# Sol SDK

Edan Chen

edan.chen@ganzin.com.tw

# Outline

- Introduction of Sol SDK
- Sol SDK Installation
- Connect PC to Sol Glasses

# Background of Sol SDK

- Current status: developing in android environment
  - Slow
  - Complex environment
  - Steep learning curve

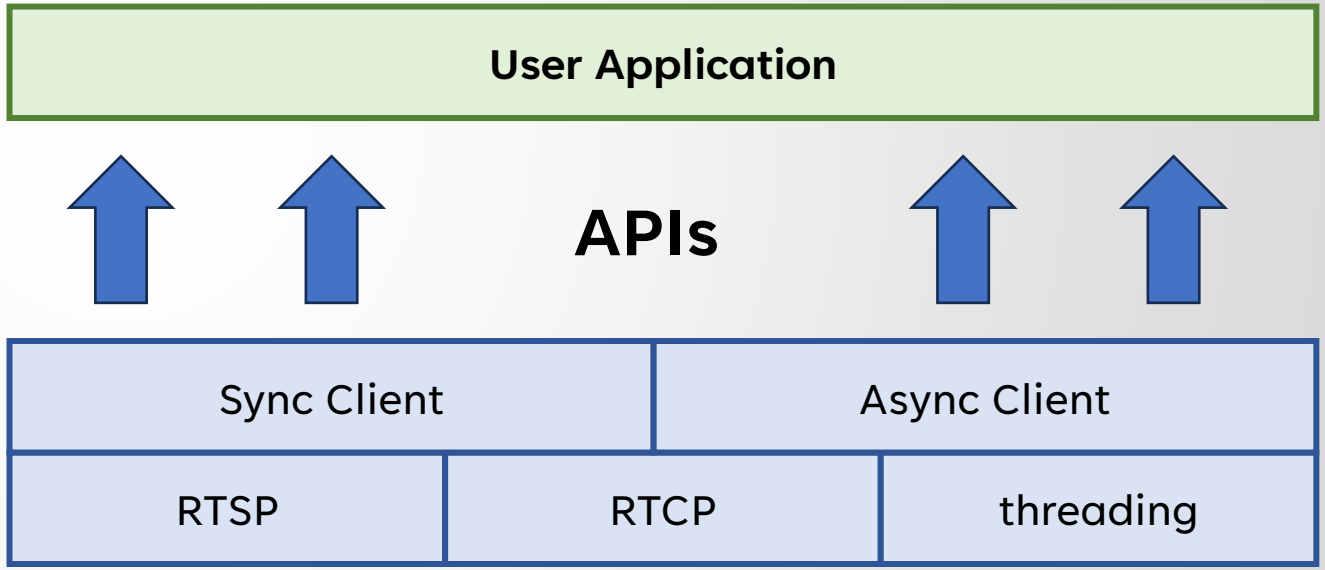
- Target: fast eyetracking application development
  - Python!!!
  - Everyone knows how to use
  - Computer programming in college also teaches Python
  - Easy to use
  - Many other packages can be integrated
  - …

Android

python

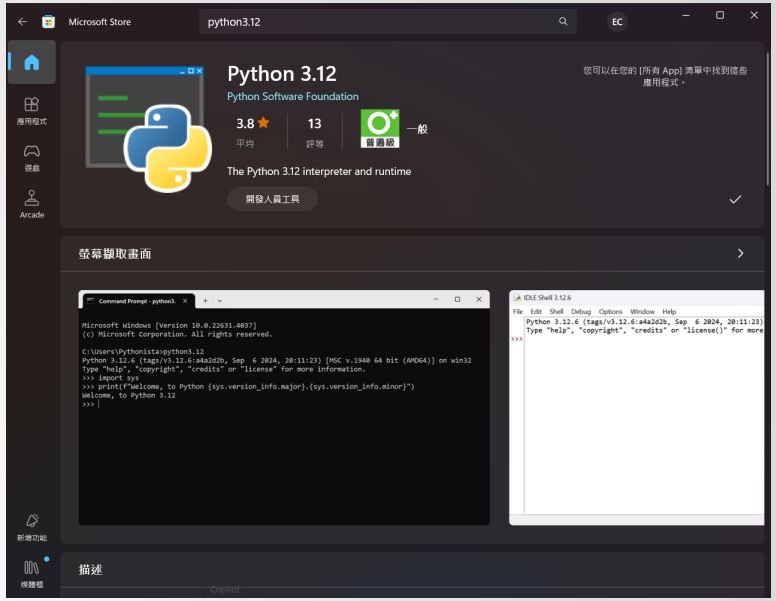
# Architecture of Sol SDK

- Communication protocol: RTSP (Real Time Streaming Protocol), RTCP (Real-time Transport Control Protocol)
- Muti-threading: Python threading library
- Provides two kinds of clients
  - Sync Client
    - Easy to program
    - Good start point
    - Poor performance due to blocking
  - Async Client
    - More complicate
    - More real-time demonstration
- Let’s learn the Sol SDK by going through the example code!!!



