Project Name: **RCC (Riverside City College) Cybersecurity Club's Raspberry Pi Networking Project**

Date completed: **July 25, 2023**

Created and performed by: **Jason Patrick Salerno**

Purpose: **Documentation for creating a DHCP & DNS server out of a Raspberry Pi**

Part 1: Assembling The Hardware

1. The first step we need to take is assembling the hardware.

2. The first step I take is attaching the ribbon cable to the screens circuit board.

3. Next, I placed the raspberry pi on top of the circuit board and screwed it down to fit in place.

4. Here we have our 3-aluminum heat sink which we will place on our raspberry pi.

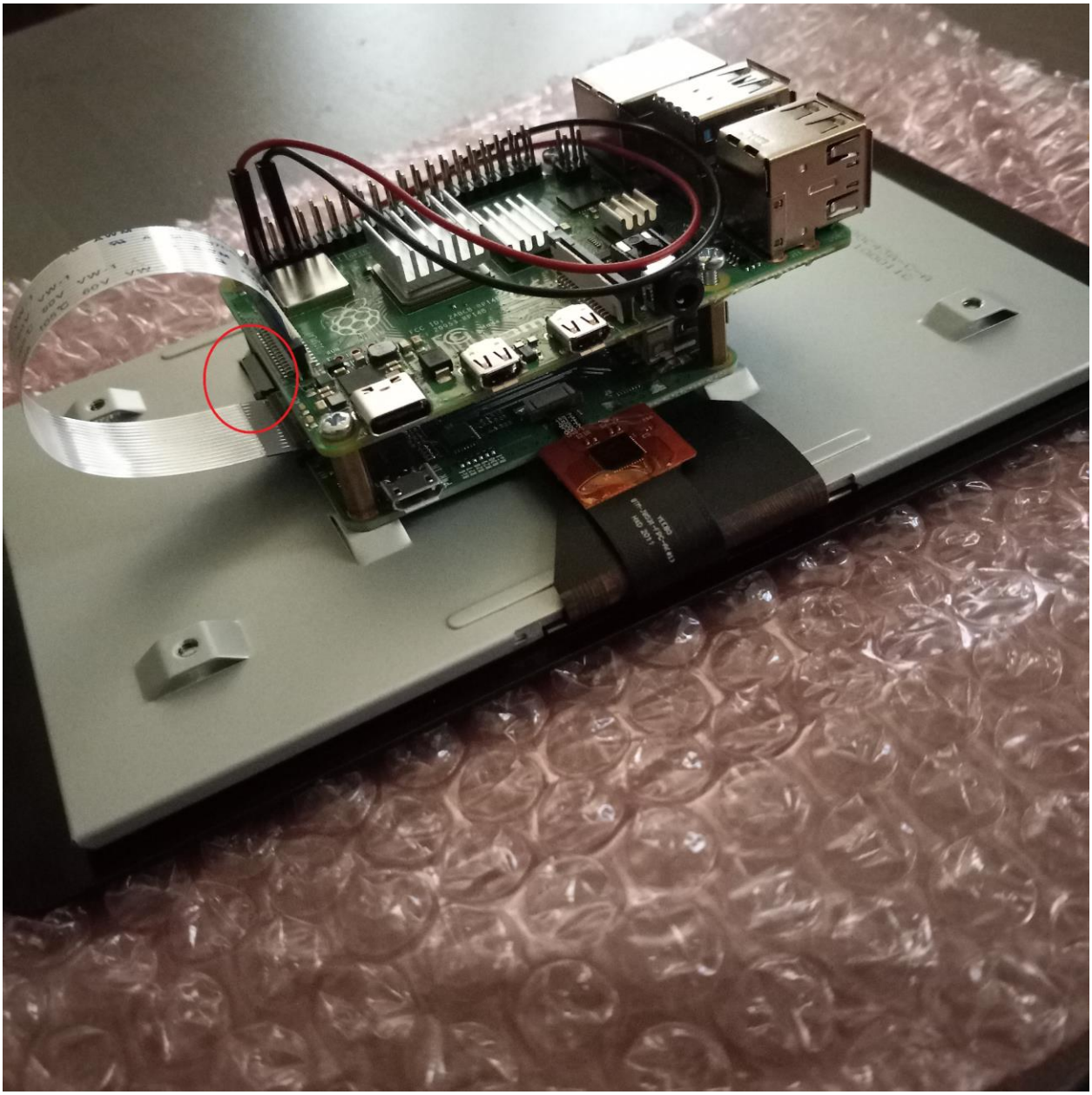
5. I have now placed our 3-aluminum heat sink for our raspberry pi.

6. I have connected the ribbon cable on our raspberry pi and have gently locked it in place.

7. The next task I did was gently placing the red cable in **5V** and placing the black cable in **GND** on the circuit board of the screen then placing those cables on the raspberry pi.

