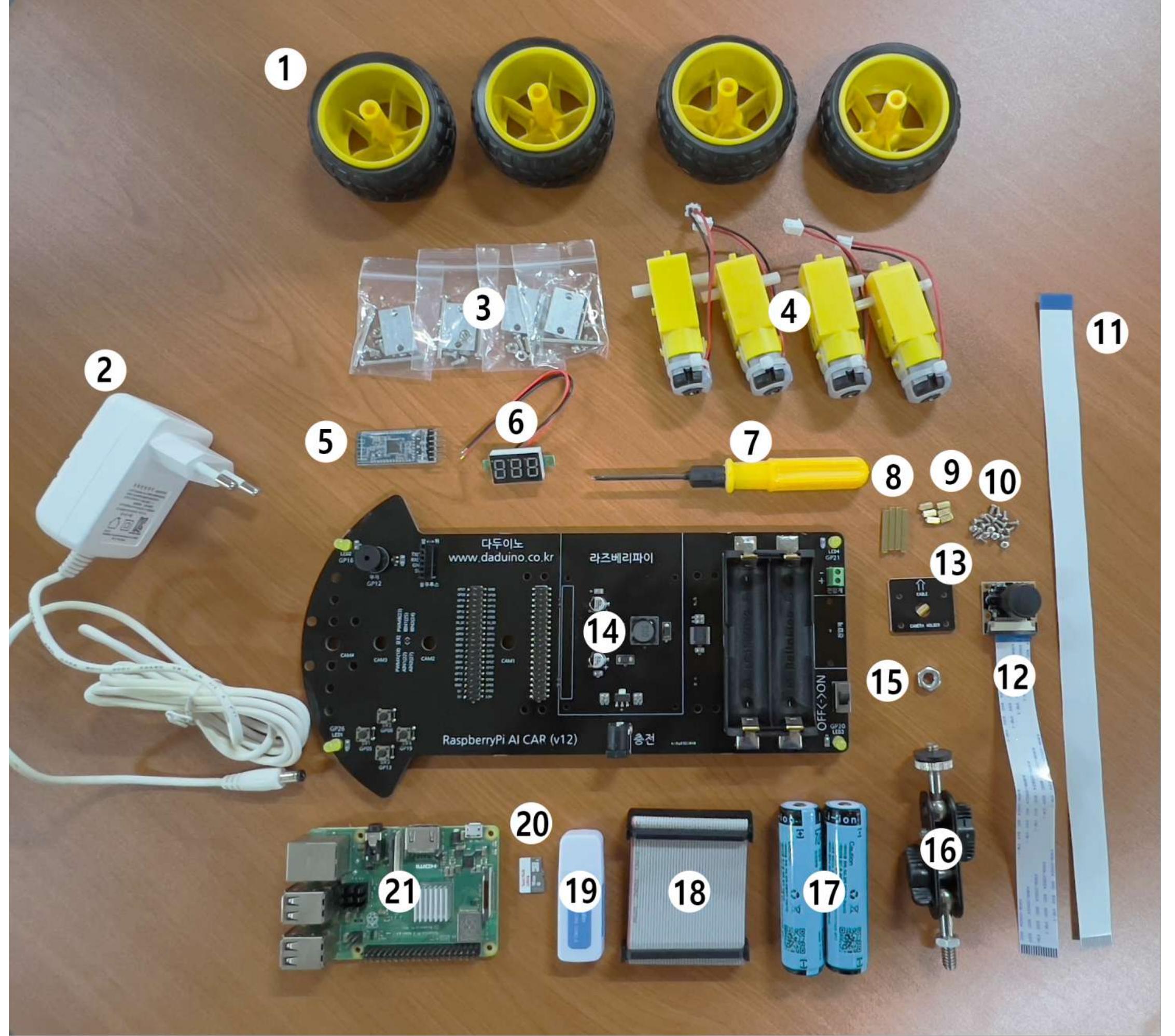




# AI 인공지능 자율주행 자동차

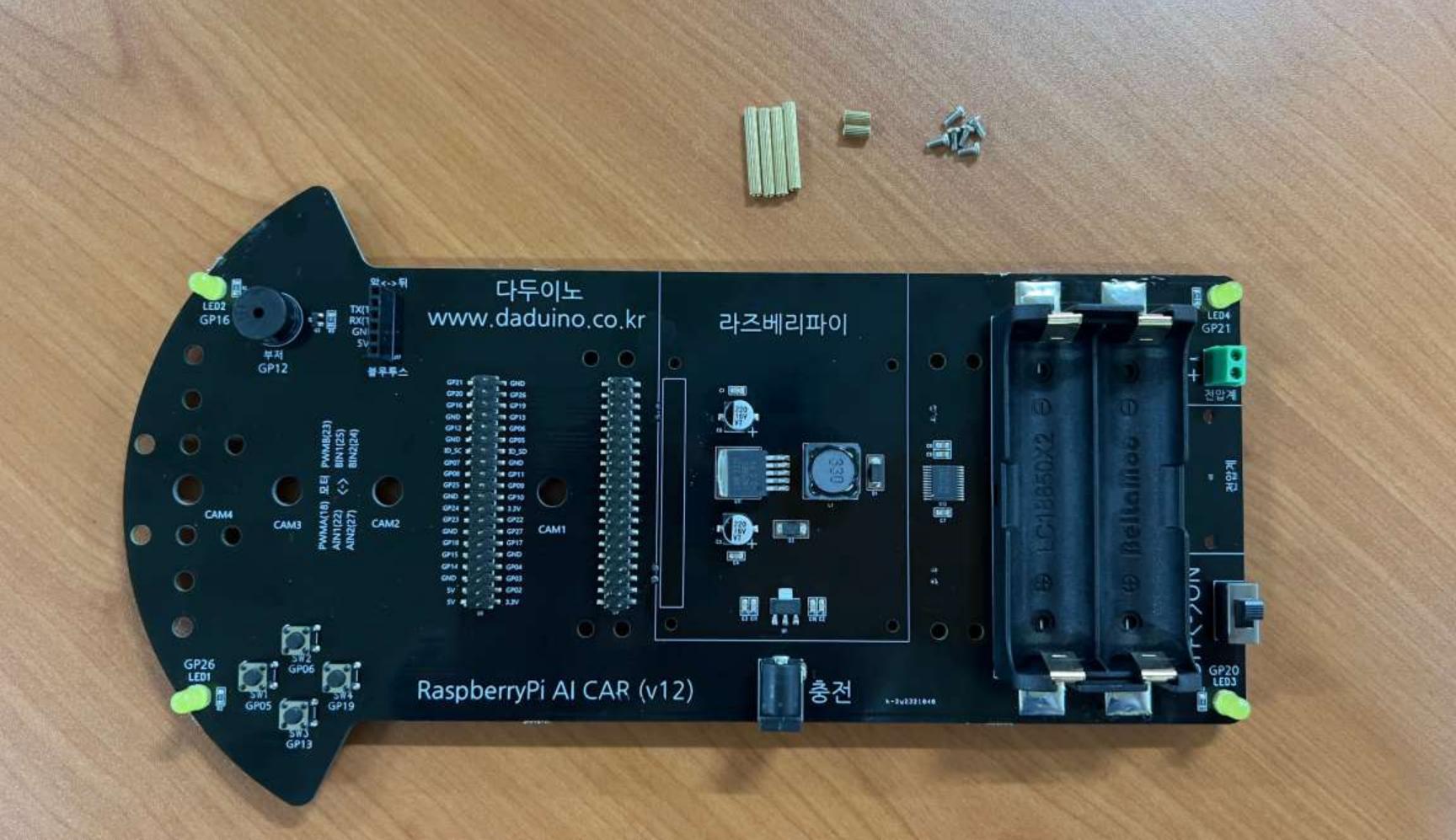
Part 1.  
만들기 with  
라즈베리파이

# 자연+인문 자동차 조립



01. 바퀴	4개
02. 충전기	1개
03. 모터 지지대 세트	4개
04. 모터 220:1	4개
05. HM-10블루투스 모듈	1개
06. 전압계	1개
07. +, - 변신드라이버	1개
08. M2 20mm 서포트	4개
09. M2 6mm 서포트	6개
10. M2 4mm 볼트	19개
11. 30cm 카메라 케이블	1개
12. 광각 카메라	1개
13. 카메라 연결보드	1개
14. 자동차 베이스보드	1개
15. 카메라 지지대용 볼트	1개
16. 카메라 지지대	1개
17. 리튬이온 배터리	2개
18. 플렛 케이블 40p 5cm	1개
19. 메모리 리더기	1개
20. SD메모리 16GB 이상 (옵션)	1개
21. 라즈베리파이 4 이상 (옵션)	1개

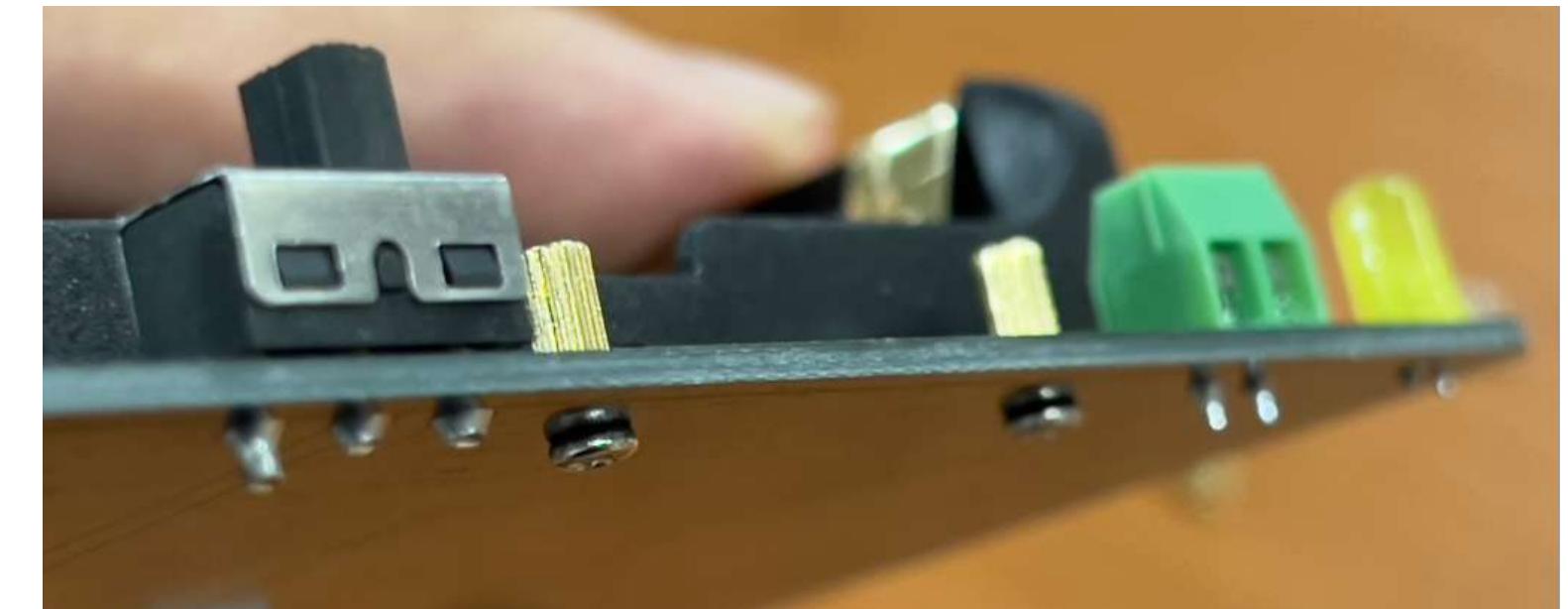
# 자동차 주행 자동화 설계



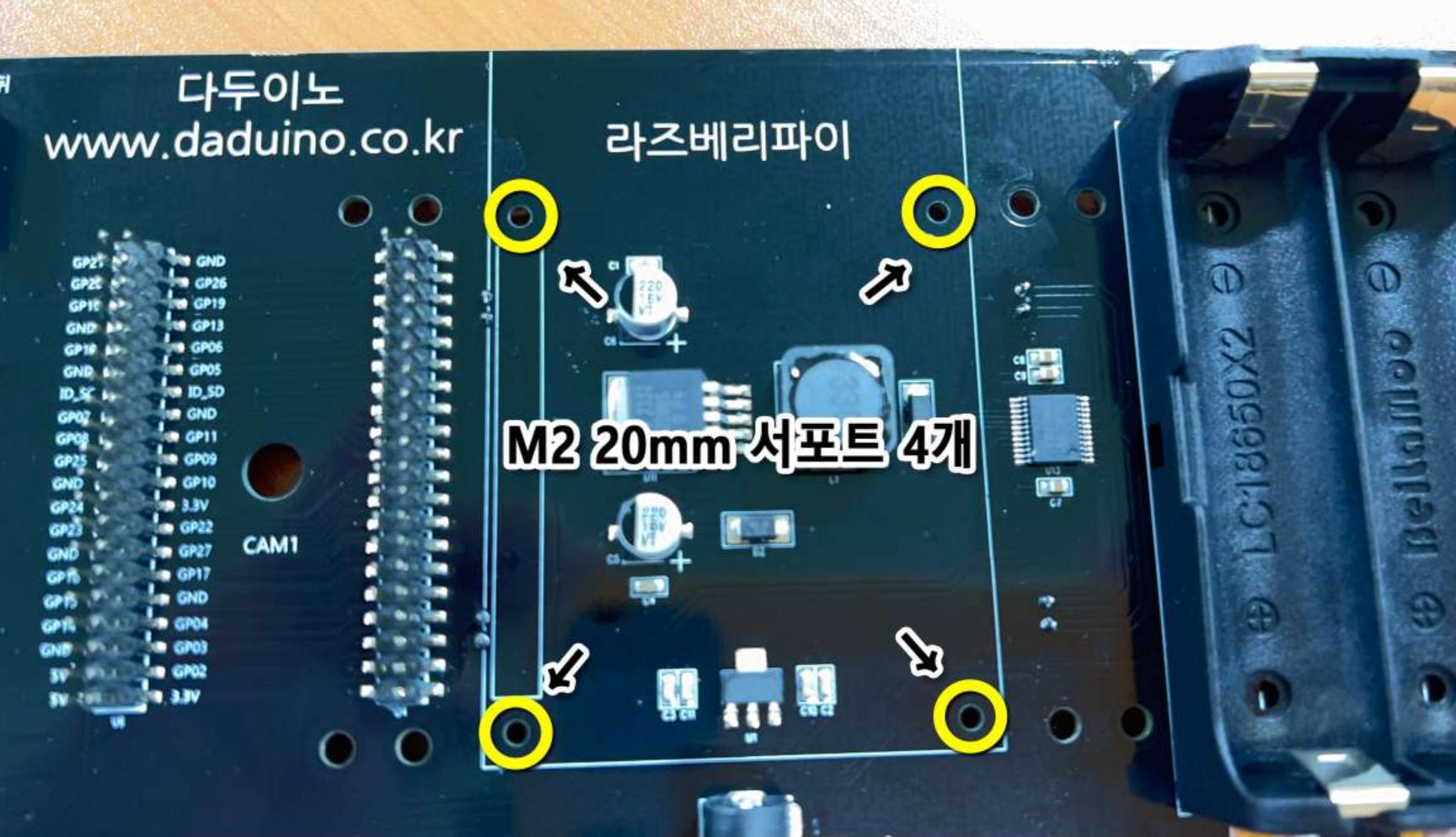
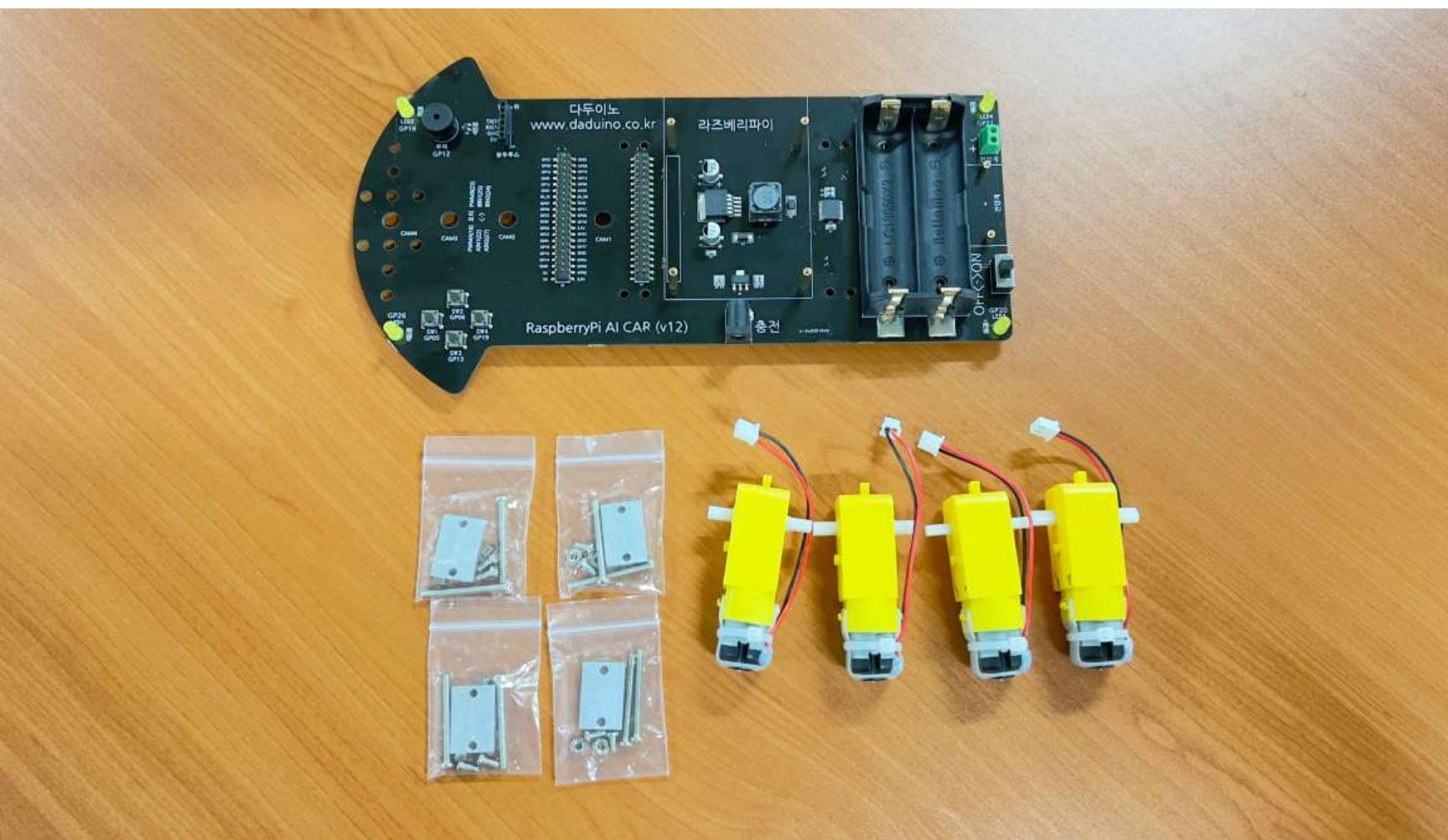
1. 자동차 베이스보드, M2 20mm 서포트 4개, M2 6mm 서포트 2개, M2 4mm 볼트 6개 전압계 부분에 M2 6mm 서포트를 조립한다.



2. M2 4m 볼트를 자동차 베이스 보드뒷면에서 위쪽으로 넣고 M2 6mm 서포트를 손으로 돌려서 두곳 모두 조립한다.



# 자연스러운 카메라 자동차 조립

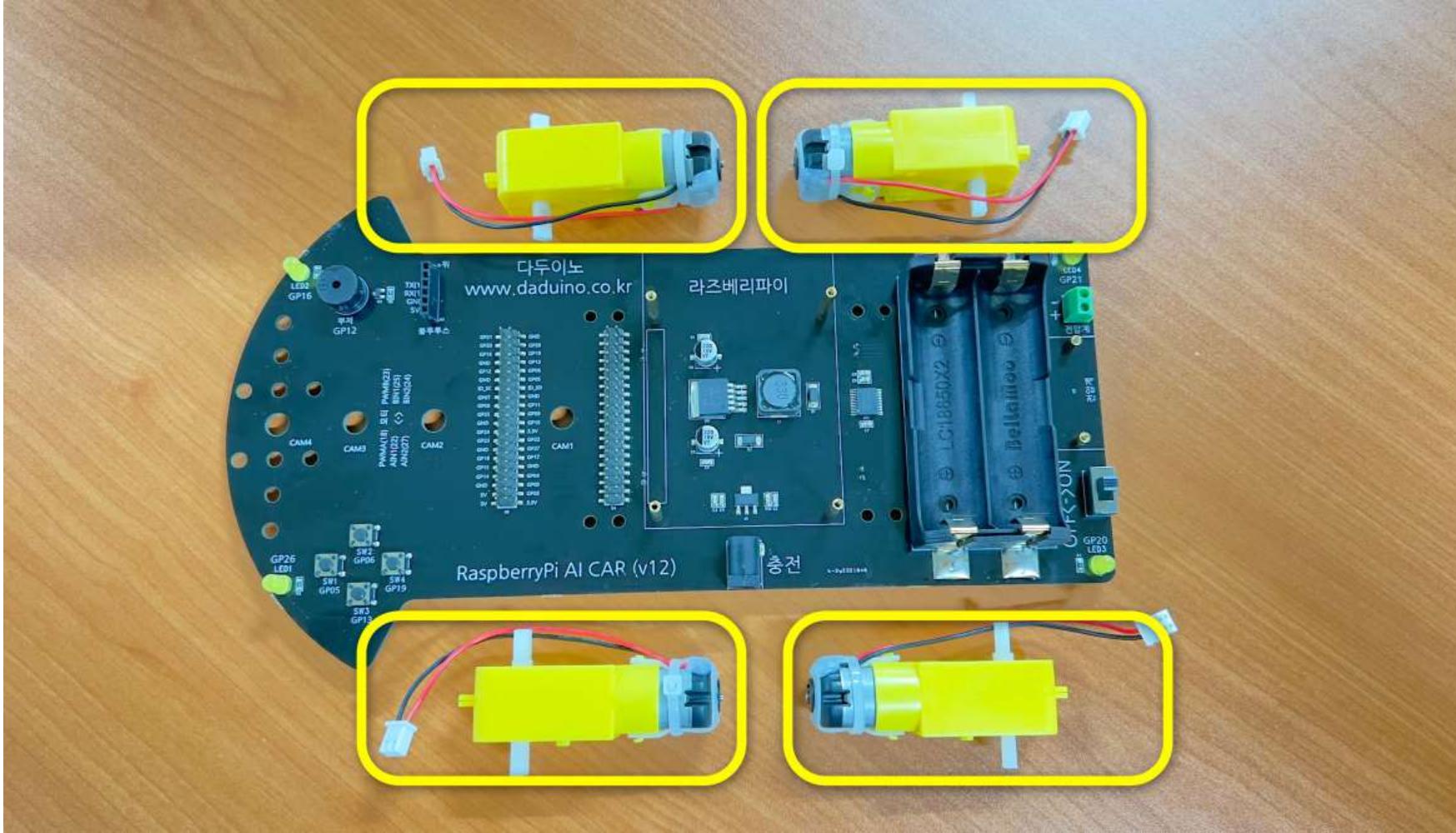
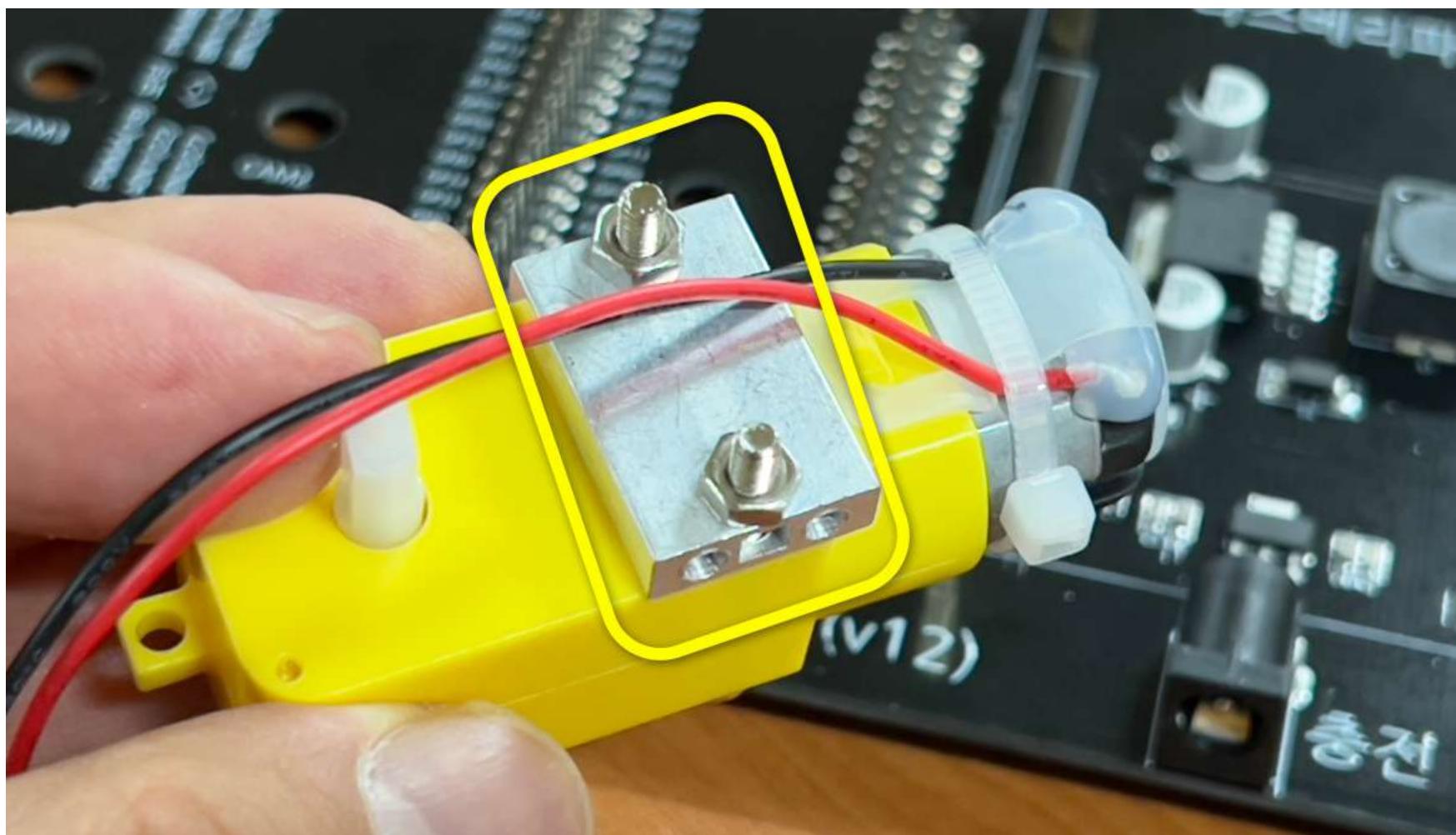


3. 라즈베리 파이 보드를 부착할 곳으로 M2 20mm 서포트 4개, M2 4mm 볼트 4개를 자동차 베이스 보드에 조립한다. 보드의 윗면에 서포트를, 베이스의 뒷면에 볼트를 넣는다.

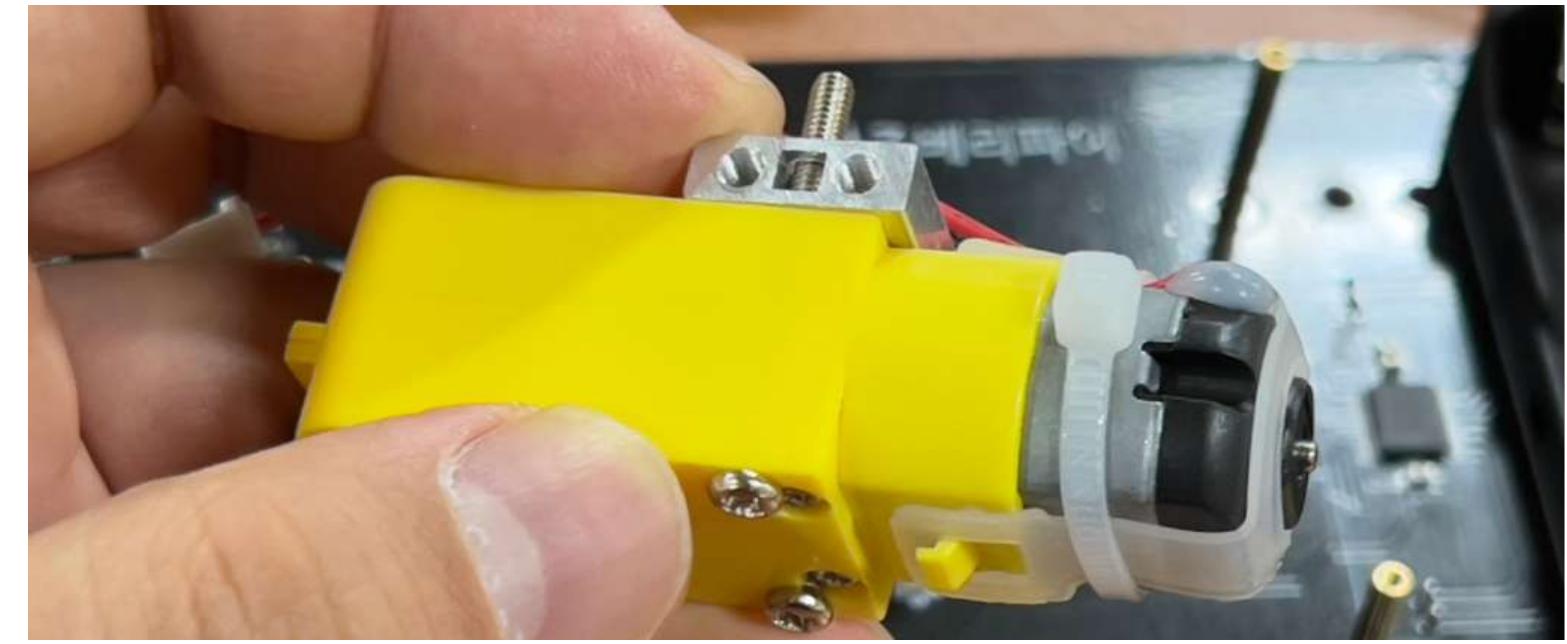


4. 모터지지대 세트 4개, 모터 220:1 4개를 준비한다.

# 자율주행 자동차 조립



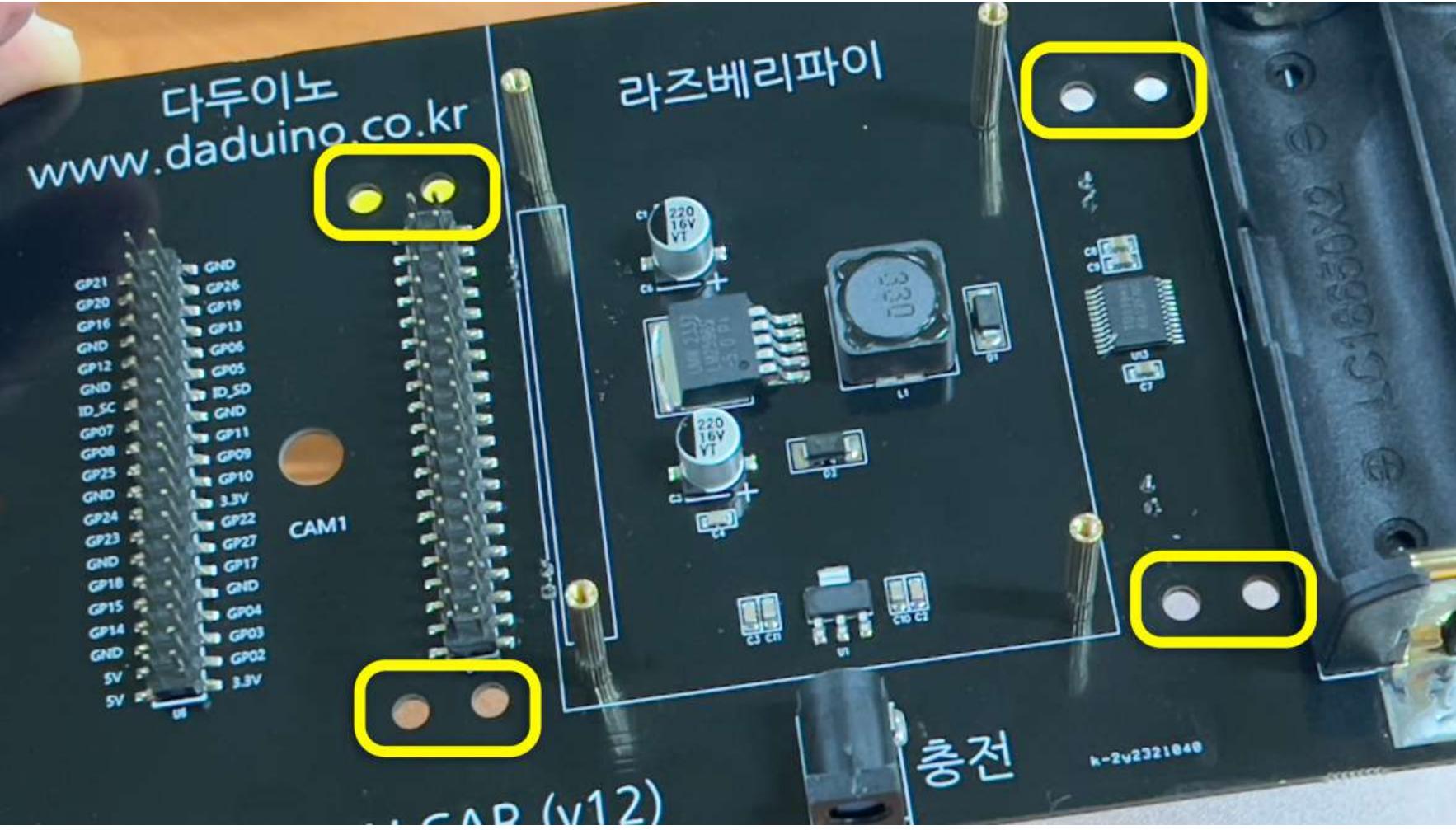
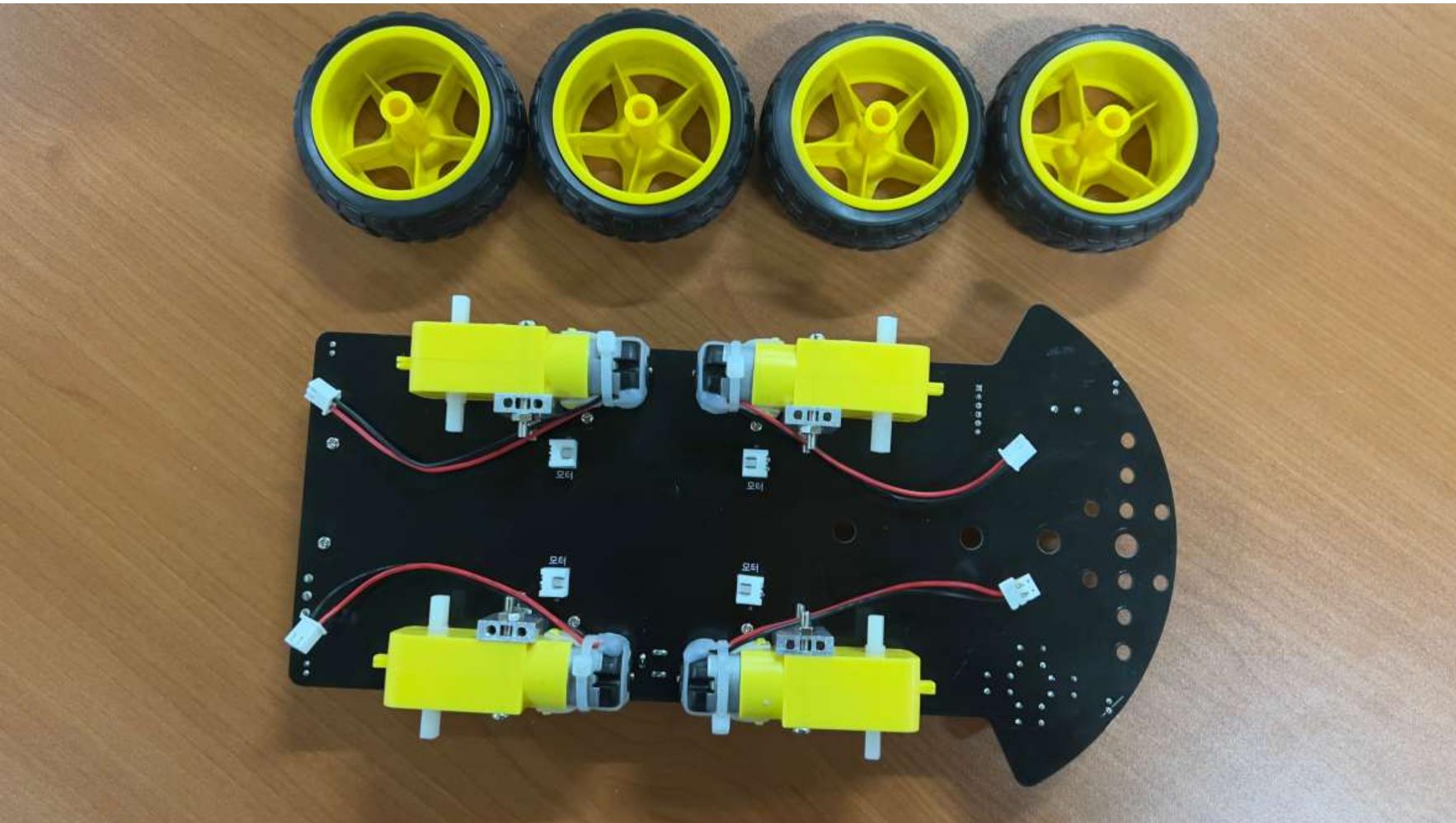
5. 모터의 선이 자동차 베이스보드 쪽으로 향하게 모터를 위치 시키고 모터와 모터지지대를 조립한다.



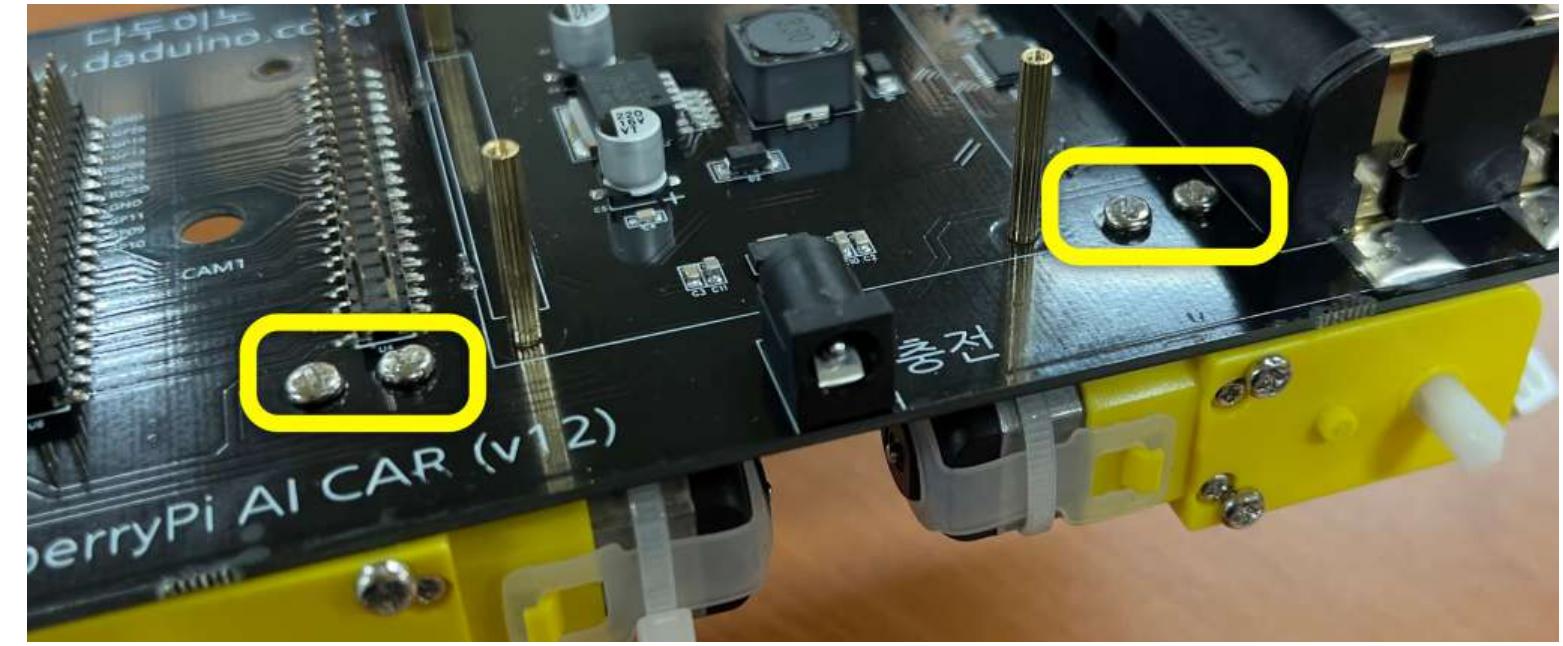
6. 모터지지대는 긴 볼트와 너트를 이용하여 조립 모터지지대와 모터의 조립 시 나사선이 있는 부분이 베이스보드에 부착되는 부분이다.



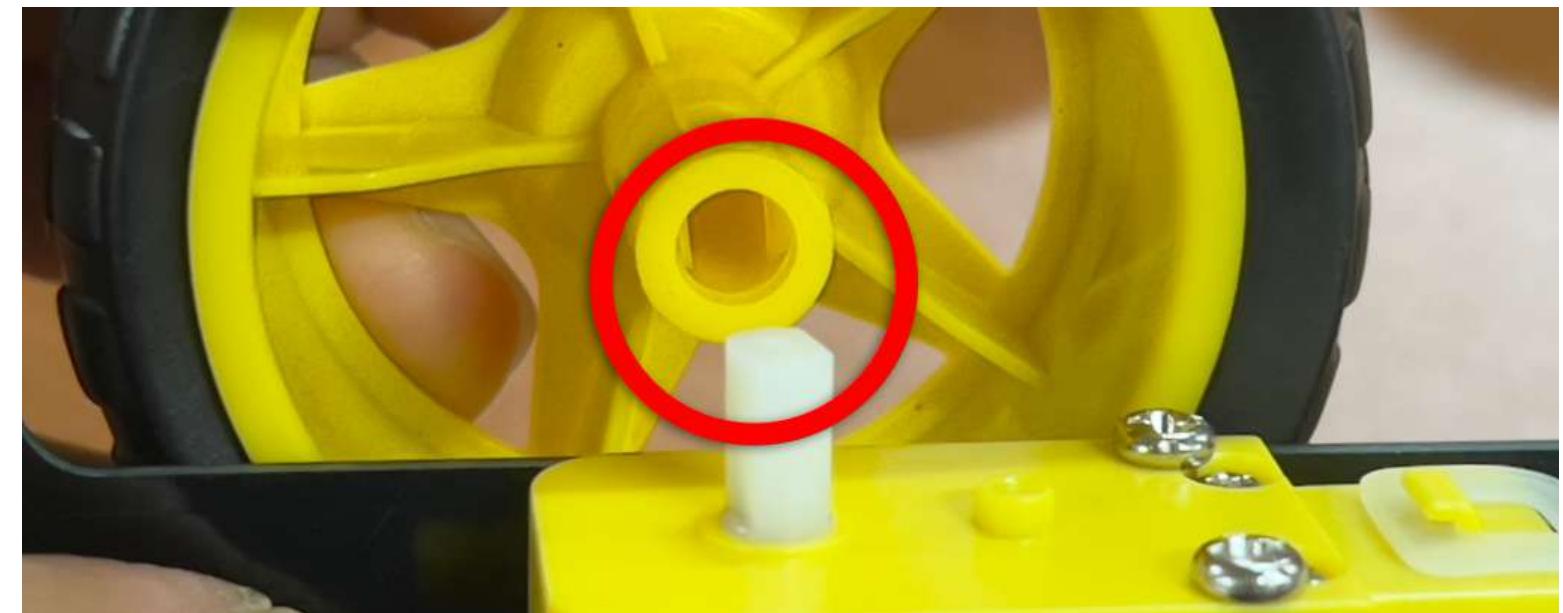
# 주제 4 로봇 자동차 제작



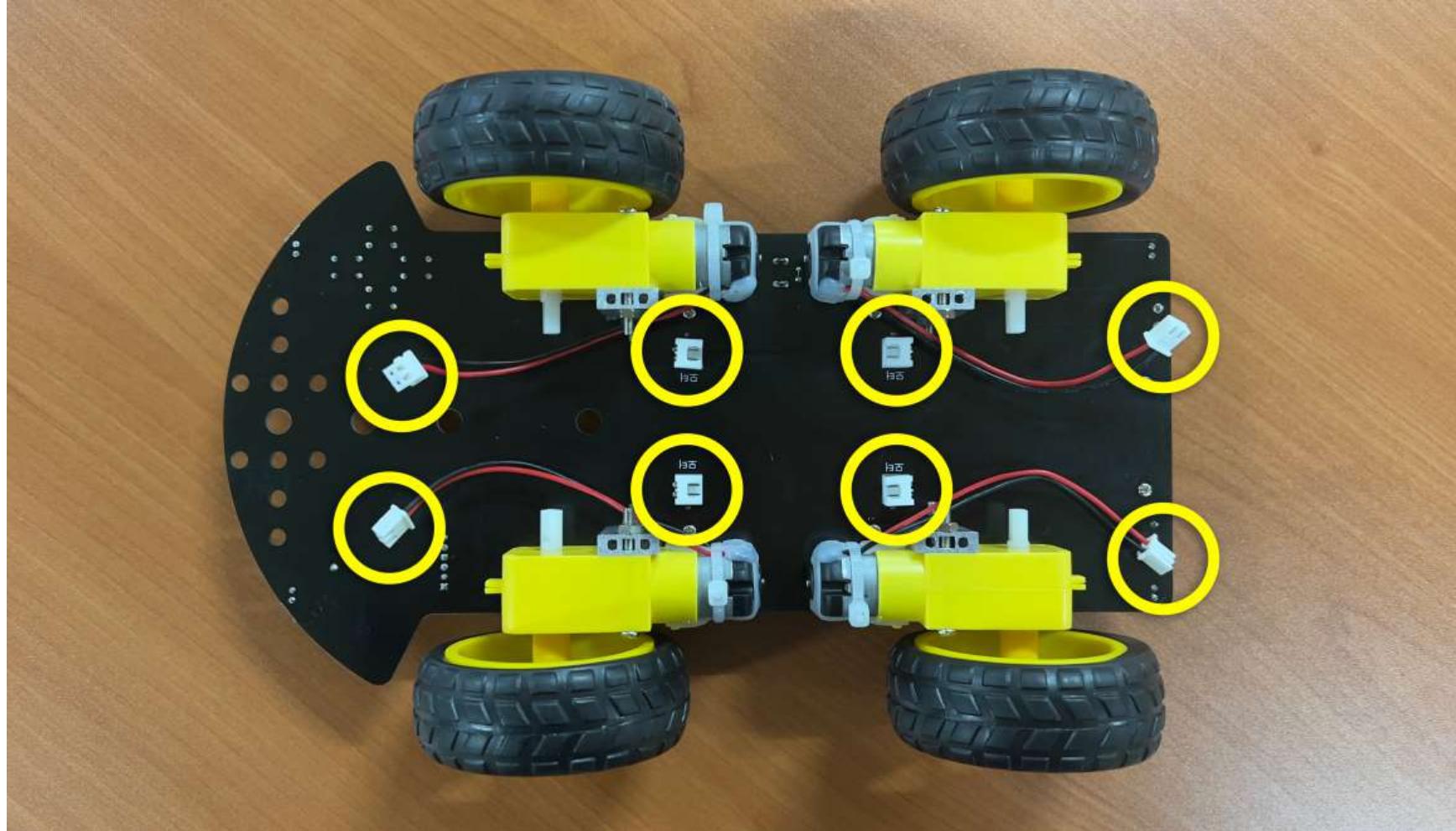
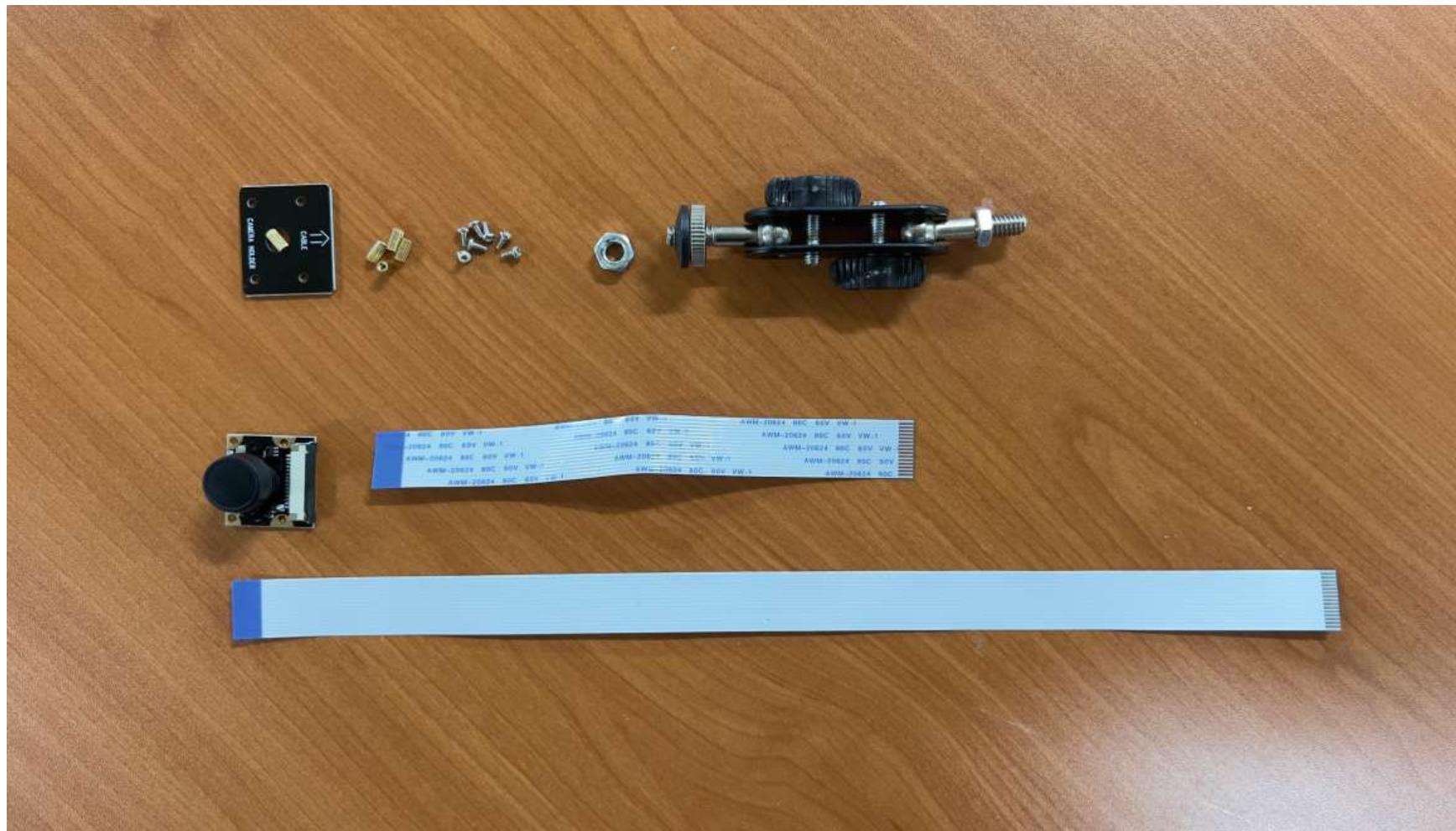
7. 모터지지대를 자동차 베이스보드 2구 구멍의 아래에 짧은 볼트를 이용하여 사진과 같이 조립한다. (선이 안쪽으로 가도록 조립)



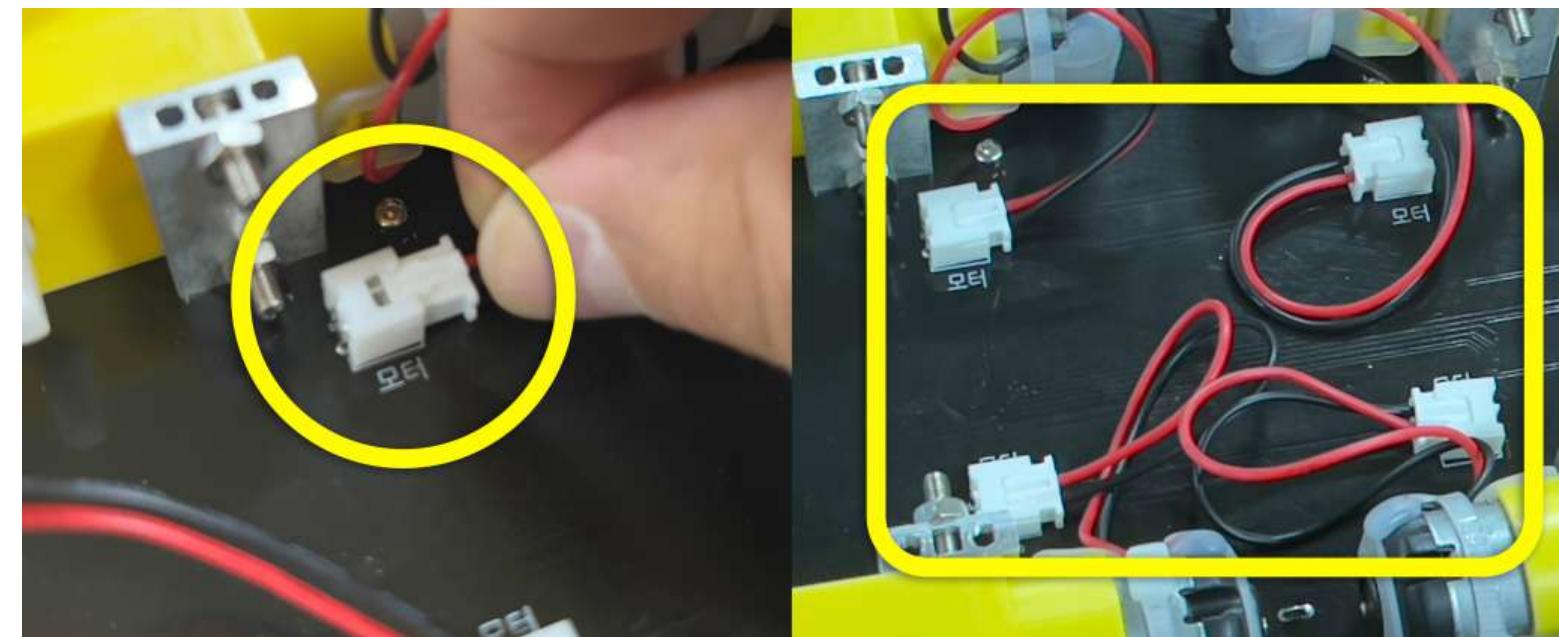
8. 바퀴 4개를 모터의 홈과 바퀴 안쪽의 홈에 맞춰 각각 조립한다.  
홈에 맞지 않는 경우 바퀴가 조립이 안된다.



