KME761G11 Lihini Hewage, Shamila Thennakoon, Upeksha Liyanage

Final Assignment Report

Hardware 1: Networks

School of ICT
Metropolia University of Applied Sciences
21.3.2024 (v1.0)

Version history

Ver	Description	Date	Author(s)
1.0	Created structure for the final assignment report.	14.1.2024	Saana Vallius
2.0	Introduction	18.2.2024	Lihini Hewage
3.0	Setting up the Network	19.2.2024	Upeksha Liyanage
4.0	Setting up the Raspberry Pi	21.2.2024	Upeksha Liyanage
5.0	Raspberry Pi Configuration	25.2.2024	Lihini Hewage
6.0	Setting up MQTT	10.3.2024	Shamila Thennakoon
7.0	Setting up the Raspberry Pi Pico	13.3.2024	Lihini / Shamila/ Upeksha
8.0	Some additional services to network project	18.3.2024	Lihini / Shamila/ Upeksha
9.0	References	18.3.2024	Shamila Thennakoon

Content

1 Introduction

This report includes the documentation for setting up a wireless network between a Raspberry Pi Pico W microcontroller and a Raspberry Pi computer, with the aim of facilitating MQTT message transmission. This document provides guidelines for the final assignment, outlining the necessary steps and configurations to achieve seamless communication between the devices.

- The goal for the assignment

The primary objective is to establish a wireless connection infrastructure between the Raspberry Pi Pico W microcontroller and the Raspberry Pi computer. Specifically, the focus is on enabling MQTT message transmission wirelessly from the Raspberry Pi Pico W to the Raspberry Pi. By accomplishing this, we establish the foundation for further hardware integration and communication protocols in subsequent project phases.

- The motivation for completing the assignment.

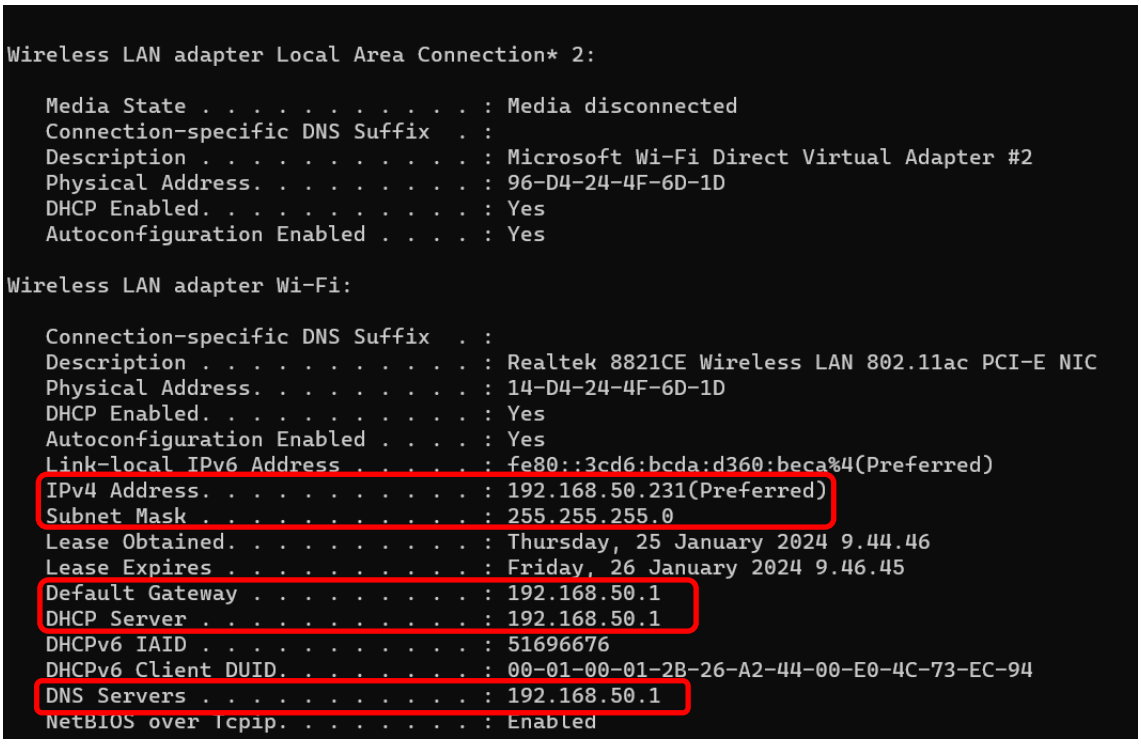
The motivation behind this is to enable efficient communication between hardware components through wireless connectivity. This not only offers flexibility in device placement but also enhances system functionality and usability. By integrating MQTT, we ensure reliable data exchange between the Raspberry Pi Pico W and the Raspberry Pi, enabling seamless interaction and data flow within the project environment.

By adhering to the instructions outlined in this assignment, we have effectively established a wireless network connecting the Raspberry Pi Pico W and the Raspberry Pi. Through this process, we have acquired invaluable knowledge and experience in network setup and integration, clearing the way for future advancements in our project journey.

2 Setting up the Network

Mainly this chapter is discussing the setup of the wireless router for individual groups (our group -11).

To access the management interface of the router, connected our computer to the default WLAN of the router and navigate to http://www.asusrouter.com. Logged into the web management interface with the default admin account. Then we checked the default gateway server, DNS server, IPV4 address and subnet mask details by using 'ipconfig' command on command prompt and it showed as follows.



- ✓ To enhance the security of ASUS router's network, followed these steps to replace the default admin password:

Access the Web Management Interface:

Log in to the Router:

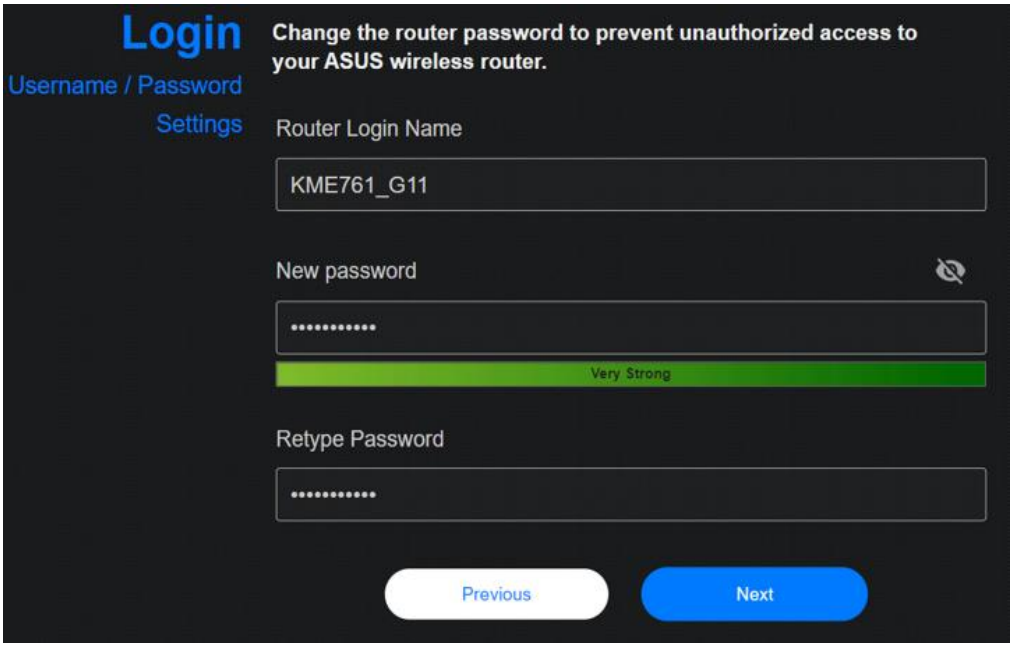
Firstly, we entered the default username and password. The default username is usually admin, and the default password is usually admin or blank. Then clicked "Login" to proceed.

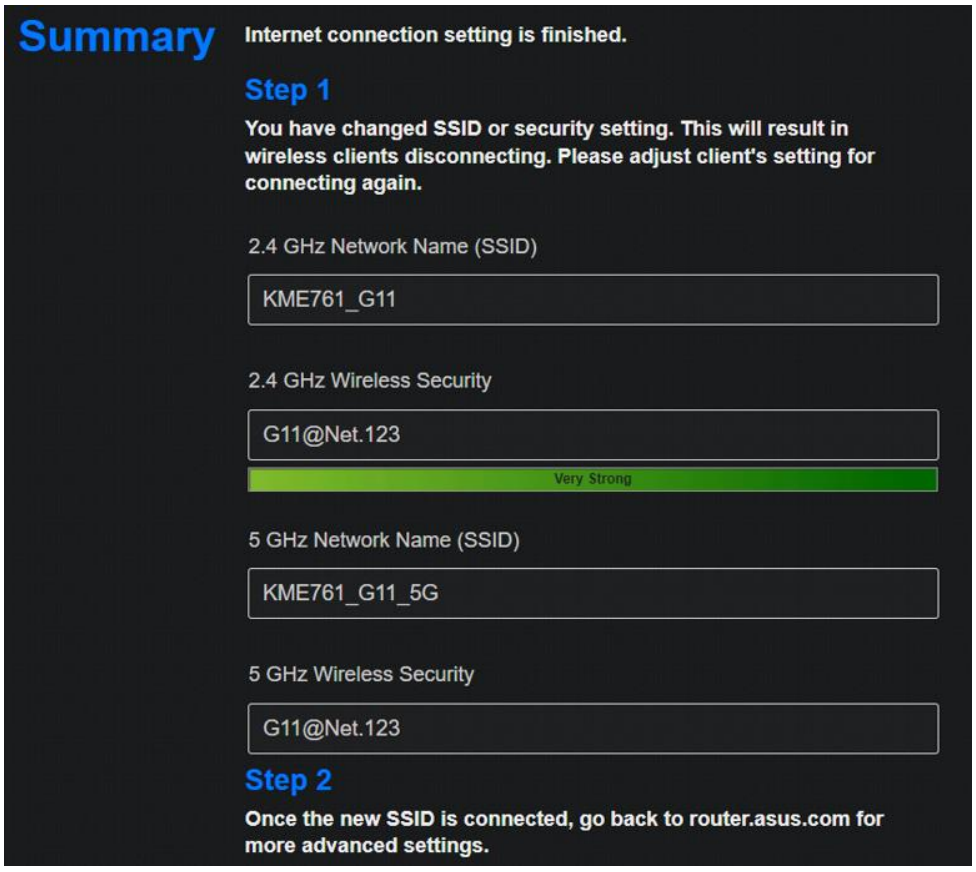
Navigate to the Administration Section & Change the Admin Password:

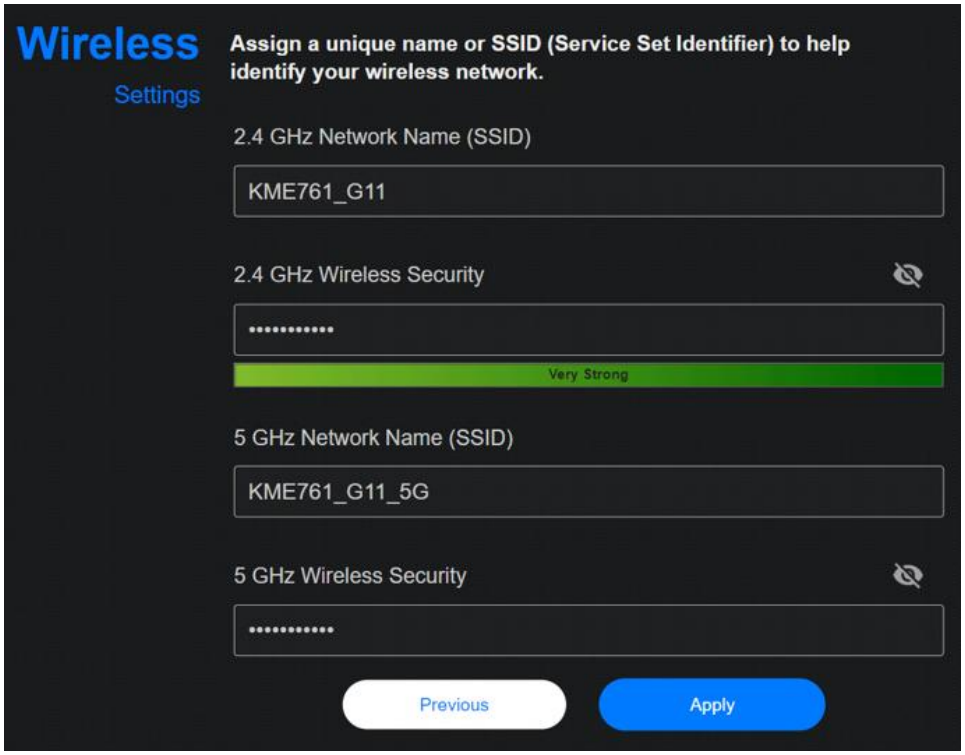
Once logged in, changed settings related to the router's configuration. We changed the Admin Password of the router. Entered a new, strong password in the designated field. (Used a combination of uppercase and lowercase letters, numbers, and special characters to create a secure password.) Re-

entered the new password to confirm it and saved changes. After entering the new password, look for a "Save" or "Apply" button on the page.

Related screenshots are as follows.



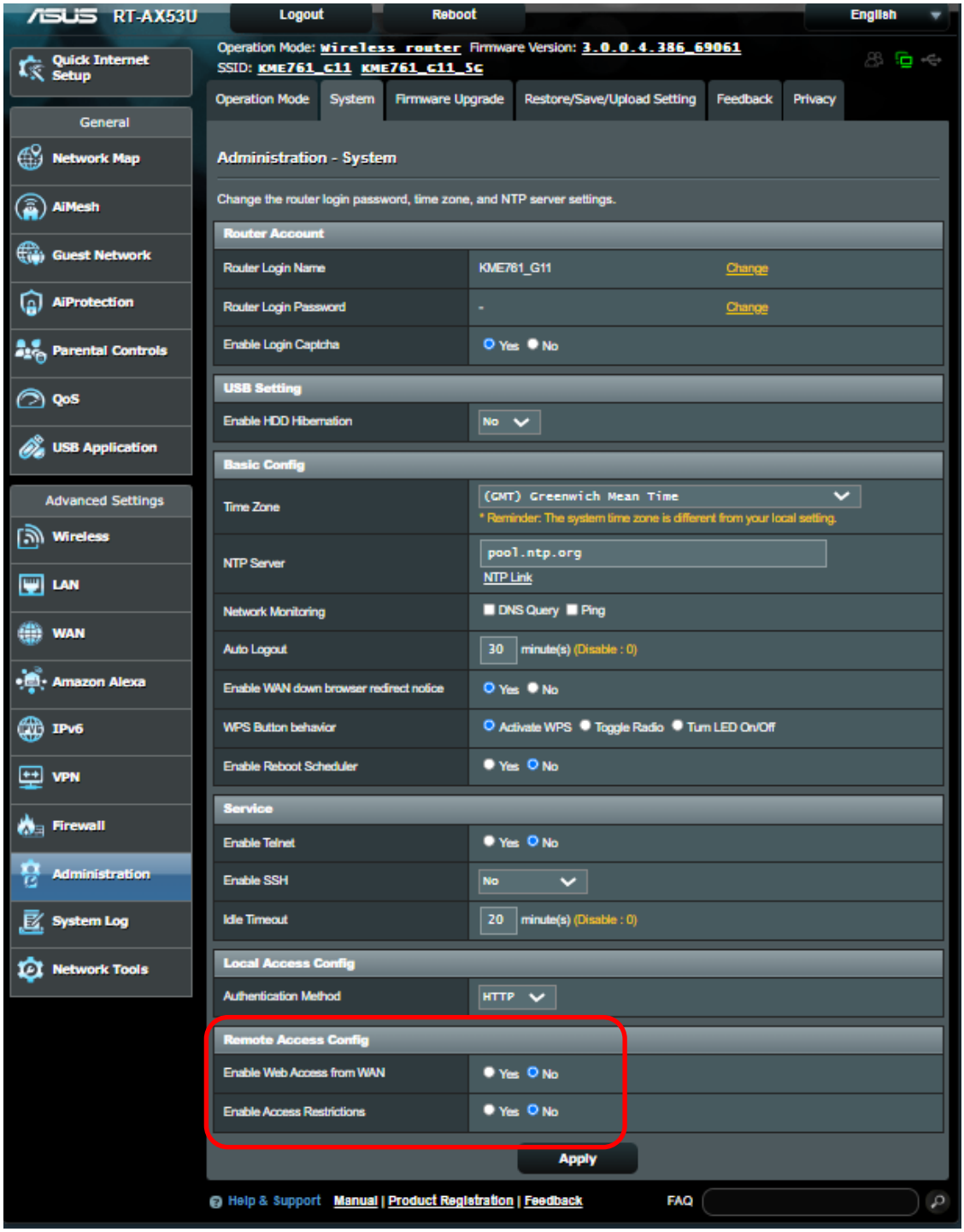

Summary
Internet connection setting is finished.
Step 1
You have changed SSID or security setting. This will result in wireless clients disconnecting. Please adjust client's setting for connecting again.
2.4 GHz Network Name (SSID)
KME761_G11
2.4 GHz Wireless Security
G11@Net.123
Very Strong
5 GHz Network Name (SSID)
KME761_G11_5G
5 GHz Wireless Security
G11@Net.123
Step 2
Once the new SSID is connected, go back to router.asus.com for more advanced settings.


Wireless
Settings
Assign a unique name or SSID (Service Set Identifier) to help identify your wireless network.
2.4 GHz Network Name (SSID)
KME761_G11
2.4 GHz Wireless Security
Very Strong
5 GHz Network Name (SSID)
KME761_G11_5G
5 GHz Wireless Security
Previous
Apply

- ✓ Disable router management from remote networks.

Disabling router management from remote networks means configuring our router, that it can only be accessed and managed from devices that are connected to the local network, rather than allowing management access from external or remote networks, such as the internet. This helps enhance the security of the router by reducing the attack surface and limiting access to its configuration interface to trusted devices within our home or school network.

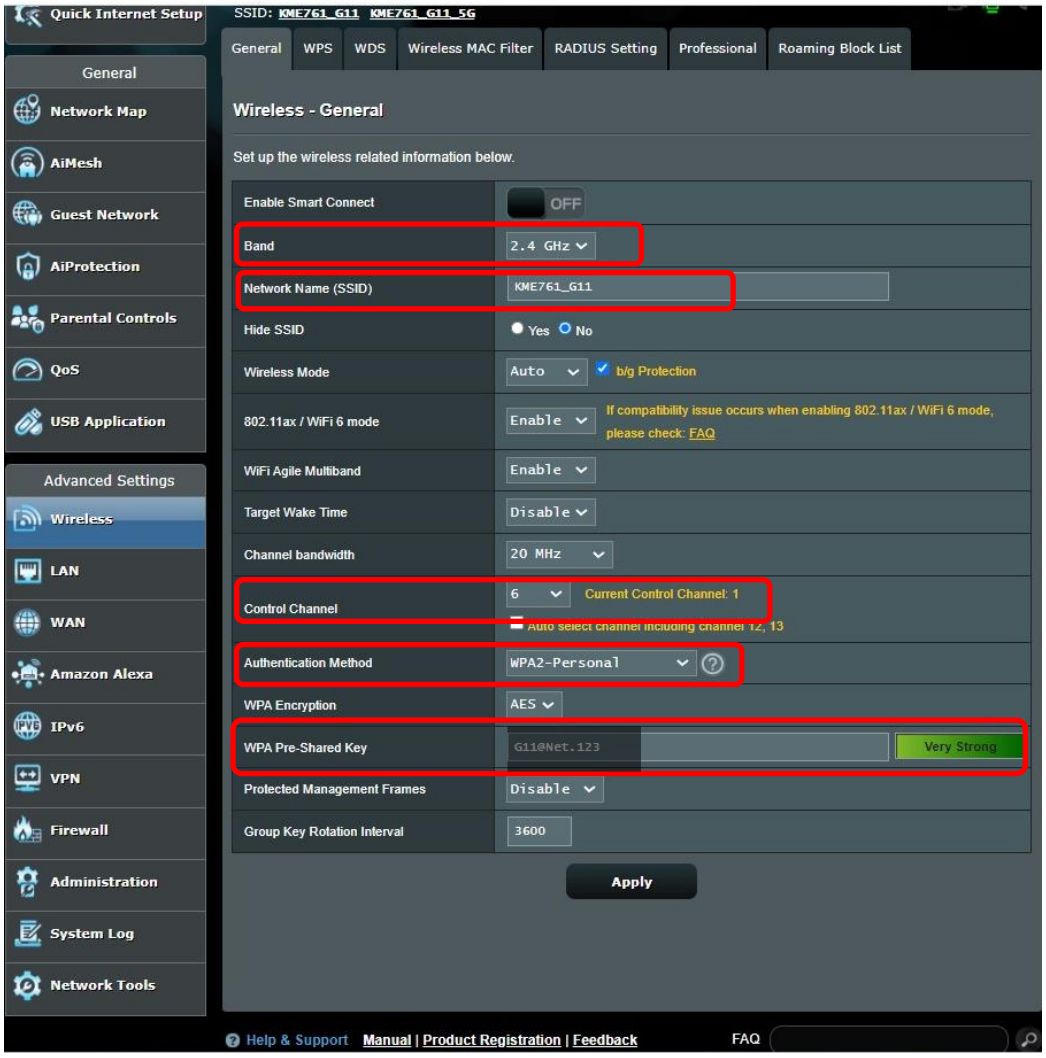
Related screenshot is as follows .



✓ Modify the WLAN settings.

