

# 깃허브 코파일럿을 이용한 바이브 코딩

김명균

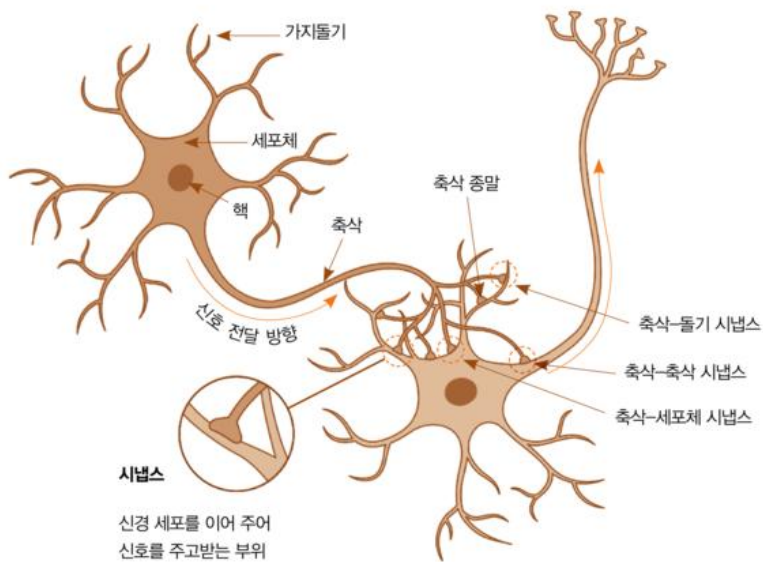
(울산대학교 IT융합학부)

# 바이브코딩을 이용한 사물인터넷

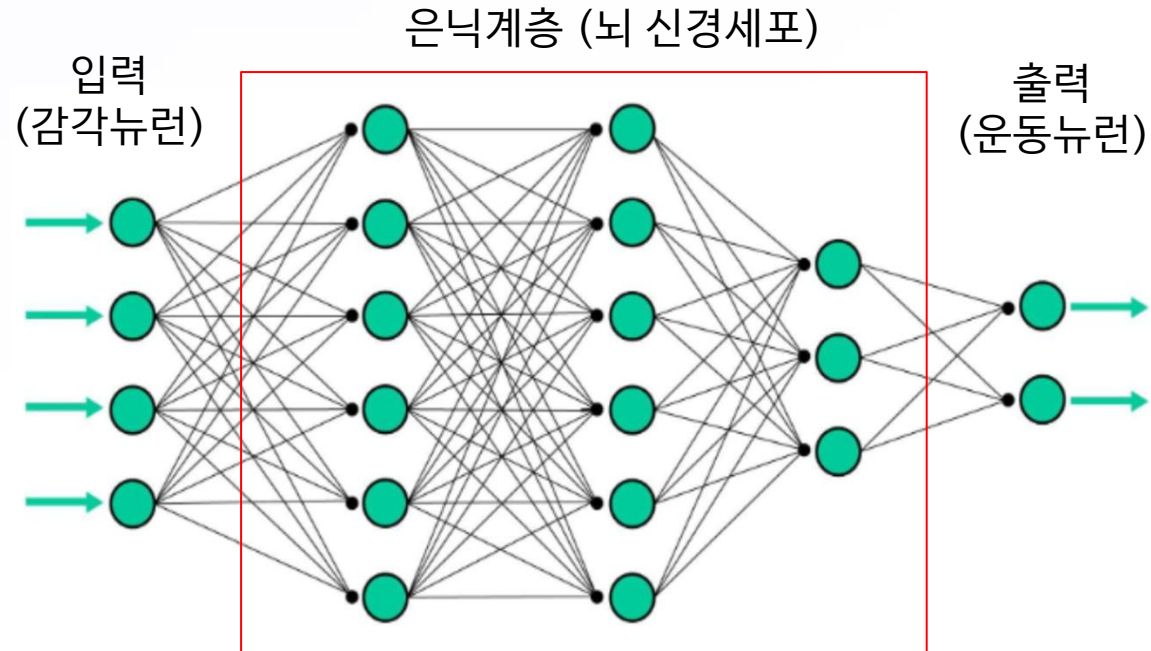
- 라즈베리파이 설치 및 환경 설정
- 깃허브 코파일럿 설치
- 바이브 코딩을 이용한 사물인터넷 – part1

# Artificial Intelligence (AI)

- 뇌 신경망 구조 : 1000억개 뉴론 (신경세포), 100조개의 시냅스(연결)로 구성



뇌 신경망

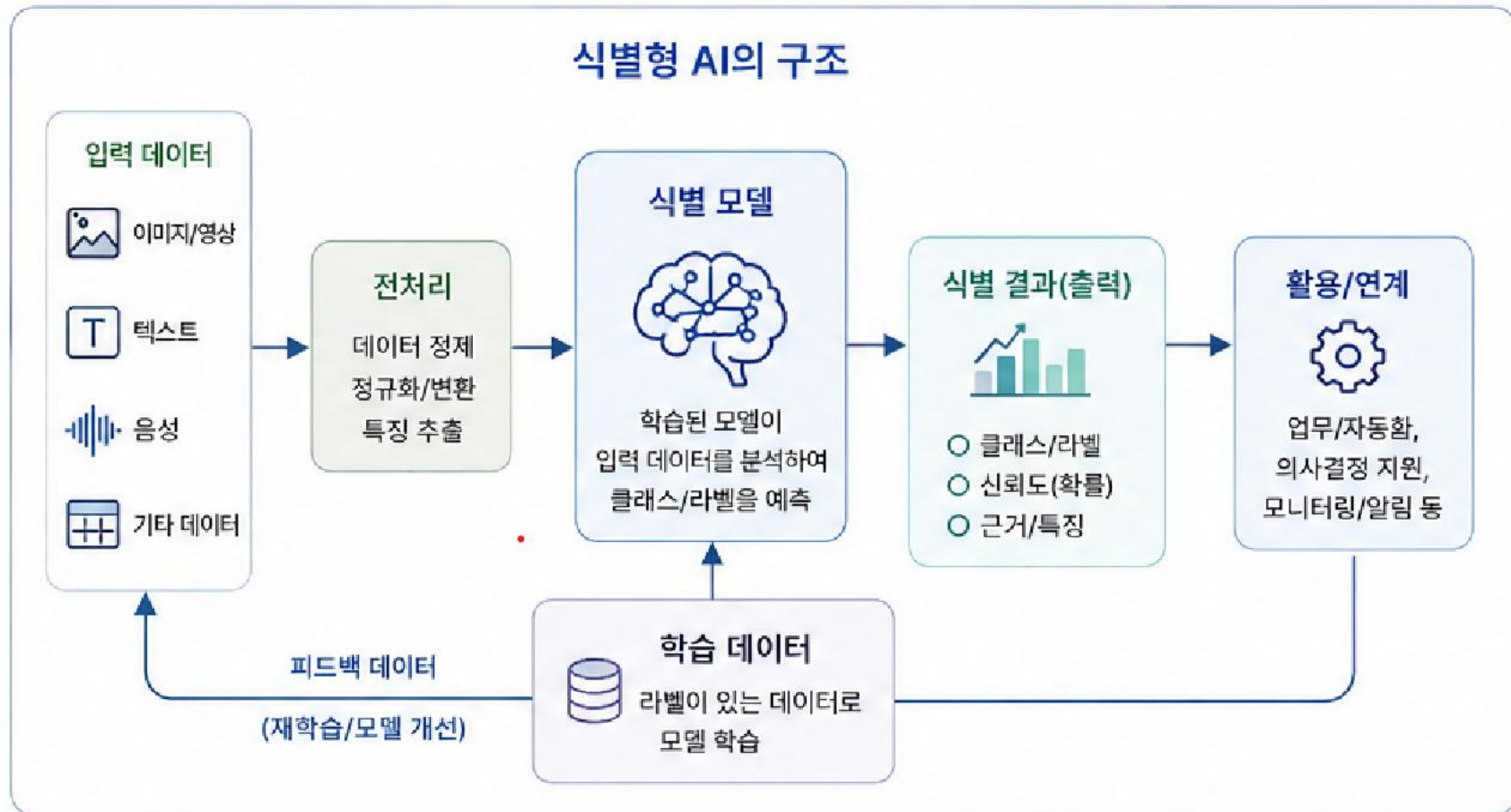


인공 신경망



# Artificial Intelligence (AI)

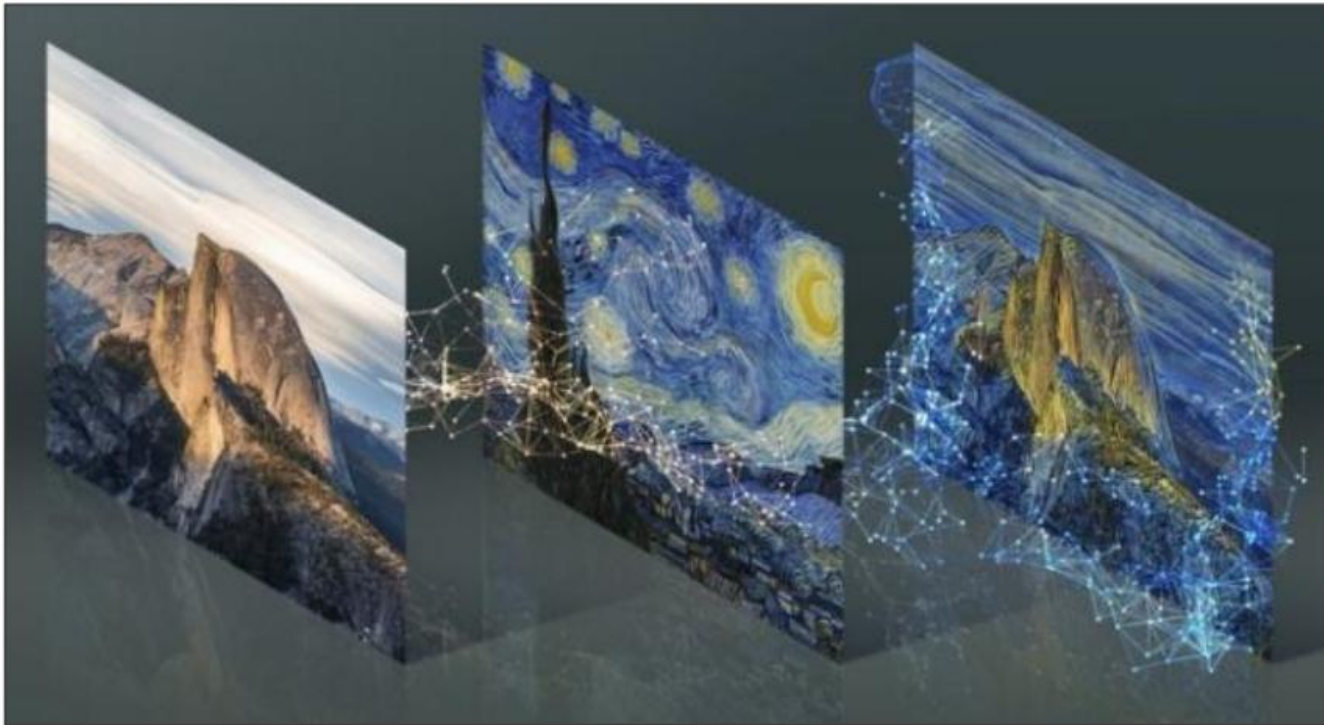
## ■ 식별형 AI



# Artificial Intelligence (AI)

## ■ 생성형 AI (Generative AI)

- 학습한 데이터를 기반으로 새로운 콘텐츠(텍스트, 이미지, 음성, 코드, 영상 등)를 생성하는 AI



# Artificial Intelligence (AI)

## ■ 생성형 AI

### 동작원리



### 기술적 기반



대규모 데이터

웹, 도서, 이미지, 코드 등 방대한 데이터



머신러닝 모델

딥러닝 모델(예: Transformer)로 패턴 학습



생성 알고리즘

확률적 예측을 통해 가장 적합한 결과 생성



후처리/정제

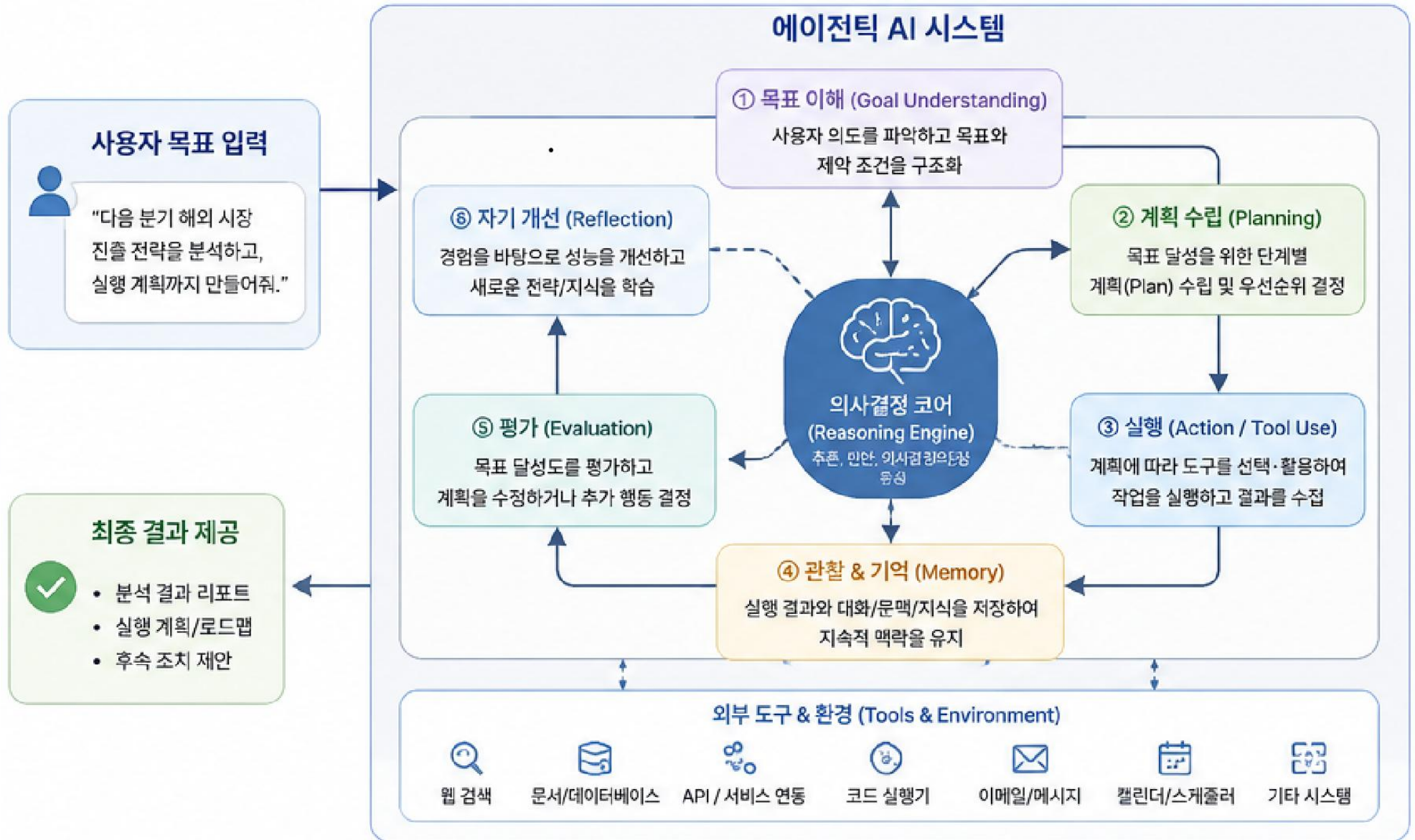
안전성 필터, 품질 향상, 형식 변환 등

# Artificial Intelligence (AI)

## ■ 에이전틱 AI (Agentic AI)

- 인간이 설정한 목표를 이해하고, 계획을 세우며, 도구를 사용해 실행·평가하고 결과를 만들어내는 자율형 AI
- **목표 달성을 위해 스스로 계획·실행·판단하는 AI**
- 여러 도구와 시스템을 연동하여 작업 수행
- 반복적 의사결정 및 자율적 행동 가능
- 인간 협업형 디지털 에이전트

# Artificial Intelligence (AI)



# Visual Studio Code Copilot

EXPLORER

COPILOT

- > apple\_game
- > data
- > my\_profile
- > todo
- > utube
- JS script.js
- # styles.css

open folder → copilot 폴더 선택

파일탐색기, 코드관리, 확장도구관리 창

OUTLINE

The active editor cannot provide outline information.

TIMELINE

```
settings.json
```

```
C: > Users > LG > AppData > Roaming > Code > User > {} settings.json > ...
```

```
1 {
2   "remote.SSH.remotePlatform": {