# Sol SDK Installation

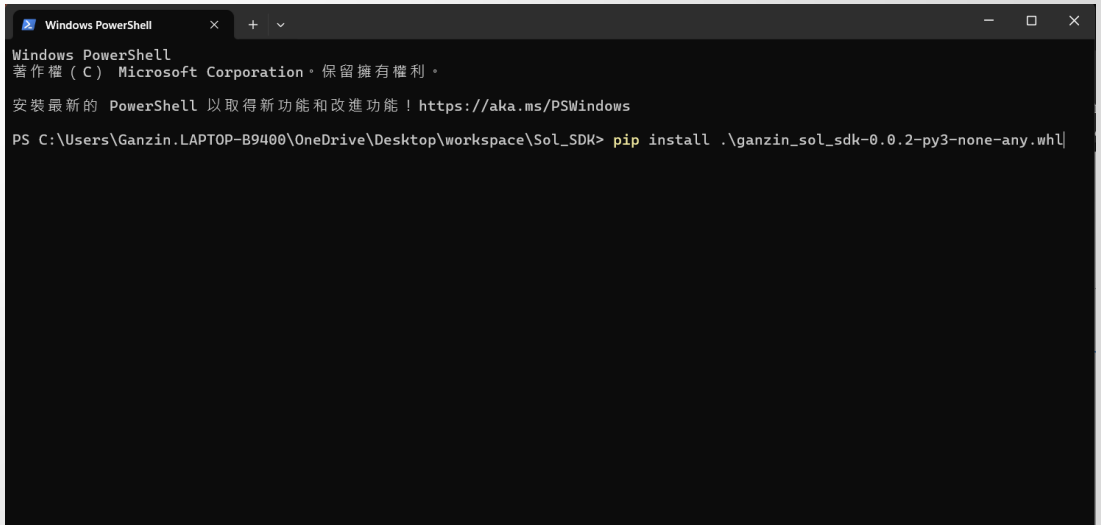
# Installation




- Install Python3 (suggested version: 3.12)
  - https://www.microsoft.com/store/productId/9NCVDN91XZQP?ocid=pdpshare
- Tutorial for installing Python3 with Anaconda on windows
  - https://www.youtube.com/watch?v=4DQGBQMvwZo

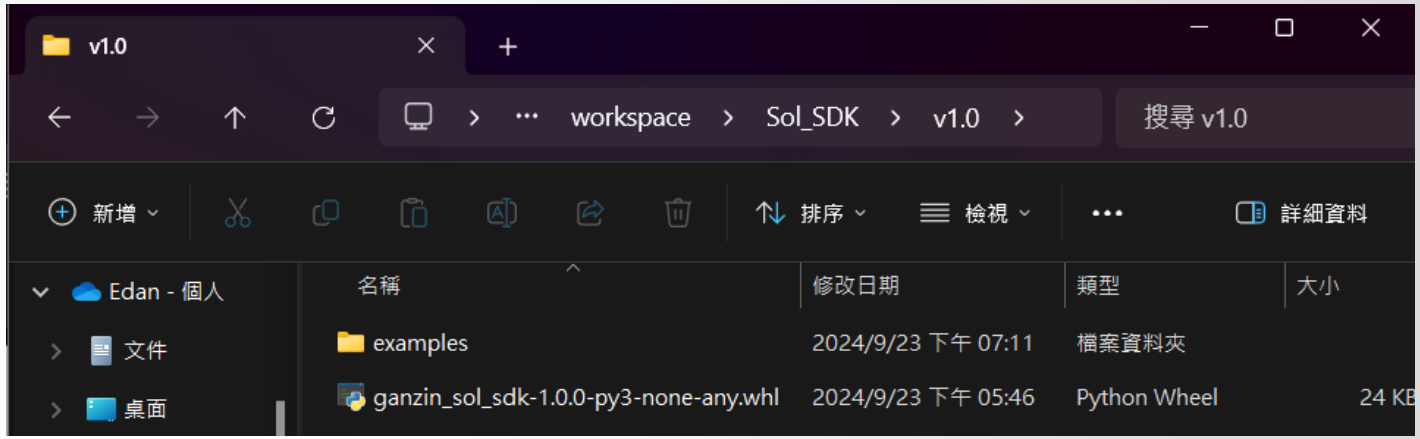
- Download Sol SDK (v1.1.1)
  - whl: https://drive.google.com/file/d/1uSti9pXNiON29pv9qIWC0QnsIf7D8d9p/view?usp=drive_link
  - examples.zip: https://drive.google.com/file/d/1UgQnnOuxsZjPXSgIPZlxF-WOsaKLD8Ym/view?usp=drive_link
  - Document: https://drive.google.com/file/d/1CWCOxSLSHWKYiONA0TkwncyMkE46B8d-/view?usp=drive_link

