



**Facultad de
Ingeniería**
Universidad Católica de la Santísima Concepción

UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN
FACULTAD DE INGENIERÍA
INGENIERÍA CIVIL INFORMÁTICA
IN1082C – REDES DE COMPUTADORES

Tarea 2:

Desarrollo de aplicación Cliente-Servidor mediante el uso de Sockets de Berkeley

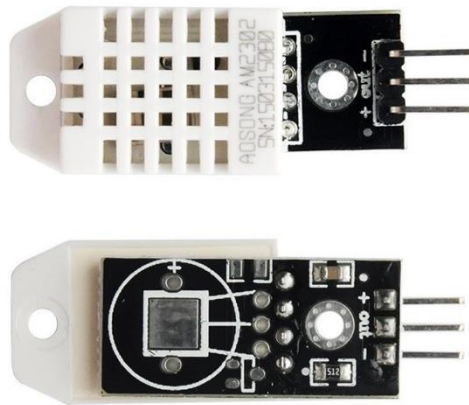
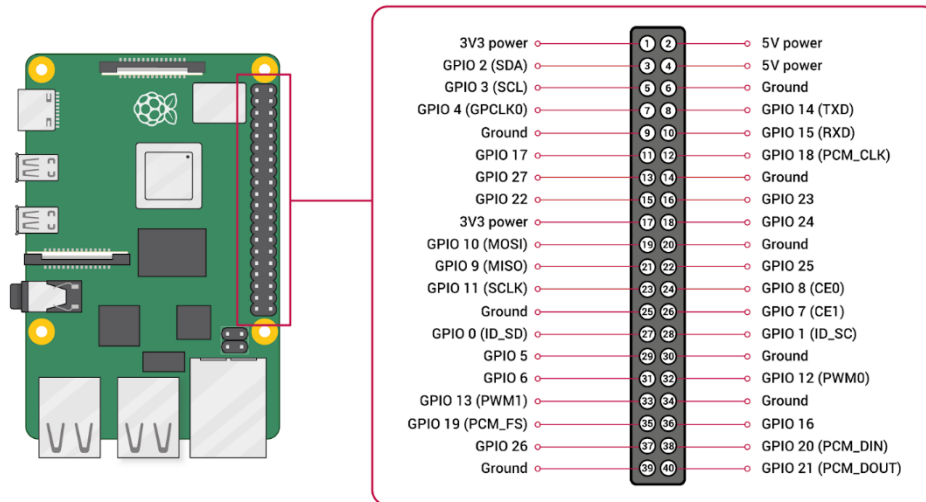
Integrante: Felipe Valentín Muñoz.

Profesor: Yasmany Prieto Hernández.

Concepción, 22 de junio 2022

¿En qué consiste la aplicación?

Consiste en recibir datos desde una Raspberry Pi (3B+) en la cual se encuentra conectado un Sensor de Temperatura y Humedad (DHT22) de forma directa en su puerto GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General). Estos datos son comunicados a través de una conexión por Socket TCP/IP con un dispositivo y aplicación Cliente.



Utilidad y Motivación

La posibilidad de poder simular una situación cercana a un proyecto real, pero a una complejidad más simple, en este caso podría ser una estación meteorológica, donde los datos que se recopilen puedan ser almacenados para luego ser usados en modelos de pronóstico, de acuerdo con el historial y la situación actual. También se puede emplear para el monitoreo de confort térmico, control de calderas, o lugares que requieran control de temperatura. De forma personal, el poder interactuar con el medio a través de sensores, me parece interesante, sobre todo donde el IoT (Internet of Things, Internet de las cosas) poco a poco se ha ido siendo parte de nuestro día a día, desde nuestro teléfono controlar a distancia cosas como televisor, refrigerador, aspiradoras o cualquier otro electrodoméstico de nuestra casa, o incluso un monitoreo de ésta, ya sea en temperatura, sensores de movimiento, alarmas, cámaras, entre otros. Construir sistemas como estos me motiva mucho, además de ya estar trabajando en proyectos similares, pero con mayor complejidad.

