Activity Guide: Setting Up a Raspberry Pi Peer-to-Peer Network

## **Objective:**

Learn to set up a peer-to-peer network using Raspberry Pi devices for a collaborative IoT project.

## **Materials Needed:**

* Raspberry Pi boards (at least two)
* MicroSD cards for each Raspberry Pi
* Power supplies for Raspberry Pi
* Ethernet cables
* USB Wi-Fi adapters (optional)
* Laptop or computer for initial configuration
* Internet connection

## **Activity Steps:**

### **Step 1: Prepare Raspberry Pi Devices**

1. **Download Raspbian OS:**
   * Download the Raspbian operating system from the official Raspberry Pi website (<https://www.raspberrypi.org/downloads/>).
2. **Install Raspbian on MicroSD Cards:**
   * Use a tool like Etcher to write the Raspbian image to the microSD cards.

### **Step 2: Initial Configuration**

1. **Boot Up Raspberry Pi Devices:**
   * Insert the microSD cards into the Raspberry Pi boards and connect the power supplies.
   * Connect the Raspberry Pi devices to a display, keyboard, and mouse for initial configuration.
2. **Configure Network Settings:**
   * Access the Raspberry Pi configuration menu using sudo raspi-config.
   * Set a hostname for each Raspberry Pi.
   * Configure both wired and wireless network settings if using Wi-Fi adapters.

### **Step 3: Establish Wired Connection (Optional)**

1. **Connect Raspberry Pi Devices Using Ethernet:**
   * Connect the Raspberry Pi devices using Ethernet cables to establish a wired connection.

### **Step 4: Establish Wireless Connection (Optional)**

1. **Configure Wi-Fi (Optional):**
   * If using Wi-Fi adapters, configure the devices to connect wirelessly.
   * Set up a secure Wi-Fi connection with WPA2 encryption.

### **Step 5: Assign Static IP Addresses**

#### **1. Open the dhcpcd.conf file for editing:**

sudo nano /etc/dhcpcd.conf

#### **2. Navigate to the end of the file and add the following lines:**

interface eth0

static ip\_address=192.168.1.2/24

static routers=192.168.1.1

static domain\_name\_servers=192.168.1.1

interface wlan0

static ip\_address=192.168.1.3/24

static routers=192.168.1.1

static domain\_name\_servers=192.168.1.1

* Replace eth0 and wlan0 with the appropriate interface names for your wired and wireless connections. You can check the available interfaces using the command ifconfig.
* Modify the static ip\_address to the desired static IP address for each Raspberry Pi. In this example, the first Raspberry Pi has IP address 192.168.1.2 and the second one has 192.168.1.3.
* Adjust the static routers and static domain\_name\_servers values to match your network configuration. In this example, they are set to the IP address of the router (192.168.1.1).

#### **3. Save and Exit:**

* In nano, you can save the changes by pressing Ctrl + X, then Y to confirm, and finally press Enter to exit.

#### **4. Restart the Raspberry Pi:**

sudo reboot

#### **5. Verify the Changes:**

After the Raspberry Pi restarts, you can use the ifconfig command to verify that the assigned static IP addresses are in effect:

ifconfig

Look for the specified IP addresses under the relevant network interfaces (eth0 and wlan0 in this example).

### **Step 6: Test Network Connectivity**

#### **1. Open a Terminal on Raspberry Pi 1:**

ping 192.168.1.3

Replace 192.168.1.3 with the static IP address of Raspberry Pi 2. This command sends ICMP echo requests to the second Raspberry Pi.

#### **2. Open a Terminal on Raspberry Pi 2:**

ping 192.168.1.2

Replace 192.168.1.2 with the static IP address of Raspberry Pi 1. This command sends ICMP echo requests to the first Raspberry Pi.

#### **3. Observe the Results:**

* If you see a series of successful responses (ping replies), it indicates that the devices can communicate over the network.
* If you encounter timeouts or errors, there may be an issue with the network configuration. Double-check the IP addresses, subnet masks, and connectivity.

#### **4. Test Wi-Fi Connectivity (If Using Wi-Fi):**

If you configured a Wi-Fi connection, repeat the above steps using the wireless IP addresses.

ping 192.168.1.3

ping 192.168.1.2

#### **5. Verify Bidirectional Communication:**

On both Raspberry Pis, you should see output similar to the following:

64 bytes from 192.168.1.X: icmp\_seq=1 ttl=64 time=XX.X ms

64 bytes from 192.168.1.X: icmp\_seq=2 ttl=64 time=XX.X ms

64 bytes from 192.168.1.X: icmp\_seq=3 ttl=64 time=XX.X ms

This indicates successful bidirectional communication between the Raspberry Pi devices.

### **Step 7: Basic Security Measures**

#### **1. Install ufw (Uncomplicated Firewall):**

sudo apt-get update

sudo apt-get install ufw

#### **2. Configure Firewall:**

sudo ufw enable

sudo ufw allow 22 # Allow SSH

This sets up a basic firewall allowing incoming SSH connections on port 22. Ensure that SSH is the only service exposed unless you have specific requirements.

