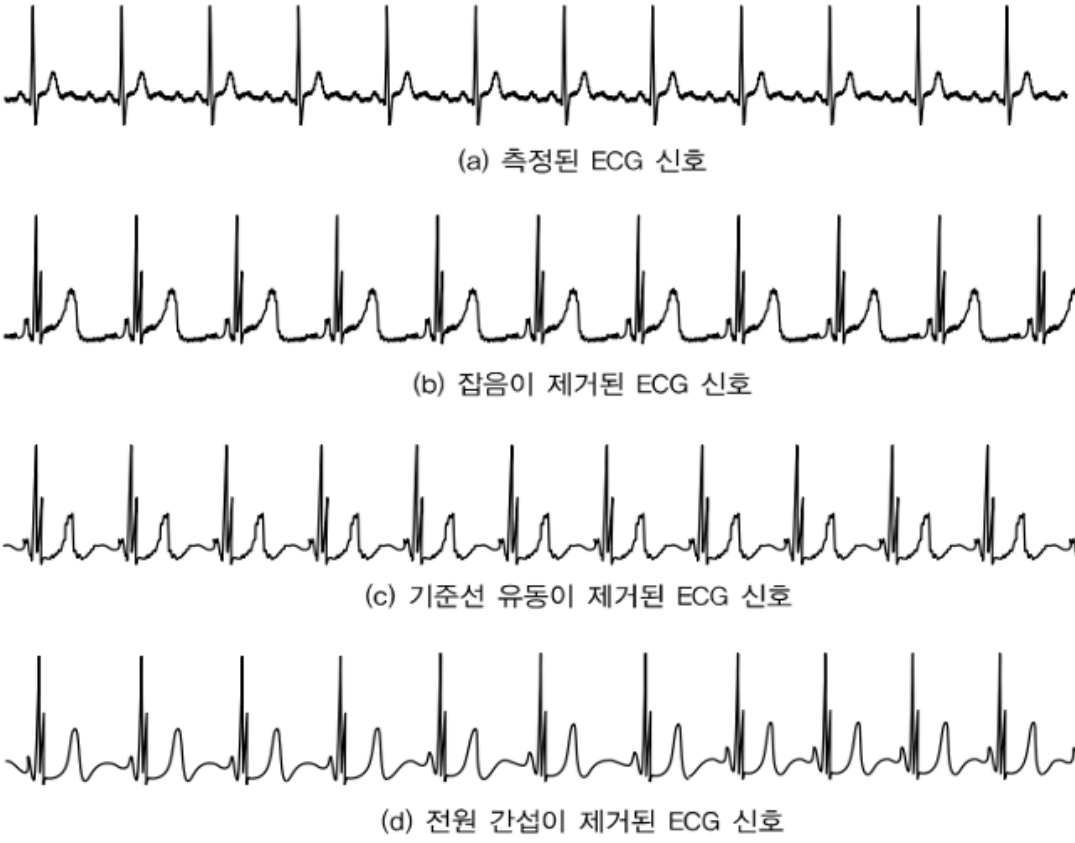
- P : 좌우 심방(`Left/righr ventricles`)의 수축(`contraction`)
- QRS : 심실의 수축이 일어나는 기간
- T : 심실의 이완기 (`repolarization`)
- P와 QRS 사이 : 심실에 혈액을 채울 수 있도록 신경 자극 전달이 지연되는 기간
- S와 T 사이 : 심실의 수축이 지속되는 기간