Herramientas

El Sensor DHT22 se compone de 3 pines, un pin positivo el cual debe ser conectado a energía a 3.3V, para este caso se conecta en el Pin 1 que corresponde a alimentación de 3.3V, otro pin negativo que debe ir a tierra, el cual será conectado en el Pin 39 que corresponde a Ground, y finalmente el pin que entrega la señal o los datos, que debe ir conectado a una entrada GPIO, en este caso se elige el Pin 7, que corresponde al GPIO 4.

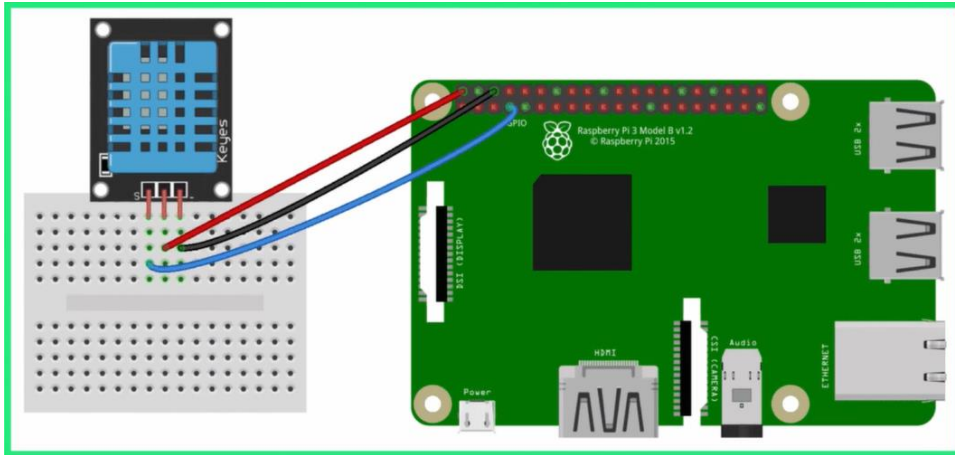
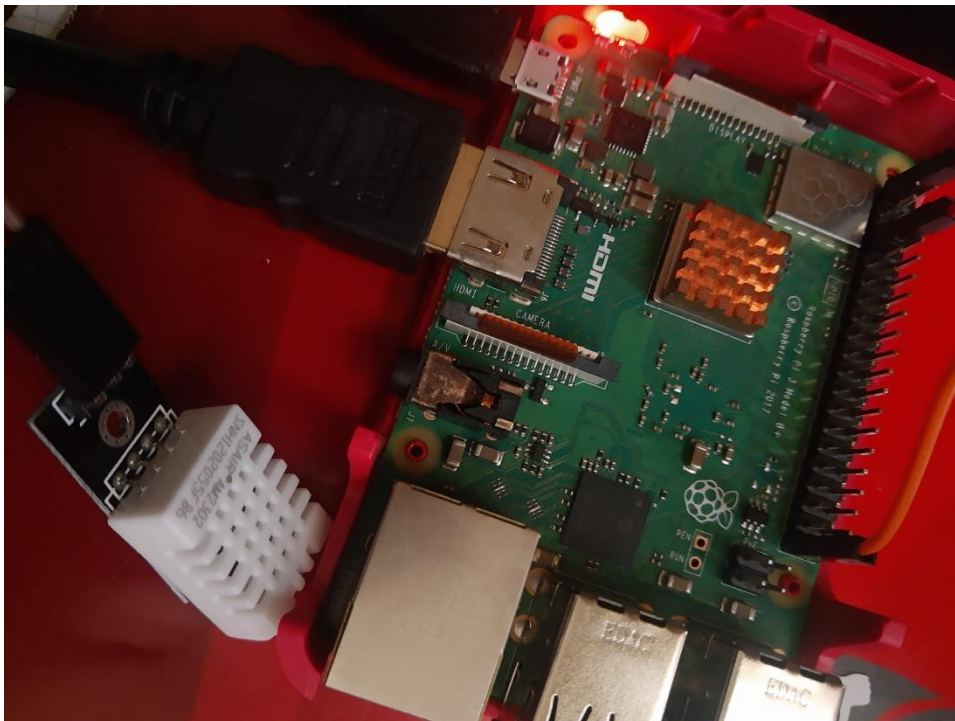
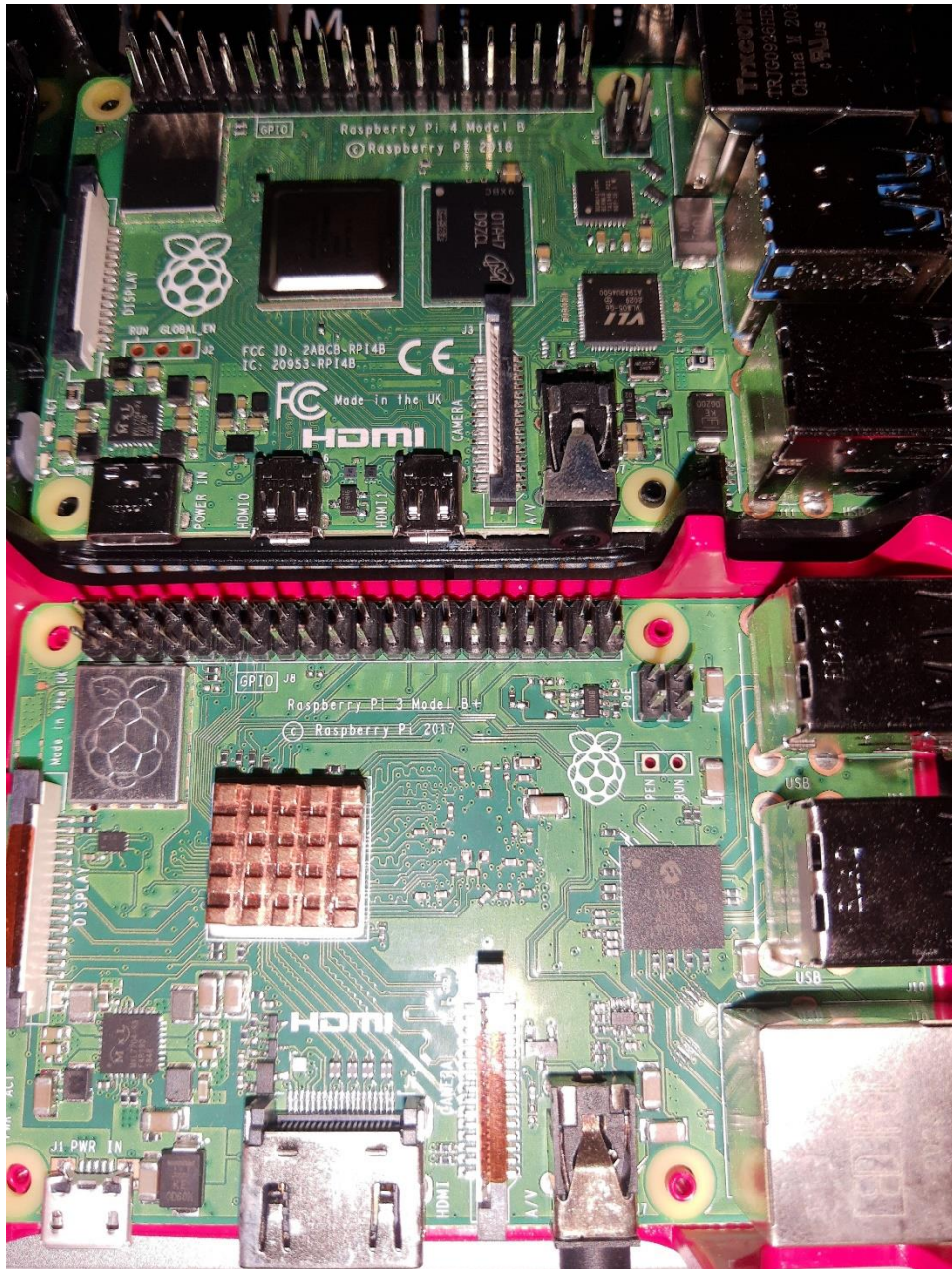


Imagen referencial de la conexión y el sensor, no es necesario el uso de protoboard.