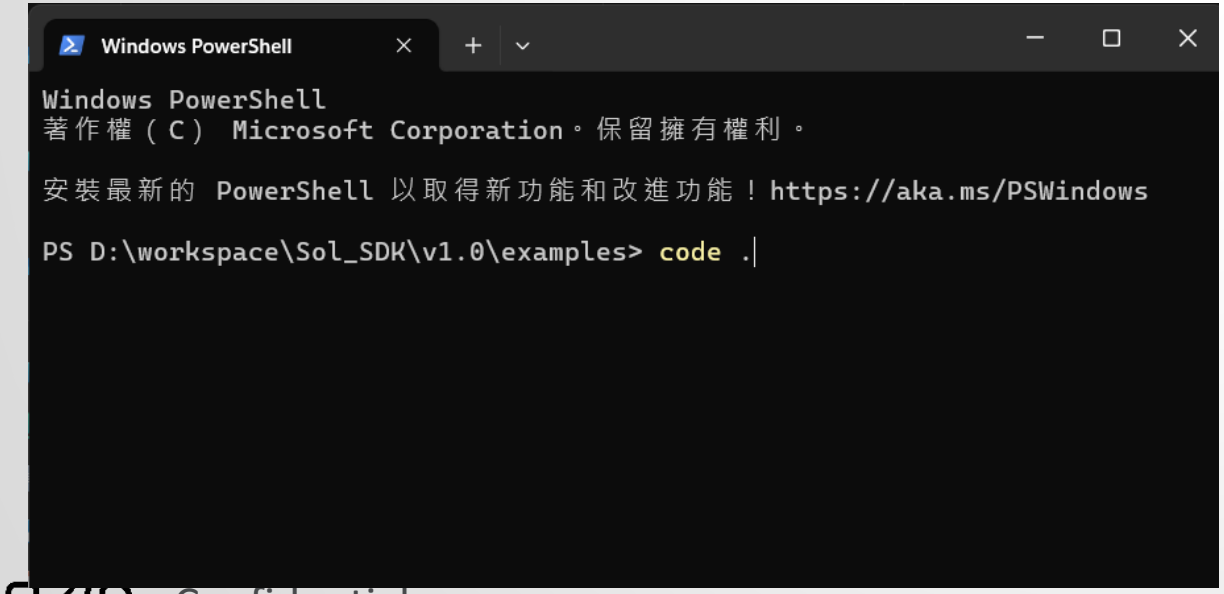
- Install SDK
  - In the Sol SDK directory
    - Right click to open the powershell
  - *$ pip install .\ganzin_sol_sdk-1.1.1-py3-none-any.whl*
- Install other packages will used later
  - *$ pip install requests opencv-python*



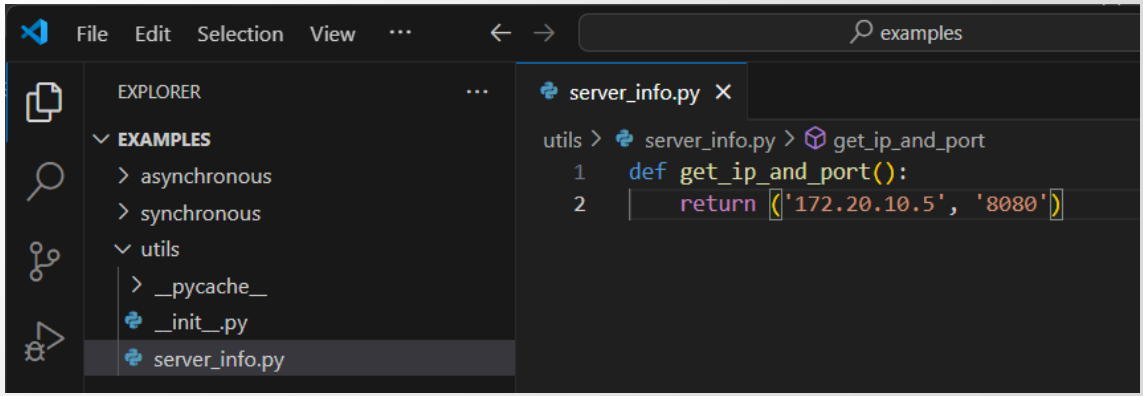
# Dive Into Examples

- Extract examples.zip
  - Check the *examples* directory exist
- Open vscode in the examples directory
  - Go into the examples directory
  - Right click and open powershell
  - *$ code .*