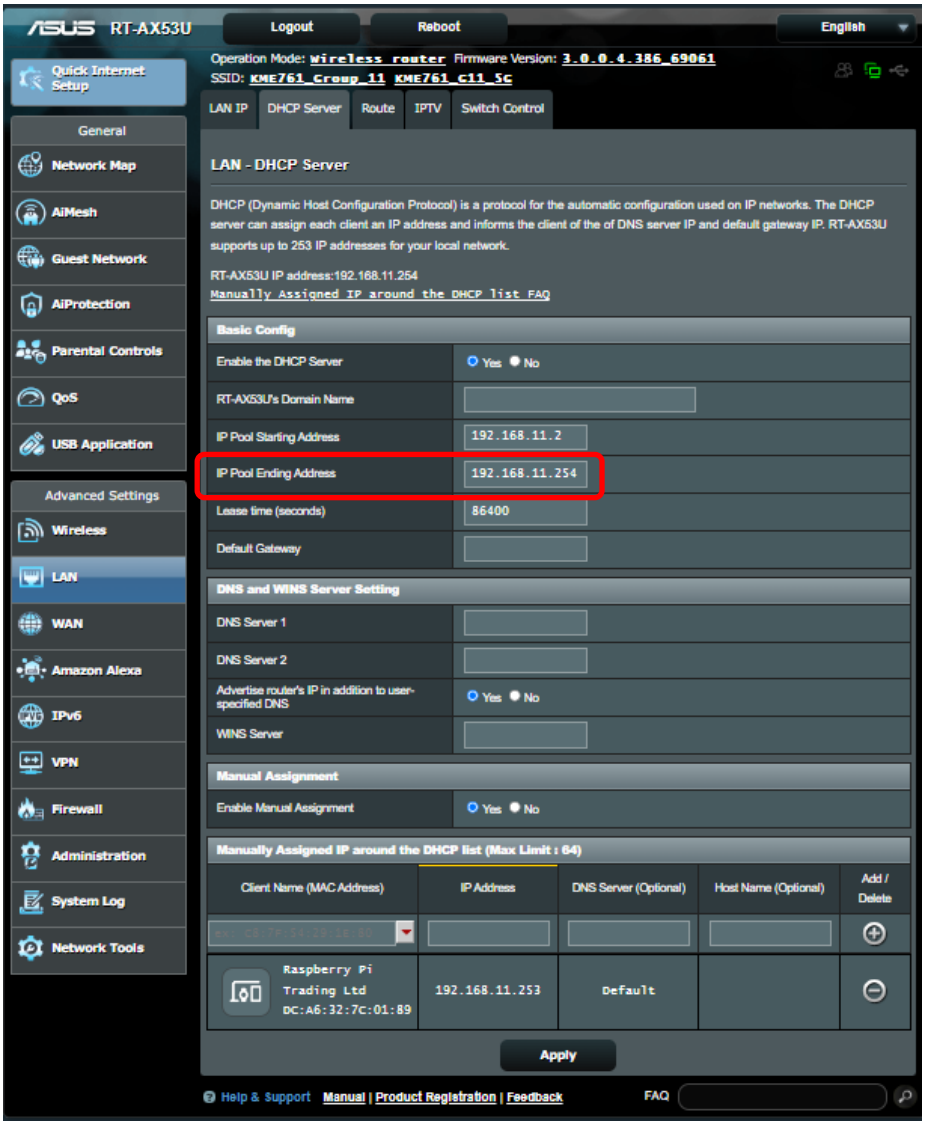
Then enabled only 2.4 GHz Network since the Raspberry Pi Pico W only supports 2.4 GHz connections. After that replaced WLAN SSID, in the Wireless -General settings by locating the SSID as **KME761_Group_11** as per the instructions given.
Then to secure the Wireless Network chose WPA2 (or WPA3 if available) as the security protocol. After that, set a Pre-Shared Key (PSK) or Wi-Fi password for the network. It should be a strong, unique password.

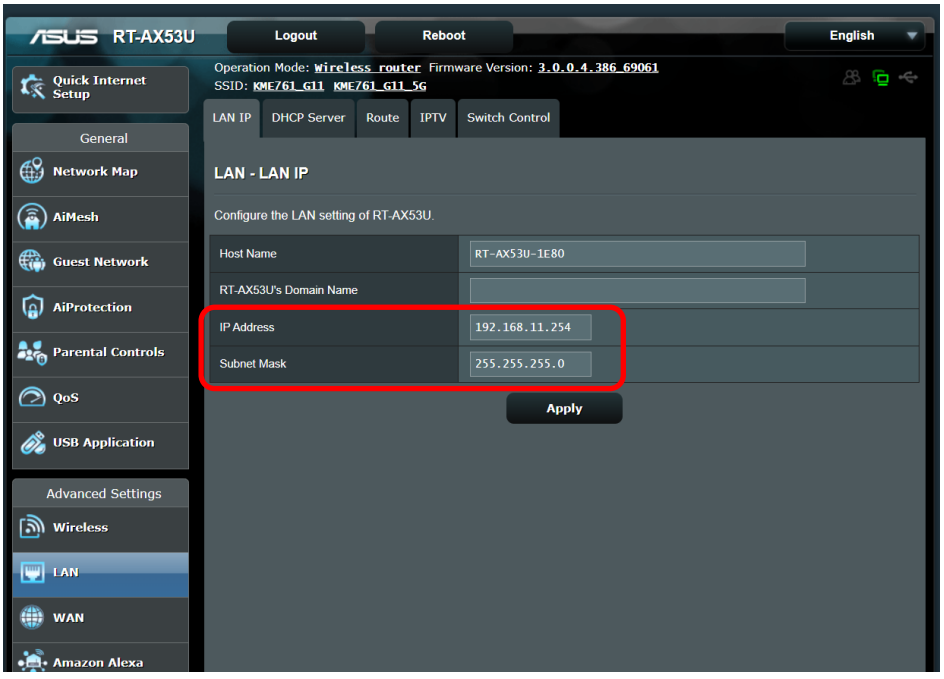
Related screenshot is as follows:



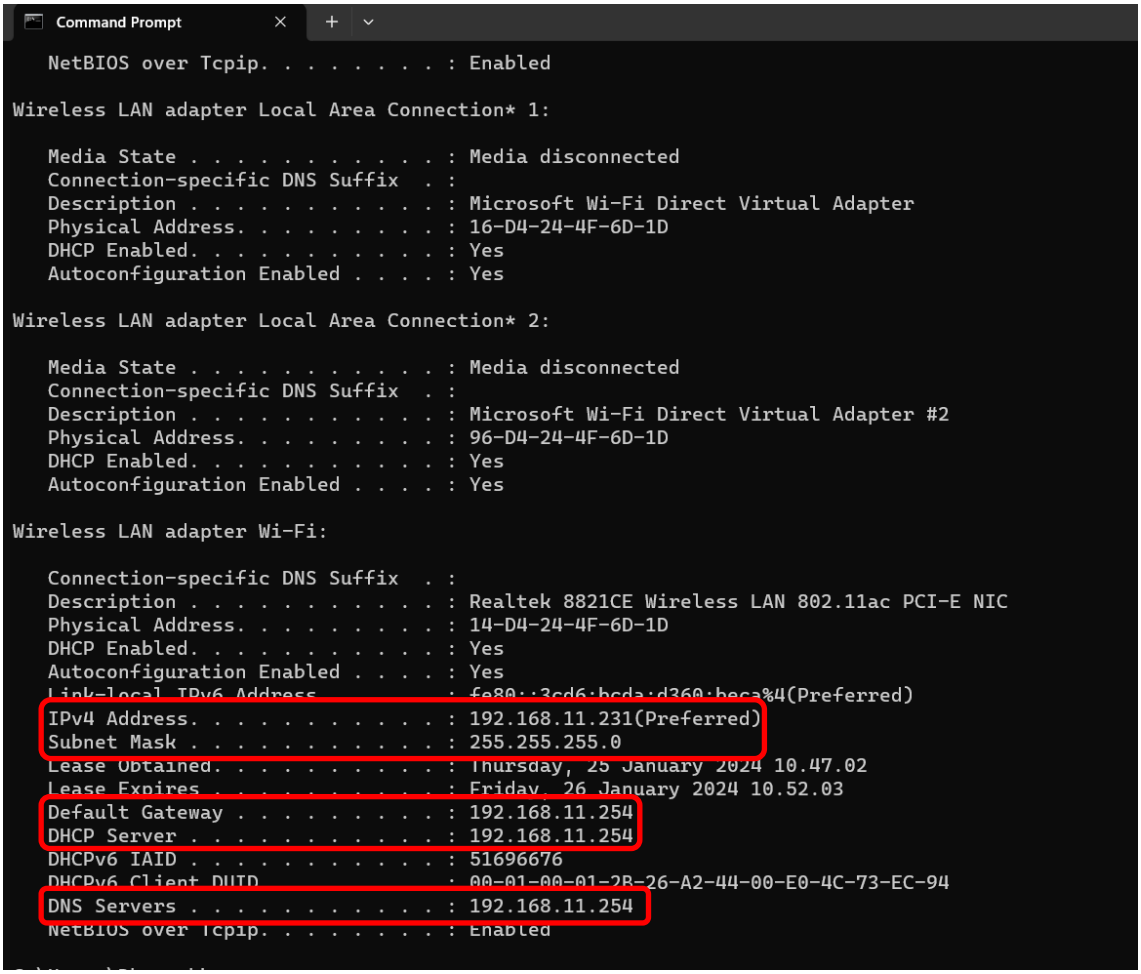
After navigating into the LAN settings section, set the IP address of the router to 192.168.11.254, where 11 is the number of our group. Then set the subnet mask to 255.255.255.0. This configuration ensures that your network address is 192.168.11.0, with the default gateway (router) address being 192.168.11.254. Finally, save changes by clicking on the "Apply" button. By following those steps, we've successfully modified the WLAN settings of our router to meet the specified requirements. This includes enabling only the 2.4 GHz network, replacing the default SSID, securing the network with WPA2 and a PSK and setting the network address and default gateway according to the provided specifications.

Related screenshots are as follows:





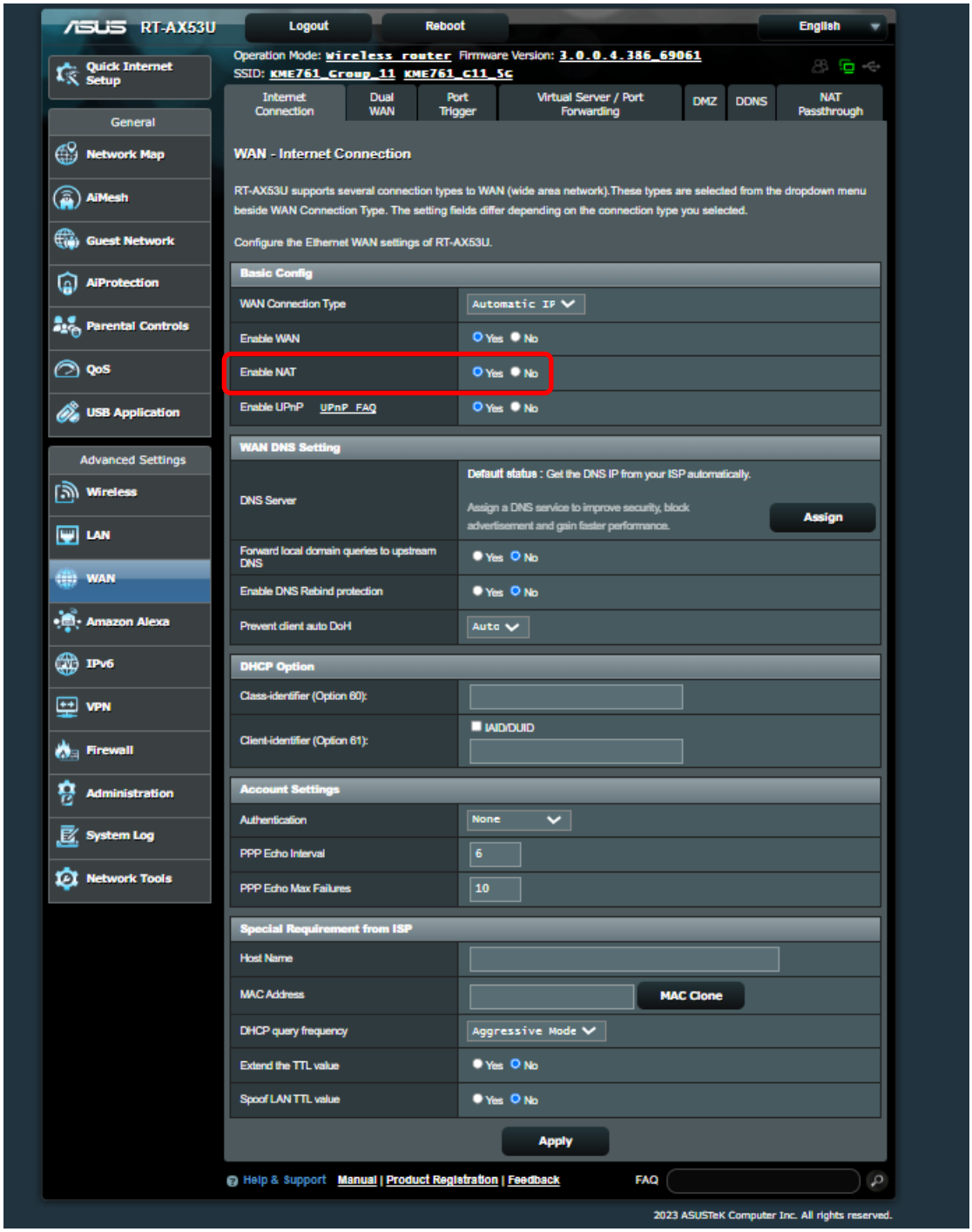
After completing the router configurations, we checked the default gateway server, DNS server, IPV4 address and subnet mask details by using ‘ipconfig’ command on command prompt and it showed as follows.



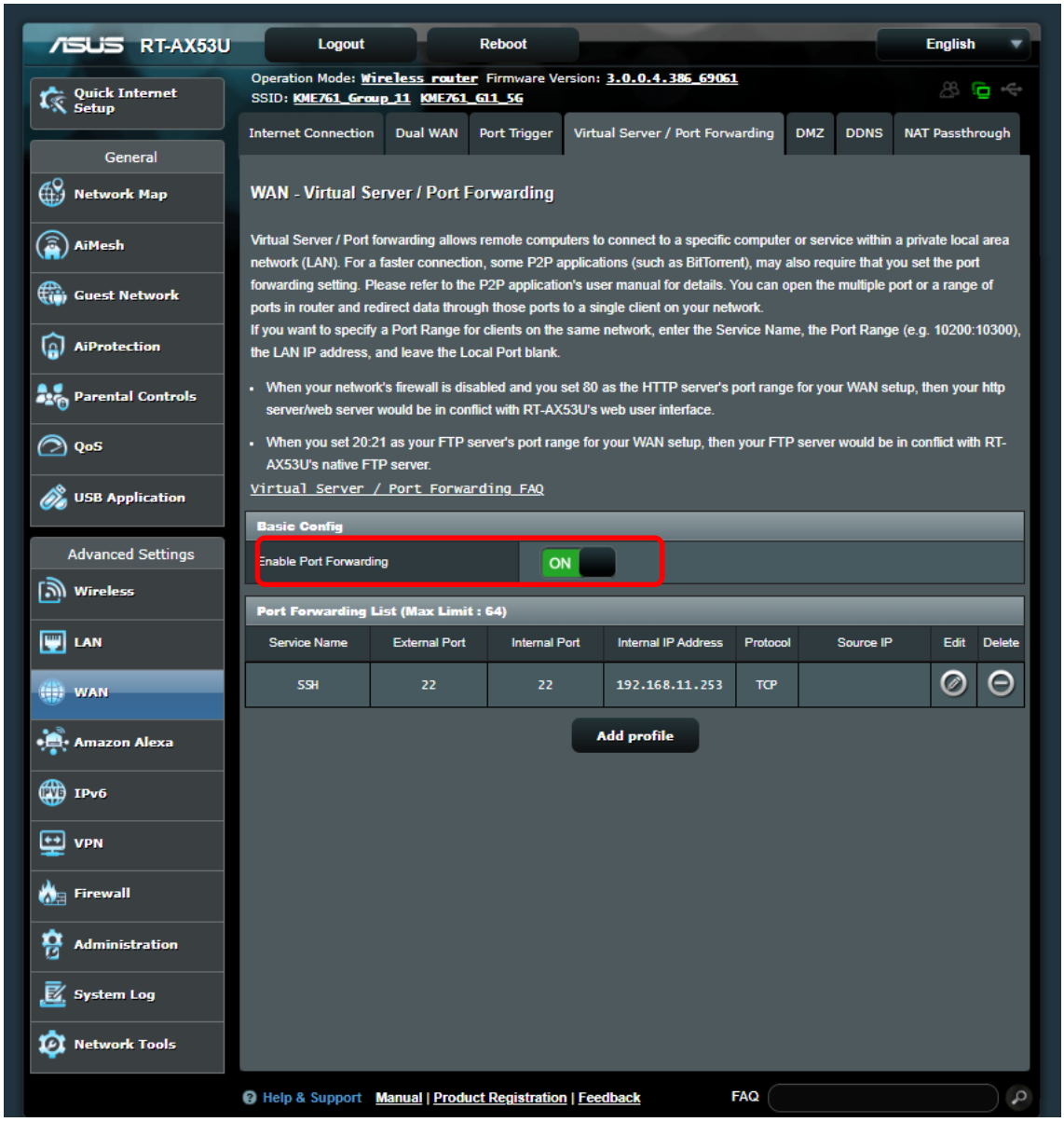
- ✓ Finally, we configured some additional services on the router.

Configured NAT (Network Address Translation) by navigating to the WAN settings, Internet connection section. Enabled NAT between the LAN (local network) and WAN (internet).

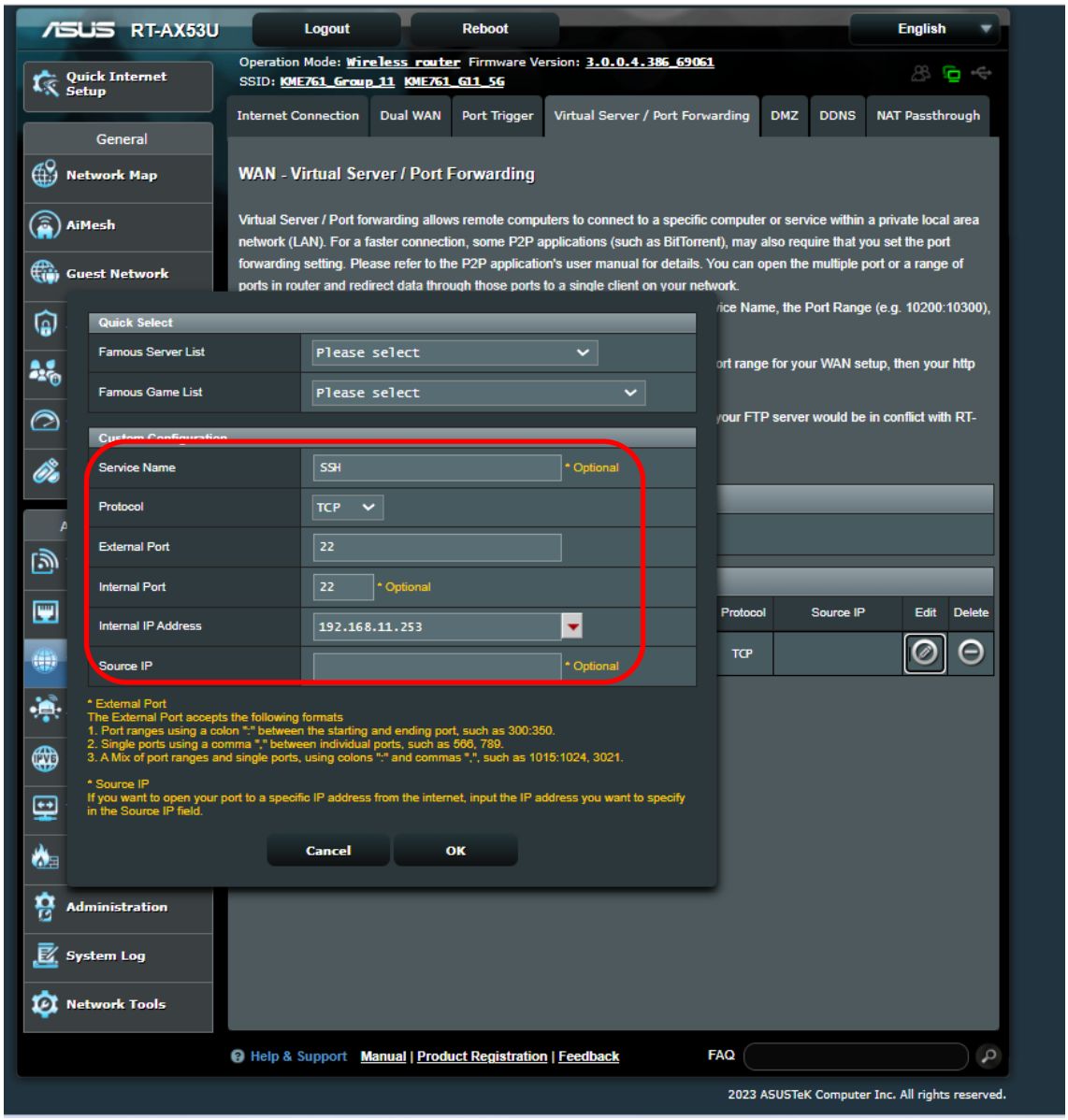
Related screenshot is as follows:



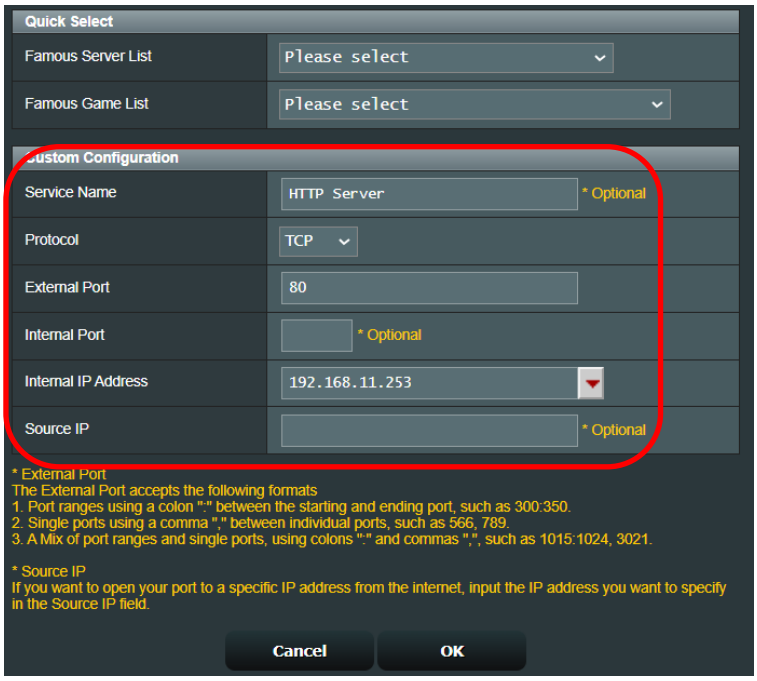
Then configured Port Forwarding or virtual server settings in the router's configuration interface by enabling port forwarding.