3     "169.254.198.89": "linux",
4     "rp-108177": "linux"
5   },
6   "python.defaultInterpreterPath": "C:\\Users\\LG\\AppData\\Local\\...
7   "workbench.editor.empty.hint": "hidden",
8   "claudeCode.preferredLocation": "panel",
9   "workbench.colorTheme": "Visual Studio Light"
10 }
```

코드편집기창

명령창

CHAT

CLAUDE CODE

COPILOT 플랜 정보 확인 방법

codex 확장을 설치하지 않아도 되나요?

네, 보통은 Codex 전용 확장을 따로 설치할 필요 없습니다.

VS Code에서 GitHub Copilot + GitHub Copilot Chat 확장만 있으면, 모델 선택기에서 Codex 계열 모델을 바로 선택해 사용할 수 있습니다.

예외적으로 필요한 경우:

1. 회사/조직 정책으로 특정 모델이 막힌 경우
2. 아직 계정에 해당 모델 권한이 없는 경우
3. Copilot/Copilot Chat 확장이 구버전인 경우

즉, 핵심은 확장 추가 설치보다 플랜 권한 + 최신 Copilot 확장 + 모델 선택 가능 여부입니다. 원하시면 지금 화면 기준으로 30초 점 검 순서 알려드릴게요.

채팅창

+ settings.json

Describe what to build

+ Agent GPT-5.3-Codex · Medium

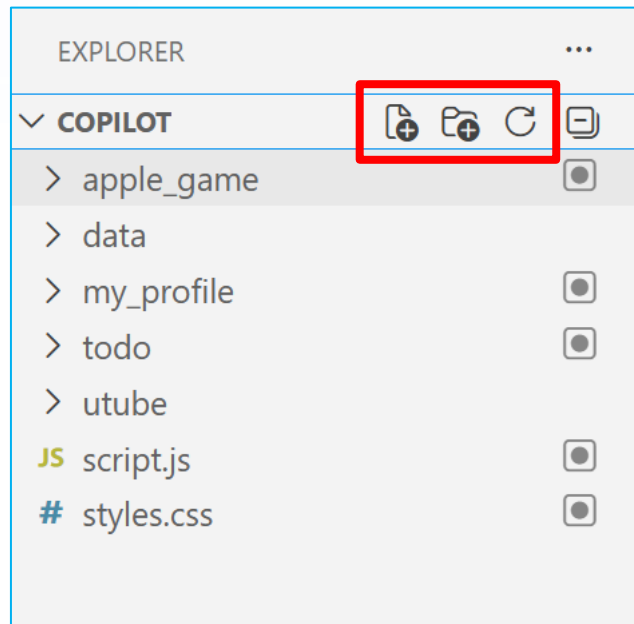
Local Default Approvals

Ln 10, Col 2 Spaces: 4 UTF-8 LF { } JSON with Comments

# Visual Studio Code Copilot

## ■ VS Code Copilot

- 채팅창: 프롬프트입력, 모드선택(**Agent/Plan/Ask**), AI모델선택
  - ✓ 모드: **Plan** (계획을 세우는 모드), **Ask** (질문 모드), **Agent** (동작 실행 모드)
- 코드 편집기창
- 명령창
- 탐색창:



New File / New Folder /  
Refresh Folder

# 바이브 코딩: 게임 프로그램 만들기

## ■ 행맨 게임 만들기

- 탐색기: "Study02" 폴더 생성:  
오른쪽버튼 클릭 → "Add Folder to Chat"
- 채팅창: Plan 모드, 프롬프트 입력

영어 단어 맞추는 hangman 게임을 웹 버전으로 만들고 싶어. 게임 규칙은 다음과 같아.

규칙:

1. 영어 단어를 선택하고 단어 길이 만큼 \_ 를 표시하고, 화면에는 교수대 그림이 나온다.
2. 사용자가 단어에 들어갈 영문자를 예측하고, 단어에 해당 문자가 들어있으면 해당위치에 표시하고, 없으면 머리, 목, 두 팔, 두 다리 순으로 틀릴 때마다 하나씩 교수대에 표시한다.
3. 6번 틀려 두 다리까지 나타나면 게임에 실패하고, 이전에 단어를 모두 맞추면 승리한다.
4. 점수계산은 만점 6점에서 교수대에 매달린 신체 숫자 만큼 감점된다.

요청하신 규칙을 기준으로 바로 구현 가능한 계획을 만들기 위해, 먼저 현재 작업 폴더 구조와 재사용 가능한 게임 예제를 다시 확인한 뒤 세부 실행 계획으로 정리하겠습니다.

# 바이브 코딩: 게임 프로그램 만들기

## ■ 행맨 게임 만들기

### Plan: 웹 Hangman 게임 구현

주신 규칙을 그대로 반영해서 Study02에 단일 페이지 게임을 만드는 계획입니다. 핵심은 상태 기반 렌더링으로 단어 표시, 교수대 진행, 승패, 점수를 한 흐름으로 동기화하는 방식입니다.

### Steps

1. Phase 1 - 화면 골격 구성: Study02 폴더에 새 HTML/CSS/JS 3파일 구조를 만들고, 화면에 제목, 교수대 영역, 단어 밑줄 표시, 알파벳 입력 버튼, 상태 메시지, 점수, 새 게임 버튼을 배치합니다.
2. Phase 1 - 기본 연결: HTML에서 스타일/스크립트를 연결하고 버튼 및 상태영역에 접근성 속성을 넣습니다.
3. Phase 2 - 상태 모델 작성: 선택 단어, 추측 문자 목록, 오답 횟수, 게임 종료 여부, 점수 상태를 정의하고 초기화 함수를 만듭니다. (depends on 1)
4. Phase 2 - 추측 처리 로직: 문자 선택 시 정답/오답 판정, 중복 입력 무시, 단어 공개 갱신을 처리합니다. (depends on 3)

5. Phase 2 - 교수대 진행/패배 처리: 오답 시 머리→목→두 팔→두 다리 순서로 6단계 진행하고, 6회 시 패배 처리 및 정답 공개를 수행합니다. (depends on 4)
6. Phase 2 - 승리 처리: 단어 전체가 공개되면 즉시 승리 처리하고 입력을 비활성화합니다. (depends on 4)
7. Phase 2 - 점수 규칙 반영: 점수는 6 - 오답횟수로 계산하고 최소 0점 보정합니다. (depends on 5, 6)
8. Phase 3 - 스타일 마감: 심플한 레이아웃, 버튼 상태(선택/정답/오답), 교수대 단계 시각화, 모바일 반응형을 적용합니다. (parallel with 7 after selector fixed)
9. Phase 4 - QA: 승리/패배/중복입력/점수/재시작 시나리오를 수동 검증합니다. (depends on 7, 8)

### Relevant files

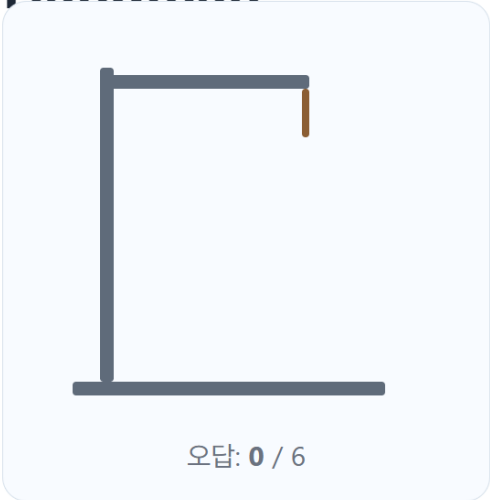
1. 새로 생성할 대상: Study02 폴더 내부의 HTML/CSS/JS 3개 파일
2. 참고 템플릿: `<> index.html`
3. 참고 로직 패턴: `js script.js`
4. 참고 스타일 패턴: `# styles.css`

# 바이브 코딩: 게임 프로그램 만들기

## ■ 행맨 게임 실행

- 채팅창: **Agent** 모드 → **<Start Implementation>** 선택

### Hangman



오답: 0 / 6

— — — **E** — — —

좋아요! 'E'가 포함되어 있습니다.

점수: 6 / 6

A	B	C	D	<b>E</b>	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

새 게임

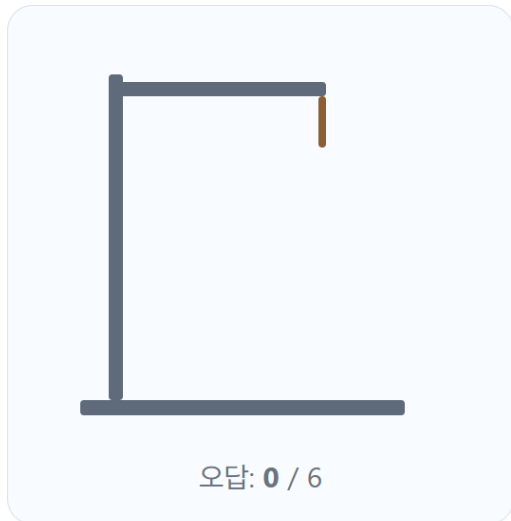
# 바이브 코딩: 게임 프로그램 만들기

## ■ 행맨 게임 실행: 게임 수정

게임 제목과 교수대가 조금 겹치는데 겹치지 않도록 다시 만들어서 "Study03" 폴더에 만들어줘.

### Hangman

영어 단어를 맞춰보세요. 오답 6회면 게임 종료입니다.



— — — — —

알파벳 버튼을 눌러 시작하세요.

점수: 6 / 6



새 게임

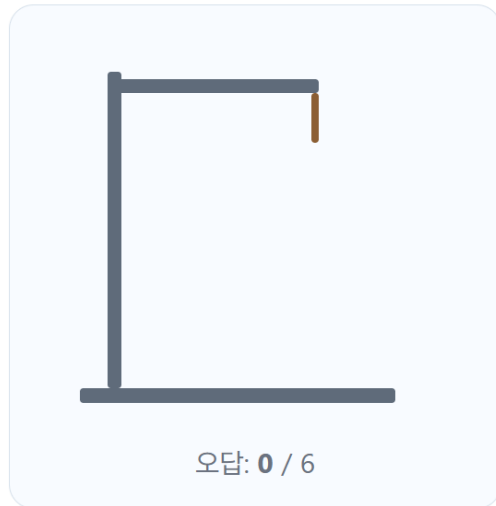
# 바이브 코딩: 게임 프로그램 만들기

## ■ 행맨 게임 실행: 게임 수정2

단어 길이가 6자로 제한되어 있는 것 같은데, 단어길이를 임의의 길이로 만들어 주고, 이 게임을 "Study04" 폴더에 만들어줘.

### Hangman

단어 길이는 매번 달라집니다. 오답 6회면 게임 종료입니다.



단어 길이: 8 글자

-----

알파벳 버튼을 눌러 시작하세요.