- Check the hierarchy in vscode

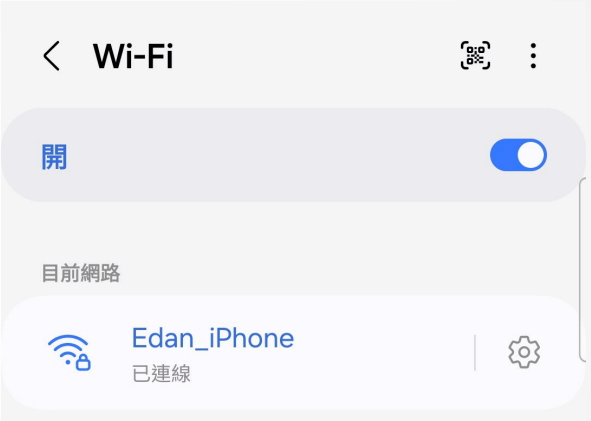


# Connect PC to Sol Glasses

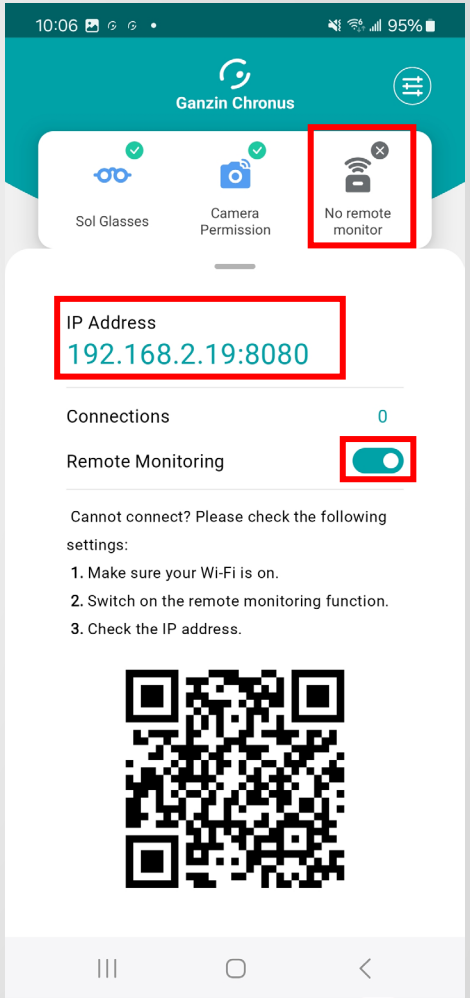
Previewing the gaze video

# Setup Environment (1/2)

- Connect Sol Glasses's phone and your PC under the same AP (e.g., your own hotspot)


- Open the Chronus app and click the remote monitor icon
  - Get the IP and Port from the phone screen
  - The remote monitoring toggle needs to be on



# Setup Environment (2/2)

- Fill the ip and port into the *utils/server_info.py*

server_info.py

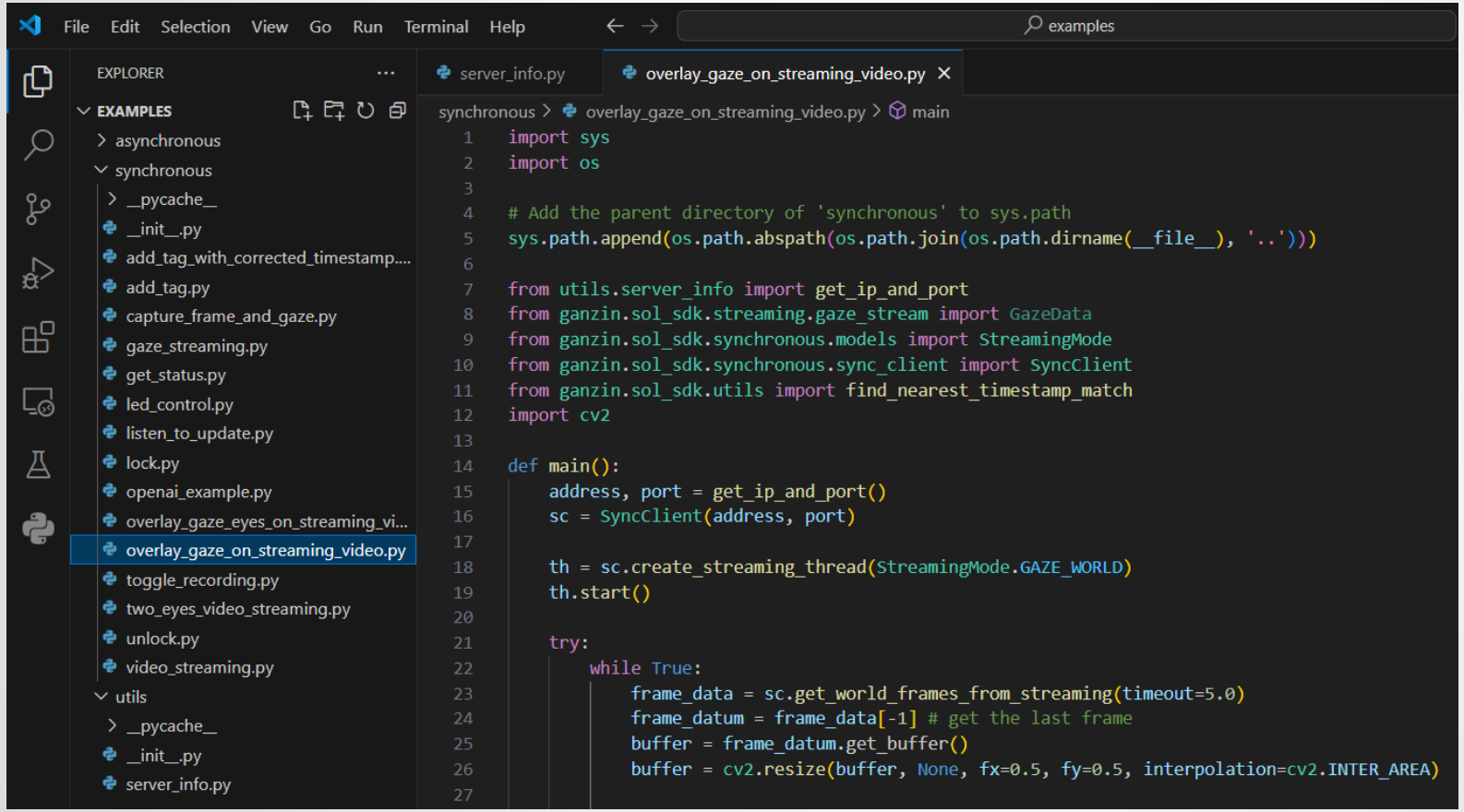
utils > server_info.py > get_ip_and_port

```python
def get_ip_and_port():
    return ('172.20.10.5', '8080')
```