[그림 1–12] **전형적인 ECG 신호**

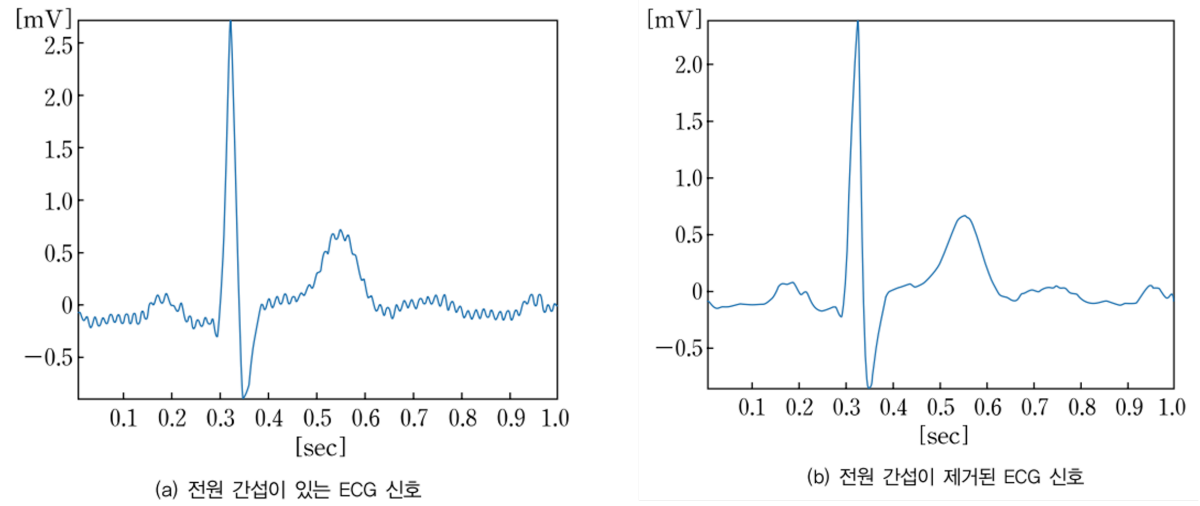


- 고주파 잡음과 기준선 유동, 전원 간섭이 혼재된 신호가 측정됨
- 고주파 잡음 : 저역 통과(LP) 필터를 이용하여 제거
- 기준선 유동 : 고역 통과(HP) 필터를 이용하여 제거
- 전원 간섭 : 노치(notch) 필터를 이용하여 제거



[그림 1–13] ECG 신호의 처리

그림1-13 심전도(ECG) 신호의 처리



[그림 1–14] 노치 필터에 의한 ECG 신호의 전원 간섭 제거

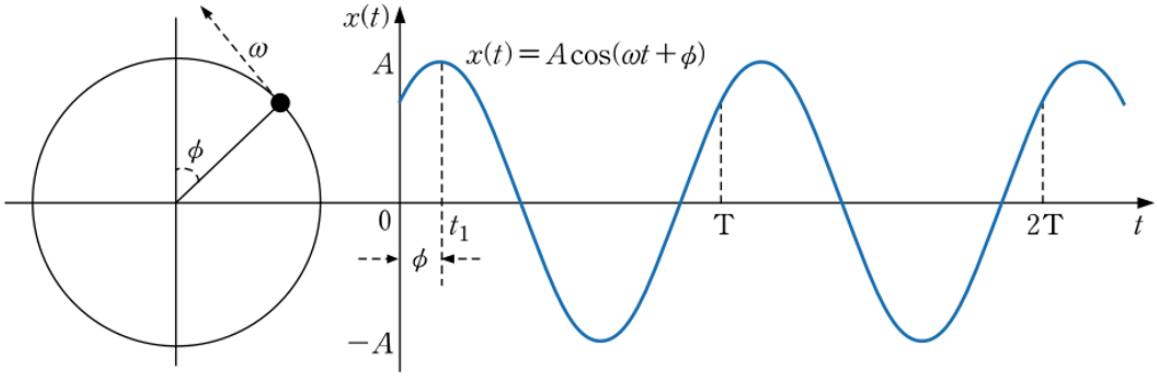
그림1-14 노치필터에 의한 심전도(ECG) 신호의 전원 간섭 제거

1.4 신호와 관련된 기초 개념

1.4.1 정현파 신호의 진폭, 위상, 주기, 주파수

- **정현파(Sinusoids)**
  - 등속 회전운동체의 위치를 시간에 대해 그린 파형을 갖는 신호
  - 진폭(A), 위상(Φ), 주파수(ω 또는 f)의 3가지 요소에 의해 정의됨

$$x(t) = A\cos(\omega t + \phi)$$



[그림 1–15] 정현파의 발생

그림1-15 정현파(Sinusoidal)의 발생

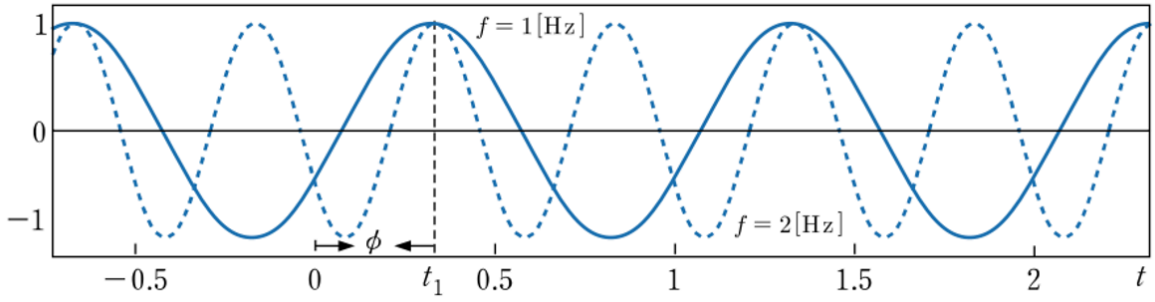
- **진폭(Amplitude)** : 정현파가 진동하면서 변할 수 있는 값의 범위
- **위상(Phase)** : 각으로 표시된 원점에서 코사인파 꼭지점(사인파 영점)의 거리
- **(기본)주기(Fundamental period)** : 정현파가 같은 파형을 반복하는 (최소) 시간 간격
- **주파수(Frequency)** : 정현파가 1초에 같은 파형을 반복하는 회수
  주파수가 높아질수록 신호의 파형은 시간적으로 더 급격한 변화를 보임
- **각주파수(Radian frequency)** : 정현파가 1초에 이동할 수 있는 라디안 각

