

WES 237A: Introduction to Embedded System Design (Winter 2026)

Lab 4: Network Communication

Due: 2/16/2026 11:59pm

To report and reflect on your WES 237A labs, please complete this Post-Lab report by the deadline by submitting the following 2 parts:

- Upload your Lab 4 report, composed by a single PDF that includes your in-lab answers to the bolded questions in the Google Doc Lab and your Jupyter Notebook code. You could either scan your written copy or simply type your answer in this Google Doc. **However, please make sure your responses are readable.**
- Answer two short essay-like questions on your Lab experience.

All responses should be submitted to Canvas. Please also be sure to push your code to your git repo as well.

Locating IP Addresses of Devices in your Network

1. Connect the PYNQ board to the network switch over Ethernet.
2. Run '\$ ifconfig'. This is the *Interface Configuration* command and will tell you the different interfaces on your PYNQ board.
 - a. **How many IPv4 addresses are assigned to the board? What is the IPv4 address assigned to the 'eth0' or Ethernet interface? What is the netmask of this address?**
 - i. Note: `eth0: 1` or `usb0` is a virtual interface through the USB cable. This assigns your board an IP address over USB. This is a static IP address, so you can always reach your board from this IP address over USB.

There is only the primary address 192.168.0.244 as the IP address, eth0. The netmask is 255.255.255.0

3. Connect your computer to the WiFi (connect to the router)
4. On your computer, open a command prompt and run '\$ ipconfig' on Windows and '\$ ifconfig' on MAC/Linux (it may take a second to connect, so wait a minute and then run the command)
 - a. **How many IPv4 addresses are assigned to this machine? What IPv4 address has the same netmask as the PYNQ board?**

There are 2 IPv4 addresses, the loopback (127.0.0.1) and WiFi (192.168.0.116). 192.168.0.116 has the same netmask as the PYNQ.

5. Right now, your local machine and your PYNQ board form a network! However, we're more interested in networking two PYNQ boards together rather than your local machine and your PYNQ board. Luckily, every device hooked up to the switch is assigned an IP address on this network. That means we can communicate with any other board in the class. **Below, compile all the IP addresses of the PYNQ boards in your group.**

I am working alone, I have my own router:

My PYNQ Board: 192.168.0.244

My Local Machine (Mac): 192.168.0.116

6. To access your PYNQ board Jupyter notebooks, go to <PYNQ-IP>:9090

PYNQ-PYNQ Communication with Python

- Here, we're going to implement basic message-sending functionality in Python between two PYNQ boards.
- Download '[sockets_example.ipynb](#)'
- Go through and complete the code. Answer the following questions.
 - **What does `socket.SOCK_STREAM` mean (hint: search the documentation link in the notebook)?**

socket.SOCK_STREAM specified the socket to use TCP. To ensure reliable delivery, it will use a formal handshake between the client and server before sending data.

- **What is the order of operations for starting a client socket and sending a message?**

- socket() creates a new socket, an abstract connection object
- connect((IP,PORT)) will reach out to the server's IP address and port to establish a connection.
- send() / sendall() will send data
- close() will terminate the connection once the message is sent

- **What is the order of operations for starting a server socket and receiving a message?**

- socket() creates a new socket, an abstract connection object
- bind((IP, PORT)) associates the socket with the specific network interface and port number
- listen() enables the server to accept connections
- accept() This blocking call will make the server wait here until a client tries to connect. It returns a socket for that connection.
- recv() reads the incoming data from the client.
- close() will terminate the connection once the message is sent

Wireshark

1. On your local machine (or lab machine), install [Wireshark](#)

If you are on Windows:

- a. Open the Firewall and Network Protection
- b. Click 'Allow an app through firewall.'
- c. Click 'Change Settings'
- d. Scroll down to 'Python'
- e. Select all 'Python' applications and all 'Public' boxes for each 'Python'

<input checked="" type="checkbox"/> Python	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No
<input checked="" type="checkbox"/> Python	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	No

- f. Open the program 'IDLE (Python 3.7 64-bit)'

If you are on Linux / MacOS:

- g. Run the following command:

```
sudo ufw status
```

- h. If you get Active status, run this:

```
Sudo ufw allow 12345/tcp
```

- i. Run idle3 from the terminal (Linux) or search for IDLE in applications (Mac)

2. Click File->New File and paste the following code (**Check for tab v space errors when copying and pasting**)

```
import socket
import time
import signal
import sys

def run_program():
    sock_1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock_1.bind(('0.0.0.0', 12345))
    sock_1.listen()
    print('Waiting for connection')
    conn, addr = sock_1.accept()
    print('Connected')
    with conn:
        data = conn.recv(1024)
    print(data.decode())

if __name__ == '__main__':
    original_sigint = signal.getsignal(signal.SIGINT)
    signal.signal(signal.SIGINT, exit)
    run_program()
```

3. Save the file, then select 'Run -> Run Module'. This is a slight variation of your server. It is waiting on port 12345 on the local lab machine.
4. From your PYNQ board, connect your client to
 - IP: local lab IP
 - Port: 12345

5. Send the message "Hello world\n"
6. You should see it displayed in the Python terminal
7. Now open Wireshark
8. Double click 'Wi-Fi' or 'Ethernet', depending on how you connected to the network.
You're now capturing a trace of the network traffic between your machine and the PYNQ board, via the router. Look at a few of the traces. Notice which are between your PYNQ board and the local machine (check the IP addresses) and which involve the router.
There will only be a difference if you also connect the PYNQ board directly to your local machine.
9. Where it says 'Apply a display filter' at the top, type 'tcp'
10. Repeat steps 9-11
11. Check the packet trace for any changes
 - a. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the PYNQ board and the local machine? What is it in the segment that identifies the segment as a SYN segment?**

Based on my traces, the relative sequence number was 0. The raw sequence number is a large random 32-bit integer. The segment is identified as a SYN segment because the SYN bit in the flags field of the TCP header is set to 1. My trace specifically shows [SYN, ECE< CWR] where SYN triggers the initial connection handshake.

- b. Right-click this trace and select 'Follow->TCP Stream'
 - c. **Repeat a few times with different messages. Describe what's happening in the 5-10 steps of the TCP sequence for this communication. You can refresh your TCP flags [here](#).**
- **SYN:** The Mac sends a synchronization request (Seq=0) to the PYNQ to start the connection.
 - **SYN, ACK:** The PYNQ acknowledges (Ack=1) and sends its own sync request to the Mac.
 - **ACK:** The Mac confirms, completing the 3-way handshake. The connection is now established.
 - **PSH, ACK:** The Mac "pushes" the actual "Hello world" data (13 bytes) to the PYNQ.
 - **ACK:** The PYNQ confirms it received the data, incrementing the acknowledgment number.
 - **FIN, ACK:** The Mac sends a FIN flag to signal it is finished and wants to close the link.
 - **ACK:** The PYNQ acknowledges the Mac's request to close.
 - **FIN, ACK:** The PYNQ sends its own FIN flag to close the server-side connection.
 - **ACK:** The Mac sends the final acknowledgment. The connection is officially closed.