#### **3. Secure SSH Configuration:**

Open the SSH configuration file:

sudo nano /etc/ssh/sshd\_config

Find and modify the following lines:

PermitRootLogin no

PasswordAuthentication no

This disables root login and password authentication, requiring users to authenticate using SSH keys.

#### **4. Restart SSH Service:**

sudo service ssh restart

#### **5. Test SSH Connection:**

From another device, attempt to SSH into the Raspberry Pi:

ssh pi@192.168.1.2

Replace 192.168.1.2 with the IP address of the Raspberry Pi. Ensure that you can connect using an SSH key.

#### **6. Test Bidirectional SSH:**

#### From Raspberry Pi 2, attempt to SSH into Raspberry Pi 1:

#### ssh pi@192.168.1.2

#### Replace 192.168.1.2 with the IP address of Raspberry Pi 1. Confirm that you can establish an SSH connection in both directions.

In both cases, if you see the command prompt changing to the remote Raspberry Pi's prompt, it indicates a successful SSH connection. You are now logged into the other Raspberry Pi via SSH.

Once connected, you can execute commands on the remote Raspberry Pi. For example:

hostname # Displays the hostname of the remote Raspberry Pi

When you're done, type exit and press Enter to exit the SSH session and return to the local terminal.

Repeat steps 1 to 5 in the opposite direction (from Raspberry Pi 2 to Raspberry Pi 1) to confirm bidirectional SSH.

#### **7. Verify Firewall Configuration:**

Check the firewall rules:

sudo ufw status

You should see rules allowing incoming SSH traffic and possibly other rules depending on your project requirements.

#### **8. Additional Considerations:**

* **Change Default Passwords:** Ensure that default passwords for user accounts are changed.
* **Regular Updates:** Keep the system updated with security patches.
* **Monitor Logs:** Regularly check system logs for suspicious activities.

#### **9. Document Security Measures:**

Update your project documentation with the implemented security measures, including firewall configurations and SSH settings.

### **Step 8: Documentation**

**1. Document the Setup:**

* + Create a documentation file detailing the hardware setup, network configuration, and security measures.
  + Include IP addresses, login credentials, and any other relevant information.

### **Step 9: Python Script for Remote Access**

#### **Step 1: Install paramiko:**

pip install paramiko

#### **Step 2: Create a Python Script (remote\_access.py):**

Create a new Python script, for example, remote\_access.py, using a text editor or an integrated development environment (IDE).

#### **Step 3: Import Required Libraries:**

import paramiko

import getpass

#### **Step 4: Define Remote Host Details:**

remote\_host = "192.168.1.3" # Replace with the IP address of the target Raspberry Pi

remote\_port = 22 # Default SSH port

remote\_user = "minion" # Username on the target Raspberry Pi

#### **Step 5: Get SSH Password (Optional):**

remote\_password = getpass.getpass("Enter SSH password: ")

If you prefer to use password-based authentication, you can use getpass to securely input the password.

#### **Step 6: Set Up SSH Connection:**

ssh = paramiko.SSHClient()

ssh.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

#### **Step 7: Connect to the Remote Host:**

try:

ssh.connect(remote\_host, port=remote\_port, username=remote\_user, password=remote\_password)

print("Connected to", remote\_host)

except Exception as e:

print("Error:", str(e))

exit()

This block establishes an SSH connection to the remote Raspberry Pi using the provided details.

#### **Step 8: Execute Commands Remotely:**

command\_to\_execute = "hostname" # Replace with the desired command

stdin, stdout, stderr = ssh.exec\_command(command\_to\_execute)

output = stdout.read().decode("utf-8")

error = stderr.read().decode("utf-8")

print("Command Output:")

print(output)

if error:

print("Error:")

print(error)

Replace "hostname" with the command you want to execute remotely.

#### **Step 9: Close the SSH Connection:**

ssh.close()

#### **Step 10: Run the Script:**

Save the script and run it:

python remote\_access.py

You may need to enter the SSH password if you didn't use SSH keys.

#### **Step 11: Verify Results:**

Check the script's output, which should display the output of the executed command on the remote Raspberry Pi.

#### **Step 12: Experiment with Additional Commands:**

Modify the command\_to\_execute variable to experiment with executing different commands remotely.

**Additional Command to Test:**

The Python script for remote access can run any command that the target Raspberry Pi supports. Here are a few examples of commands you might run remotely:

1. **Check System Information:**

command\_to\_execute = "uname -a"

**List Files in a Directory:**

command\_to\_execute = "ls /path/to/directory"

**Check Disk Space:**

command\_to\_execute = "df -h"

**View Network Interfaces:**

command\_to\_execute = "ifconfig"

**Check Connected Devices:**

command\_to\_execute = "lsusb"

**Show Running Processes:**

command\_to\_execute = "ps aux"

**Get Raspberry Pi Model:**

command\_to\_execute = "cat /proc/device-tree/model"

**View IP Configuration:**

command\_to\_execute = "ip a"