$$\omega = \frac{2\pi}{T} = 2\pi f$$

$$T = \frac{2\pi}{\omega} = \frac{1}{f}$$

$$f = \frac{1}{T} = \frac{\omega}{2\pi}$$

- 주파수와 위상의 물리적 의미



[그림 1–16] 정현파에 의한 주파수와 위상의 개념

그림1-16 정현파에 의한 주파수와 위상의 개념

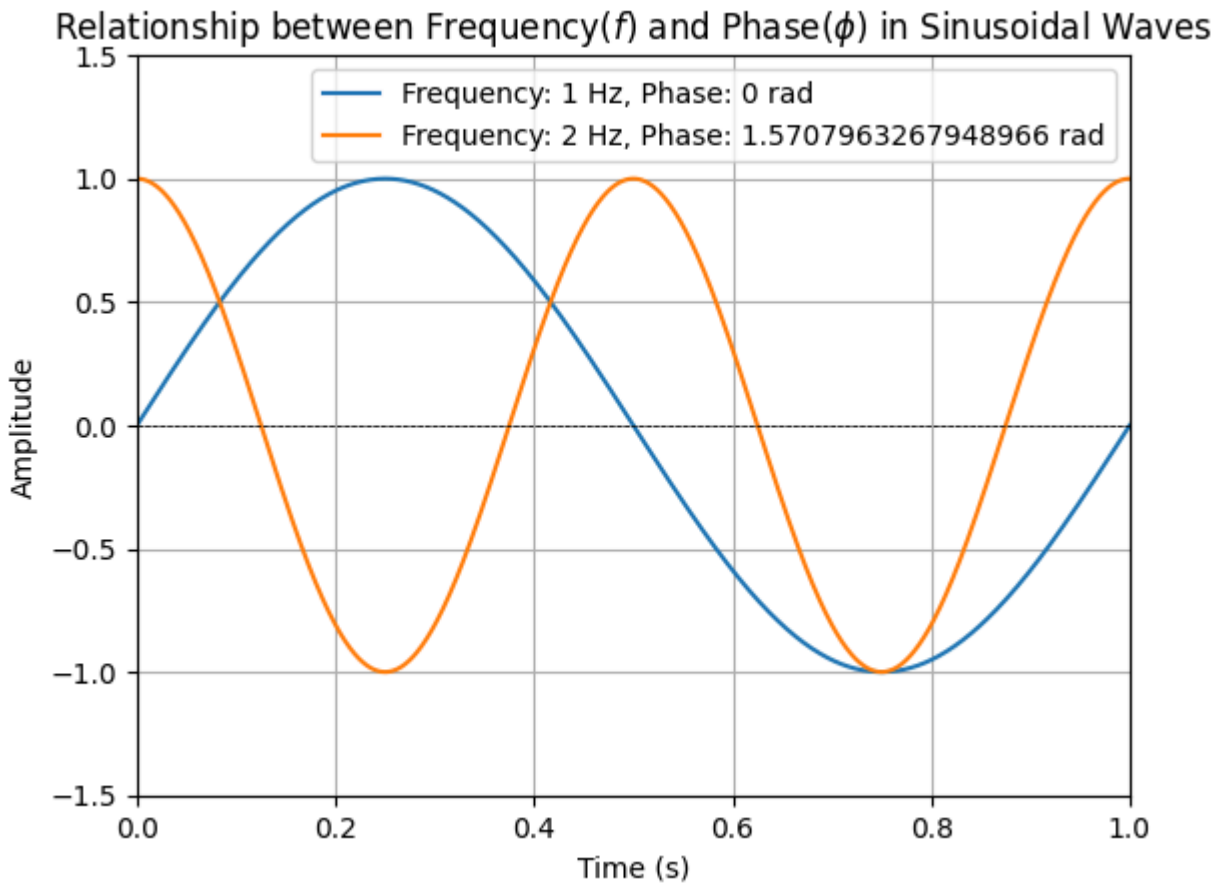
- 두 정현파는 지연 시간 t_1을 각으로 환산한 ϕ만큼의 위상을 가짐.
- 위상은 정현차의 모양에는 영향을 주지 않음.
- 파형의 시간 이동과 연관 → 지연 : 음(뒤진) 위상

$$\phi = -2\pi\frac{t_1}{T} = -2\pi f t_1$$

$$\cos(2\pi f(t - t_1)) = \cos(2\pi f t - 2\pi f t_1) = \cos(2\pi f t + \phi))$$

- 주파수와 위상의 관계를 그래프로 표시하는 코드

```
import numpy as np
import matplotlib.pyplot as plt

# 시간 변수 설정
t = np.linspace(0, 1, 1000)  # 0초에서 1초까지 1000개의 점

# 주파수와 위상 설정
frequencies = [1, 2]  # 1Hz, 2Hz
phases = [0, np.pi/2]  # 0도, 90도 (π/2 라디안)

# 그래프 생성
#plt.figure(figsize=(12, 6))

for f, p in zip(frequencies, phases):
    y = np.sin(2 * np.pi * f * t + p)  # 정현파 계산
    plt.plot(t, y, label=f'Frequency: {f} Hz, Phase: {p} rad')

# 그래프 설정
plt.title('Relationship between Frequency($f$) and Phase($\phi$) in Sinusoidal Waves
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.axhline(0, color='black', lw=0.5, ls='--')
plt.axvline(0, color='black', lw=0.5, ls='--')
plt.legend()
plt.grid()
plt.xlim(0, 1)
plt.ylim(-1.5, 1.5)

# 그래프 표시
plt.show()