8. Here we have our 32GB SD card, that already has been loaded with NOOBS installation.

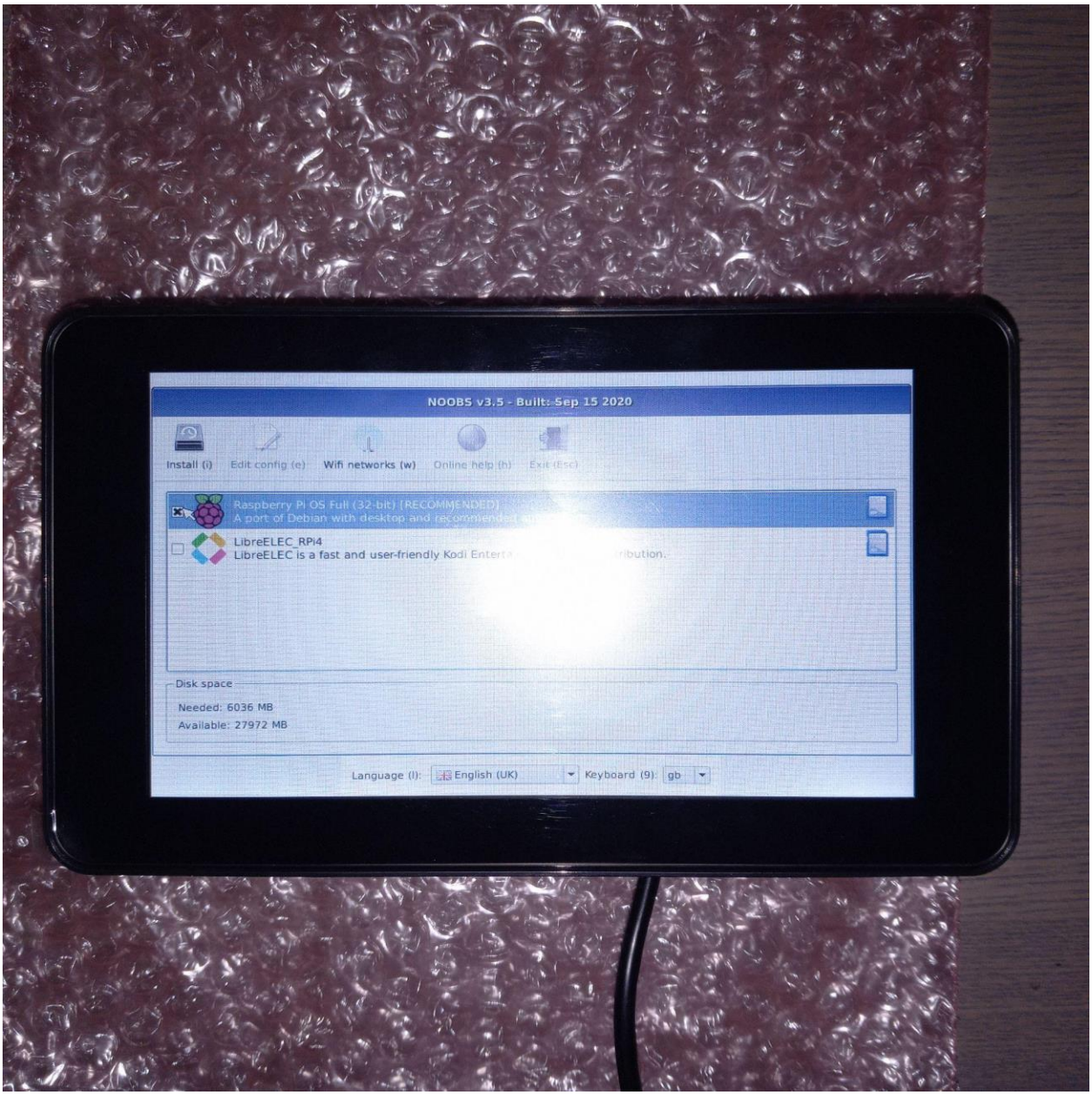
9. The next task I did was simply placing the 32GB sd card into the raspberry pi sd card slot.



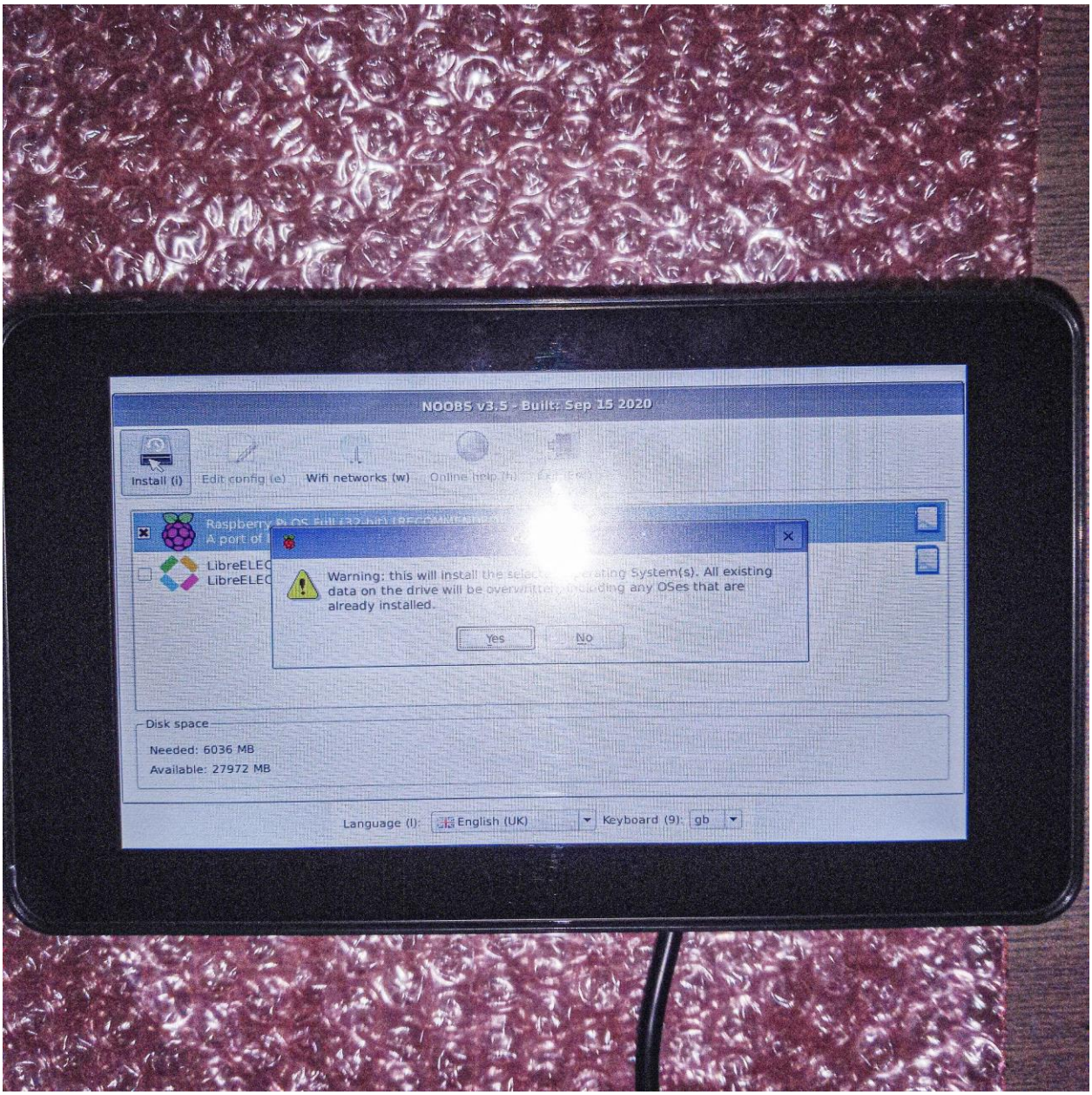
10. Here we have our case ready and its 4 screws