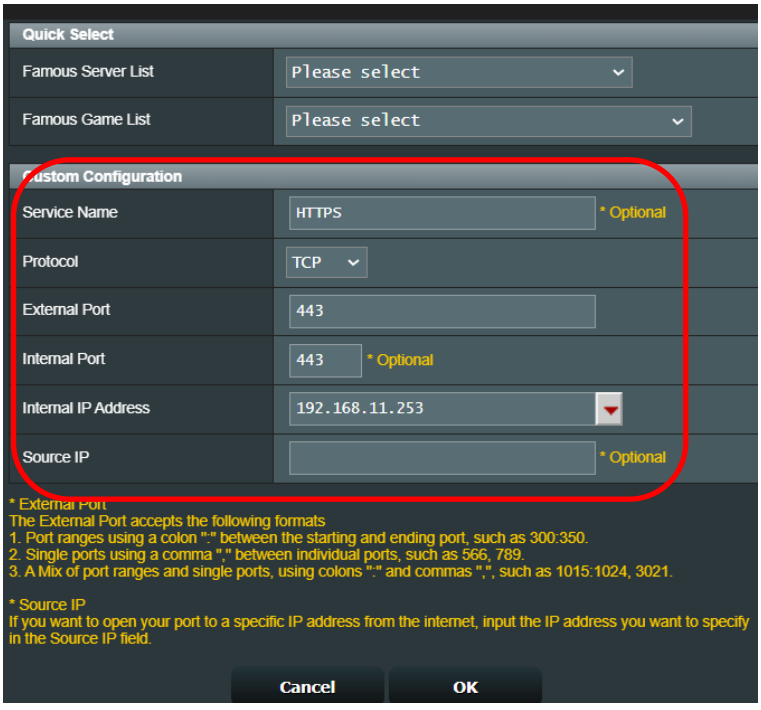
- TCP 22 (SSH): Forwarded this port to the internal IP address of the device, 192.168.11.253 to run the SSH service.



- TCP 80 (HTTP): Forwarded this port to the internal IP address of the device, 192.168.11.253, to run the HTTP service.



- TCP 443 (HTTPS): Forwarded this port to the internal IP address of the device, 192.168.11.253, to run the HTTPS service.

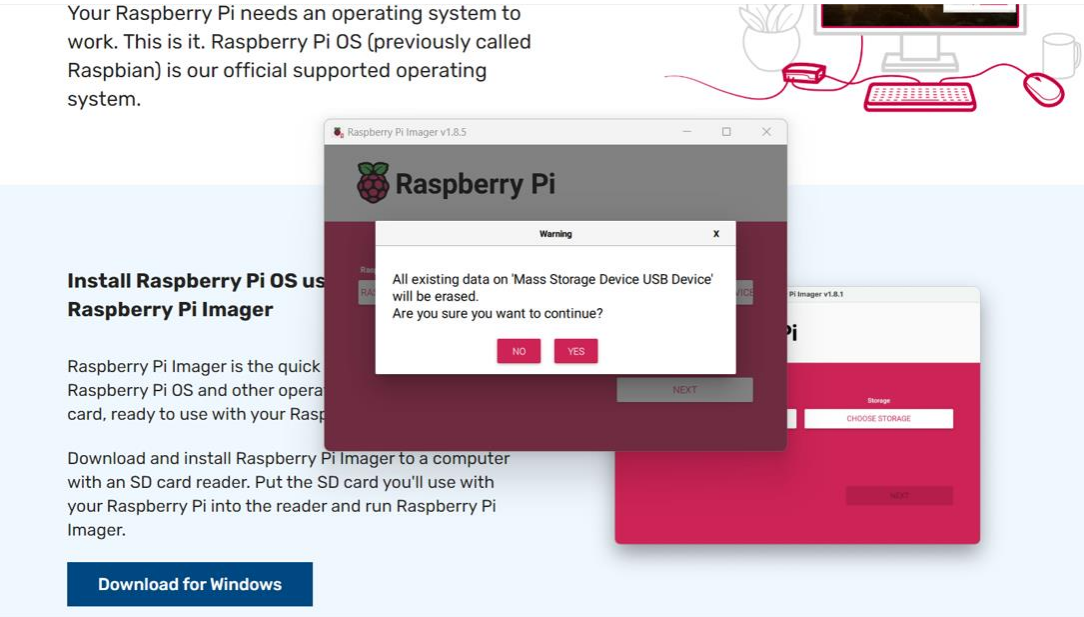


After configuring DHCP, NAT, and port forwarding, saved the changes made to the router's settings.
These configurations allow devices on our network to obtain IP addresses automatically, access the internet through NAT, and receive incoming connections for SSH, HTTP, HTTPS services from external networks.

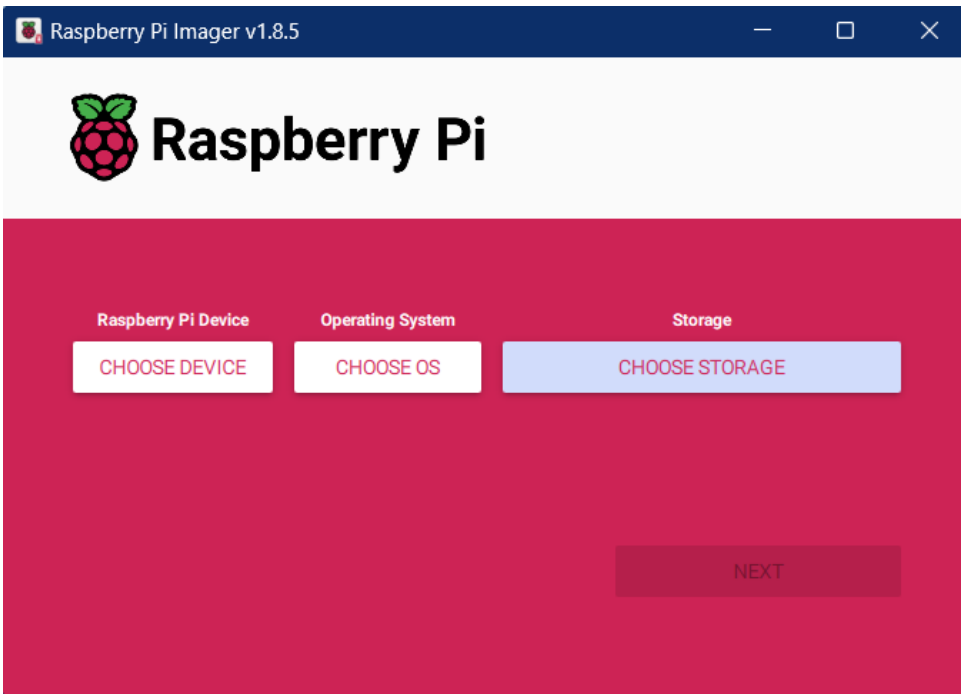
3 Setting up the Raspberry Pi

As a group we completed the following steps to choose the correct operating system image to install.

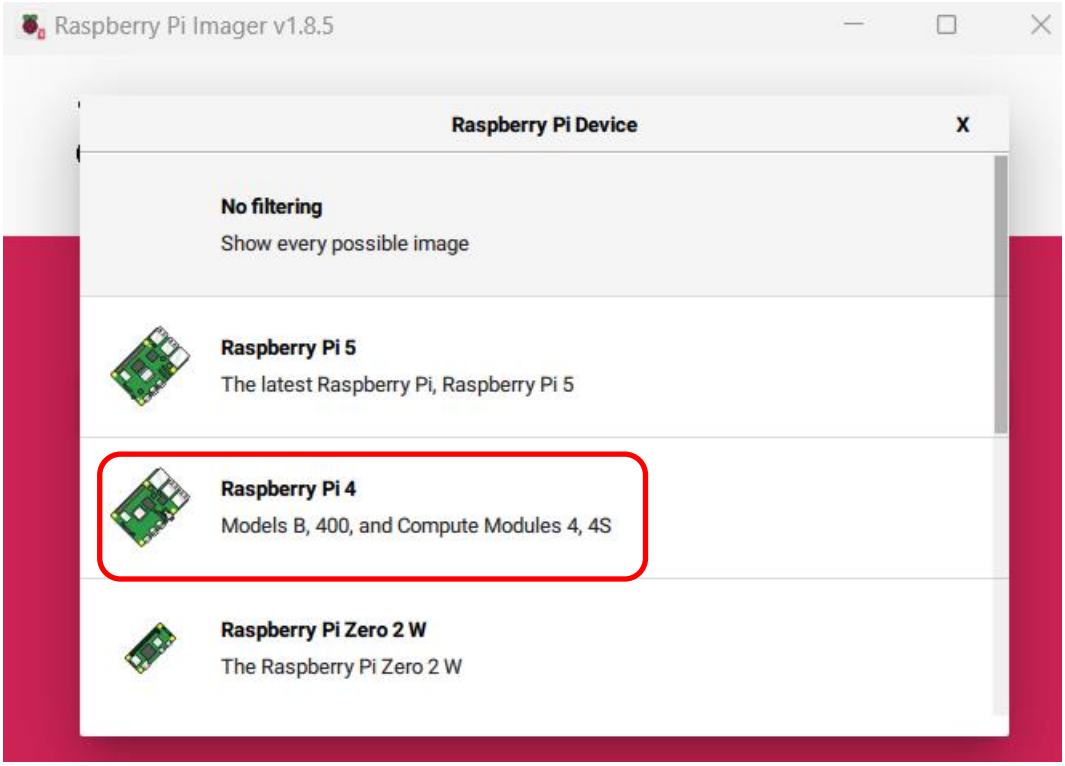
1. Downloaded the Raspberry Pi Imager from https://www.raspberrypi.com/software/ to one of our group members.



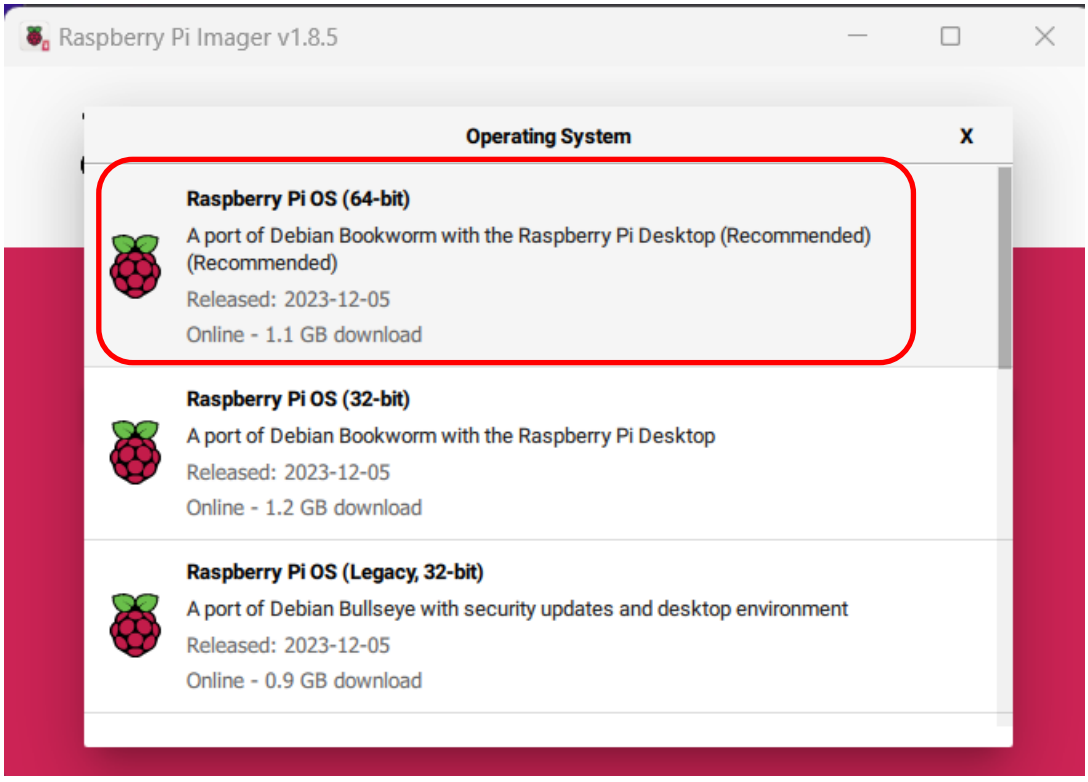
2. Inserted the SD card to an SD card reader and then opened the Raspberry Pi Imager.



3. For the device, we chose the Raspberry Pi 4 device, because we had it for our group work.



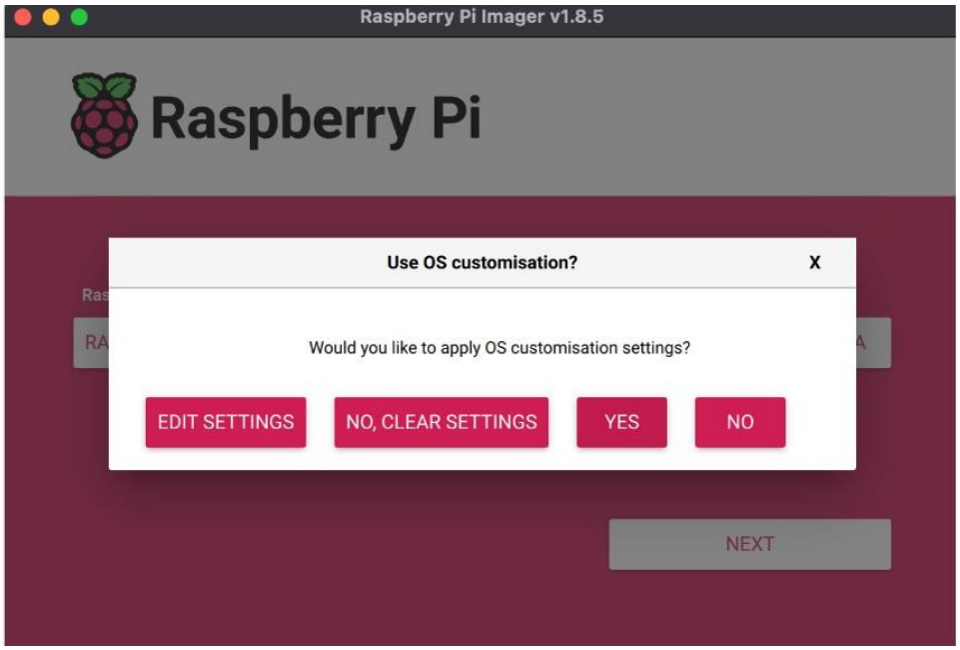
4. For the operating system, chose the default 64-bit Raspberry Pi OS.



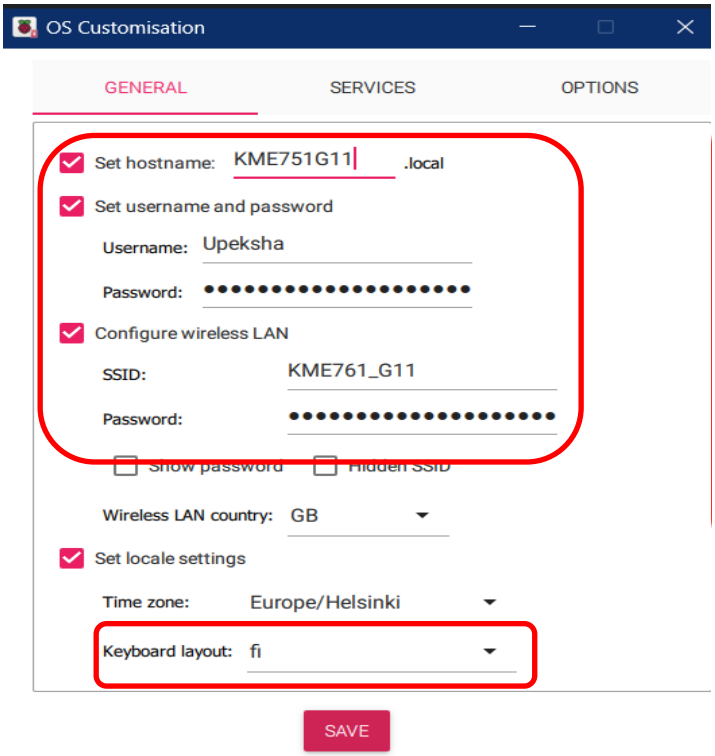
5. For storage, chose the SD card.

Before proceeding with writing the image to the SD card, ensure that we have selected the correct options in the previous steps. To customize the operating system before writing the image to the SD card using the Raspberry Pi Imager, followed these steps:

- ✓ Create a Username and Password:



In the customization settings (EDIT SETTINGS), locate the option to create a username and password. Then applied the desired username and password of one group member. It was the initial login credential for accessing the Raspberry Pi OS.



✓ Configure Network Settings:

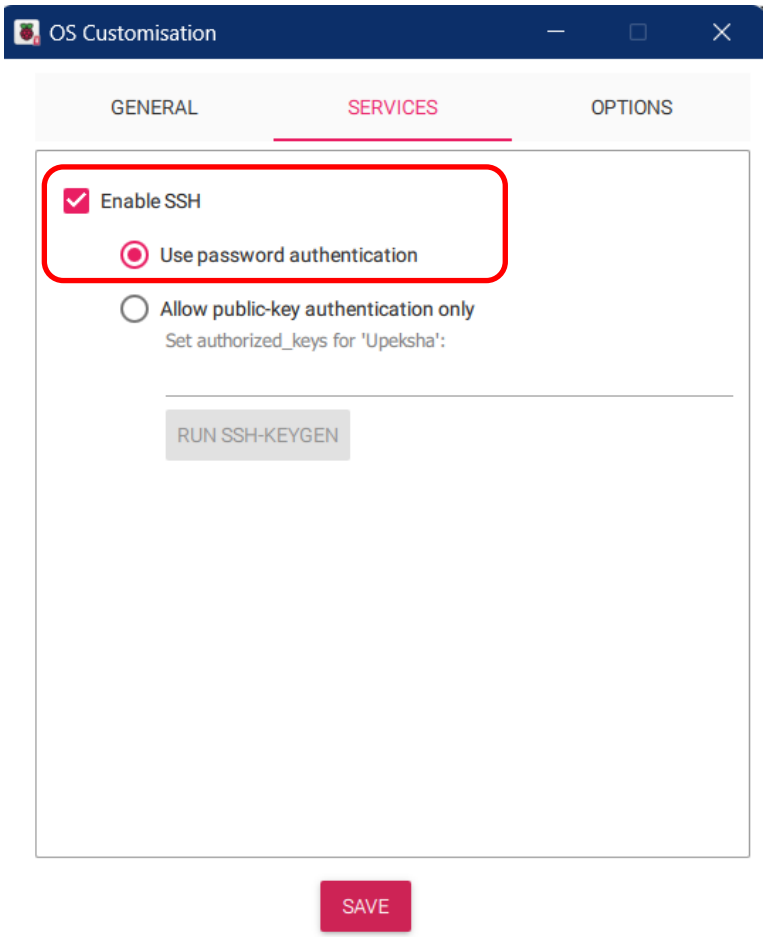
After setting up the username password, we configured the wireless LAN by giving the SSID and password for our group's Wi-Fi network to ensure the Raspberry Pi connects to the correct network upon booting.

✓ Set Keyboard Layout to Finnish (fi):

Then setting up local settings by selecting time zone and the correct keyboard layout to
fi (Finnish) ensure that special characters are mapped correctly for your keyboard.

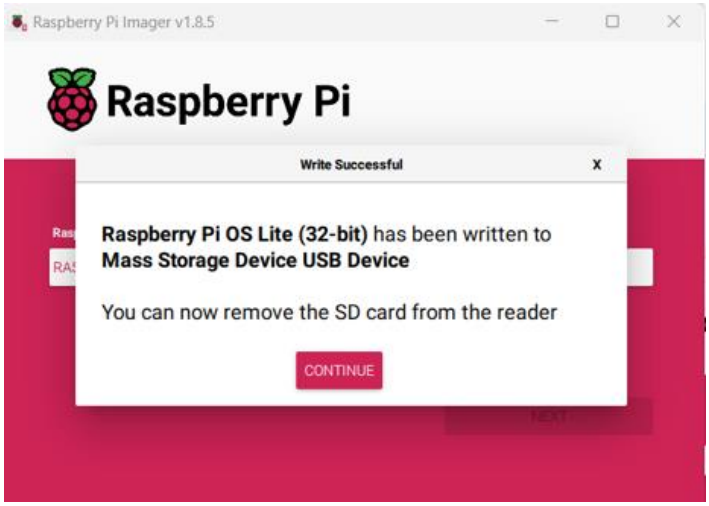
✓ Enable SSH with Password Authentication:

Then configured SSH settings by services, enabled SSH password authentication to allow SSH access using a password.