```



Example: python programming for digital signal processing

```
1 !pip list
```

Google Colab에서 `!pip list` 명령어를 실행하는 이유는 현재 설치된 Python 패키지의 목록을 확인하기 위해서임. 이 명령어는 다음과 같은 정보를 제공 함:

1. 설치된 패키지 목록:
   - `!pip list` 는 현재 환경에 설치된 모든 패키지의 이름과 버전을 보여줌. 이를 통해 어떤 패키지가 설치되어 있는지 쉽게 확인.
2. 패키지 버전 확인
   - 각 패키지의 버전 정보를 함께 표시하여, 특정 패키지가 최신인지, 또는 특정 버전을 사용하고 있는지를 확인.
3. 의존성 관리
   - 프로젝트에서 필요한 패키지를 관리할 때, 어떤 패키지가 설치되어 있는지 확인하는 것이 필요함. 이를 통해 의존성 문제를 예방하거나 해결.
4. 환경 설정 확인

- 다른 환경에서 코드를 실행할 때, 동일한 패키지와 버전을 사용하는지 확인하여 코드의 일관성을 유지.

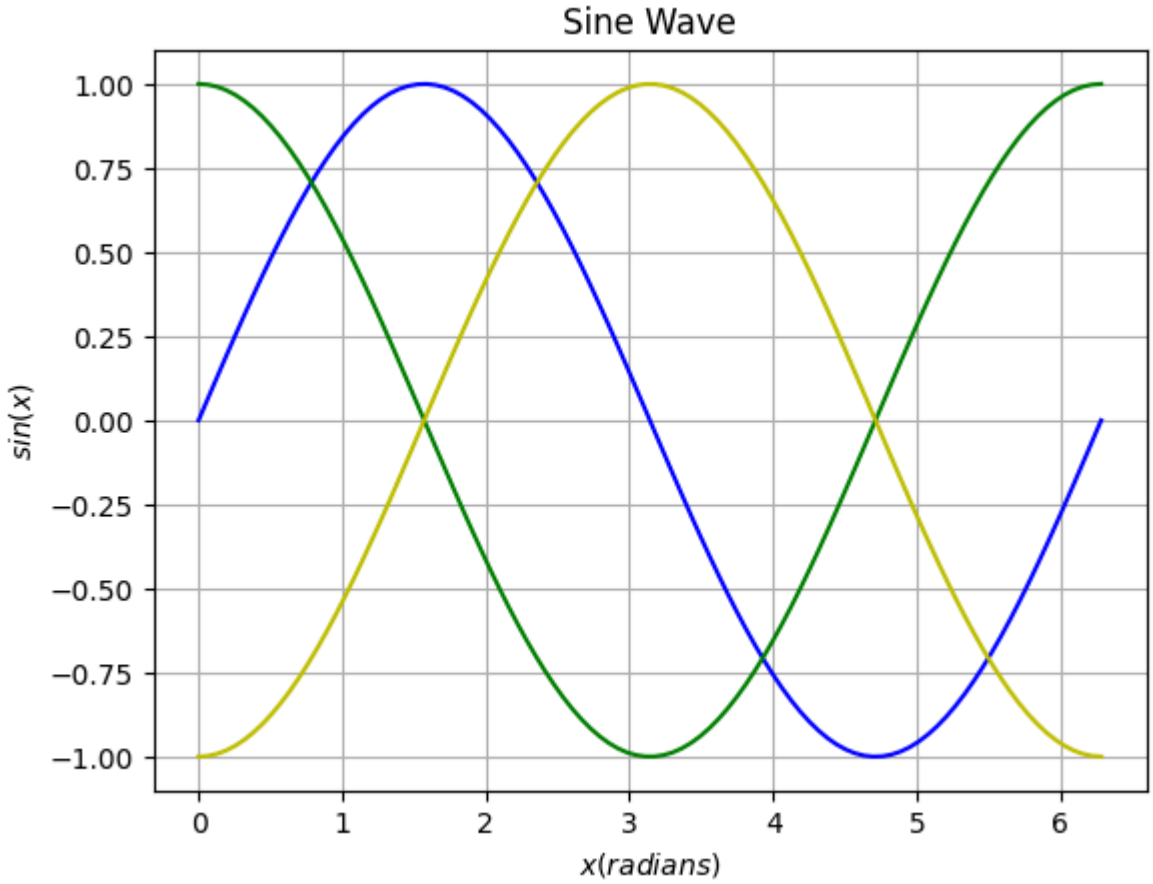
```
1 import math
2 import numpy as np
3 import matplotlib.pyplot as plt
```