11. This is the last part of assembling the hardware, which is placing the 4 screws into our raspberry pi.

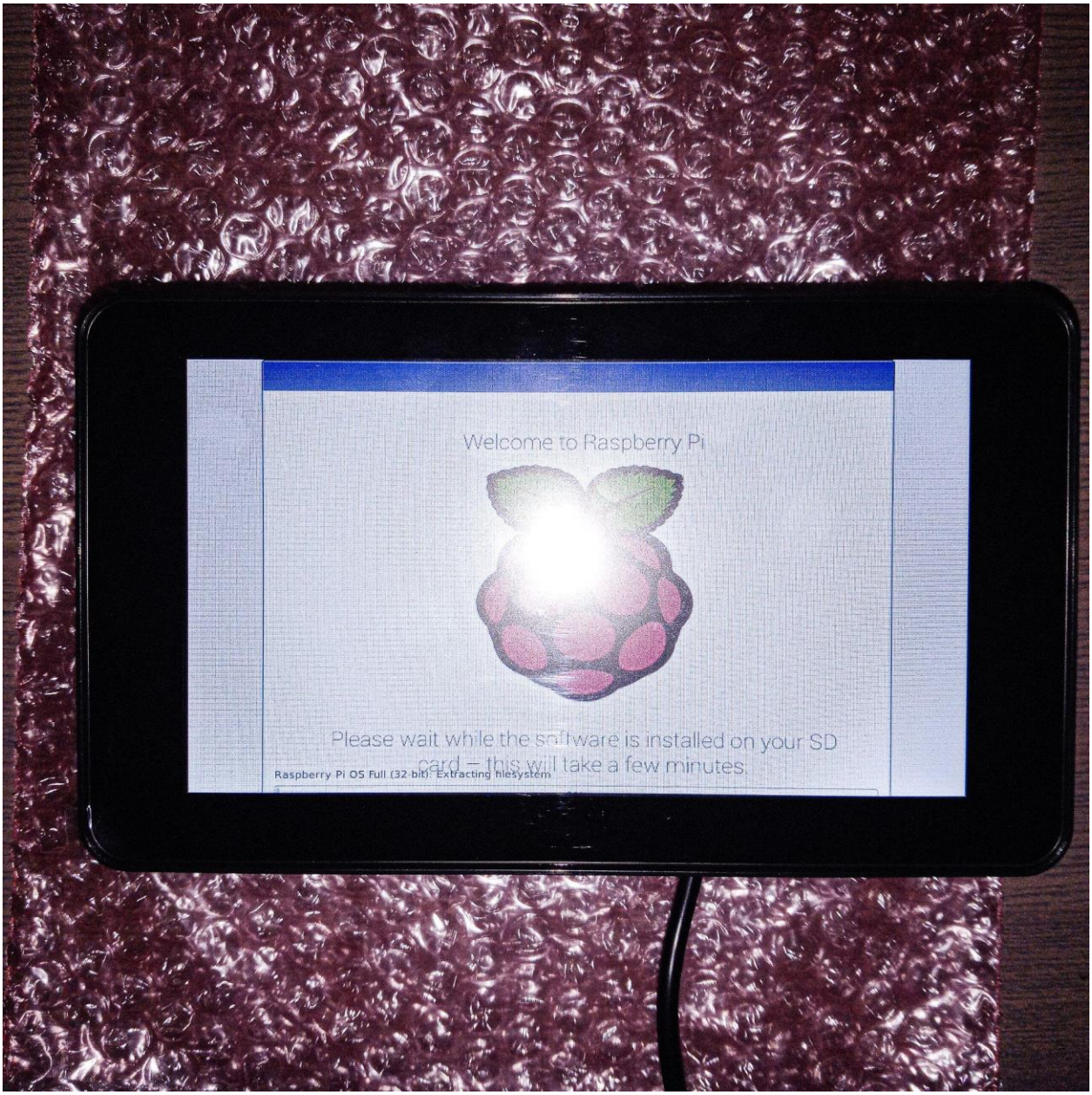
12. Here we are at the NOOBS installation process, where we can select one OS out of the following: Raspbian OS FULL (32bit) or LibreELEC_RPI4 So, I will select **Raspbian OS FULL (32bit)**.