✓ Save the settings and proceed with writing the operating system image to the SD card.

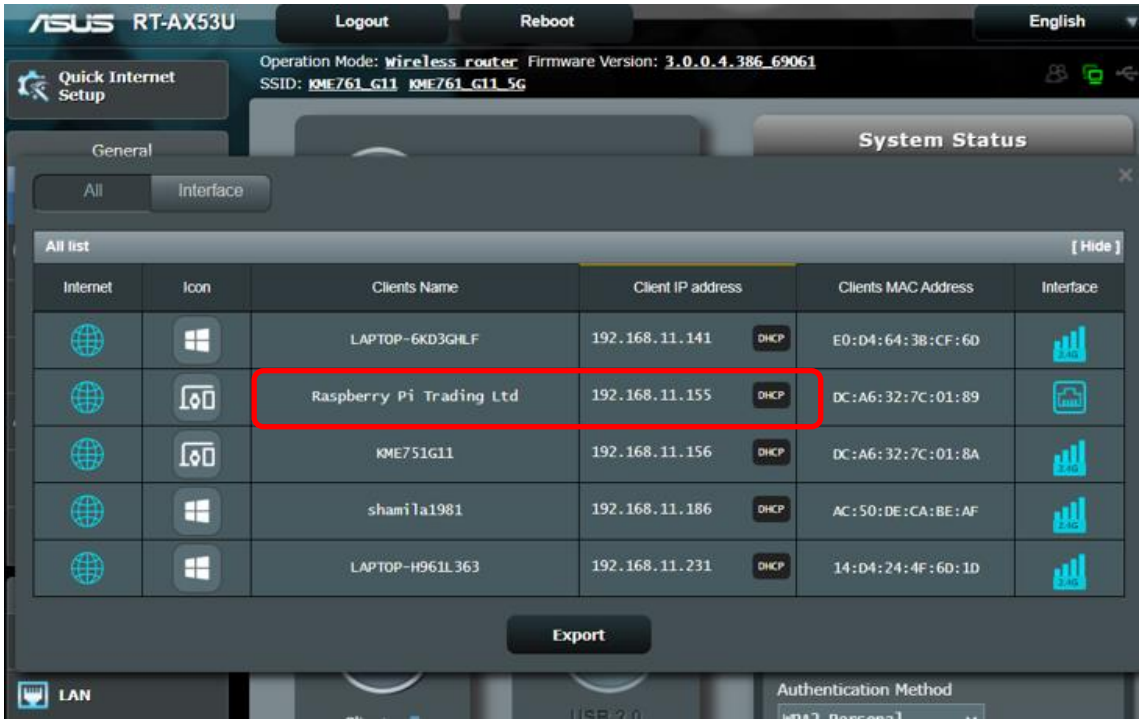
After modifying the settings as required, saved the changes. Verified that all settings are correctly configured according to the assignment and proceeded to write the Operating System Image to the SD Card.



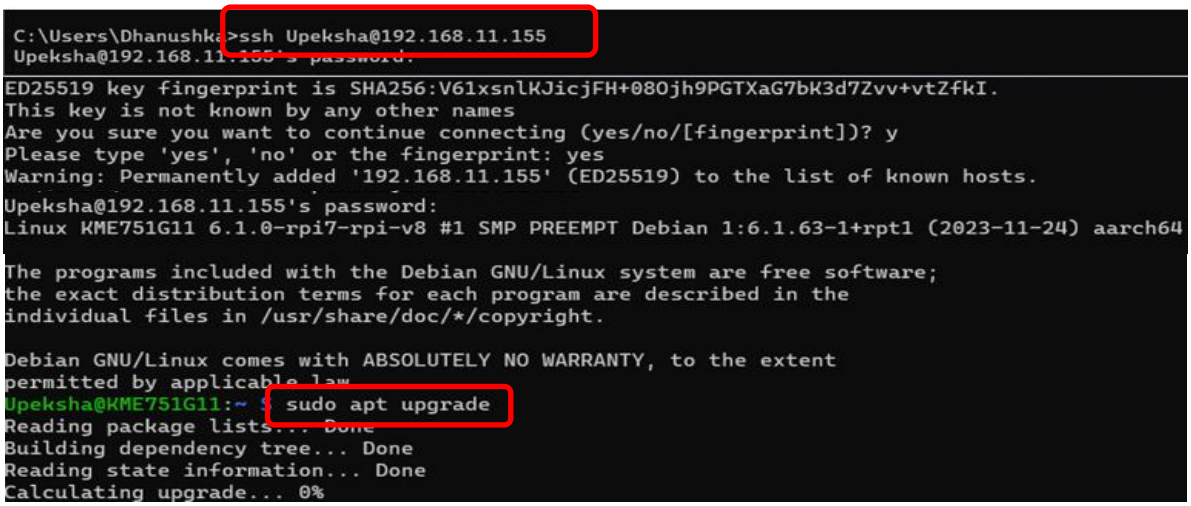
After removing the SD card, insert the SD card into the Raspberry Pi device, ensuring it is powered off. Powered on the Raspberry Pi to initialize the operating system with the customized settings.

- ✓ Connect to the Raspberry Pi remotely by using SSH.

Found the Raspberry Pi's IP Address by logging in to our router's web management interface. Identified the Raspberry Pi among the listed devices and noted down its IP address.



Then opened the command prompt of the computer and used the following method to establish a SSH connection to the Raspberry Pi remotely. It is normal for PuTTY (or another tool you choose to use) to give a warning about accepting encryption keys when you log onto the Raspberry Pi for the first time using SSH. Select “Yes” to accept the connection.



By following these steps, each group member can access the Raspberry Pi remotely using their own user account, facilitating collaborative work and individual customization preferences.

4 Raspberry Pi Configuration

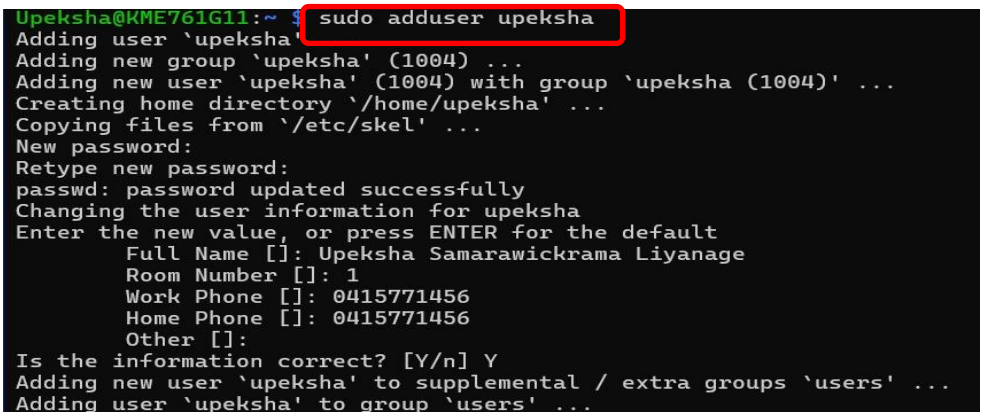
In this exercise it is mainly focusing on managing users, files and settings as well as using ***sudo*** commands on the Linux command line. In the previous exercise 2, as the initial step we created a user account for one member in the group. And in this exercise our task is to create user accounts for the remaining members in the group and adjust the user settings on the Raspberry Pi.

Firstly, we accessed the Raspberry Pi over an SSH connection and then we completed the following tasks.

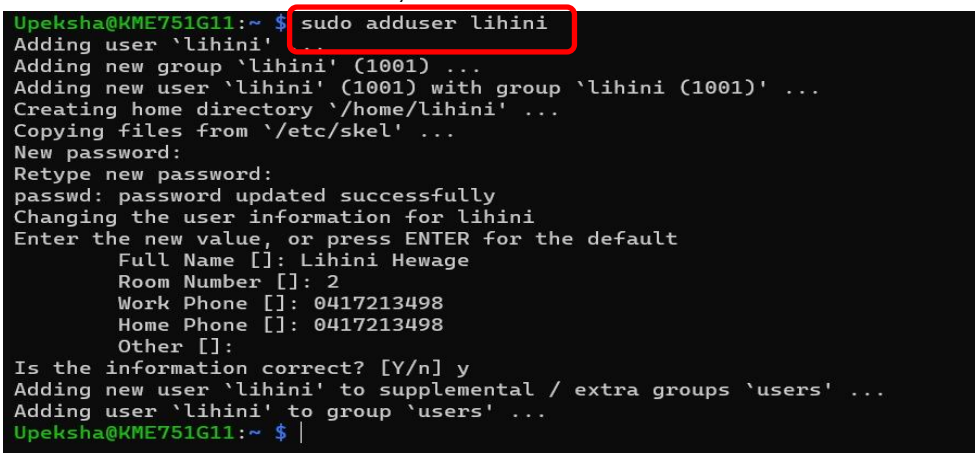
- ✓ Create user accounts for the members who do not have accounts yet & create a password for each user.

Firstly, we created the user account for the 1st member who does not have an account. We used the ***sudo adduser*** command for that. And then we created a new password for the user account we created and after the password updated successfully then we changed the user information for the created user.

Screenshot is as follows,



And we followed the same steps for the remaining group member as well and the screenshot of that is as follows,



```
Upeksha@KME751G11:~ $ sudo adduser shamila
Adding user `shamila' ...
Adding new group `shamila' (1002) ...
Adding new user `shamila' (1002) with group `shamila (1002)' ...
Creating home directory `/home/shamila' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for shamila
Enter the new value, or press ENTER for the default
        Full Name []: Shamila Tennakoon
        Room Number []: 3
        Work Phone []: 94714053560
        Home Phone []: 94714053560
        Other []:
Is the information correct? [Y/n] y
Adding new user `shamila' to supplemental / extra groups `users' ...
Adding user `shamila' to group `users' ...
Upeksha@KME751G11:~ $
```

✓ Create a group and add all group members to that group.

Then we created a group using the ***sudo groupadd*** command by naming the group as KME761G11. Then we used ***sudo usermod -aG KME761G11*** command to add members to the created group. Following you can find the screenshot of this,

```
Upeksha@KME761G11: ~
Upeksha@KME761G11:~ $ sudo groupadd KME761G11
Upeksha@KME761G11:~ $ getent group | grep KME761G11
KME761G11:x:1003:
Upeksha@KME761G11:~ $ sudo usermod -aG team Shamila
usermod: group 'team' does not exist
Upeksha@KME761G11:~ $ sudo usermod -aG KME761G11 Shamila
usermod: user 'Shamila' does not exist
Upeksha@KME761G11:~ $ sudo usermod -aG KME761G11 lihini
Upeksha@KME761G11:~ $ sudo usermod -aG KME761G11 shamila
Upeksha@KME761G11:~ $ cat /etc/group | grep KME761G11
KME761G11:x:1003:lihini,shamila
Upeksha@KME761G11:~ $
```

```
Upeksha@KME761G11:~ $ getent group KME761G11
KME761G11:x:1003:lihini,shamila
Upeksha@KME761G11:~ $ sudo usermod -aG KME761G11 upeksha
Upeksha@KME761G11:~ $ getent group | grep KME761G11
KME761G11:x:1003:lihini,shamila,upeksha
Upeksha@KME761G11:~ $
```

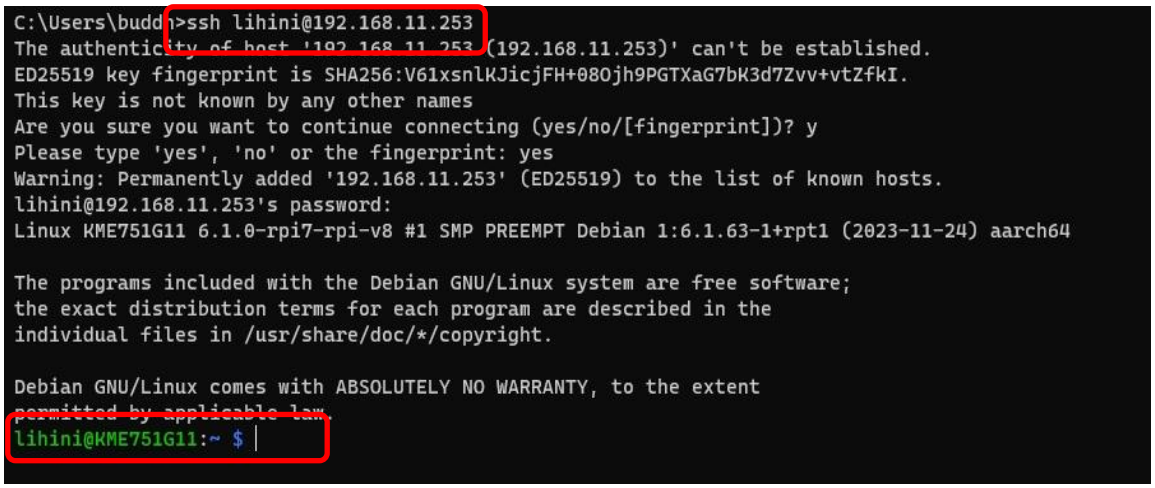
✓ Make sure that all group members can use sudo commands.

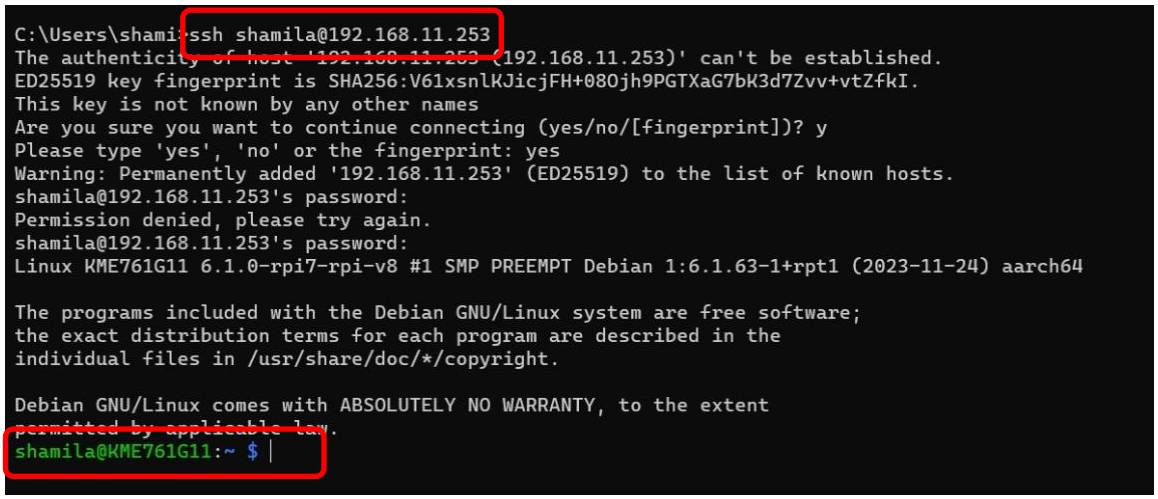
Then each member tried the sudo commands and following are the screenshots for that,