- Please notice that when using the Sol SDK, the Chronus app need to stay on the foreground

# Dive into the Gaze Video Example

- Open the *synchronous/overlay_gaze_on_streaming_video.py* example



# Dive into the Code (1/4)

- Import modules
  - sys, os: to make the server_info importable
  - ganzin.sol_sdk: the sdk to remote control the Sol Glasses
  - cv2: visualize the result get from the Sol SDK

```
import sys
import os

# Add the parent directory of 'synchronous' to sys.path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))

from utils.server_info import get_ip_and_port
from ganzin.sol_sdk.streaming.gaze_stream import GazeData
from ganzin.sol_sdk.synchronous.models import StreamingMode
from ganzin.sol_sdk.synchronous.sync_client import SyncClient
from ganzin.sol_sdk.utils import find_nearest_timestamp_match
import cv2
```

# Dive into the Code (2/4)

- Sync Client setup
  - Get the address and port from *get_ip_and_port()*
  - Create a Sync Client using the address and port
  - Create a streaming thread for the Sync Client
  - Start the thread

```python
def main():
    address, port = get_ip_and_port()
    sc = SyncClient(address, port)

    th = sc.create_streaming_thread(StreamingMode.GAZE_WORLD)
    th.start()

```

# Dive into the Code (3/4)

- Getting the gazes and the frames
  - Create an infinite while loop to keep getting the gazes and frames
  - Get the frame data from *get_world_frames_from_streaming()*
  - Get the gazes data from *get_gazes_from_streaming()*
  - Get the exactly gaze using the timestamp of the frame

```
try:
    while True:
        frame_data = sc.get_world_frames_from_streaming(timeout=5.0)
        frame_datum = frame_data[-1] # get the last frame
        buffer = frame_datum.get_buffer()

        gazes = sc.get_gazes_from_streaming(timeout=5.0)
        gaze = find_nearest_timestamp_match(frame_datum.get_timestamp(), gazes)
```