# 자율주행 자동차 조립

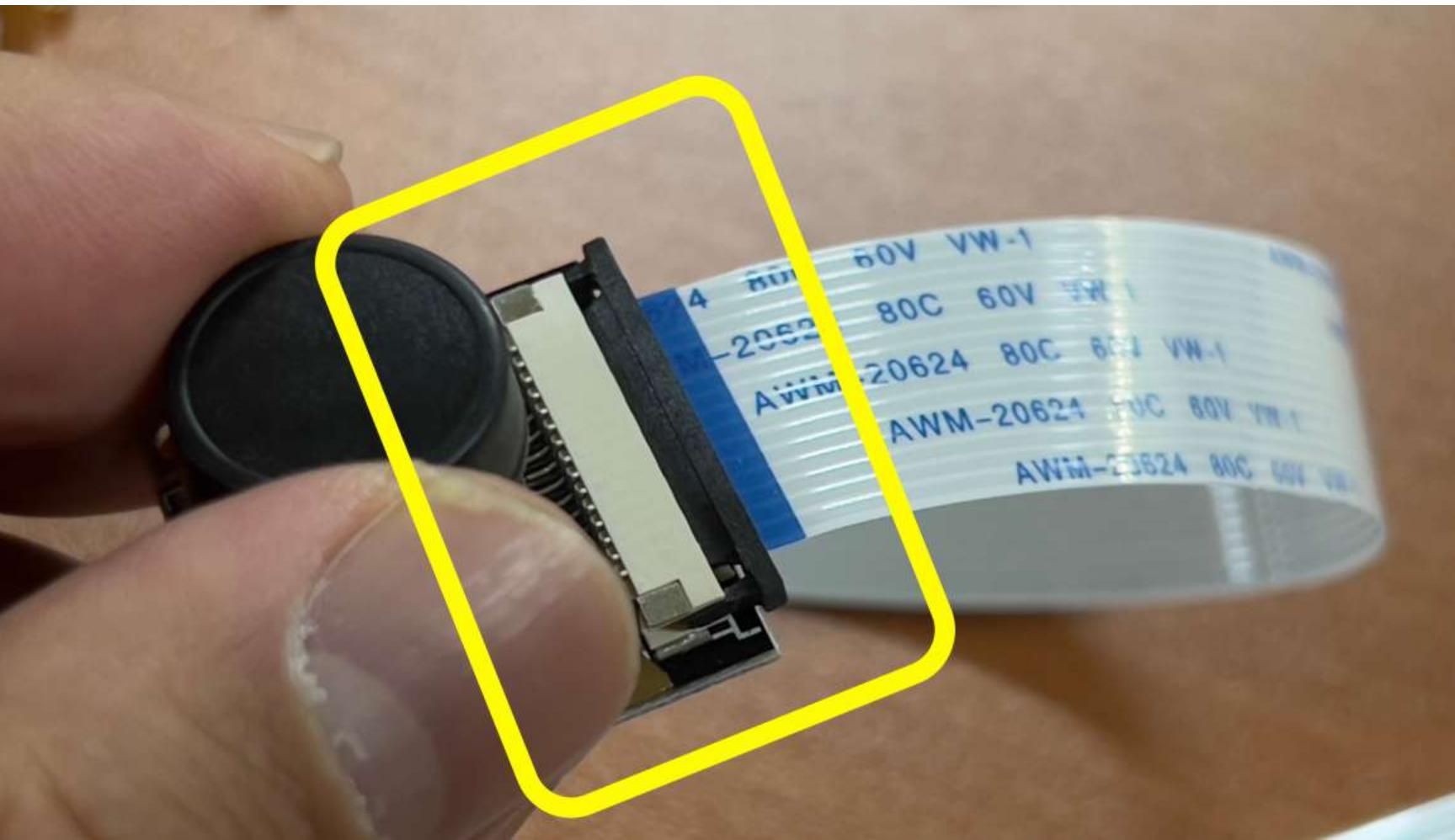


9. 모터 선을 자동차 베이스보드의 커넥터홈에  
커넥터와의 방향을 맞춰서 연결한다.



10. 카메라 연결을 위한 광각 카메라,  
카메라 연결보드, 카메라 지지대용 볼트,  
30cm 카메라 케이블, M2 6m 서포트 4개,  
M2 4mm 볼트 7개를 준비한다.  
(서포트는 4개지만 카메라 연결보드와  
카메라는 카메라 부품 손상의 위험성이 있어  
볼트를 3개만 고정하기 때문에 볼트의 수는  
7개만 준비한다)

# 차량용 카메라 설치 방법

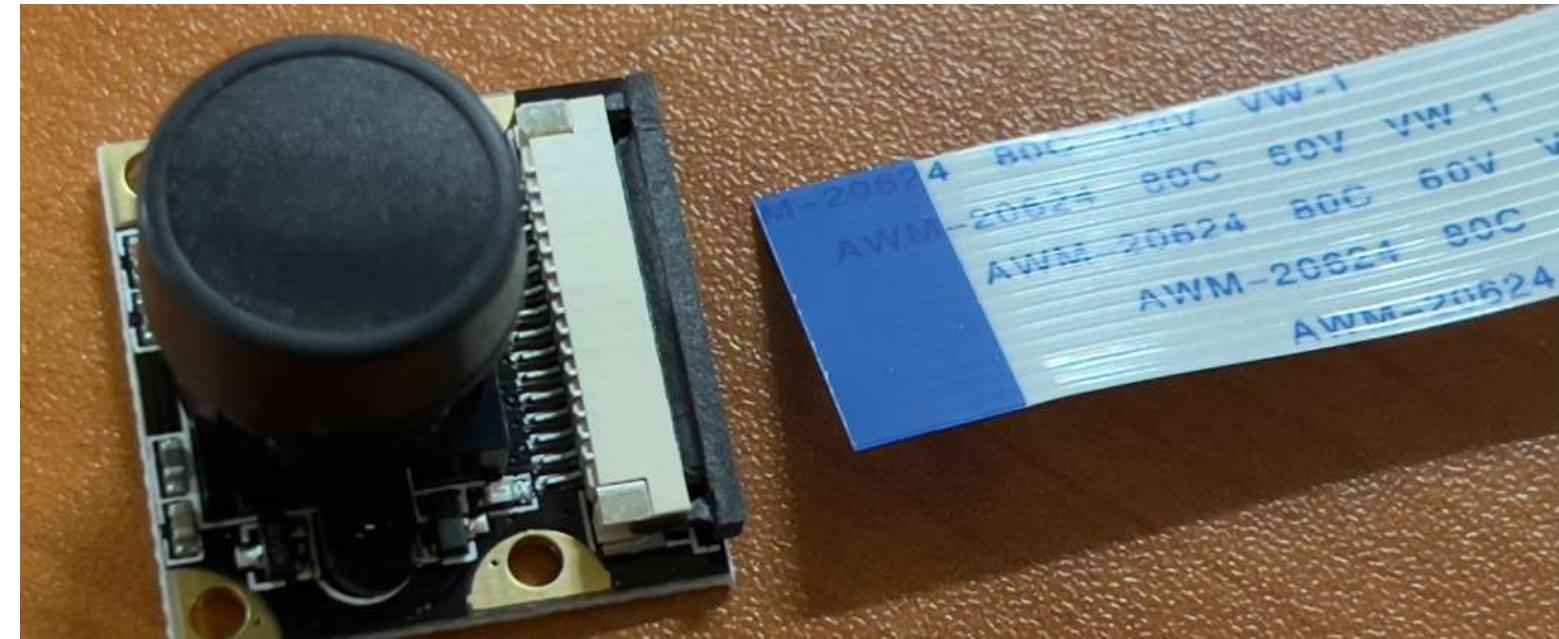


커넥터 부분을  
밖으로 밀어서  
빼어낸 후  
필름케이블을  
분리한다.

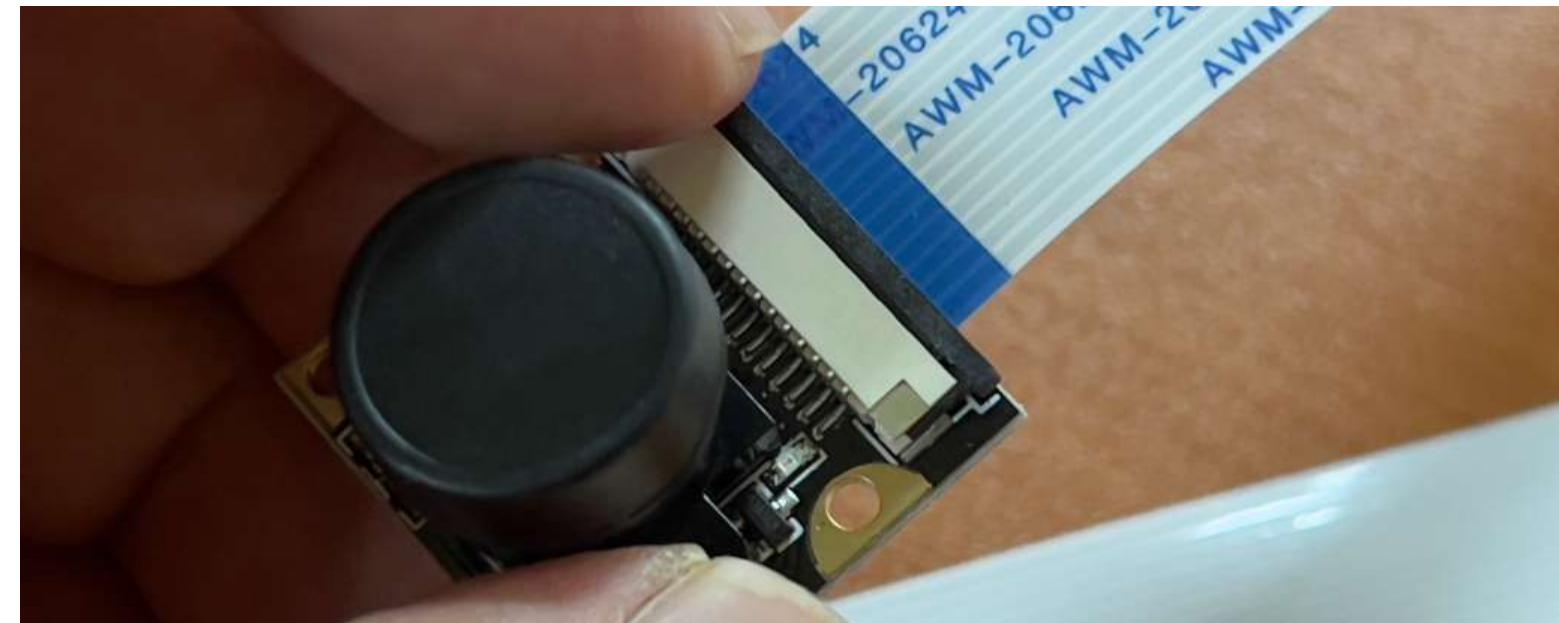


커넥터 부분을  
밖으로 밀어서  
빼어낸 후  
필름케이블을  
분리한다.

11. 카메라 모듈에서 커넥터 부분의 날개부분을 양쪽 손톱으로 잡고 필름케이블 방향으로 살짝 당겨서 빼어낸 후 필름케이블을 분리



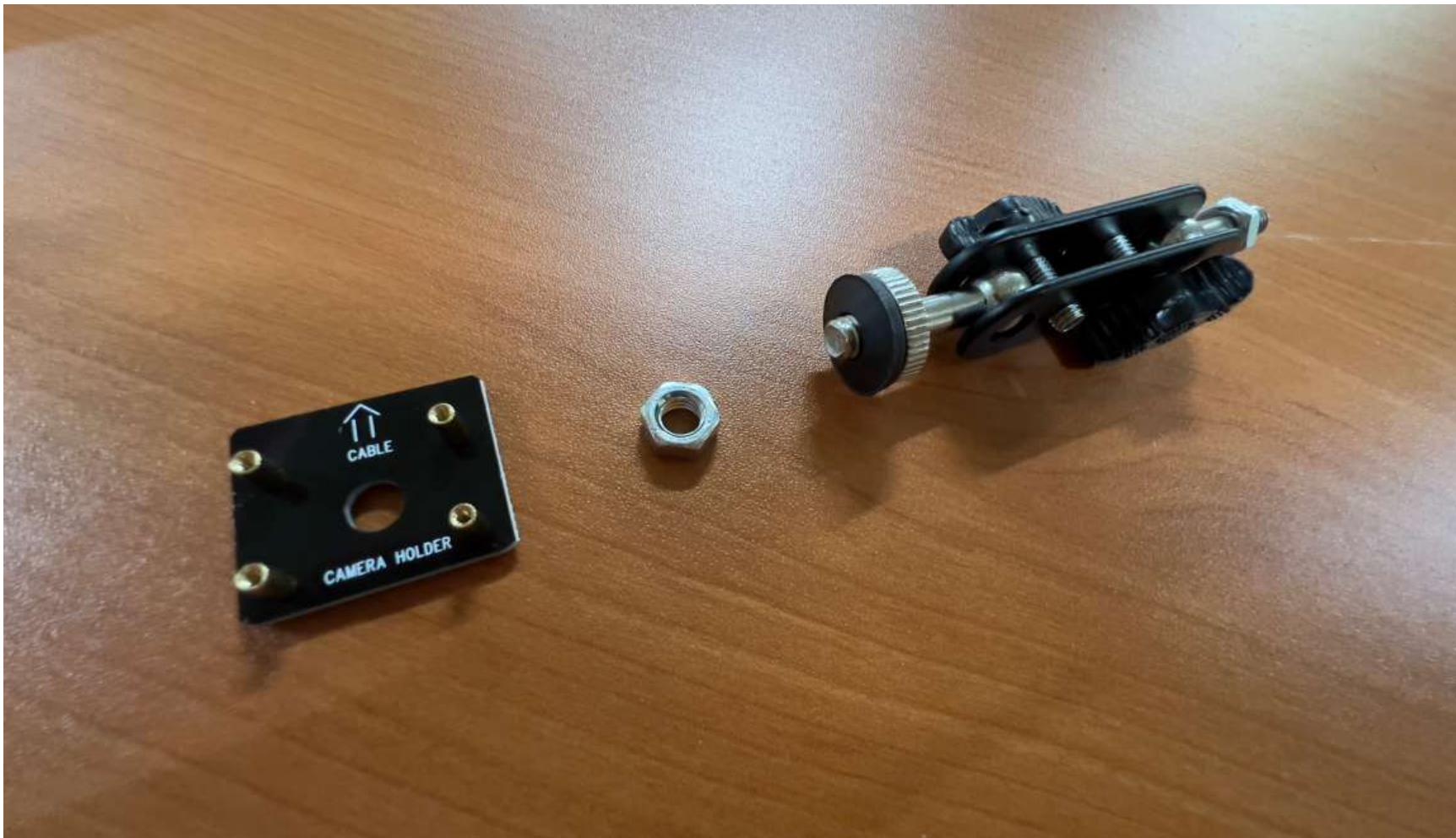
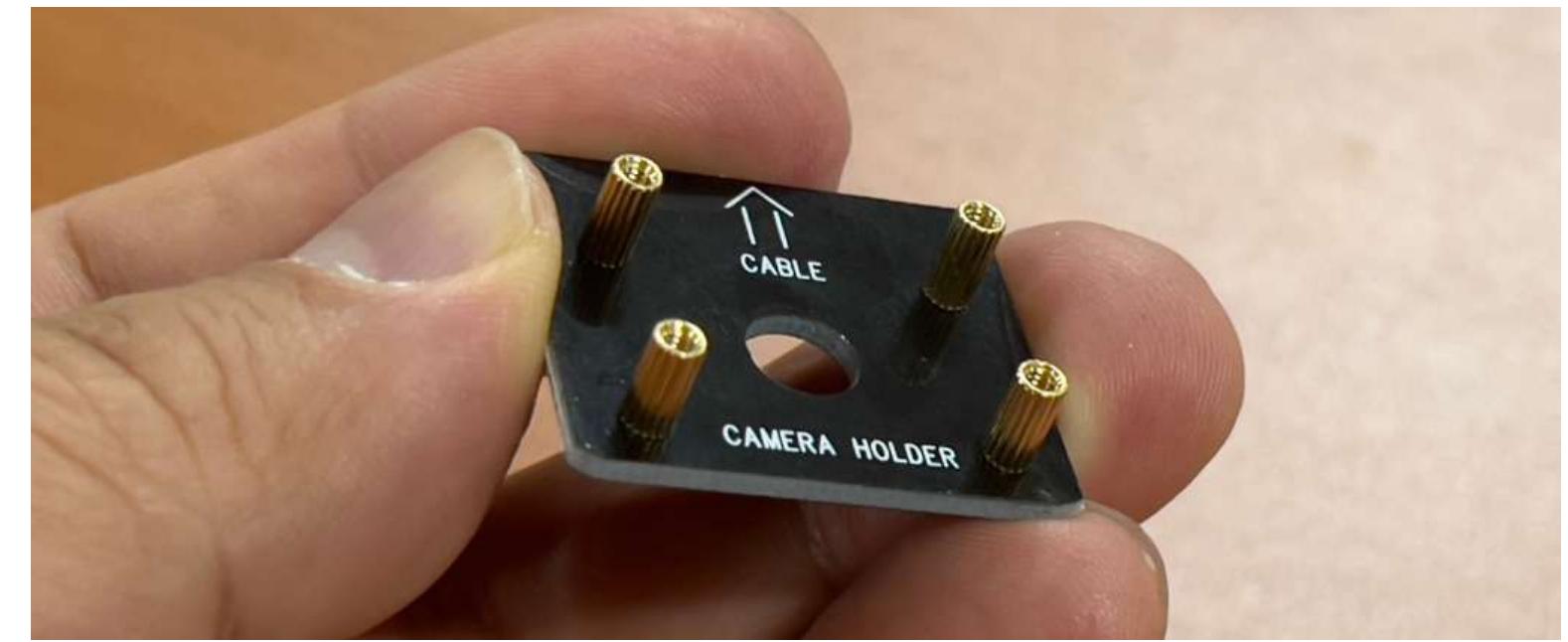
12. 기존 필름케이블 분리 후 30cm 카메라 케이블의 파란색 라인 부분이 위로 올라오게 케이블을 밀어넣고 커넥터핀을 밀어 넣는다.



# 자동차 주행 장치 설정



13. 카메라 연결보드와 M2 6mm 서포트 4개, M2 4mm 볼트 4개를 이용하여 연결보드를 조립한다.



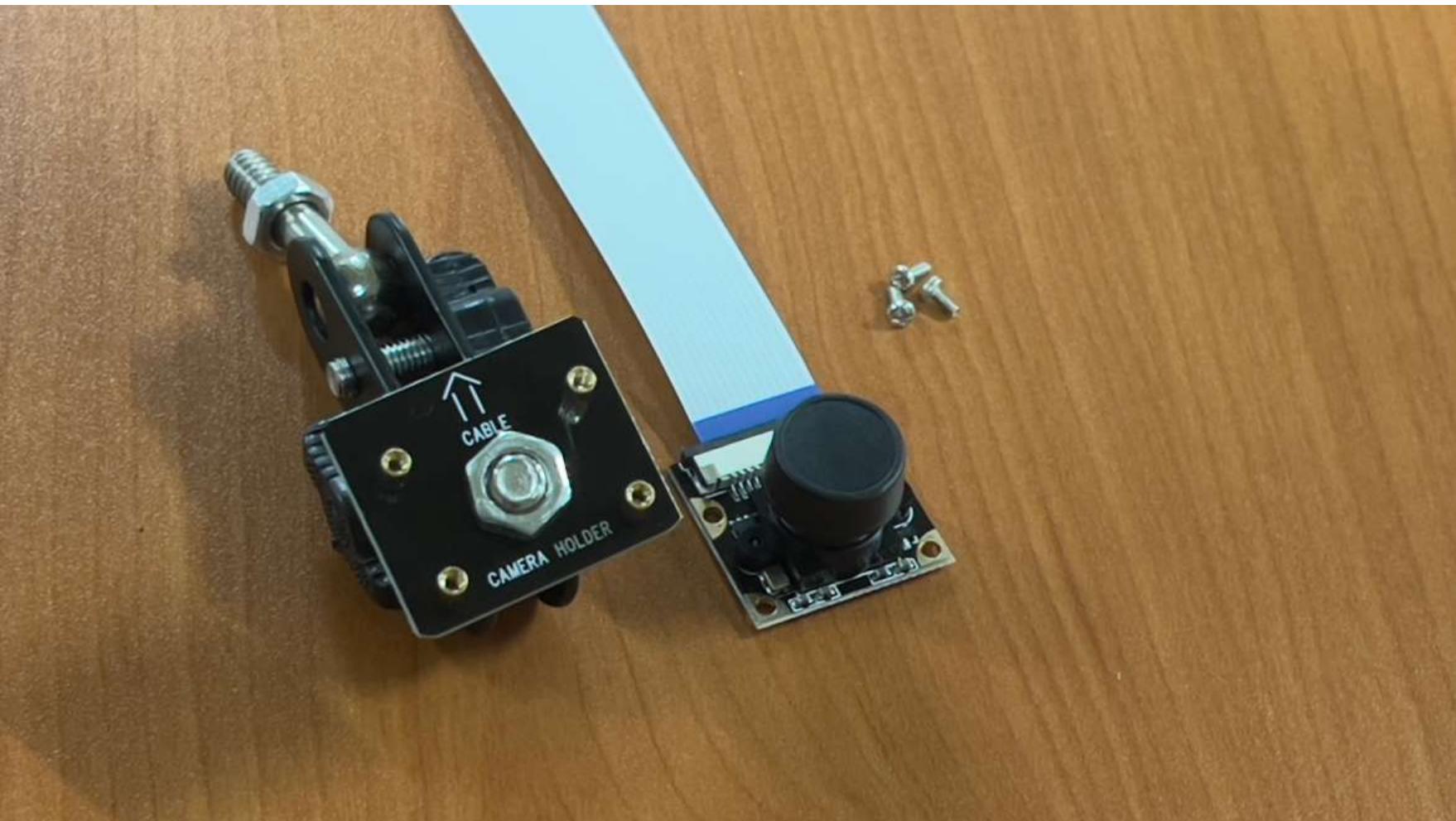
14. 카메라 연결보드, 카메라 지지대 및 볼트를 이용하여 카메라 연결보드를 지지대에서 흔들리지 않도록 단단하게 결합한다.



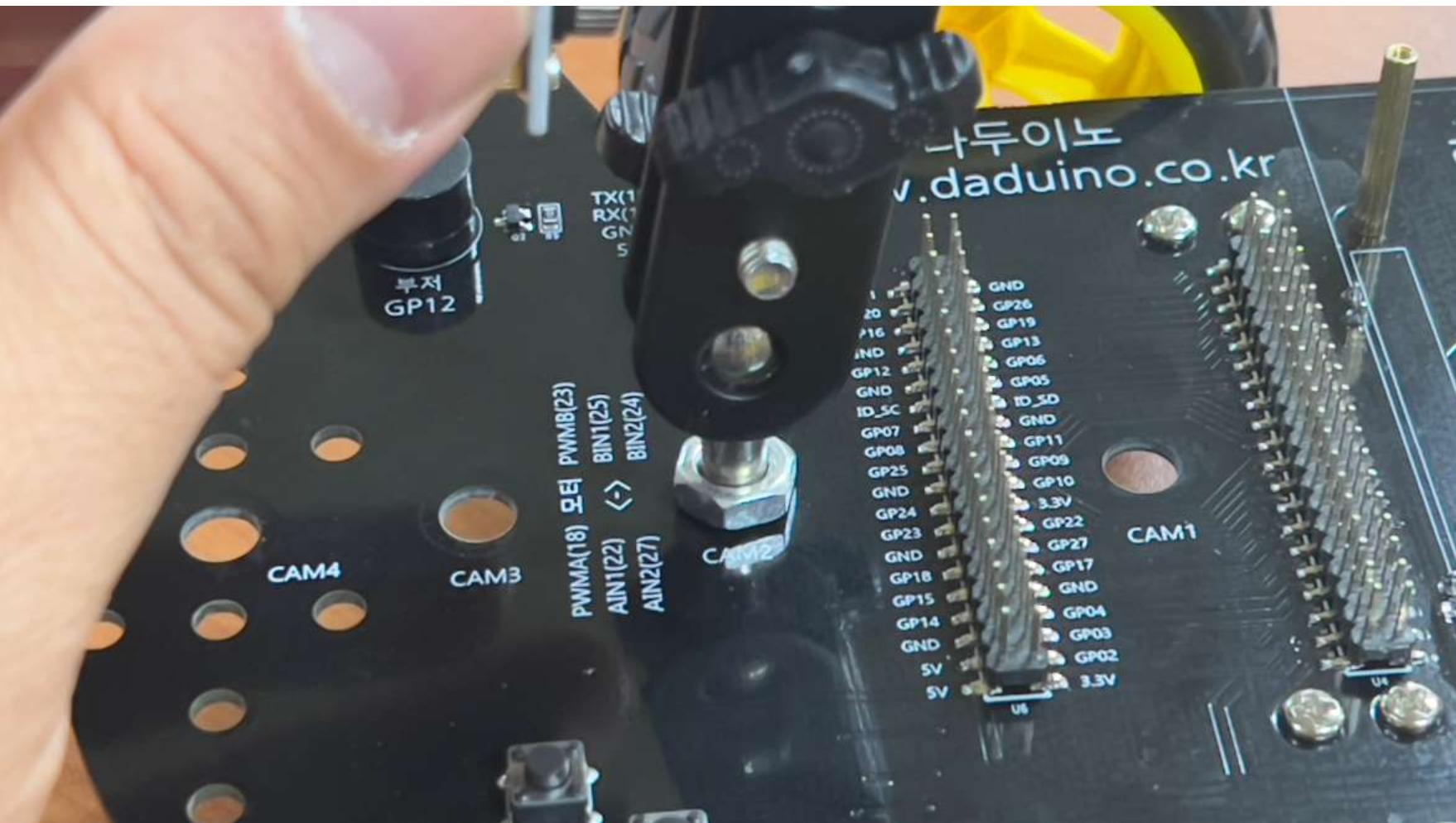
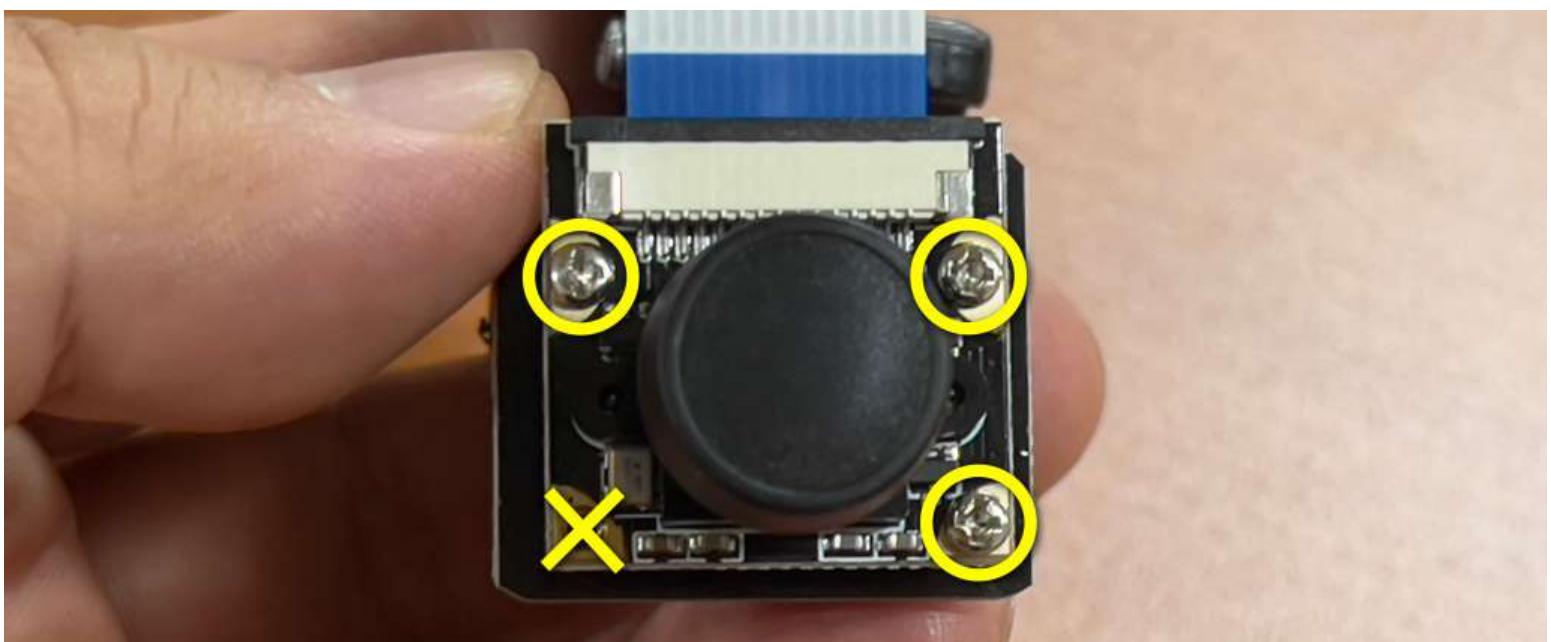
볼트를 돌려 결합

볼트를 돌려 단단히 고정

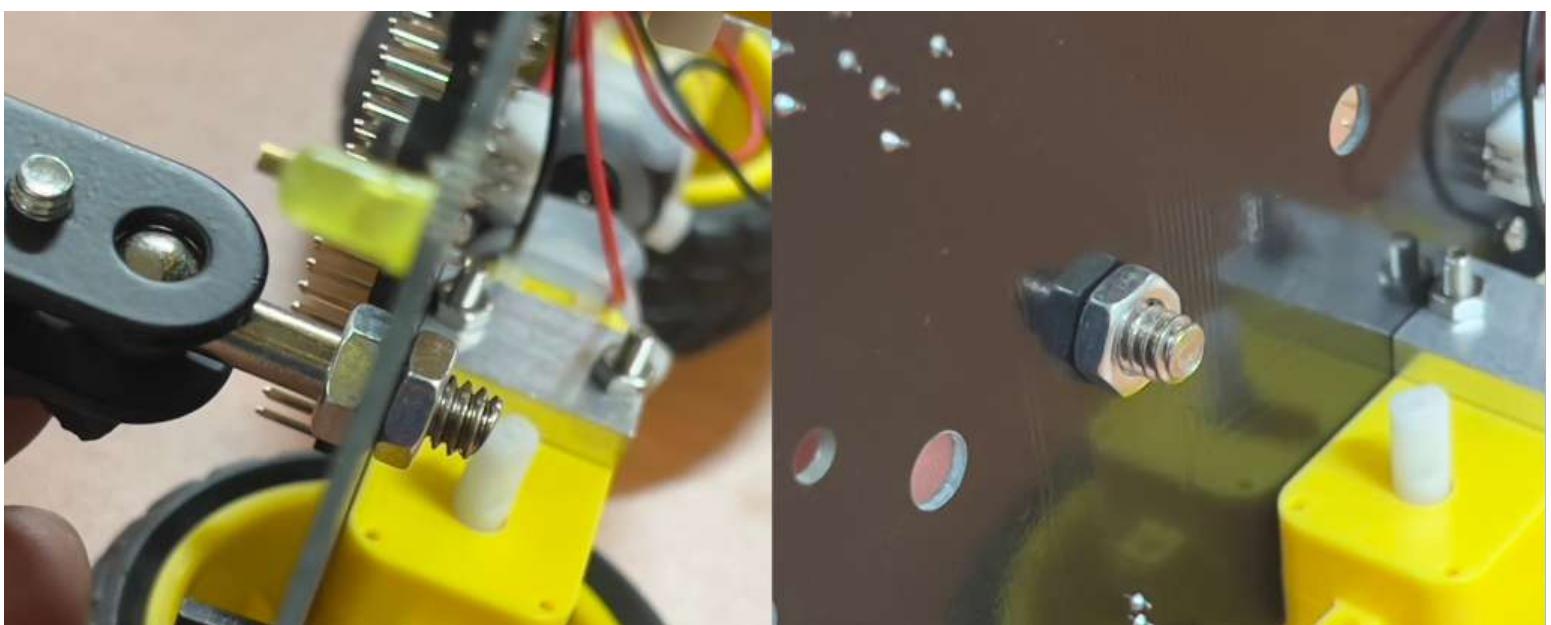
자연을  
존중하는  
환경  
생태  
친화  
도시



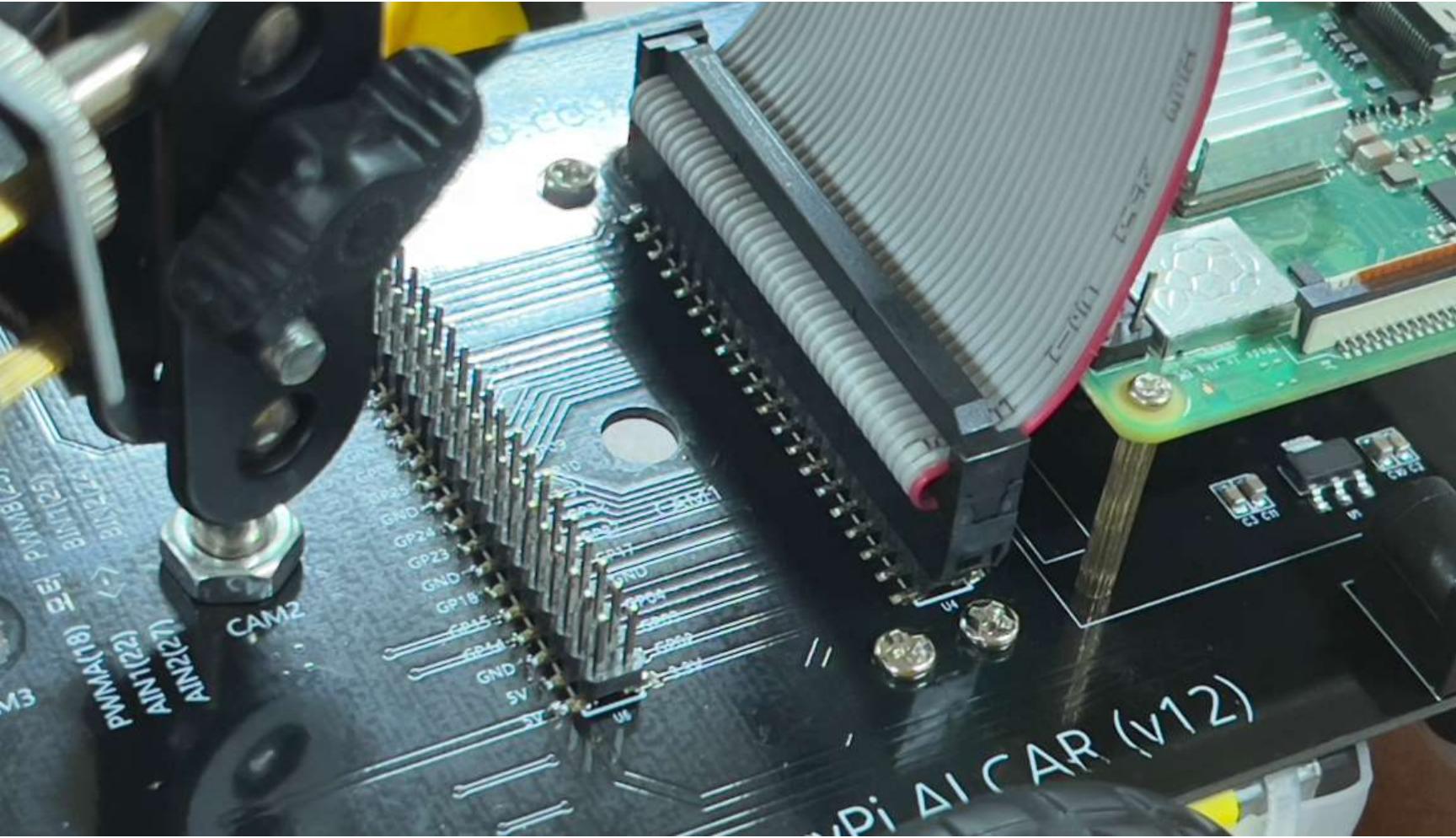
15. 카메라 모듈과 카메라 지지대를 M2 4mm  
볼트 3개를 이용하여 고정하는데 아래의  
사진에 표시된 3군데만 볼트로 고정한다.



16. 카메라 지지대를 자동차보드의 CAM2 홀에 맞춰서 지지대 고정 볼트를 이용하여 조립한 후 고정한다.



# 자동차 주행 자동화



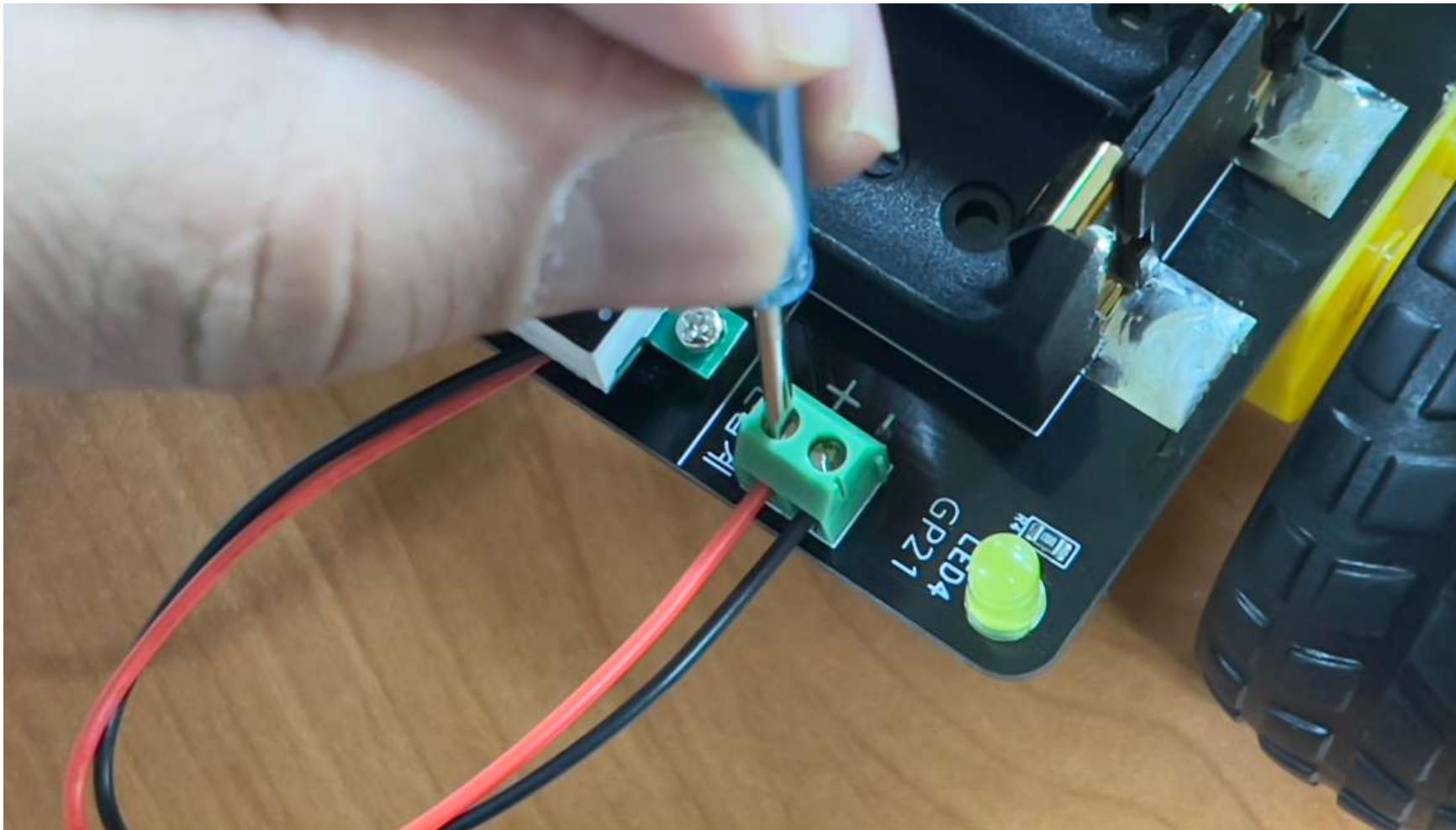
17. 라즈베리파이를 M2 4mm 볼트 4개를 이용  
자동차보드에 조립하여 고정한다.



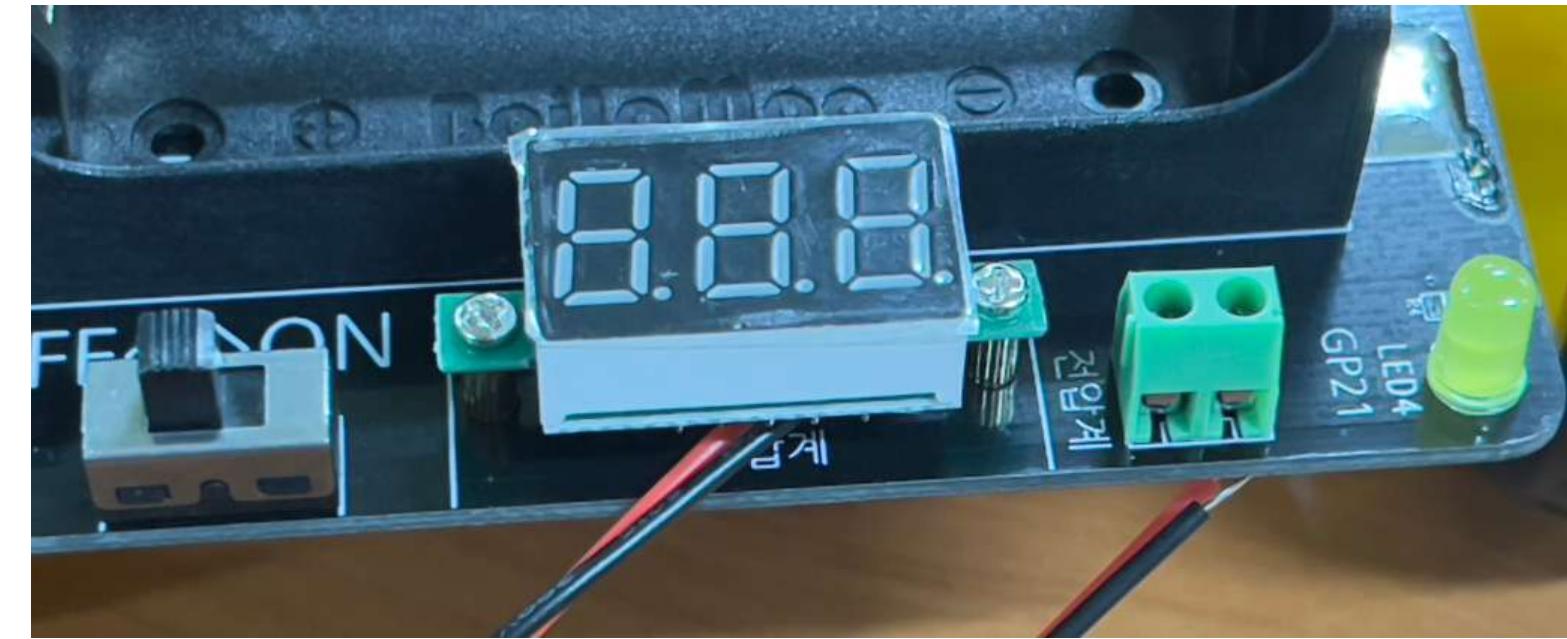
18. 플랫케이블 40p 5cm 를 라즈베리파이와  
자동차 베이스보드에 사진과 같이 핀 방향을  
맞춰 연결한다.



# 자연수 출력 자동차 조립

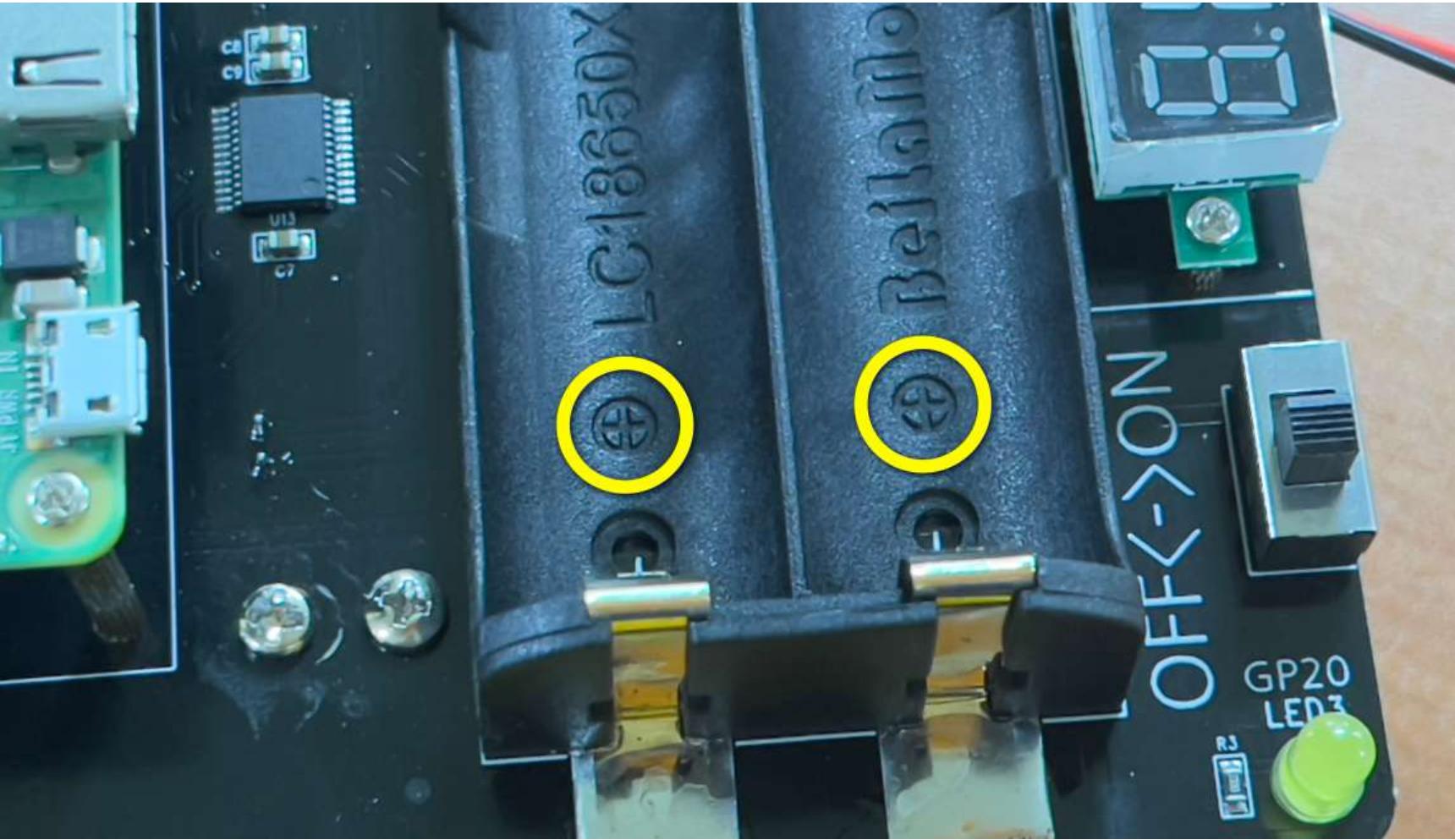


19. 전압계의 점이 있는 부분이 아래쪽이 되게 M2 4mm 볼트 2개를 이용하여 자동차 베이스보드의 전압계 자리에 조립한다.

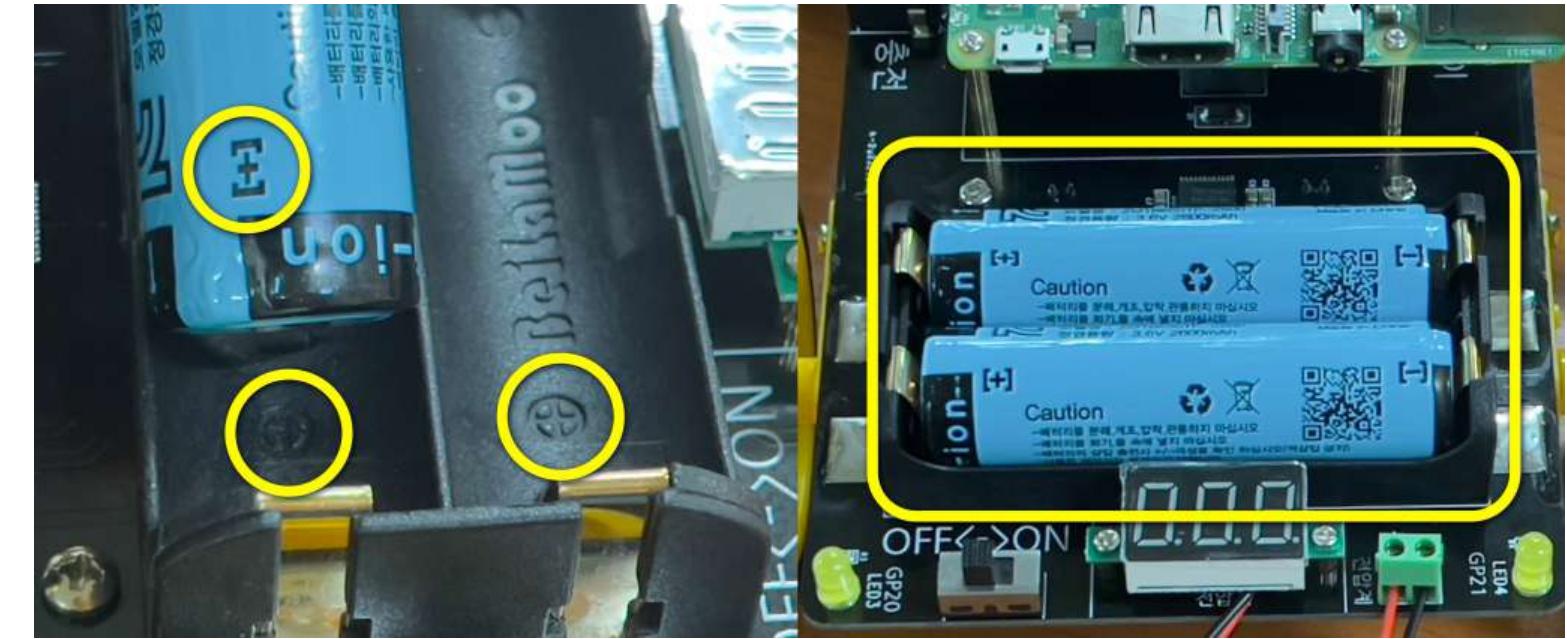


20. 컫트트에 들어 있는 작은 드라이브를 이용하여 자동차 베이스보드의 전압계 단자를 돌려 커넥터에 전압계선이 들어갈 수 있게 한다. 전압계의 선은 빨간색을 + 단자에, 검정색은 - 단자에 꽂아주고 드라이브를 이용하여 커넥터의 단자를 돌려 선이 빠지지 않도록 조립한다.