Conexión real, mediante puertos GPIO de Raspberry Pi 3B+, cables Jumper y el sensor DHT22, integrado a una microplaca con una resistencia de 10k Ohms.



▪ Físicas:

- Raspberry Pi 3B+ (OS: Raspbian).
- Raspberry Pi 4B+ (OS: Ubuntu 22.04).
- Sensor DHT 22.
- Cables Jumper.

▪ No físicas:

- Python 3.7 y 3.10.
- Librería de Adafruit para uso de sensores DHT 11 y DHT 22.
- Thonny IDE y Visual Studio Code.

Código

Se utiliza la librería socket de Python, cuya librería viene incluida en el Kernel de Python3, la cual nos permite la comunicación de los Sockets de Berkeley y la debida interpretación para encomendar estas tareas a nuestra tarjeta de red de cada dispositivo. Se utiliza además la librería de Adafruit para el uso de sensores DHT, en este caso esta librería debe instalarse de forma externa. La instalación es mediante un comando en la terminal

- En caso de tener Python instalado:
 - **sudo pip3 install Adafruit_DHT**
- En caso de no tener Python instalado, se debe instalar mediante los siguientes comandos, y luego ejecutar el punto anterior para instalar la librería.
 - **sudo apt-get install python3-dev python3-pip**
 - **sudo python3 -m pip install --upgrade pip setuptools wheel**

Ambos códigos se encuentran debidamente comentados, pero se resume el funcionamiento de cada uno.

- ✓ Servidor.py: Es el encargado de percibir los datos del medio en tiempo real, mediante el sensor de Temperatura y Humedad, para luego enviarlos mediante un Socket TCP, el cual esperará una solicitud de conexión para luego enviarlos.
- ✓ Cliente.py: Es el encargado de conectarse al Servidor para obtener los datos percibidos por este, mediante un Socket TCP, y mostrar estos datos recibidos, además de elegir una cantidad finita de datos a recibir.

La ejecución de ambos programas debe ser mediante la terminal, situándose en la carpeta contenedora de los archivos.

- ✓ Servidor: **python3 Servidor.py** o **py Servidor.py**. (Dependiendo de la configuración de Python).
- ✓ Cliente: **python3 Cliente.py** o **py Cliente.py**. (Dependiendo de la configuración de Python).