이 코드는 Python에서 수학적 계산과 데이터 시각화를 위한 라이브러리들을 가져오는 코드임. 개별 라이브러리의 역할은 다음과 같음:

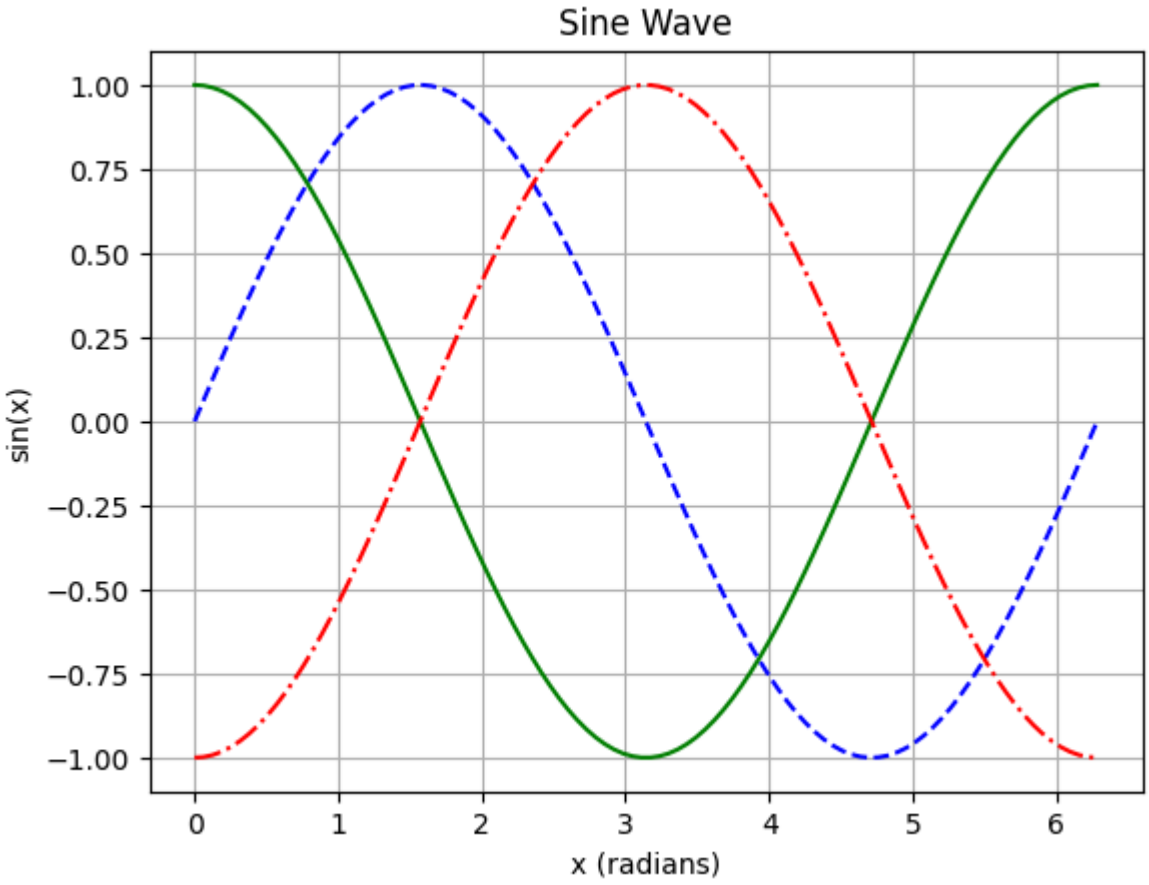
1. `import math`
   - **설명**: Python의 내장 모듈로, 기본적인 수학 함수와 상수를 제공합니다.
   - **주요 기능**:
     - 삼각 함수 (예: `math.sin()`, `math.cos()`)
     - 로그 함수 (예: `math.log()`)
     - 제곱근 (예: `math.sqrt()`)
     - 상수 (예: `math.pi`, `math.e`)
2. `import numpy as np`
   - **설명**: NumPy는 고성능 수치 계산을 위한 라이브러리로, 다차원 배열 객체와 다양한 수학 함수들을 제공합니다.
   - **주요 기능**:
     - 다차원 배열 (ndarray) 생성 및 조작
     - 배열 간의 수학적 연산
     - 선형 대수, 푸리에 변환 및 난수 생성 등의 기능 제공
3. `import matplotlib.pyplot as plt`
   - **설명**: Matplotlib는 데이터 시각화를 위한 라이브러리로, `pyplot` 모듈은 MATLAB 스타일의 그래프를 쉽게 그릴 수 있는 기능을 제공합니다.
   - **주요 기능**:
     - 2D 그래프, 히스토그램, 산점도 등 다양한 유형의 그래프를 생성
     - 그래프의 레이블, 제목, 축 범위 등을 설정
     - 그래프를 파일로 저장하거나 화면에 출력

```
 1 # 데이터 생성
 2 x = np.linspace(0, 2 * math.pi, 100)  # 0부터 2π까지 100개의 점 생성
 3 y1 = np.sin(x)  # x에 대한 sin 함수 값 계산
 4 y2 = np.sin(x + math.pi/2)
 5 y3 = np.sin(x - math.pi/2)
 6
 7 # 그래프 그리기
 8 #plt.figure(figsize=(12, 6))
 9 plt.plot(x,y1,'b',x,y2,'g',x,y3,'y')
10 plt.title('Sine Wave')  # 그래프 제목
11 plt.xlabel('$x (radians)$')  # x축 레이블
```