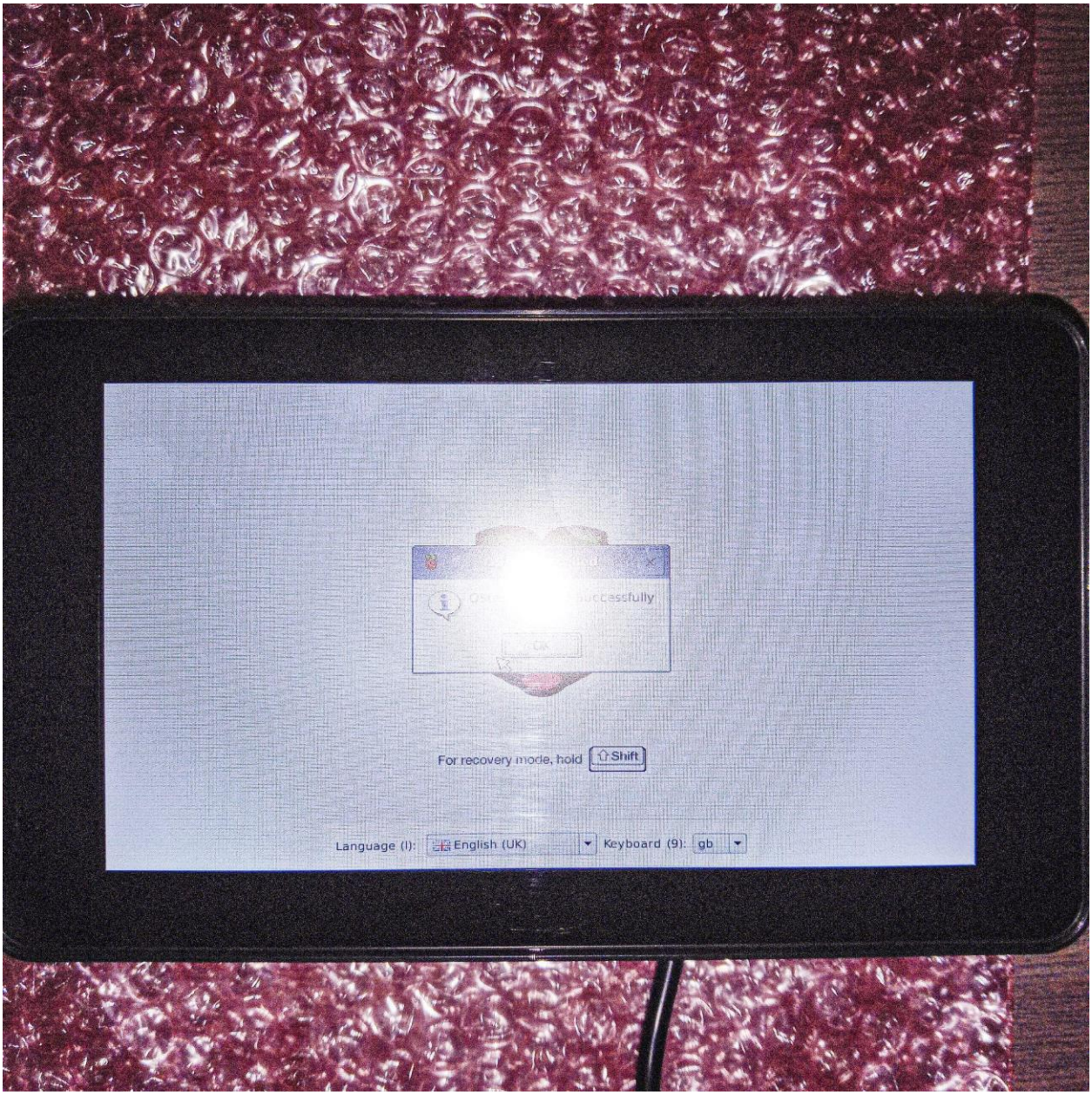
13. Here I am prompted with a Warning which displays “Warning: This will install the selected Operating System(s). All existing data on the drive will be overwritten including any OSes that are already installed”. I click yes