Ejemplos:

```
py Cliente.py
```

```
python3 Servidor.py
```

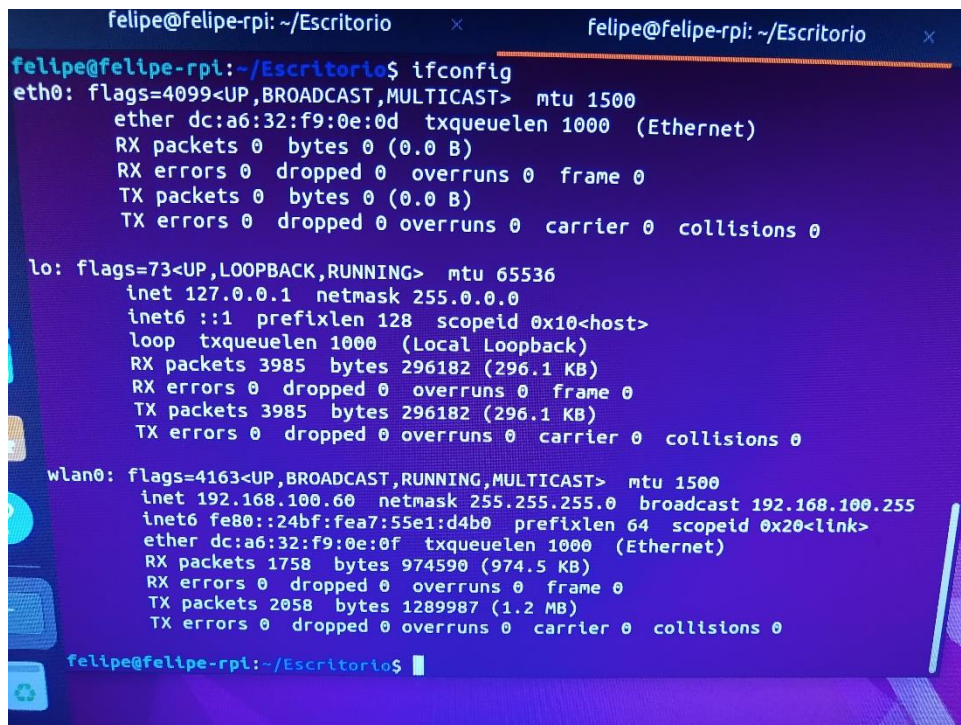
Elección de Socket

Se elige un socket TCP/IP por razones de carácter teórico y personal, comenzando en que TCP a diferencia de UDP, es un protocolo orientado a la conexión, que nos proporciona un mejor manejo de errores, y nos garantiza la entrega de los datos. A pesar de que UDP es no orientado a la conexión, parece ser más rápido que TCP, ya que este está continuamente enviando datos, sin verificar si llegaron a destino. Ahora la razón personal es que anteriormente había trabajado con Sockets TCP en un curso anterior (en lenguaje Python), a lo cual ya sabía cómo funcionaba la librería que permite el funcionamiento de estos.

Funcionamiento

Se establecen las direcciones IP de las maquinas Servidor y Cliente.

- Cliente: 192.168.100.60



```
felipe@felipe-rpi: ~/Escritorio
felipe@felipe-rpi:~/Escritorio$ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether dc:a6:32:f9:0e:0d txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

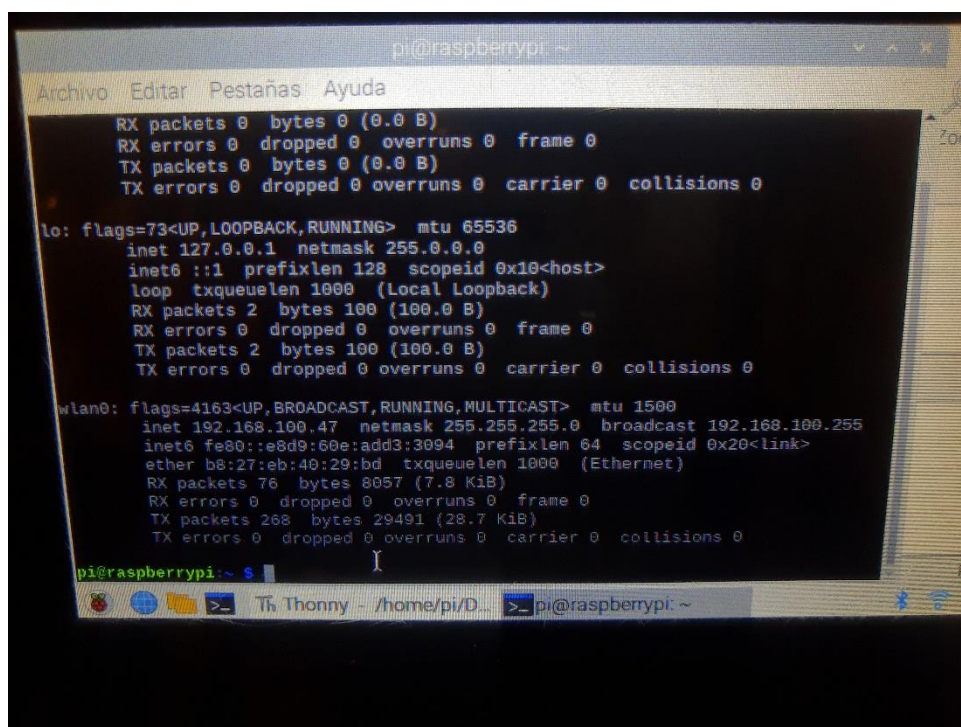
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 3985 bytes 296182 (296.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3985 bytes 296182 (296.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.60 netmask 255.255.255.0 broadcast 192.168.100.255
    inet6 fe80::24bf:fea7:55e1:d4b0 prefixlen 64 scopeid 0x20<link>
    ether dc:a6:32:f9:0e:0f txqueuelen 1000 (Ethernet)
    RX packets 1758 bytes 974590 (974.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2058 bytes 1289987 (1.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

felipe@felipe-rpi:~/Escritorio$
```

→

- Servidor: 192.168.100.47



```
pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 2 bytes 100 (100.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2 bytes 100 (100.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

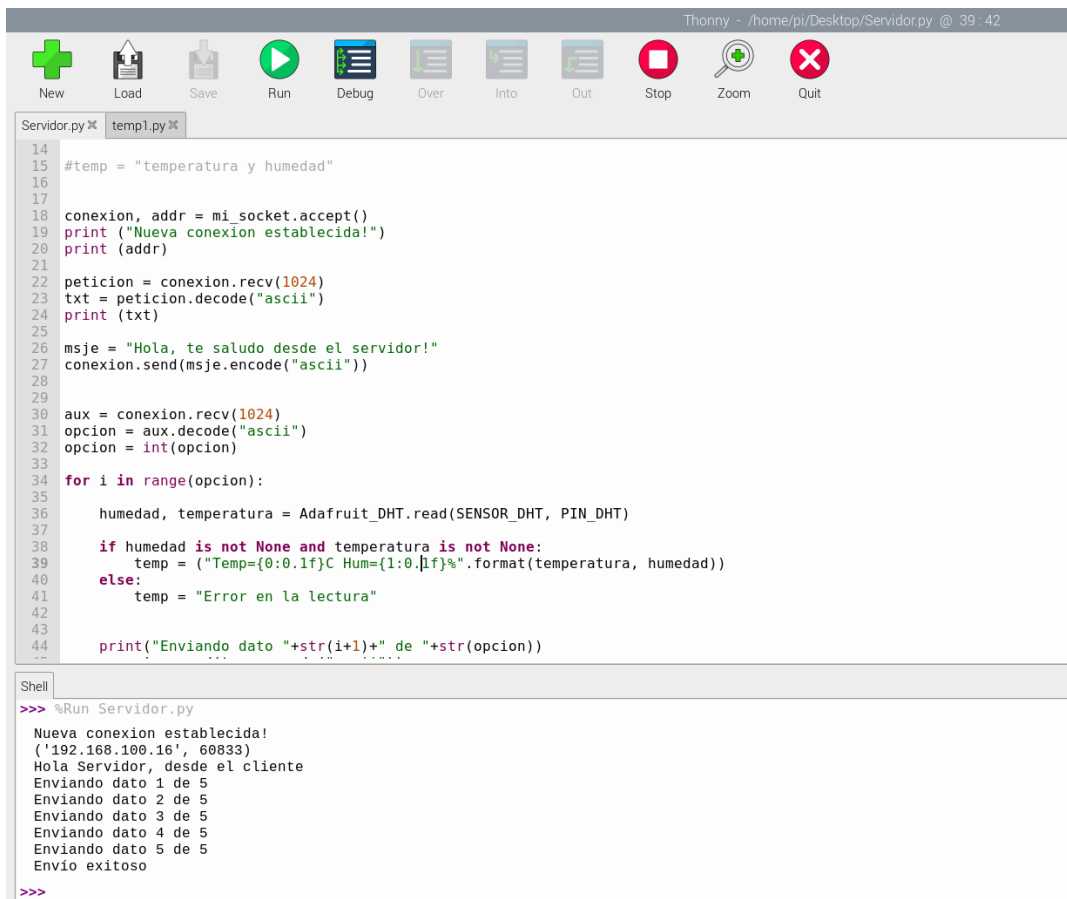
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.100.47 netmask 255.255.255.0 broadcast 192.168.100.255
    inet6 fe80::e8d9:60e:add3:3094 prefixlen 64 scopeid 0x20<link>
    ether b8:27:eb:40:29:bd txqueuelen 1000 (Ethernet)
    RX packets 76 bytes 8057 (7.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 268 bytes 29491 (28.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~$
```

→

El Cliente se conectará al puerto 37430 del Servidor, y escogerá un puerto disponible al azar para conectarse al puerto del Servidor.

→ Operación del Servidor:

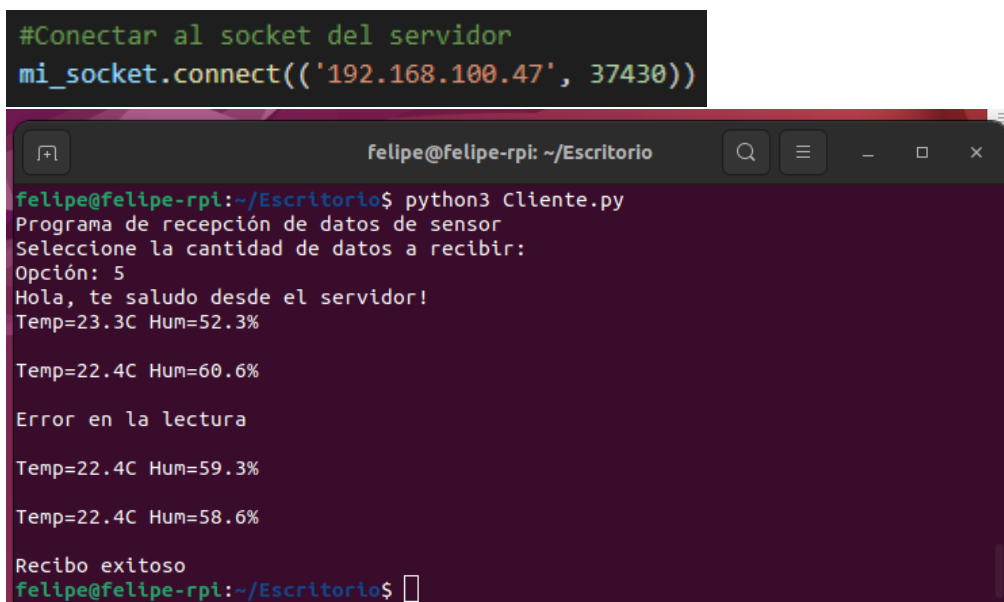


The screenshot shows the Thonny IDE interface. The top toolbar includes icons for New, Load, Save, Run, Debug, Over, Into, Out, Stop, Zoom, and Quit. The main editor window displays the code for 'Servidor.py'. The code uses the Adafruit DHT library to read temperature and humidity data from a sensor. It listens for a connection on port 37430, receives a request for a specific number of data points, and then sends the requested data back to the client. The output window shows the successful execution of the server, including the connection establishment, the received request, and the subsequent data transmissions.

```
14
15 #temp = "temperatura y humedad"
16
17
18 conexion, addr = mi_socket.accept()
19 print ("Nueva conexión establecida!")
20 print (addr)
21
22 petición = conexion.recv(1024)
23 txt = petición.decode("ascii")
24 print (txt)
25
26 msje = "Hola, te saludo desde el servidor!"
27 conexion.send(msje.encode("ascii"))
28
29
30 aux = conexion.recv(1024)
31 opcion = aux.decode("ascii")
32 opcion = int(opcion)
33
34 for i in range(opcion):
35
36     humedad, temperatura = Adafruit_DHT.read(SENSOR_DHT, PIN_DHT)
37
38     if humedad is not None and temperatura is not None:
39         temp = ("Temp={0:0.1f}C Hum={1:0.1f}%".format(temperatura, humedad))
40     else:
41         temp = "Error en la lectura"
42
43     print("Enviando dato "+str(i+1)+" de "+str(opcion))
44
45
```

```
>>> %Run Servidor.py
Nueva conexión establecida!
('192.168.100.16', 60833)
Hola Servidor, desde el cliente
Enviando dato 1 de 5
Enviando dato 2 de 5
Enviando dato 3 de 5
Enviando dato 4 de 5
Enviando dato 5 de 5
Envío exitoso
>>>
```

→ Operación del Cliente:



The screenshot shows a terminal window with the client code and its execution output. The code connects to the server at IP 192.168.100.47 on port 37430. The user is prompted to select the number of data points to receive, and the terminal shows the subsequent data transmissions from the server. The output includes the connection message, the received request, and the subsequent data transmissions.

```
#Conectar al socket del servidor
mi_socket.connect(('192.168.100.47', 37430))
```

```
felipe@felipe-rpi: ~/Escritorio
felipe@felipe-rpi:~/Escritorio$ python3 Cliente.py
Programa de recepción de datos de sensor
Seleccione la cantidad de datos a recibir:
Opción: 5
Hola, te saludo desde el servidor!
Temp=23.3C Hum=52.3%

Temp=22.4C Hum=60.6%

Error en la lectura

Temp=22.4C Hum=59.3%

Temp=22.4C Hum=58.6%

Recibo exitoso
felipe@felipe-rpi:~/Escritorio$
```

→ Operación en Wireshark:

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes Archivo, Edición, Visualización, Ir, Captura, Analizar, Estadísticas, Telefonía, Wireless, Herramientas, and Ayuda. The main window is divided into three panes: Packet List, Packet Details, and Packet Bytes.

Packet List: Shows a list of captured packets. The selected packet (No. 14) is a TCP segment from 192.168.100.60 to 192.168.100.47, Seq=35, Ack=33, Len=20.

Packet Details: Shows the structure of the selected packet. It includes Ethernet II (Type: IPv4), Internet Protocol Version 4 (Source: 192.168.100.60, Destination: 192.168.100.47), and Transmission Control Protocol (Source Port: 57354, Destination Port: 57354, Seq: 35, Ack: 33, Len: 20).

Packet Bytes: Shows the raw data of the selected packet in hexadecimal and ASCII.

Terminal Window: An inset terminal window shows the execution of a Python script named `python3 Cliente.py`. The script prompts the user to select the number of data points to receive (5) and displays sensor data (Temperature and Humidity) received from a Raspberry Pi.

Es posible notar como se establece la conexión entre las líneas 4 y 5 de Wireshark, y como se establece por ejemplo el acuerdo de tres vías, de la línea 7 hasta la 23 se puede ver el intercambio de datos útiles, en este caso se selecciona una línea, la cual corresponde al primer dato del sensor desde el Servidor (192.168.100.47) hacia el Cliente (192.168.100.60).

Alcance

Como se menciona en un ítem anterior, este proyecto enfocado a gran escala puede llegar a ser, por ejemplo, una especie de estación meteorológica, o una plataforma de control de confort térmico, la cual para una empresa pueda significar ganar o perder dinero al tener o no estos niveles estabilizados, por lo tanto, su desarrollo y su aplicación puede motivar a grandes ideas. Para este caso, es posible encontrar alguna relación entre la temperatura y la humedad relativa, y es que a mayor temperatura pues el aire se vuelve más seco y la humedad relativa disminuye. Ahora a menor temperatura pues el aire se vuelve más húmedo, aumentando la humedad relativa. Finalmente, como conclusión tenemos que estas variables son inversamente proporcionales. Ahora este análisis es posible aplicarlo a un proyecto más complejo, realizar ciencia de datos para encontrar relaciones o estadísticas entre variables.

Resumen experiencia

El uso de sockets resultó ser una experiencia muy enriquecedora, debido a que anteriormente en el curso anterior donde usé sockets pues no entendía sobre qué estaba trabajando, que era cada función de la librería y como esta trabaja en el fondo. El entender los conceptos de forma teórica en las cátedras me hizo entender la aplicación en la práctica. Dentro de las dificultades en el proceso fue el funcionamiento de un sensor, el cual inicialmente era un sensor DHT11 pero que resultó estar malo, entregaba siempre la misma temperatura (20° Celsius) y valores poco reales en la humedad relativa, esto provocó que no pudiera avanzar hasta tener nuevamente un sensor en correcto funcionamiento, imposibilitándome de mostrar avances de interacción entre el sensor y el servidor, sin embargo, durante ese tiempo se avanzó en la interacción entre el servidor y el cliente, el correcto funcionamiento del socket, con el envío de mensajes y de también de archivos. Finalmente, se logró el objetivo de interactuar entre el sensor que entrega datos al servidor de acuerdo con lo que evalúa en el medio, y como luego estos datos son enviados por un socket TCP a un cliente que tendrá acceso a la evaluación del medio que realiza el sensor en el otro dispositivo.