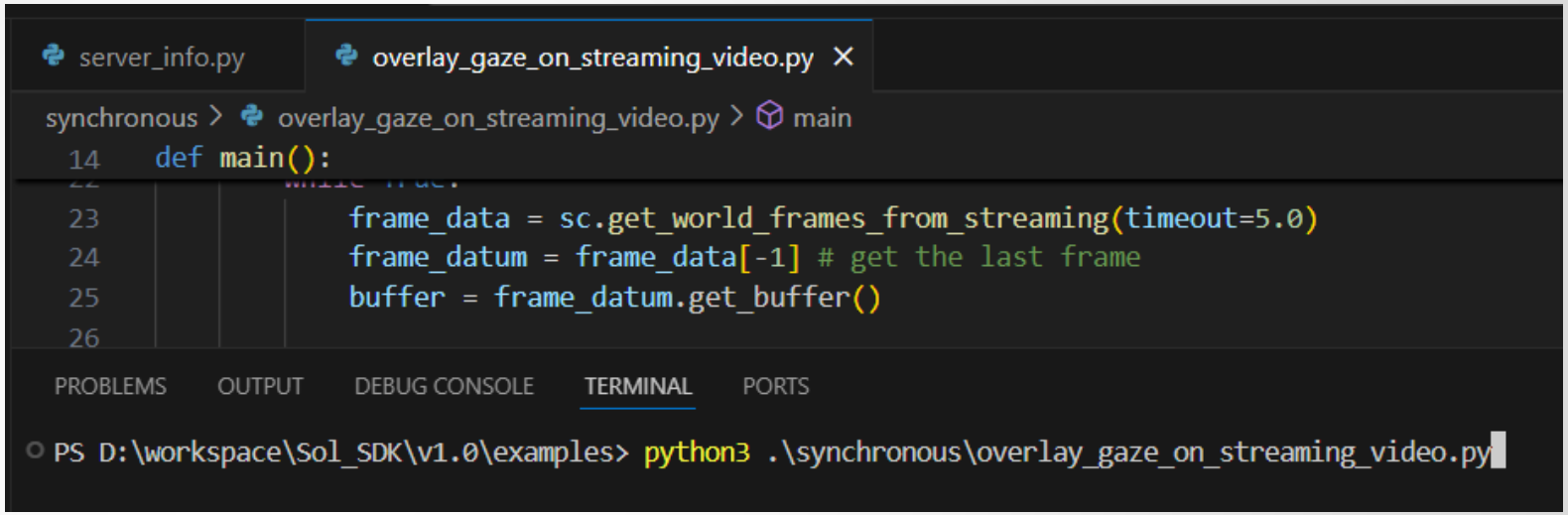
# Dive into the Code (4/4)

- Show the result using cv2
  - Resize the buffer as the resolution of the Sol Glasses front facing camera is 1328x1200 (exceed 1080)
  - Calculate the center of the circle
  - Set the circle's radius, color, and thickness
  - Draw the circle using *cv2.circle()*
  - Show the result using *cv2.imshow()*

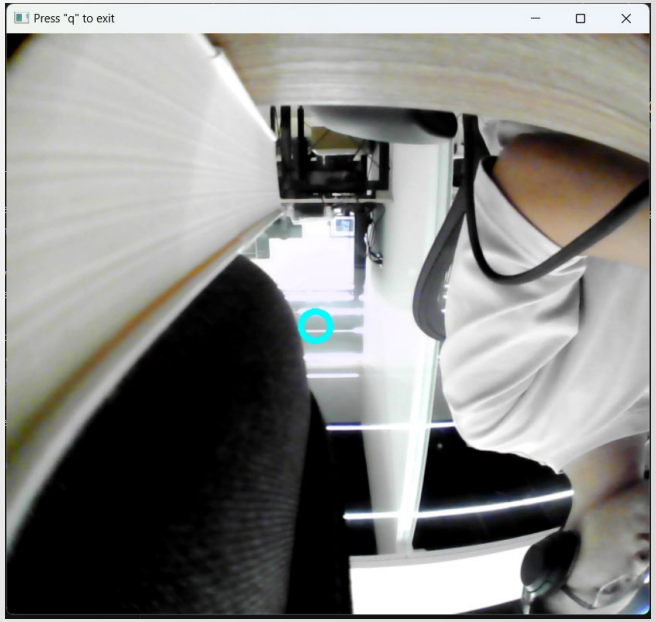
```
buffer = cv2.resize(buffer, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_AREA)
center = (int(gaze.combined.gaze_2d.x/2), int(gaze.combined.gaze_2d.y/2))
radius = 15
bgr_color = (255, 255, 0)
thickness = 5
cv2.circle(buffer, center, radius, bgr_color, thickness)

cv2.imshow('Press "q" to exit', buffer)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

# Run the Example

- Open terminal from the vscode

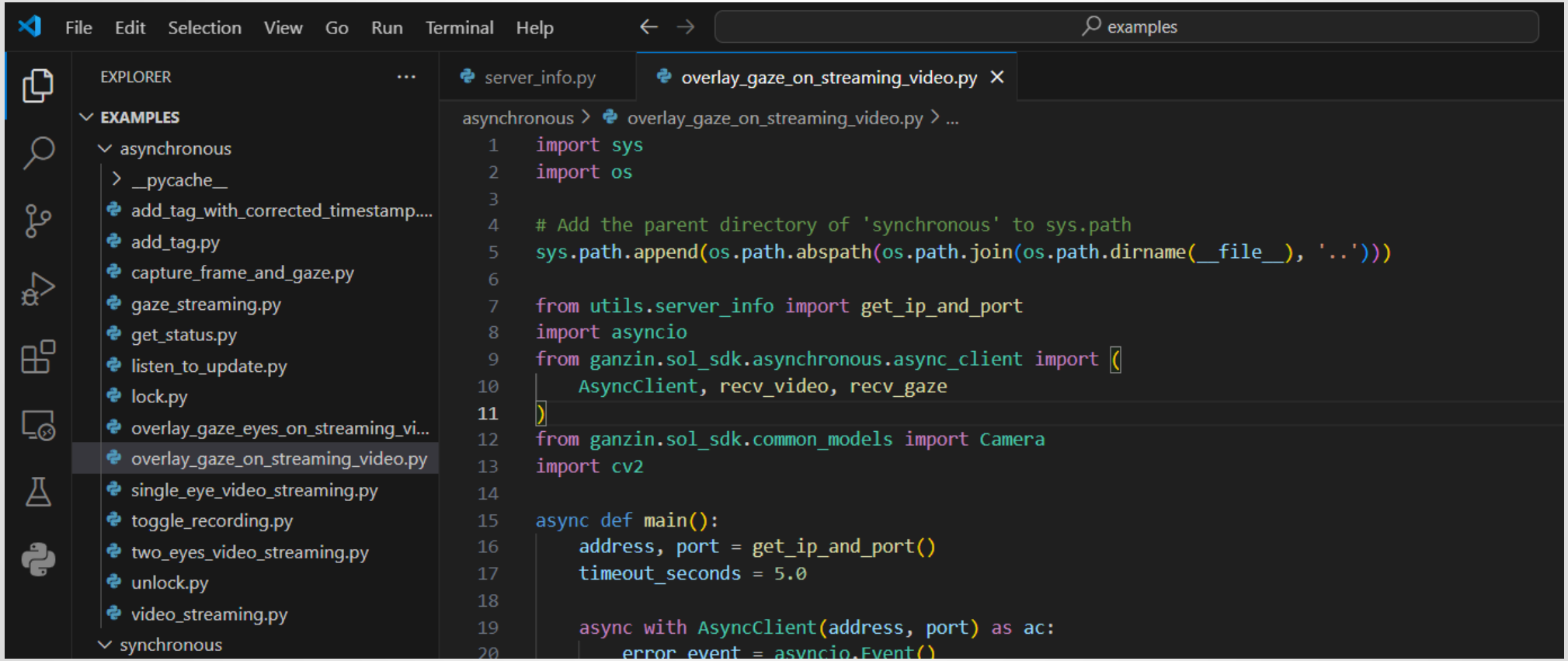



- *$ python3 .\synchronous\overlay_gaze_on_streaming_video.py*
  - Check whether the streaming window appear
  - Press ‘q’ to exit



# Dive into the Async Counterpart

- Open the *asynchronous/overlay_gaze_on_streaming_video.py* example



# Dive into the Code (1/5)

- Module import
  - asyncio: modules for asynchronous programming, enables the execution of I/O-bound tasks without blocking the execution of other tasks.
  - ganzin.sol_sdk: import the AsyncClient