14. The raspbian OS installation begins.

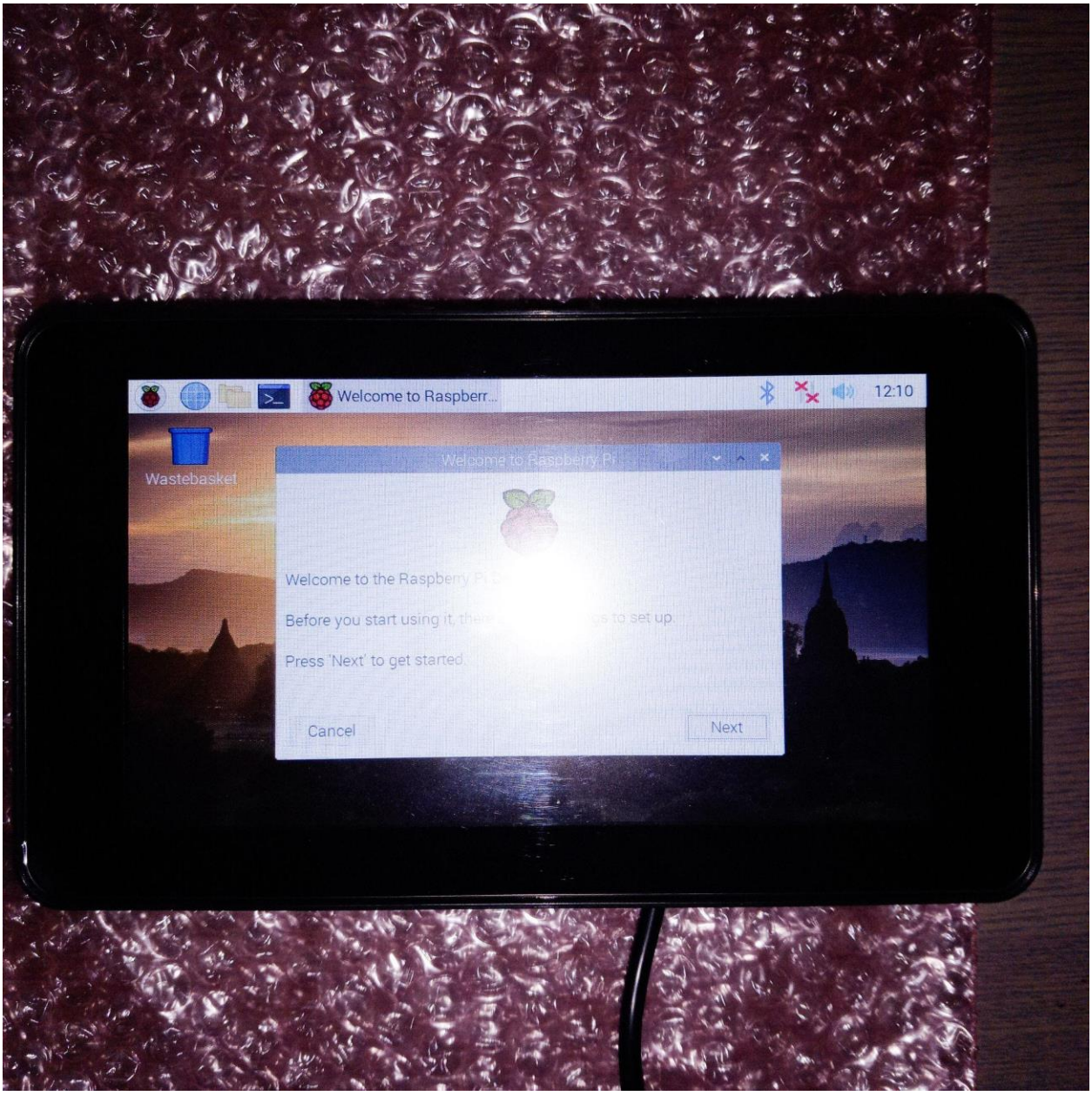


15. The raspbian installation process is now finished.



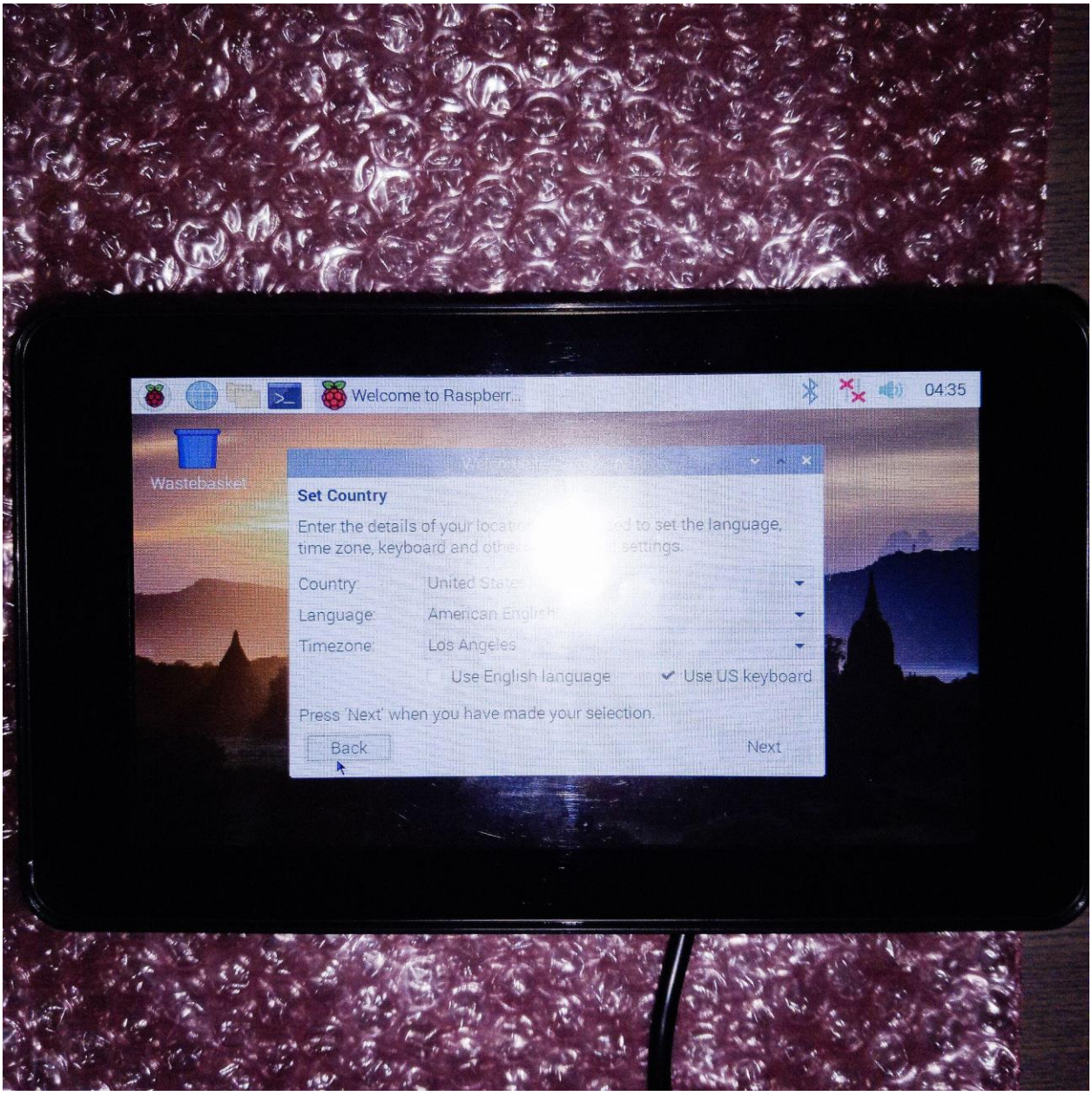
16. Here we are at the raspbian welcome screen, I click next.