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 29 09:00:18 2024 from 192.168.11.231
Upeksha@KME761G11:~ $ sudo usermod -a -G sudo lihini
Upeksha@KME761G11:~ $ sudo usermod -a -G sudo shamila
Upeksha@KME761G11:~ $ grep sudo /etc/group
sudo:x:27:Upeksha,lihini,shamila
Upeksha@KME761G11:~ $
```

- ✓ Test that all group members can now access the Raspberry Pi remotely using their own credentials.

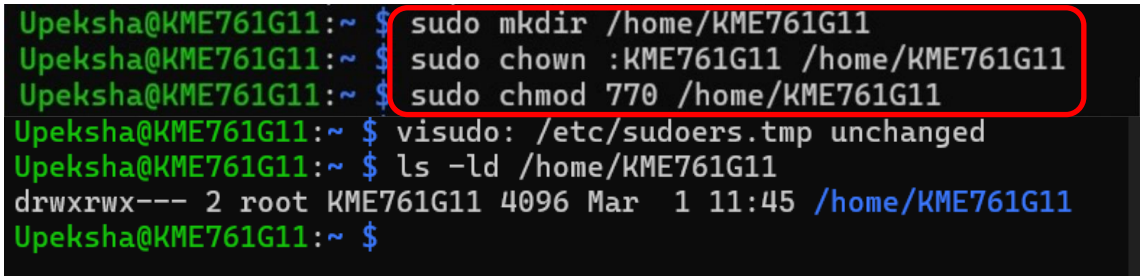
Then we accessed the Raspberry Pi remotely with the created user's own credentials, and we have attached the screenshots of it here,





- ✓ Create a group directory.

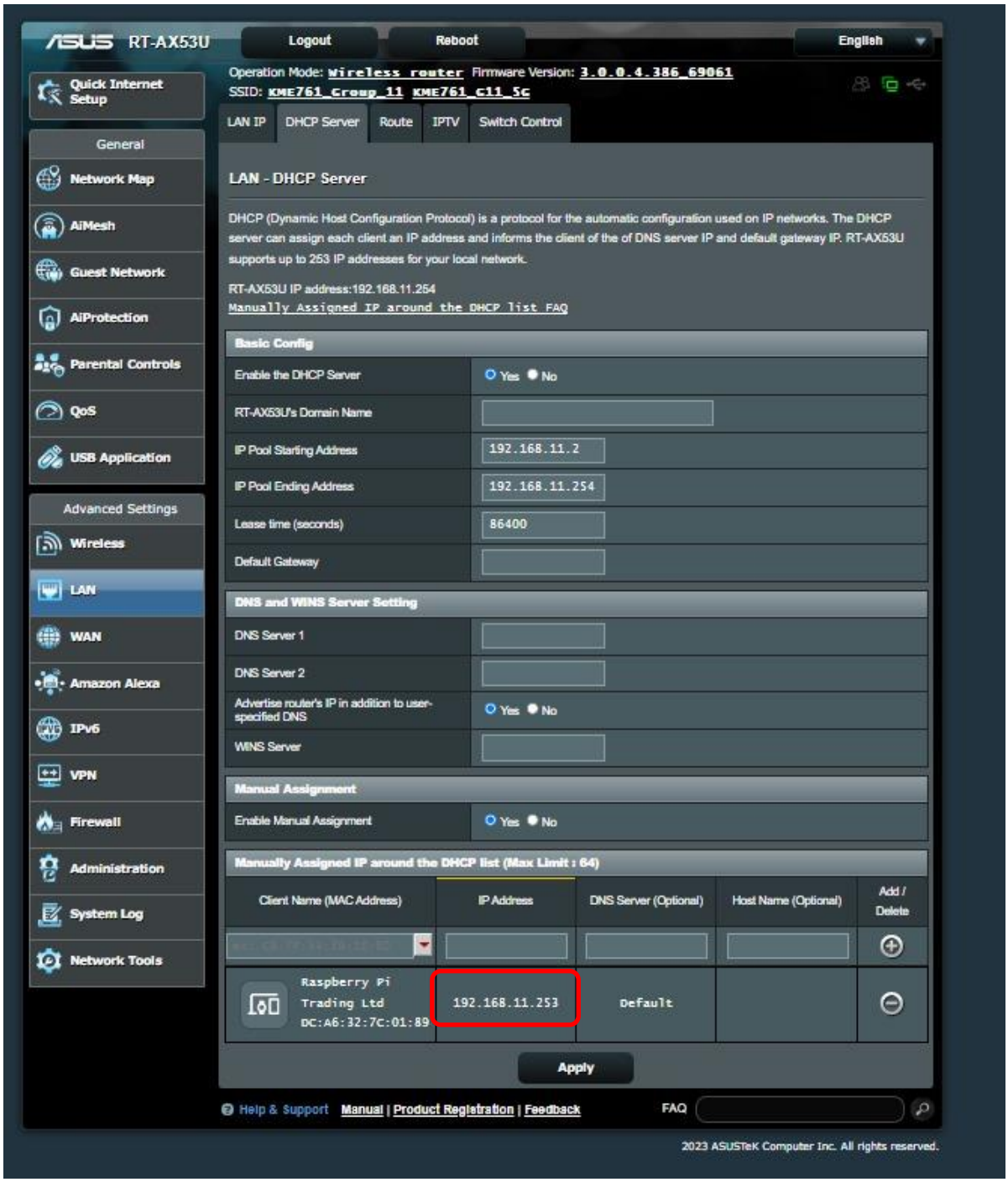
Created a directory, where KME761G11 is our group name, configured the directory in that way, that only members of group 11 can read, write, and execute that directory and files within it. Screenshots are as follows.



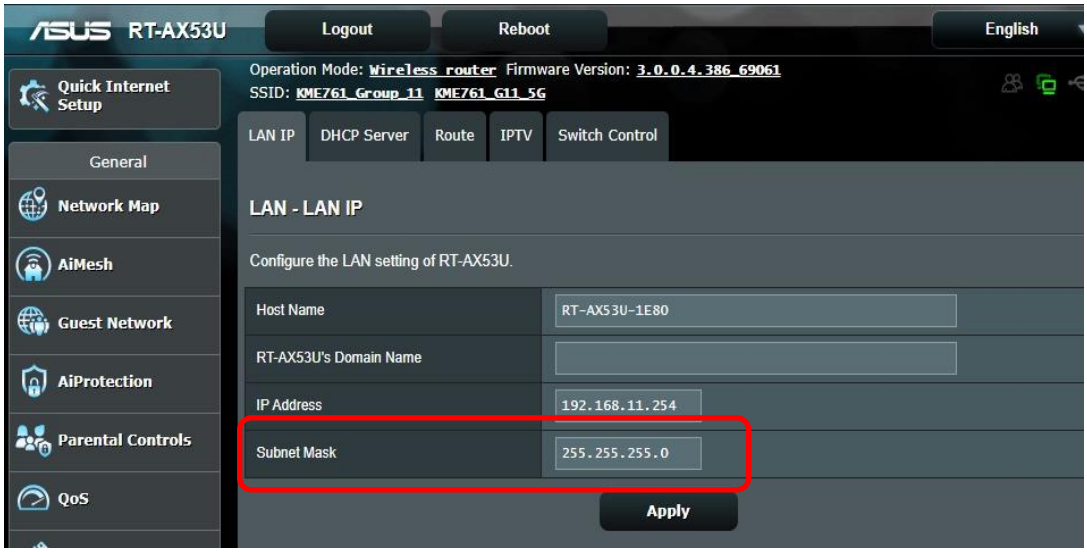
✓ Setting a static IP for the Raspberry Pi

To enable seamless communication between the Pico board and the computer, we'll utilize the Raspberry Pi as an MQTT message broker, taking a static IP address for the Raspberry Pi. The IP addressing information should be as follows:

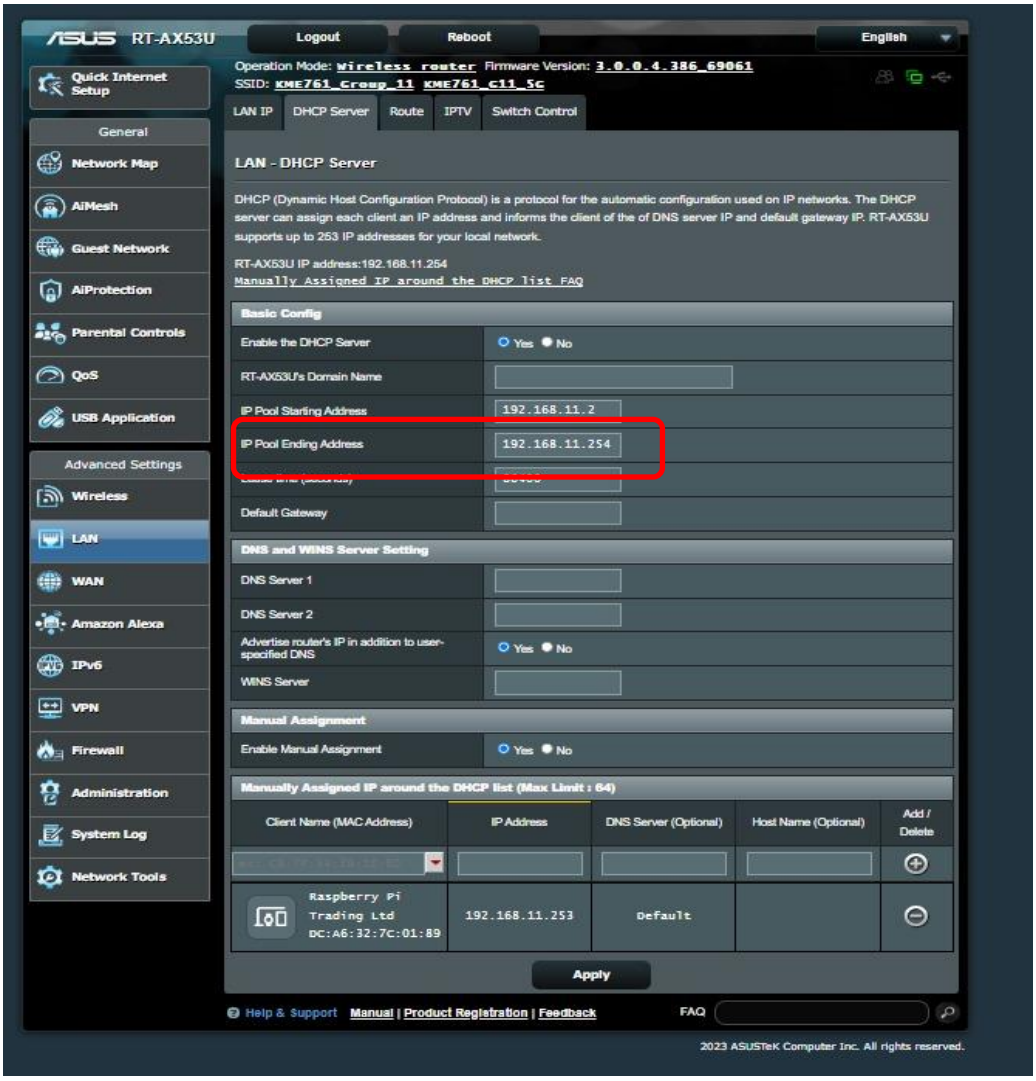
1. IP address - 192.168.11.253 (11 is the group number).



2. Subnet mask - 255.255.255.0



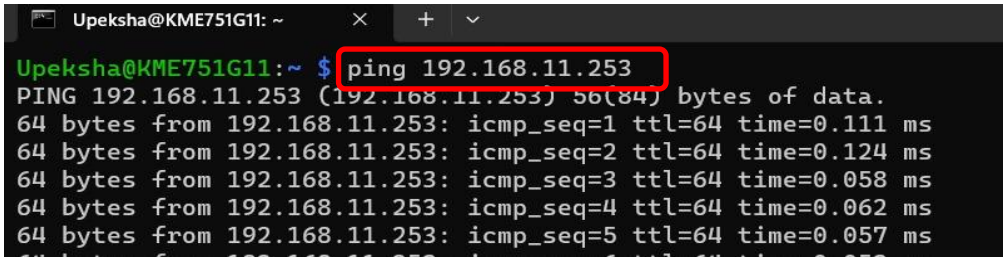
3. Default gateway 192.168.11.254 (Router address). & DNS 192.168.11.254 (Router address)

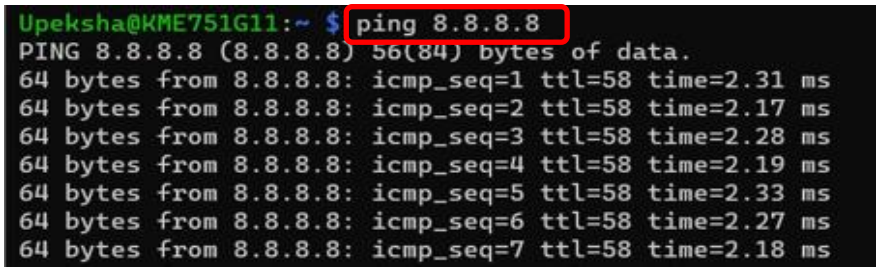


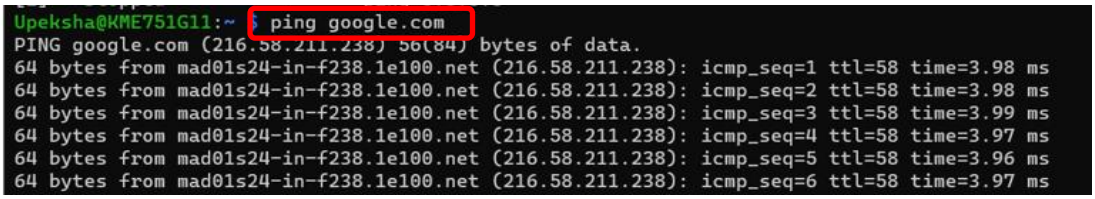
✓ Testing connectivity to the Raspberry Pi.

Ping the default gateway to check local connectivity, ping Google's DNS server at 8.8.8.8 to test connection to the Internet and ping Google's website www.google.com to test DNS.

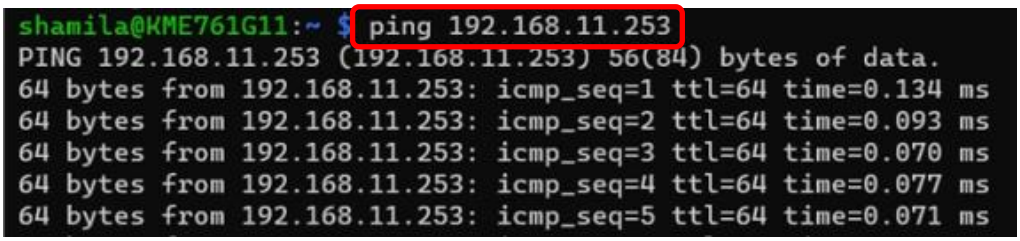
User 1:

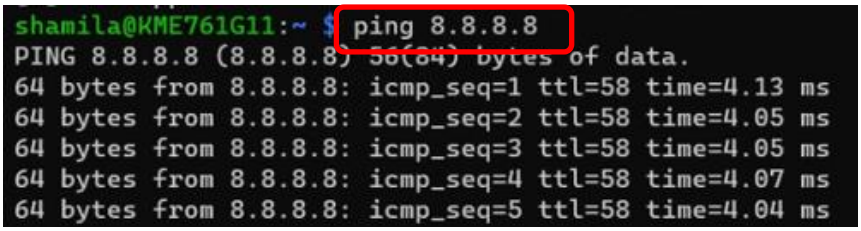


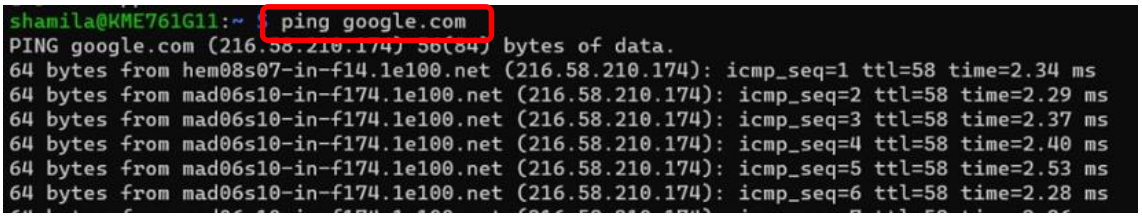




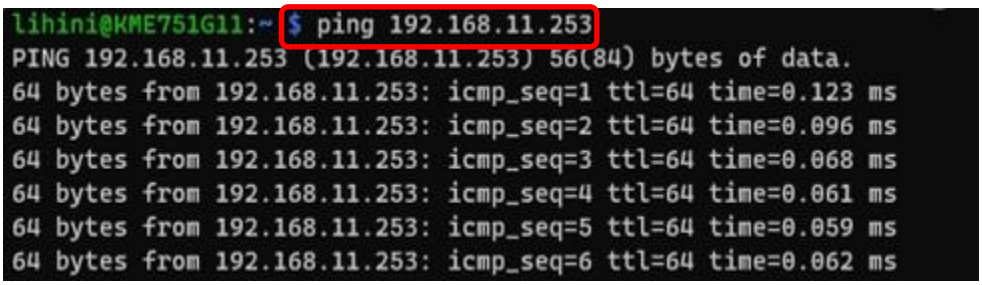
User 2:

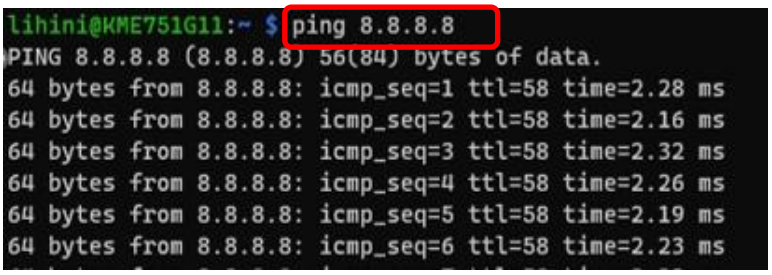


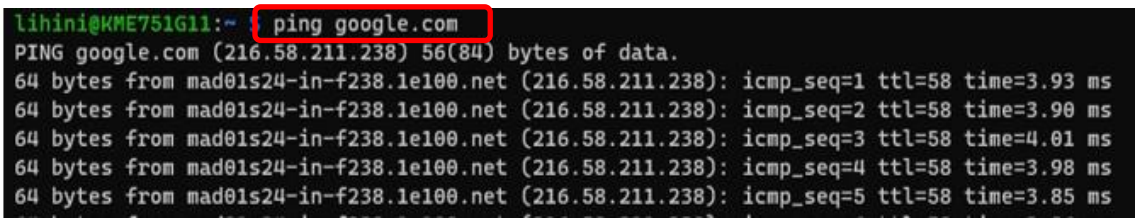




User 3:

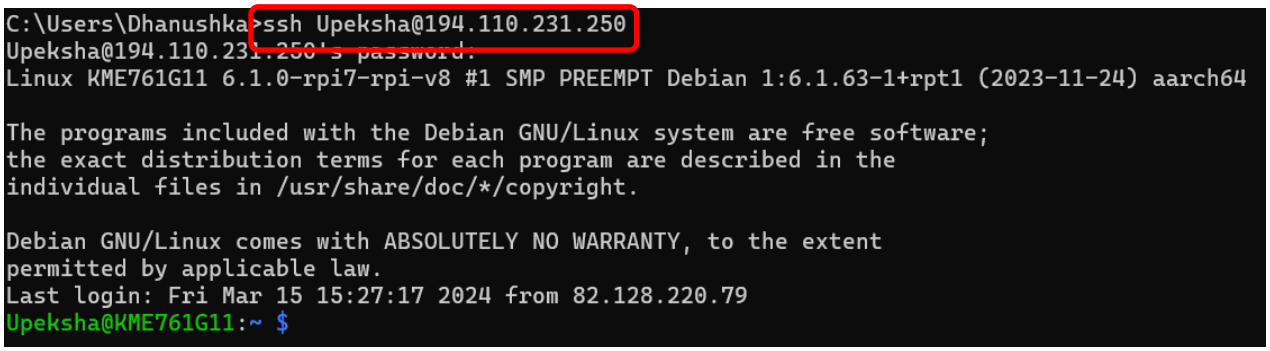




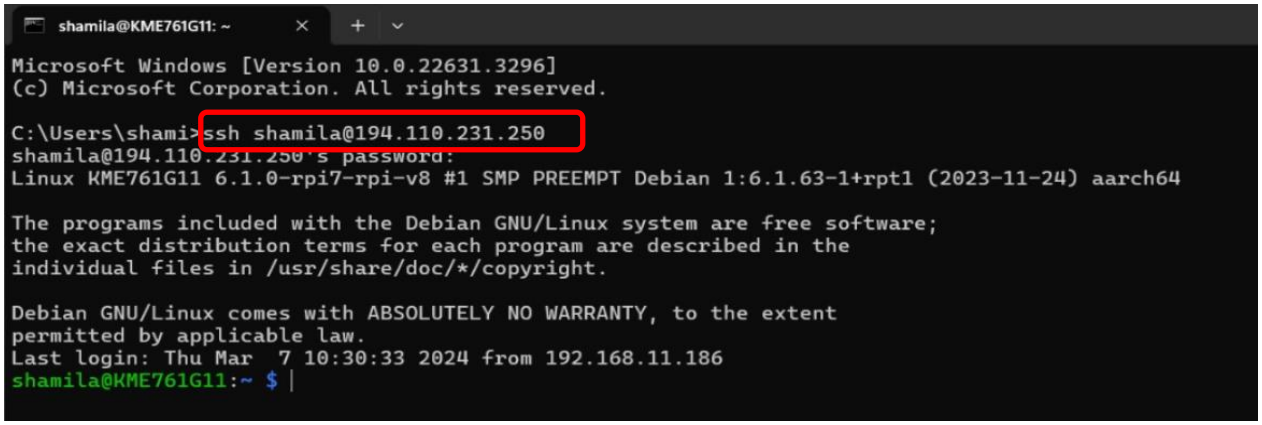


- ✓ Verify that all group members can access the Raspberry Pi over SSH using the ***new IP address***.

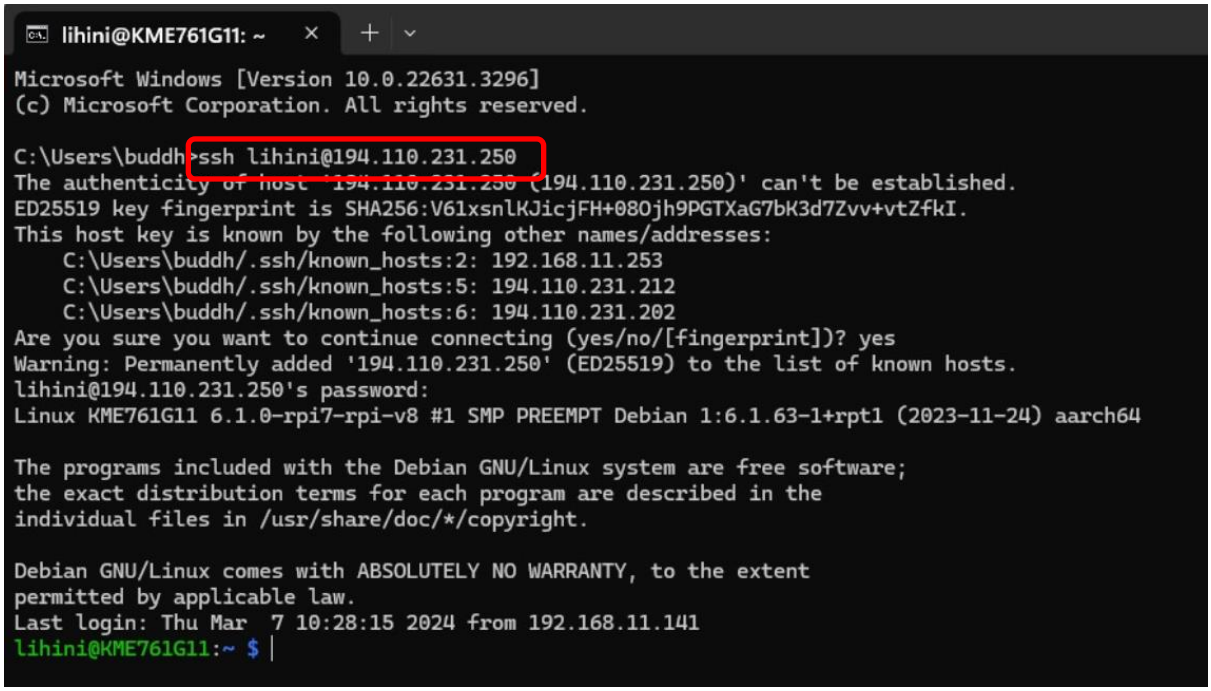
User1:



User2:



User3:

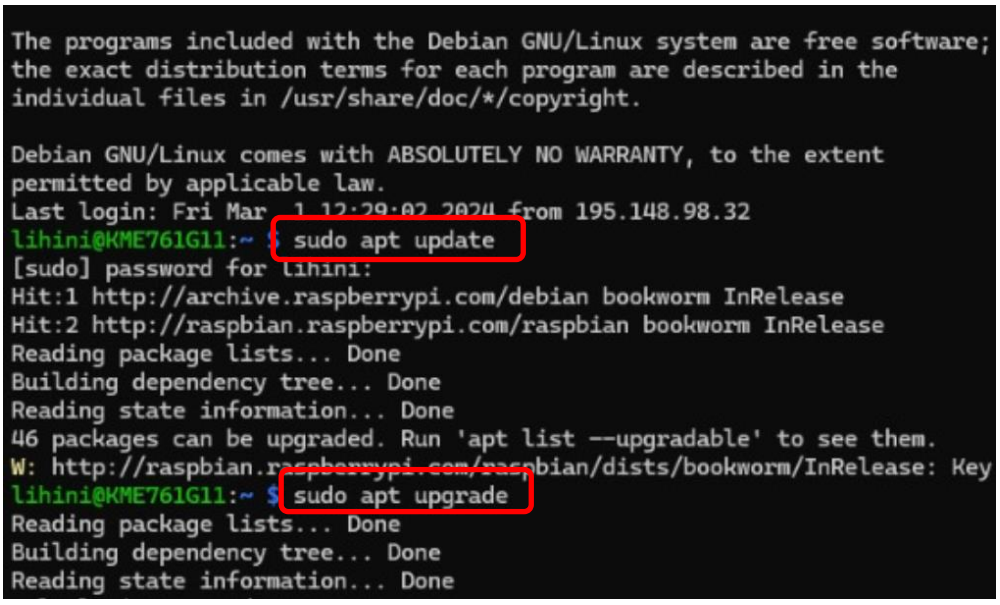


- ✓ Install Apache2 web server:

As a next step it is to install the Apache 2 web server to connect the host to the web. Before Install Apache 2 it is needed to make sure that the Raspberry pi OS is updated.

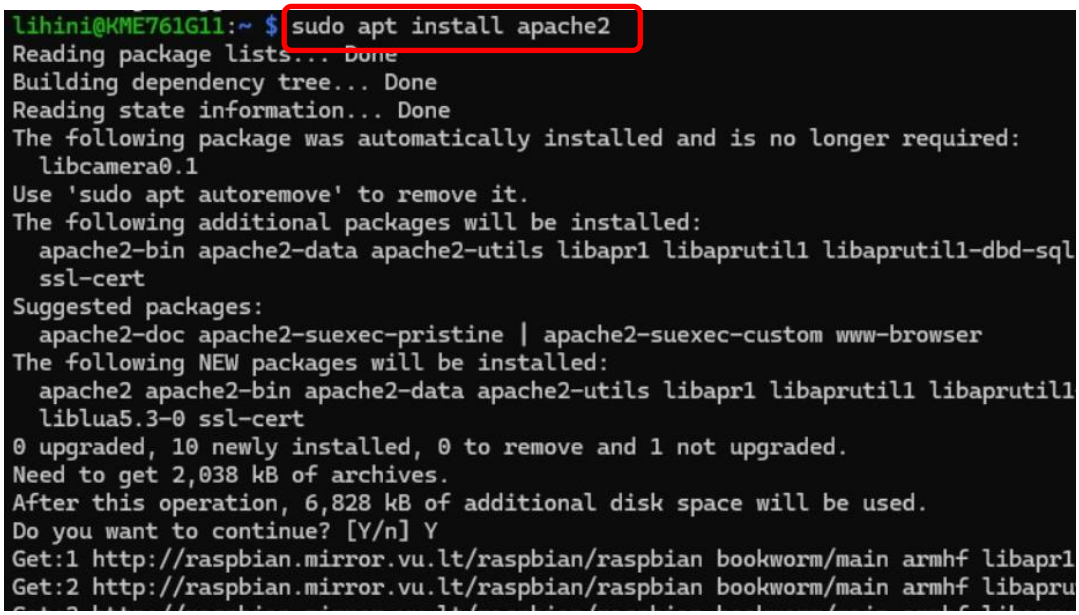
1. Update and upgrade the Raspberry Pi operating system.

Raspberry Pi Pico is updated by using the command ***sudo apt update*** and upgrade using command ***sudo apt upgrade***.
Screenshots of update and upgrade are as follows.

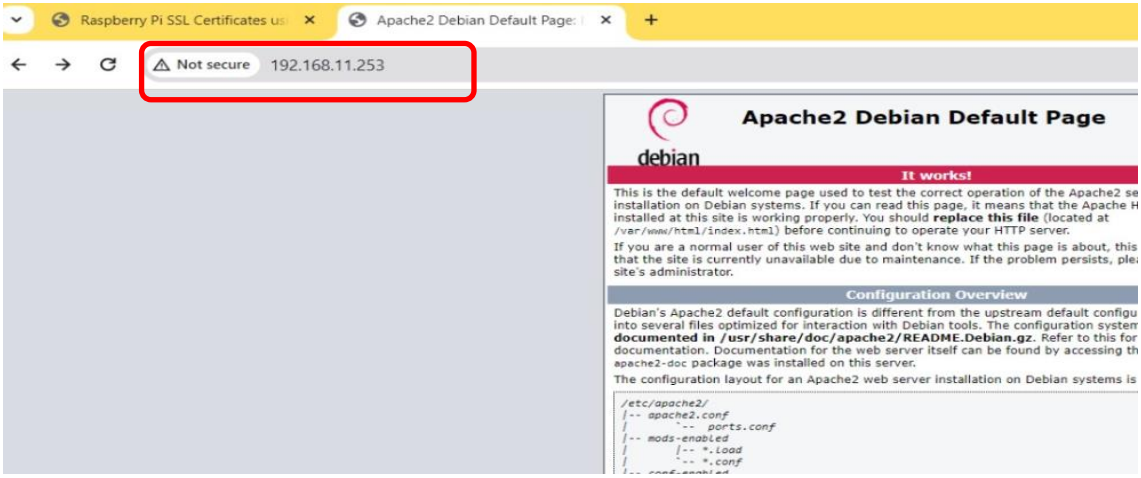


2. Then Apache 2 web server is Installed.

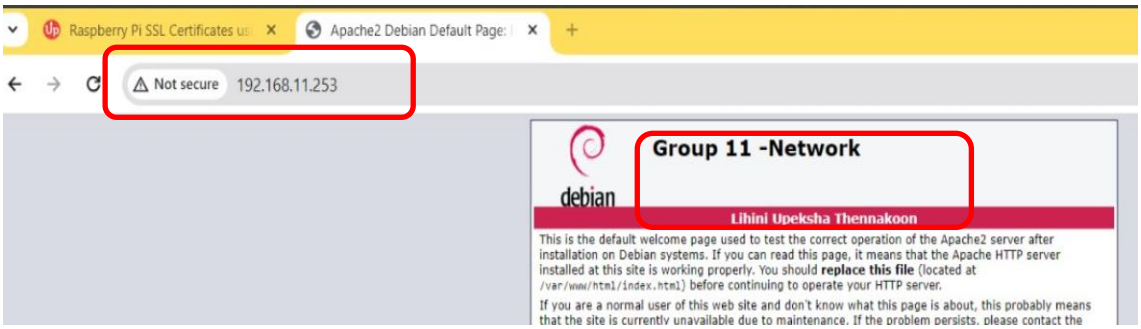
   sudo apt install apache2 is the command used for installation and the relevant screenshot of it is as follows.



3. Then it is tested the apache2 default install web page interface in the following address http://192.168.11.253 and the interface was as follows



4. The webpage is edited to contain the group members' names. Here is the screenshot of it:



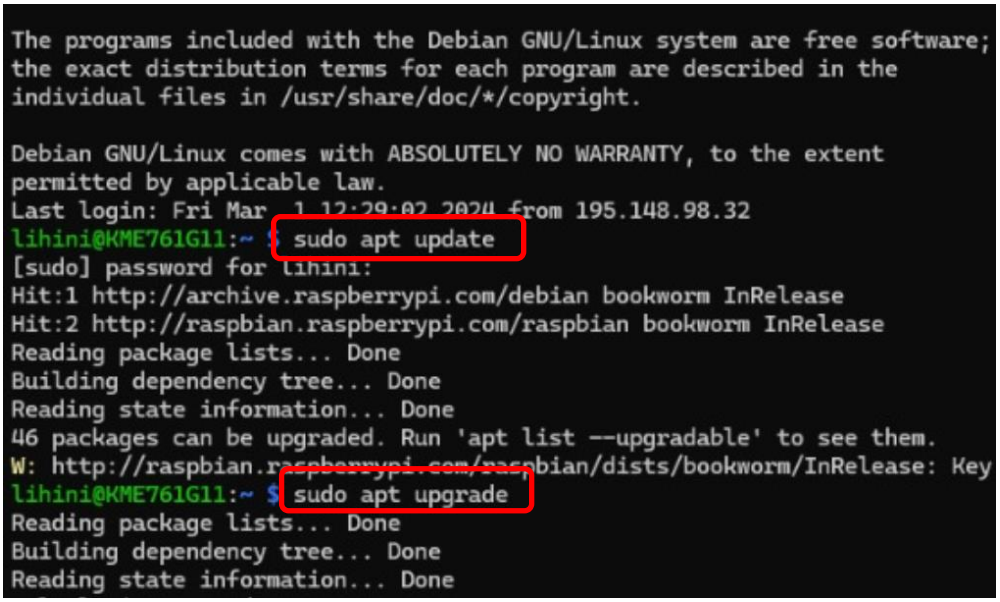
5 Setting up MQTT

Message Queuing Telemetry Transport (MQTT) service is set up on the Raspberry Pi. The Raspberry Pi will be used as an MQTT broker to deliver messages between the Raspberry Pi Pico and computer. But before installing new packages, it is needed to make sure that the Raspberry Pi operating system is up to date.

At first, Raspberry Pi is accessed through an SSH connection and then the following tasks are completed.

1. Update and upgrade the Raspberry Pi operating system.

The Raspberry Pi Pico is updated by all members using the same command as previously done. ***sudo apt update*** is the updating command and is upgraded using the command ***sudo apt upgrade***.



2. Installation Mosquitto for MQTT

Mosquitto server is installed and client application packages on the Raspberry Pi can be observed. The following commands are used for that purpose.

sudo apt-get install mosquitto mosquitto-clients.

