**Socket UDP**

**Moises Ceniceros Carrillos  
Joseph de Jesus Lemus Pineda  
Oliver Carrillo Cariño  
Hugo Gerardo Macias Navarrete**

# 

# Códigos de ejecución y prueba:

Los siguientes códigos ejecutados en visual studio simulan sockets de cliente y servidor orientado a comunicación UDP.

# Código del Cliente:

## En un nuevo archivo en la misma carpeta que el anterior escribimos:

## #Codigo del Cliente

## 

## import socket

## import time

## 

## def main():

## print("="\*50)

## print("🌐 CLIENTE UDP - CONFIGURACIÓN INICIAL")

## print("="\*50)

## 

## try:

## # Configuración con valores por defecto

## SERVER\_HOST = input("IP del servidor [localhost]: ") or "127.0.0.1"

## SERVER\_PORT = int(input("Puerto del servidor [12345]: ")) or 12345

## TIMEOUT = 5

## 

## client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

## client\_socket.settimeout(TIMEOUT)

## 

## print("\n" + "="\*50)

## print(f"✅ Conectado a {SERVER\_HOST}:{SERVER\_PORT}")

## print("Escriba 'exit' para salir")

## print("="\*50 + "\n")

## 

## while True:

## try:

## message = input(">> ")

## if message.lower() == 'exit':

## break

## 

## start\_time = time.time()

## client\_socket.sendto(message.encode('utf-8'), (SERVER\_HOST, SERVER\_PORT))

## 

## try:

## response, \_ = client\_socket.recvfrom(1024)

## elapsed = round((time.time() - start\_time) \* 1000, 2)

## print(f"\n🟢 Respuesta ({elapsed} ms): {response.decode()}\n")

## except socket.timeout:

## print(f"\n🔴 Error: Timeout ({TIMEOUT}s) - Servidor no responde\n")

## 

## except KeyboardInterrupt:

## print("\nOperación cancelada")

## break

## 

## except ValueError:

## print("Error: Puerto debe ser un número entero")

## except Exception as e:

## print(f"Error crítico: {e}")

## finally:

## client\_socket.close()

## print("\nConexión cerrada")

## 

## if \_\_name\_\_ == "\_\_main\_\_":

## main()

## 

## 

## Código del Servidor:

# Escribimos en un archivo en la carpeta recién creada:”Comunicación\_UDP”.

# #Codigo del servidor

# 

# import socket

# import signal

# import sys

# from datetime import datetime

# 

# # Configuración del servidor

# SERVER\_HOST = '0.0.0.0' # Escucha en todas las interfaces

# SERVER\_PORT = 12345

# BUFFER\_SIZE = 1024

# 

# # Estadísticas

# stats = {

# 'total\_clients': 0,

# 'total\_bytes': 0,

# 'start\_time': datetime.now(),

# 'messages': {}

# }

# 

# # Manejo de Ctrl+C

# def signal\_handler(sig, frame):

# print("\n\n--- Estadísticas Finales ---")

# print(f"Tiempo activo: {datetime.now() - stats['start\_time']}")

# print(f"Clientes atendidos: {stats['total\_clients']}")

# print(f"Bytes recibidos: {stats['total\_bytes']}")

# print("\nDetalle de mensajes:")

# for addr, count in stats['messages'].items():

# print(f" {addr}: {count} mensajes")

# 

# print("\nCerrando servidor...")

# server\_socket.close()

# sys.exit(0)

# 

# signal.signal(signal.SIGINT, signal\_handler)

# 

# # Crear socket UDP IPv4

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

# server\_socket.bind((SERVER\_HOST, SERVER\_PORT))

# 

# print(f"🚀 Servidor UDP escuchando en {SERVER\_HOST}:{SERVER\_PORT}")

# print("Presione Ctrl+C para detener el servidor\n")

# 

# try:

# while True:

# # Recibir datagrama

# data, client\_addr = server\_socket.recvfrom(BUFFER\_SIZE)

# client\_ip, client\_port = client\_addr

# 

# # Actualizar estadísticas

# stats['total\_bytes'] += len(data)

# if client\_addr not in stats['messages']:

# stats['total\_clients'] += 1

# stats['messages'][client\_addr] = 0

# stats['messages'][client\_addr] += 1

# 

# # Decodificar y mostrar mensaje

# message = data.decode('utf-8')

# timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# print(f"[{timestamp}] Mensaje de {client\_ip}:{client\_port} -> {message}")