    Note: I had trouble with taking pictures for this project due to the low brightness of my phone's camera, so I turned on the flash.



17. On the next display, it wants us to change/create a password for the user. **Note**: The password for this raspberry pi is **Rasp_714**.

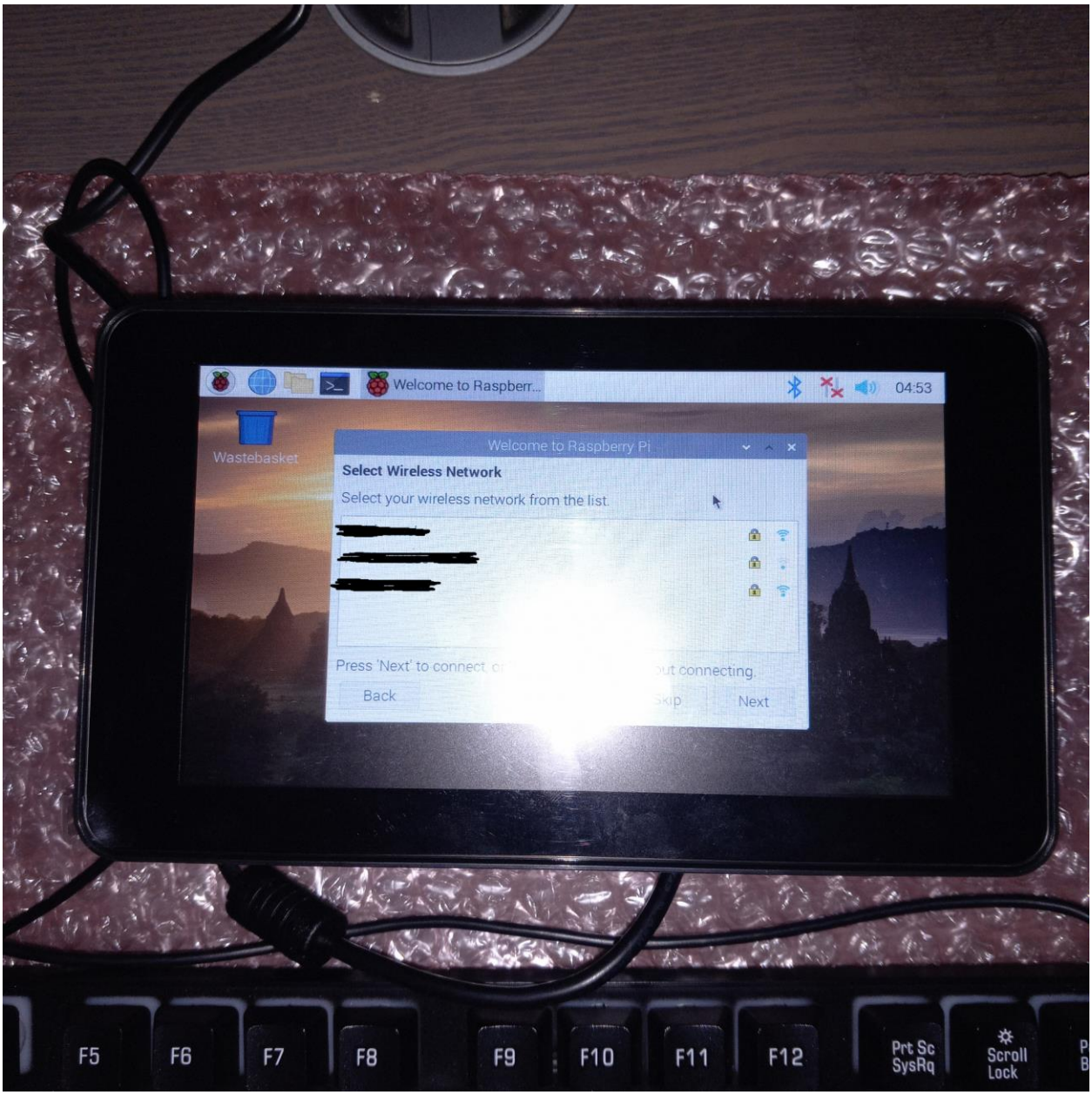
18. Next is to select a country, language, and time zone. The country I have selected is the united states, for the language is american english, and lastly for the time zone is los angeles.



19. Next is Setting up the screen to full display, since it still has space for the display that is not fully utilized, I check on the checkbox and click next.