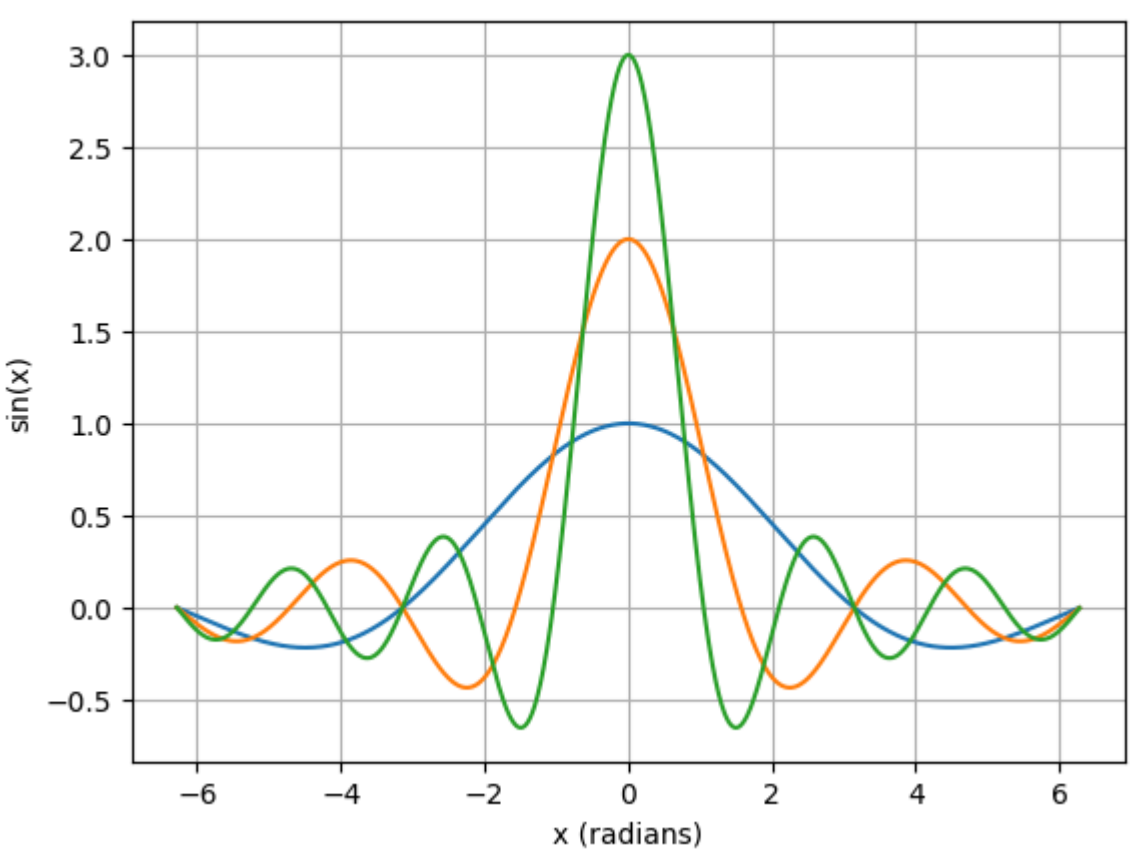
```
plt.ylabel('$sin(x)$')  # y축 레이블
plt.grid(True)  # 그리드 표시
plt.show()  # 그래프 출력
```




```
plt.plot(x,y1,'b--',x,y2,'g',x,y3,'r-.')
plt.title('Sine Wave')  # 그래프 제목
plt.xlabel('x (radians)')  # x축 레이블
plt.ylabel('sin(x)')  # y축 레이블
plt.grid(True)  # 그리드 표시
plt.show()  # 그래프 출력
```




```
# 2nd Exanples
x = np.arange(-2*np.pi,2*np.pi,0.01)
y1 = np.sin(1*x)/x
y2 = np.sin(2*x)/x
y3 = np.sin(3*x)/x

plt.plot(x,y1)
plt.plot(x,y2)
plt.plot(x,y3)
plt.xlabel('x (radians)')  # x축 레이블
plt.ylabel('sin(x)')  # y축 레이블
plt.grid(True)  # 그리드 표시
plt.show()  # 그래프 출력
```