```python
import sys
import os

# Add the parent directory of 'synchronous' to sys.path
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), '..')))

from utils.server_info import get_ip_and_port
import asyncio
from ganzin.sol_sdk.asynchronous.async_client import (
    AsyncClient, recv_video, recv_gaze
)
from ganzin.sol_sdk.common_models import Camera
import cv2
```

# Dive into the Code (2/5)

- Main function
  - Create two queues: frames, gazes, for putting the frame/gaze into the queue
  - Create two tasks: collect_video_task, collect_gaze_task, for putting the frame get from the AsyncClient into the queue
  - Await for the *draw_gaze_on_frame()*, draw the frame and gaze when there is gaze/frame in the queues

```python
async def main():
    address, port = get_ip_and_port()
    timeout_seconds = 5.0

    async with AsyncClient(address, port) as ac:
        error_event = asyncio.Event()

        frames = asyncio.Queue(1)
        collect_video_task = asyncio.create_task(keep_last_video_frame(ac, frames, error_event))

        gazes = asyncio.Queue()
        collect_gaze_task = asyncio.create_task(collect_gaze(ac, gazes, error_event, timeout_seconds))

        try:
            await draw_gaze_on_frame(frames, gazes, error_event, timeout_seconds)
        finally:
            collect_video_task.cancel()
            collect_gaze_task.cancel()
```

# Dive into the Code (3/5)

- *keep_last_video_frame()*
  - async for loop for getting frame from *recv_video()*
    - This will be blocked by the IO
  - Remove the item in queue if full ASAP, the queue has only one seat
  - Put the frame into the queue ASAP
- *collect_gaze()*
  - async for loop for getting gaze from *rec_gaze()*
  - put the gaze into the queue
    - The queue can preserve multiple gazes

```python
async def keep_last_video_frame(ac: AsyncClient, queue: asyncio.Queue, error_event: asyncio.Event) -> None:
    async for frame in recv_video(ac, Camera.WORLD):
        if error_event.is_set():
            break

        if queue.full():
            queue.get_nowait()
        queue.put_nowait(frame)

async def collect_gaze(ac: AsyncClient, queue: asyncio.Queue, error_event: asyncio.Event, timeout) -> None:
    try:
        async for gaze in recv_gaze(ac):
            if error_event.is_set():
                break

            await asyncio.wait_for(queue.put(gaze), timeout=timeout)
    except Exception as e:
        error_event.set()
```

# Dive into the Code (4/5)

- *draw_gaze_on_frame()*
  - Get the frame from the *frame_queue*
  - Get the gaze from the *gaze_queue*
  - Draw the gaze on the frame using cv2

