To achieve this, we'll create a **headless webcam streaming script** that:

✅ **Runs in the background** (no display window).  
✅ **Streams the video feed to a remote computer** over a network.

**Approach:**

1. **Capture the webcam feed** using OpenCV.
2. **Send the feed to a remote machine** using **socket streaming**.
3. **Run the script in the background** (as a hidden process).

**1️⃣ On Your Windows Laptop (Sender)**

This script captures video and streams it to a remote computer.

**Install dependencies**

pip install opencv-python numpy

**Run this on the Windows machine (Sender):**

import cv2

import socket

import struct

import pickle

# Setup socket connection

HOST = "REMOTE\_IP\_ADDRESS" # Replace with your remote computer's IP

PORT = 9999 # Port to send video

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.connect((HOST, PORT))

cap = cv2.VideoCapture(0) # Open webcam

while True:

ret, frame = cap.read()

if not ret:

break

# Serialize frame

data = pickle.dumps(frame)

message\_size = struct.pack("Q", len(data))

# Send frame to remote computer

server\_socket.sendall(message\_size + data)

cap.release()

server\_socket.close()

🔹 **Replace REMOTE\_IP\_ADDRESS** with your actual remote machine's IP.  
🔹 Run this script on your **Windows laptop** (Sender).

**2️⃣ On Your Remote Computer (Receiver)**

This script receives and displays the webcam feed.

**Run this on the Remote Machine (Receiver)**

import cv2

import socket

import struct

import pickle

# Setup socket server

HOST = "0.0.0.0" # Listen on all available network interfaces

PORT = 9999 # Same port as the sender

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind((HOST, PORT))

server\_socket.listen(1)

conn, addr = server\_socket.accept()

print(f"Connection from {addr}")

data = b""

payload\_size = struct.calcsize("Q")

while True:

while len(data) < payload\_size:

packet = conn.recv(4 \* 1024) # Receive data in chunks

if not packet:

break

data += packet

packed\_msg\_size = data[:payload\_size]

data = data[payload\_size:]

msg\_size = struct.unpack("Q", packed\_msg\_size)[0]

while len(data) < msg\_size:

data += conn.recv(4 \* 1024)

frame\_data = data[:msg\_size]

data = data[msg\_size:]

# Deserialize frame

frame = pickle.loads(frame\_data)

cv2.imshow("Remote Webcam Feed", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

conn.close()

cv2.destroyAllWindows()

**Running in the Background (Headless Mode)**

To run the **sender script in the background** on Windows:

1. Save webcam\_sender.py.
2. Open **Command Prompt (cmd)** and run:
3. pythonw webcam\_sender.py

🔹 The pythonw command runs the script without opening a window.

**Final Steps**

✔ **Run the Receiver Script** on your remote machine.  
✔ **Start the Sender Script** on your laptop.  
✔ **Watch the live stream from your webcam remotely!**

💡 **Do you want it to start automatically on boot?** I can help you set it up! 🚀

To **run your script in the background** without showing a terminal window, use one of these methods:

**🎯 Method 1: Use pythonw.exe (No Terminal Window)**

Python provides pythonw.exe, which runs scripts **without opening a command window**.

**How to Use It**

1. **Modify the Task Scheduler / Startup Folder command** to:
2. pythonw "C:\path\to\webcam\_streamer.py"

✅ This runs your script **silently** in the background.

**🎯 Method 2: Convert Python Script to a Background .exe**

You can convert webcam\_streamer.py into a **background executable file**.

**Steps:**

1. **Install pyinstaller**
2. pip install pyinstaller
3. **Convert your script into a background .exe**
4. pyinstaller --noconsole --onefile webcam\_streamer.py
   * --noconsole hides the terminal window.
   * --onefile creates a single .exe file.
5. **Find the .exe** in the dist/ folder and move it to:
6. C:\path\to\webcam\_streamer.exe
7. **Modify Startup / Task Scheduler to use the .exe instead of .py**.

✅ **Your script will now run at startup without showing any window.**

**🎯 Best Option for You?**

✔ **Quick solution:** Use pythonw.exe.  
✔ **Best for background execution:** Convert to .exe with pyinstaller.

Would you like to also make it **auto-restart if it crashes**? 🚀

To **automatically restart your script if it crashes**, you have two solid approaches:

**🎯 Method 1: Use a Loop in the Script (Simple & Effective)**

Modify webcam\_streamer.py to restart itself if it crashes:

**Modify Your Script:**

import subprocess

import time

while True:

try:

subprocess.run(["python", "actual\_webcam\_script.py"], check=True)

except subprocess.CalledProcessError:

print("Script crashed! Restarting...")

time.sleep(5) # Wait before restarting

🔹 **How it Works:**

* Runs actual\_webcam\_script.py (your real script).
* If it crashes, it waits 5 seconds and restarts.
* Runs **silently in the background** if started with pythonw.exe.

**🎯 Method 2: Use Windows Task Scheduler with Auto-Restart**

This method **automatically restarts the script on failure** using Task Scheduler.

**Modify Task Scheduler Task:**

1. **Open Task Scheduler (Win + R → taskschd.msc)**
2. **Right-click "Webcam Auto-Start" task** → Click **Properties**
3. Go to **Settings** tab:
   * Check ✅ **"Restart the task if it fails"**
   * Set **Restart every:** 1 minute
   * Set **Attempt restart up to:** 5 times

Now, if the script crashes, **Windows will restart it automatically!**

**🎯 Best Option for You?**

✔ **If you want a Python-only fix:** Use **Method 1 (Loop in Script)**.  
✔ **If you want Windows to handle everything:** Use **Method 2 (Task Scheduler)**.  
✔ **For best reliability:** **Combine both methods!** 🚀

Would you like to **log errors** when the script crashes for debugging?

To **run your script in the background** without showing a terminal window, use one of these methods:

**🎯 Method 1: Use pythonw.exe (No Terminal Window)**

Python provides pythonw.exe, which runs scripts **without opening a command window**.

**How to Use It**

1. **Modify the Task Scheduler / Startup Folder command** to:
2. pythonw "C:\path\to\webcam\_streamer.py"

✅ This runs your script **silently** in the background.

**🎯 Method 2: Convert Python Script to a Background .exe**

You can convert webcam\_streamer.py into a **background executable file**.

**Steps:**

1. **Install pyinstaller**
2. pip install pyinstaller
3. **Convert your script into a background .exe**
4. pyinstaller --noconsole --onefile webcam\_streamer.py
   * --noconsole hides the terminal window.
   * --onefile creates a single .exe file.
5. **Find the .exe** in the dist/ folder and move it to:
6. C:\path\to\webcam\_streamer.exe
7. **Modify Startup / Task Scheduler to use the .exe instead of .py**.

✅ **Your script will now run at startup without showing any window.**

**🎯 Best Option for You?**

✔ **Quick solution:** Use pythonw.exe.  
✔ **Best for background execution:** Convert to .exe with pyinstaller.

Would you like to also make it **auto-restart if it crashes**? 🚀