1.4.2 신호의 에너지(Energy)와 전력(Power)

- 신호의 에너지(Energy)
  - 실제 에너지와 다름
  - 신호 특성의 정량적 표현 또는 신호들의 비교 시 유용한 기준

$$\text{연속신호}: E = \lim_{T\to\infty} \int_{-\frac{T}{2}}^{\frac{T}{2}} |x(t)|^2 dt$$

$$\text{이산신호}: E = \lim_{N\to\infty} \sum_{n=-N}^{N} |x[n]|^2$$

- 신호의 전력
  - 신호의 평균 에너지

$$\text{연속신호}: P = \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} |x(t)|^2 dt$$

$$\text{이산신호}: P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x[n]|^2$$

  - 신호의 실효값

    에너지 관점에서 실제 효과는 변함없도록 정의된 전 구간에 불변인 신호의 대푯값

$$\text{연속신호: } x_{rms} = \left(\lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} |x(t)|^2 dt \right)^{\frac{1}{2}}$$

$$\text{이산신호: } x_{rms} = \left(\lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x[n]|^2 \right)^{\frac{1}{2}}$$

- 데시벨(`decibell`)
  신호의 상대적인 크기를 나타내는 데 사용되는 단위
  전력비에 대해서는 $10\log_{10}$, 크기(이득)비에 대해서는 $20\log_{10}$를 취함

$$10\log_{10}\frac{P_o}{P_i} = 10\log_{10}\frac{|x_o|^2}{|x_i|^2} = 20\log_{10}\frac{|x_o|}{|x_i|}\text{[dB]}$$