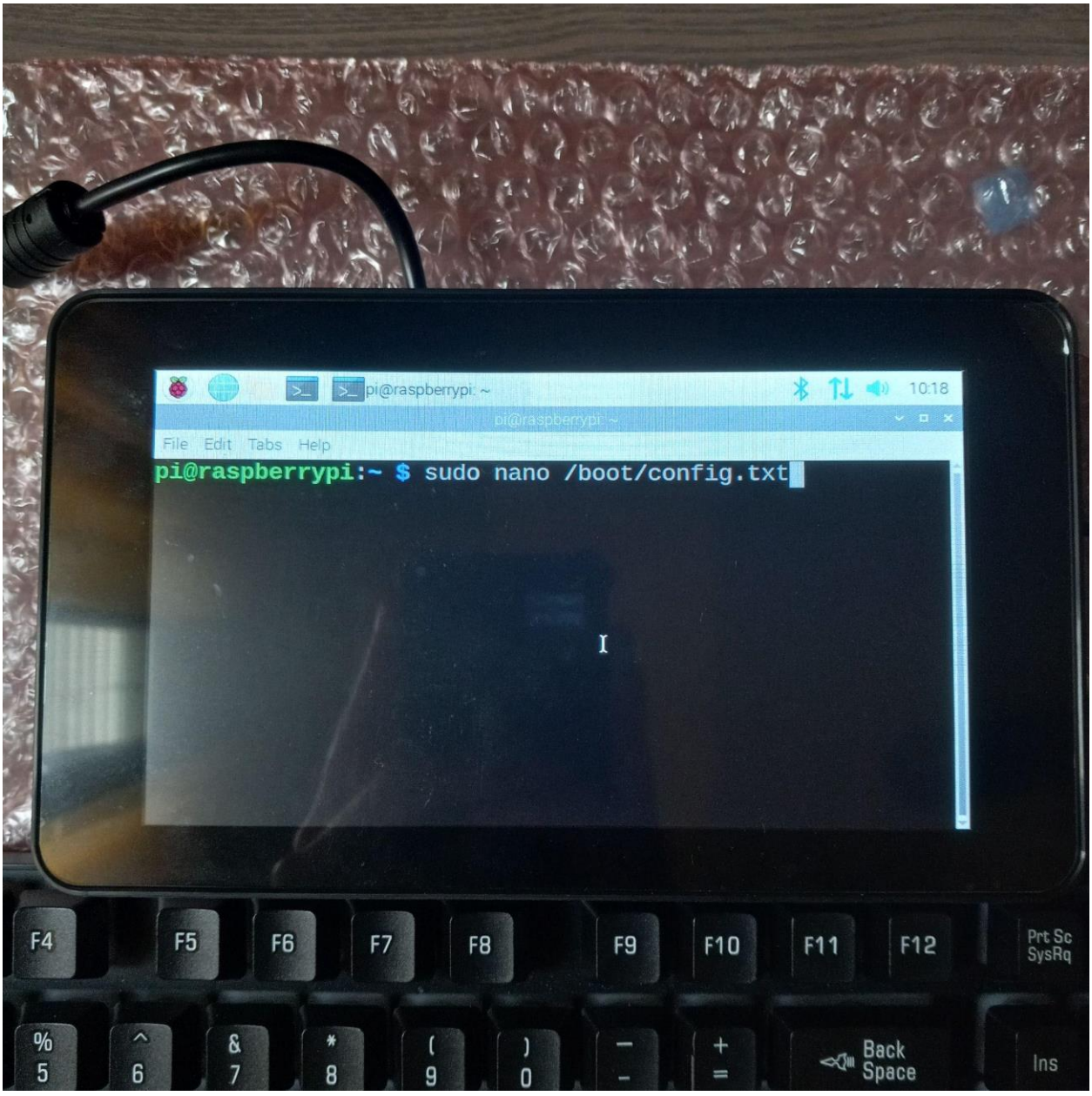
20. Lastly, we must select a wireless connection.

    Note: I skipped this part and used my mobile data (Tethering) instead because I do not have permission to connect to those networks.