# 

# # Enviar confirmación

# ack\_msg = f"ACK: Mensaje recibido ({len(data)} bytes)"

# server\_socket.sendto(ack\_msg.encode('utf-8'), client\_addr)

# 

# except Exception as e:

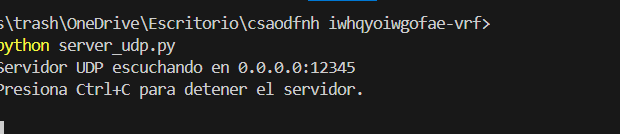
# print(f"Error: {e}")

# finally:

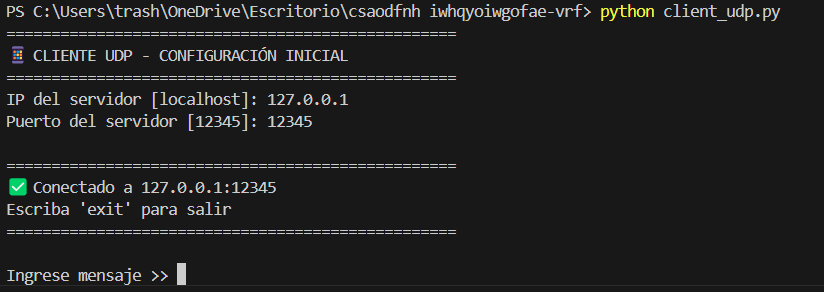
# server\_socket.close()

# Capturas de ejecución

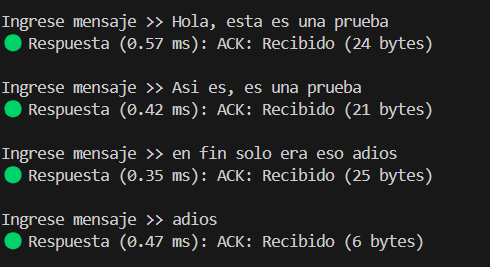
Tras poner en la terminal “python server\_udp.py”, se activa automáticamente el servidor.



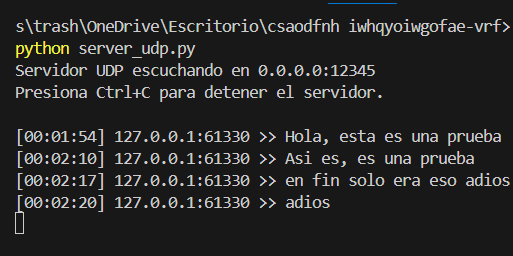
Creamos una nueva terminal y en ella escribimos “python client\_udp.py”, tras escribirlo se activará el cliente y tendremos que poner una dirección ip predeterminada como un port para conectarnos al servidor.



Iniciamos una prueba de mensajería entre el cliente y el servidor.



El servidor recibe los mensajes correctamente, con eso podemos detener el servidor con ctrl + c