# 자율주행 자동차 조립



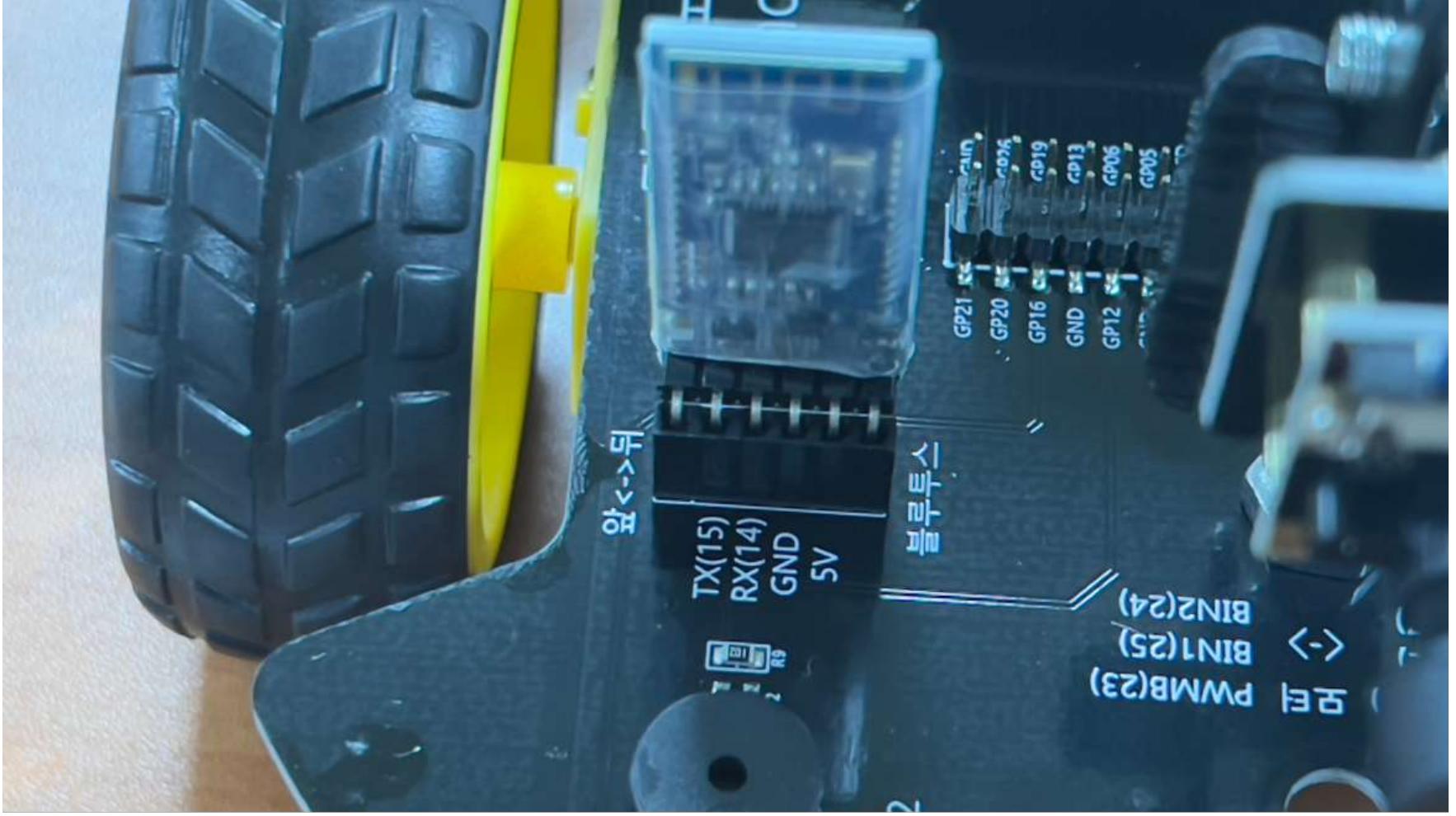
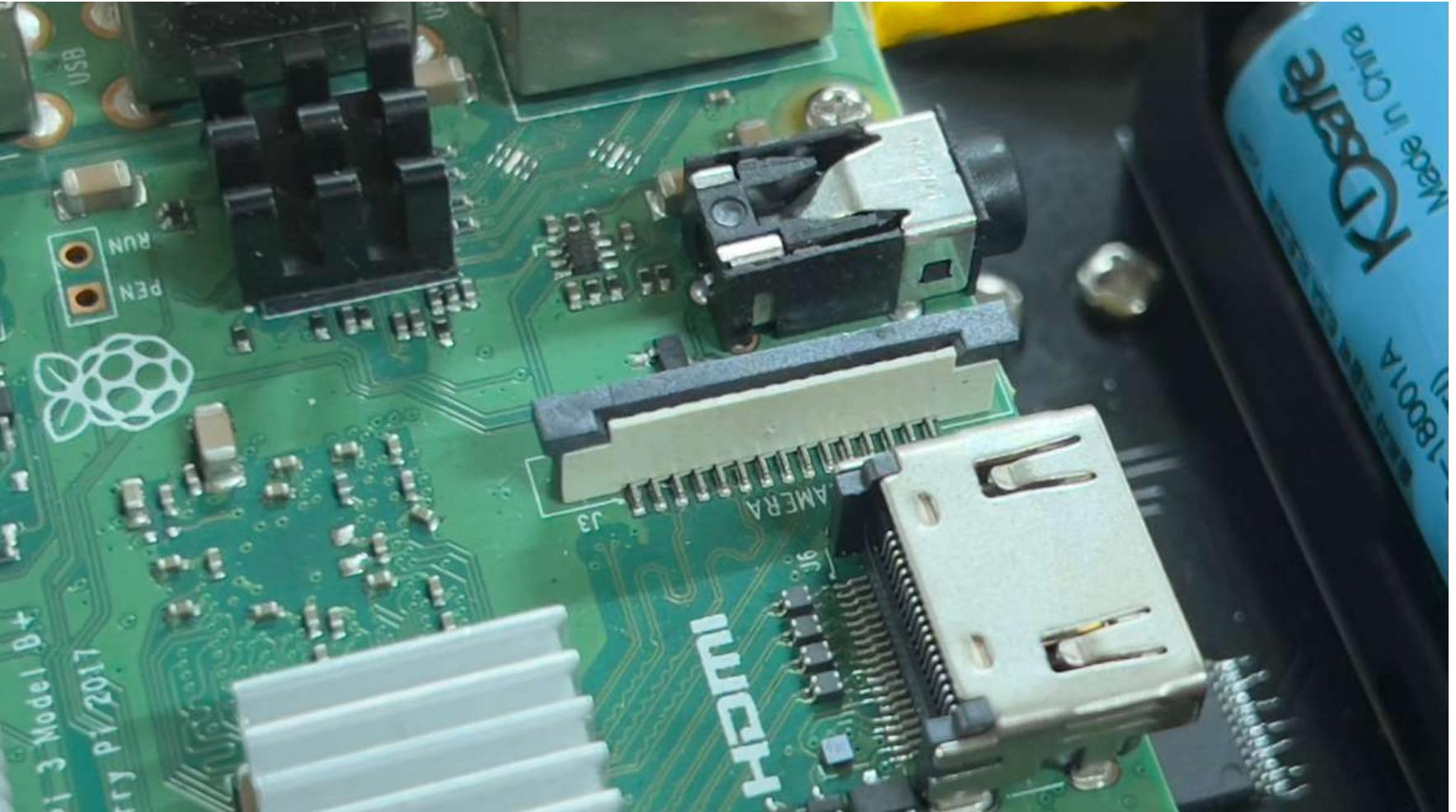
21. 배터리홀더의 극성을 확인한 후 극성에 맞게 리튬이온 배터리의 +, - 를 맞춰서 홀더에 넣는다. (틀릴 경우 부품이 탈 수 있음)



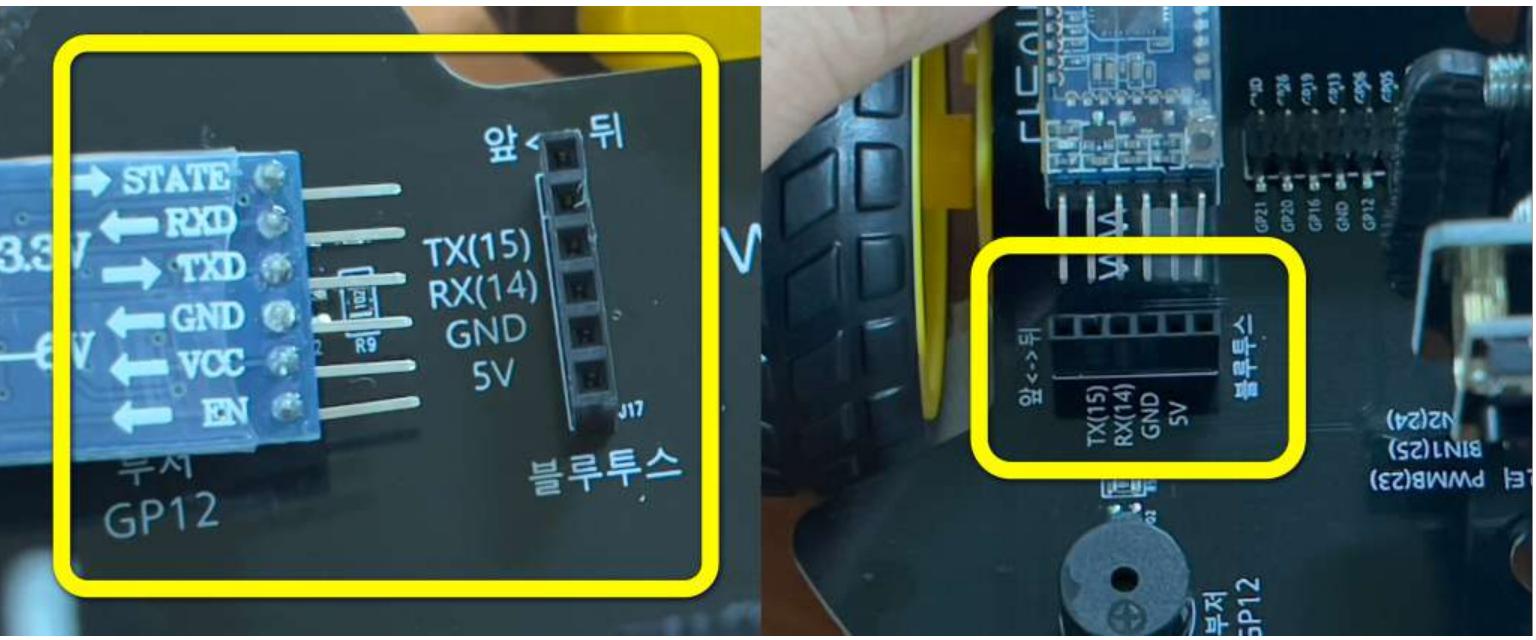
22. 전원 스위치를 ON으로 하여 전압계를 확인 전압이 7.1 ~ 8.4v인지 확인한다. 전압이 7.1v 이하인 경우 충전기를 연결하여 충전



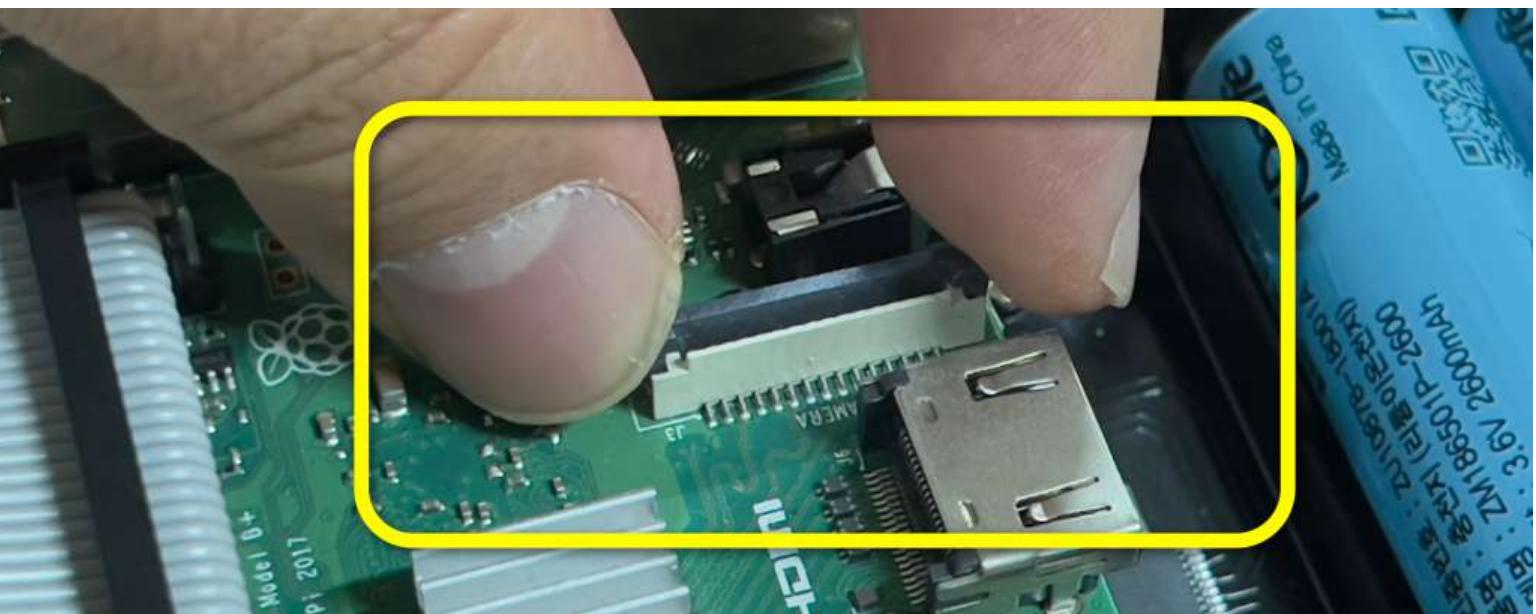
# 자동주행 자동차 조립



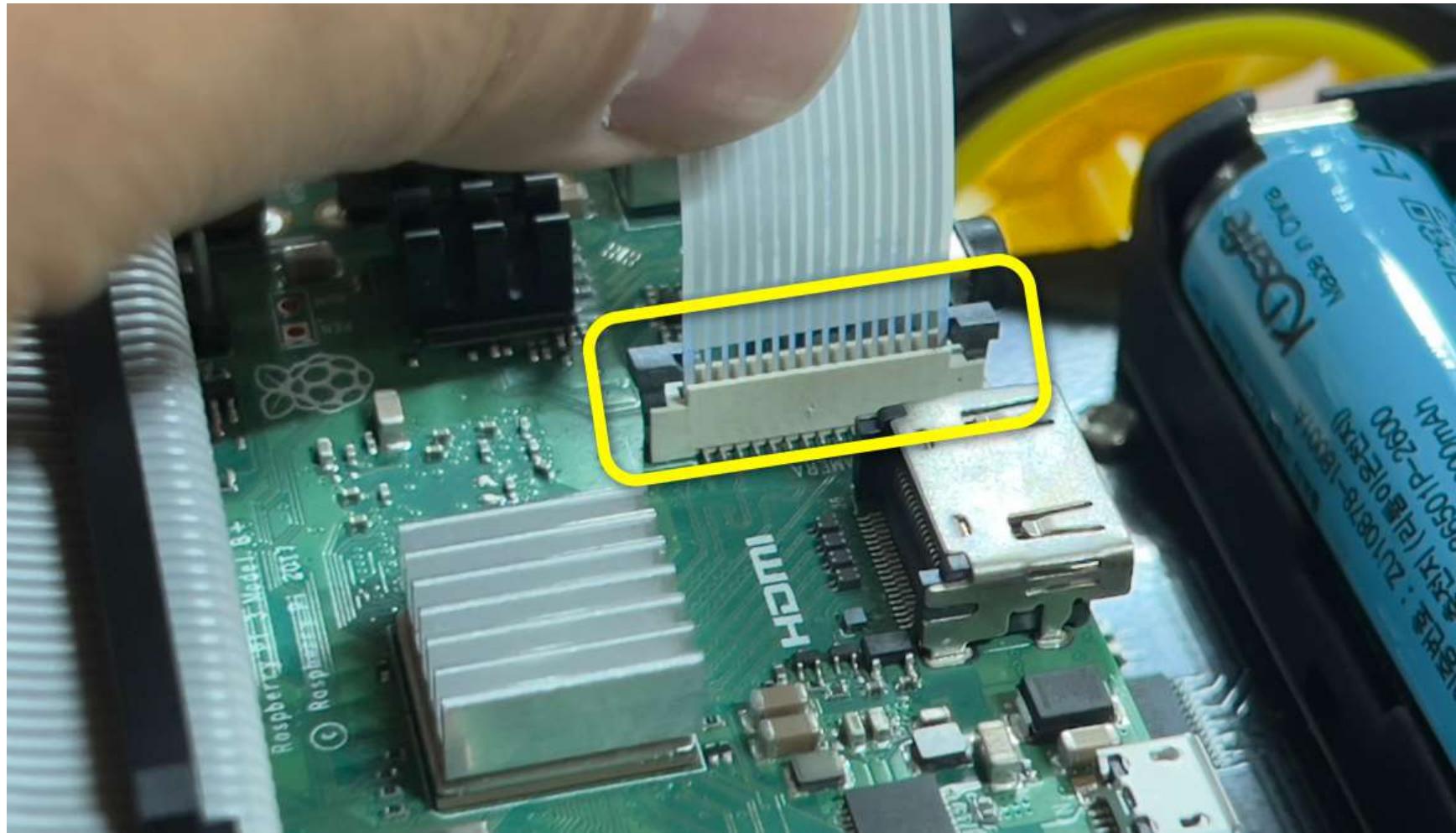
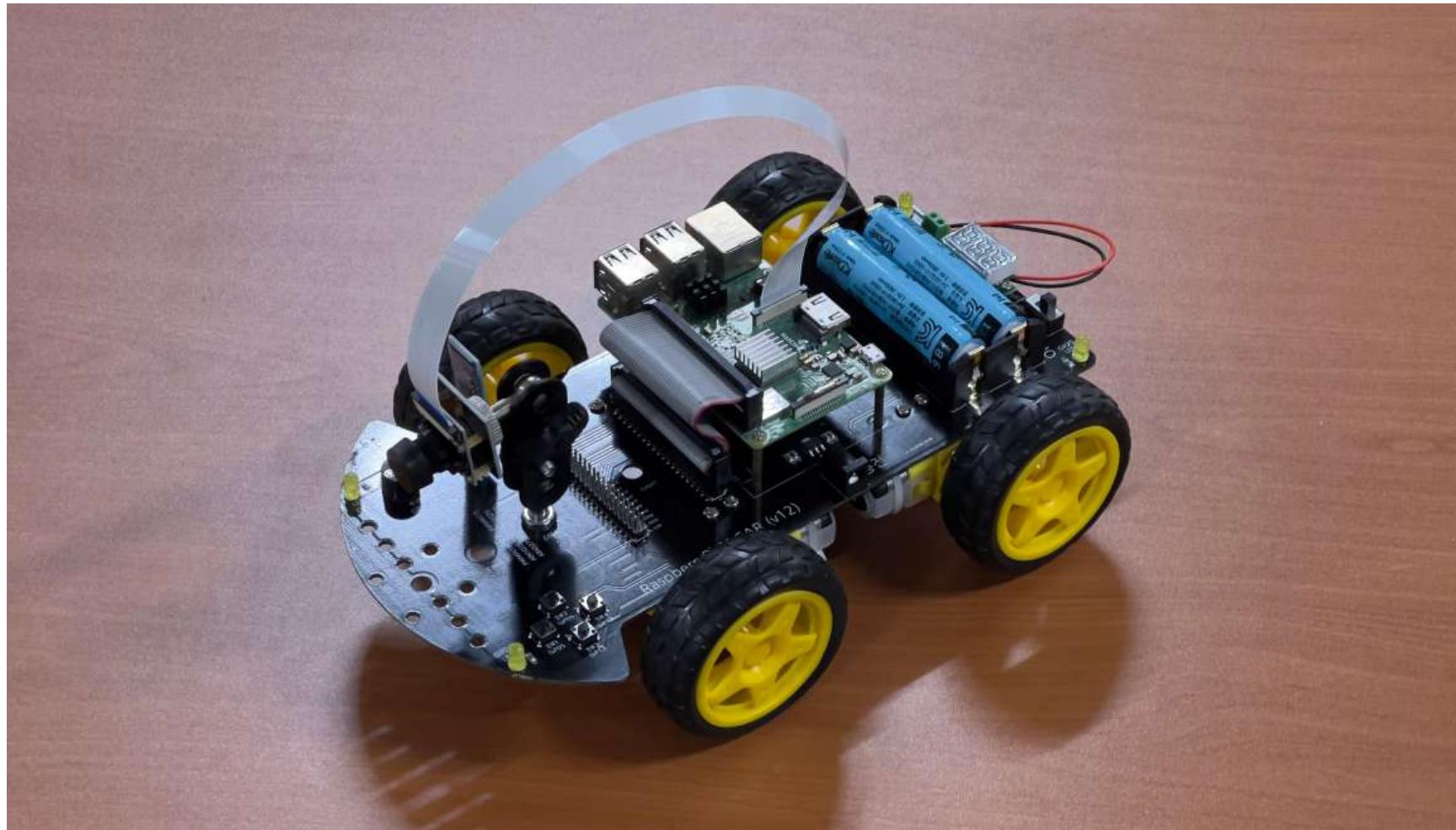
23. HM-10 블루투스 모듈을 블루투스 커넥터에 모듈의 방향과 극성을 맞춰 연결한다.  
(블루투스 모듈에 극성이 적힌 방향이 뒷면)



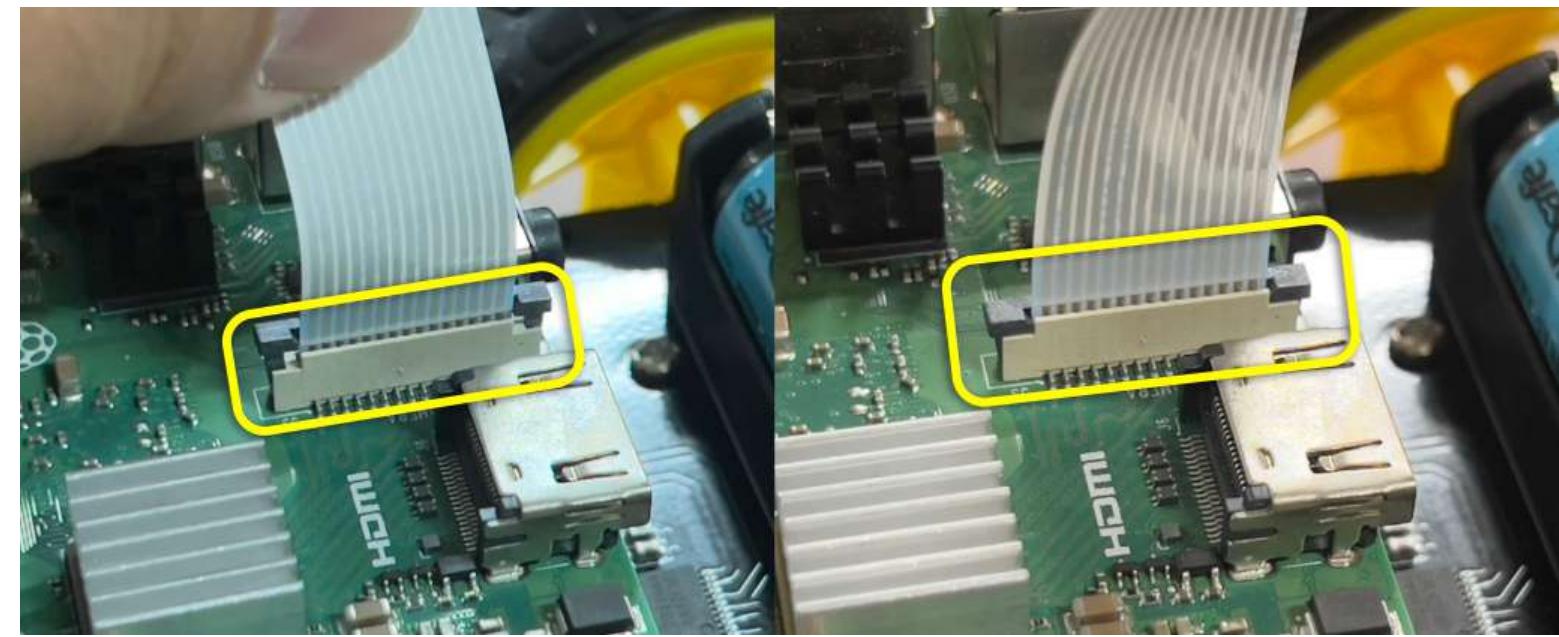
24. 라즈베리파이의 카메라 커넥터 부분을 잡고 위쪽으로 들어올린 후 카메라 모듈 케이블을 연결할 수 있도록 준비한다.



# 자율주행 자동차 조립

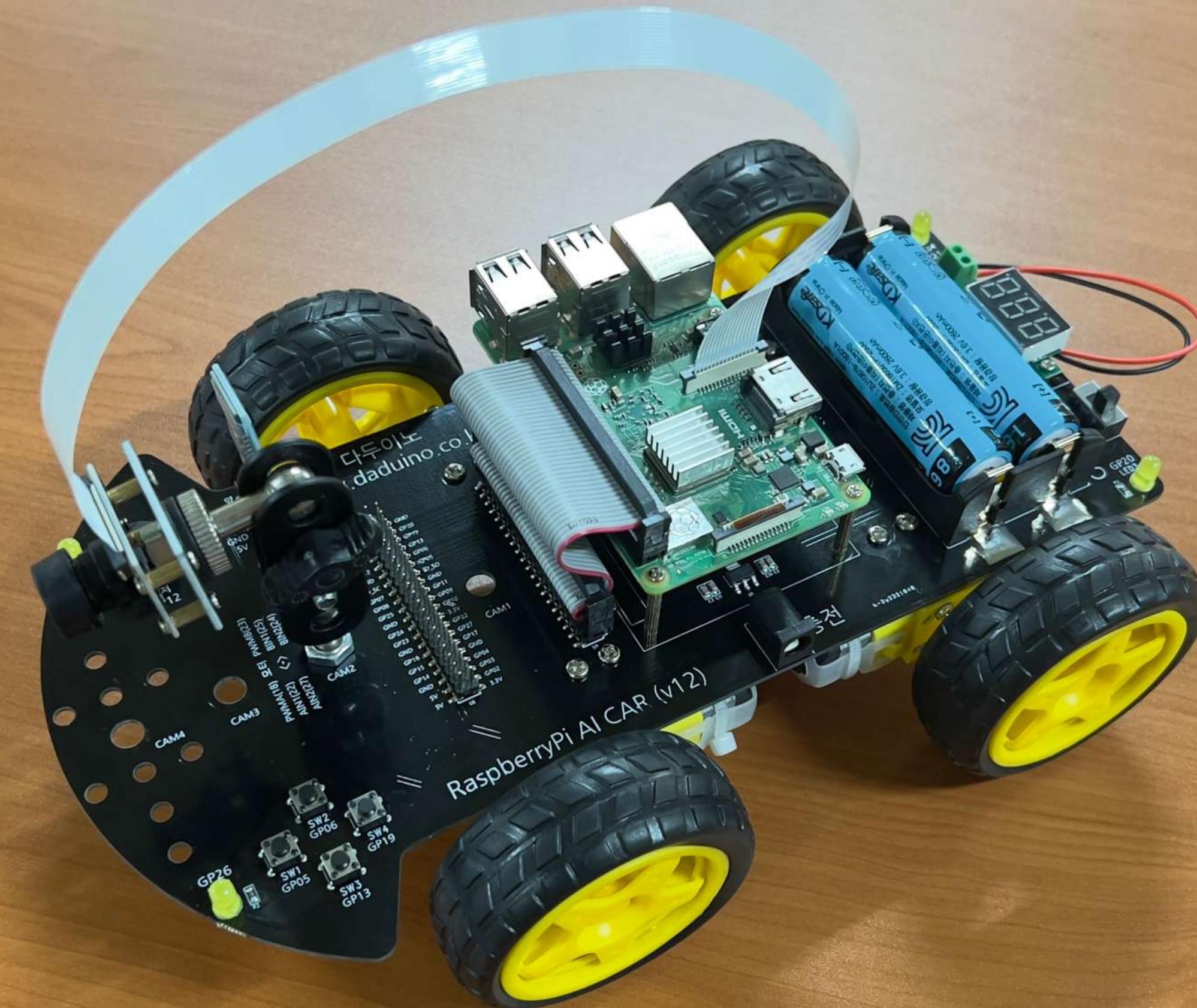


25. 카메라 모듈 케이블은 카메라 커넥터의 핀과 같은 방향에 케이블의 은색 핀의 방향을 맞춰 케이블을 커넥터 바닥에 밀어넣어 고정한다.



26. 자율주행 자동차 조립이 완료되었다.

# 자율주행 자동차 조립

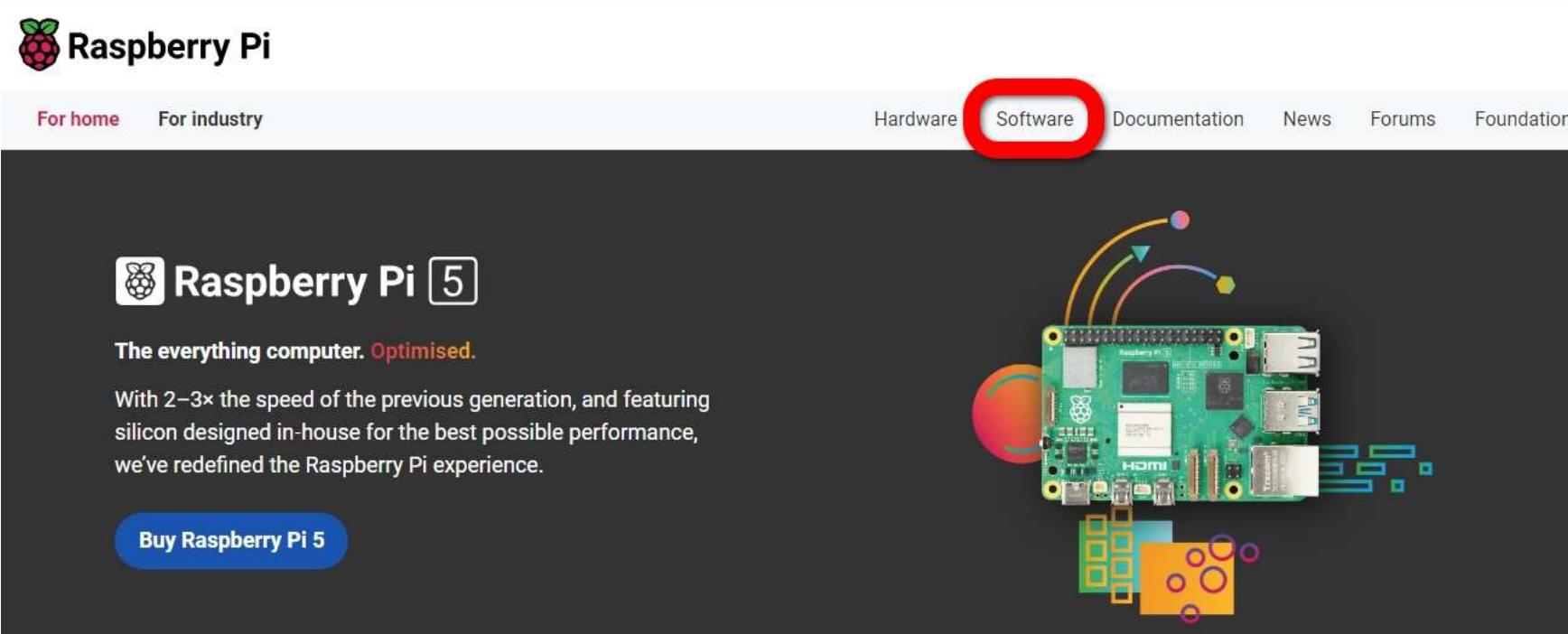


# AI 인공지능 자율주행 자동차



Part 2.  
**라즈베리파이**  
**이미지 설치**

# 라즈베리파이 이미지 설치



**Install Raspberry Pi OS using Raspberry Pi Imager**

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

**Download for Windows** (button highlighted with a red box)

[Download for macOS](#)

[Download for Ubuntu for x86](#)



## 1. 라즈베리파이 이미지 다운로드

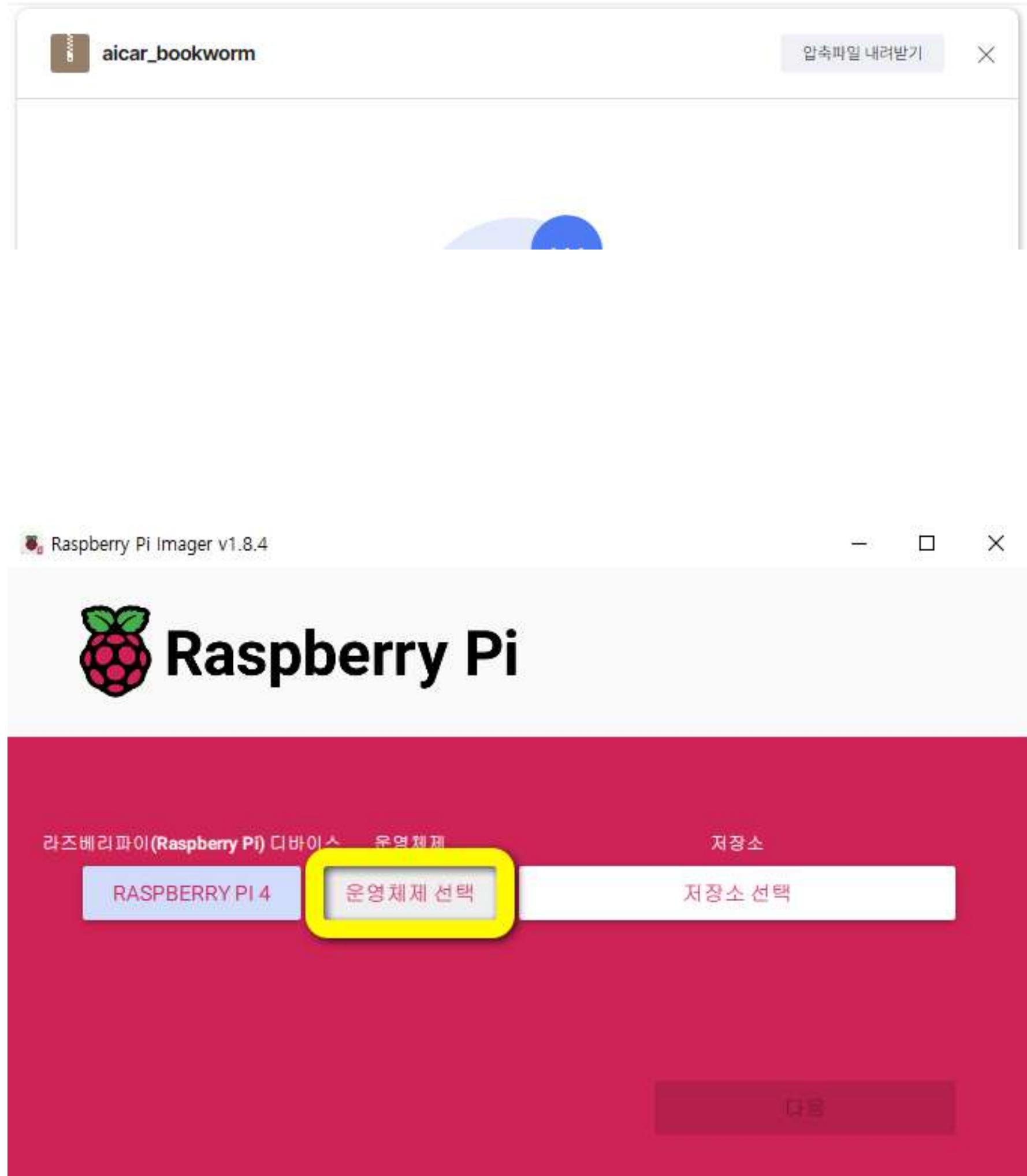
링크의 주소에서 자율주행 라이브러리 설치된 이미지 파일을 다운로드 받는다.  
OPEN CV 와 tensorflow 라이브러리가  
라즈베리파이5 업데이트 이후 지원되지 않음

## 2. Software에서 스크롤을 내린 후 프로그램을 다운로드 한다.

**Download for Window**를 클릭  
다운로드 폴더에 다운받은 파일을 설치한다.

## 3. 마이크로SD를 USB 리더기에 넣고 PC 또는 노트북의 USB 포트에 리더기를 넣는다.

# 라즈베리파이 이미지 설치



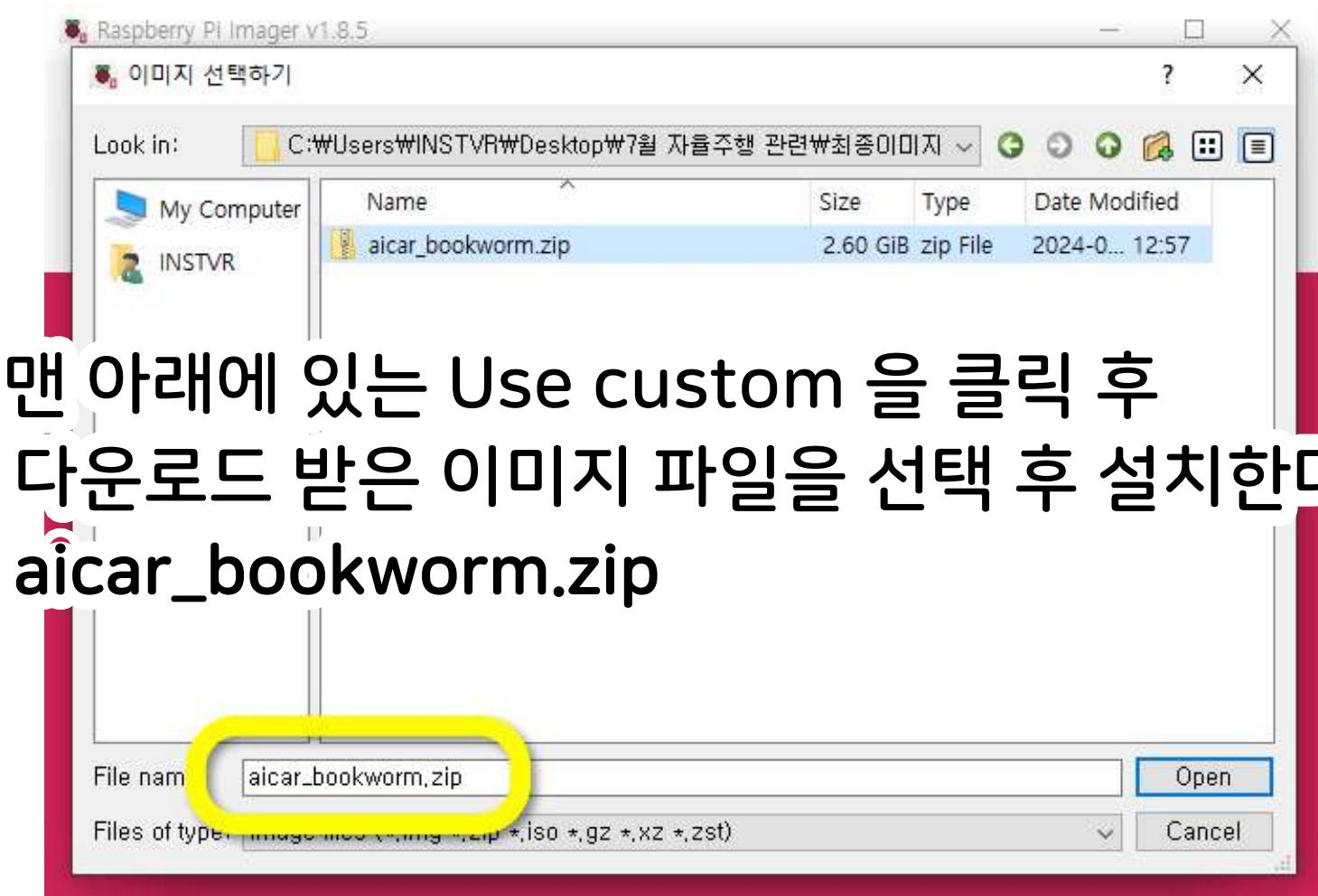
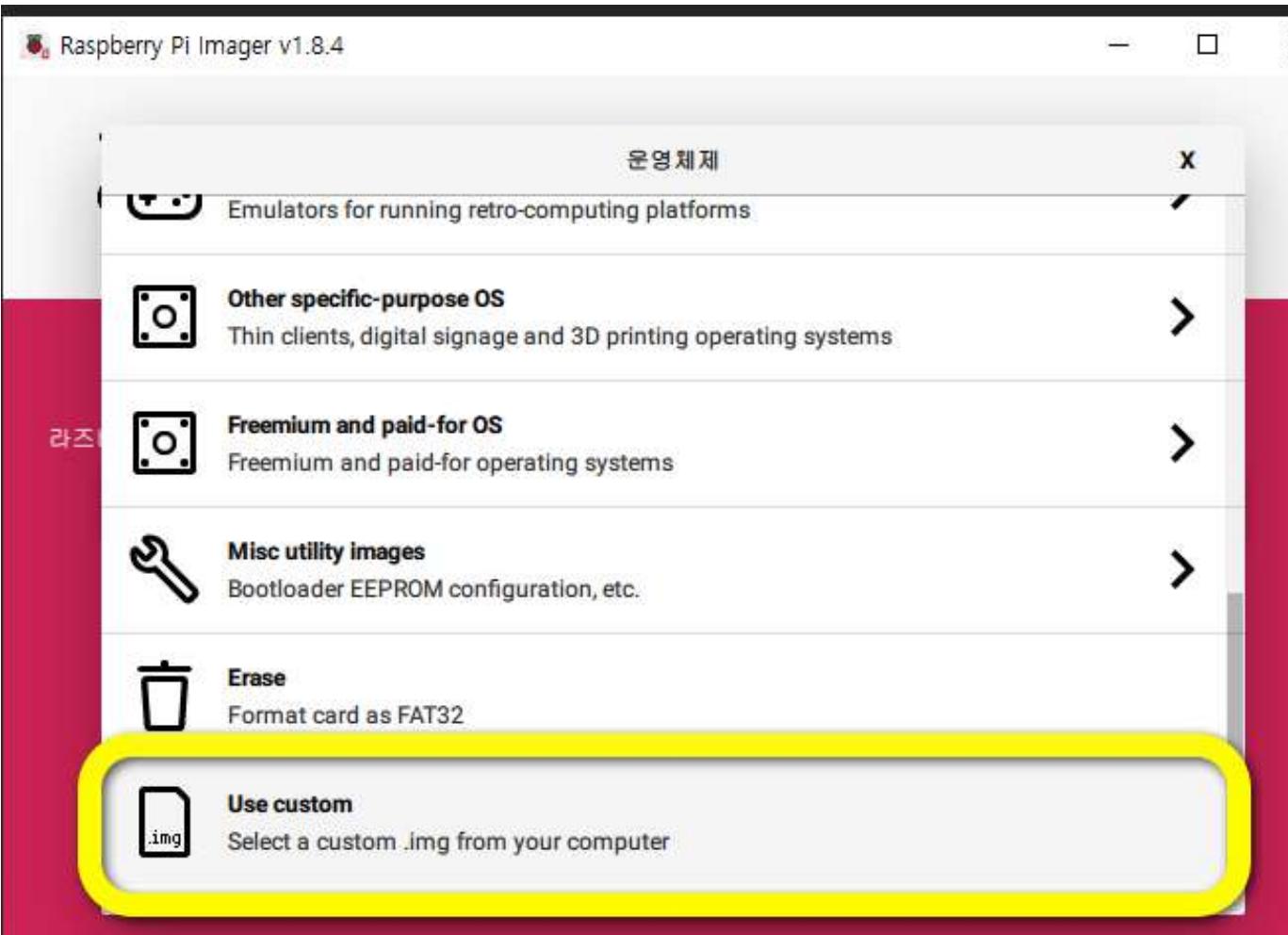
4. 라즈베리파이 이미지를 아래의 링크에서 다운로드 받는다.

<http://naver.me/GJT01yaW>

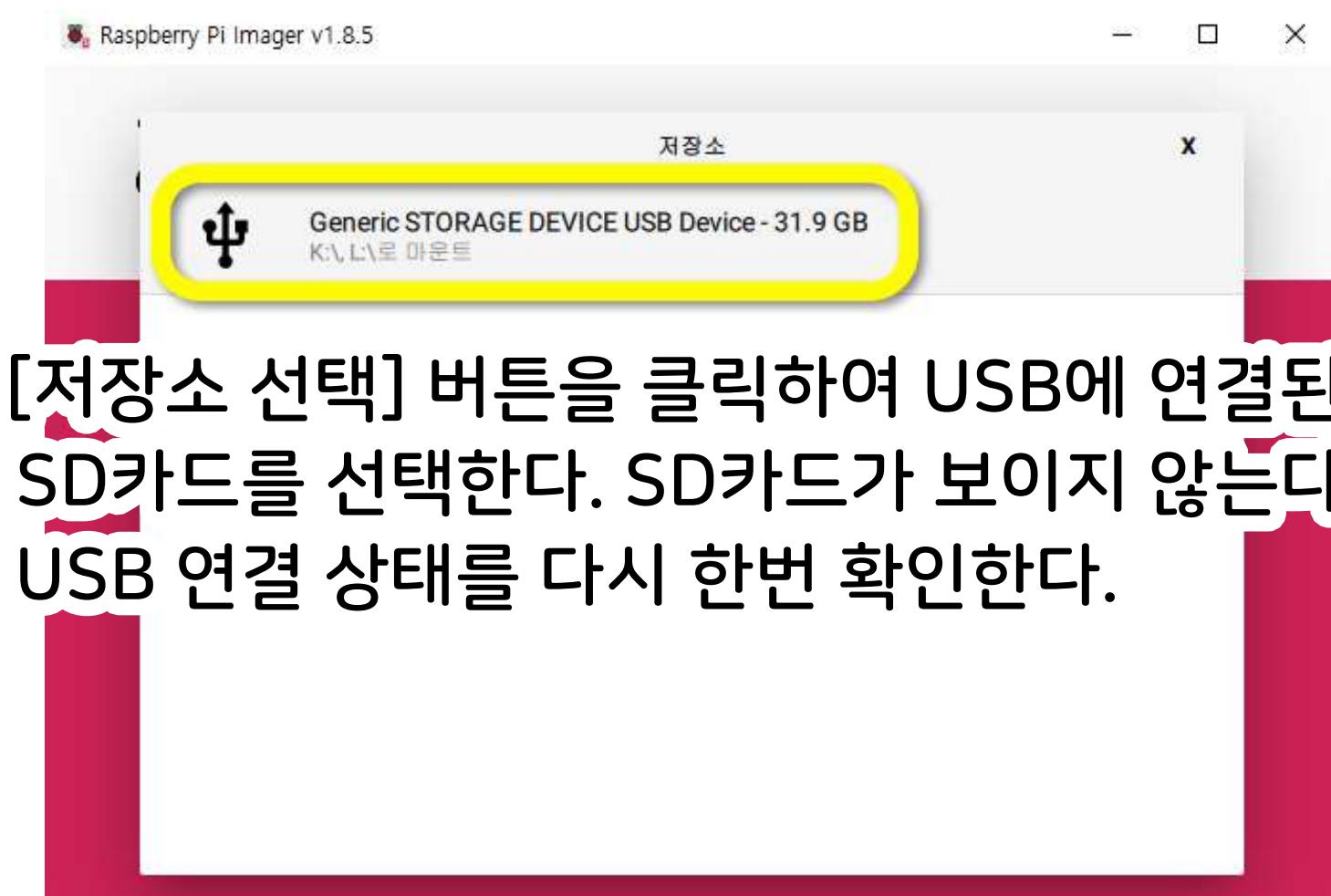


5. [장치선택] 버튼을 클릭하여 RASPBERRY PI 4 를 선택 한 후 [운영체제 선택] 버튼을 클릭한다.

# 라즈베리파이 이미지 설치

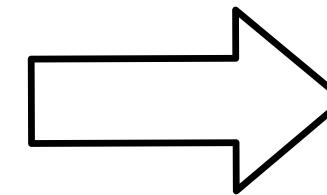
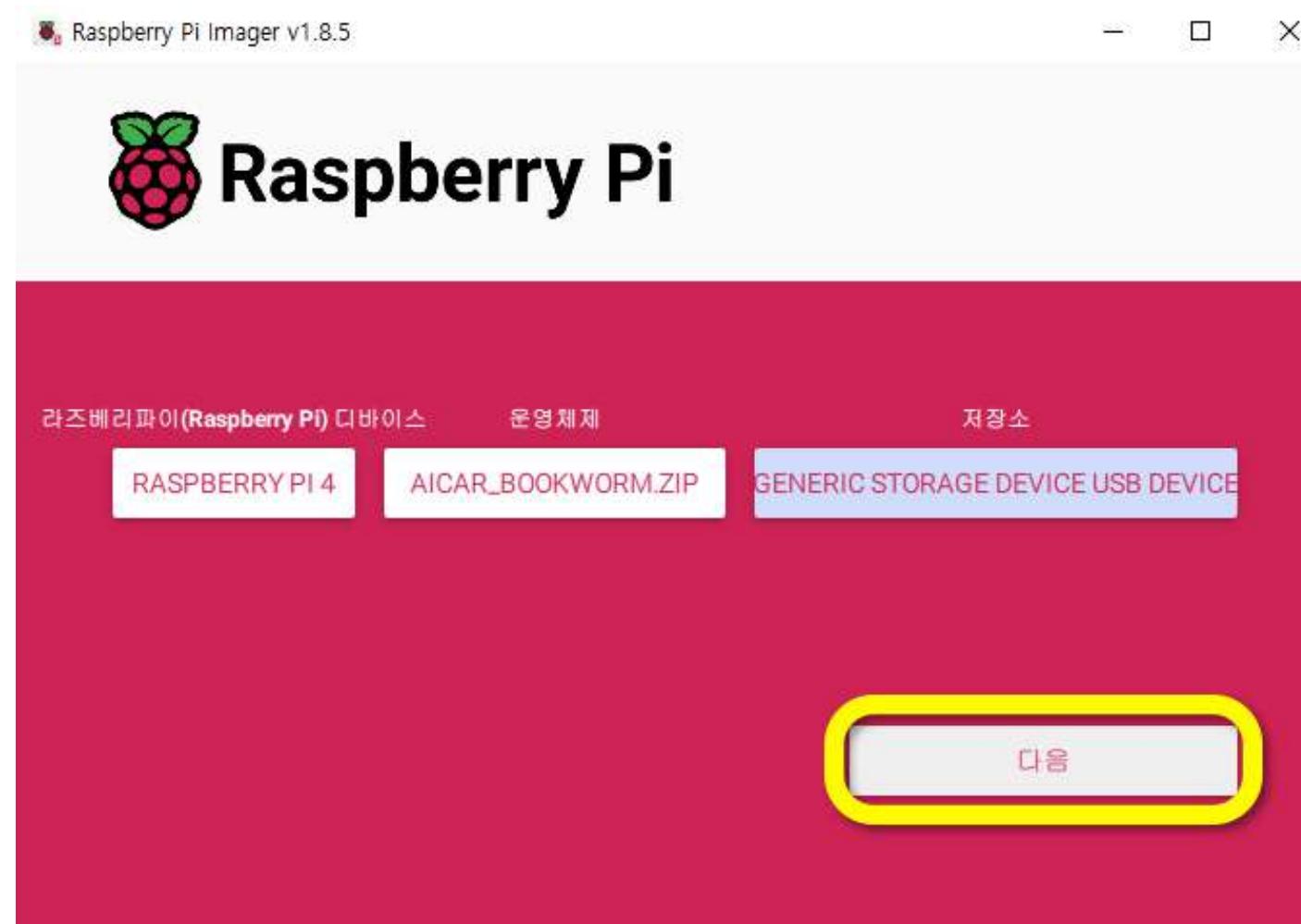


6. 맨 아래에 있는 Use custom 을 클릭 후  
다운로드 받은 이미지 파일을 선택 후 설치한다.  
**aicar\_bookworm.zip**

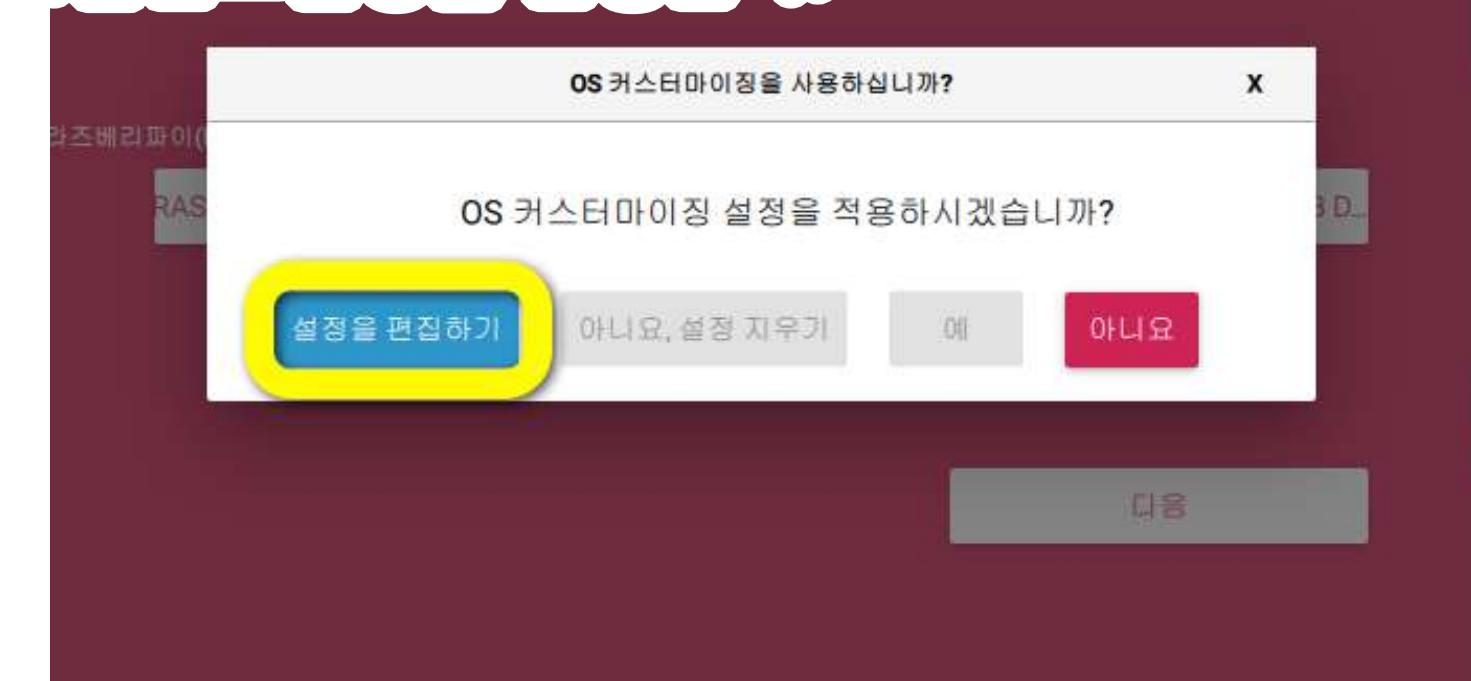


7. [저장소 선택] 버튼을 클릭하여 USB에 연결된  
SD카드를 선택한다. SD카드가 보이지 않는다면  
USB 연결 상태를 다시 한번 확인한다.

# 라즈베리파이 이미지 설치



8. 다음을 클릭하여 [설정을 편집하기] 버튼을 클릭한 후 hostname 설정과 사용자 이름 및 비밀번호 설정을 진행한다.

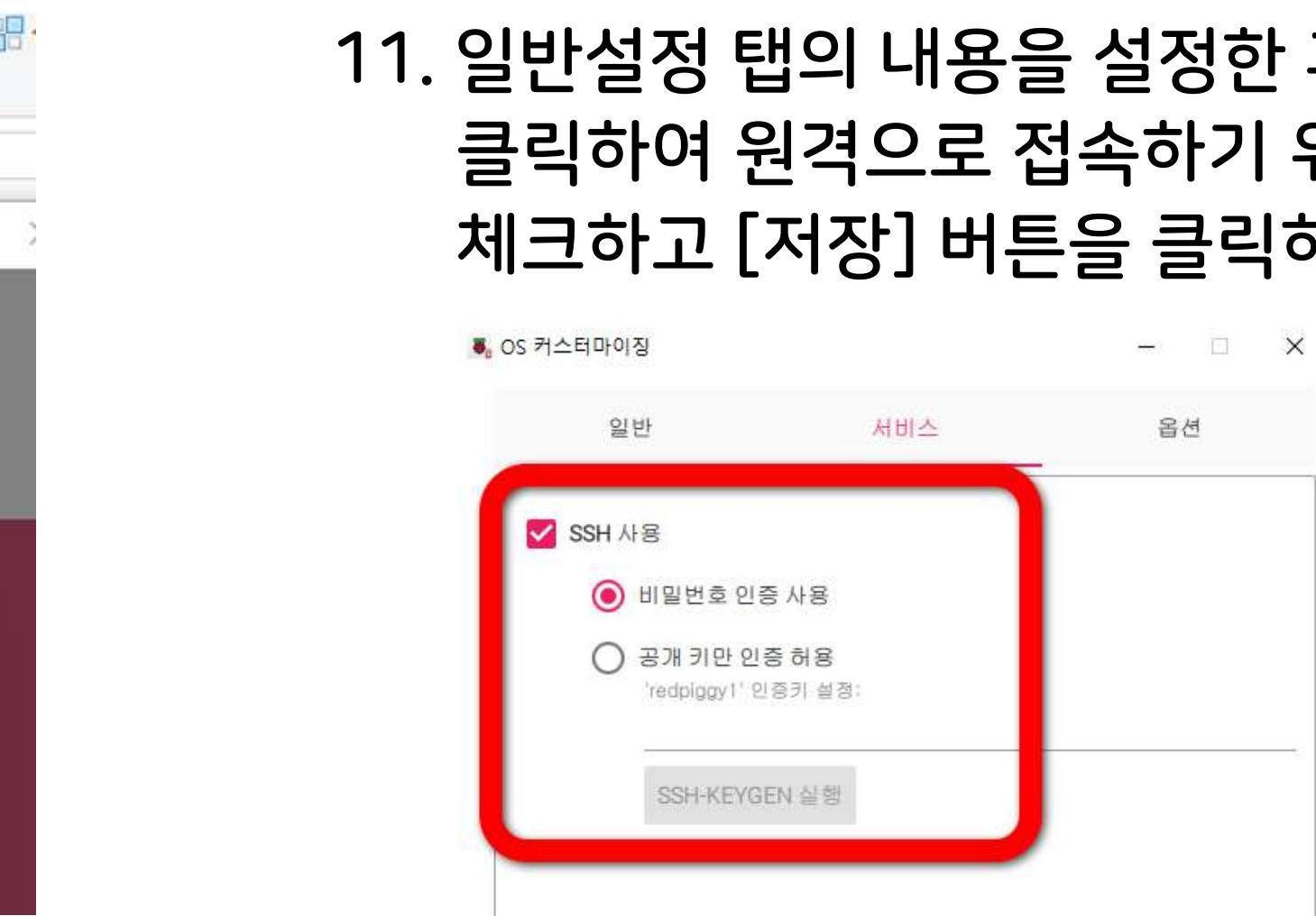


9. hostname 설정에 체크한 후 호스트 네임은 팀명으로 작성한다.  
사용자 이름과 비밀번호는 팀명으로 설정하거나 pi로 지정할 수 있다.  
팀명으로 지정 시 라즈베리파이 코드 실행 시 결과값 저장 폴더 및 데이터 폴더의 경로명에 pi로 지정되어 있는 폴더명을 팀명으로 변경해야 하므로 신중하게 선택한다.

# 라즈베리파이 OS 설치



10. 무선LAN설정을 체크한 후 SSID 및 비밀번호를 설정한다.  
SSID - 팀이 사용하고자 하는 이름  
비밀번호 - 팀이 사용하고자 하는 비번



11. 일반설정 탭의 내용을 설정한 후 서비스 탭을 클릭하여 원격으로 접속하기 위해 SSH 사용에 체크하고 [저장] 버튼을 클릭하여 설정을 저장

# 라즈베리파이 이미지 설치



12. OS 커스터마이징 설정을 적용하기 위해 [예] 버튼을 클릭하여 진행한다.



13. [예]를 눌러 이미지를 쓴다. (다소 시간이 걸림)

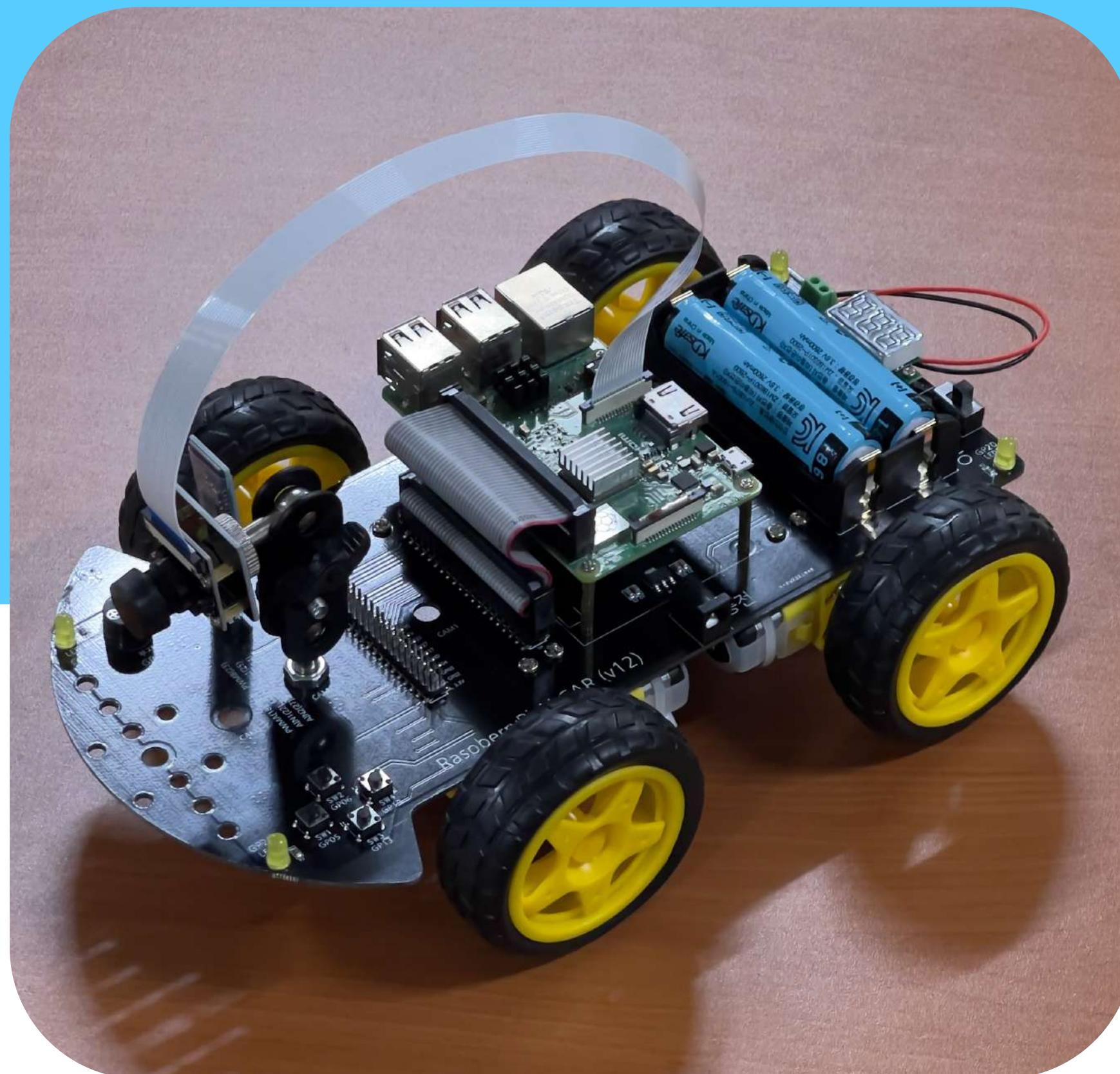


# 라즈베리파이 이미지 설치



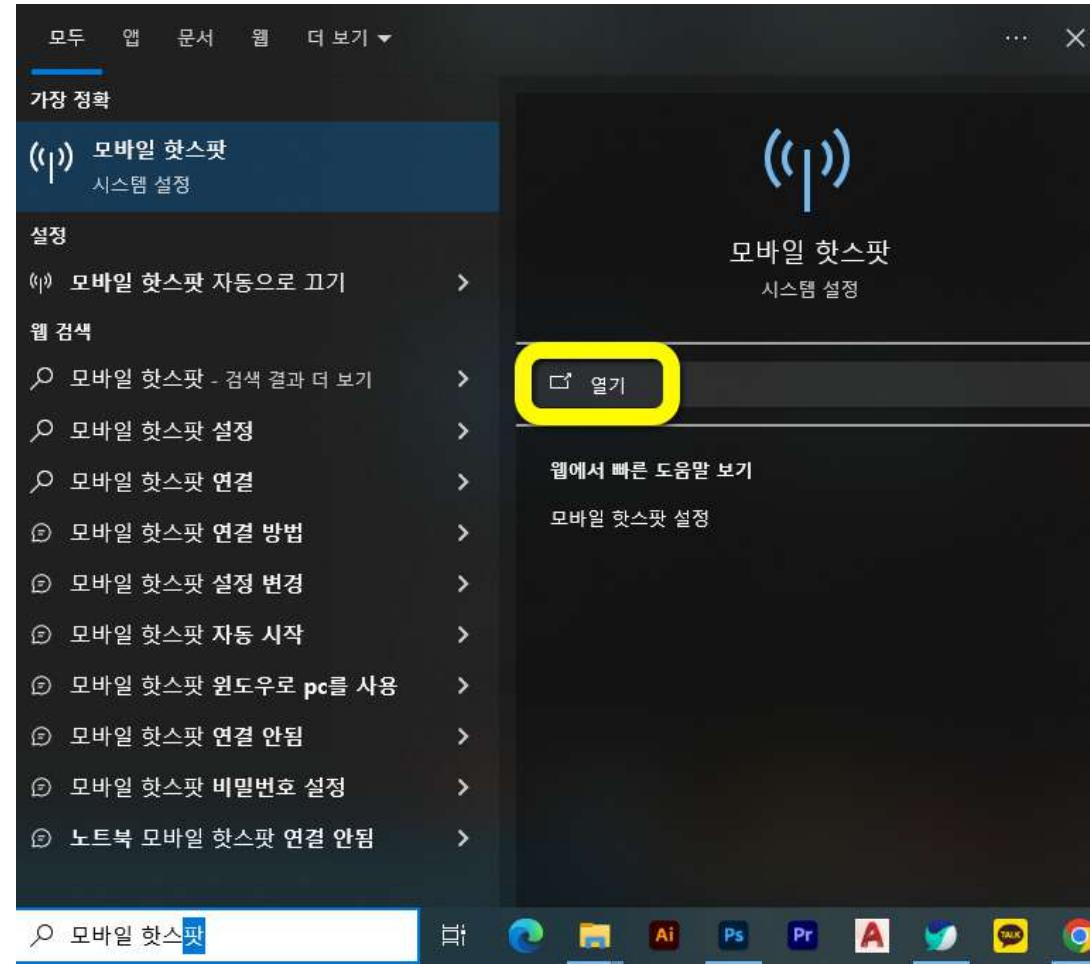
14. 완료 된 후 [계속] 을 누르고 SD카드를 PC에서 제거한다.  
※ 이미지 검증 오류나 쓰기 완료가 되지 않는 경우에는 12번 절차부터 다시 진행한다.

# AI 인공지능 자율주행 자동차



Part 3.  
**자율주행 자동차**  
**원격 접속 사용하기**

# 자율주행 자동차 운전석 사용하기



1. 핫스팟을 검색 후 설정으로 이동한다.  
윈도우 작업표시줄에 [모바일 핫스팟] 을  
검색하여 시스템 설정 [열기]를 클릭한다.  
또는 작업표시줄 오른쪽 Wi-Fi / 블루투스  
설정에서 [모바일 핫스팟] 아이콘에서 마우스  
오른쪽 버튼을 클릭하여 [설정으로 이동]  
팝업창을 클릭한다.

2. [편집]을 클릭하여 이름, 암호, 대역을 설정한다.

# 자율주행 자동차 원격접속 사용하기

## 네트워크 정보 편집

다른 사용자가 공유 연결에 사용할 네트워크 이름 및 암호를 변경합니다.

네트워크 이름  
redpiggy1

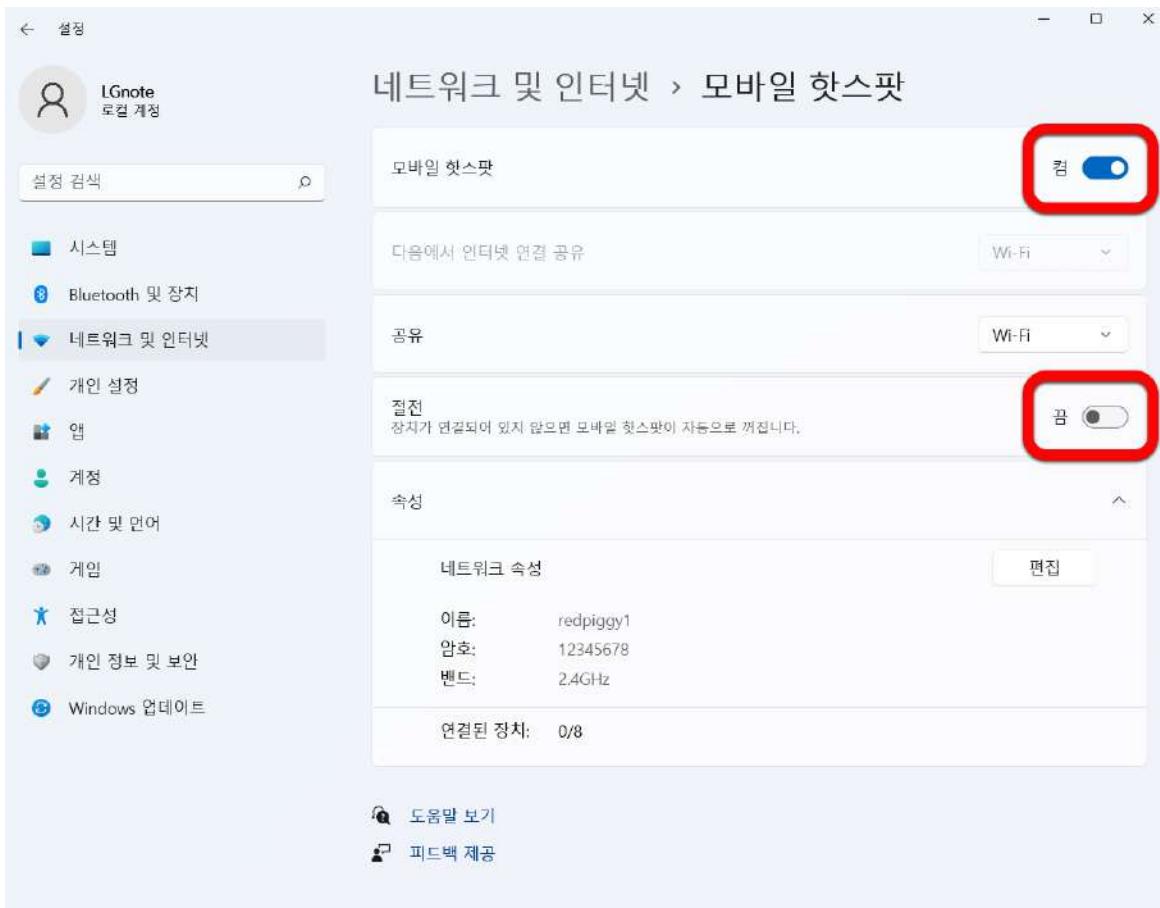
네트워크 암호(8자 이상)  
12345678

네트워크 대역  
2.4GHz

**저장** **취소**

## 3. 네트워크 이름은 한글을 사용하지 않는다.

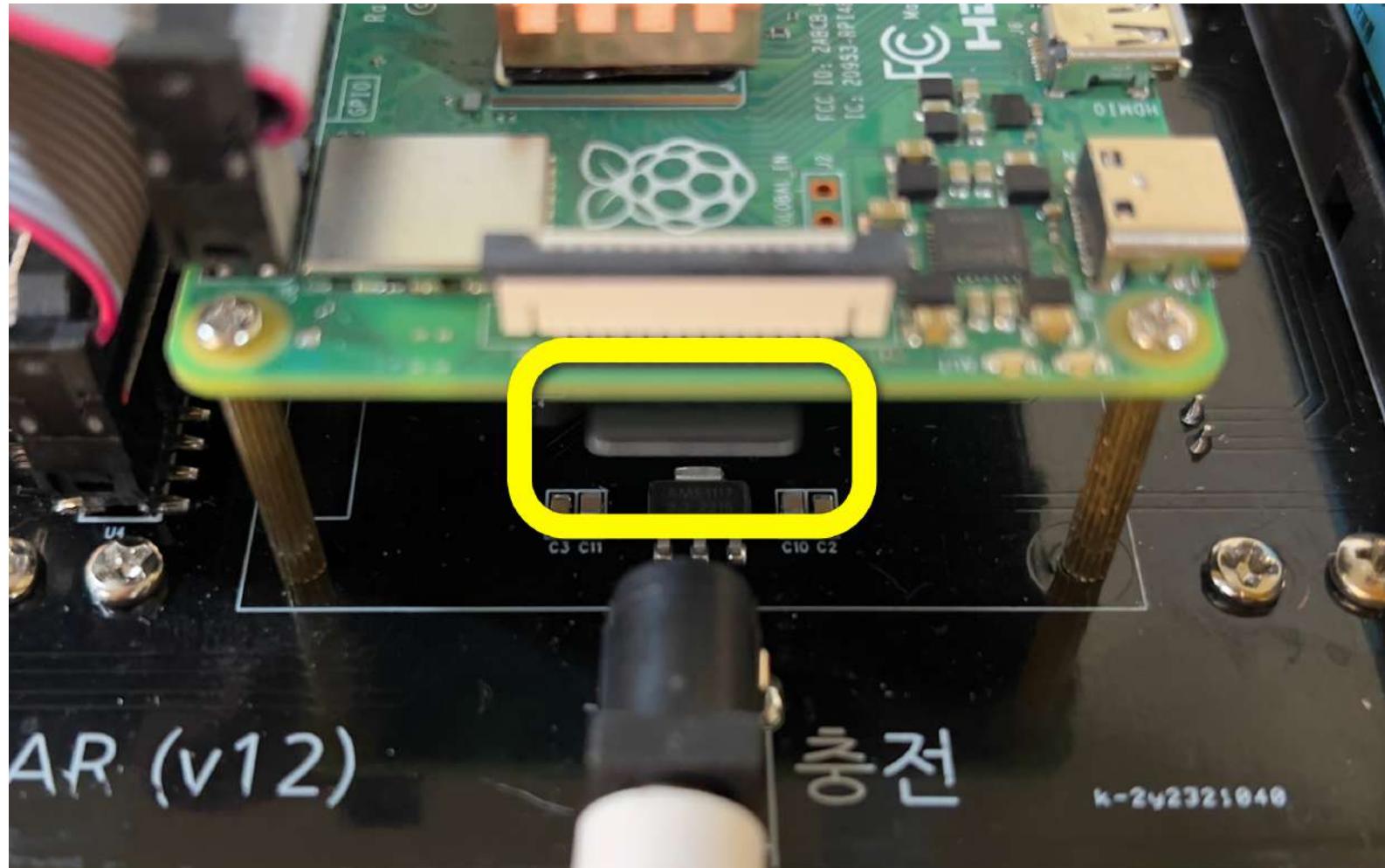
네트워크 대역은 2.4Ghz로 한다. 라즈베리파이 처음 원격 접속 시에는 2.4Ghz를 이용한다. 라즈베리파이 이미지를 쓸때 설정했던 WiFi의 접속정보대로 핫스팟의 SSID와 비밀번호를 설정한 후 [저장]버튼을 눌러 저장한다.



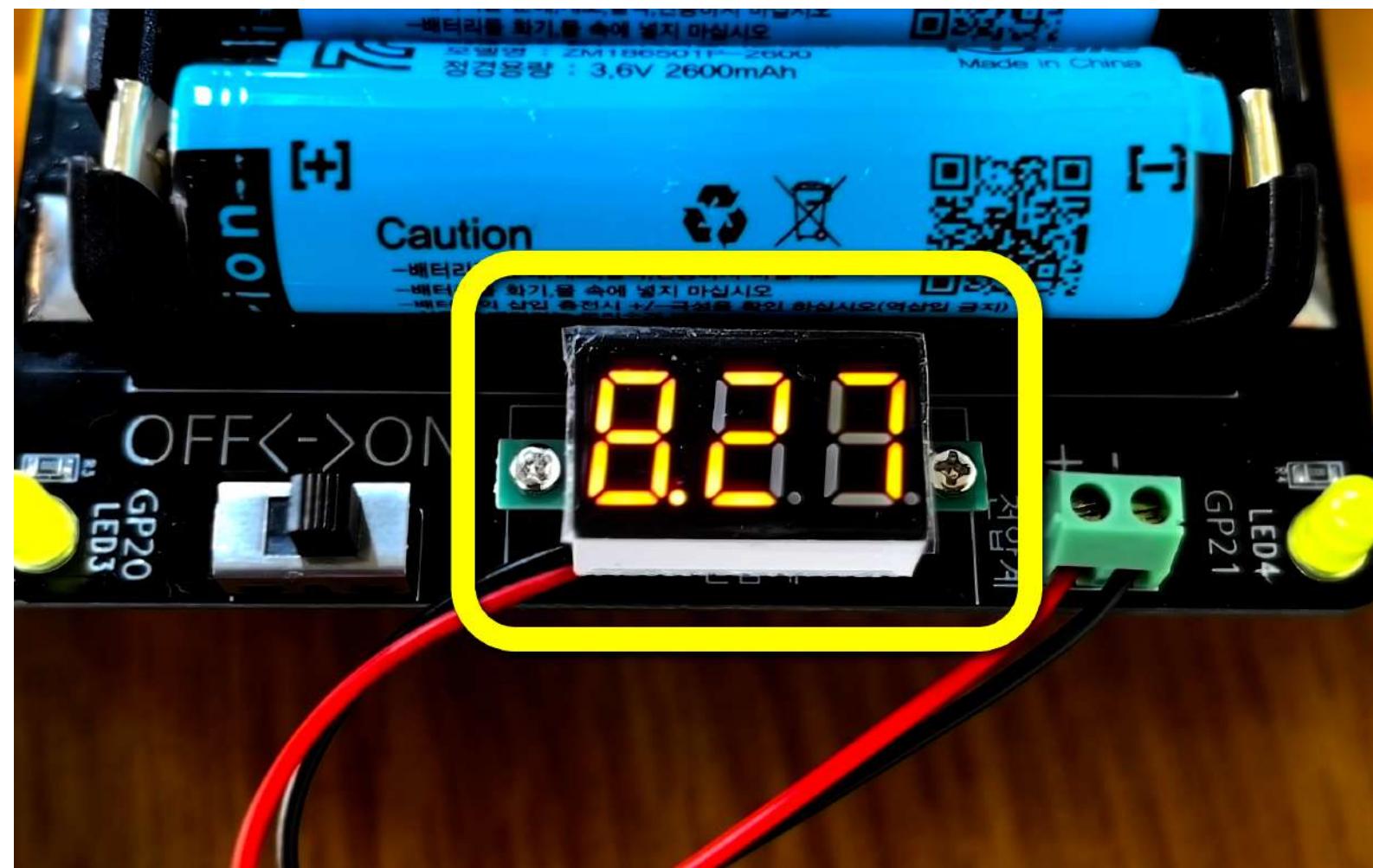
## 4. 모바일 핫스팟을 켜준다. 절전 기능은 꺼둔다.

가끔 핫스팟이 자동으로 꺼져 있어 다시 켜줘야 하는 경우가 발생하므로 모바일 핫스팟 설정 시 모바일 핫스팟을 켜주고 절전은 꺼둔다.

# 자율주행 자동차 원격접속 사용하기

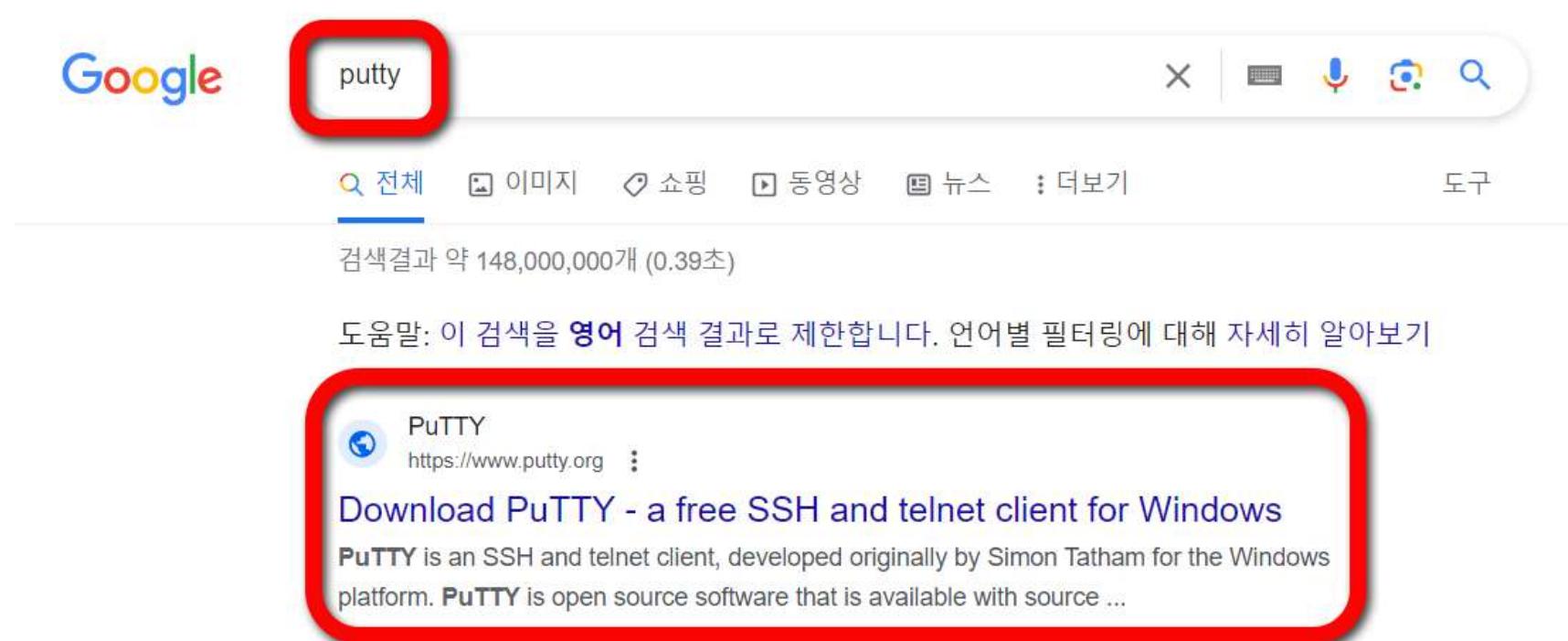
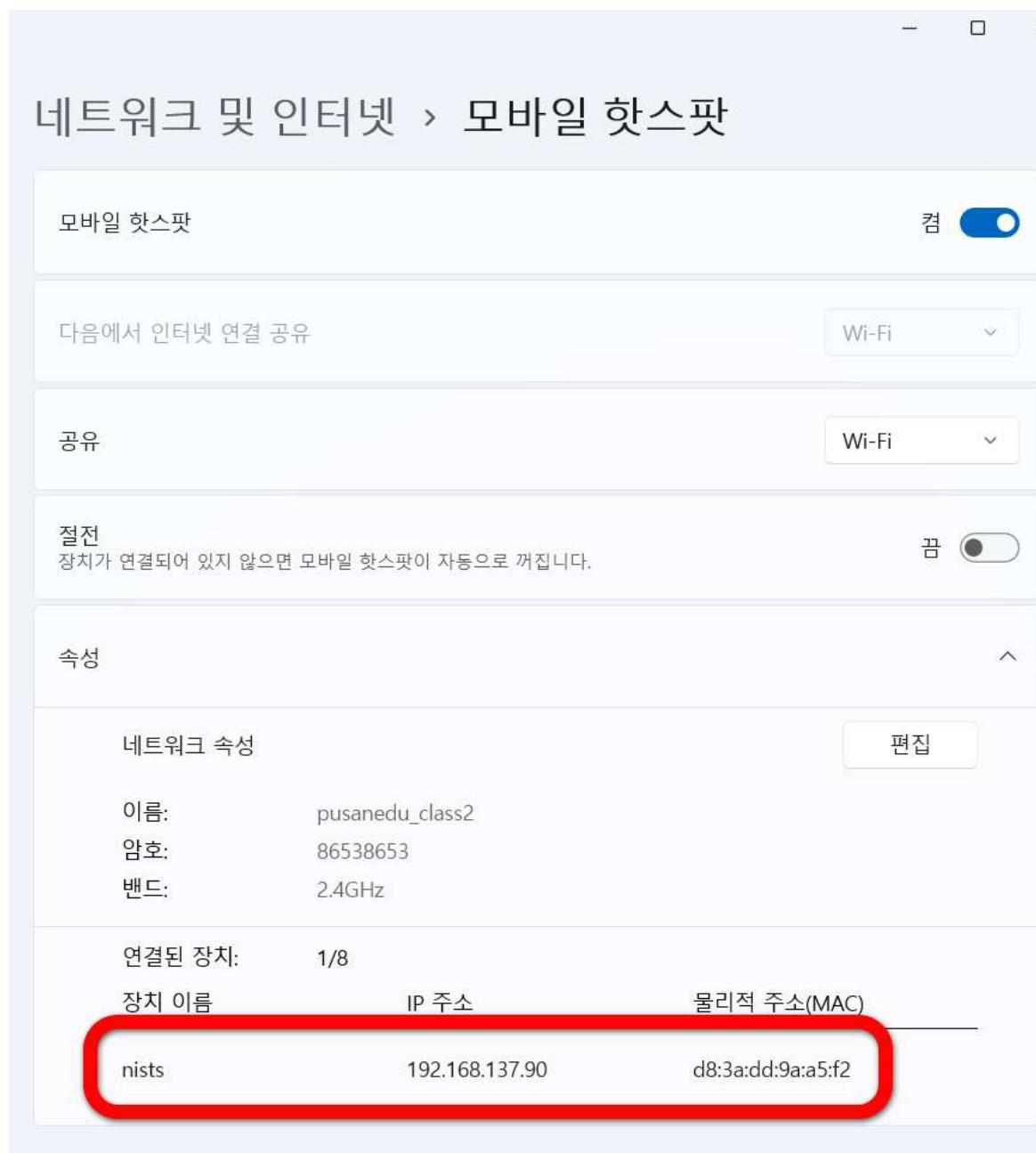


5. SD 메모리를 라즈베리파이에 넣는다.



6. 자동차 전원스위치를 ON으로 하여 전원을 켜고  
배터리 전압을 체크하여 7.1v 아래의 전압이  
나오는 경우 반드시 충전하여 사용한다.  
완충상태 - 8.4v  
일반전압 - 7.4v  
충전전압 - 7.1v

# 자율주행 자동차 원격 접속 사용하기



7. 라즈베리파이의 부팅이 완료되면 설정했던 노트북의 모바일 핫스팟을 통해서 WiFi에 접속되므로 모바일 핫스팟에서 연결장치의 IP주소 정보가 확인된다. 이 정보를 이용하여 라즈베리파이에 접속이 가능하므로 [PuTTY] 프로그램을 이용하여 라즈베리파이에 원격으로 접속할 수 있다.

8. 구글에 "putty"를 검색 하여 putty 다운로드 페이지에 접속한다.  
[Download PuTTY]를 클릭한다.



# 자율주행 자동차 원격접속 사용하기

Package files

You probably want one of these. They include versions of all the PuTTY utilities (except the new archive formats).

(Not sure whether you want the 32-bit or the 64-bit version? Read the [FAQ entry](#).)

We also publish the latest PuTTY installers for all Windows architectures as a free-of-charge download.

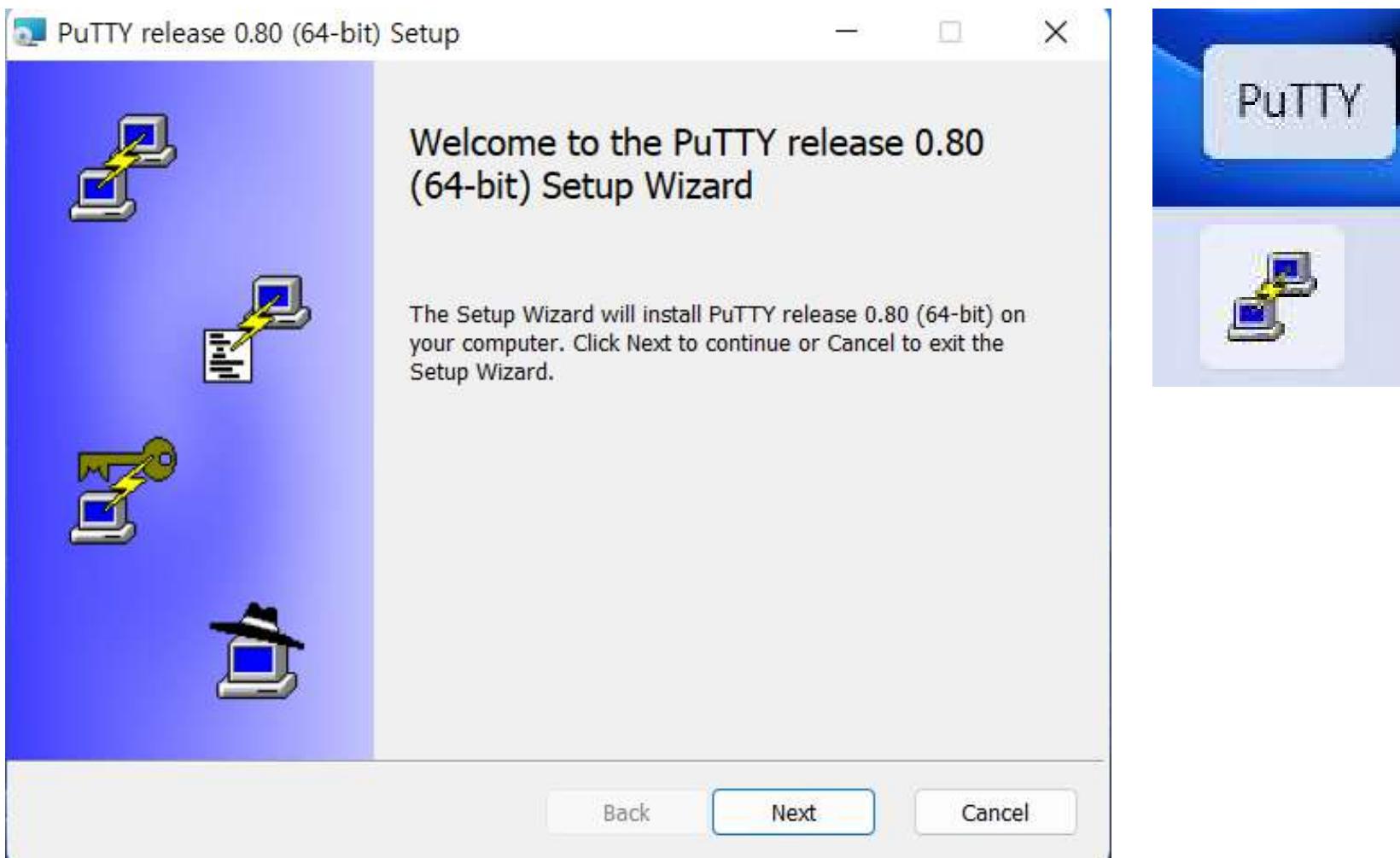
them.

**MSI ('Windows Installer')**

64-bit x86:	<a href="#">putty-64bit-0.80-installer.msi</a>	<a href="#">(signature)</a>
64-bit Arm:	<a href="#">putty-arm64-0.80-installer.msi</a>	<a href="#">(signature)</a>
32-bit x86:	<a href="#">putty-0.80-installer.msi</a>	<a href="#">(signature)</a>

**Unix source archive**

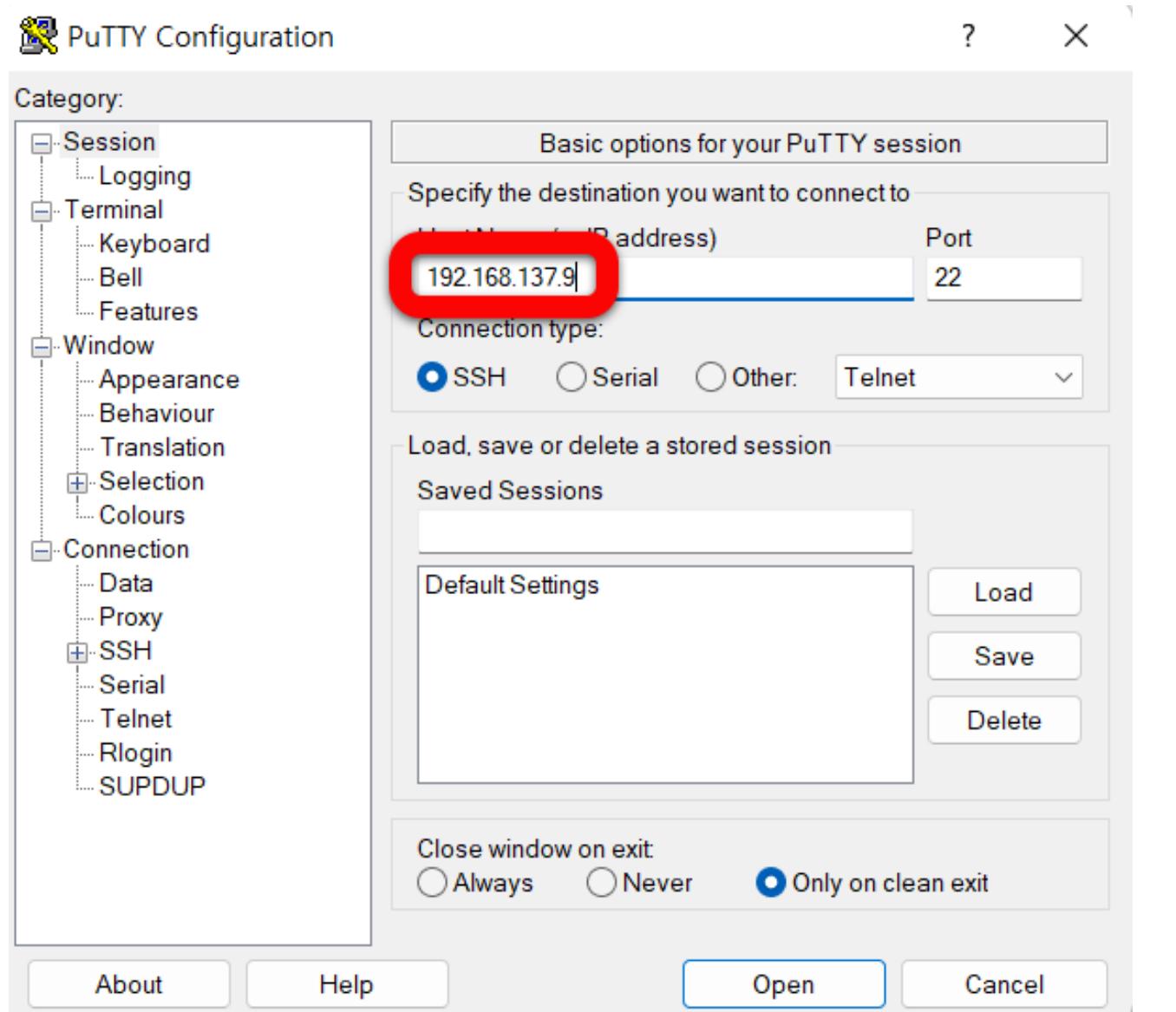
.tar.gz:	<a href="#">putty-0.80.tar.gz</a>	<a href="#">(signature)</a>
----------	-----------------------------------	-----------------------------



9. 사용중인 컴퓨터의 환경에 맞춰 윈도우용 설치파일을 클릭하여 다운로드 받는다.

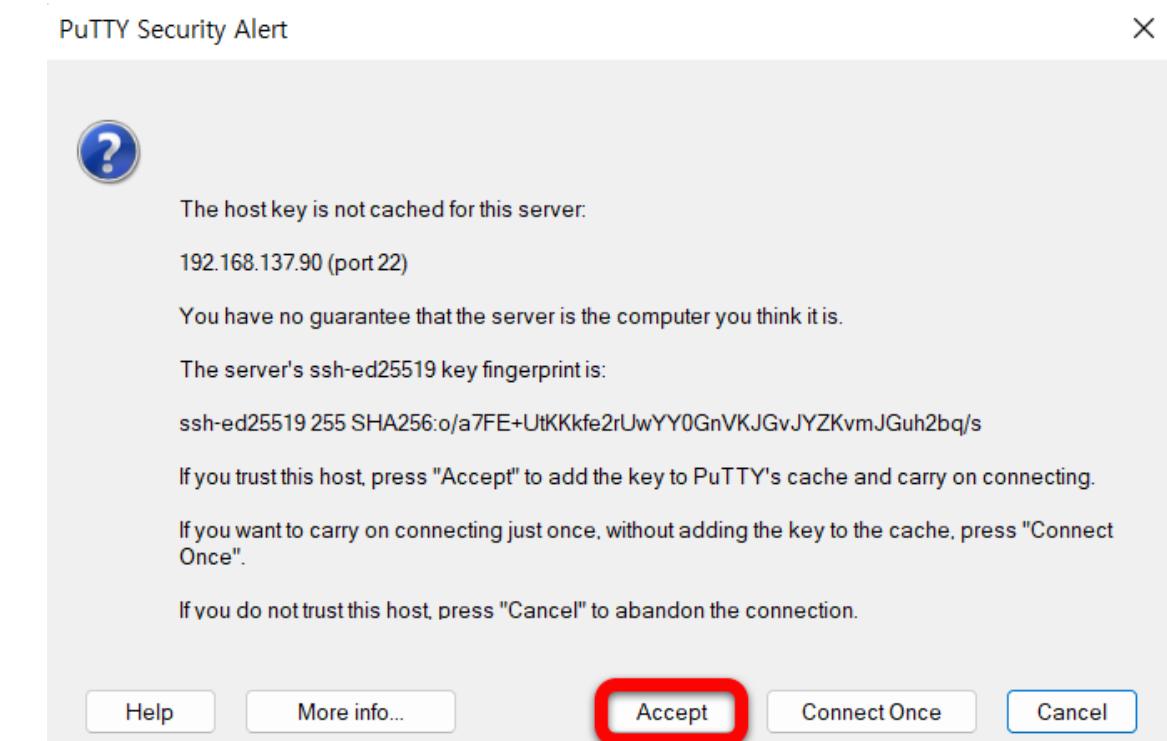
10. 다운로드 완료 후 설치한다.  
설치완료 후 putty를 실행한다.

# 자율주행 자동차 원격 접속 사용하기



A screenshot of the PuTTY terminal window titled '192.168.137.212 - PuTTY'. It shows the initial login prompt:  
login as: redpiggy1  
redpiggy1@192.168.137.212's password: [REDACTED]

11. Host Name 부분에 라즈베리파이가 접속된 IP주소를 넣은 후 [OPEN] 버튼을 클릭한다. 처음 접속 시 팝업에서 [Accept]를 눌러 계속 진행한다.



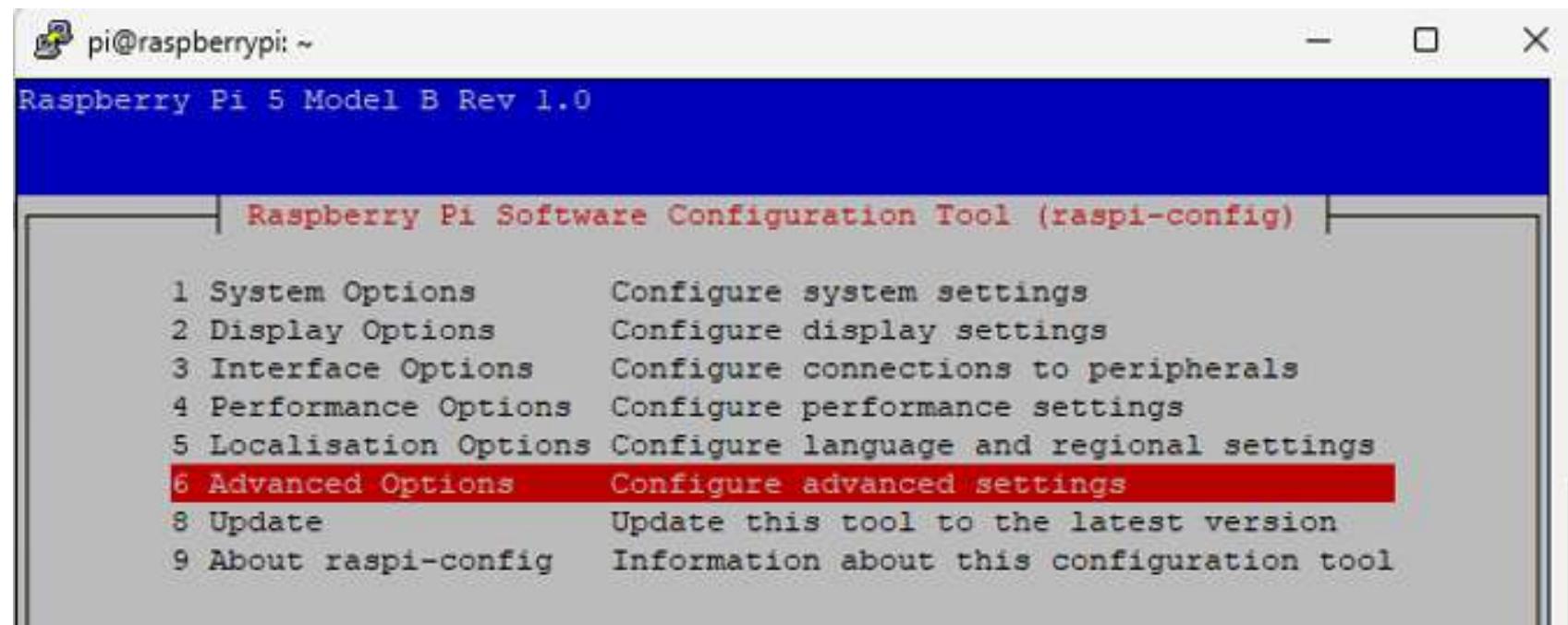
12. 로그인 ID는 라즈베리파이 이미지 설치 시 설정한 사용자 이름 [팀명] 로그인 비밀번호는 사용자 이름 설정에 입력한 비밀번호를 입력

A screenshot of a terminal window titled 'redpiggy1@redpiggy: ~'. It shows the following output:  
login as: redpiggy1  
redpiggy1@192.168.137.212's password:  
Linux redpiggy 5.10.17-v7l+ #1414 SMP Fri Apr 30 13:20:47 BST 2021 armv7l  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*copyright.  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon Jan 15 01:50:53 2024 from 192.168.137.1  
redpiggy1@redpiggy: ~ [REDACTED]

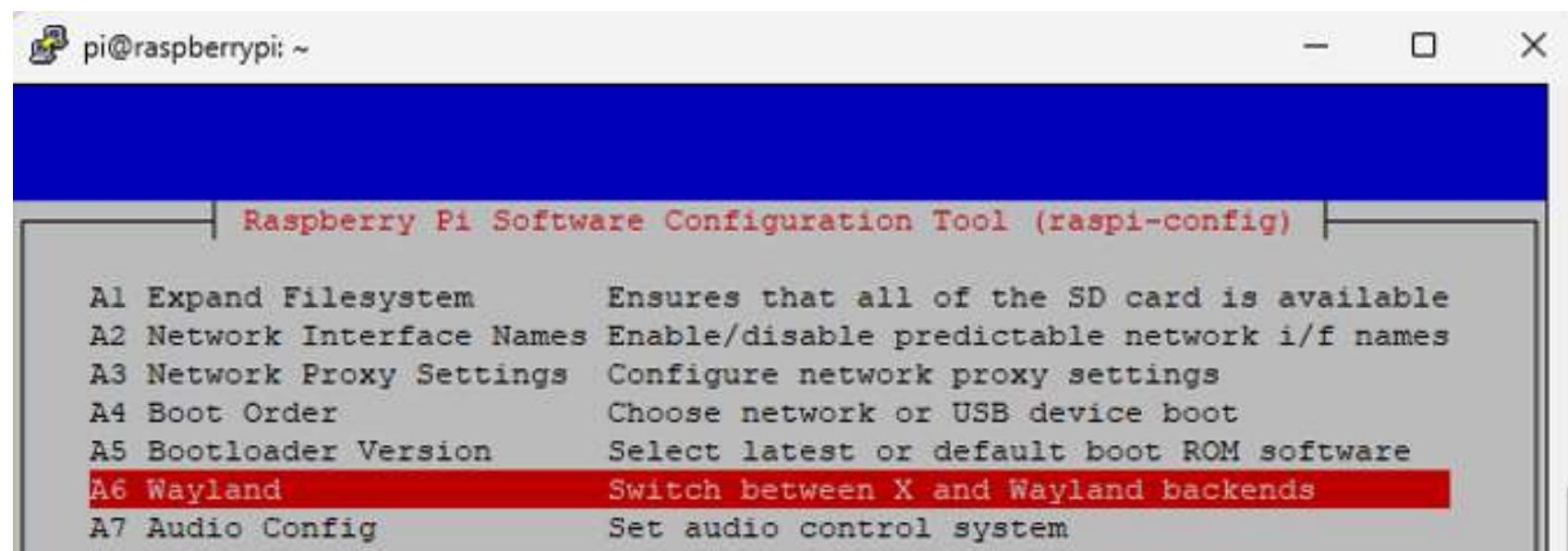
# GUI환경 구성

```
sudo raspi-config
```

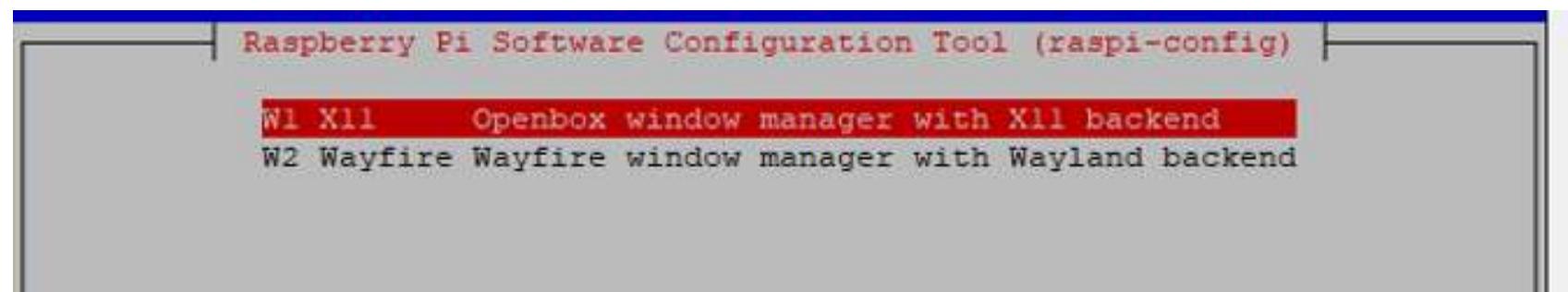
13. 라이베리파이 설정으로 진입하기 위한 명령어를 입력한다.



14. 6 Advanced Options로 이동한다.  
화살표키를 눌러 이동한 다음 엔터로 이동할 수 있다.

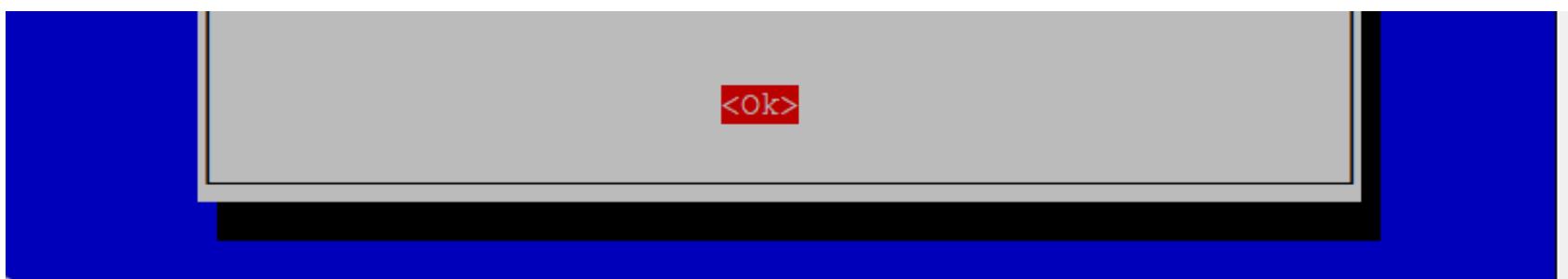


15. [A6 Wayland]로 이동

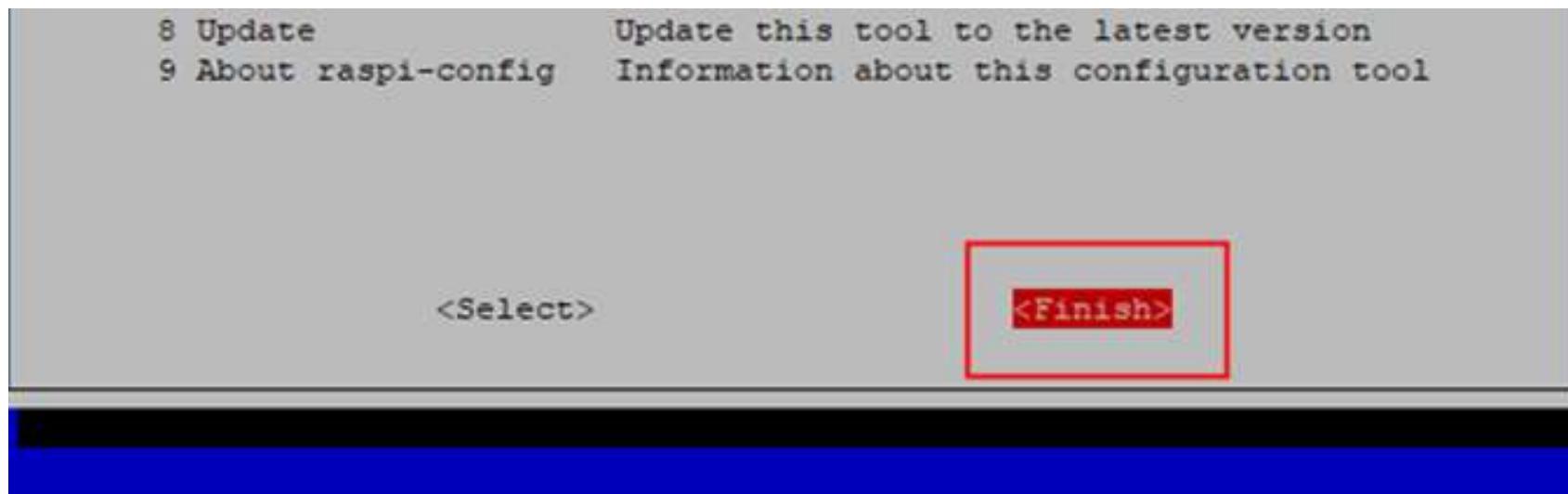


16. X11이 아닌 Wayfire가 기본설정으로 되어 있어 VNC에서 화면을 보여 줄 수 없어서 기본 버전인 X11로 변경한다.

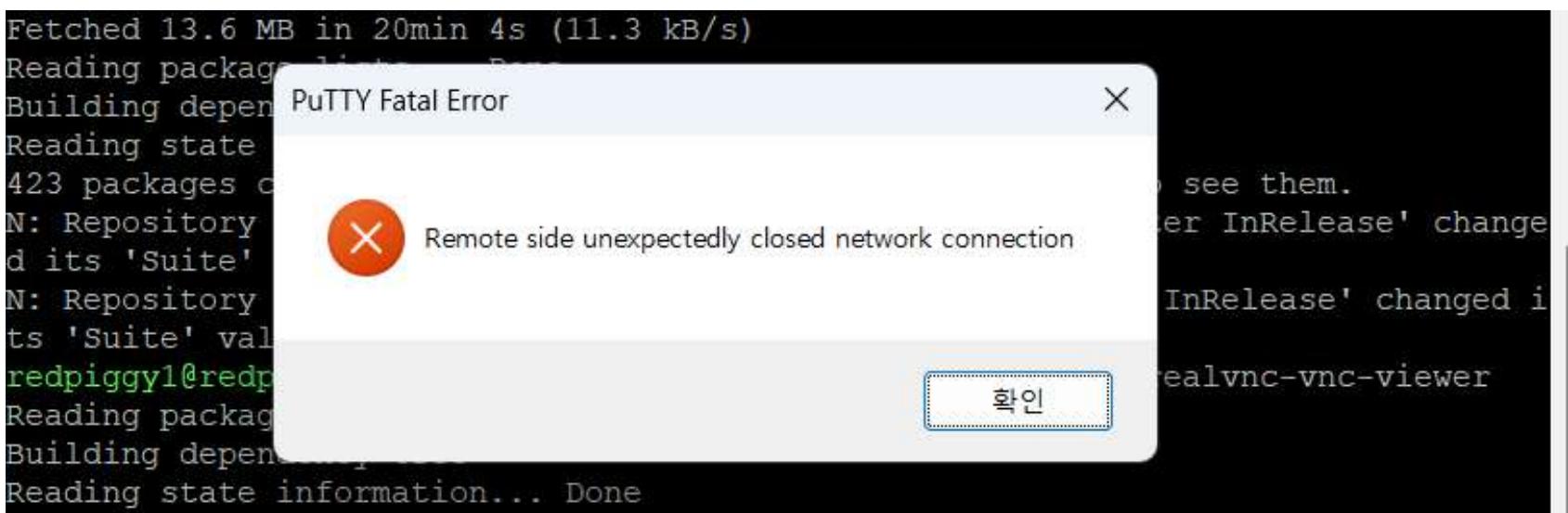
# 로봇 GUI환경 구성



17. <ok> 를 엔터키로 선택한다.



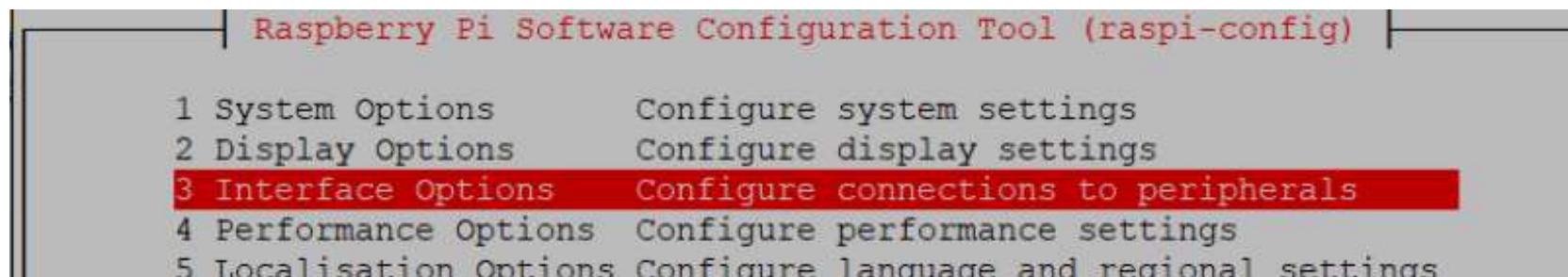
18. [탭] 키를 두번 눌러 <Finish>로 이동한 다음 엔터를 입력한다.



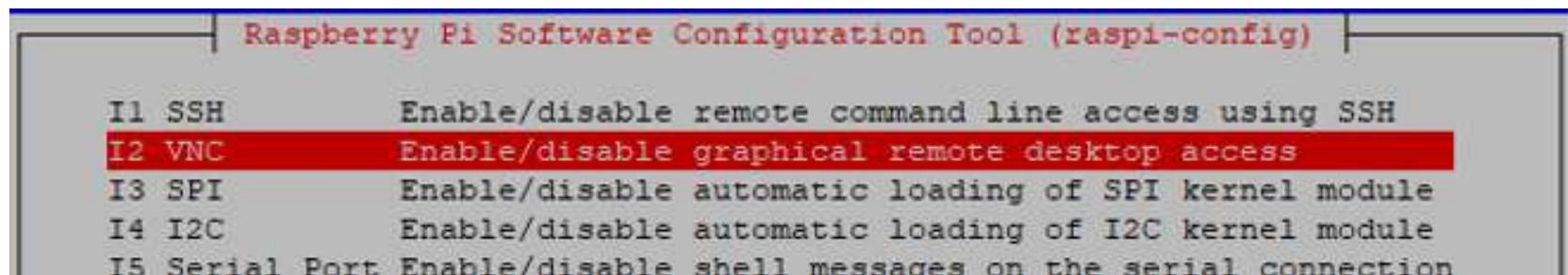
19. 자동으로 PUTTY 창이 접속 종료된다.  
[확인] 버튼을 눌러 종료한다.

20. 11번 ~ 13번까지 반복하여 라즈베리파이 설정툴로 들어간다.

# 환경 구성



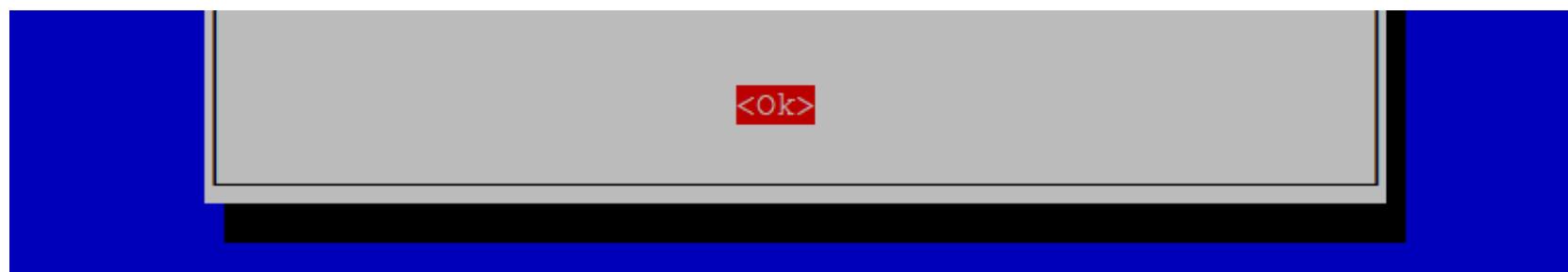
21. VNC 옵션을 켜기 위해서 [3 Interface Options]로 이동한다.



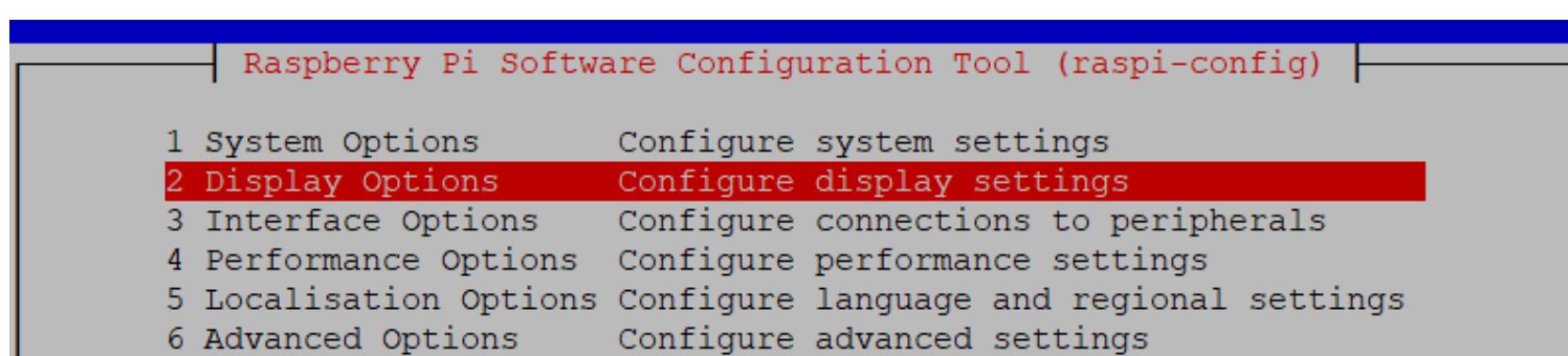
22. [I2 VNC]를 클릭한다.



23. <Yes>를 눌러 VNC를 사용함으로 설정한다.

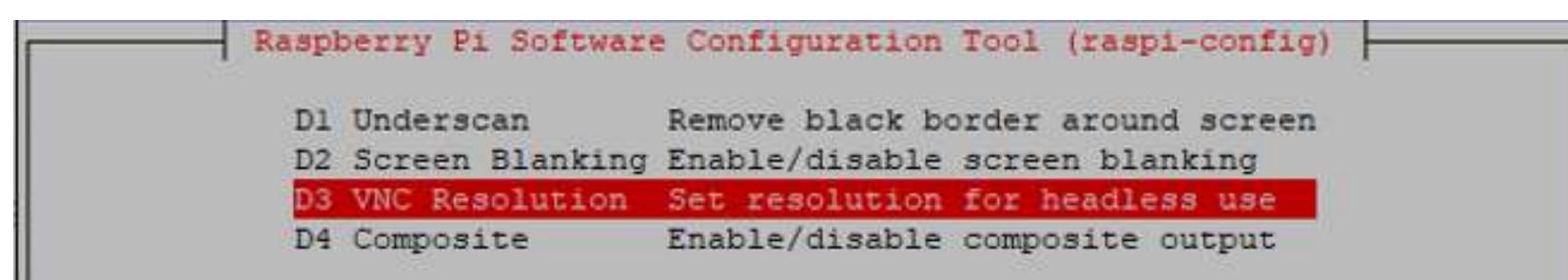


24. <ok>를 엔터키로 선택한다.

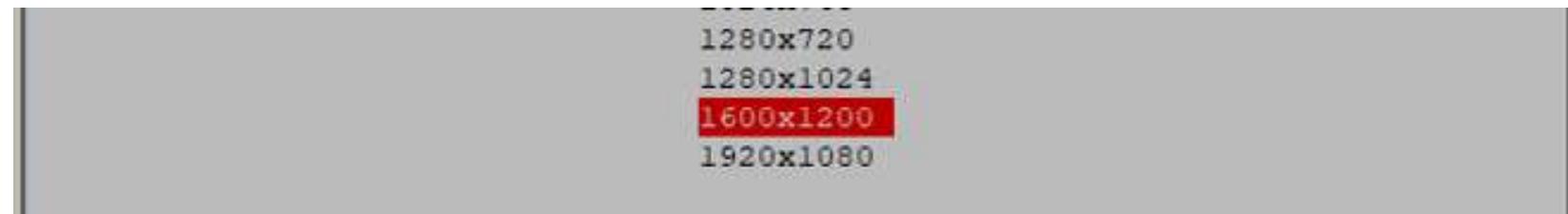


25. [2 Display Options]로 이동한다.

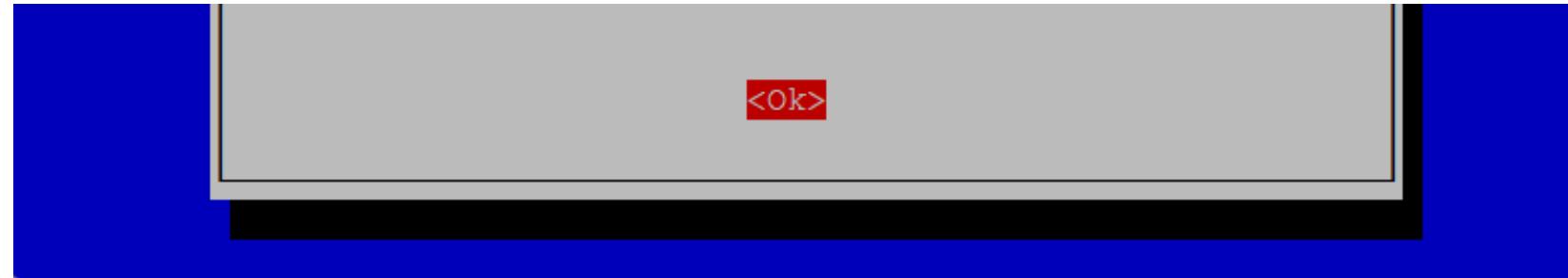
# 설정 GUI 환경 구성



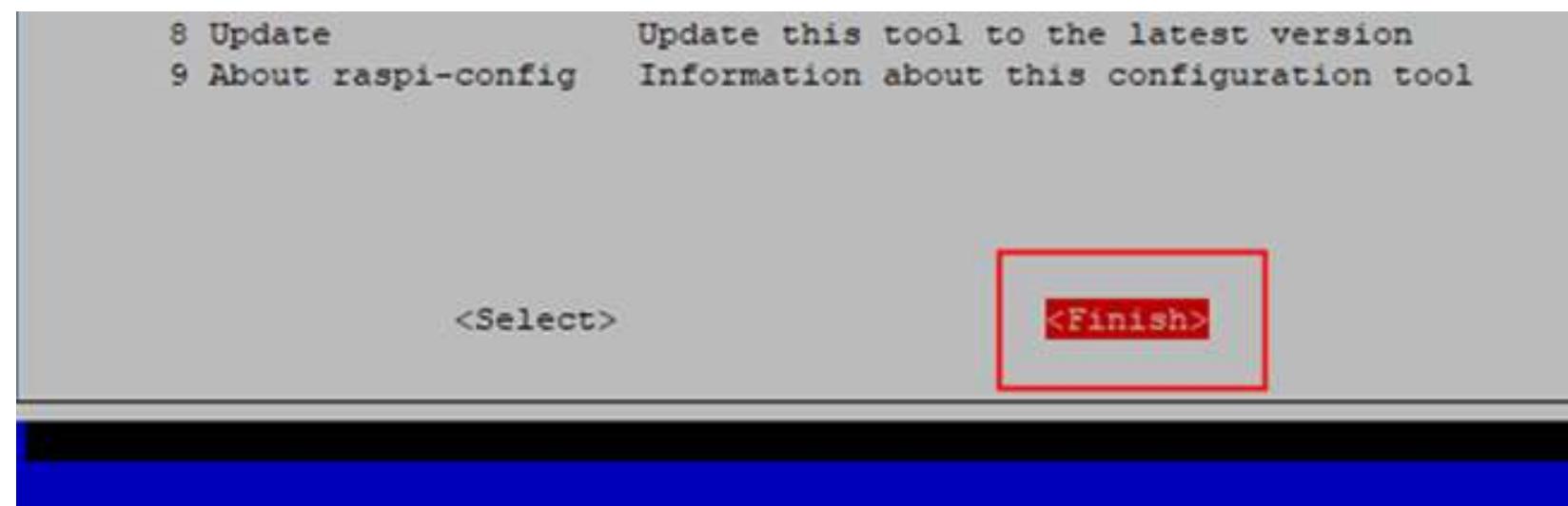
26. [D3 VNC Resolution] 으로 이동한다.



27. 화면비를 1600x 1200 으로 선택한다.



28. <ok> 를 엔터키로 선택한다.



29. [탭] 키를 두번 눌러 <Finish>로 이동한 다음 엔터를 입력한다.



30. <Yes> 를 눌러 재부팅한다.

# VNC Viewer로 GUI 접속하기

연결된 장치:	1/8	
장치 이름	IP 주소	물리적 주소(MAC)
redpiggy	192.168.137.167	d8:3a:dd:9a:a5:f2

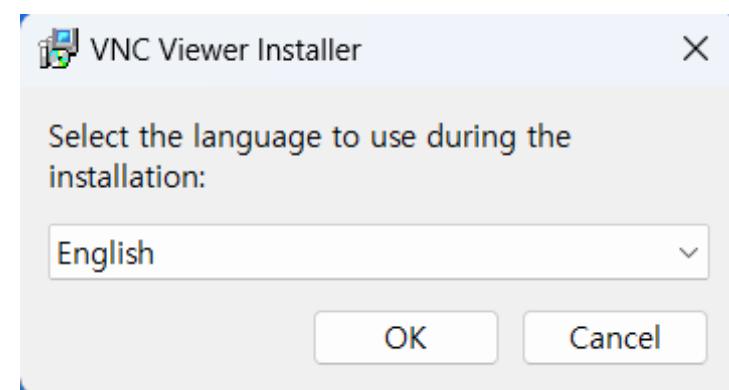
A screenshot of a Google search results page. The search query "real vnc download" is entered in the search bar. Below the search bar, there are filters for "전체", "동영상", "이미지", "뉴스", "쇼핑", and "더보기". The search results show a result from "RealVNC" with the URL <https://www.realvnc.com>. The result title is "Download VNC Viewer for Windows | VNC® Connect". Below the title, there is a message: "Thank you for downloading. Your download of VNC Viewer for has started. Now create an account to manage your devices and get a 14-day free trial to access all ...". It also mentions "macOS · Raspberry Pi · Linux".

A screenshot of the RealVNC website's download page for Windows. At the top, there are tabs for "Desktop" and "Mobile". Below the tabs, there are icons for "Windows", "macOS", "Linux", and "Raspberry Pi". Under the "Windows" section, there is a dropdown menu set to "EXE x86/x64" and a large yellow button labeled "Download RealVNC Viewer" with a download icon.

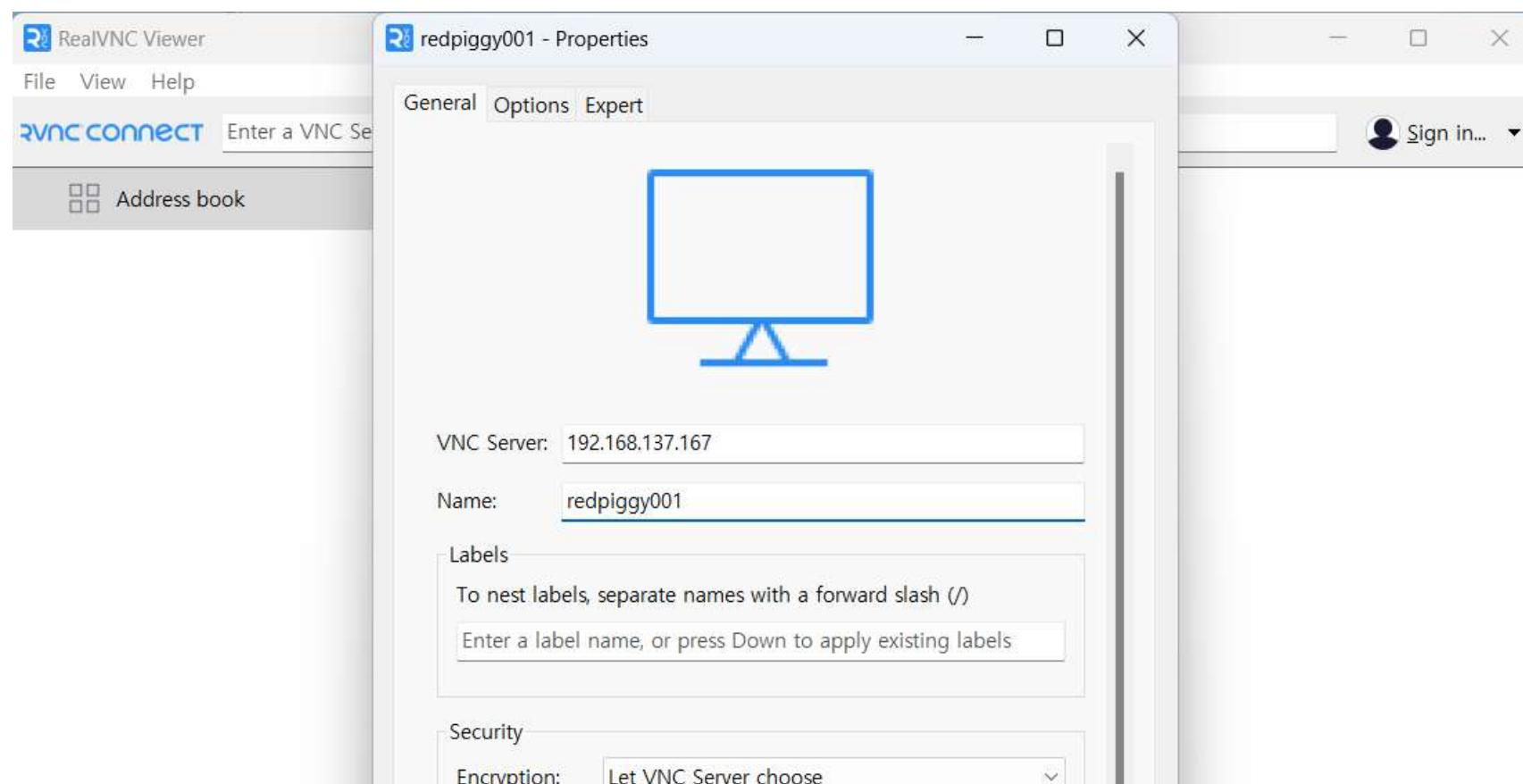
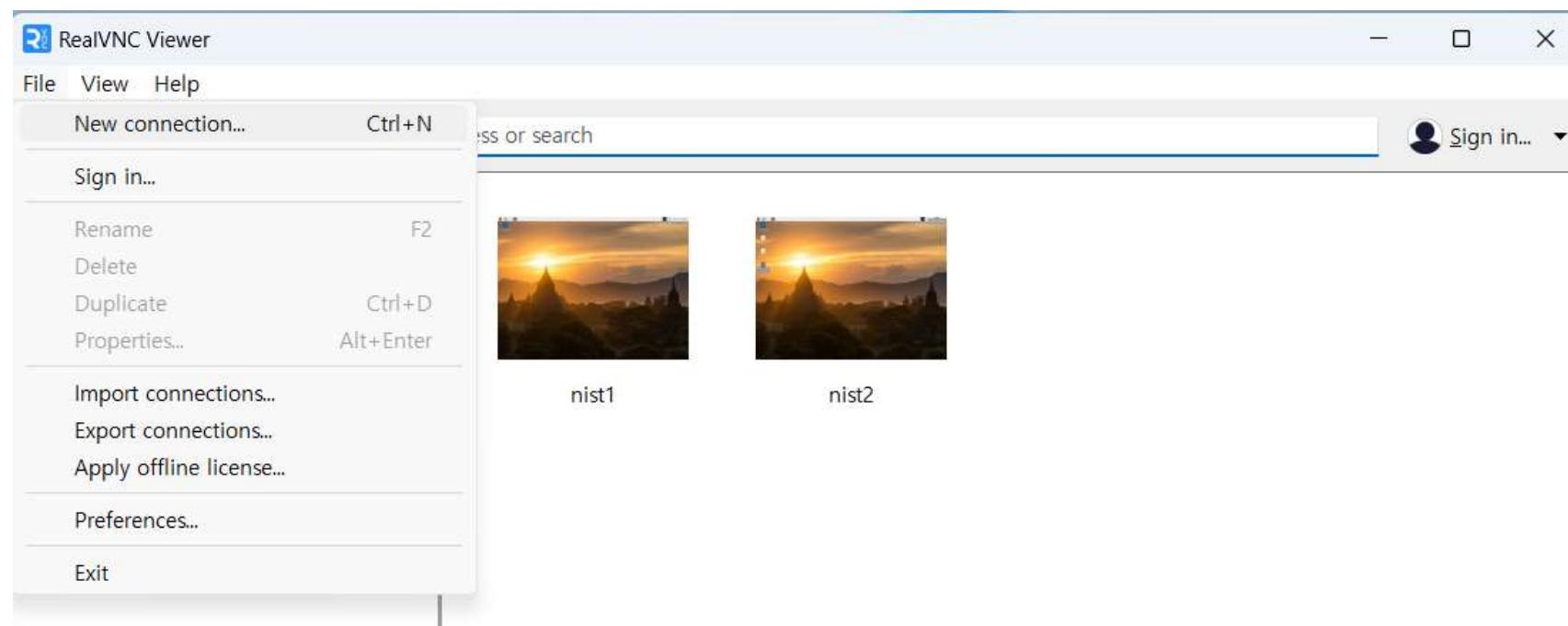
1. 모바일 핫스팟에서 라즈베리파이가 연결된 것을 확인한다. IP 주소는 변경될 수 있으므로 반드시 확인해야 한다.

2. 윈도우에서 vnc 접속프로그램을 설치한다. 구글에서 "real vnc download"를 검색 후 해당 사이트에 접속한다.

3. 윈도우를 선택 후 다운로드 받는다. 언어는 영어로 선택 후 [OK] 버튼을 눌러 설치한다.



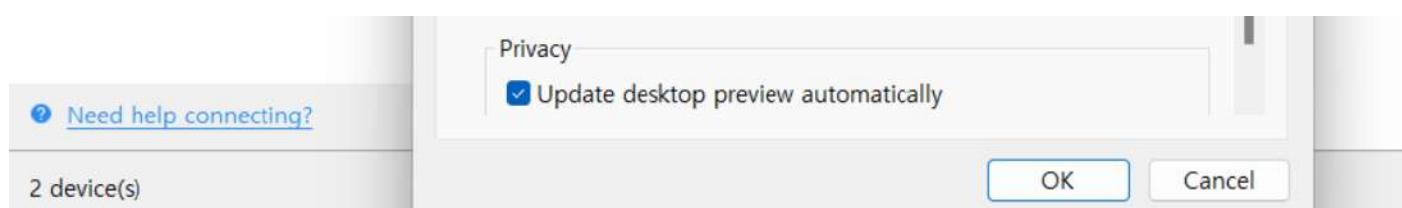
# VNC Viewer로 GUI 접속하기



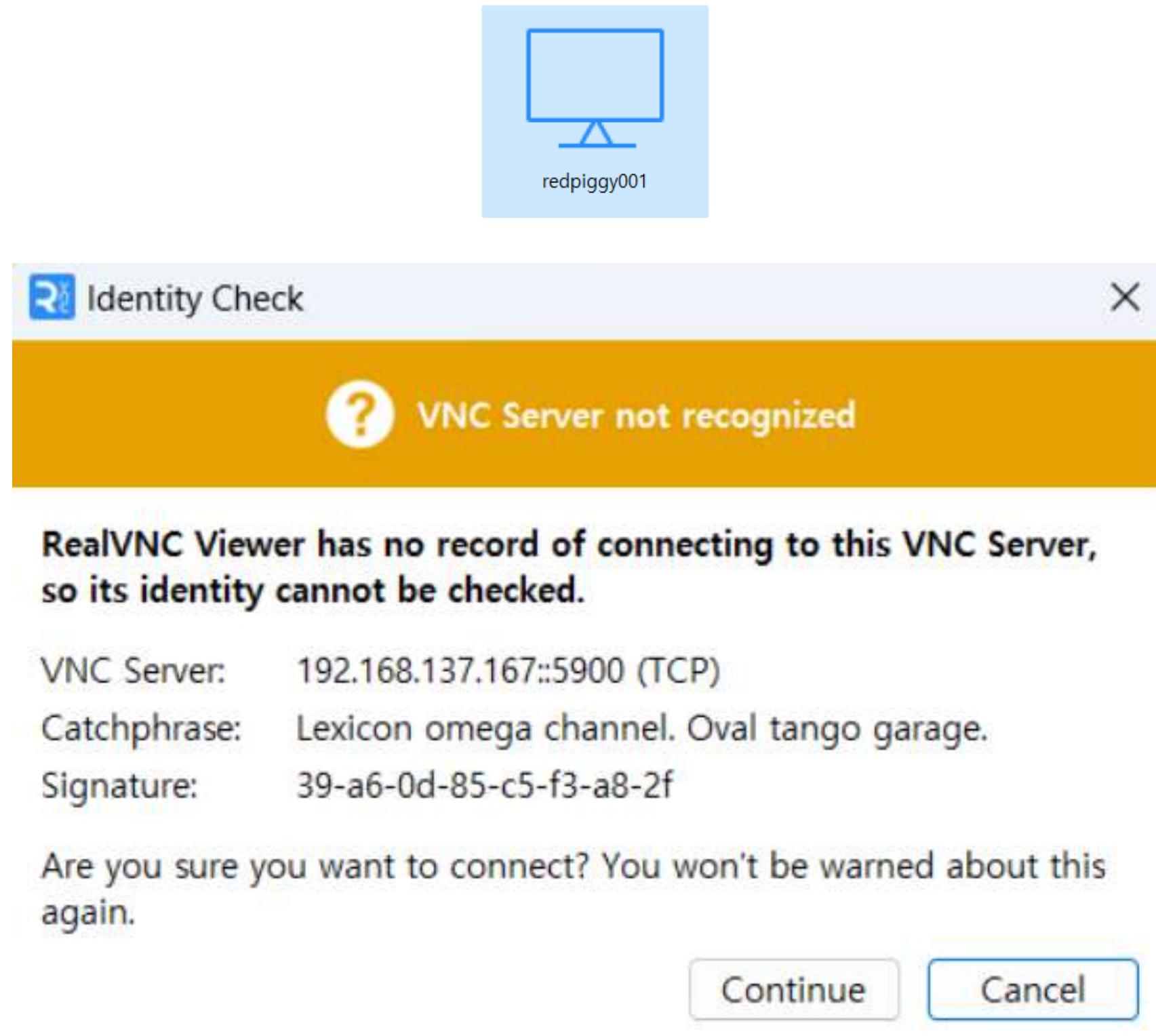
4. 설치된 VNC Viewer를 실행한다.  
회원가입 유도 팝업창이 뜨는 경우 [Cancel] 을  
눌러 취소한다.

5. [File] → [New connection..] 을 클릭한다.

6. VNC Server → 모바일 핫스팟에 연결된  
라즈베리파이의 IP 주소를 확인하여 입력한다.  
Name은 VNC Viewer에 표시될 이름을  
영문으로 입력한 후 [OK] 버튼을 클릭한다.

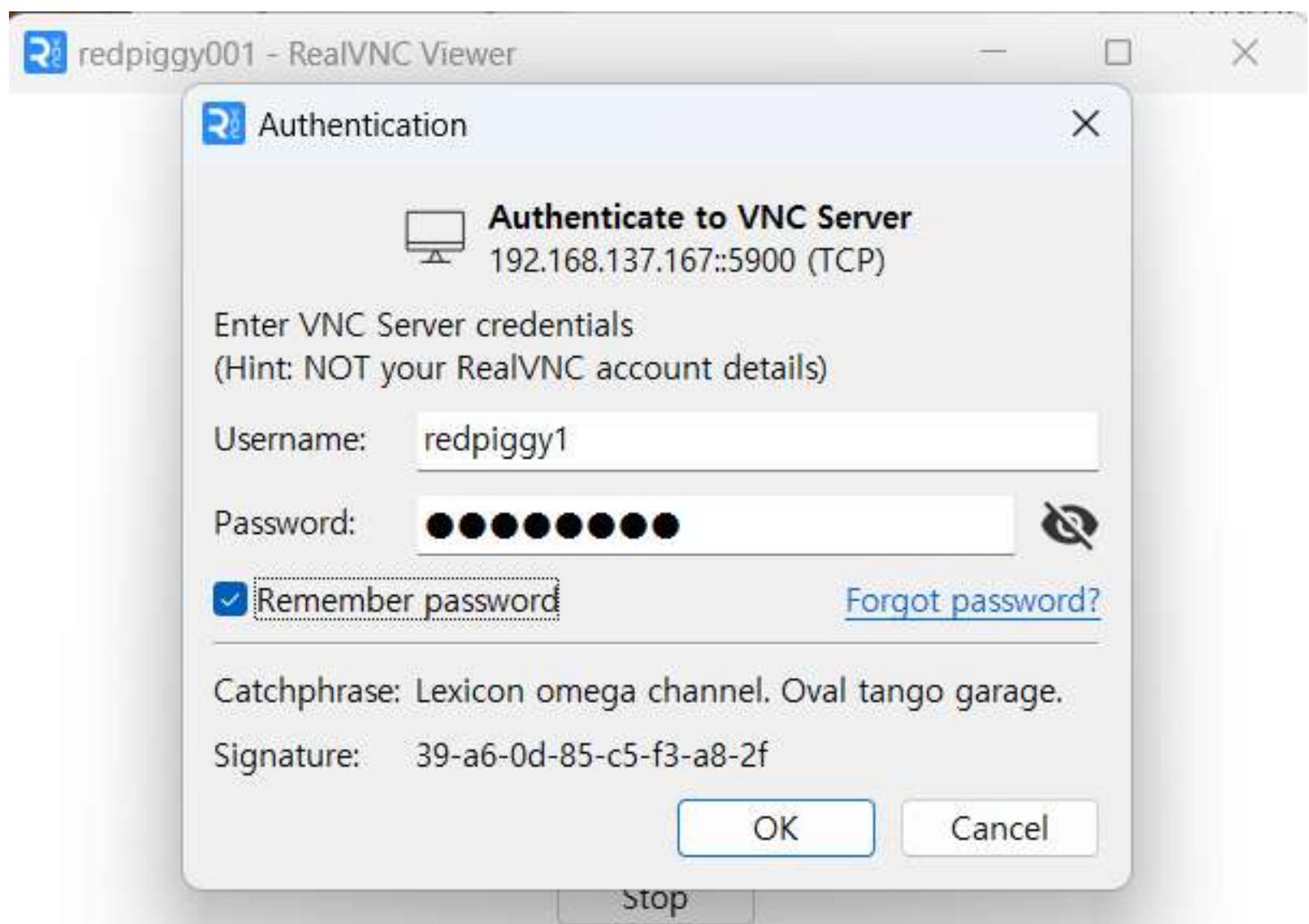


# VNC Viewer GUI 접속하기



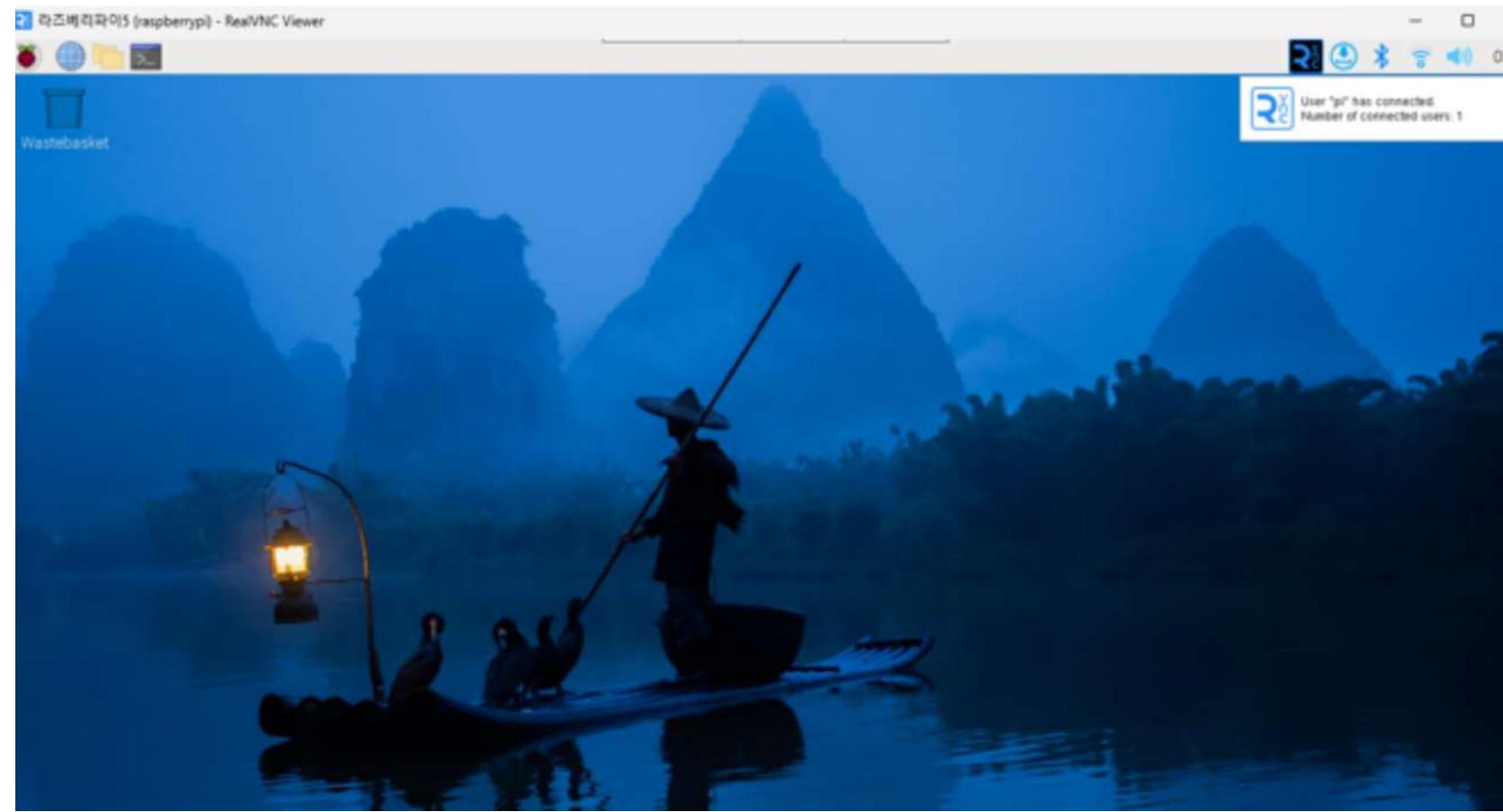
8. 처음 접속 시 연결 알림창은 [Continue] 버튼을 눌러 계속 진행한다.

7. 생성된 컴퓨터 모양의 아이콘을 더블클릭하여 접속한다.



9. Username과 Password는 라즈베리파이 이미지 설치 시 지정한 이름과 비밀번호를 입력하고 Remember password를 체크한 후 [OK] 버튼을 눌러 접속한다.

# VNC Viewer로 GUI 접속하기



10. 라즈베리파이에 GUI 환경으로 원격 접속하였다.  
VNC Viewer를 통해서 라즈베리파이를 직접  
조작하여 사용할 수 있다.

11. 라즈베리파이의 원격화면에 아이콘이 보이면  
원격으로 접속했음을 확인할 수 있다.



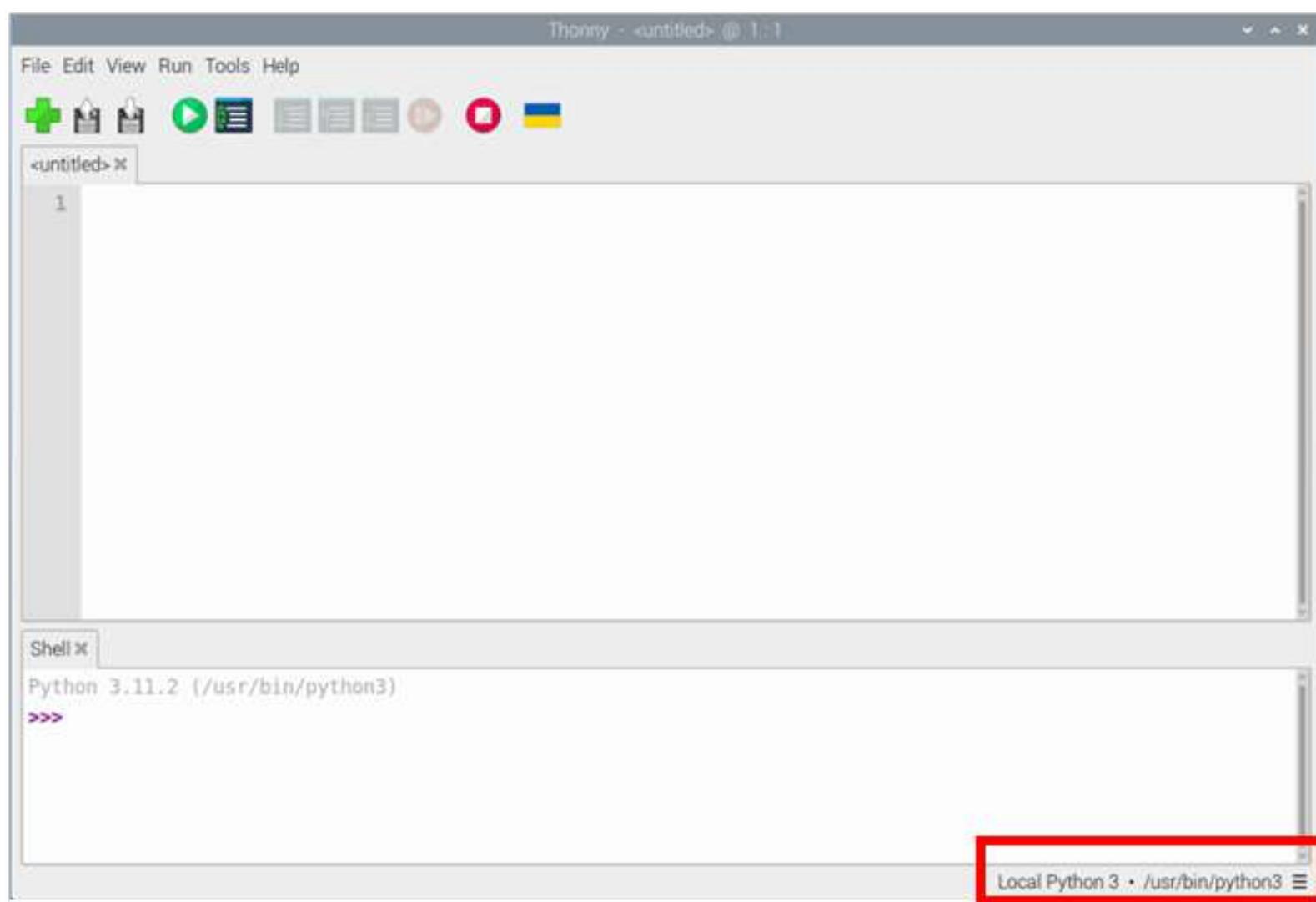
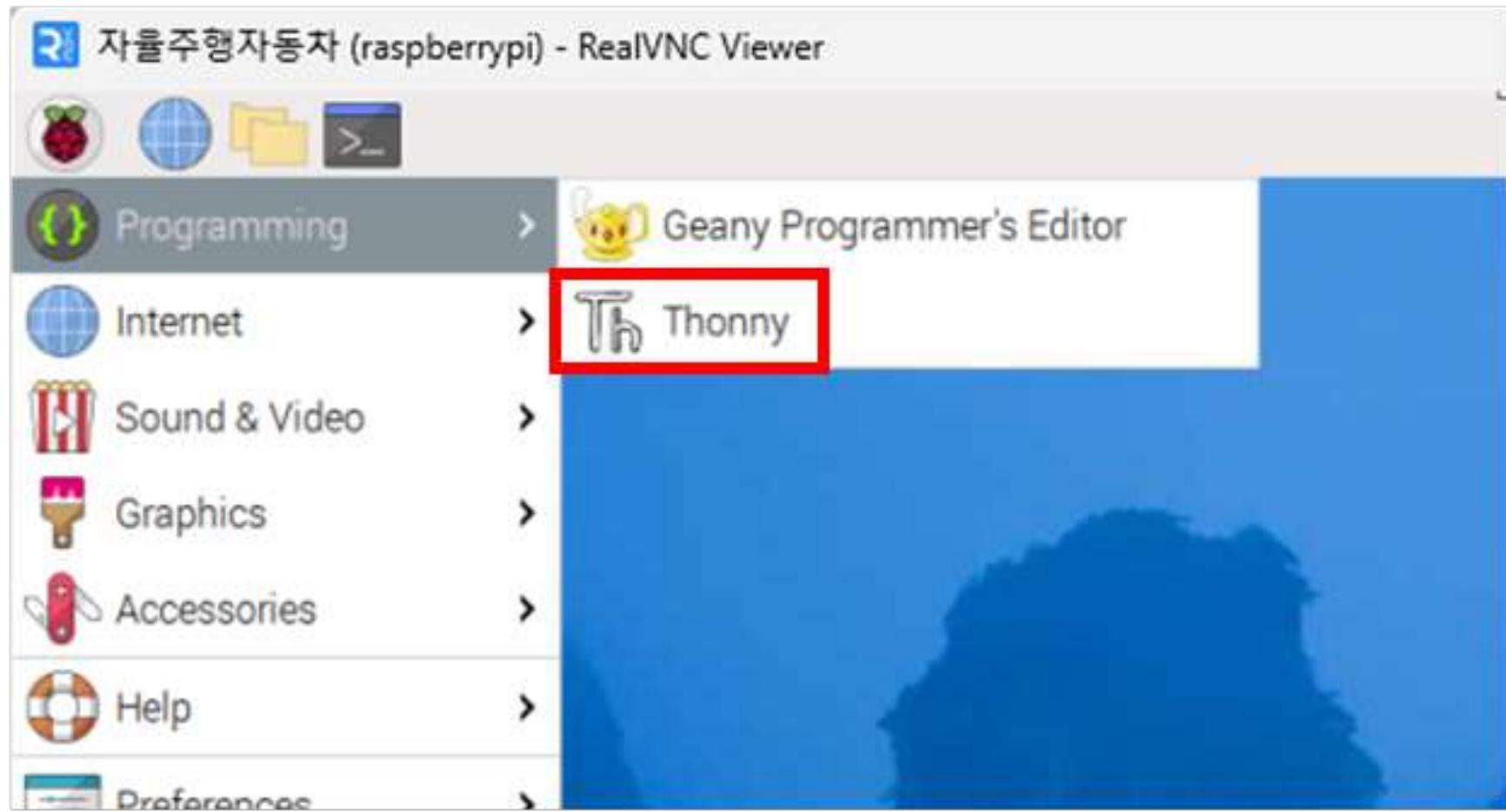
# AI 인공지능 자율주행 자동차

Part 4.