```
lihini@KME761G11: ~
lihini@KME761G11:~ $ sudo apt-get install mosquitto mosquitto-clients
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libcamera0.1
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libcjson1 libdlt2 libmosquitto1
The following NEW packages will be installed:
  libcjson1 libdlt2 libmosquitto1 mosquitto mosquitto-clients
0 upgraded, 5 newly installed, 0 to remove and 1 not upgraded.
Need to get 597 kB of archives.
After this operation, 1,844 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://raspbian.mirror.vu.lt/raspbian/raspbian bookworm/main armhf libcjson1
```

3. Make sure that the Mosquito service is running:

As a next step it should be checked that installed Mosquitto service is running or not. For testing the service, the command of ***sudo systemctl status mosquitto*** is used.

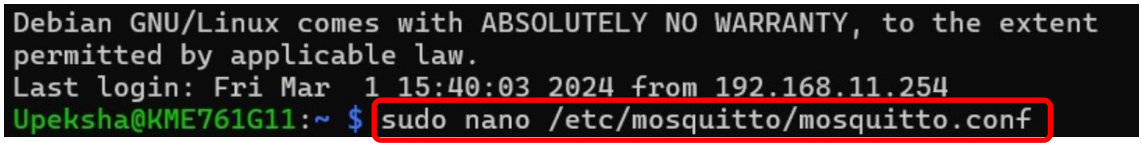
```
lihini@KME761G11:~ $ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
     Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; preset: enabled)
     Active: active (running) since Fri 2024-03-01 14:52:52 EET; 1min 19s ago
       Docs: man:mosquitto.conf(5)
             man:mosquitto(8)
    Process: 1205 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto (code=exited, s
    Process: 1206 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto (code=exited, s
    Process: 1207 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited, statu
    Process: 1208 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited, statu
   Main PID: 1209 (mosquitto)
      Tasks: 1 (limit: 3920)
        CPU: 68ms
     CGroup: /system.slice/mosquitto.service
             └─1209 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

4. Make MQTT broker start automatically at reboot. The command is ***sudo systemctl enable mosquitto.***

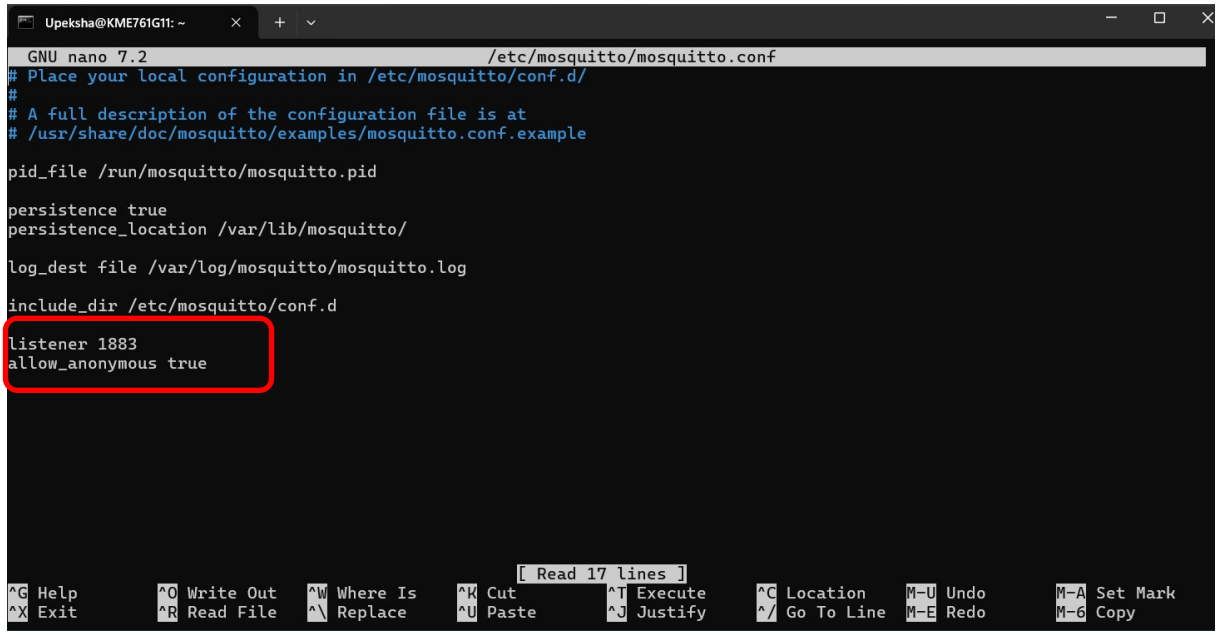
```
lihini@KME761G11:~ $ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /lib/systemd/systemd-
Executing: /lib/systemd/systemd-sysv-install enable mosquitto
lihini@KME761G11:~ $
```

Then it tried to publish a message using pub and sub commands, but it was not successful. Then as default, we tried to change the mosquitto settings to be able to receive messages from other devices such as the Pico board, because mosquitto only allows local messaging.

5. Mosquitto settings are changed to be able to receive messages from other devices. We used ***sudo nano /etc/mosquitto/mosquitto.conf*** command for the purpose of it.

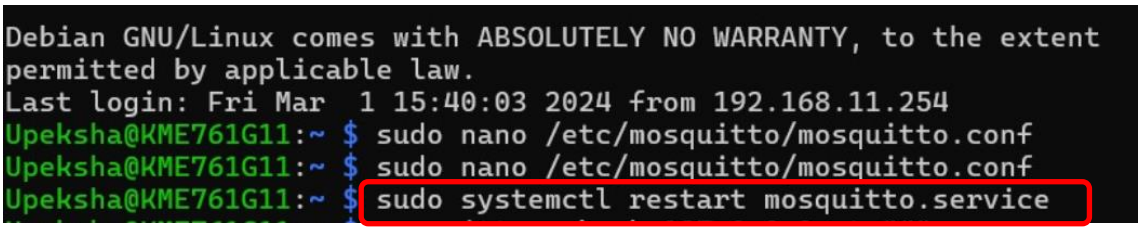


6. Modify the file by adding ***listener 1883, allow_anonymous true*** lines at the end and finished file as follows:



For exit the nano editor it can be used the key combination ctrl + x. Nano asks to save user's modifications before exit.

7. Then restart the MQTT by using command,
   sudo systemctl restart mosquitto. service



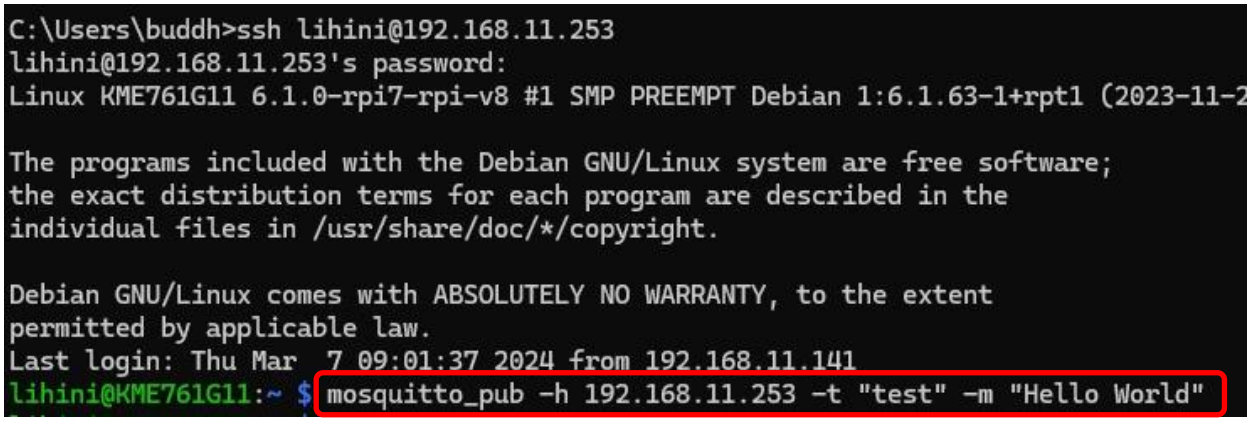
8. pub and sub commands are used to publish and subscribe to a message.

On one window, subscribe to a test topic as follows. All group members published and subscribed to each other using their own computers.

Publishing to topic “test” using,

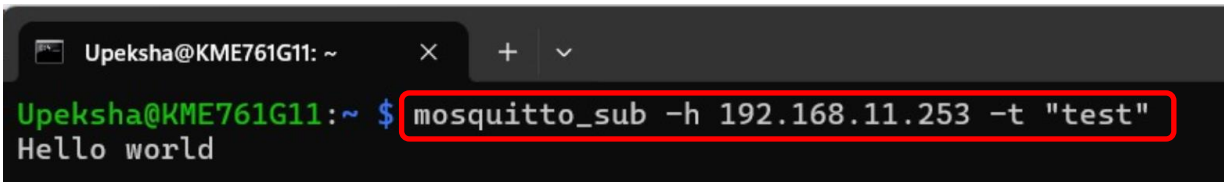
mosquitto_pub -h 192.168.11.253 -t “test” -m “Hello world” command.

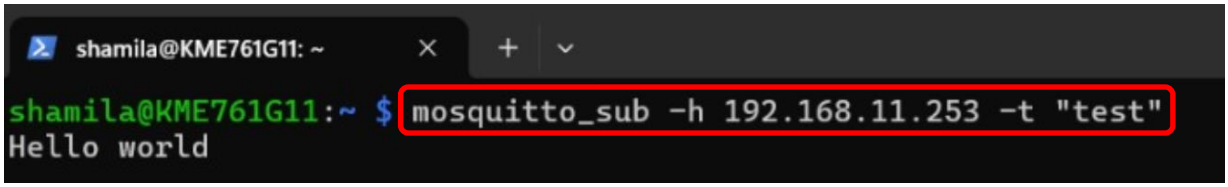
User 1:



Subscribe a topic “test” using ***mosquitto_sub -h 192.168.11.253 -t “test”.***

User 2 & User 3:

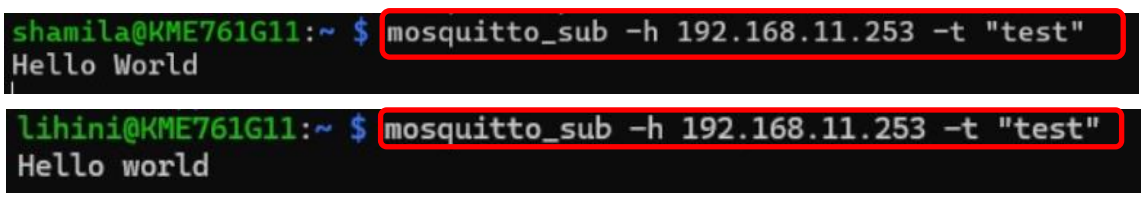




We did this by three of the users.

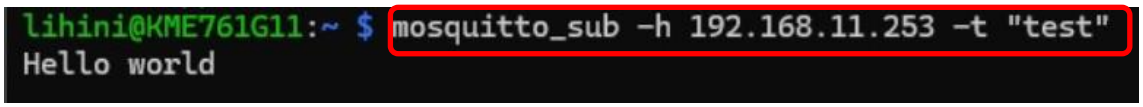
User 2:

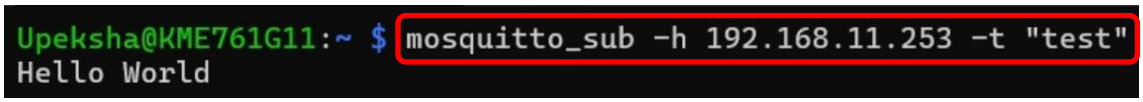
User 1 & User 3:



User 3:

User 1 & User 2:

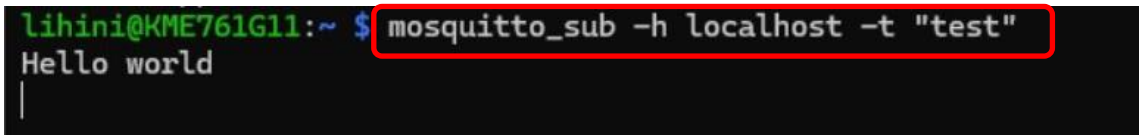


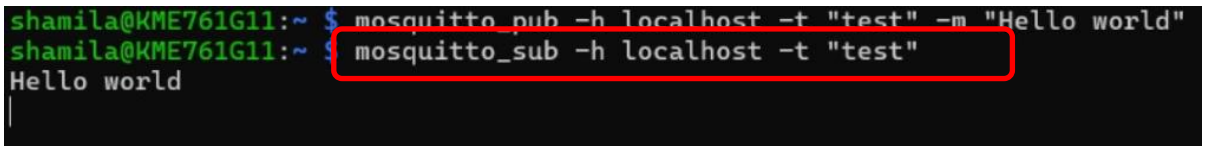


9. Two command line windows on the Raspberry Pi (over SSH) are used for testing MQTT locally. It can use the same computer or different ones for that purpose.

Subscribing to topic “test” using ***mosquitto_sub -h localhost -t “test***.

User 1 & User 2:

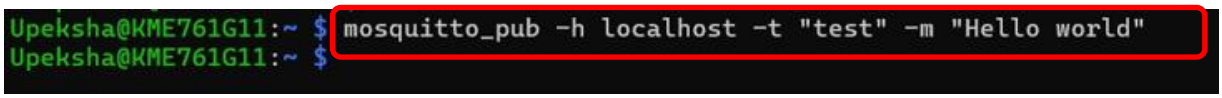




Publishing by another user a topic “test” using,

mosquitto_pub -h localhost -t “test” -m “Hello world”.

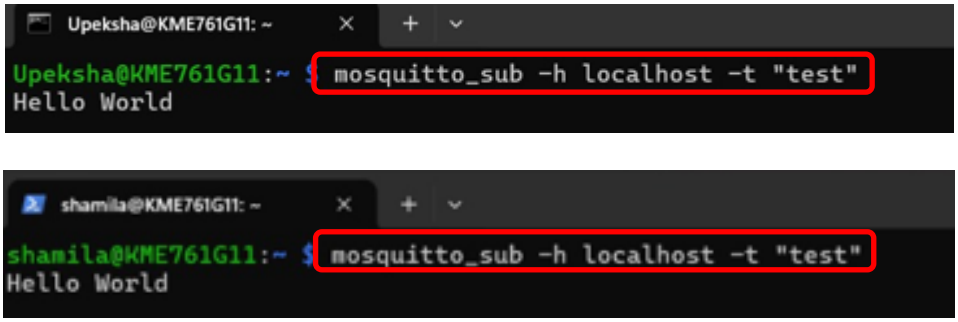
User3:



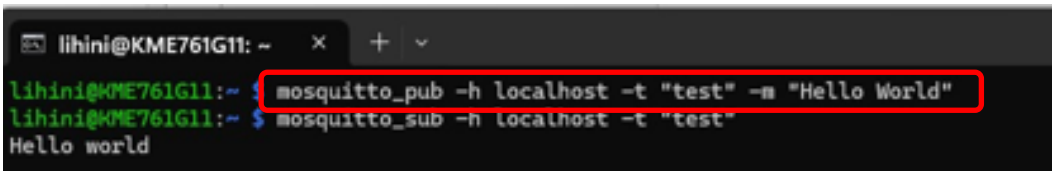
It showed the text “Hello world” printed on the subscribe windows.