```python
53  async def draw_gaze_on_frame(frame_queue, gazes, error_event: asyncio.Event, timeout):
54      while not error_event.is_set():
55          frame = await get_video_frame(frame_queue, timeout)
56          gaze = await find_gaze_near_frame(gazes, frame.get_timestamp(), timeout)
57          frame_buffer = frame.get_buffer()
58          frame_buffer = cv2.resize(frame_buffer, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_AREA)
59
60          center = (int(gaze.combined.gaze_2d.x/2), int(gaze.combined.gaze_2d.y/2))
61          radius = 15
62          bgr_color = (255, 255, 0)
63          thickness = 3
64          cv2.circle(frame_buffer, center, radius, bgr_color, thickness)
65
66          cv2.imshow('Press "q" to exit', frame_buffer)
67          if cv2.waitKey(1) & 0xFF == ord('q'):
68              return
```

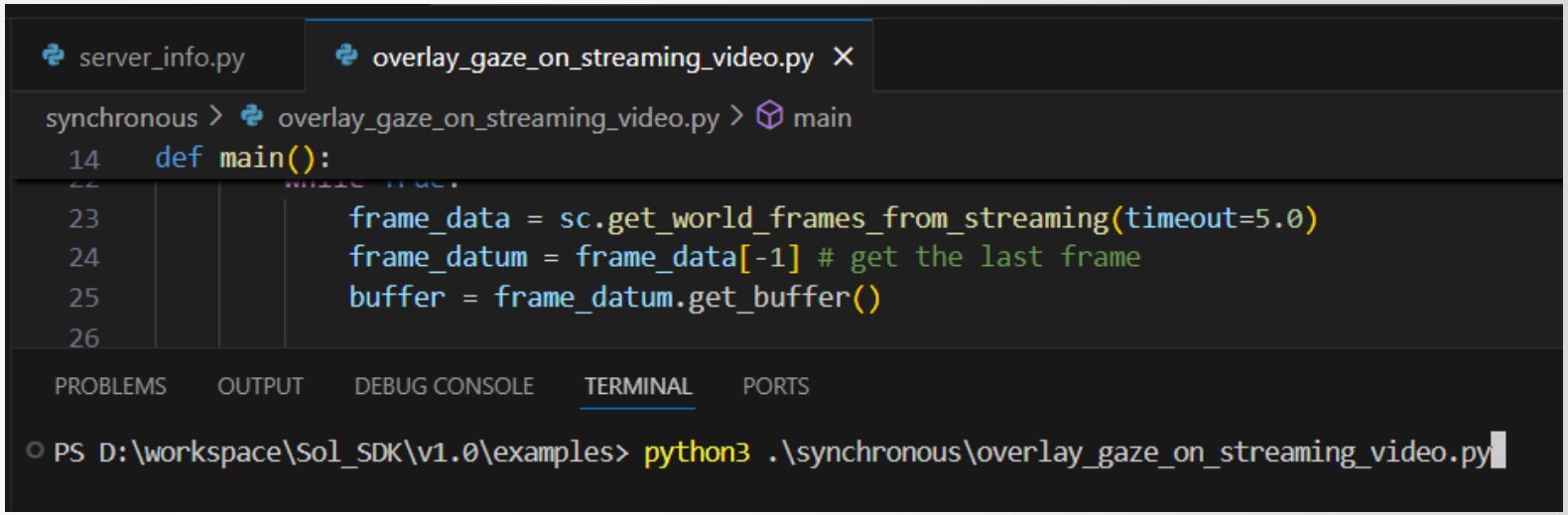
# Dive into the Code (5/5)

- *get_video_frame()*
  - Async function for getting the frame out of the queue
- *find_gaze_near_frame()*
  - Async function for finding the most appropriate gaze in the gaze queue

```python
async def get_video_frame(queue, timeout):
    return await asyncio.wait_for(queue.get(), timeout=timeout)

async def find_gaze_near_frame(queue, timestamp, timeout):
    item = await asyncio.wait_for(queue.get(), timeout=timeout)
    if item.get_timestamp() > timestamp:
        return item

    while True:
        if queue.empty():
            return item
        else:
            next_item = queue.get_nowait()
            if next_item.get_timestamp() > timestamp:
                return next_item
            item = next_item
```

# Run the Example

- Open terminal from the vscode




- *$ python3 .\asynchronous\overlay_gaze_on_streaming_video.py*
  - Check whether the streaming window appear
  - Press ‘q’ to exit
  - Same result from the synchronous counterpart