점수: 6 / 6



새 게임

# 바이브 코딩: 게임 프로그램 만들기

## ■ 웹사이트 구성

- HTML 파일: .html
  - ✓ 웹 문서의 구성 정보 포함
- CSS 파일: .css
  - ✓ 웹 페이지 화면을 어떻게 시각적 표시할 지 결정 (배경 이미지, 글자 크기 및 폰트 등)
- Java Script 파일: .js
  - ✓ 웹 사이트에 동적인 기능을 추가 (버튼 클릭시 동작 등)

# 바이브코딩을 이용한 사물인터넷

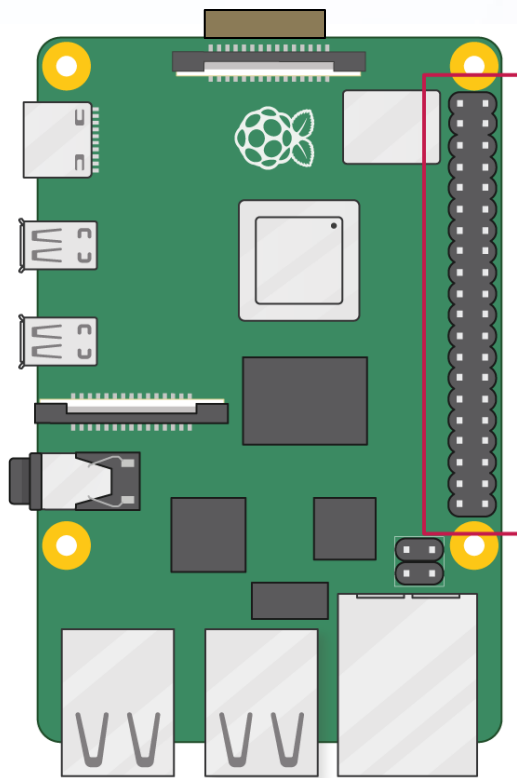
- 라즈베리파이 설치 및 환경 설정
- 깃허브 코파일럿 설치
- 바이브 코딩을 이용한 사물인터넷 – part2

# 라즈베리파이 GPIO

- **라즈베리파이 GPIO (General Purpose Input and Output) : 외부 디바이스와 I/O**
  - GPIO port: 26 ports in Raspberry Pi
  - some port can be used for serial communications (SPI, I2C, UART, etc.)
  - power pin: 3.3V, 5V pin
  - Raspberry Pi operates 3.3V internally; GPIO pin operates 3.3V as well
  - need to use voltage level converter to use 5V device in GPIO pin

# 라즈베리파이 GPIO

## GPIO PINs

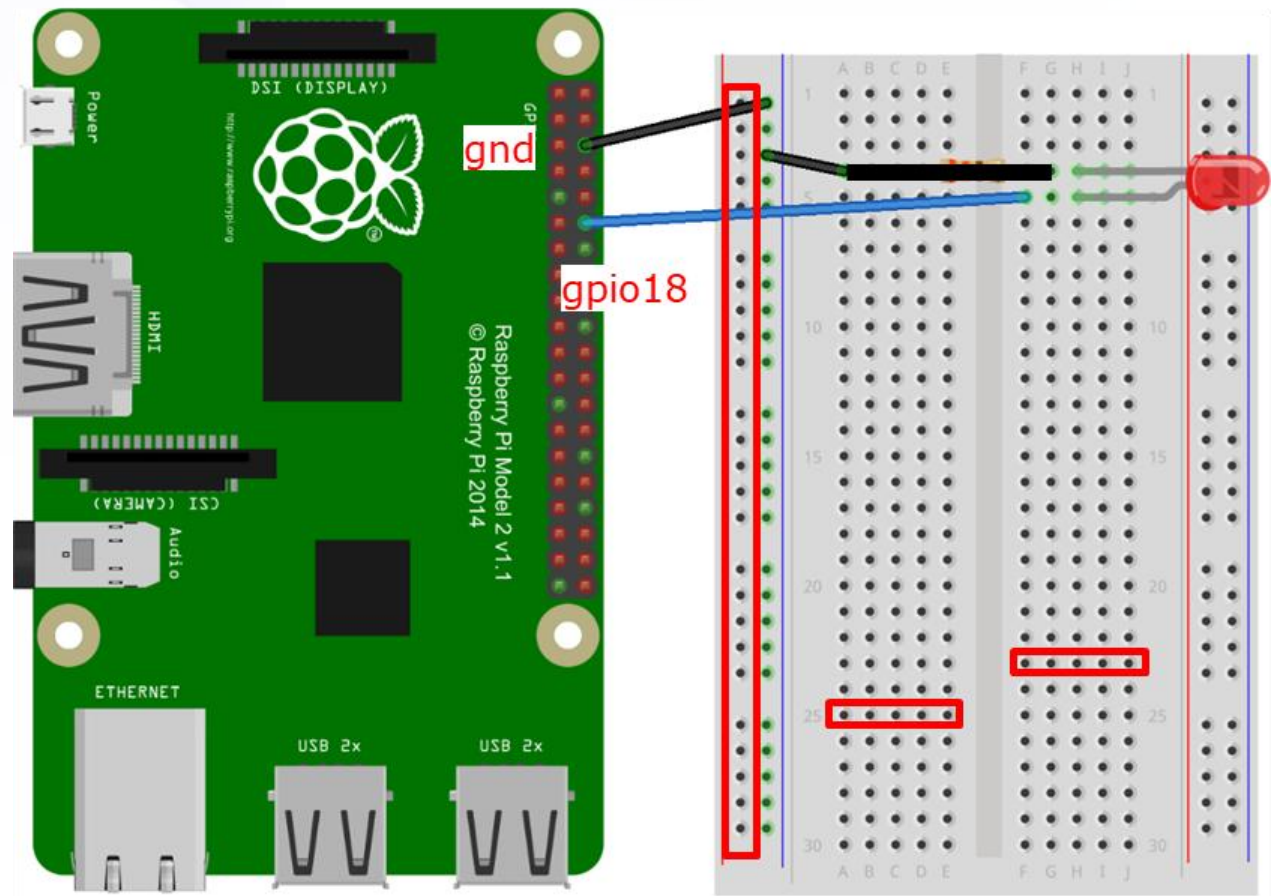
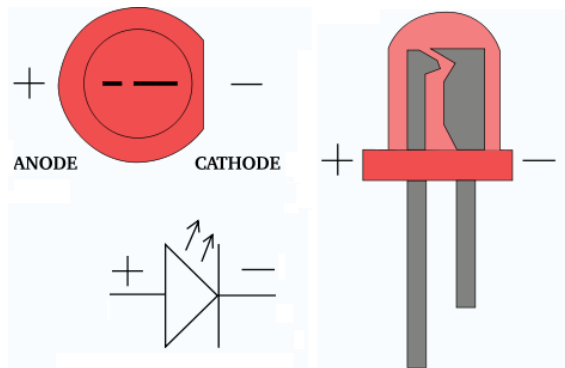
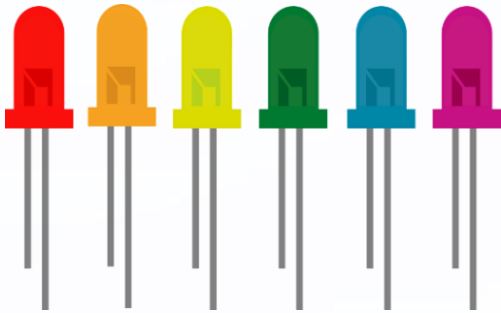