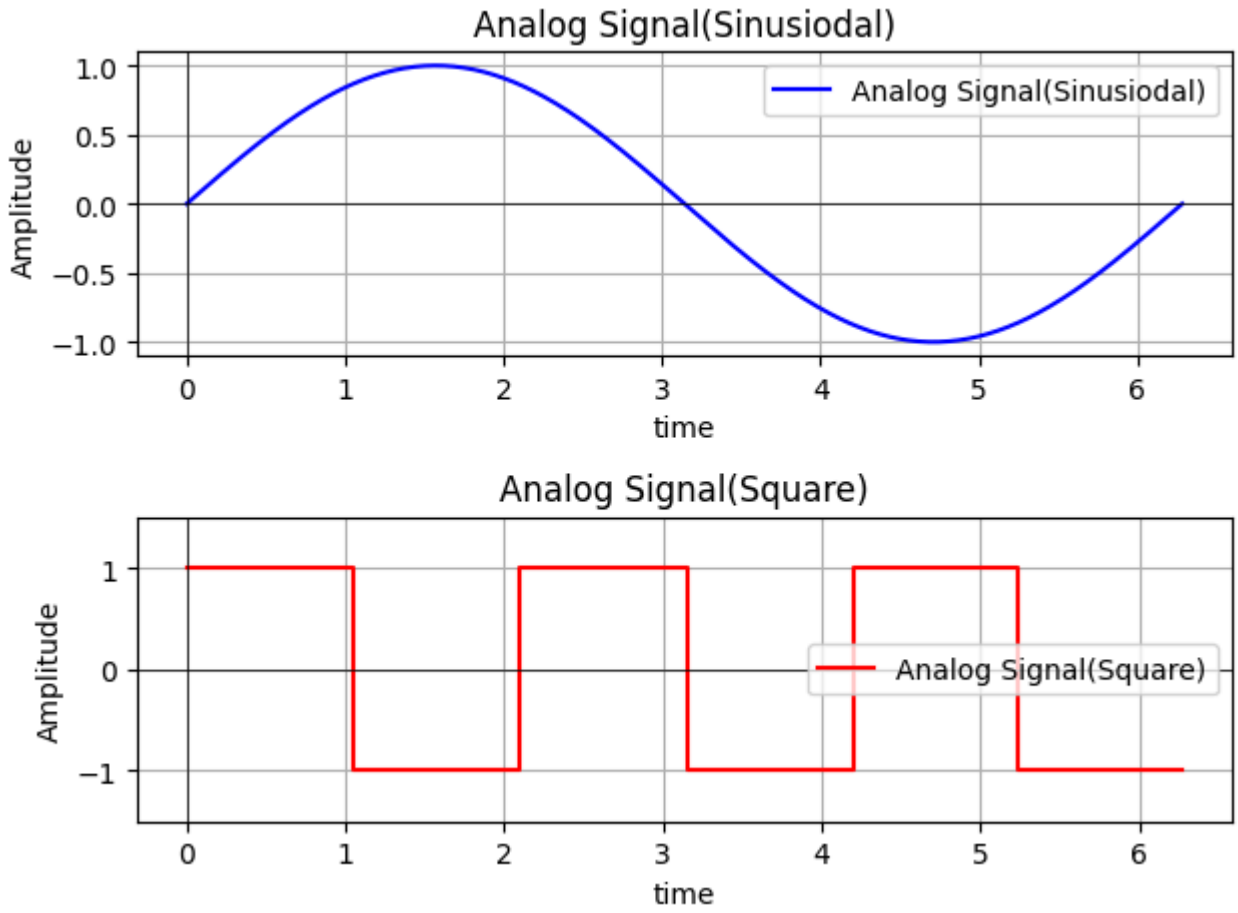
Q&A

[예제] 아날로그 및 디지털 신호 그리기 파이썬 코드

아래는 대표적인 **아날로그 신호** (사인파)와 **디지털 신호** (구형파)를 그리는 파이썬 코드임. `matplotlib` 라이브러리를 사용하며, 각 신호의 특징을 시각적으로 확인할 수 있음.

```
import numpy as np
import matplotlib.pyplot as plt

# --- 아날로그 신호 (사인파) 생성 및 그리기 ---
#plt.figure(figsize=(10, 6)) # 그래프 크기 설정

# 시간 축 (0에서 2파이까지 500개의 점)
t = np.linspace(0, 2 * np.pi, 500)
# 아날로그 사인 신호
analog_signal = np.sin(t)

plt.subplot(2, 1, 1) # 2행 1열의 첫 번째 서브플롯
plt.plot(t, analog_signal, color='blue', label='Analog Signal(Sinusiodal)')
plt.title('Analog Signal(Sinusiodal)')
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.grid(True)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.legend()

# --- 디지털 신호 (구형파) 생성 및 그리기 ---
# 시간 축 (0에서 2파이까지 500개의 점)
t_digital = np.linspace(0, 2 * np.pi, 500, endpoint=False)
# 디지털 구형 신호 (np.sign 함수를 사용하여 양수/음수를 1/-1로 변환)
digital_signal = np.sign(np.sin(t_digital * 3)) # 주파수를 높여 더 많은 구형파 주기 표시
digital_signal[digital_signal == 0] = 1 # 0인 부분을 1로 처리하여 깔끔한 구형파
```

```
28
29 plt.subplot(2, 1, 2) # 2행 1열의 두 번째 서브플롯
30 plt.step(t_digital, digital_signal, where='post', color='red', label='Analog Signal
31 plt.title('Analog Signal(Square)')
32 plt.xlabel('time')
33 plt.ylabel('Amplitude')
34 plt.grid(True)
35 plt.ylim(-1.5, 1.5) # y축 범위 설정
36 plt.axhline(0, color='black', linewidth=0.5)
37 plt.axvline(0, color='black', linewidth=0.5)
38 plt.legend()
39
40 plt.tight_layout() # 서브플롯 간 간격 자동 조절
41 plt.show()
```



코드 설명

1. **라이브러리 import**:
   - `numpy` 는 수학 계산 (특히 배열 연산)을 위해 사용.
   - `matplotlib.pyplot` 은 그래프를 그리는 데 사용.
2. **아날로그 신호 (사인파, 정현파)**:
   - **시간 축 생성**: `np.linspace(0, 2 * np.pi, 500)` 를 사용하여 0부터 2π (한 주기)까지 500개의 균일한 간격의 점을 생성.
   - **사인 신호**: `np.sin(t)` 를 사용하여 각 시간 값에 대한 사인 값을 계산. 사인파는 **연속적인 진폭**을 가지며, 시간의 흐름에 따라 **부드럽게 변화**하는 아날로그의 대표적인 신호임.

- `plt.subplot(2, 1, 1)` : 2행 1열의 서브플롯 중 첫 번째에 그래프를 그림.

3. **디지털 신호 (구형파)**:
   - **시간 축 생성**: 아날로그 신호와 유사하게 시간 축을 생성하지만, `endpoint=False` 를 사용하여 마지막 점을 포함하지 않음. 이는 `step` 함수 사용 시 경계 문제를 줄여줌.
   - **구형파 생성**: `np.sign(np.sin(t_digital * 3))` 을 사용하여 사인파의 부호를 기반으로 구형파를 생성. `np.sign()` 함수는 양수는 1, 음수는 -1, 0은 0으로 반환하므로, 이를 통해 구형파의 형태를 만듬. `t_digital * 3` 은 주파수를 높여 더 많은 구형파 주기를 보여주기 위함.
   - `digital_signal[digital_signal == 0] = 1` : `np.sign()` 함수는 0을 0으로 반환하므로, 0인 부분을 1로 강제로 설정하여 깔끔한 구형파의 형태를 만듬.
   - `plt.step(t_digital, digital_signal, where='post')` : `plt.plot` 대신 `plt.step` 함수를 사용하여 디지털 신호의 계단식 변화를 명확하게 보여줌. `where='post'` 는 계단이 해당 데이터 포인트 이후에 시작됨을 의미함. 디지털 신호는 **유한한 수의 이산적인 값**만을 가지며, **갑작스럽게 변화**하는 특징을 가짐.
   - `plt.subplot(2, 1, 2)` : 2행 1열의 서브플롯 중 두 번째에 그래프를 그림.