We did this by three of the users.

User 2 & User 3:

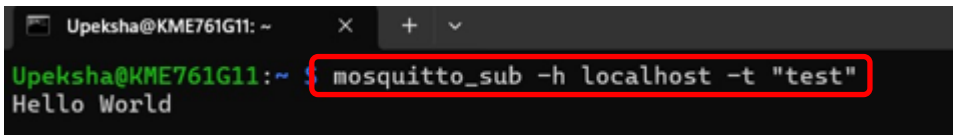


User1:



User 1 & User 3:

c



User 2:

6 Setting up the Raspberry Pi Pico

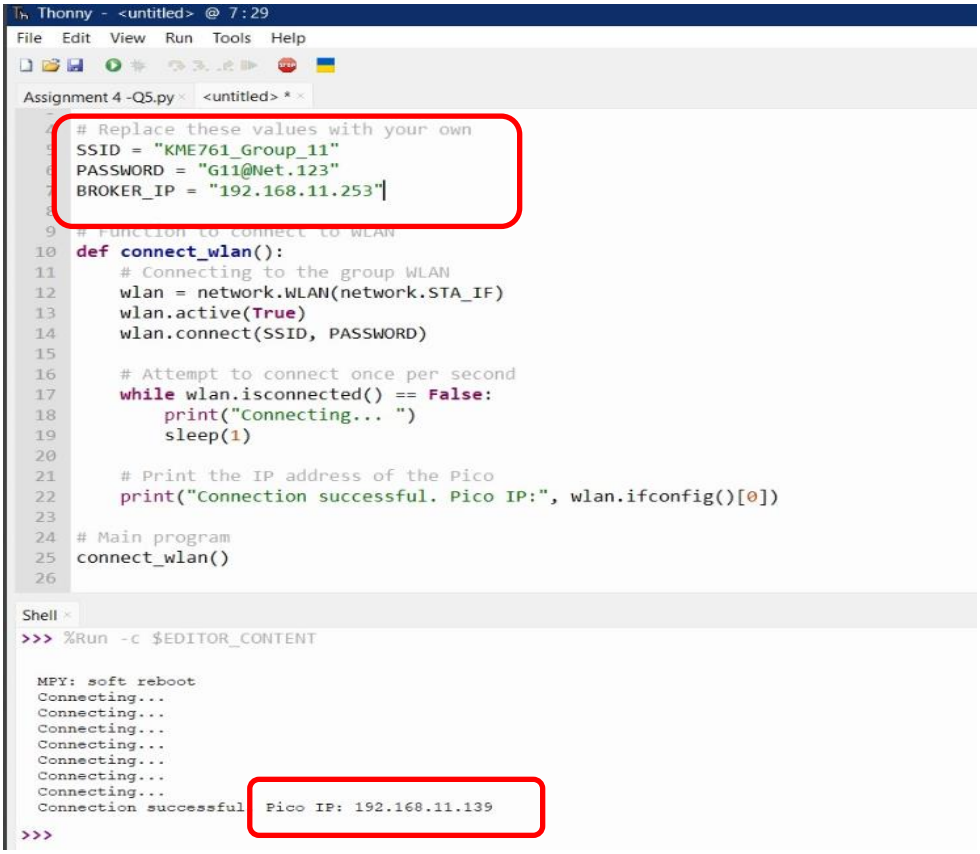
In this phase, we've completed our network configuration and configured the Raspberry Pi Pico to transmit MQTT messages. Before proceeding, ensure that the Raspberry Pi Pico W has the latest firmware installed. Additionally, Thonny IDE is required to install the MQTT module and develop your MicroPython code.

Transfer the example programs to Thonny. Before utilizing any of these example programs, update the default values of the SSID, PASSWORD and BROKER_IP located at the top of the program:

Run the example programs in the following order:

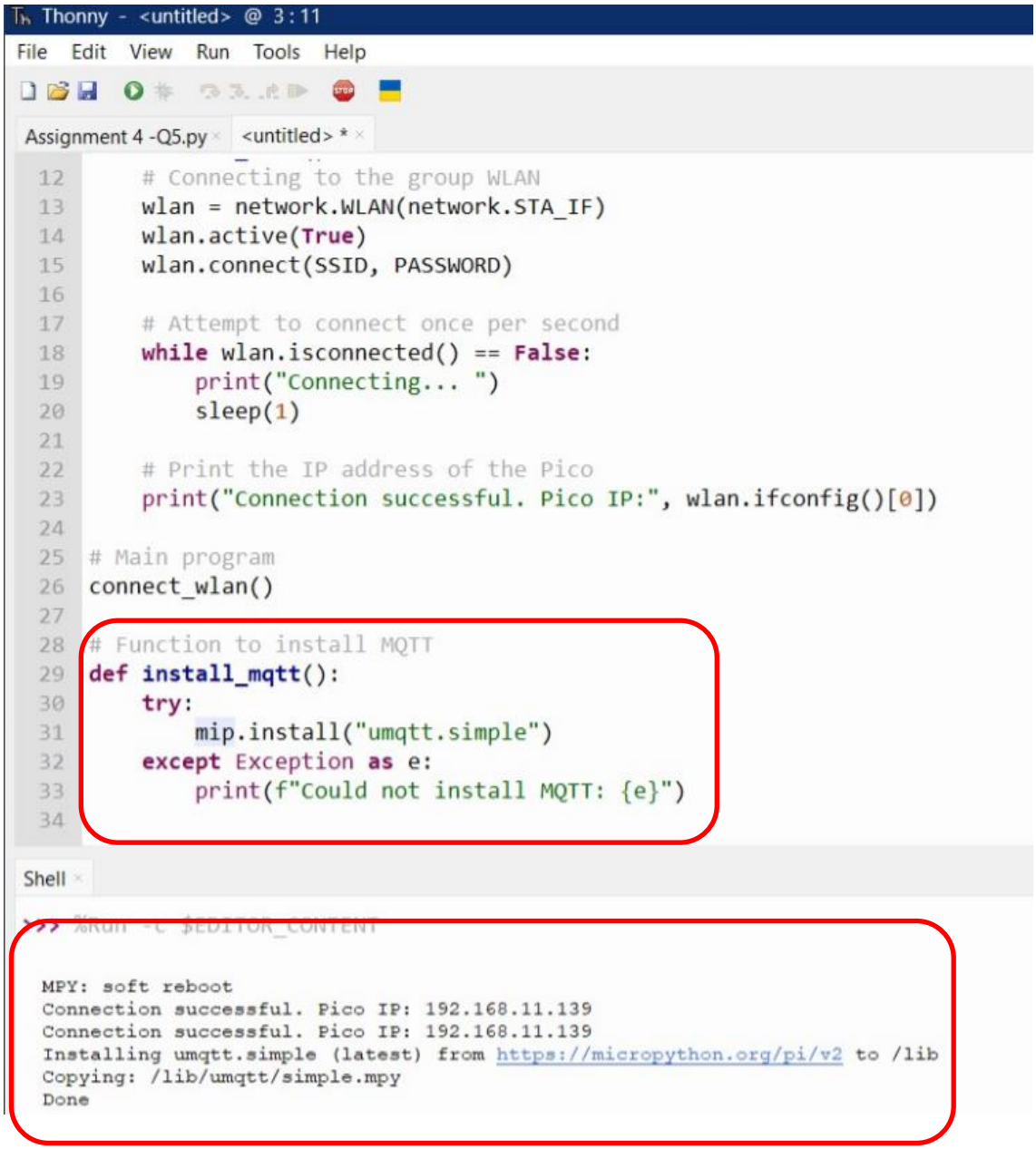
1.***connect_to_wlan.py:*** This script facilitates the connection of the Raspberry Pi Pico to the group WLAN. Upon successful connection, the IP address of the Pico board will be displayed.

Captured a screenshot as follows:



2.***Install_mqtt.py***: This script connects the Pico board to the group WLAN and endeavors to install an MQTT library on the Pico board. Upon successful installation, the following output should be visible in Thonny:

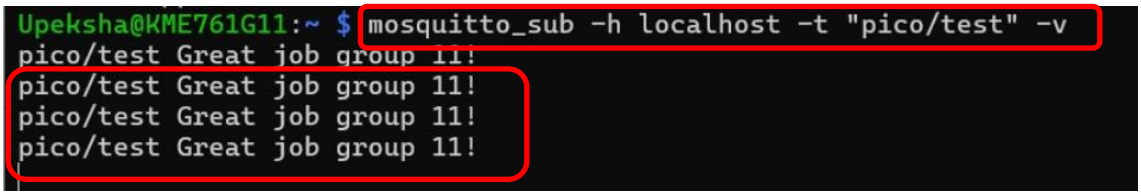
Capture a screenshot as follows:

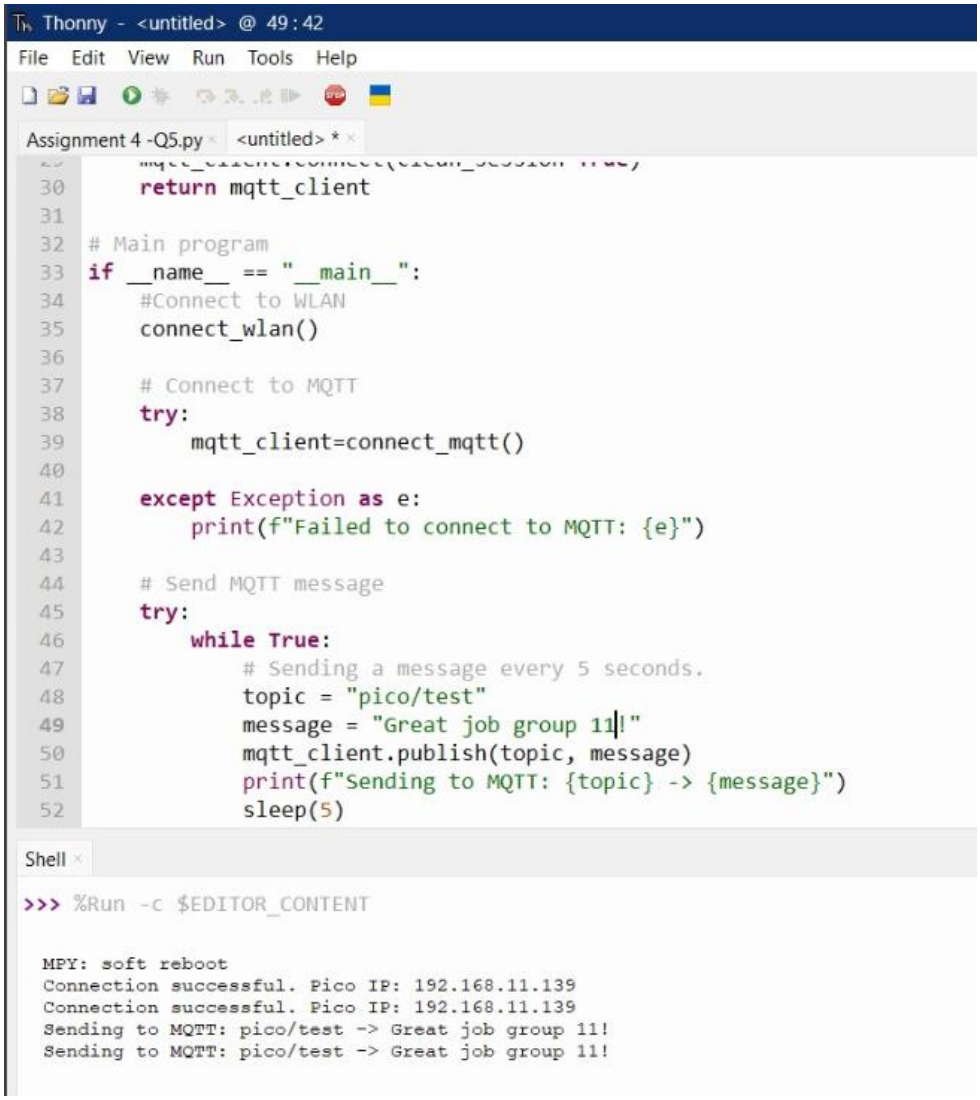


3.***mqtt_publish_test.py***: This test script establishes a connection between Pico W and the group WLAN. It then initiates an MQTT connection and dispatches MQTT messages to the "pico/test" topic every five seconds. To subscribe to these messages, an SSH connection must be opened to the Raspberry Pi, and the following command should be executed:

mosquitto_sub -h localhost -t "pico/test"

Upon successful execution, the output in the Raspberry Pi terminal should resemble the following:





7 Some additional services to network project

❖ Implementation of HTTPS Support with Let's Encrypt Certificate

We have successfully enhanced the security of our web server by adding HTTPS support through a free SSL certificate provided by letsencrypt.org. Below are the steps we took to accomplish this task:

- ✓ Update and upgrade the Raspberry Pi operating system.

The Raspberry Pi Pico is updated by all members using the same code as previously done. ***sudo apt update*** is the updating command and is upgraded using the command ***sudo apt upgrade.***

```
Upeksha@KME761G11:~ $ sudo apt update
Hit:1 http://archive.raspberrypi.com/debian bookworm InRelease
Hit:2 http://raspbian.raspberrypi.com/raspbian bookworm InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
W: http://raspbian.raspberrypi.com/raspbian/dists/bookworm/InRelease: Key is
N section in apt-key(8) for details.
Upeksha@KME761G11:~ $ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer require
  libcamera0.1 linux-headers-6.1.0-rpi8-common-rpi linux-headers-6.1.0-rpi8-r
  linux-headers-6.1.0-rpi8-rpi-v7l linux-image-6.1.0-rpi8-rpi-v6 linux-image-
  linux-image-6.1.0-rpi8-rpi-v7l
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
  linux-image-rpi-v8:arm64
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

- ✓ Certbot Installation:

We installed Certbot, a tool designed to automate the process of obtaining and renewing SSL certificates. Used ***sudo apt install python3-certbot-apache*** command and ***sudo apt install certbot*** command.

```
Upeksha@KME761G11:~ $ sudo apt install python3-certbot-apache
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libcamera0.1
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  augeas-lenses certbot libaugeas0 python3-acme python3-augeas python3-certbot
  python3-configargparse python3-configobj python3-cryptography python3-icu py
  python3-parsedatetime python3-rfc3339 python3-tz
Suggested packages:
```

```
Setting up python3-certbot-apache (2.1.0-2) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for libc-bin (2.36-9+rpt2+deb12u4) ..
Upeksha@KME761G11:~ $ sudo apt install certbot
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
certbot is already the newest version (2.1.0-4).
certbot set to manually installed.
```

- ✓ SSL Certificate Request:

Utilizing Certbot, we requested an SSL certificate for our domain from Let's Encrypt. Before this we need to make sure that port 80 and port 443 are port forwarded. Used ***sudo certbot --apache*** command here.

Here we needed to give our email address and domain name. Mistakenly we missed the domain name and again we entered it. Screenshots are as follows.

```
  libcamera0.1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
Upeksha@KME761G11:~ $ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): slusanjeewani@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2022.pdf.
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.3-September-21-2022.pdf. You must
agree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Y
Account registered.
Please enter the domain name(s) you would like on your certificate (comma and/or
space separated) (Enter 'c' to cancel): c
Please specify --domains, or --installer that will help in domain names autodiscover
Ask for help or search for solutions at https://community.letsencrypt.org. See the l
r more details.
```

```
Upeksha@KME761G11:~ $ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Please enter the domain name(s) you would like on your certificate (comma and
space separated) (Enter 'c' to cancel): kme761g11.asuscomm.com
Requesting a certificate for kme761g11.asuscomm.com

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/kme761g11.asuscomm.com/fullcha
Key is saved at:         /etc/letsencrypt/live/kme761g11.asuscomm.com/privkey
This certificate expires on 2024-06-05.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate i

Deploying certificate
Successfully deployed certificate for kme761g11.asuscomm.com to /etc/apache2/
Congratulations! You have successfully enabled HTTPS on https://kme761g11.asu

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Upeksha@KME761G11:~ $
```

- ✓ Manage our website's details and verify HTTPS functionality:

/var/www/html is a common directory where web content is stored on many Linux distributions, particularly those using Apache as the web server.