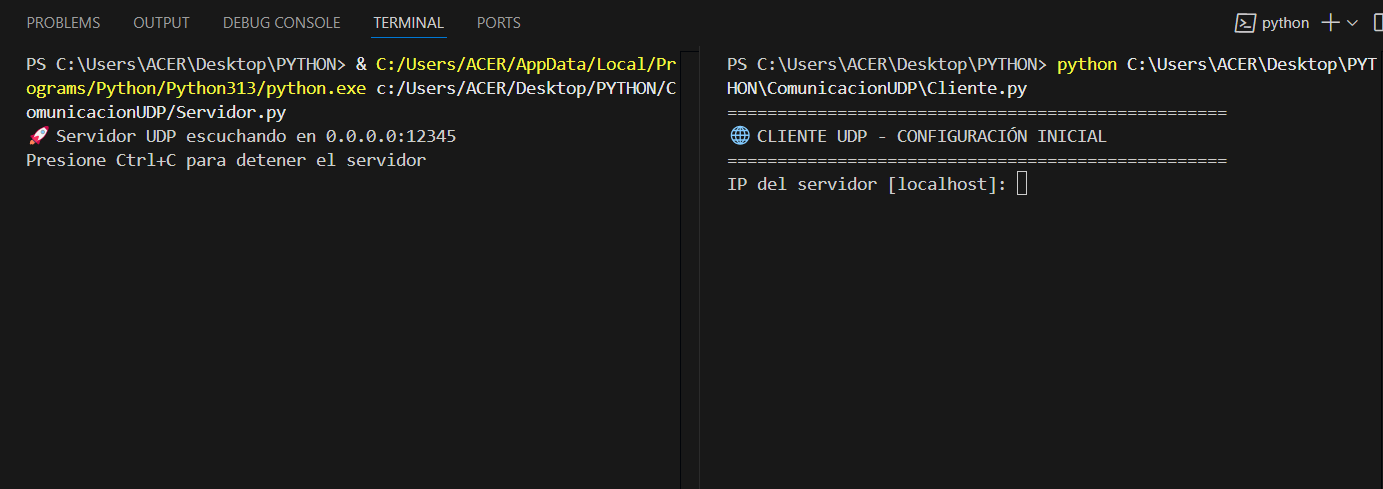


**Socket UDP, Segunda prueba**

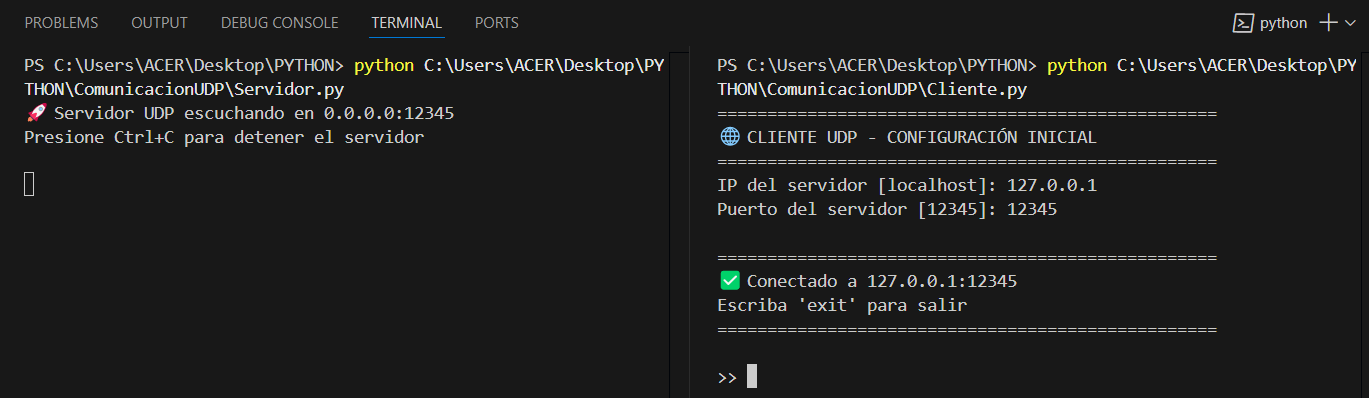
### Capturas de ejecución en paralelo:

Una vez ejecutado ambos códigos en terminales distintas podemos ver:

Del lado izquierdo vemos que está activado el servidor, del lado derecho vemos que el cliente va a hacer conexión con el servidor.



Conectamos el cliente con el servidor:



Iniciamos prueba de mensajería entre cliente y servidor:

