In [ ]:
```
import time
from pynq.overlays.base import BaseOverlay
import socket

base = BaseOverlay("base.bit")
btns = base.btns_gpio
leds = base.leds
```

Sockets

This notebook has both a client and a server functionality. One PYNQ board in the group will be the client and SENDS the message. Another PYNQ board will be the server and RECEIVES the message.

Server

Here, we'll build the server code to LISTEN for a message from a specific PYNQ board.

When we send/receive messages, we need to pieces of information which will tell us where to send the information. First, we need the IP address of our friend. Second, we need to chose a port to listen on. For an analogy, Alice expects her friend, Bob, to deliver a package to our back door. With this information, ALICE (server ip address) can wait at the BACK DOOR (port) for BOB (client ip address) to deliver the package.

Format of the information ipv4 address: ###.###.###.### (no need for leading zeros if the number is less than three digits) port: ##### (it could be 4 or 5 digits long, but must be >1024)

Use the socket documentation (Section 18.1.3) to find the appropriate functions
https://python.readthedocs.io/en/latest/library/socket.html

In [ ]:
```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 1: Bind the socket to the pynq board <CLIENT-IP> at port <LISTENING-PORT>
server_ip = '192.168.2.1'  # Replace with the actual client IP
listening_port = 8080  # Replace with the actual port number
sock.bind((server_ip, listening_port))

# 2: Accept connections
sock.listen(1)
print("The server is ready to receive")

while True:
    connection_socket, client_address = sock.accept()
    try:
        # 3: Receive bytes from the connection
        message = connection_socket.recv(2048)

        # 4: Print the received message
        print("Received message:", message.decode())
```

```
    finally:
        connection_socket.close()
```

Client

Now, we can implement the CLIENT code.

Back to the analogy, now we're interested in delivering a package to our friend's back door. This means BOB (client ip address) is delivering a package to ALICE (server ip address) at her BACK DOOR (port)

Remember to start the server before running the client code

In [ ]:
```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 1: Connect the socket (sock) to the <SERVER-IP> and chosen port <LISTENING-PORT>
server_ip = '192.168.2.1'  # Replace with the actual server IP
listening_port =8080  # Replace with the actual port number
sock.connect((server_ip, listening_port))

# 2: Send the message "Hello world!\n"
message = "Hello world!\n"
sock.sendall(message.encode())

# 3: Close the socket
sock.close()
```

On your server, you should see the message and then the server will shutdown! When we close a socket, both the client and the server are disconnected from the port.

Instead, change the function above to send 5 messages before closing.

The pseudocode looks like this

- connect the socket
- for i in range(5)
  - msg = input("Message to send: ")
  - send the message (msg)
- close the socket

In [4]:
```
import socket
```

In [ ]:
```
import socket

server_ip = '192.168.2.1'  # Replace with the actual server IP address
server_port = 12345

try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # Create a new socket
    sock.connect((server_ip, server_port))  # Connect the socket
    print("Connected to server")

    # Send the message
    message = "Hello world\n"
    sock.sendall(message.encode())  # Encode and send the message
    print("Message sent")

finally:
    # Close the socket to ensure it's properly closed
    sock.close()
    print("Socket closed")
```

In [ ]:
```

```