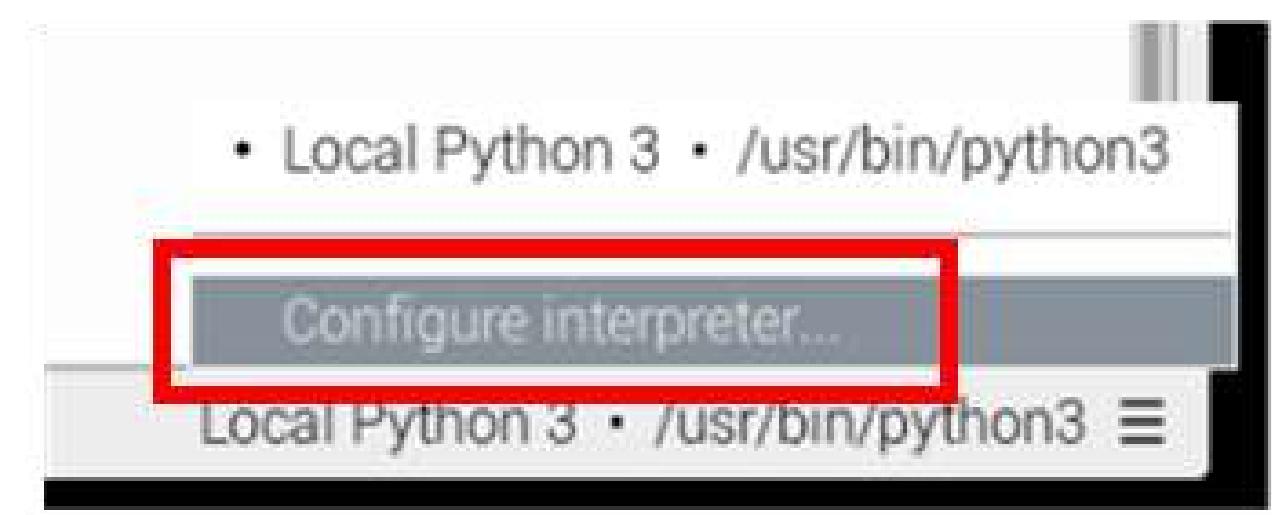
Raspberry Pi

개발환경 구성

# Raspberry Pi 개발환경구성



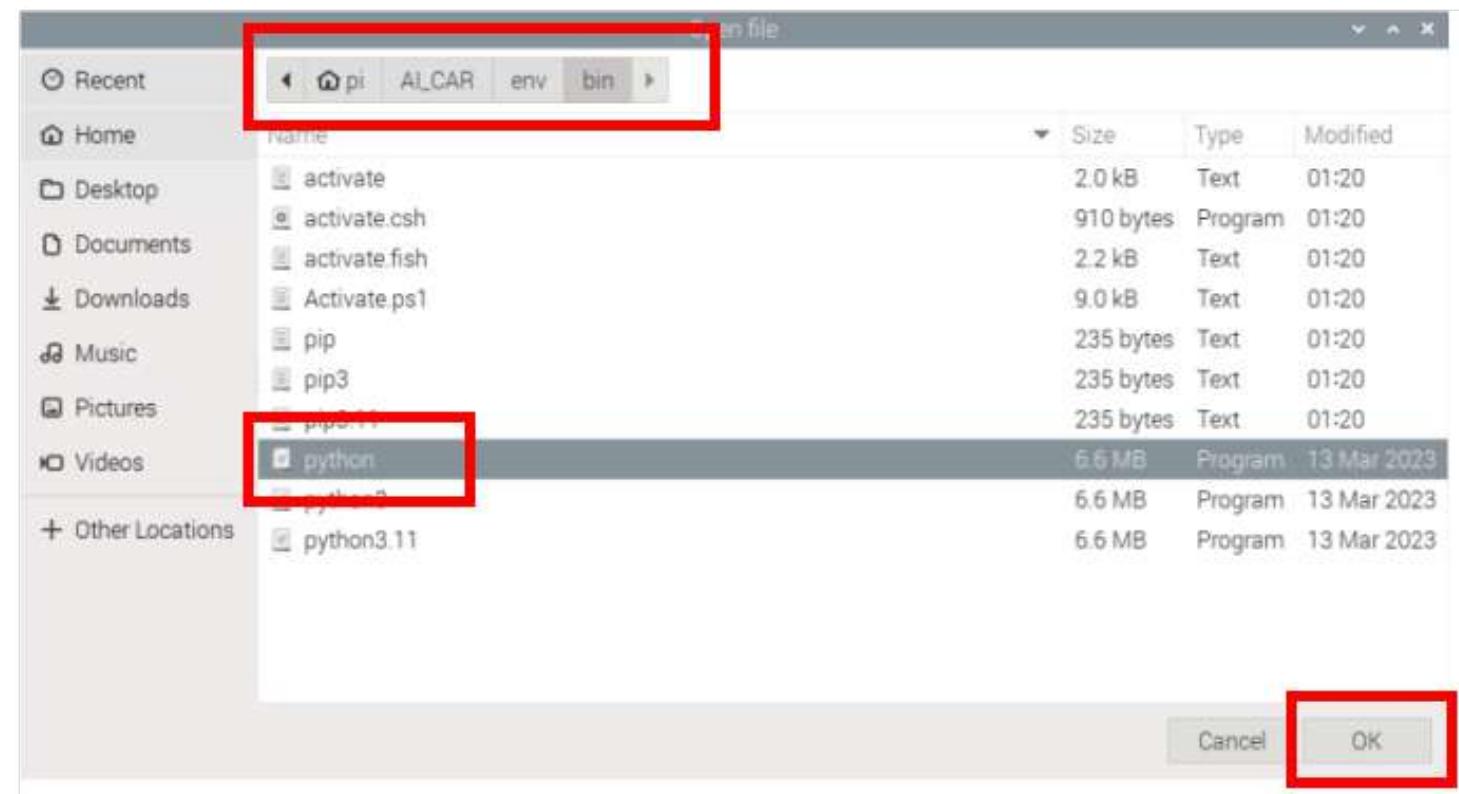
1. 라즈베리파이에서 자율주행자동차를 구현하기 위해서 파이썬 프로그램을 작성하고 실행하며 이러한 기능을 하는 에디터 툴이 바로 [Thonny Python IDE] 이다.  
라즈베리파이 아이콘을 클릭  
→ Programming  
→ [Thonny Python IDE] 메뉴 클릭
2. [Thonny Python IDE] 가 열리면 오른쪽 아래 Python3 부분을 클릭한다.  
파이썬의 실행파일을 설치한 가상환경으로 변경하기 위해서 Configure interpreter 를 클릭한다.



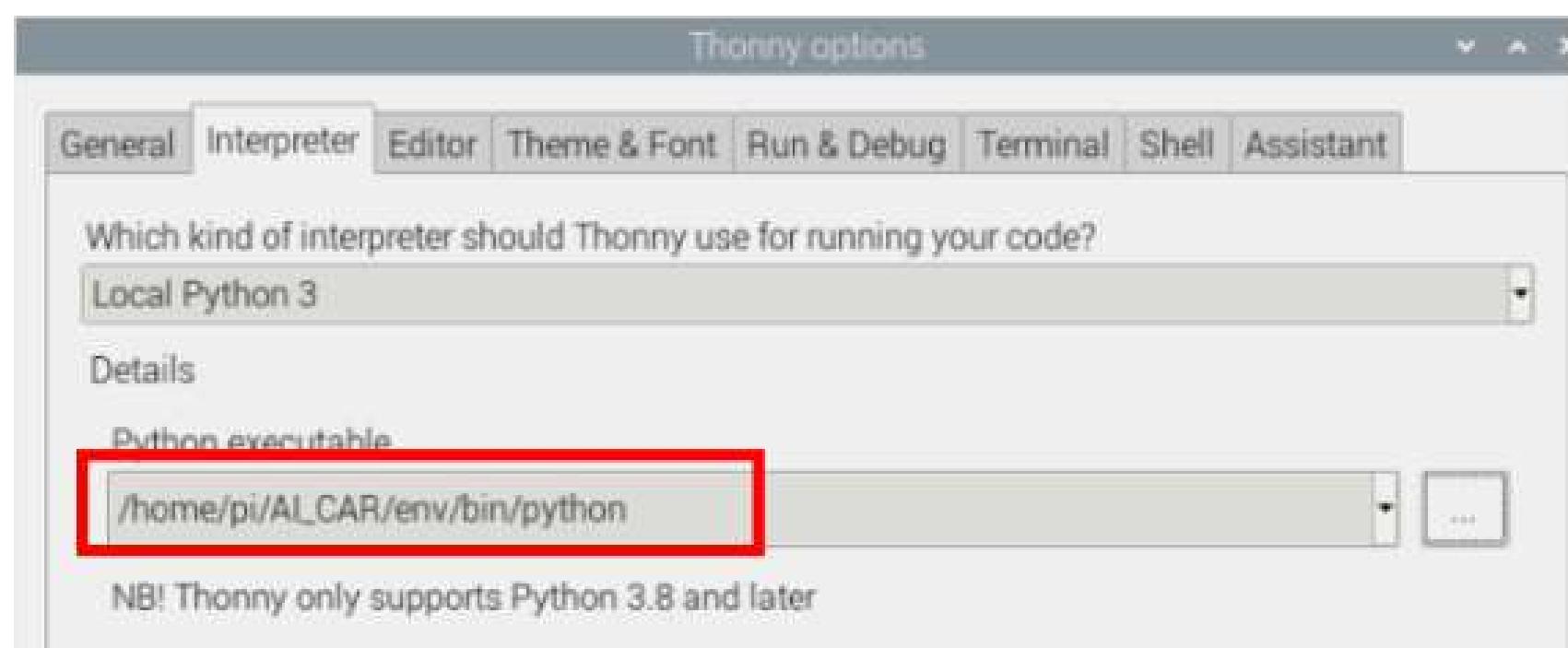
# Raspberry Pi 개발환경 구성



3. 파이썬을 변경하기 위해 [...] 을 클릭한다.



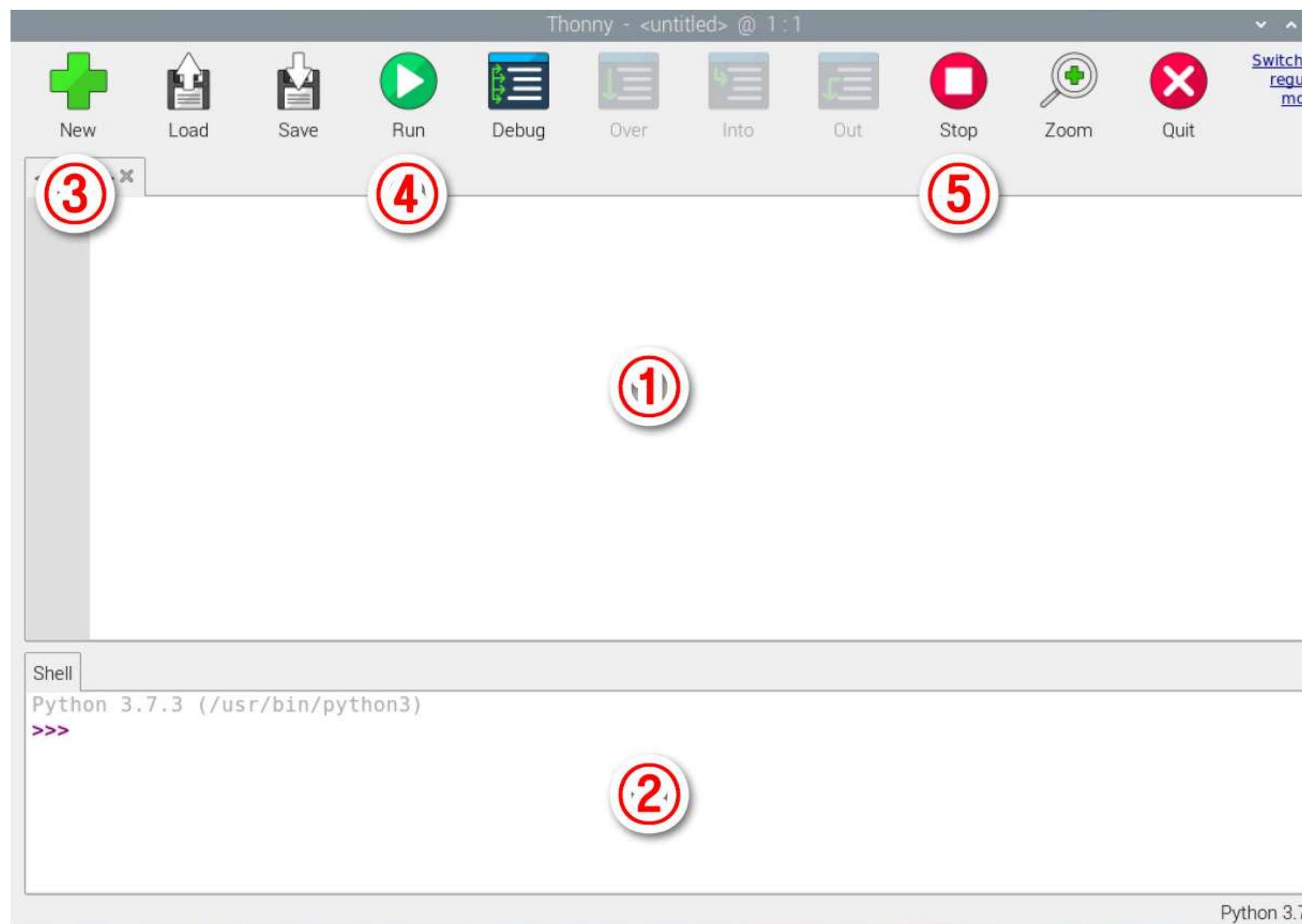
/home/py/AI\_CAR/env/bin 경로에 python를 선택 후 [OK]를 눌러 가상환경으로 구성된 파이썬을 선택한다. 여기서 /py/ 는 각자 팀별로 지정한 라즈베리파이 이름으로 만들어진 폴더명으로 변경되어 있다.



4. 가상환경으로 구성된 파이썬이 선택되었다.  
[OK] 를 눌러 설정을 마무리 한다.



# Raspberry Pi 개발환경 구성



## 5. Thonny IDE 의 구성

### ① [코드영역]

코드를 작성하는 부분

### ② [shell 영역]

코드가 실행될때 보여짐, 파이썬 직접 입력이 가능한 영역

### ③ [New]

새로운 파일 작성

### ④ [Run]

코드 실행

### ⑤ [Stop]

코드 멈춤

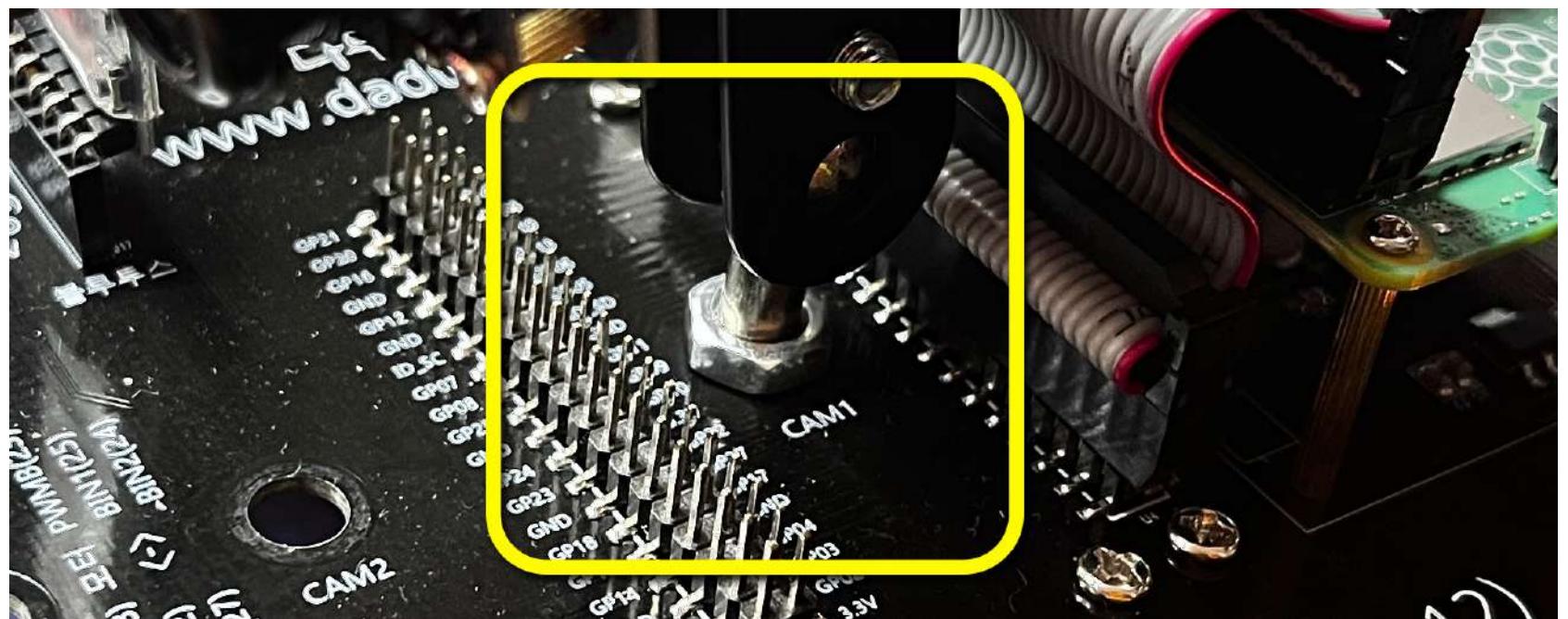
본 교육과정에서 사용되는 이미지 파일에는 이미 각각의 코드가 들어가 있습니다.  
이 코드를 사용하기 위해서는 불러온 후에 데이터를 불러오거나 결과값이 저장되는 폴더의 경로를  
반드시 각팀이 정한 라즈베리파이 이미지의 이름으로 변경해주셔야 합니다.  
이 부분이 번거로운 경우 이미지 파일 설정 시 이름은 pi로 지정하시면 됩니다.



# AI 인공지능 자율주행 자동차

Part 5.  
딥러닝 자율주행 만들기  
데이터 획득

# 딥러닝 자율주행 만들기 - 데이터 확보



1. OpenCV 를 이용하여 자율주행 자동차의 데이터를 획득하기 위해서는 카메라의 위치를 [CAM1] 홀로 위치시킨다.

```
mycamera.py
01 from picamera2 import Picamera2 # required for camera module v3
02 import numpy as np
03
04 class MyPiCamera():
05
06     def __init__(self,width,height):
07         self.cap = Picamera2()
08
09         self.width = width
10         self.height = height
11         self.is_open = True
12
13     try:
```

2. AI\_CAR 폴더에 mycamera.py 코드를 불러온다.

이 코드는 라즈베리파이에서 기본제공되는 picamera2 라이브러리를 이용해서 opencv를 사용할 수 있는 형식으로 변경해주는 코드이다.

RUN 실행 시 경고 메시지가 발생하나 동작에는 아무 문제가 없다.

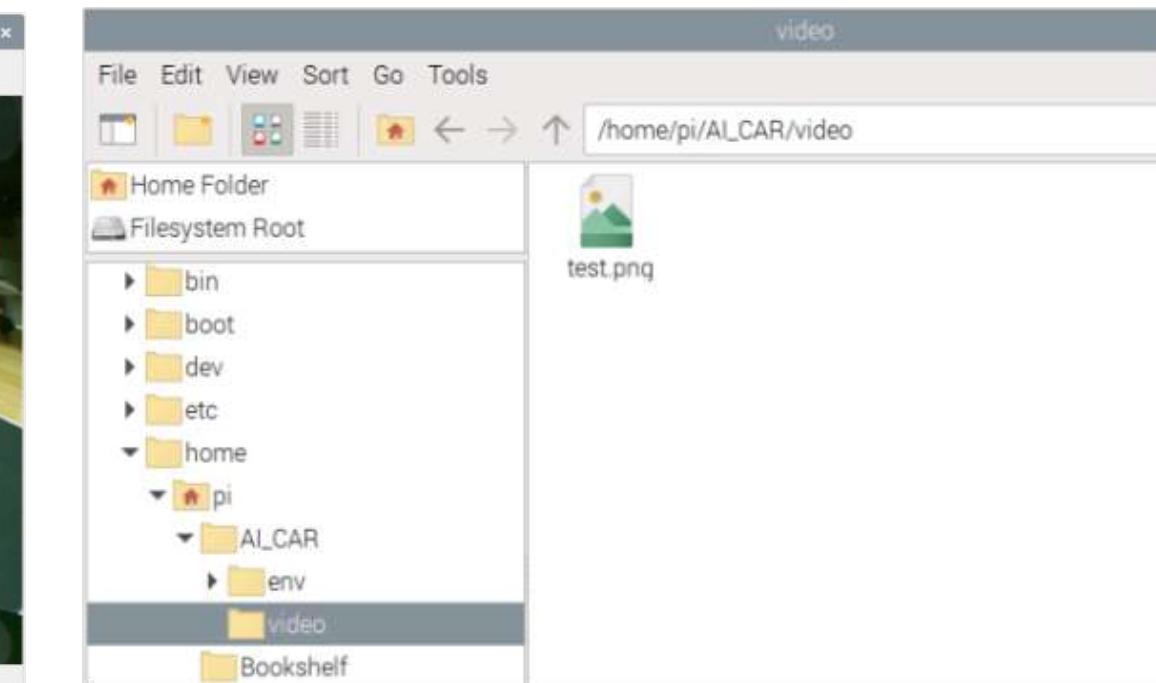
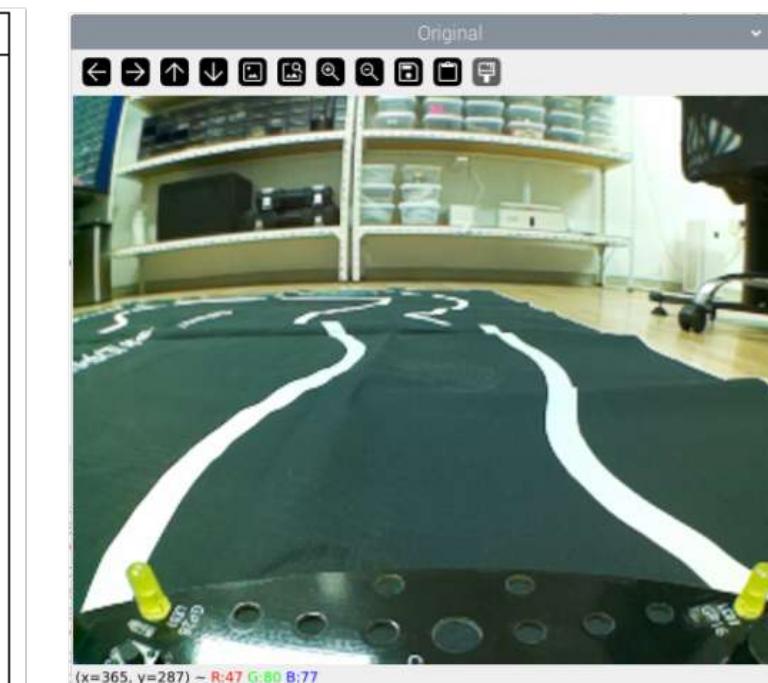
정상적으로 카메라가 출력되면 다음 코드로 넘어간다.

```
Shell x
5 to Unicam device /dev/media2 and ISP device /dev/media3
[0:30:19.090527091] [3383] INFO RPI pipeline_base.cpp:1144 Using configuration file '/usr/share/libcamera/pipeline/rpi/vc4/rpi_apps.yaml'
[0:30:19.7089010885] [3374] INFO Camera camera.cpp:1183 configuring streams: (0) 640x480-RGB888 (1)
640x480-SGBRG10_CSI2P
[0:30:19.709592999] [3383] INFO RPI vc4.cpp:611 Sensor: /base/soc/i2c0mux/i2c@1/ov5647@30 - Selected sensor format: 640x480-SGBRG10_1X10 - Selected unicam format: 640x480-pGAA
qt.qpa.xcb: QXcbConnection: XCB error: 148 (Unknown), sequence: 186, resource id: 0, major code: 14
0 (Unknown), minor code: 26
```

### 3. 이미지 저장 - 6\_3.py

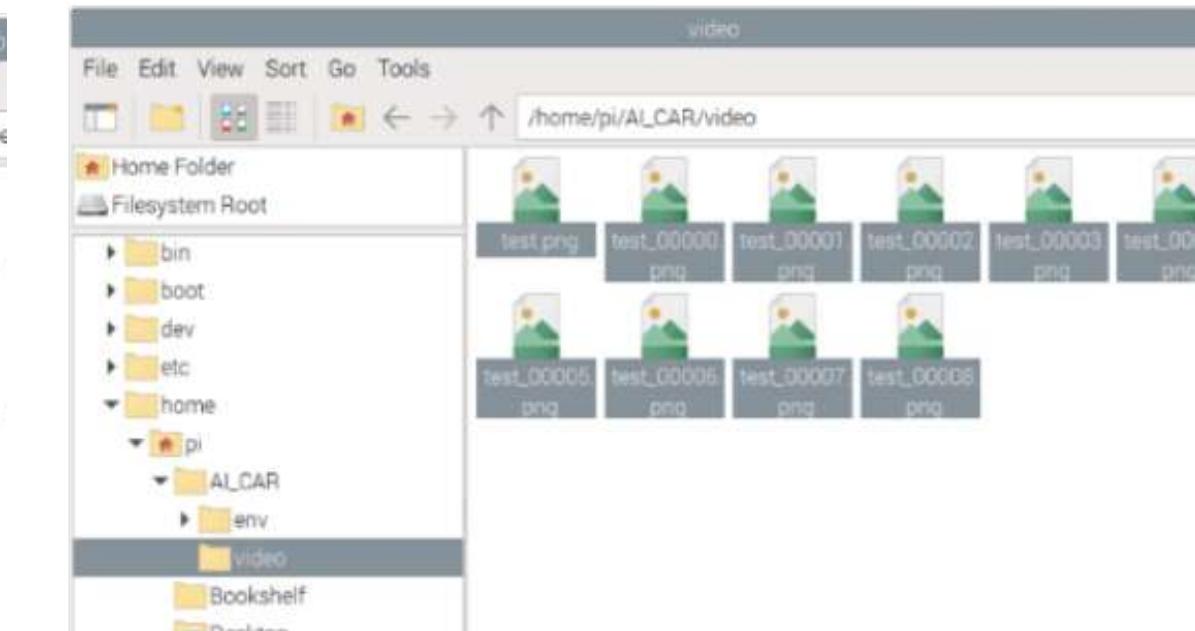
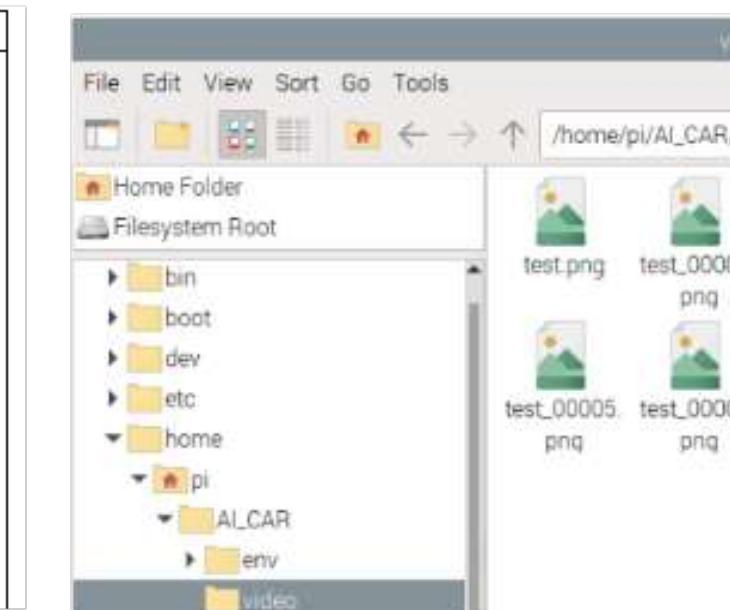
결과 값이 video 폴더에 test.png 파일로 생성된다. (1초마다 이미지가 갱신되어 사진이 저장)

```
6_3.py
01 import mycamera
02 import cv2
03 import time
04
05 def main():
06     camera = mycamera.MyPiCamera(640,480)
07
08     while( camera.isOpened() ):
09
10         keyValue = cv2.waitKey(10)
11
12         if keyValue == ord('q'):
13             break
14
15         image = camera.read()
16         image = cv2.flip(image,-1)
17         cv2.imshow('Original',image)
```



### 4. 새로운 사진을 새로운 이름으로 저장 - 6\_3\_2.py

```
6_3_2.py
01 import mycamera
02 import cv2
03 import time
04
05 def main():
06     camera = mycamera.MyPiCamera(640,480)
07     filepath = "/home/pi/AI_CAR/video/test"
08     i = 0
09
10     while( camera.isOpened() ):
11
12         keyValue = cv2.waitKey(10)
13
14         if keyValue == ord('q'):
15             break
```



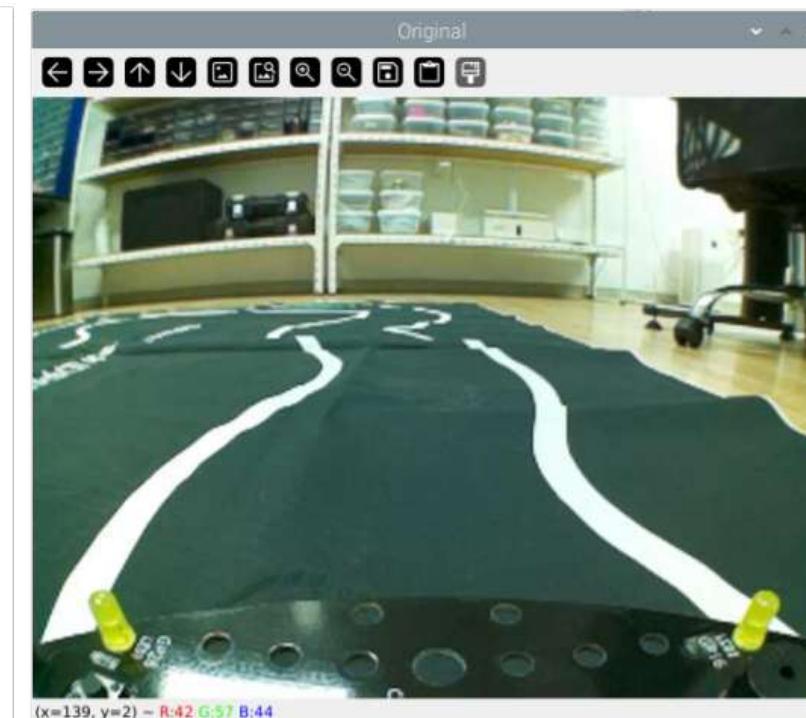
코드를 실행하면 test\_00000 부터 1씩 증가하여 사진이 저장된다.

데이터 삭제는 [컨트럴+a] 를 눌러 전체 선택 후 [Delete] 키를 눌러 휴지통으로 이동한다.

또는 [쉬프트+Delete]를 눌러 완전 삭제 할 수 있다.

## 5. 키보드 입력 조건문 실행 코드 - 6\_4.py

```
6_4.py
01 import mycamera
02 import cv2
03 import time
04
05 def main():
06     camera = mycamera.MyPiCamera(640,480)
07     filepath = "/home/pi/AI_CAR/video/train"
08     i = 0
09     carState = "stop"
10     while( camera.isOpened() ):
11
12         keyValue = cv2.waitKey(10)
13
14         if keyValue == ord('q'):
15             break
16         elif keyValue == 82:
17             print("go")
18             carState = "go"
19         elif keyValue == 84:
20             print("stop")
```

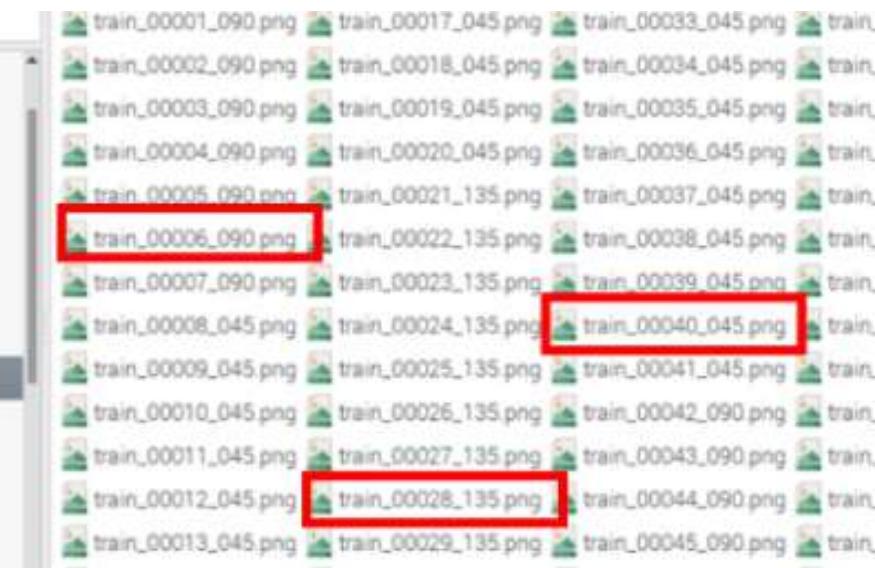
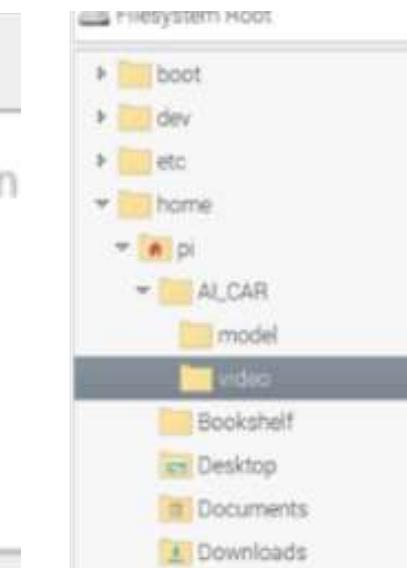


방향키 위를 누르면  
go 출력 후 G가 계속  
출력된다.

18~29번 라인  
딱 한번 실행되는 조건문  
42~47번 라인  
계속 실행되는 조건문

## 6. 사진을 항상 저장하는 코드 - 6\_4\_2.PY

```
6_4_2.py
01 import mycamera
02 import cv2
03 import time
04
05 def main():
06     camera = mycamera.MyPiCamera(640,480)
07     filepath = "/home/pi/AI_CAR/video/train"
08     i = 0
09     carState = "stop"
10     while( camera.isOpened() ):
11
12         keyValue = cv2.waitKey(10)
13
14         if keyValue == ord('q'):
```



6\_4.py 코드의 42~47번 라인에 사진을 저장하는 코드를 추가

학습모델을 만들기 위해 자동차가 직진할때 90도, 왼쪽으로 이동할때 45도, 오른쪽으로 이동할때 135도로 지정  
키보드 조작 시 사진이 저장되며 각도 값이 함께 파일에 저장된다.

```

6_4_3.py
001 import mycamera
002 import cv2
003 import time
004 from gpiozero import DigitalOutputDevice
005 from gpiozero import PWMOutputDevice
006
007 PWMA = PWMOutputDevice(18)
008 AIN1 = DigitalOutputDevice(22)
009 AIN2 = DigitalOutputDevice(27)
010
011 PWMB = PWMOutputDevice(23)
012 BIN1 = DigitalOutputDevice(25)
013 BIN2 = DigitalOutputDevice(24)
014
015 def motor_go(speed):
016     AIN1.value = 0

```



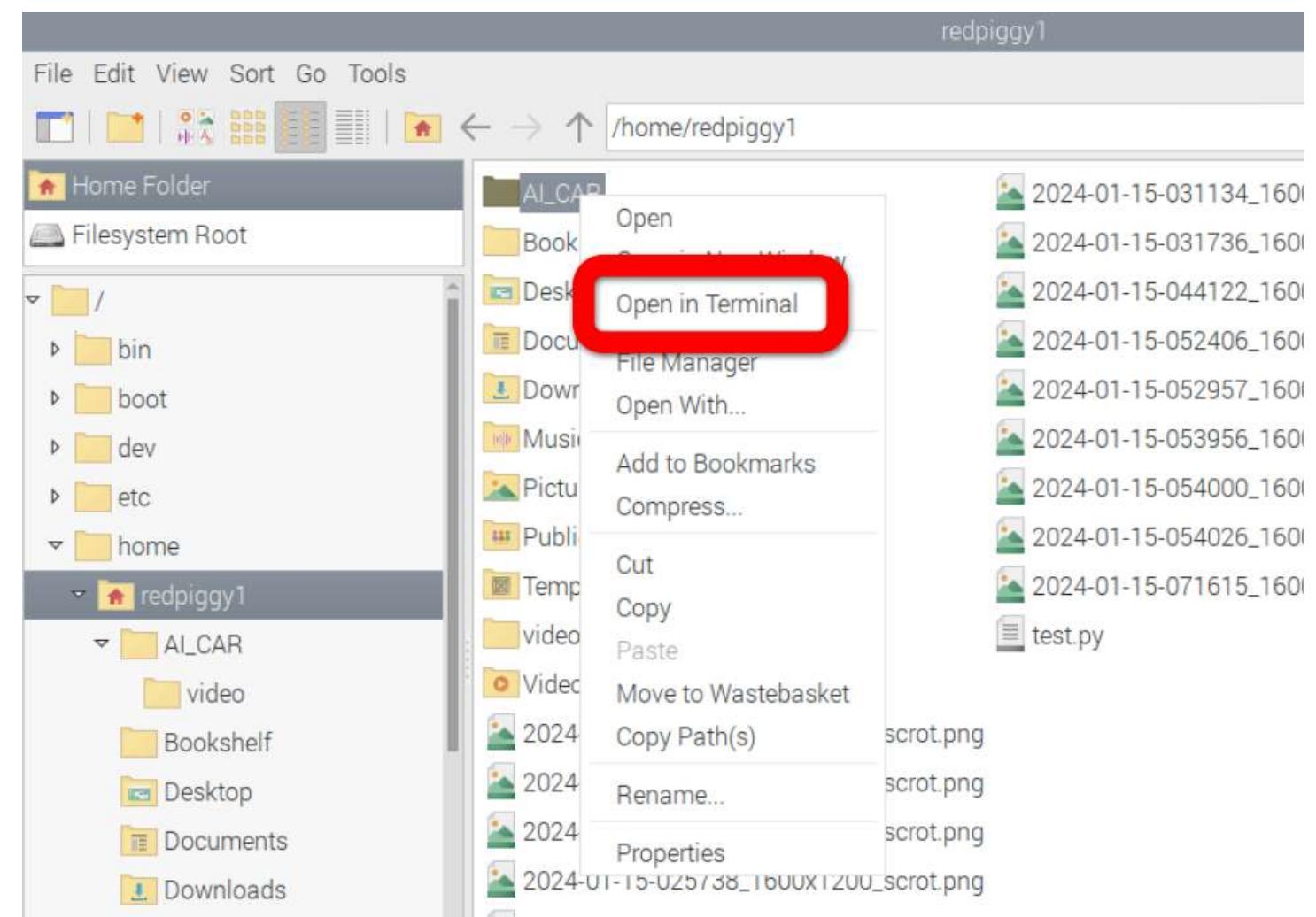
## 7. 기본 데이터 학습 코드 - 6\_4\_3.py

- 1) 데이터 학습은 정방향과 역방향을 각각 모두 2~3회 (총 4~6회) 회전하여 데이터를 학습
- 2) 데이터 학습은 PC의 화면을 보고 조종
- 3) 조종 중 키보드는 방향을 바꿀때만 조작

학습데이터의 양보다 중요한 것은 라인의 가운데로 정확하게 주행하는 것이 좋다. 조작이 많아지며 라인의 학습과정에서 좌우로 흔들림이 많을 수록 학습데이터를 통해 만들어진 주행 모듈 적용 시 차량의 주행 중 흔들림이 많이 발생하며, 차선의 이탈과 라인 침범과 같이 주행안정성이 떨어진다.

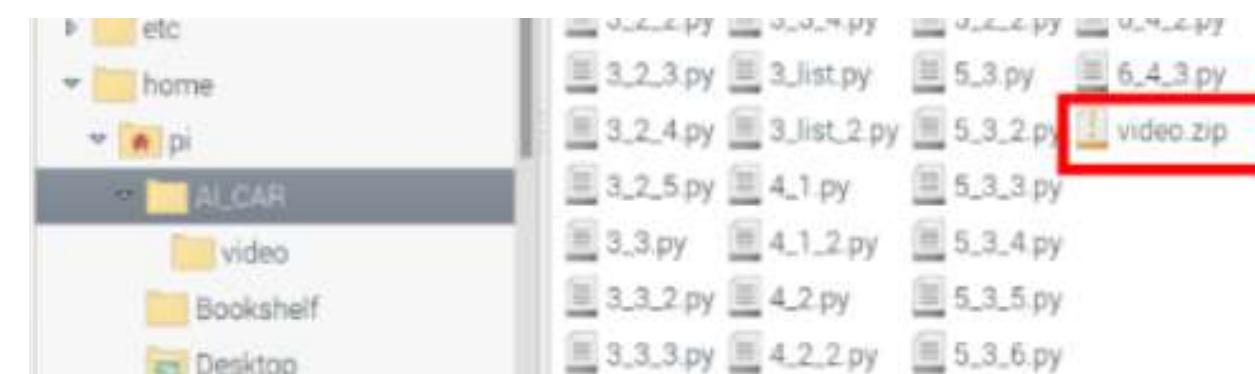
- 1) 학습데이터의 양보다 안정적인 주행 데이터가 이득
- 2) 주행 시 키의 조작은 최소화
- 3) 학습데이터의 취득 시 카메라의 각도에 따라 학습모듈의 정확도에도 차이가 발생

# 화상 데이터를 압축하여 PC로 이동하기

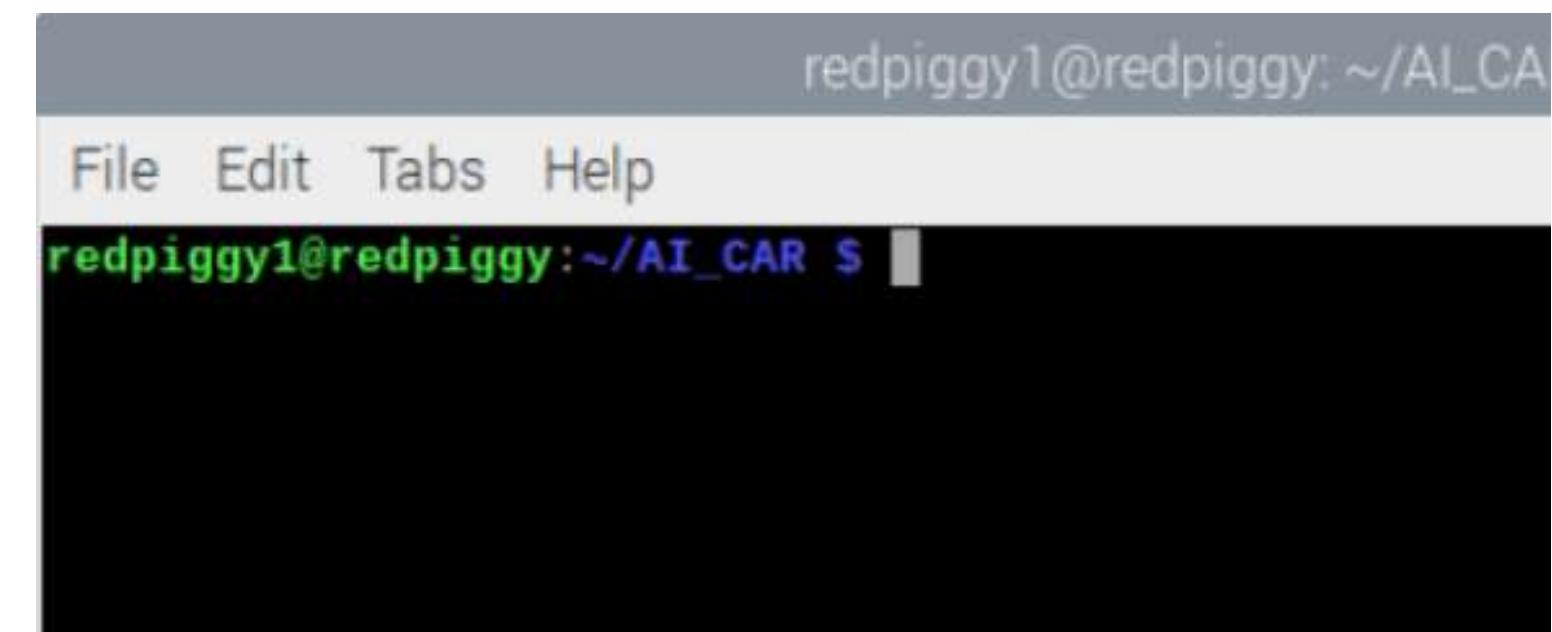


```
zip -r video.zip video
```

```
redpiggy1@redpiggy: ~/AI_CAR
File Edit Tabs Help
redpiggy1@redpiggy:~/AI_CAR$ zip -r video.zip video
```



1. [파일 매니저] 를 열어 [AI\_CAR] 폴더에 마우스 오른쪽 버튼을 클릭 후 [Open in Terminal] 을 클릭한다.

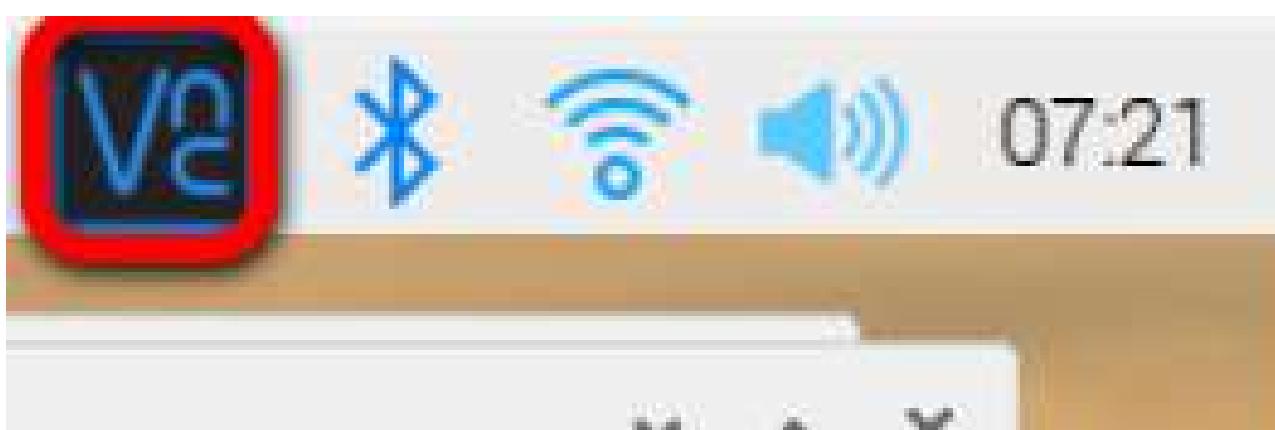


2. 터미널 창에 video 폴더를 video.zip 파일로 압축하기 위한 압축명령어를 입력한다.

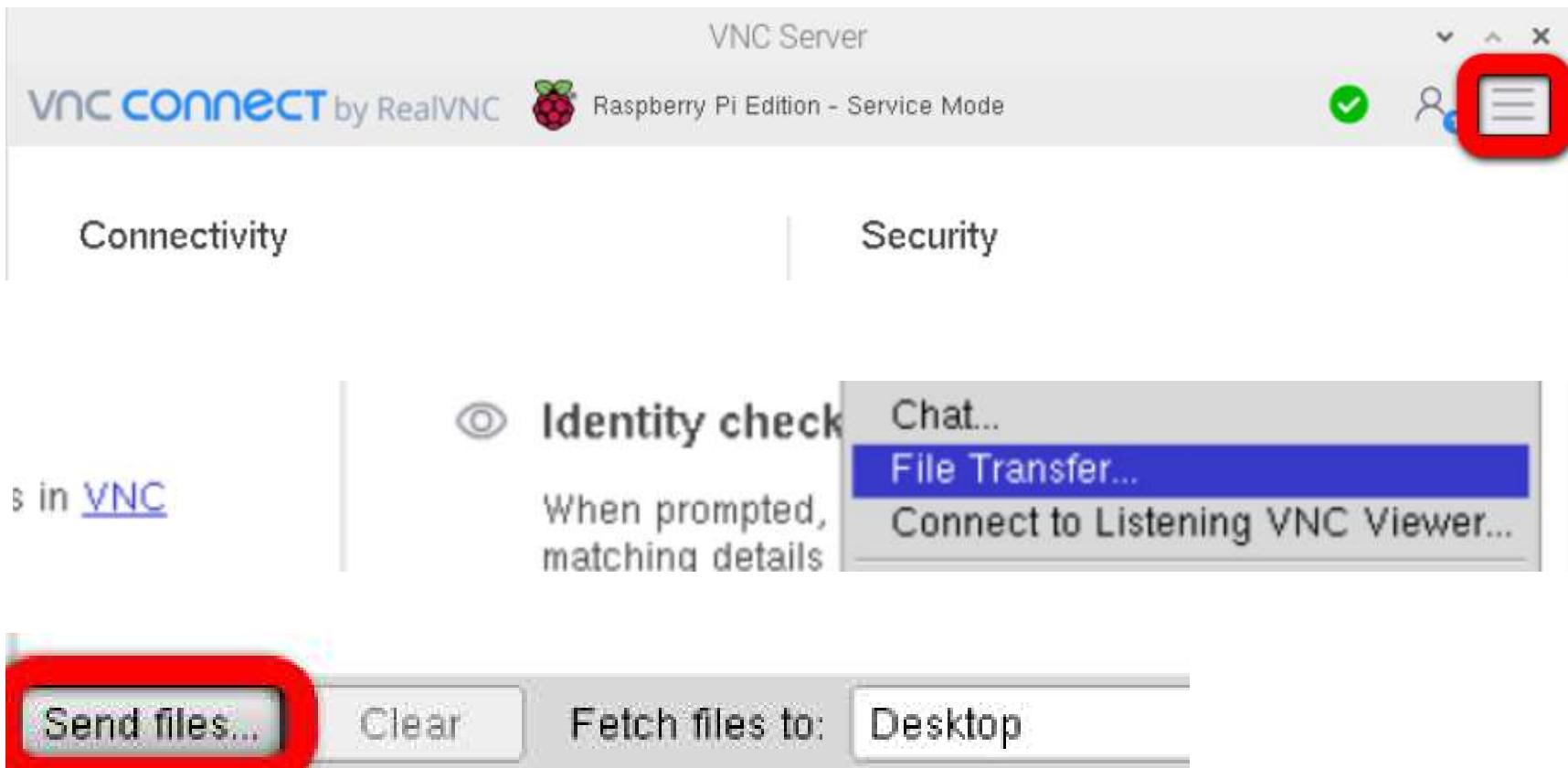
-r : 폴더내의 모든 파일을 압축하라는 옵션  
video 폴더 안의 모든 파일을 video.zip 이라는 이름으로 압축한다는 명령어

AI\_CAR 폴더에 video.zip 파일이 생성되었다.

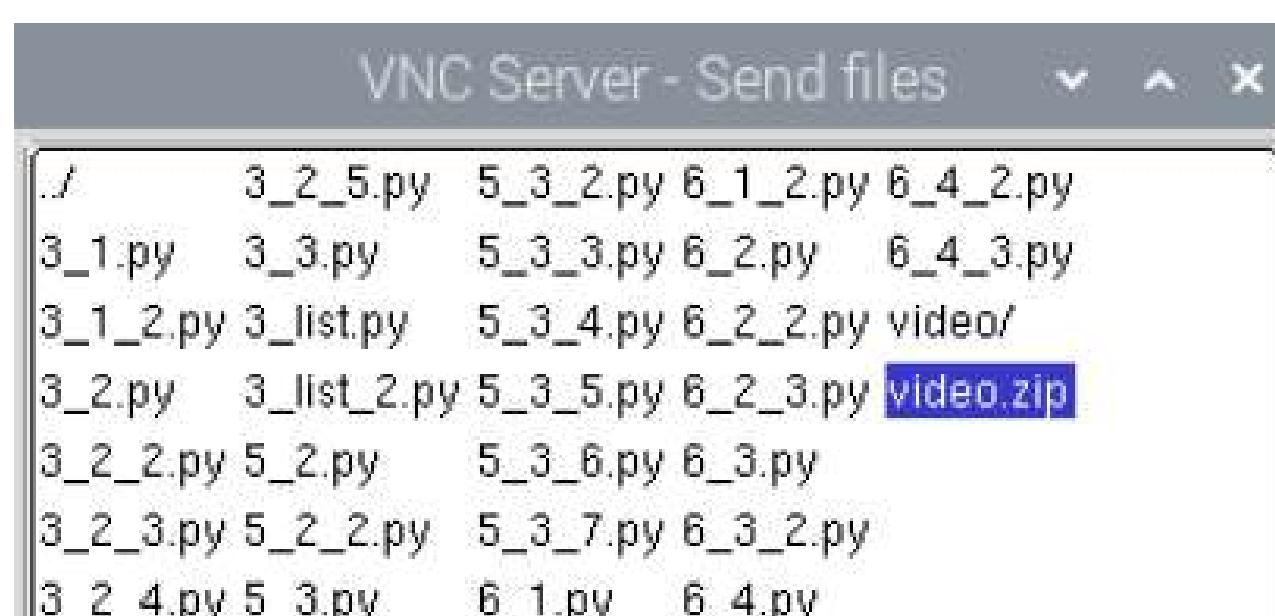
# 화면 데이터를 암축하여 PC로 이동하기



3. 라즈베리파이에는 GPU가 없기 때문에 이미지 학습에 시간이 너무 오래 걸린다. 그래서 PC로 데이터를 옮겨와서 학습한다.  
VNC 아이콘을 클릭한다.



4. 오른쪽 상단의 [3선 아이콘 (더보기)] 버튼을 클릭한다.

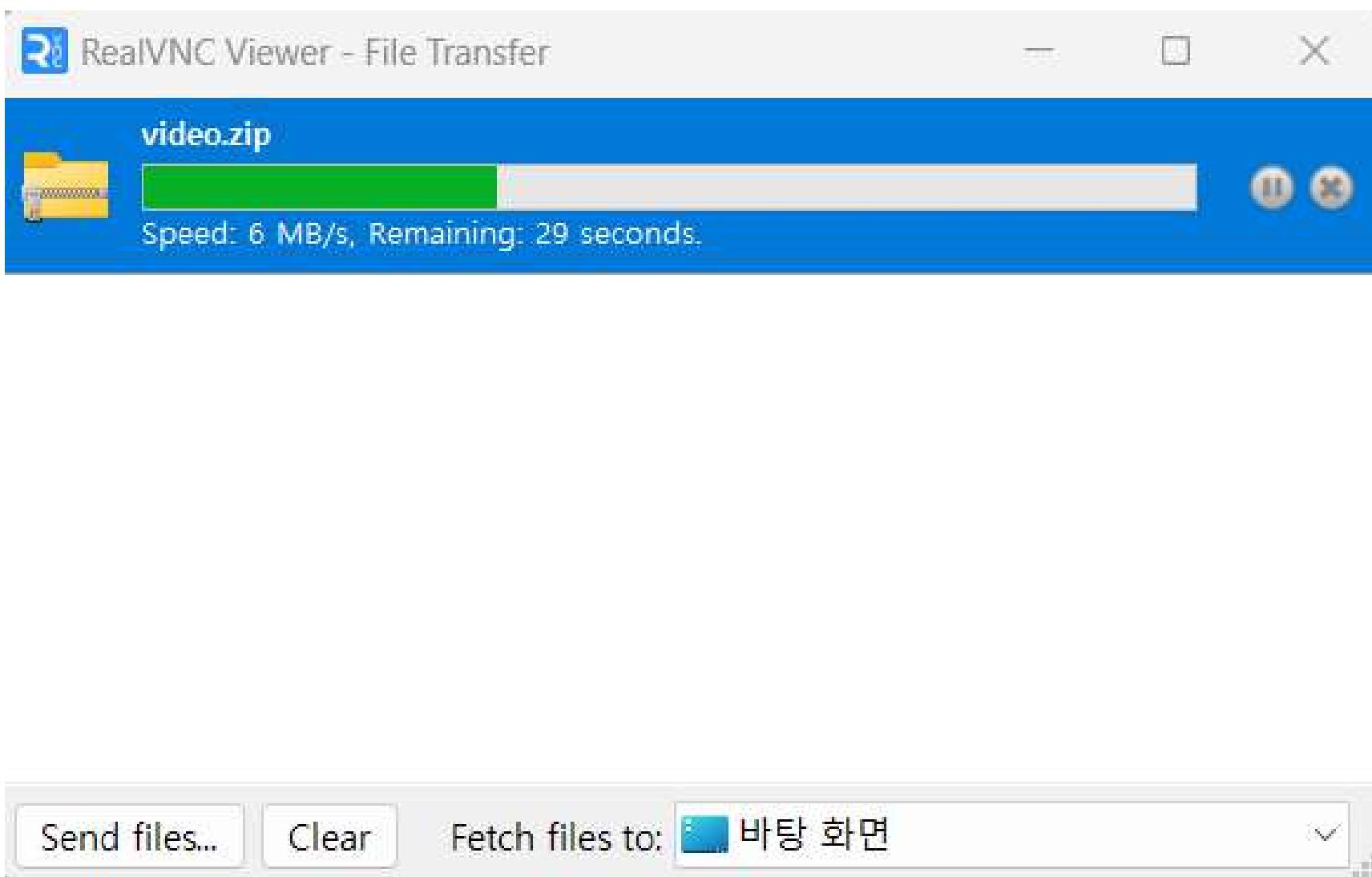


5. [File Transfer...] 버튼을 클릭한다.

6. [Send file...] 버튼을 클릭한다.

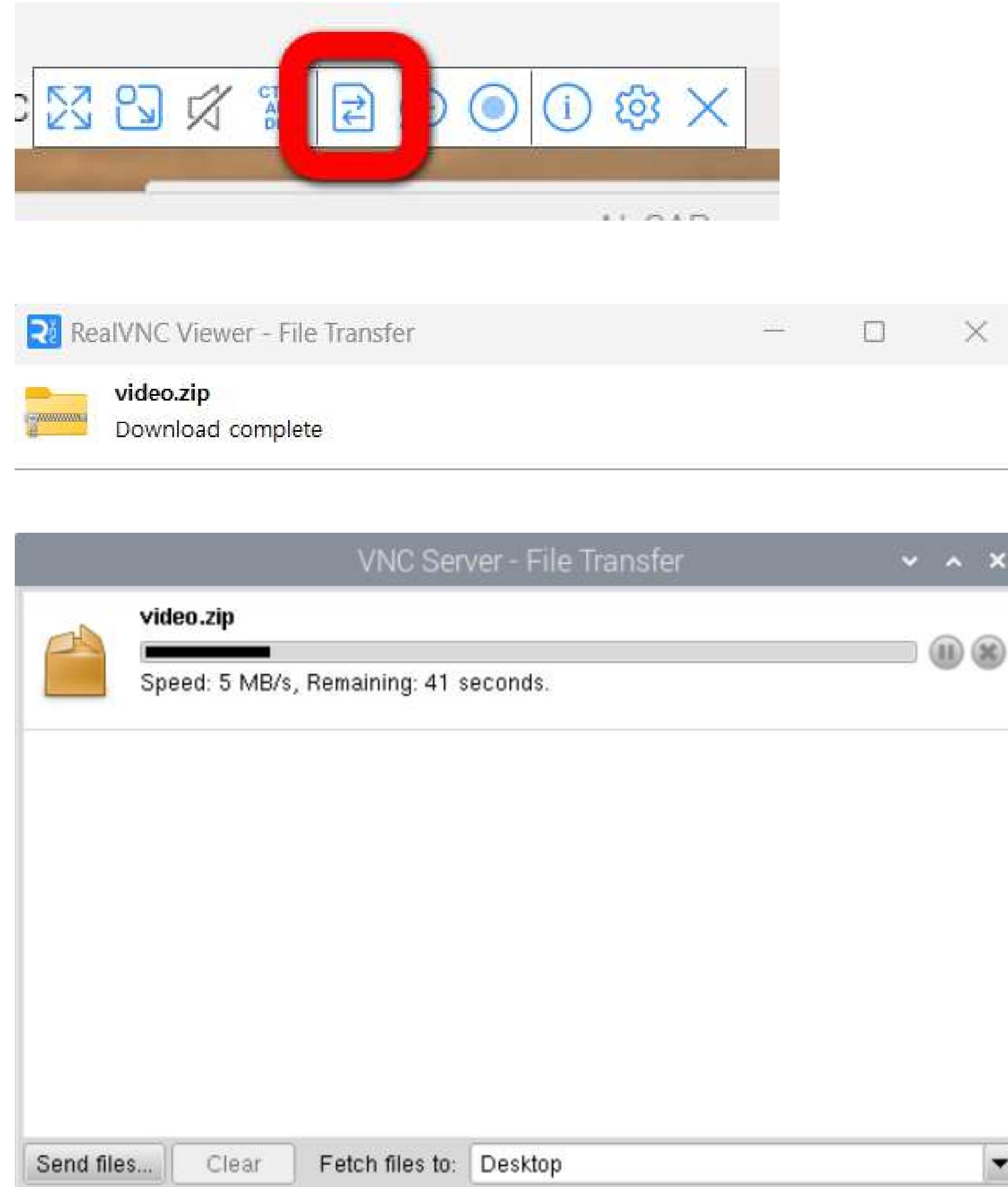
7. AI\_CAR 폴더에서 video.zip 파일을 선택하여 [OK] 버튼을 눌러 PC로 전송한다.

# 화상 데이터를 암축하여 PC로 이동하기



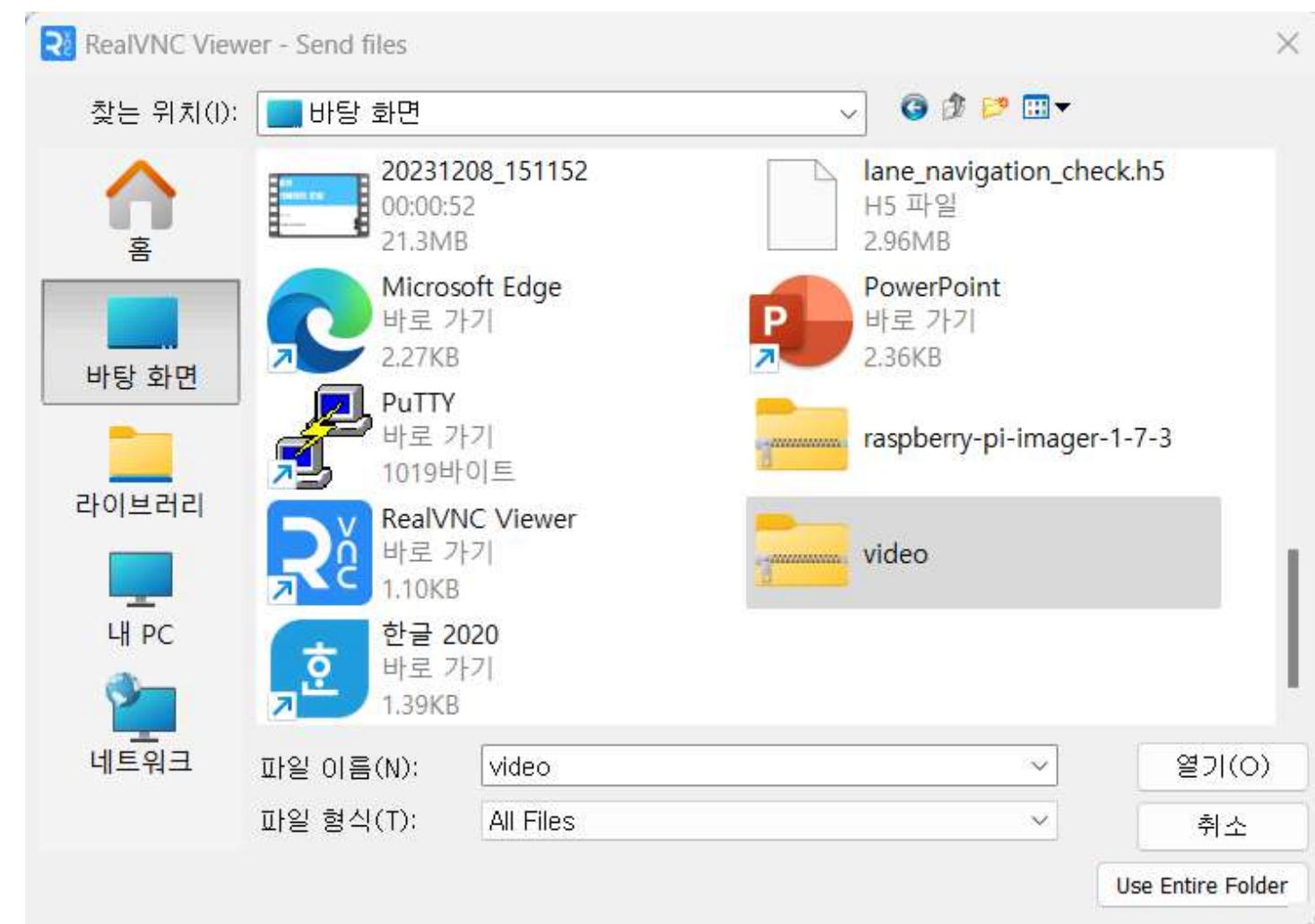
- video.zip 파일이 원도우의 바탕화면으로 전송된다.
- 전송된 파일은 원도우 바탕화면에서 찾아본다.

# PC데이터를 라즈베리파이로 전송하기

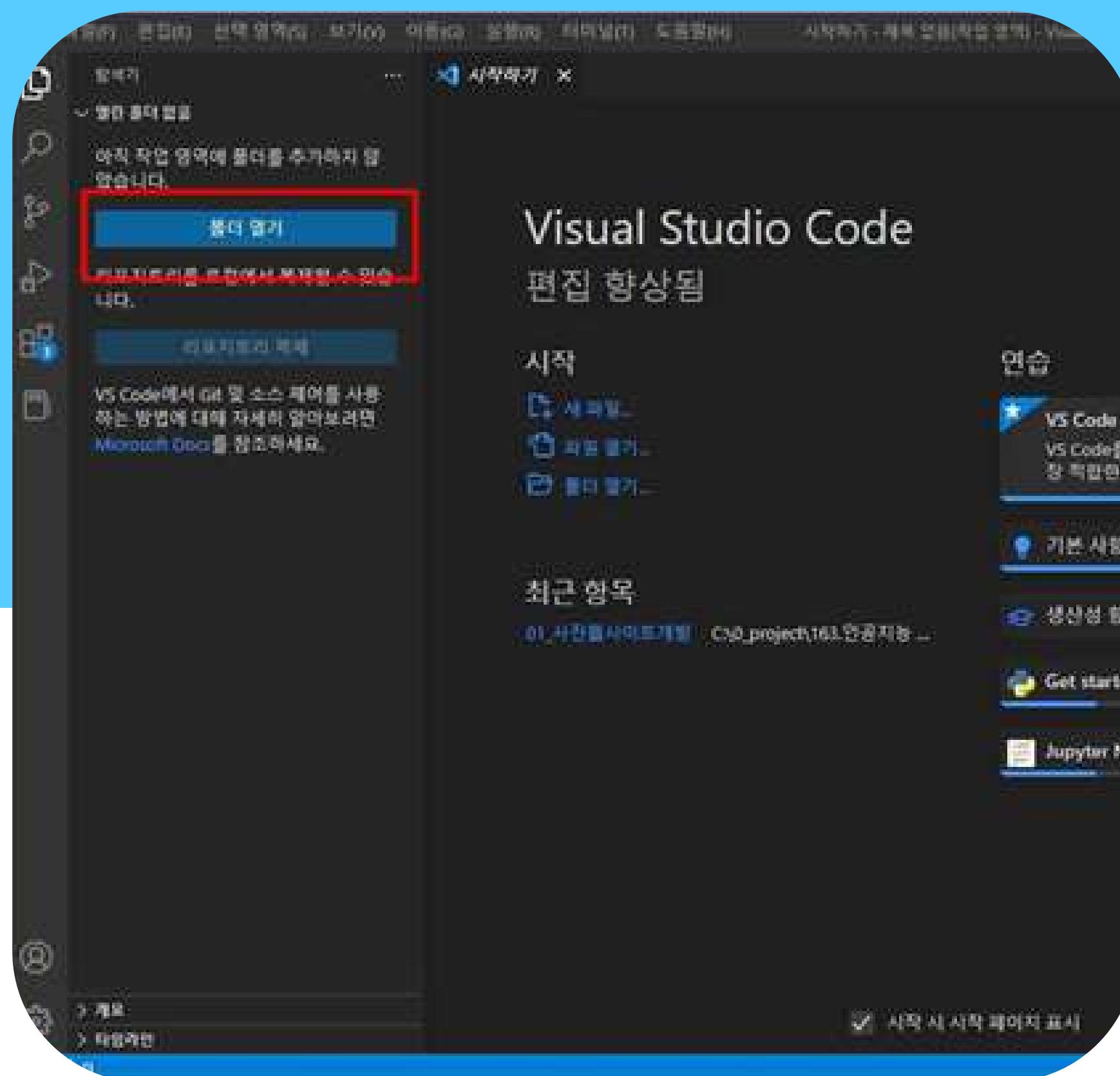


1. PC에 저장되어 있는 파일을 라즈베리파이로도 가져올 수 있다. VNC뷰어 가운데 위쪽으로 마우스를 가져가면 왼쪽의 아이콘이 나온다. 이 중 파일 이동 아이콘을 클릭한다.

2. [Send files...] 버튼을 클릭하여 PC 바탕화면 video.zip 파일을 선택하여 [열기] 버튼을 누르면 video.zip 파일이 PC에서 넘어온다.

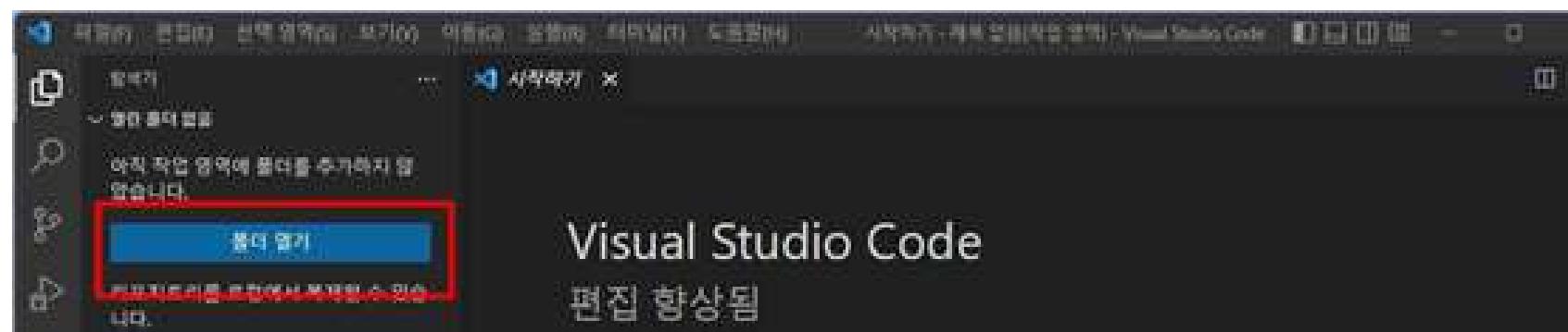


# AI 인공지능 자율주행 자동차

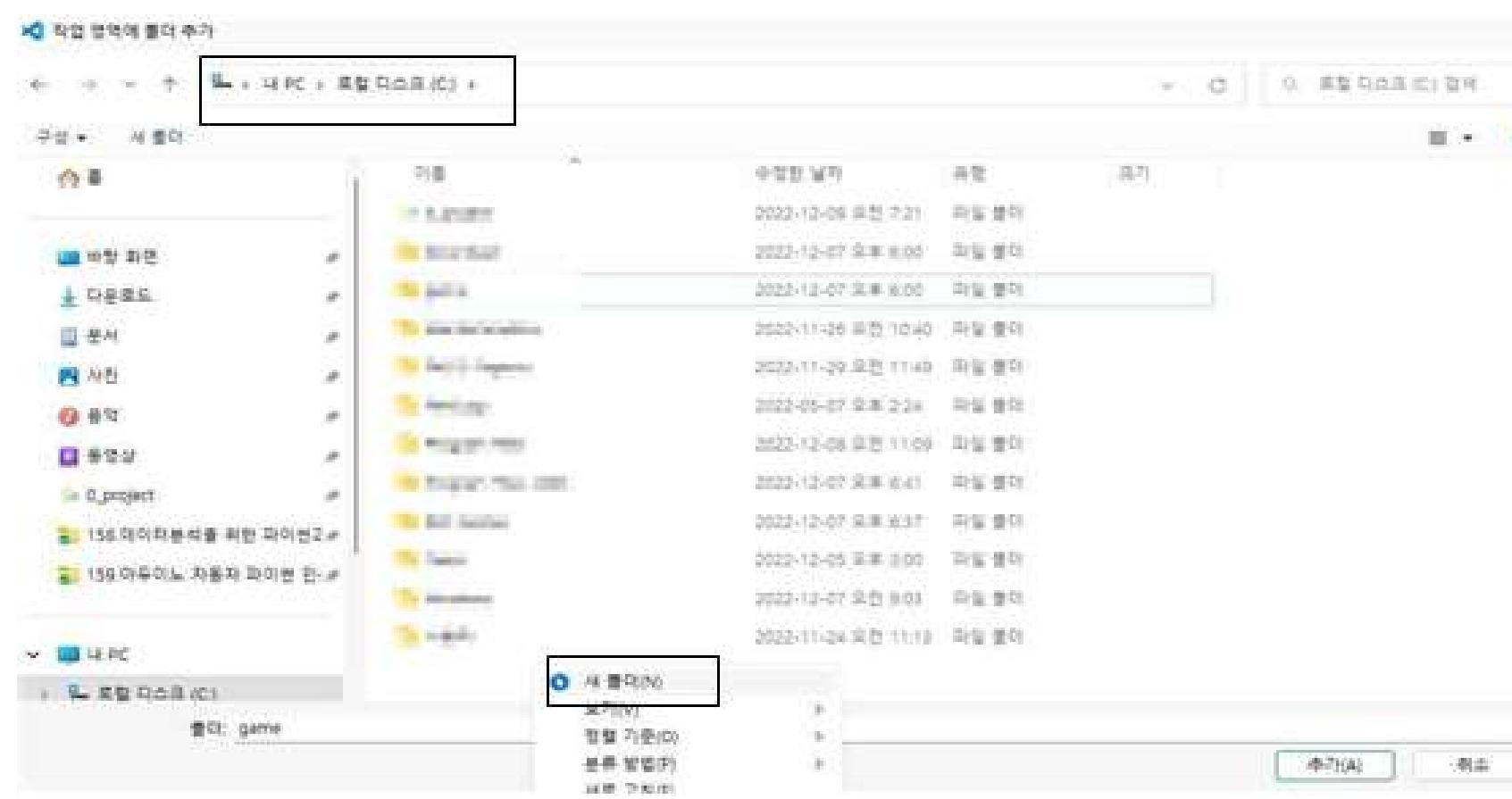


Part 5.  
**자율주행 학습모델 생성**  
**비주얼 스튜디오 코드**

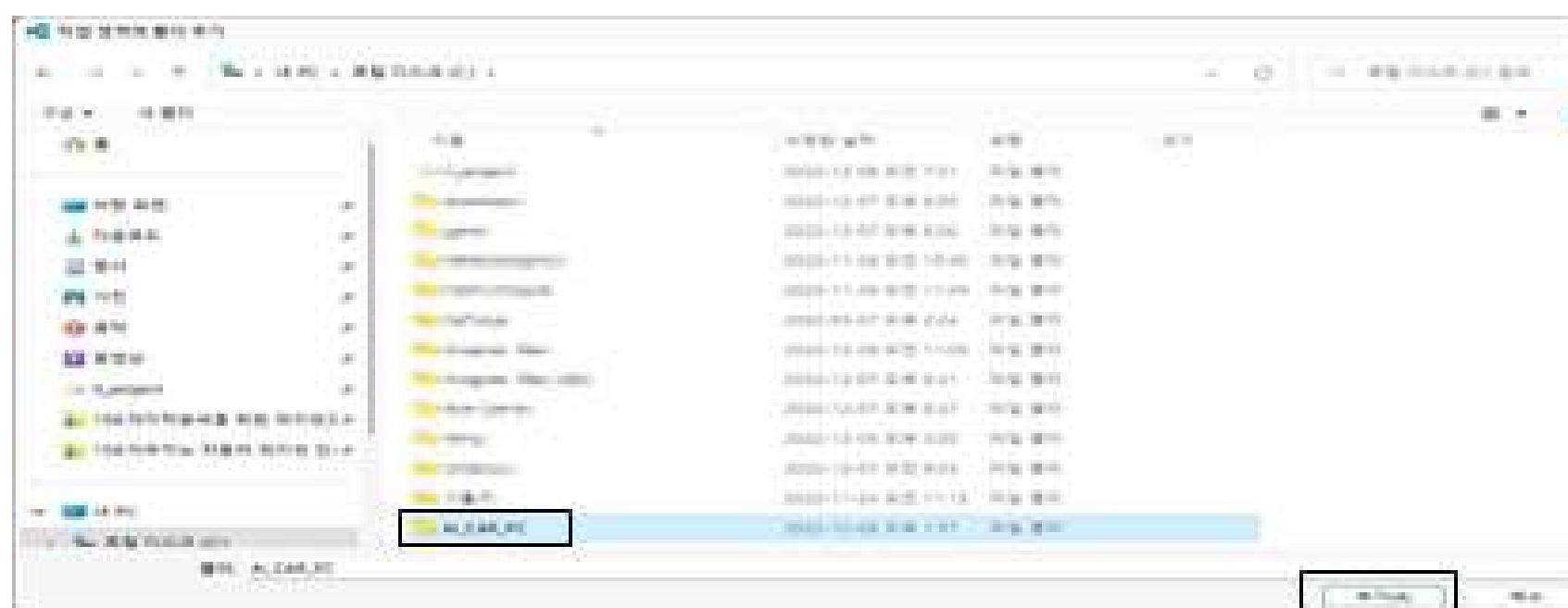
# 자율주행 화성모듈 생성



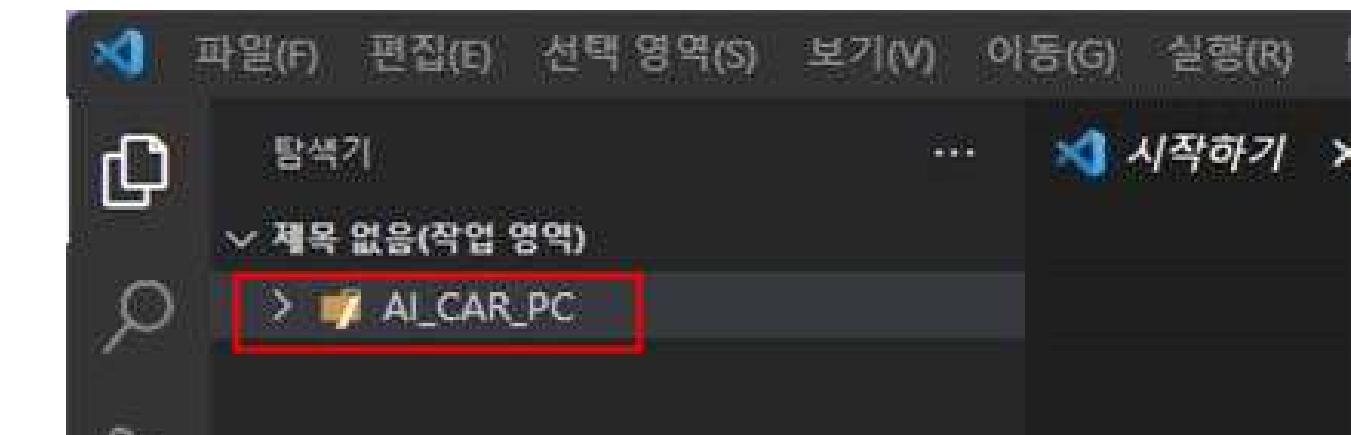
1. [폴더열기]를 클릭하여 작업폴더를 지정



2. C드라이브 아래 마우스 오른쪽을 클릭하여 [새폴더]를 생성 - [AI\_CAR\_PC]로 폴더명



3. 생성된 폴더를 클릭한 후 [추가] 버튼을 클릭  
VS Code 에 작업영역이 추가됨



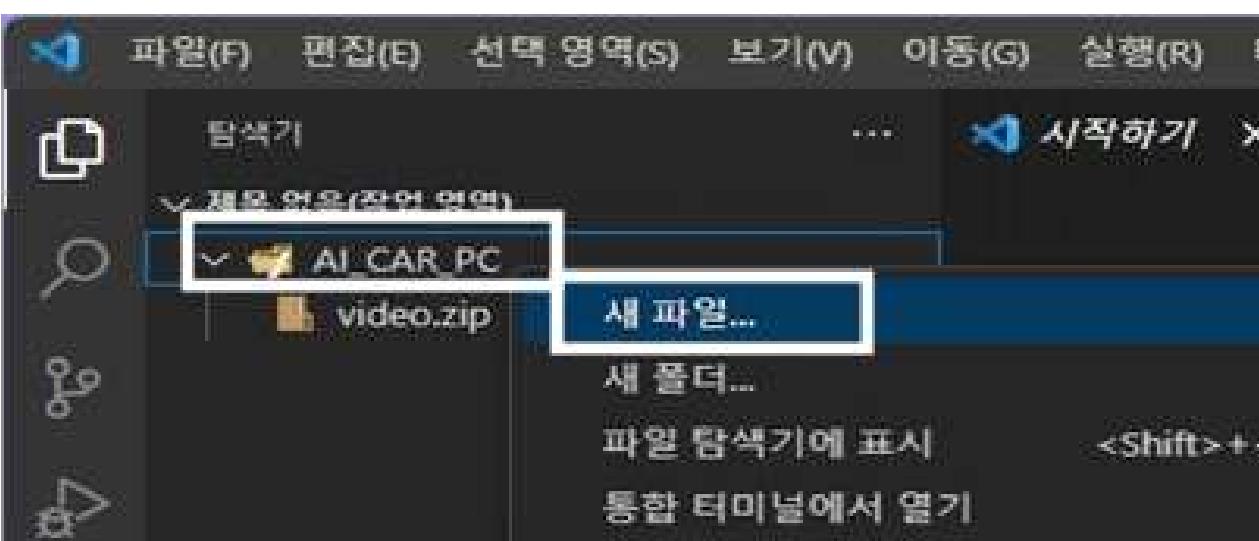
# 자율주행 학습모듈 생성



4. 바탕화면에 저장된 video.zip 파일을 AI\_CAR\_PC 폴더로 이동

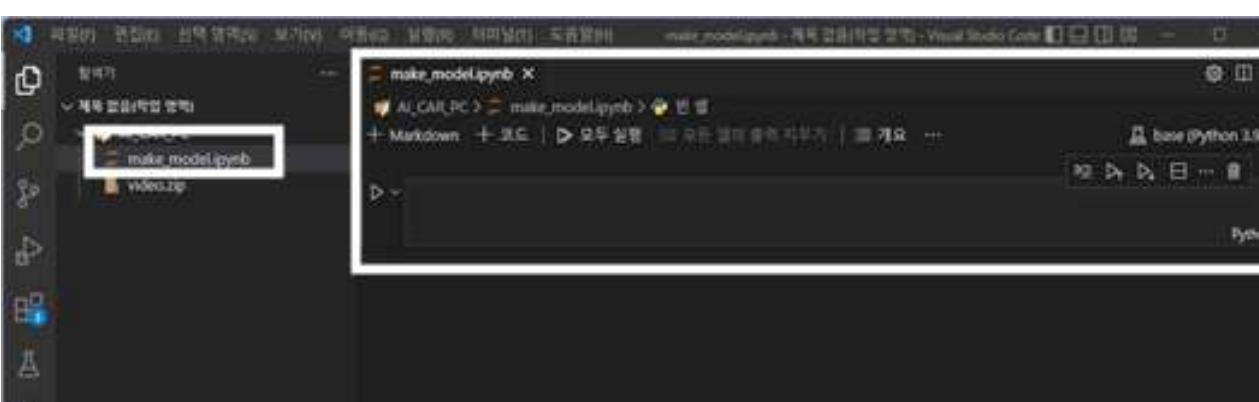


5. 작업영역에 정상적으로 video.zip 파일이 생성



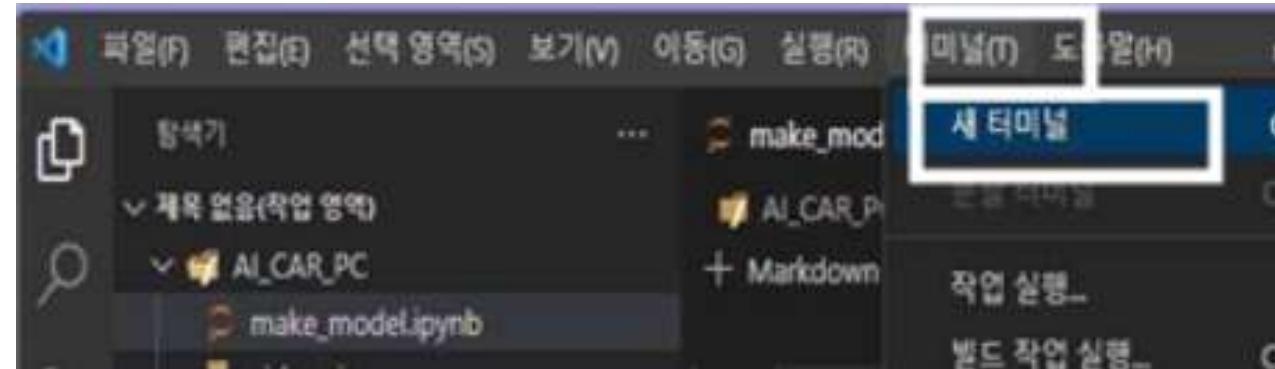
6. [AI\_CAR\_PC] 폴더에서 마우스 오른쪽을 클릭 [새파일...]을 클릭하여 새파일을 생성

파일명 : **make\_model.ipynb**



7. .ipynb는 주피터 노트북방식의 파일로 코드 작성과 실행이 가능

# 자율주행 화성 모듈 생성



8. 자율주행 모델 생성을 위해서는 텐서플로우 버전과 파이썬 버전이 호환가능해야 하므로 호환이 가능한 3.7.3 버전의 가상환경을 생성  
make\_model.ipynb 파일 선택 후 [터미널] -> [새터미널]

9. 터미널에 가상환경 생성 명령어를 입력하여 aicar\_373 이름으로 파인썬 버전 3.7.3버전의 가상환경을 생성되며 아나콘다의 라이브러리가 함께 설치된 (약 5~ 20분 소요)

`conda create -n aicar_373 python=3.7.3 anaconda -y`



10. 설치완료 후 오른쪽 위 vase 부분을 클릭하여 파이썬 선택 후 [aicar\_373] 으로 선택하여 변경

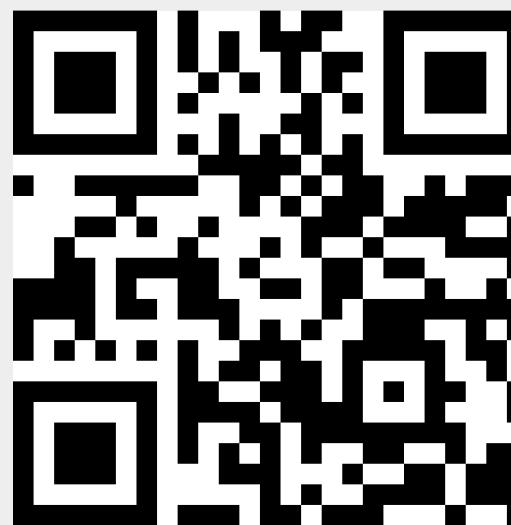


## 학습모듈 코드 공유

학습모듈 코드는 다운을 받아서 순서대로 실행합니다.

### 자율주행 자동차 학습모듈 코드

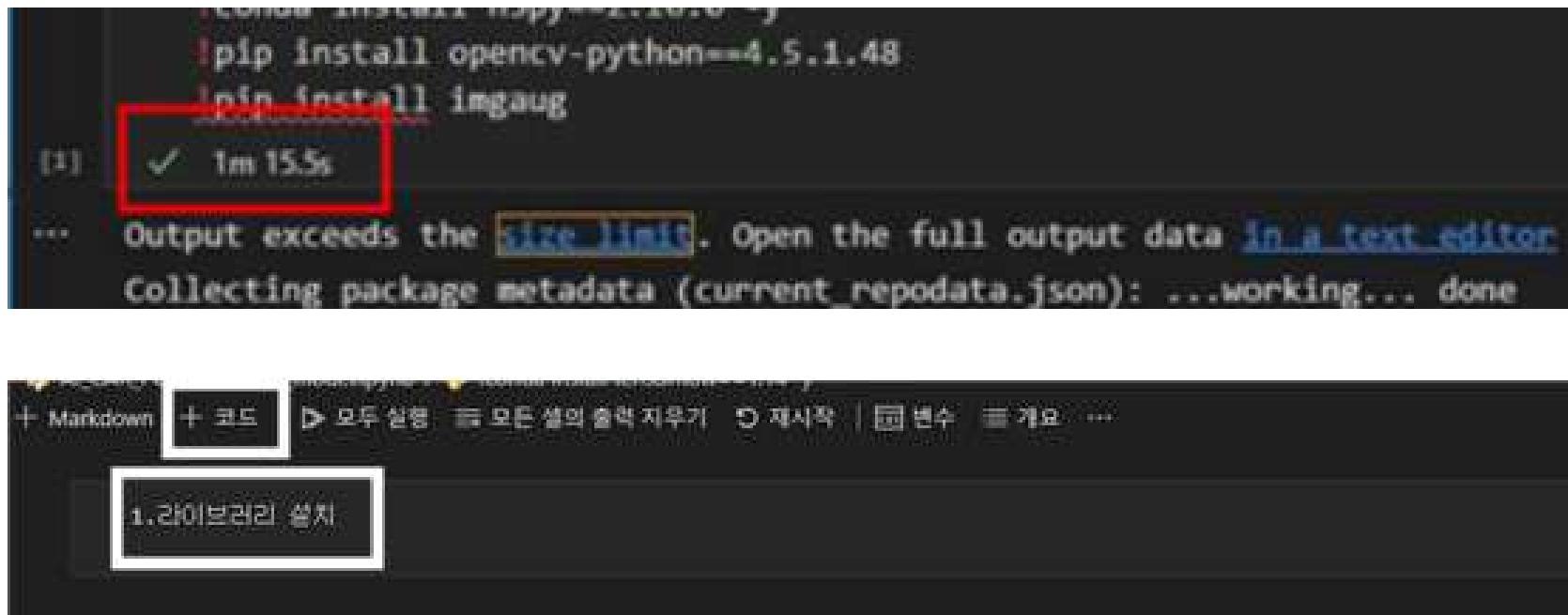
<http://naver.me/xHgyrxeJ>



#### ※ 코드입력 시 주의사항

실행 중 에러나 오류가 나는 경우 코드에 일부 데이터를 불러오지 못하는 경우이므로 에러난 부분에 필요한 라이브러리를 import 명령어로 불러오면 해결됨

```
!conda install tensorflow==1.14 -y  
!conda install keras==2.2.5 -y  
!conda install h5py==2.10.0 -y  
%pip install opencv-python==4.5.1.48  
%pip install imgaug
```



```
import zipfile  
  
zip_file=zipfile.ZipFile("video.zip")  
zip_file.extractall()
```

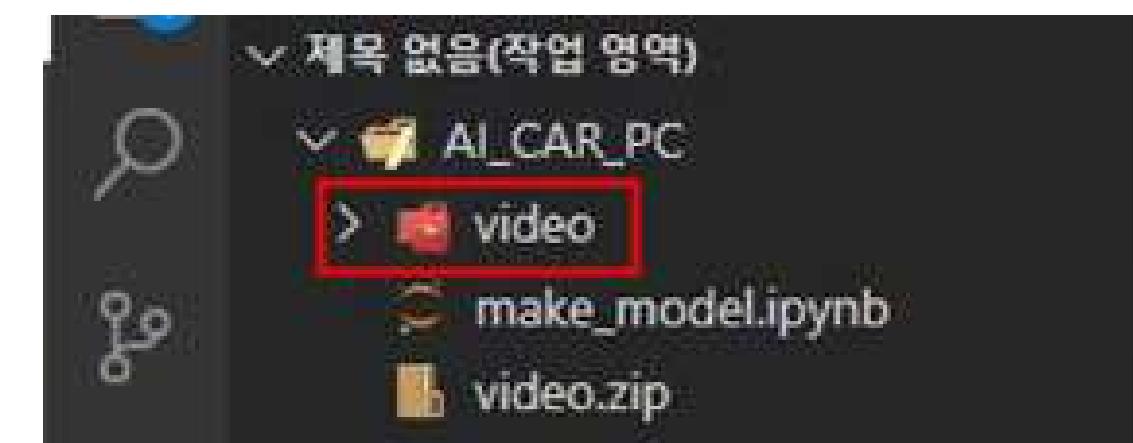
## 1. 라이브러리 설치

학습에 필요한 모듈 라이브러리를 설치합니다.  
설치과정에 오류가 나는 경우 재설치를  
진행해봅니다.

설치가 완료되면 실행된 시간이 출력됩니다.

[+코드]를 눌러 코드 영역을 추가 합니다.

## 2. video.zip 파일의 압축을 풀어줍니다.



```
import os
import random
import fnmatch
import datetime
import pickle

# data processing
import numpy as np
np.set_printoptions(formatter={'float_kind':lambda x: "% .4f" % x})

import pandas as pd
pd.set_option('display.width', 300)
pd.set_option('display.float_format', '{:.4f}'.format)
pd.set_option('display.max_colwidth', 200)

# tensorflow
import tensorflow as tf
import tensorflow.keras
from tensorflow.keras.models import Sequential # V2 is tensorflow.keras.xxxx, V1
is keras.xxxx
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model

print(f'tf.__version__: {tf.__version__}')
print(f'keras.__version__: {tensorflow.keras.__version__}')

# sklearn
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split

# imaging
import cv2
from imgaug import augmenters as img_aug
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

### 3. 라이브러리를 불러옵니다.

- 01~05 : 일반적 라이브러리
- 08~13 : numpy, pandas 등 데이터 처리
- 17~25 : 텐서플로, 케라스 인공지능에 필요
- 24~25 : 텐서플로, 케라스 버전 출력
- 28~29 : sklearn 라이브러리
- 32~37 : OpenCV, matplotlib 등 이미지 처리,  
그래프 그리기에 필요한 라이브러리

```
tf.__version__: 1.14.0
.keras.__version__: 2.2.4-tf
```

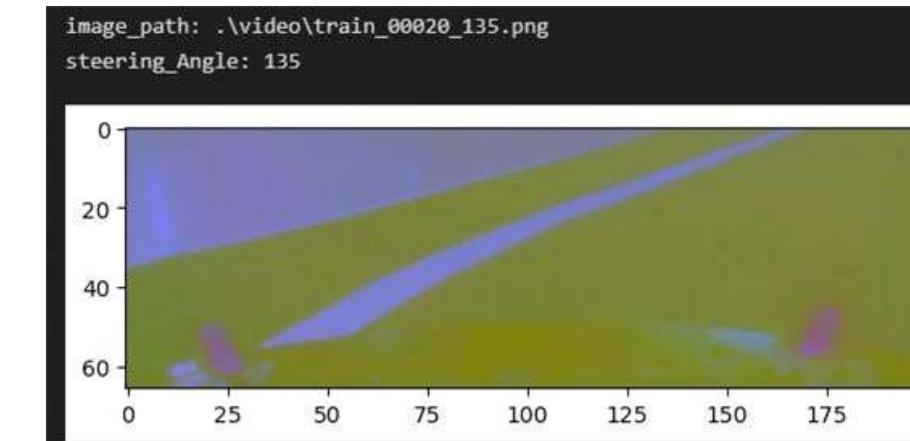
```
import os
import fnmatch
import matplotlib.pyplot as plt
from PIL import Image
import pandas as pd

data_dir = r'\\video'
file_list = os.listdir(data_dir)
image_paths = []
steering_angles = []
pattern = "*.png"
for filename in file_list:
    if fnmatch.fnmatch(filename, pattern):
        image_paths.append(os.path.join(data_dir,filename))
        angle =int(filename[-7:-4])
        steering_angles.append(angle)

image_index =20
plt.imshow(Image.open(image_paths[image_index]))
print("image_path: %s" % image_paths[image_index] )
print("steering_Angle: %d" % steering_angles[image_index] )
df = pd.DataFrame()
df['ImagePath'] = image_paths
df['Angle'] = steering_angles
```

#### 4. 데이터 불러오기

- 01 : video 폴더를 지정한다.
- 02 : 파일의 리스트를 지정
- 03 : 이미지의 경로를 저장하는 변수
- 03 : 이미지의 각도를 저장하는 변수
- 06 : 파일의 개수만큼 반복한다.
- 07 : \*.png의 파일 규칙을 비교한다.  
.png의 파일만 참으로 반환한다.
- 08 : image\_paths 리스트에 파일명을 추가한다.
- 09 : 각도는 사진데이터에서 끝의 -7부터  
-3까지만 분리한다.
- 10 : steering\_angles 리스트에 각도를  
추가한다.
- 12~18 : 20번째 데이터를 확인해본다.



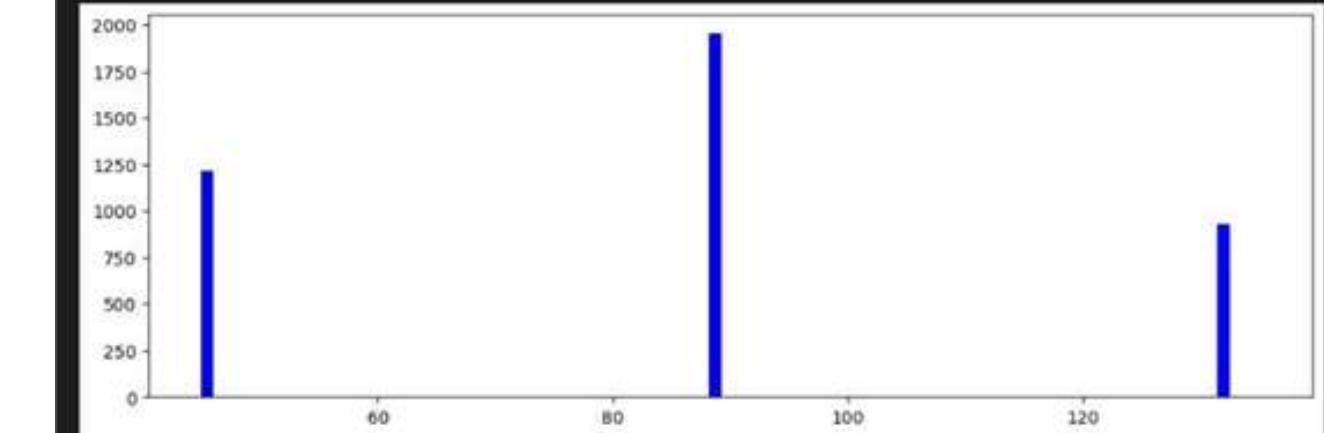
## 5. 조향각 분포

왼쪽, 중앙, 오른쪽의 조향각의 빈도수를 그래프로 그려준다.

```
num_of_bins = 25
hist, bins = np.histogram(df['Angle'], num_of_bins)

fig, axes = plt.subplots(1,1, figsize=(12,4))
axes.hist(df['Angle'], bins=num_of_bins, width=1, color='blue')
```

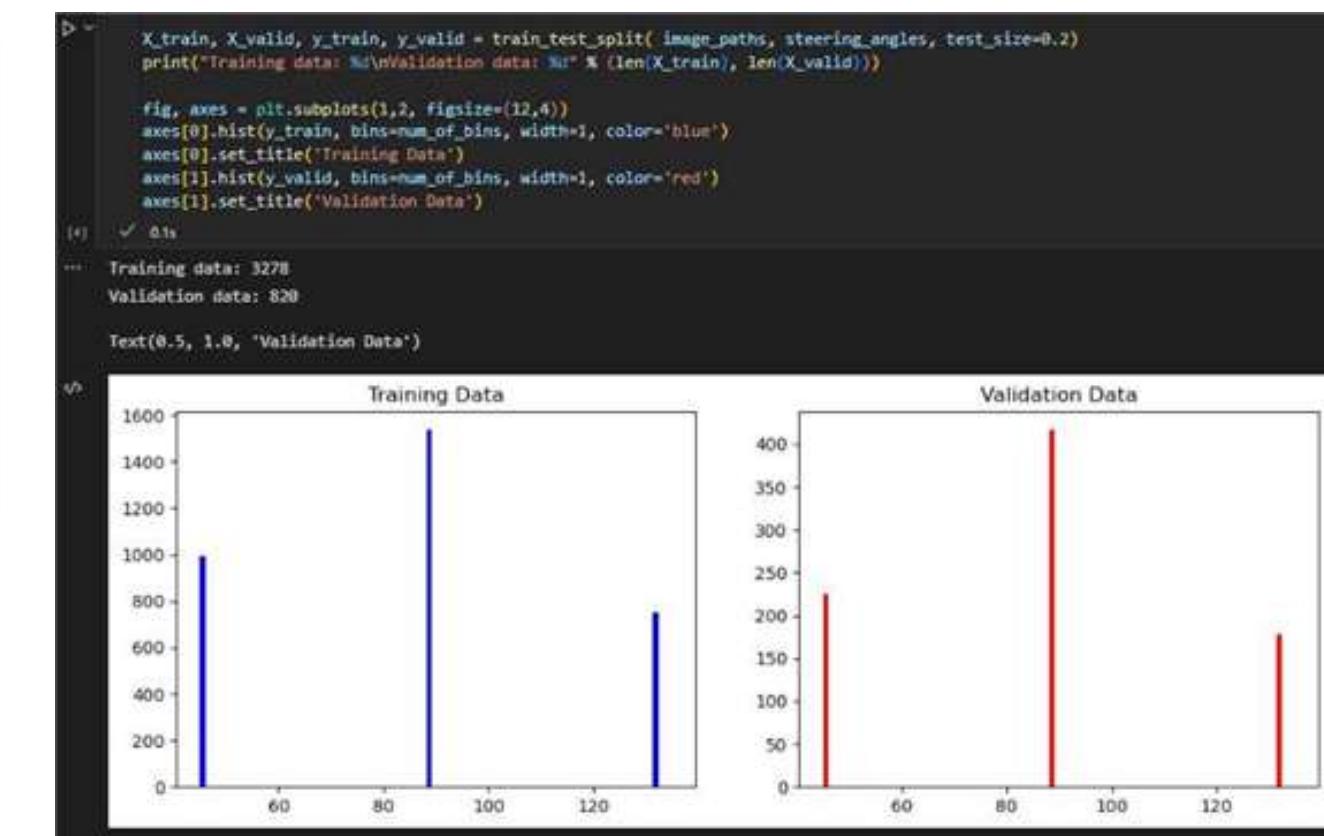
```
(array([1216.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]),  
 array([45.0, 48.5, 52.2, 55.8, 59.4, 63.0, 66.6, 70.2, 73.8, 77.4, 81.0, 84.6, 88.2, 91.8, 95.4, 99.0, 102.6, 106.2, 109.8, 113.4, 117.0, 120.6, 124.2, 127.8, 131.4, 135.0]),  
<BarContainer object of 25 artists>)
```



## 6. 학습데이터와 검증데이터를 분리

```
X_train, X_valid, y_train, y_valid = train_test_split( image_paths, steering_angles, test_size=0.2)
print("Training data: %d\nValidation data: %d" % (len(X_train), len(X_valid)))

fig, axes = plt.subplots(1,2, figsize=(12,4))
axes[0].hist(y_train, bins=num_of_bins, width=1, color='blue')
axes[0].set_title('Training Data')
axes[1].hist(y_valid, bins=num_of_bins, width=1, color='red')
axes[1].set_title('Validation Data')
```



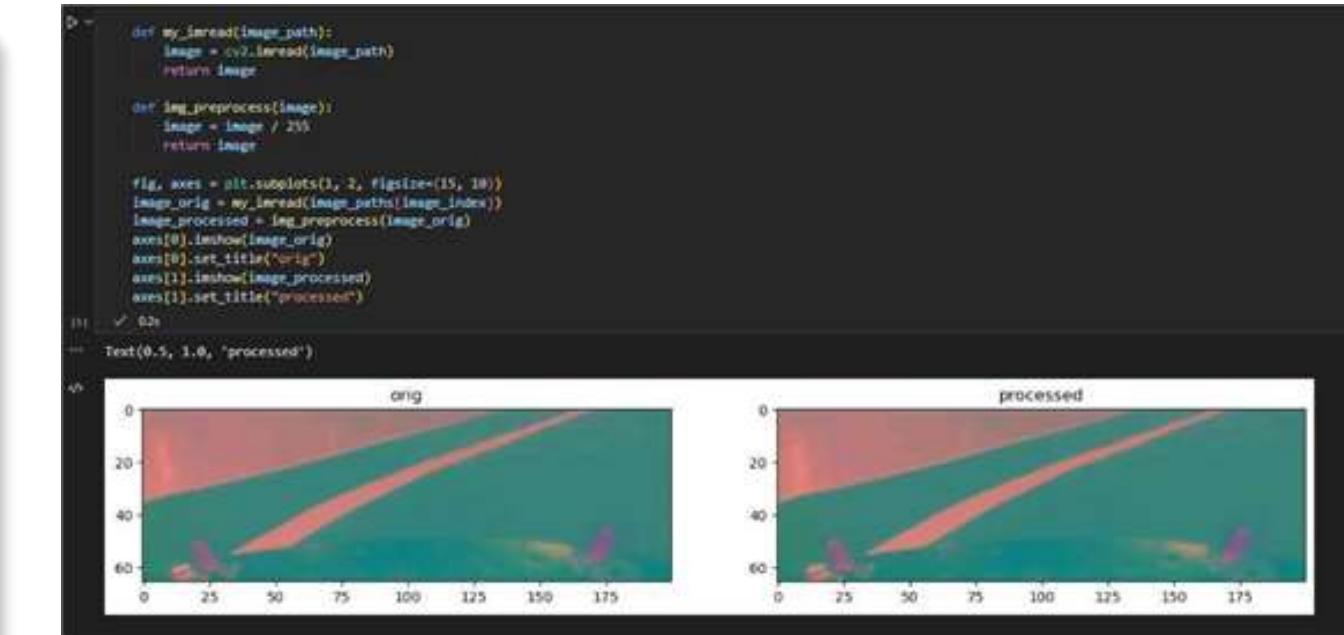
## 7. 이미지 읽어오기 및 정규화

이미지를 읽어오고 값을 특정값 사이로 맞추는 정규화 과정을 진행합니다.

```
def my_imread(image_path):
    image = cv2.imread(image_path)
    return image

def img_preprocess(image):
    image = image / 255
    return image

fig, axes = plt.subplots(1, 2, figsize=(15, 10))
image_orig = my_imread(image_paths[image_index])
image_processed = img_preprocess(image_orig)
axes[0].imshow(image_orig)
axes[0].set_title("orig")
axes[1].imshow(image_processed)
axes[1].set_title("processed")
```



## 8. nvidia 모델구성

```
def nvidia_model():
    model = Sequential(name='Nvidia_Model')

    model.add(Conv2D(24, (5, 5), strides=(2, 2), input_shape=(66, 200, 3), activation='elu'))
    model.add(Conv2D(36, (5, 5), strides=(2, 2), activation='elu'))
    model.add(Conv2D(48, (5, 5), strides=(2, 2), activation='elu'))
    model.add(Conv2D(64, (3, 3), activation='elu'))
    model.add(Dropout(0.2))
    model.add(Conv2D(64, (3, 3), activation='elu'))

    model.add(Flatten())
    model.add(Dropout(0.2))
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))

    model.add(Dense(1))

    optimizer = Adam(lr=1e-3)
    model.compile(loss='mse', optimizer=optimizer)

    return model

model = nvidia_model()
print(model.summary())
```

```
Output exceeds the size limit. Open the full output data in a text editor
WARNING:tensorflow:From c:\Users\jmc\anaconda3\envs\aicar_373\lib\site-packages\tensorflow\python\util\pywrap_tensorflow.py:15: PyC API: This function or variable may be removed without notice, or moved to a different public API. This was done to better separate TensorFlow's internal implementation from its public Python API. If you rely on this API, please consider using the TensorFlow C++ API instead.
version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Model: "Nvidia_Model"
+-----+-----+-----+
Layer (type)      Output Shape     Param #
+-----+-----+-----+
conv2d (Conv2D)   (None, 31, 98, 24)  1824
+-----+-----+-----+
conv2d_1 (Conv2D) (None, 14, 47, 36)  21636
+-----+-----+-----+
conv2d_2 (Conv2D) (None, 5, 22, 48)   43248
+-----+-----+-----+
conv2d_3 (Conv2D) (None, 3, 20, 64)   27712
+-----+-----+-----+
dropout (Dropout) (None, 3, 20, 64)   0
+-----+-----+-----+
conv2d_4 (Conv2D) (None, 1, 18, 64)   36928
+-----+-----+-----+
flatten (Flatten) (None, 1152)        0
+-----+-----+-----+
dropout_1 (Dropout) (None, 1152)        0
+-----+-----+-----+
dense (Dense)    (None, 100)          115300
+-----+-----+-----+
...
Trainable params: 252,219
Non-trainable params: 0
+-----+-----+-----+
None
```

총 252,219 파라메터로 모델이 생성

# 자율주행 학습모델 생성

## 9. 학습데이터 생성

```
def image_data_generator(image_paths, steering_angles, batch_size):
    while True:
        batch_images = []
        batch_steering_angles = []

        for i in range(batch_size):
            random_index = random.randint(0, len(image_paths) - 1)
            image_path = image_paths[random_index]
            image = my_imread(image_paths[random_index])
            steering_angle = steering_angles[random_index]

            image = img_preprocess(image)
            batch_images.append(image)
            batch_steering_angles.append(steering_angle)

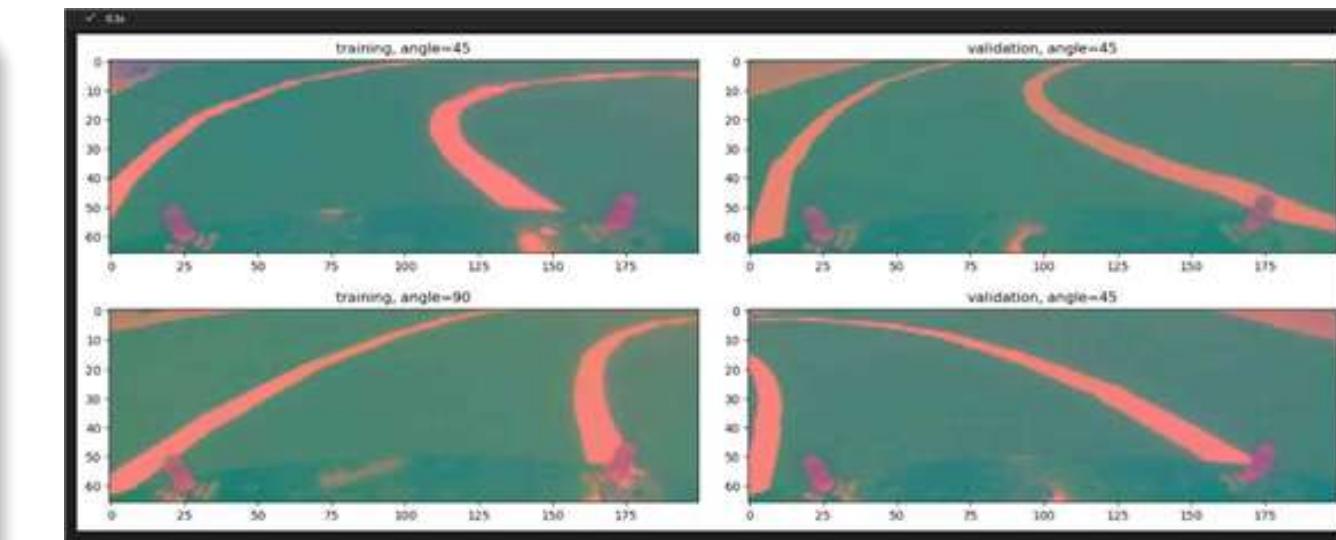
        yield(np.asarray(batch_images), np.asarray(batch_steering_angles))

ncol = 2
nrow = 2

X_train_batch, y_train_batch = next(image_data_generator(X_train, y_train, nrow))
X_valid_batch, y_valid_batch = next(image_data_generator(X_valid, y_valid, nrow))

fig, axes = plt.subplots(nrow, ncol, figsize=(15, 6))
fig.tight_layout()

for i in range(nrow):
    axes[i][0].imshow(X_train_batch[i])
    axes[i][0].set_title("training, angle=%s" % y_train_batch[i])
    axes[i][1].imshow(X_valid_batch[i])
    axes[i][1].set_title("validation, angle=%s" % y_valid_batch[i])
```



왼쪽은 학습데이터 오른쪽은 검증데이터가 불러와진다.

## 10. 모델학습

```
model_output_dir = r""

checkpoint_callback = tensorflow.keras.callbacks.ModelCheckpoint(filepath=os.path.join(model_output_dir,'lane_navigation_check.h5'), verbose=1, save_best_only=True)

history = model.fit_generator(image_data_generator(X_train, y_train, batch_size=100),
                               steps_per_epoch=300,
                               epochs=10,
                               validation_data = image_data_generator(X_valid, y_valid, batch_size=100),
                               validation_steps=200,
                               verbose=1,
                               shuffle=1,
                               callbacks=[checkpoint_callback])

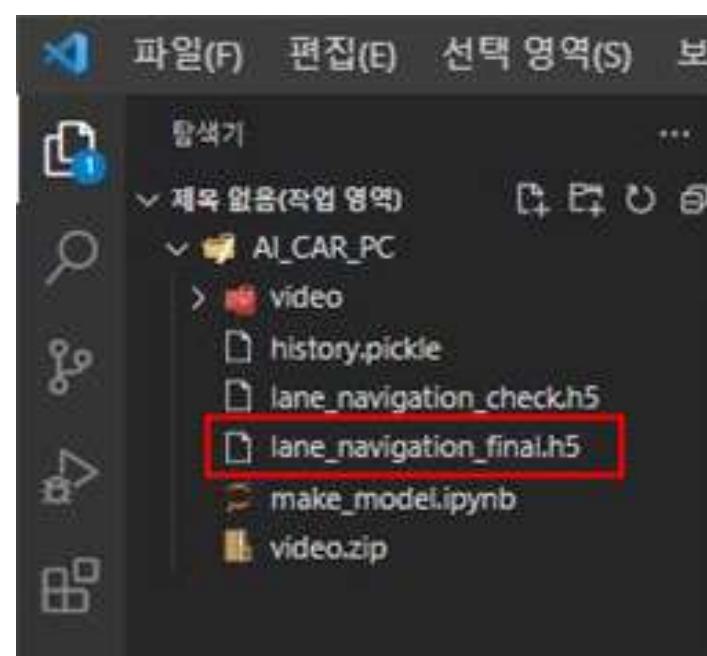
model.save(os.path.join(model_output_dir,'lane_navigation_final.h5))

history_path = os.path.join(model_output_dir,'history.pickle')
with open(history_path, 'wb') as f:
    pickle.dump(history.history, f, pickle.HIGHEST_PROTOCOL)
```

```
pickle.dump(history.history, f, pickle.HIGHEST_PROTOCOL)
8m 02s
...
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/10
299/300 [=====>.] - ETA: 0s - loss: 1309.2217
Epoch 00001: val_loss improved from inf to 517.48675, saving model to ./lane_navigation_check.h5
300/300 [=----] - 57s 190ms/step - loss: 1306.6166 - val_loss: 517.4868
Epoch 2/10
299/300 [=====>.] - ETA: 0s - loss: 484.9141
Epoch 00002: val_loss improved from 517.48675 to 400.36557, saving model to ./lane_navigation_check.h5
300/300 [=----] - 47s 158ms/step - loss: 485.1588 - val_loss: 400.3656
Epoch 3/10
299/300 [=====>.] - ETA: 0s - loss: 416.4627
Epoch 00003: val_loss improved from 400.36557 to 387.45208, saving model to ./lane_navigation_check.h5
300/300 [=----] - 46s 155ms/step - loss: 416.5438 - val_loss: 387.4521
Epoch 4/10
299/300 [=====>.] - ETA: 0s - loss: 394.3343
Epoch 00004: val_loss improved from 387.45208 to 350.49482, saving model to ./lane_navigation_check.h5
300/300 [=----] - 47s 155ms/step - loss: 394.4374 - val_loss: 350.4948
Epoch 5/10
299/300 [=====>.] - ETA: 0s - loss: 351.3313
Epoch 00005: val_loss improved from 350.49482 to 277.45476, saving model to ./lane_navigation_check.h5
300/300 [=----] - 46s 154ms/step - loss: 351.3705 - val_loss: 277.4548
Epoch 6/10
299/300 [=====>.] - ETA: 0s - loss: 284.1210
Epoch 00006: val_loss improved from 277.45476 to 218.64062, saving model to ./lane_navigation_check.h5
300/300 [=----] - 47s 156ms/step - loss: 284.0698 - val_loss: 218.6406
Epoch 7/10
...
Epoch 10/10
299/300 [=====>.] - ETA: 0s - loss: 112.7280
Epoch 00010: val_loss improved from 122.68211 to 93.36373, saving model to ./lane_navigation_check.h5
300/300 [=----] - 47s 158ms/step - loss: 112.6370 - val_loss: 93.3637
```

모델학습은 약 5~30분 가량 소요된다.  
(CPU i5 이상 RAM 8GB 이상 기준)

lane\_navigation\_final.h5 이름으로 데이터가 저장



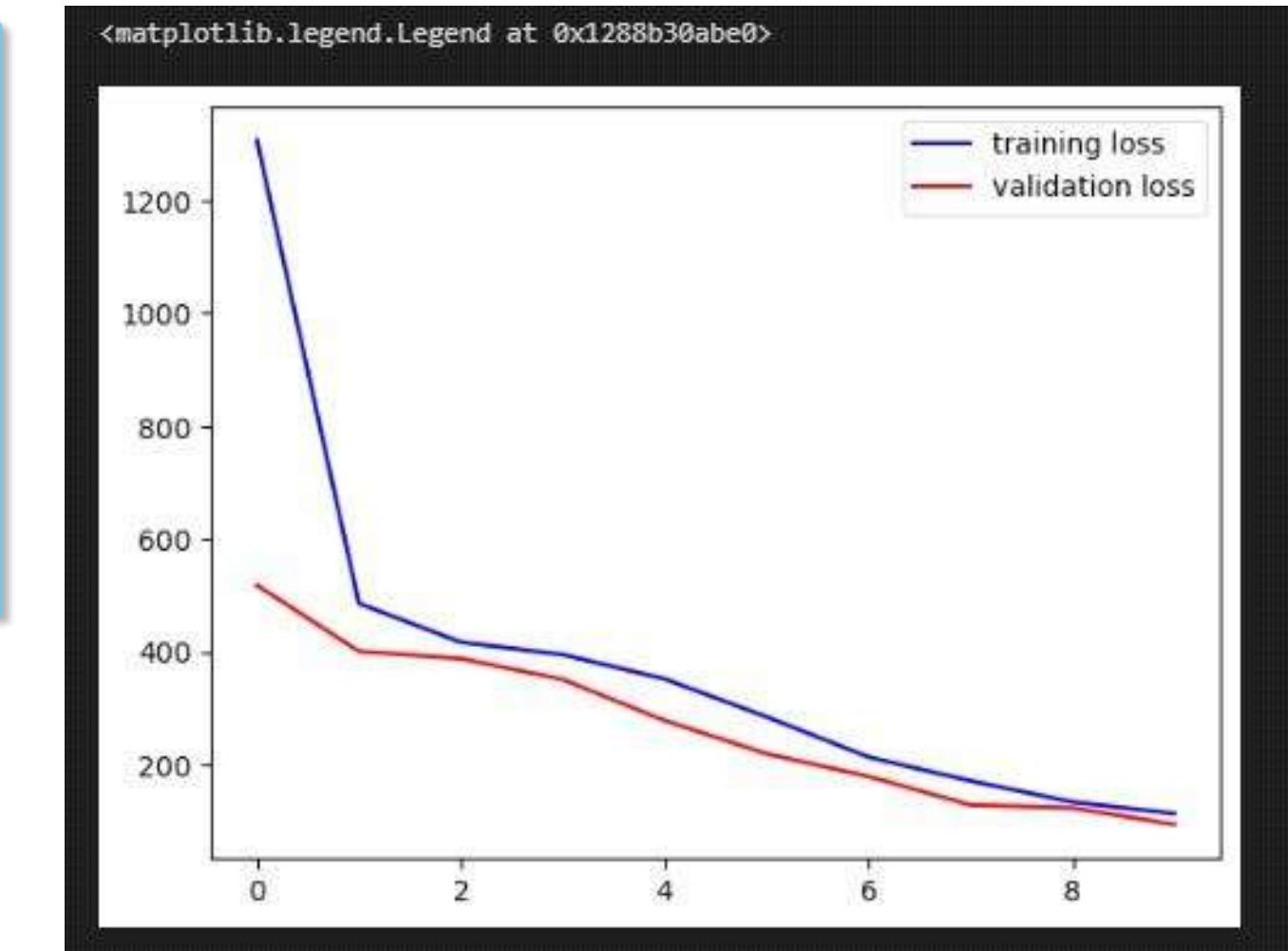
## 11. 결과 확인

```
history.history

history_path = os.path.join(model_output_dir,'history.pickle')
with open(history_path, 'rb') as f:
    history = pickle.load(f)

history
plt.plot(history['loss'],color='blue')
plt.plot(history['val_loss'],color='red')
plt.legend(["training loss", "validation loss"])
```

파란색은 학습데이터, 빨간색은 검증데이터로  
오차가 점점 줄어드는 그래프의 형태로 확인 가능



## 12. 검증

```
from sklearn.metrics import mean_squared_error, r2_score

def summarize_prediction(Y_true, Y_pred):

    mse = mean_squared_error(Y_true, Y_pred)
    r_squared = r2_score(Y_true, Y_pred)

    print(f'mse = {mse:.2f}')
    print(f'r_squared = {r_squared:.2%}')
    print()

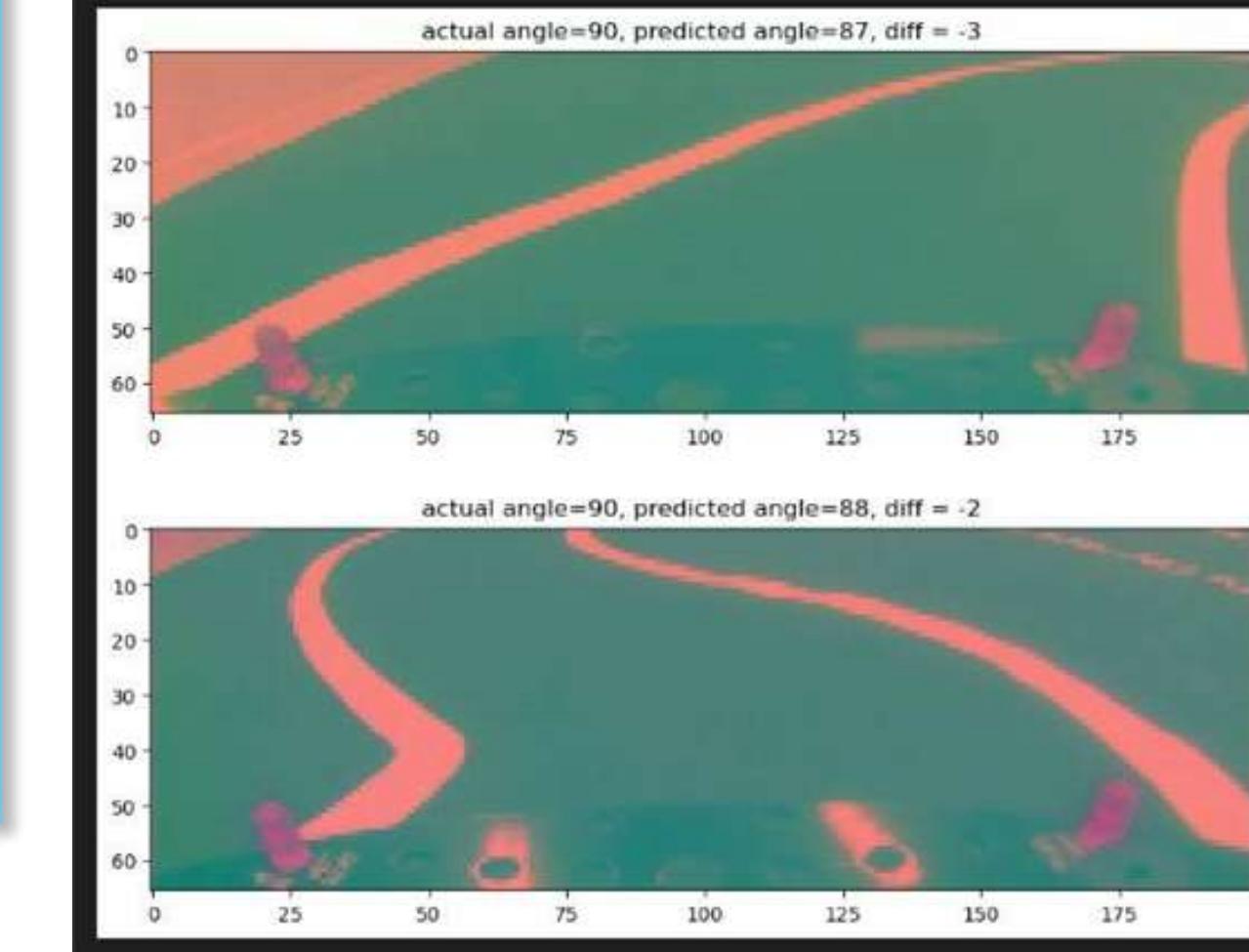
def predict_and_summarize(X, Y):
    model = load_model(f'{model_output_dir}/lane_navigation_check.h5')
    Y_pred = model.predict(X)
    summarize_prediction(Y, Y_pred)
    return Y_pred

n_tests = 100
X_test, y_test = next(image_data_generator(X_valid, y_valid, 100))

y_pred = predict_and_summarize(X_test, y_test)

n_tests_show = 2
fig, axes = plt.subplots(n_tests_show, 1, figsize=(10, 4 * n_tests_show))
for i in range(n_tests_show):
    axes[i].imshow(X_test[i])
    axes[i].set_title(f"actual angle={y_test[i]}, predicted angle={int(y_pred[i])}, diff = {int(y_pred[i])-y_test[i]}")
```

```
WARNING:tensorflow:From c:\Users\jmc\anaconda3\envs\aicar_373\lib\site-packages\tensorflow\python\ops\init_ops.py:146: get_init is deprecated and will be removed after 2020-10-01.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From c:\Users\jmc\anaconda3\envs\aicar_373\lib\site-packages\tensorflow\python\ops\init_ops.py:146: get_init is deprecated and will be removed after 2020-10-01.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
mse      = 1.4e+02
r_squared = 87.72%
```



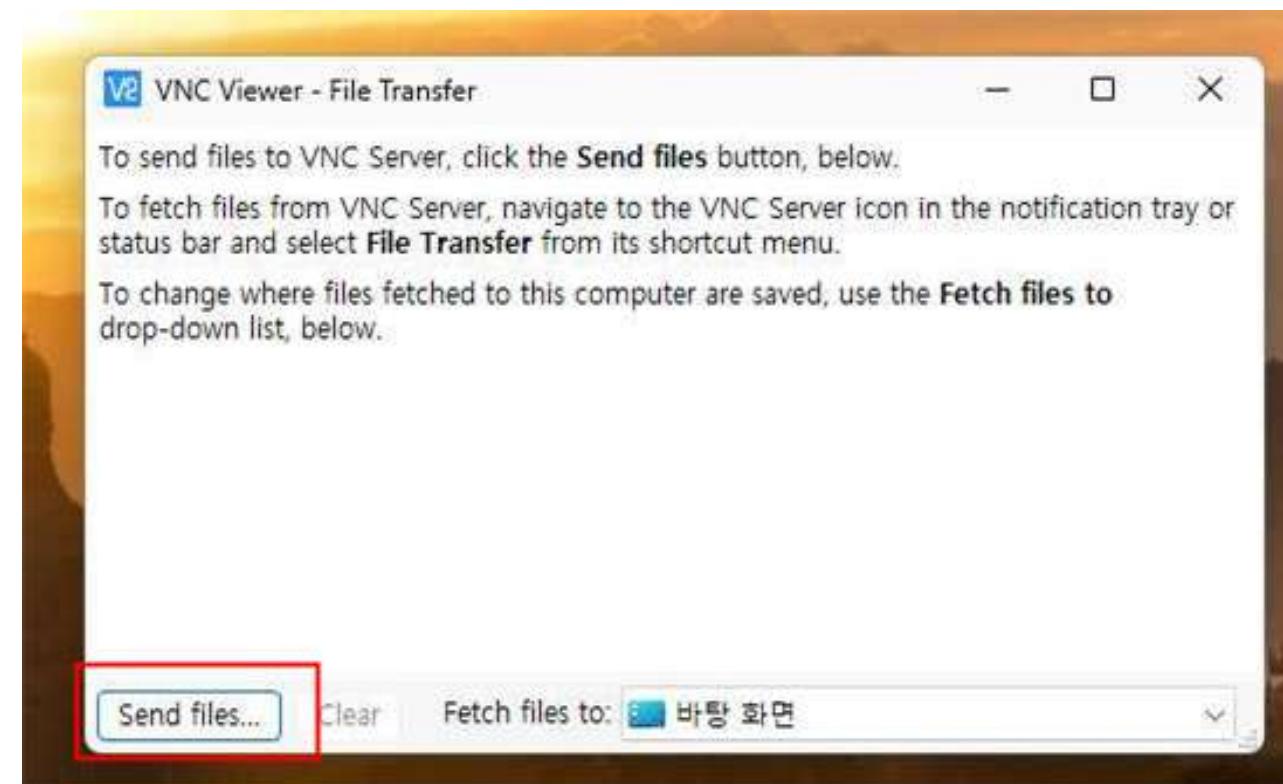
학습데이터의 성능 평가를 하였다.

actual angle은 실제 각도, predicted angle은 예측각도 diff는 실제각도와 예측각도의 차이이다.  
diff가 0에 가까울수록 학습이 잘되었다고 판단한다.

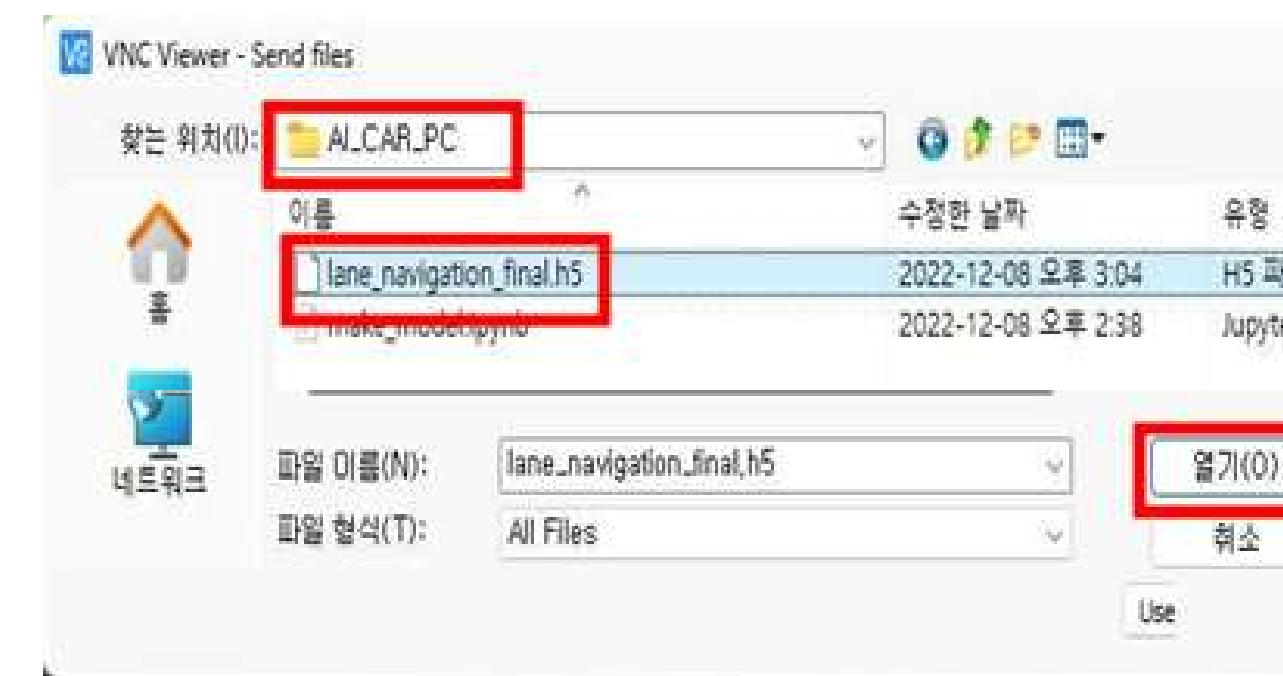
# 모델파일 라즈베리파이로 옮기기



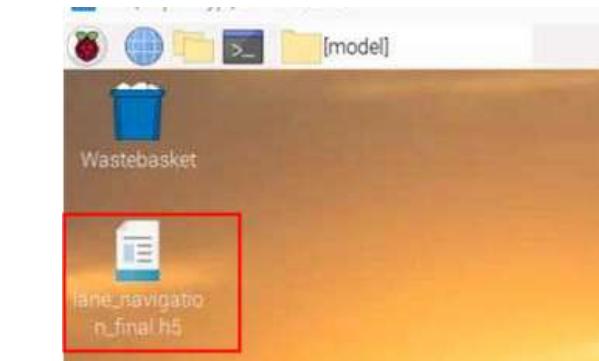
1. VNC 뷰어 중앙 윗부분에 마우스를 가져가 메뉴바를 연다.  
[Transfer file]를 클릭



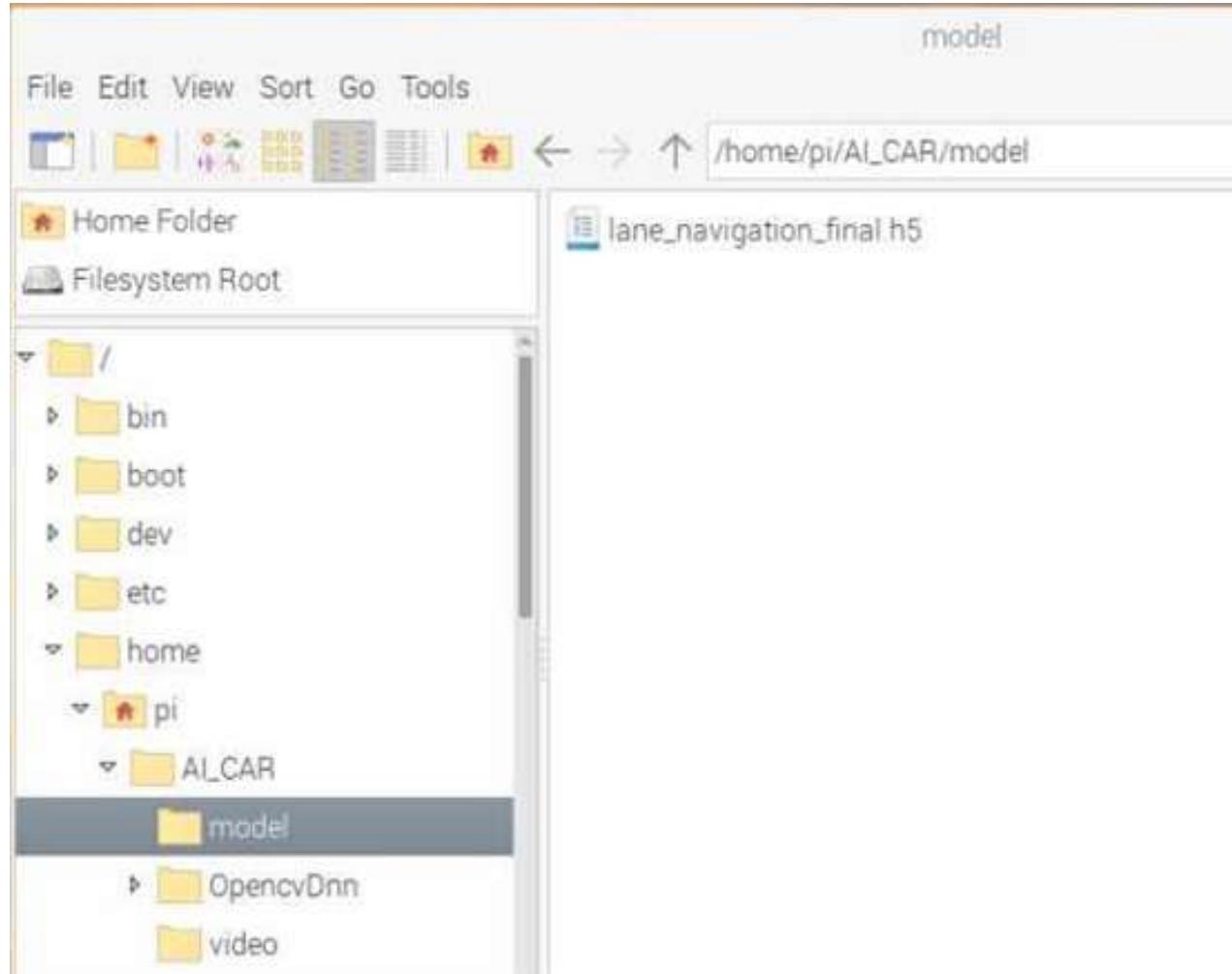
2. [Send files]를 클릭



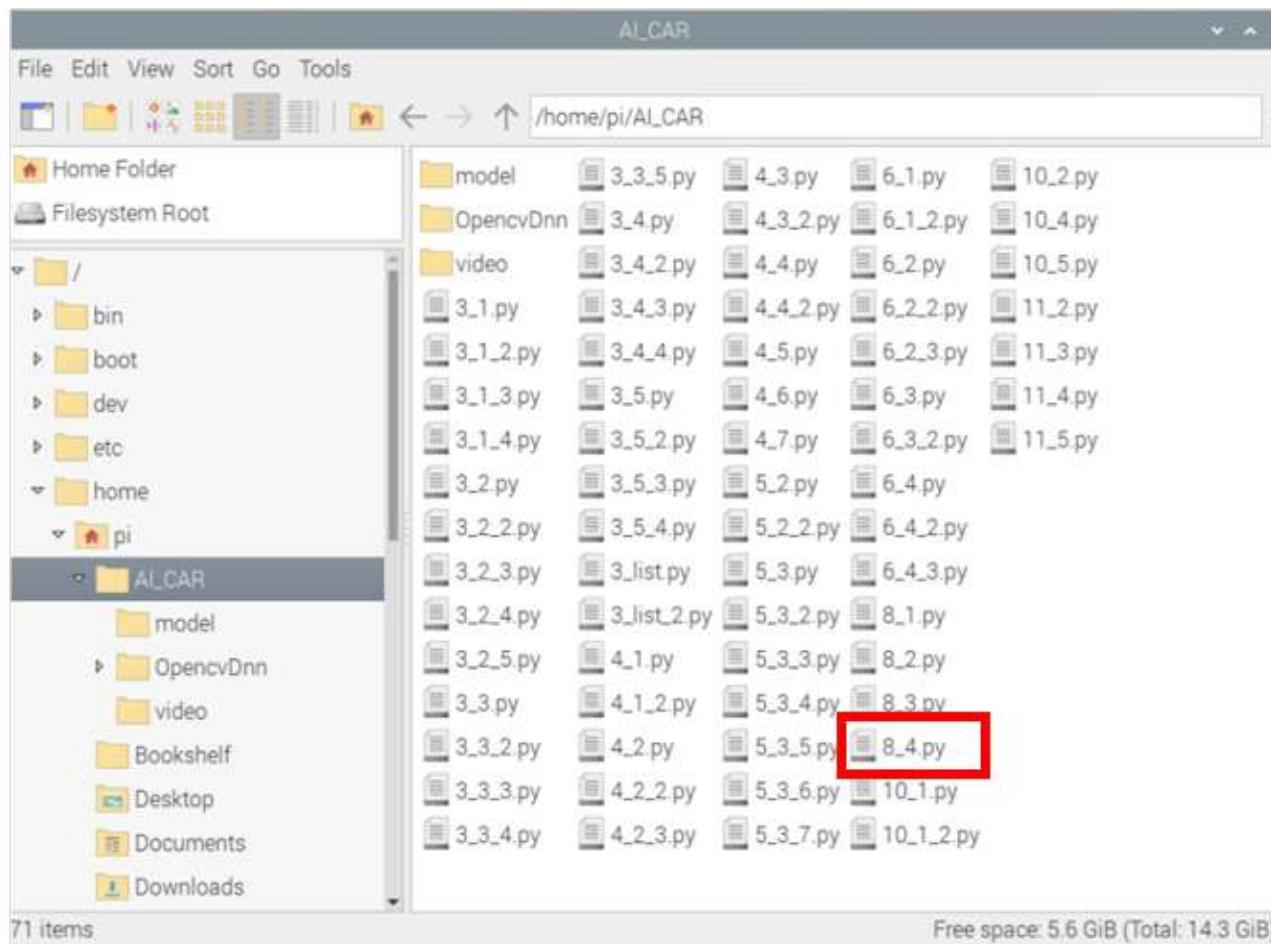
3. 윈도우의 AI\_CAR\_PC 폴더에서  
[lane\_navigation\_final.h5]를 선택 후 [열기]를 클릭  
PC파일을 라즈베리파이로 복사



# 모델보드 로즈베리파이로 운영



4. 라즈베리파이의 [AI\_CAR] 폴더의 [model] 폴더에 [lane\_navigation\_final.h5]파일을 이동하여 사용

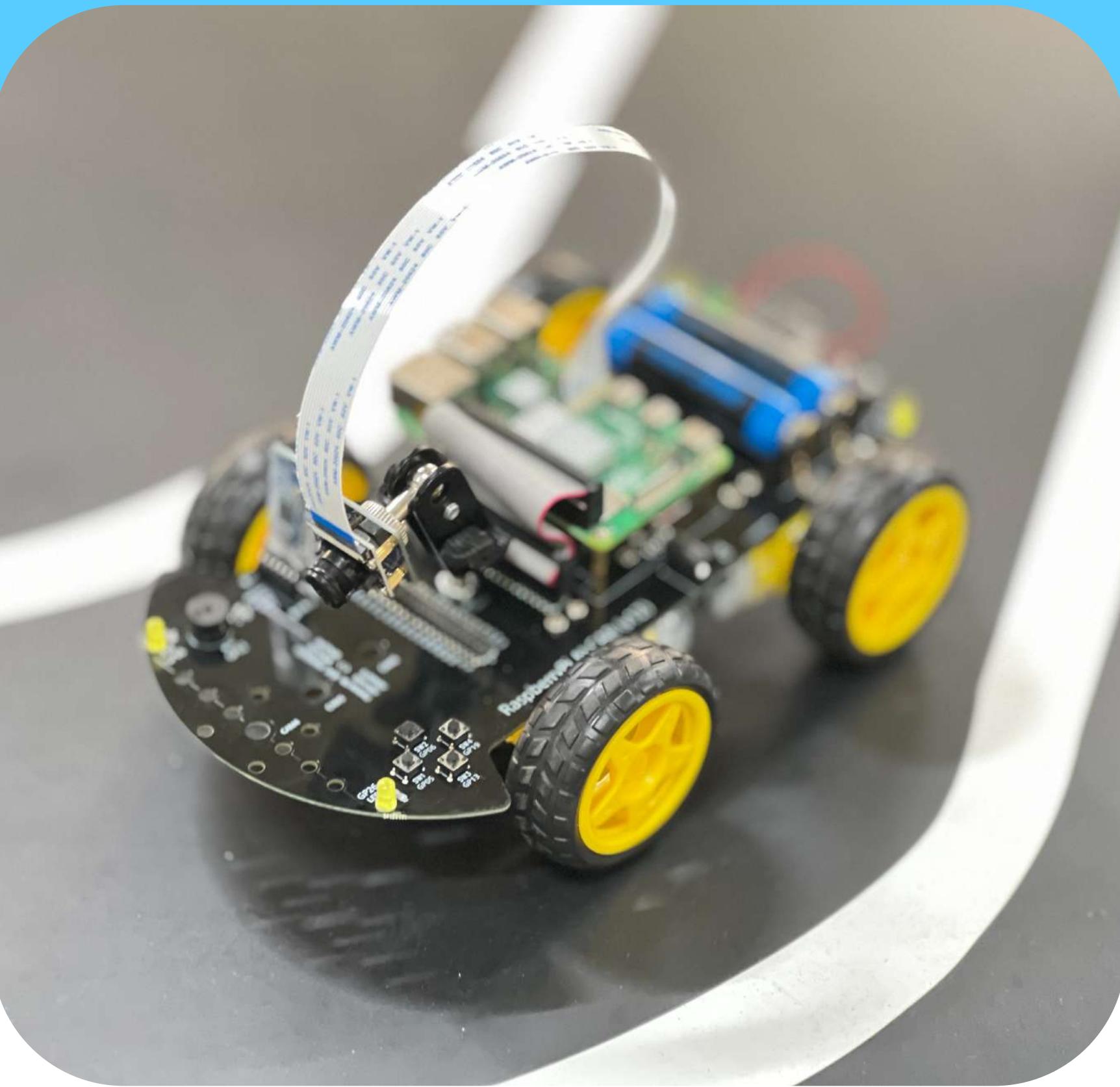


5. 자율주행 테스트 - 8\_4.py

Thonny 를 실행하여 8\_4.py 를 열고 [RUN] 을 실행하여  
자율주행을 정상적으로 수행하는지 확인한다.

코드에 스피드 조절은 speedSet = 0.7 이 부분을  
추가하여 조절 가능하다.

단, 전진, 좌회전, 우회전 각각 스피드 셋을 motor\_\*\*\* 의  
윗줄에 코드 추가하여야 적용가능함



# AI 인공지능 자율주행 자동차

Part 6.  
**자율주행 학습모델**  
**필터 적용을 통한 성능향상**

```
import time
import mycamera
import cv2

camera = mycamera.MyPiCamera(640,480)

def main():
    try:
        while True:

            keyValue = cv2.waitKey(1)

            if keyValue == ord('q'):
                break

            _, image = camera.read()
            image = cv2.flip(image,-1)
            cv2.imshow('Original', image)

            height, _, _ = image.shape
            image = image[int(height/2):,:,:]
            image = cv2.cvtColor(image, cv2.COLOR_BGR2YUV)
            image = cv2.resize(image, (200,66))
            image = cv2.GaussianBlur(image,(5,5),0)
            _,image = cv2.threshold(image,200,255,cv2.THRESH_BINARY_INV)
            cv2.imshow('processed', image)

    except KeyboardInterrupt:
        pass

if __name__ == '__main__':
    main()
    cv2.destroyAllWindows()
```

## 1. 빛의 영향을 덜 받는 코드 - 10\_1\_2.py

카메라만 사용하는 자율주행 차량의 주행성능을 향상 시키기 위해서는 빛에 의해 반사되는 광이 카메라에서 인식되는 라인과 중첩되거나 또는 산란되는 빛에 의해 발생하는 노이즈에 의한 주행라인 인식율이 저하되는 것을 별도의 필터를 적용하여 주행라인의 인식율을 높여줌으로서 주행성능 향상에 영향을 주게 된다.

# 更深 의 교 재 를 통 해 학 습 하 는 방 법



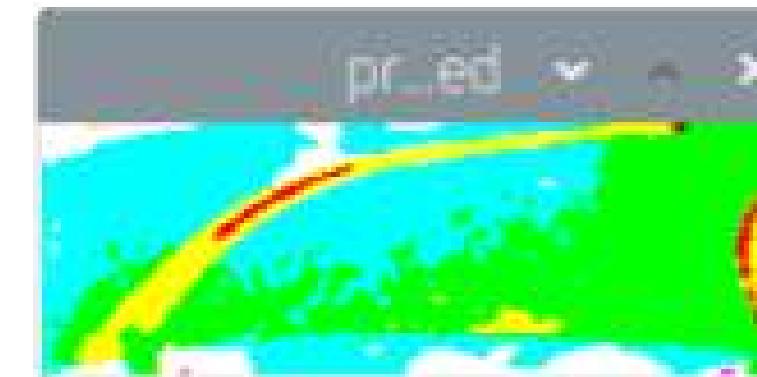
```
_image = cv2.threshold(image,200,255,cv2.THRESH_BINARY_INV)
```

임계점 200일때로 차선도 잘보이고 적당한 노이즈가 있다.  
본 교재에서는 200의 값으로 학습하였다.



```
_image = cv2.threshold(image,100,255,cv2.THRESH_BINARY_INV)
```

100일때는 차선이 인식하지 못한다.



```
_image = cv2.threshold(image,110,255,cv2.THRESH_BINARY_INV)
```

130일때는 차선은 인식하나 노이즈가 너무 많아서 사용하기 힘들다.



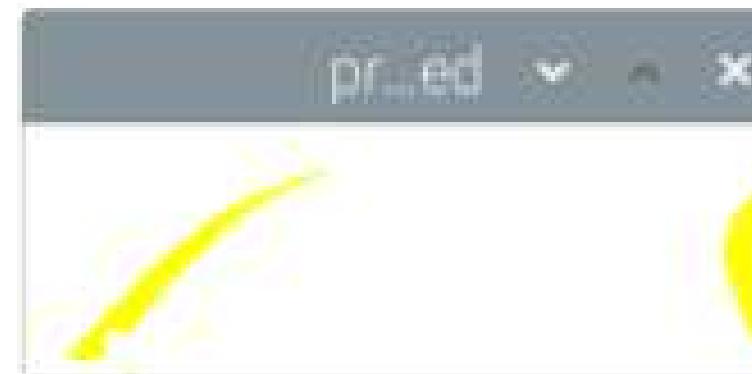
```
_image = cv2.threshold(image,200,255,cv2.THRESH_BINARY_INV)
```

200일때는 노이즈도 많이 줄고 차선인식률도 좋다.

☞ 쇼트컷 예약  
+ 앞선이  
앞아



`_image = cv2.threshold(image,220,255,cv2.THRESH_BINARY_INV)`  
220일 때 노이즈는 많이 줄었으나 차선의 끝쪽이 희미하게 보인다.



`_image = cv2.threshold(image,240,255,cv2.THRESH_BINARY_INV)`  
240일 때 차선도 안보이기 시작한다.



`_image = cv2.threshold(image,250,255,cv2.THRESH_BINARY_INV)`  
250일때 차선도 안보이기 시작한다.  
임계점의 값은 상황에 따라 틀리므로 유동적으로 정해서 사용하면 된다.

임계점을 변경하는 것은 학습데이터를 취득하는 장소의 조명과 주행라인의 빛 반사강도에 따라서 다르게 적용해야 한다. 즉, 차선이 깨끗하게 보이고 노이즈가 최소화된 상태일 때 자율주행 차량의 주행성능도 최대한 향상된다고 볼 수 있다.

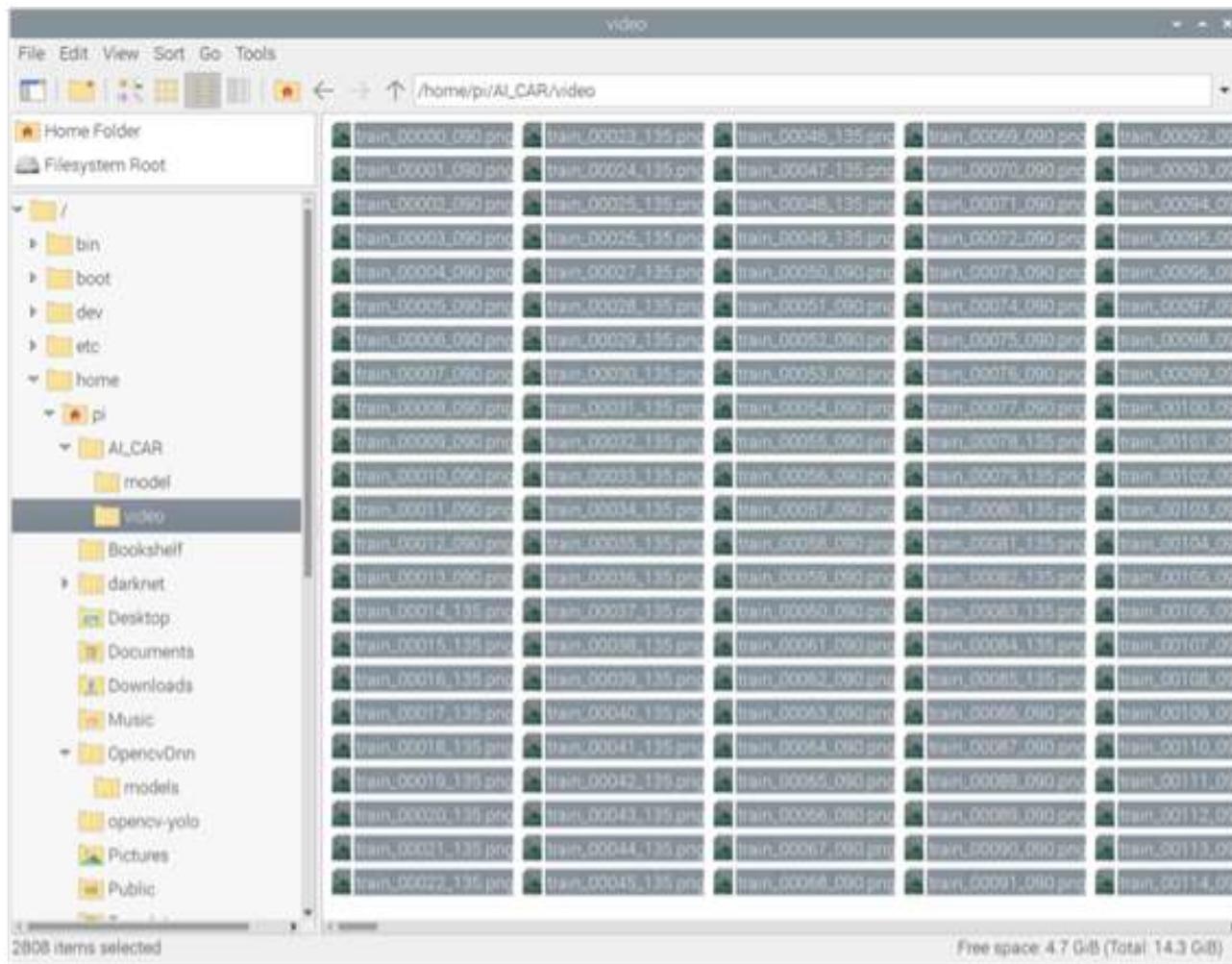
## 2. 필터를 적용한 데이터 획득 코드 - 10\_2.py

```
10_2.py
001 import time
002 import mycamera
003 import cv2
004 from gpiozero import DigitalOutputDevice
005 from gpiozero import PWMOutputDevice
006
007 PWMA = PWMOutputDevice(18)
008 AIN1 = DigitalOutputDevice(22)
009 AIN2 = DigitalOutputDevice(27)
010
011 PWMB = PWMOutputDevice(23)
012 BIN1 = DigitalOutputDevice(25)
013 BIN2 = DigitalOutputDevice(24)
014
015 def motor_on(road):
```

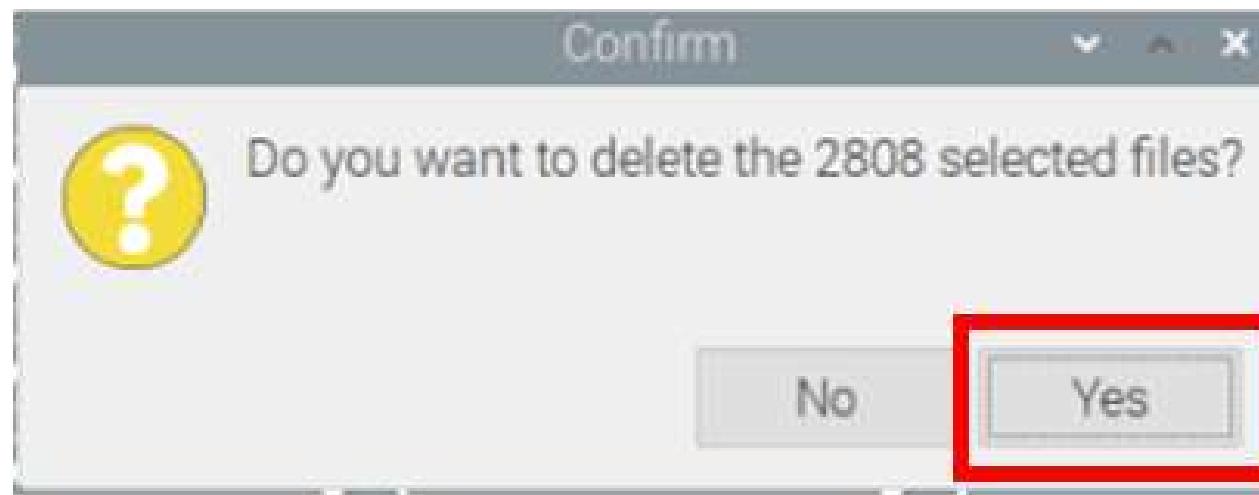
# 更深 의 조 양 을 통 한 자 연 상 이 양 상

3. 데이터 취득 전 기존 /video 폴터의 주행이미지 파일은 삭제한 후 데이터를 취득한다.

1) [컨트럴 + A] 클릭하여 전체 선택



2) [쉬프트 + Delete] 키를 눌러 모두 삭제  
[Yes]를 누르면 모두 삭제된다.



## 4. 취득한 데이터를 이용하여 학습모듈 만들기



비주얼 스튜디오 코드를 이용하여 학습모듈을 생성하고 생성된 모듈을 다시 라즈베리파이로 가지고 온다.

## 5. 필터를 적용한 학습모델로 주행하기 - 10\_4.py

## 10\_4.py

```
001 import threading
002 import time
003 import mycamera
004 import cv2
005 import numpy as np
006 import tensorflow as tf
007 from tensorflow.keras.models import load_model
008 from gpiozero import DigitalOutputDevice
009 from gpiozero import PWMOutputDevice
010
011 PWMA = PWMOutputDevice(18)
012 AIN1 = DigitalOutputDevice(22)
013 AIN2 = DigitalOutputDevice(27)
014
015 PWMB = PWMOutputDevice(23)
```

## 6. 자율주행 자동차의 속도를 조절하여 주행성능 향상 - 10\_5.py

```
10_5.py
001 import threading
002 import time
003 import mycamera
004 import cv2
005 import numpy as np
006 import tensorflow as tf
007 from tensorflow.keras.models import load_model
008 from gpiozero import DigitalOutputDevice
009 from gpiozero import PWMOutputDevice
010
011 PWMA = PWMOutputDevice(18)
012 AIN1 = DigitalOutputDevice(22)
013 AIN2 = DigitalOutputDevice(27)
014
015 D18 = DigitalOutputDevice(26)
```

- 1) 직진 주행성이 빠르다고 해서 주행성능이 100% 뛰어나다고 볼 수 없다.
- 2) 코너의 회전속도는 일정 속도 이상을 초과하는 경우 직진 주행성에도 영향을 줄 수 있다.
- 3) 필터의 적용과 속도의 안정성 두가지를 모두 해결해야 빠르고 정확한 주행이 가능하다.

노력에 따른 결과에는 승자도 패자도 없다.  
시작과 끝만 있을 뿐이다.

마지막까지 모두 화이팅입니다.