python			python		
WPI	BCM/Name	물리적 구성(핀)	BCM/Name	WPI	
	3.3V	1	2	5V	
8	GPIO 2/I2C SDA	3	4	5V	
9	GPIO 3/I2C SCL	5	6	Ground	
7	GPIO 4	7	8	GPIO 14/TXD	
	Ground	9	10	GPIO 15/RXD	
0	GPIO 17	11	12	GPIO 18	
2	GPIO 27	13	14	Ground	
3	GPIO 22	15	16	GPIO 23	
	3.3 V	17	18	GPIO 24	
12	GPIO 10/SPI MOSI	19	20	Ground	
13	GPIO 9/SPI MISO	21	22	GPIO 25	
14	GPIO 11/SPI CLK	23	24	GPIO 8/SPI_CE0	
	Ground	25	26	GPIO 7/SPI_CE1	
30	ID_SD	27	28	ID_SC	
21	GPIO 5	29	30	Ground	
22	GPIO 6	31	32	GPIO 12	
23	GPIO 13	33	34	Ground	
24	GPIO 19	35	36	GPIO 16	
25	GPIO 26	37	38	GPIO 20	
	Ground	39	40	GPIO 21	

# 사물인터넷 프로그래밍: LED

## ■ 바이브코딩: led on/off

### ■ led 연결하기:



# 사물인터넷 프로그래밍: LED

## ■ 바이브코딩: led on/off

- 작업 폴더: `PS C:\MKKIM\Class\바이브코딩\Copilot\RaspberryPi\led> code`
- VS code 채팅창:

현재 라즈베리파이가 연결되어 있어 호스트 이름은 `rp84041` 이고 사용자는 pi 야. ssh 로 연결하고 홈폴더 아래에 IoT 라는 폴더를 만들어줘.

Worked for 44s >

`pi@rp84041` 에 SSH로 접속해서 `~/IoT` 폴더를 생성했습니다.

실행 결과:

- `mkdir -p ~/IoT`
- 응답: `CREATED`

명령창을 통해 라즈베리파이 접속하여 확인

# 사물인터넷 프로그래밍: LED

## ■ 바이브코딩: led on/off

- VS code 채팅창:  
<Agent> 모드

라즈베리파이의 gpio18 에 led 가 연결되어 있어, 이 led를 1초 간격으로 켜다가 끄다가 하는 파이썬 프로그램을 만들어줘. gpio 제어는 RPi.GPIO 라이브러리를 이용해줘.

9:33 PM

Worked for 47s >

blink\_led.py 를 만들었고, 라즈베리파이의 `~/IoT/blink_led.py` 로도 복사해줬습니다.

파일:

- [blink\\_led.py](#)

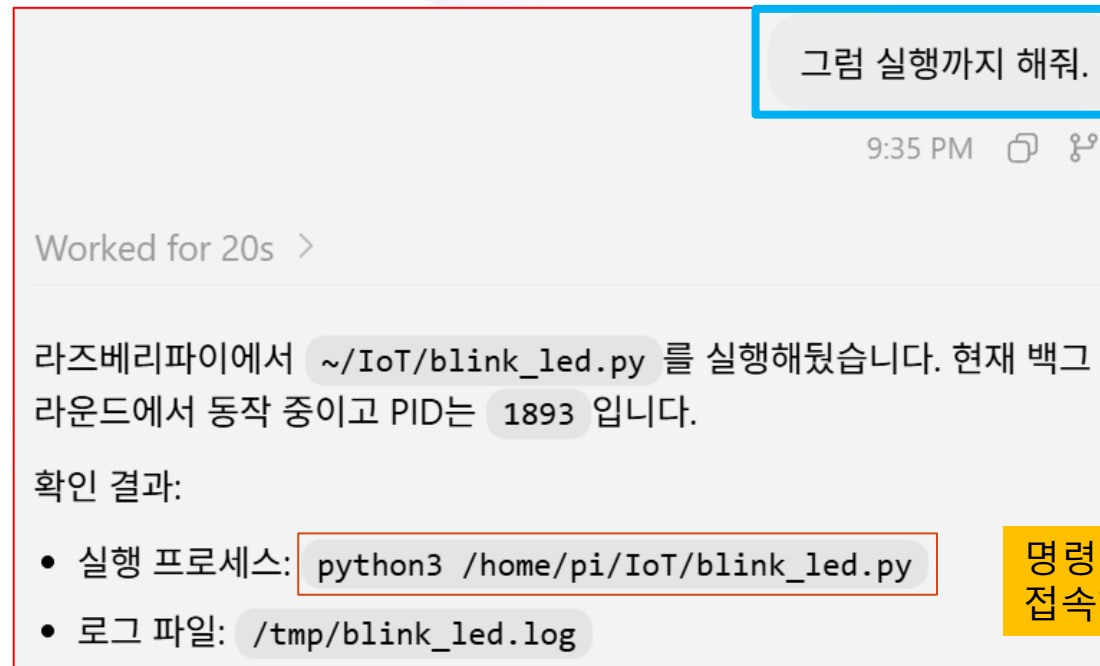
동작:

- `GPIO.BCM` 모드 사용
- `GPIO18` 을 출력으로 설정
- 1초 켜짐, 1초 꺼짐 반복
- `Ctrl+C` 로 종료하면 LED를 끄고 `GPIO.cleanup()` 수행

# 사물인터넷 프로그래밍: LED

## ■ 바이브코딩: led on/off

### ■ 실행:



그럼 실행까지 해줘.

9:35 PM

Worked for 20s >

라즈베리파이에서 `~/IoT/blink_led.py` 를 실행해줬습니다. 현재 백그라운드에서 동작 중이고 PID는 `1893` 입니다.