```
Upeksha@KME761G11:~ $ cd /var/www/html
Upeksha@KME761G11:/var/www/html $ ls -l
total 12
-rw-r--r-- 1 root root 10701 Mar  1 12:43 index.html
Upeksha@KME761G11:/var/www/html $ sudo nano index.html
```

After configuring SSL with Certbot, navigated to this directory to modify our website's details.

By using ***sudo nano index.html*** command we opened the file named index.html for editing using the Nano text editor with superuser (root) privileges.

By editing index.html, modified existing ones to improve the appearance and functionality of our website.

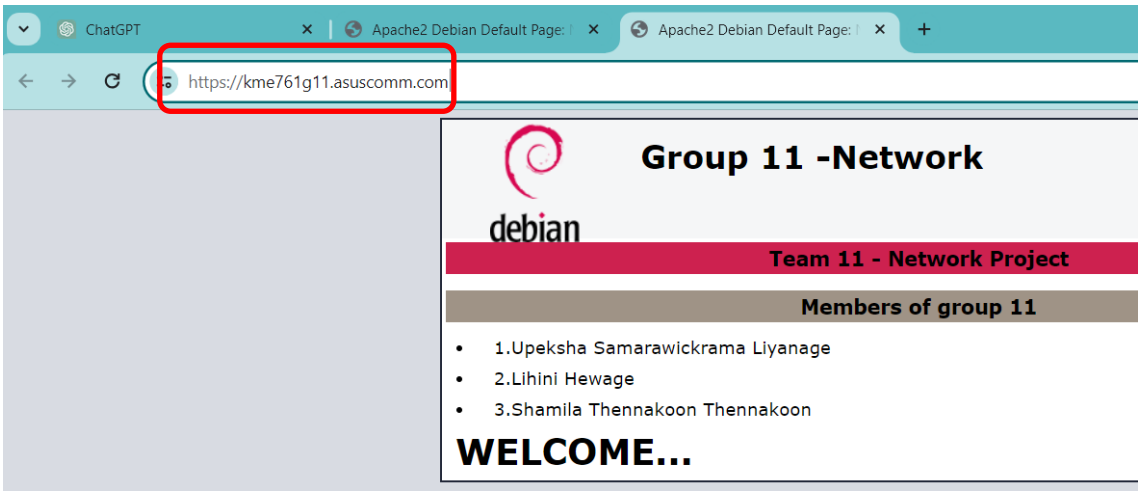
```
Upeksha@KME761G11: /var/w
 GNU nano 7.2                                        index.html *
  div.validator {
  }
    </style>
  </head>
  <body>
    <div class="main_page">
      <div class="page_header floating_element">
        <img src="/icons/openlogo-75.png" alt="Debian Logo" class="floating_element"/>
        <span class="floating_element">
          Group 11 - Network
                Lihini Hewage
                Upeksha Samarawickrama Liyanage
                Shamila Thennakoon
        </span>
      </div>
<!--      <div class="table_of_contents floating_element">
        <div class="section_header section_header_grey">
          TABLE OF CONTENTS
        </div>
        <div class="table_of_contents_item floating_element">
          <a href="#about">About</a>
        </div>
        <div class="table_of_contents_item floating_element">
          <a href="#changes">Changes</a>
        </div>
        <div class="table_of_contents_item floating_element">
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line
```

✓ SSL Implementation Confirmation:

After configuring SSL with Certbot, verified that our website is serving content correctly over HTTPS. Editing index.html allowed us to modify details that are specific to HTTPS to confirm that SSL is functioning as expected. We used our domain name kme761g11.asuscomm.com to get the confirmation.

Our group website is https://kme761g11.asuscomm.com

Relevant screenshot is as follows.



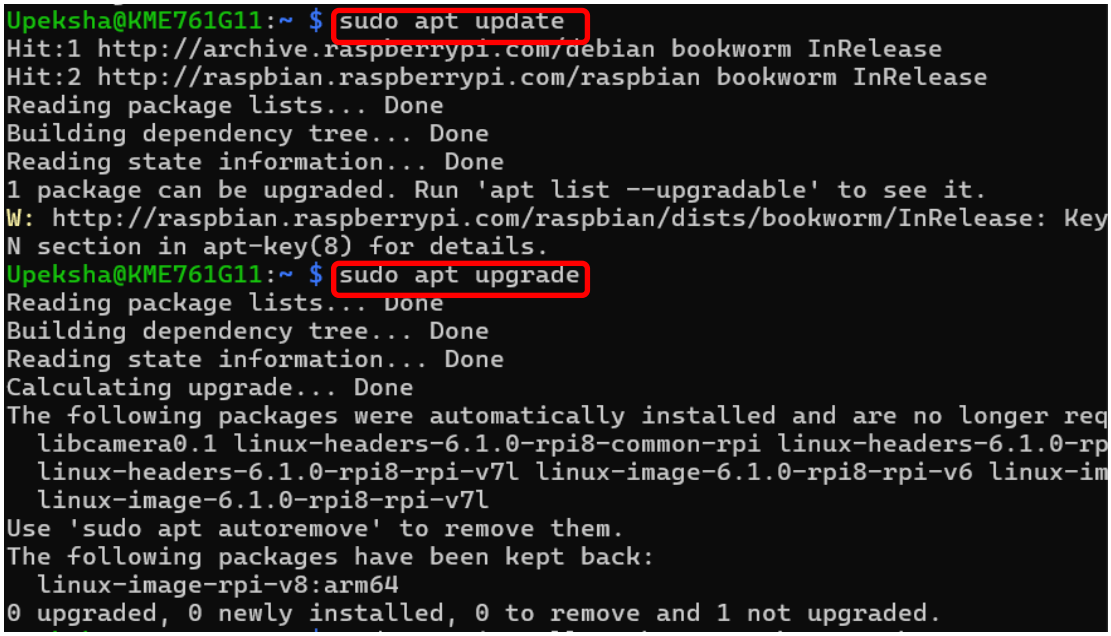
❖ **Create SSH Public/private keypair for all users and configure Raspberry Pi to only accept authentication by these SSH keys.**

Using SSH key-based authentication on the Raspberry Pi makes it safer because it switches out the risky password method with strong cryptographic keys. This change not only makes it harder for unauthorized users to get in but also makes logging in easier for users since they don't have to remember or type passwords.

Below are the steps we took to accomplish this task:

✓ Update and upgrade the Raspberry Pi operating system.

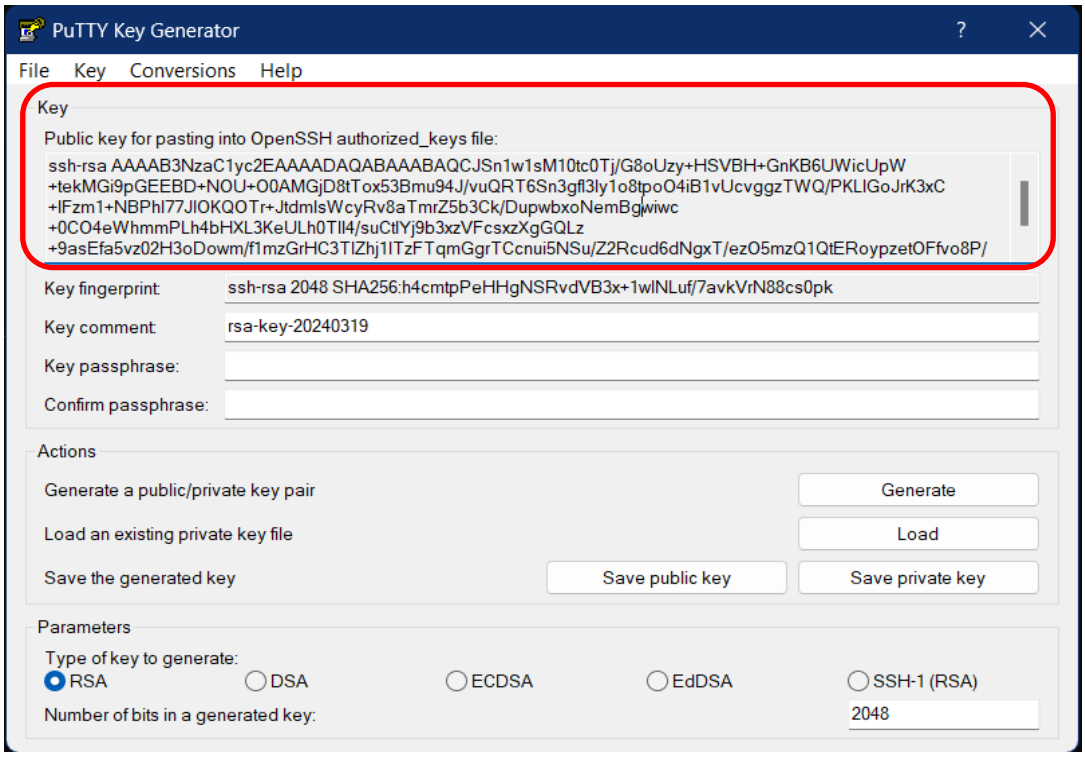
The Raspberry Pi Pico is updated by all members using the same code as previously done. ***sudo apt update*** is the updating command and is upgraded using the command ***sudo apt upgrade.***



✓ Generate a New SSH Key Pair:

Firstly, downloaded ***PuTTYgen*** and installed it on computer.

In the ***PuTTYgen*** window, clicked on the "Generate" button. Moved the mouse randomly in the blank area to generate randomness for key generation.
After the key pair was generated, we can see the public key and the key fingerprint displayed in the window. Screenshot is as follows.



✓ Save the Public and Private Keys:

Clicked on the "Save public key" button to save the public key to a file on the computer. Clicked on the "Save private key" button to save the private key to file on the computer. Also, make sure to end the files in **.ppk** so that PuTTY can pick them up.

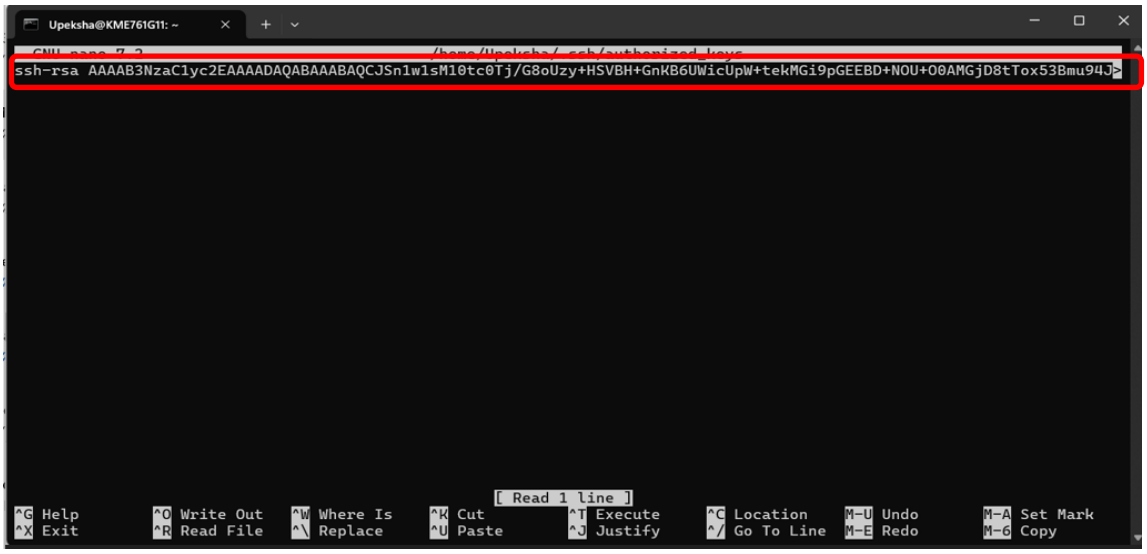
✓ Copying the Public Keys Manually:

1. Then we set up our ***authorized_keys*** file. This is the file that the SSH daemon will check when a private key is used for authentication. And used the following command with a few parameters to set the correct permissions and run it.
   install -d -m 700 ~/.ssh

2. Put our public key in the ***authorized_keys*** file. To do this run the following command.
   nano ~/.ssh/authorized_keys

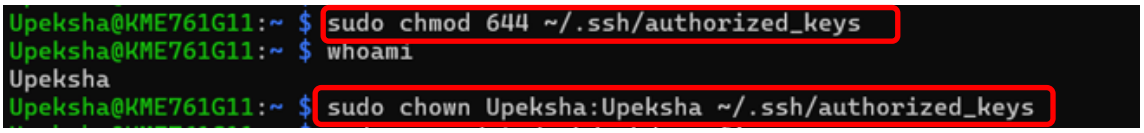
```
Upeksha@KME761G11:~ $ install -d -m 700 ~/.ssh
Upeksha@KME761G11:~ $ nano ~/.ssh/authorized_keys
```

3. Here we needed to copy and paste the contents of the public SSH key that, generated. Here we should save the public SSH key entered in the authorized_keys file and quit it.



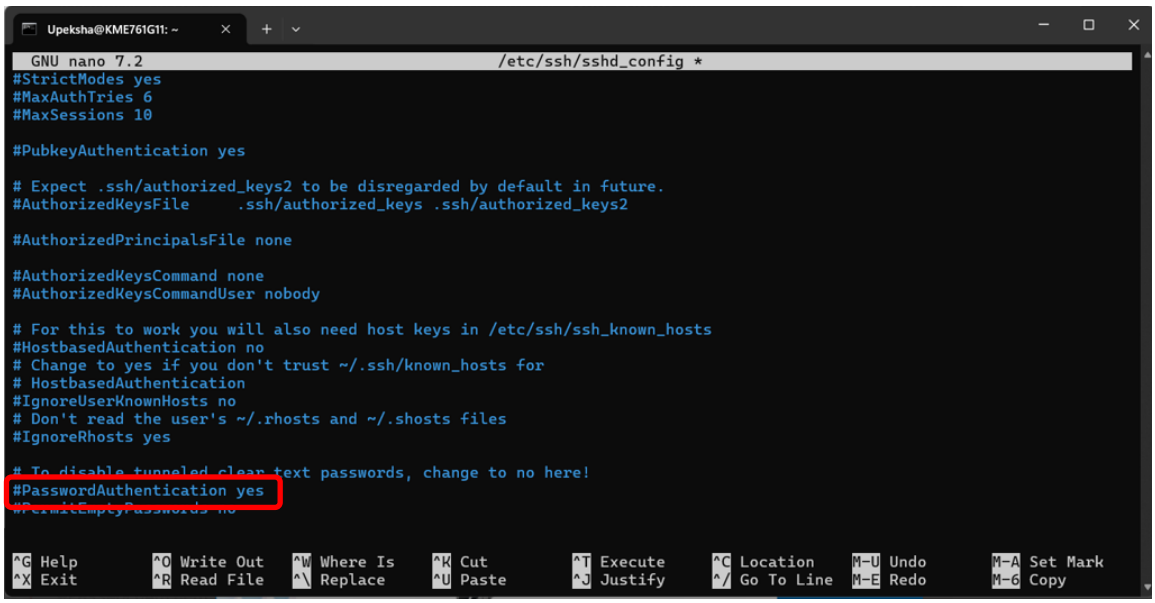
4. With the file now saved we need to make sure it has the correct permissions. To do this, we need to run the following ***chmod*** and ***chown*** commands. These commands will assign the correct permissions to the file. In the following command we want to use the name of the user.

   sudo chmod 644 ~/.ssh/authorized_keys
   sudo chown Upeksha: Upeksha ~/.ssh/authorized_keys

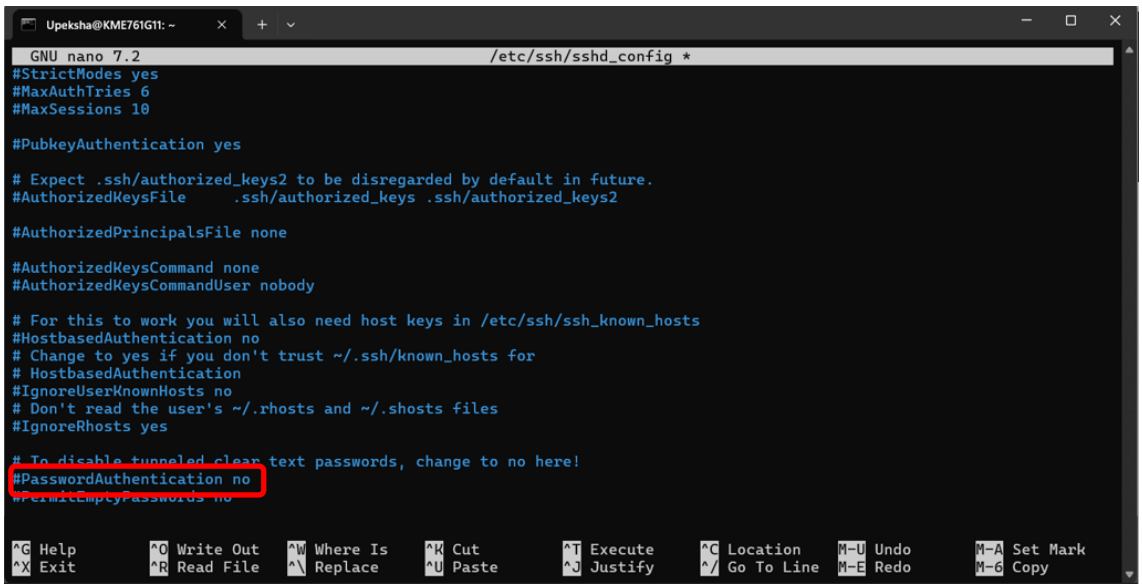


5. Then we need to disable password authentication, because we do not need to lock ourselves out of our Raspberry Pi. We need to modify the **sshd_config** file. To modify this file run the following command on your Raspberry Pi
   sudo nano /etc/ssh/sshd_config

In nano editor need to change password authentication. This simple change will completely disable the ability to login to your Raspberry Pi with just a password.
#PasswordAuthentication yes



#PasswordAuthentication no

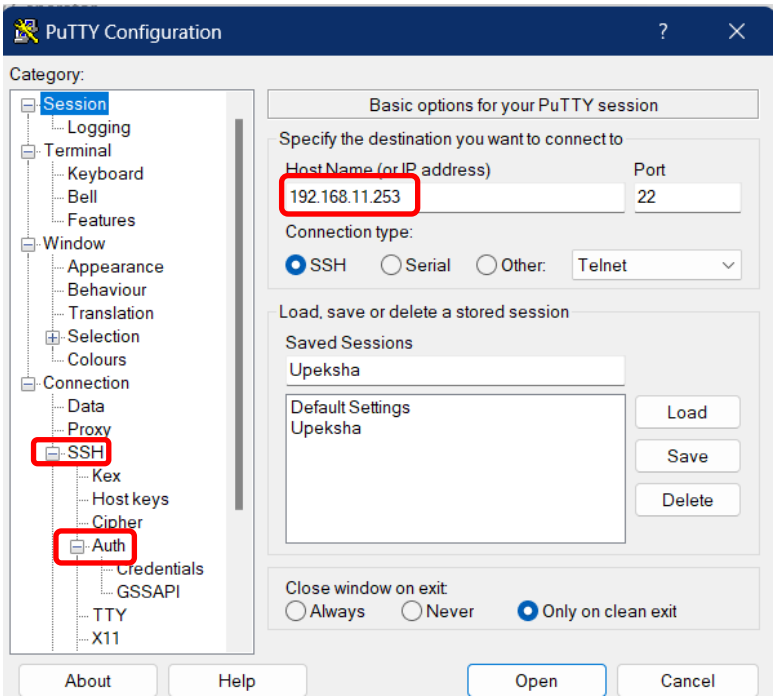


After modifying the password authentication as “no”, we should save those in the nano editor.

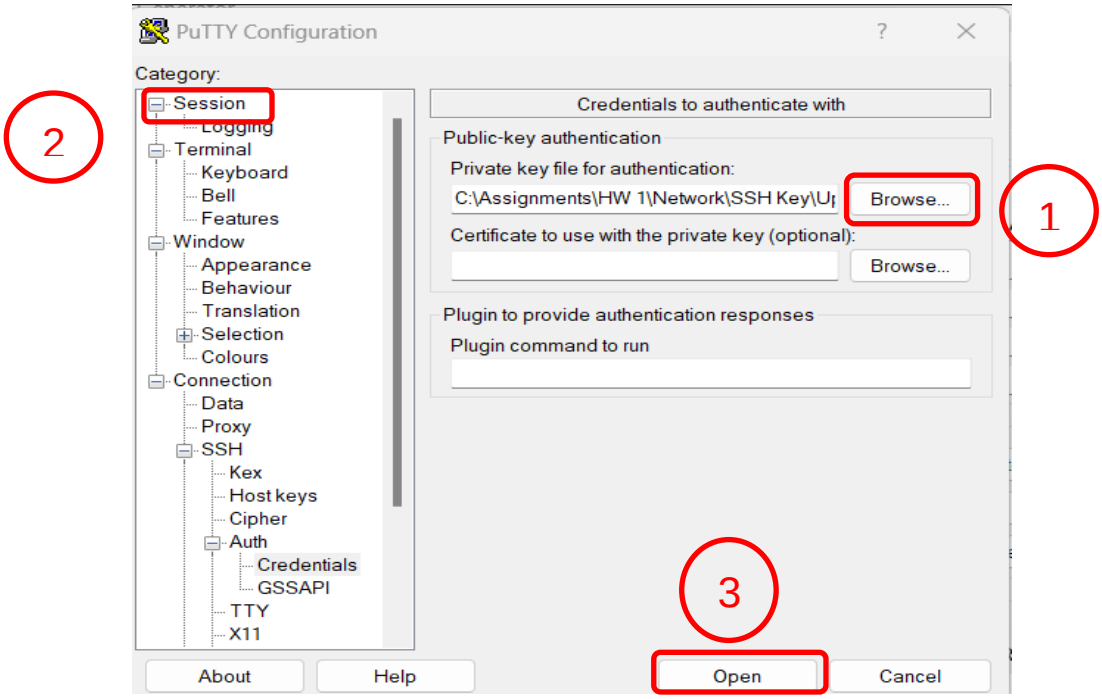
6. With the changes now made to the ***sshd_config*** file, we should restart our Raspberry Pi to ensure the changes are loaded in. The command is as follows.
   sudo reboot

✓ Connecting the Raspberry Pi using a Private Key with PuTTY:

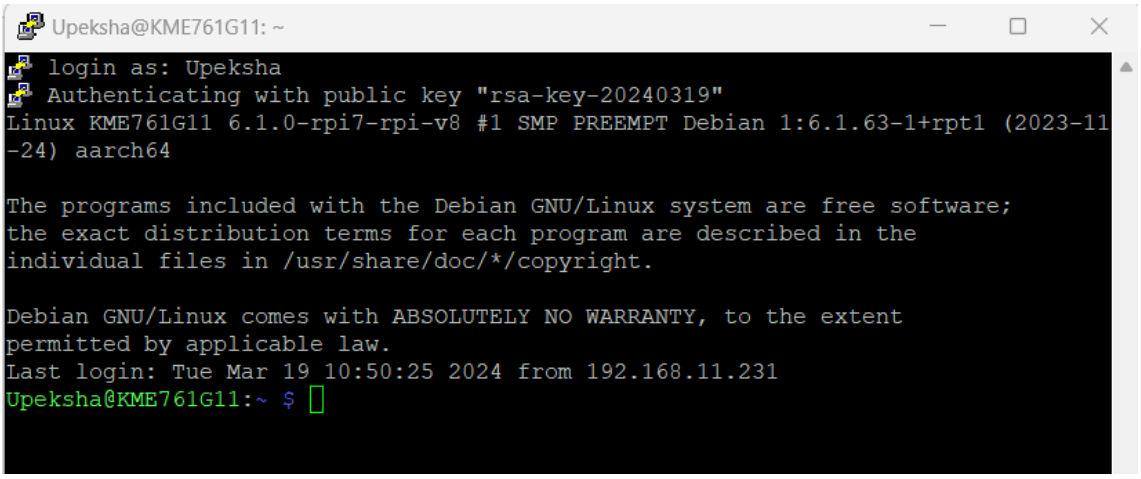
1. Started by opening PuTTY on the computer and first, entered the Raspberry Pi’s IP address and then clicked on “Auth” under the “SSH” section as the following screenshot.



2. Next, pressed the “Browse” button. This button will allow us to find and select the private key that we saved earlier. After selecting this file will allow PuTTY to try and use it for authentication. After selecting the private key from the browser, then click “Session” and pressed the “Open” button to start the connection.



3. Then we can login to raspberry pi using the correct username, but without any password.



After that, we followed those steps for two other users in our project group. All the steps were carried out as described above, and we were all able to log in without password authentication for the Raspberry Pi. The following screenshots show the commands we ran and the Putty login process.

User 2:

```
lihini@KME761G11:~ $ install -d -m 700 ~/.ssh
lihini@KME761G11:~ $ nano ~/.ssh/authorized_keys
lihini@KME761G11:~ $ sudo chmod 644 ~/.ssh/authorized_keys
[sudo] password for lihini:
lihini@KME761G11:~ $ whoami
lihini
lihini@KME761G11:~ $ sudo chown lihini:lihini ~/.ssh/authorized_keys
lihini@KME761G11:~ $ sudo nano /etc/ssh/sshd_config
lihini@KME761G11:~ $ sudo reboot

Broadcast message from root@KME761G11 on pts/3 (Tue 2024-03-19 11:20:42 EET):

The system will reboot now!
```

lihini@KME761G11: ~

```
login as: lihini
Authenticating with public key "rsa-key-20240319"
Linux KME761G11 6.1.0-rpi7-rpi-v8 #1 SMP PREEMPT Debian 1:6.1.63-1+rpt1 (2023-11
-24) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Mar 19 12:26:14 2024 from 192.168.11.141
lihini@KME761G11:~ $
```

User 3:

```
shamila@KME761G11:~ $ install -d -m 700 ~/.ssh
shamila@KME761G11:~ $ nano ~/.ssh/authorized_keys
shamila@KME761G11:~ $ sudo chmod 644 ~/.ssh/authorized_keys
shamila@KME761G11:~ $ whoami
shamila
shamila@KME761G11:~ $ sudo chown shamila:shamila ~/.ssh/authorized_keys
shamila@KME761G11:~ $ sudo nano /etc/ssh/sshd_config
shamila@KME761G11:~ $ sudo reboot

Broadcast message from root@KME761G11 on pts/1 (Wed 2024-03-20 11:37:43 EET)

The system will reboot now!
```

shamila@KME761G11: ~

```
login as: shamila
Authenticating with public key "rsa-key-20240320"
Linux KME761G11 6.1.0-rpi7-rpi-v8 #1 SMP PREEMPT Debian 1:6.1.63-1+rpt1 (2023-11
-24) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 20 11:28:57 2024 from 192.168.11.186
shamila@KME761G11:~ $
```

8 References

- Weekly Exercises
  - Week 1: Setting up the Network.
  - Week 2: Setting up the Raspberry Pi
  - Week 3: Raspberry Pi Configuration
  - Week 4: Setting up MQTT
  - Week 5: Setting up the Raspberry Pi Pico

- Networks Project spring 2024.pdf

- Final Assignment Report Template

- https://pimylifeup.com/raspberry-pi-ssh-keys/

- https://pimylifeup.com/raspberry-pi-ssl-lets-encrypt/

- https://www.raspberrypi.com/software/

- https://gitlab.metropolia.fi/saanapi/hardware-1-networks-final-assignment