확인 결과:

- 실행 프로세스: `python3 /home/pi/IoT/blink_led.py`
- 로그 파일: `/tmp/blink_led.log`

명령창을 통해 라즈베리파이 접속하여 실행해도 됨

### ■ LED 깜박임 확인

# 사물인터넷 프로그래밍: LED

## ■ 바이브코딩: led on/off

- 파이썬 코드: 코드편집기창

```
import time

import RPi.GPIO as GPIO

LED_PIN = 18
BLINK_INTERVAL_SECONDS = 1
```

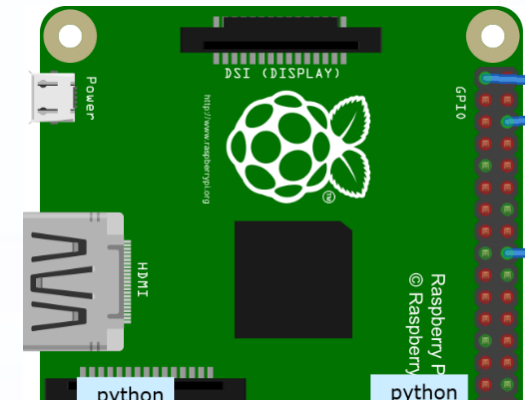
```
def main() -> None:
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(LED_PIN, GPIO.OUT, initial=GPIO.LOW)

    try:
        while True:
            GPIO.output(LED_PIN, GPIO.HIGH)
            time.sleep(BLINK_INTERVAL_SECONDS)
            GPIO.output(LED_PIN, GPIO.LOW)
            time.sleep(BLINK_INTERVAL_SECONDS)
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.output(LED_PIN, GPIO.LOW)
        GPIO.cleanup()

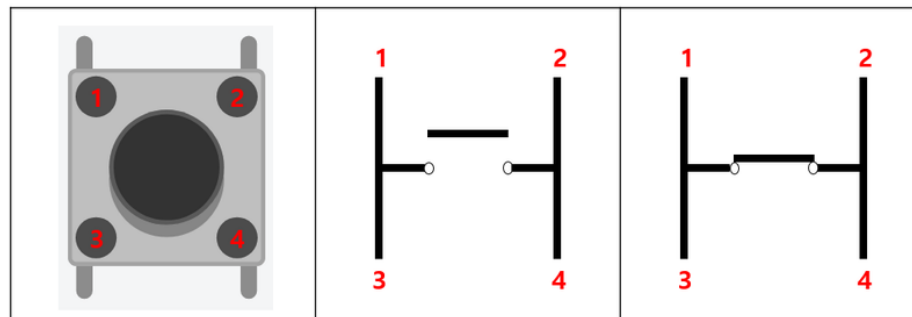
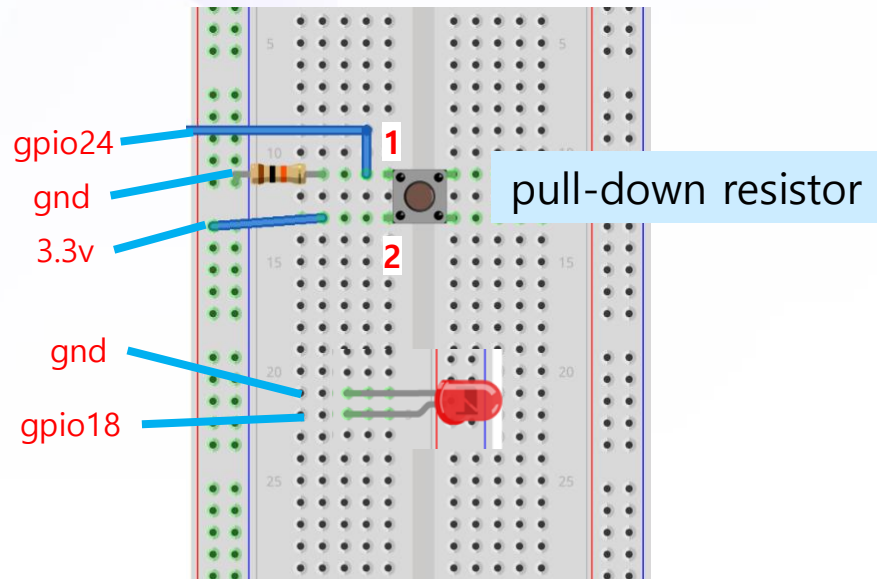
if __name__ == "__main__":
    main()
```

# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 사용하기: 버튼, led 연결



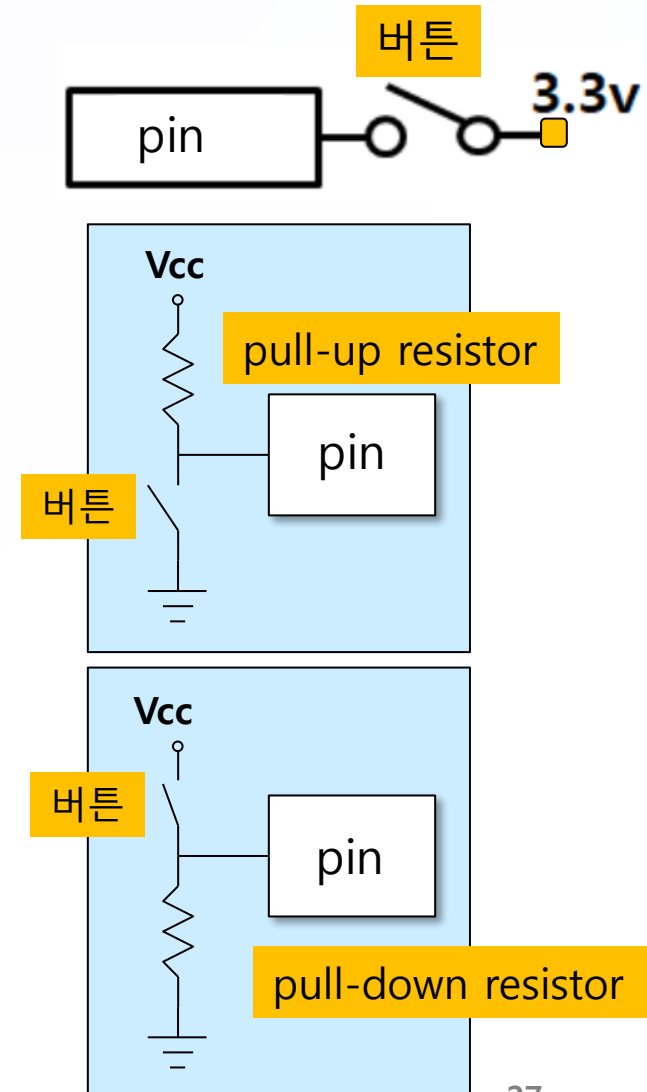
WPI	BCM/Name	물리적 구성(핀)	BCM/Name	WPI
	3.3V	1	2	5V
8	GPIO 2/I2C SDA	3	4	5V
9	GPIO 3/I2C SCL	5	6	Ground
7	GPIO 4	7	8	GPIO 14/TXD
	Ground	9	10	GPIO 15/RXD
0	GPIO 17	11	12	GPIO 18
2	GPIO 27	13	14	Ground
3	GPIO 22	15	16	GPIO 23
	3.3 V	17	18	GPIO 24
12	GPIO 10/SPI MOSI	19	20	Ground
13	GPIO 9/SPI MISO	21	22	GPIO 25
14	GPIO 11/SPI CLK	23	24	GPIO 8/SPI_CE0
	Ground	25	26	GPIO 7/SPI_CE1
30	ID_SD	27	28	ID_SC
21	GPIO 5	29	30	Ground
22	GPIO 6	31	32	GPIO 12
23	GPIO 13	33	34	Ground
24	GPIO 19	35	36	GPIO 16
25	GPIO 26	37	38	GPIO 20
	Ground	39	40	GPIO 21
				29



# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 연결

- 저항 연결: to remove floating state
- Pull-up resistor
  - connect resistor to Vcc
  - if SW=off, Vcc is supplied to device
  - if SW=on, 0v is supplied to device
- Pull-down resistor
  - connect resistor to GND
  - if SW=off, 0v is supplied to device
  - if SW=on, Vcc is supplied to device



# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 이용하기: vs code 채팅창

- 작업폴더: button-led

로컬네트워크 상의 rp84041 이라는 라즈베리파이 호스트에 아래와 같은 센서들이 연결되어 있어

[버튼] 풀다운 저항으로 gpio24 번 핀에 연결

[led] gpio18 번 핀에 연결, 초기상태는 led off

버튼을 누르면 led가 켜지고, 다시 누르면 꺼지고, 이렇게 계속 반복되는 프로그램을 RPi.GPIO 라이브러리를 이용하여 만들어줘. 버튼 동작은 인터럽트를 이용해줘. Ctrl-c 를 누르면 프로그램이 종료됨.

이 프로그램을 만들어서 라즈베리파이 pi 홈폴더 아래의 IoT 폴더에 저장해줘.

# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 이용하기: vs code 채팅창

- 작업폴더: button-led
- 결과: 쓰레드를 이용하는 프로그램을 작성 (이해하기 어려움)
- 프로그램 수정:

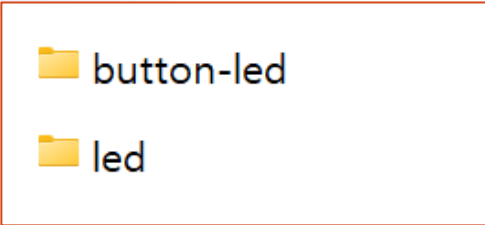
프로그램을 쓰레드를 이용하지 말고 하나의 쓰레드만 동작하도록 만들어줘.

이 프로그램을 라즈베리파이 pi 홈폴더 아래에 있는 IoT 폴더에 저장해줘.

# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 이용하기: vs code 채팅창

- 작업폴더: button-led



button-led

led

- 라즈베리파이 접속

```
pi@rp84041:~ $ cd IoT
pi@rp84041:~/IoT $ ls
blink_led.py  button_led_interrupt.py  __pycache__
```

```
pi@rp84041:~/IoT $ python3 button_led_interrupt.py
Ready. Press the button to toggle the LED. Press Ctrl-C to exit.
Button pressed: LED ON
Button pressed: LED OFF
Button pressed: LED ON
Button pressed: LED OFF
Button pressed: LED ON
Button pressed: LED OFF
^C
GPIO cleaned up. Bye.
```

# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 이용하기: 생성 코드

```
3 import time
4
5 import RPi.GPIO as GPIO
6
7
8 BUTTON_PIN = 24
9 LED_PIN = 18
10 BOUNCE_TIME_MS = 200
11
12 led_on = False
13
14
15 def toggle_led():
16     global led_on
17
18     led_on = not led_on
19     GPIO.output(LED_PIN, GPIO.HIGH if led_on else GPIO.LOW)
20     print(f"Button pressed: LED {'ON' if led_on else 'OFF'}")
```

## ■ 버튼 이용하기: 생성 코드

```
23 def main():
24     GPIO.setmode(GPIO.BCM)
25     GPIO.setwarnings(False)
26
27     GPIO.setup(LED_PIN, GPIO.OUT, initial=GPIO.LOW)
28     GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
29
30     print("Ready. Press the button to toggle the LED. Press Ctrl-C")
31
32     try:
33         while True:
34             channel = GPIO.wait_for_edge(
35                 BUTTON_PIN,
36                 GPIO.RISING,
37                 bouncetime=BOUNCE_TIME_MS,
38             )
39
40             if channel == BUTTON_PIN:
41                 toggle_led()
42                 time.sleep(0.05)
43     except KeyboardInterrupt:
44         pass
45     finally:
46         GPIO.output(LED_PIN, GPIO.LOW)
47         GPIO.cleanup()
48         print("\nGPIO cleaned up. Bye.")
49
50
51 if __name__ == "__main__":
52     main()
```

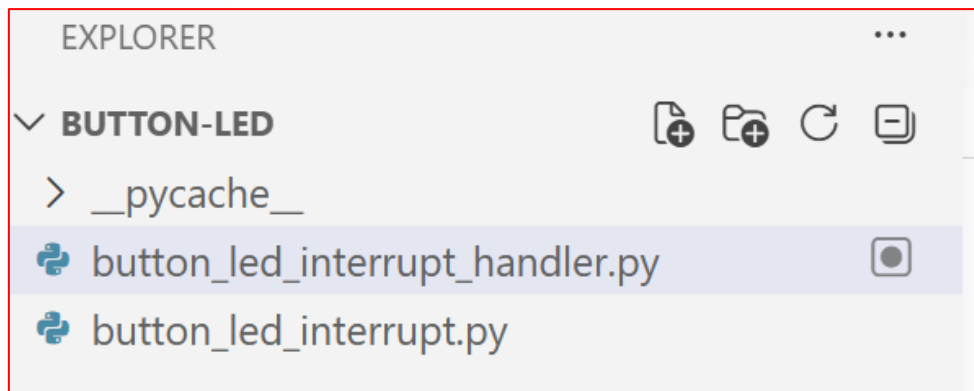
# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 이용하기: 인터럽트 핸들러 이용

### ■ 프롬프트:

이 프로그램을 `wait_for_edge` 함수를 사용하지 말고, 인터럽트 핸들러를 사용하는 프로그램으로 만들고 이 프로그램을 새로운 파일에 저장해줘.

### ■ 작업폴더: button-led



# 사물인터넷 프로그래밍: Button, LED

## ■ 버튼 이용하기: 인터럽트 핸들러 이용

```
3 import time
4
5 import RPi.GPIO as GPIO
6
7 BUTTON_PIN = 24
8 LED_PIN = 18
9 BOUNCE_TIME_MS = 200
10
11 led_on = False
12
13
14 def button_callback(channel):
15     global led_on
16
17     led_on = not led_on
18     GPIO.output(LED_PIN, GPIO.HIGH if led_on else GPIO.LOW)
19     print(f"Button pressed: LED {'ON' if led_on else 'OFF'}")
```

## ■ 버튼: 인터럽트 핸들러

```
def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)

    GPIO.setup(LED_PIN, GPIO.OUT, initial=GPIO.LOW)
    GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

    GPIO.add_event_detect(
        BUTTON_PIN,
        GPIO.RISING,
        callback=button_callback,
        bouncetime=BOUNCE_TIME_MS,
    )

    print("Ready. Press the button to toggle the LED. Press Ctrl-C to exit.")

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.output(LED_PIN, GPIO.LOW)
        GPIO.cleanup()
        print("\nGPIO cleaned up. Bye.")

if __name__ == "__main__":
    main()
```