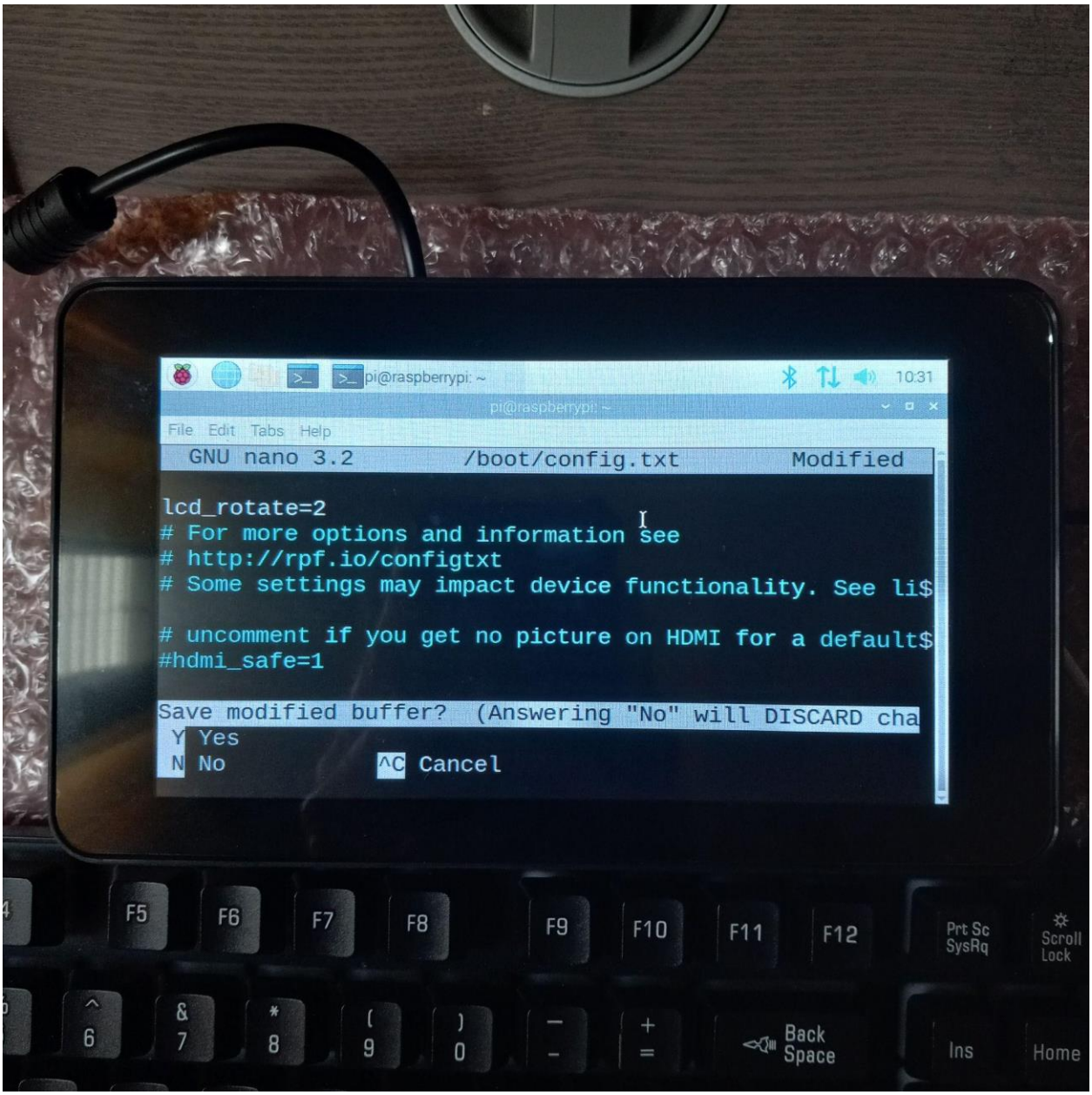


21. Now it is time to rotate the display of the display screen since it is upside down, I have now opened the terminal and it is time to run the command: **sudo nano /boot/config.txt** this command will open the boot configuration file, so we can edit the configuration file

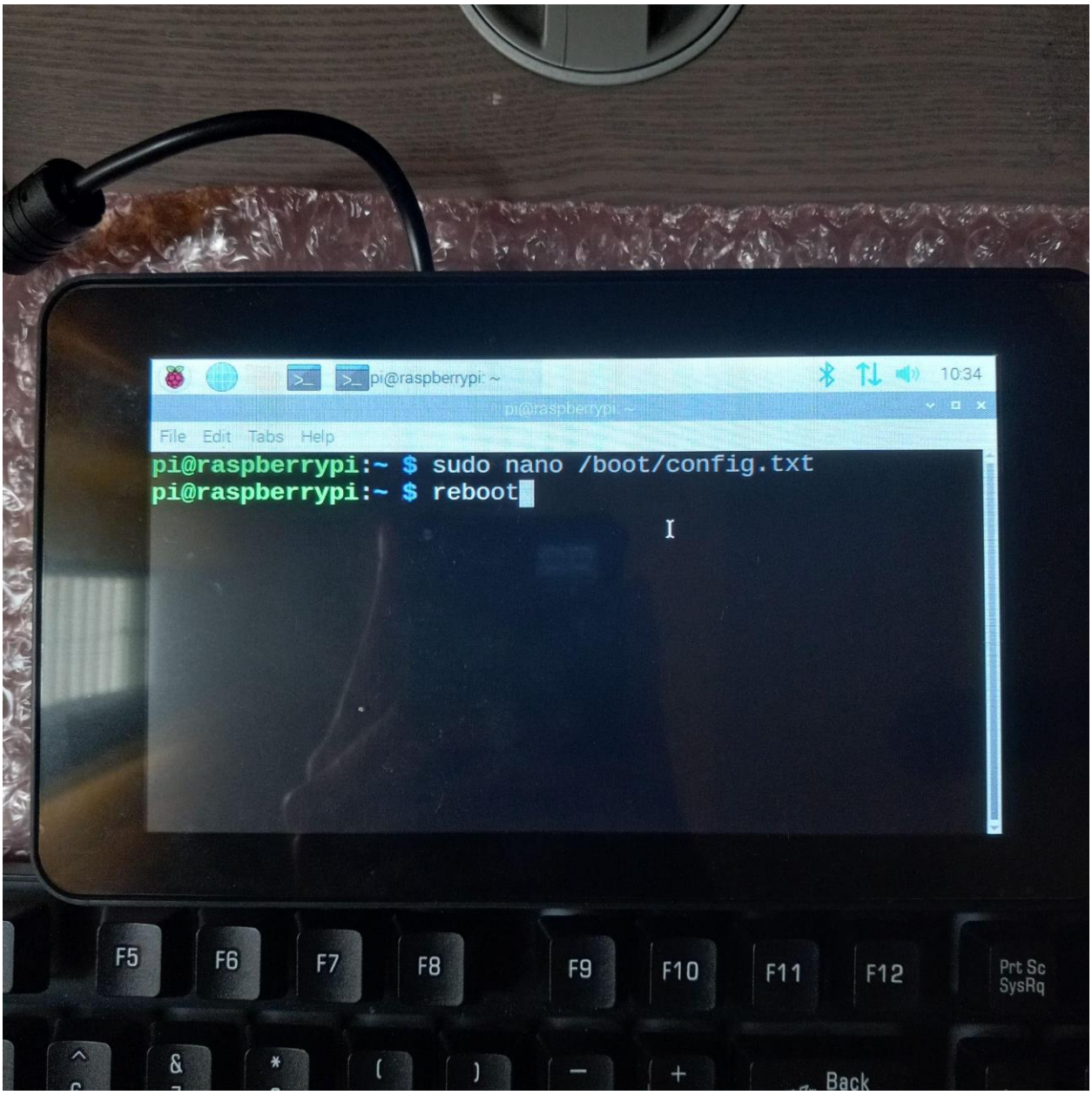
    Note: I have placed the raspberry Pi/screen display on its back because its display is upside down, I will fix that in the following slides.



22. I added a line called **lcd_rotate=2.** What this line does is rotate the display output on the Raspberry Pi's LCD or screen by 180 degrees. Next, I press the key **CTRL+X** and **y + enter** to save the changes made.

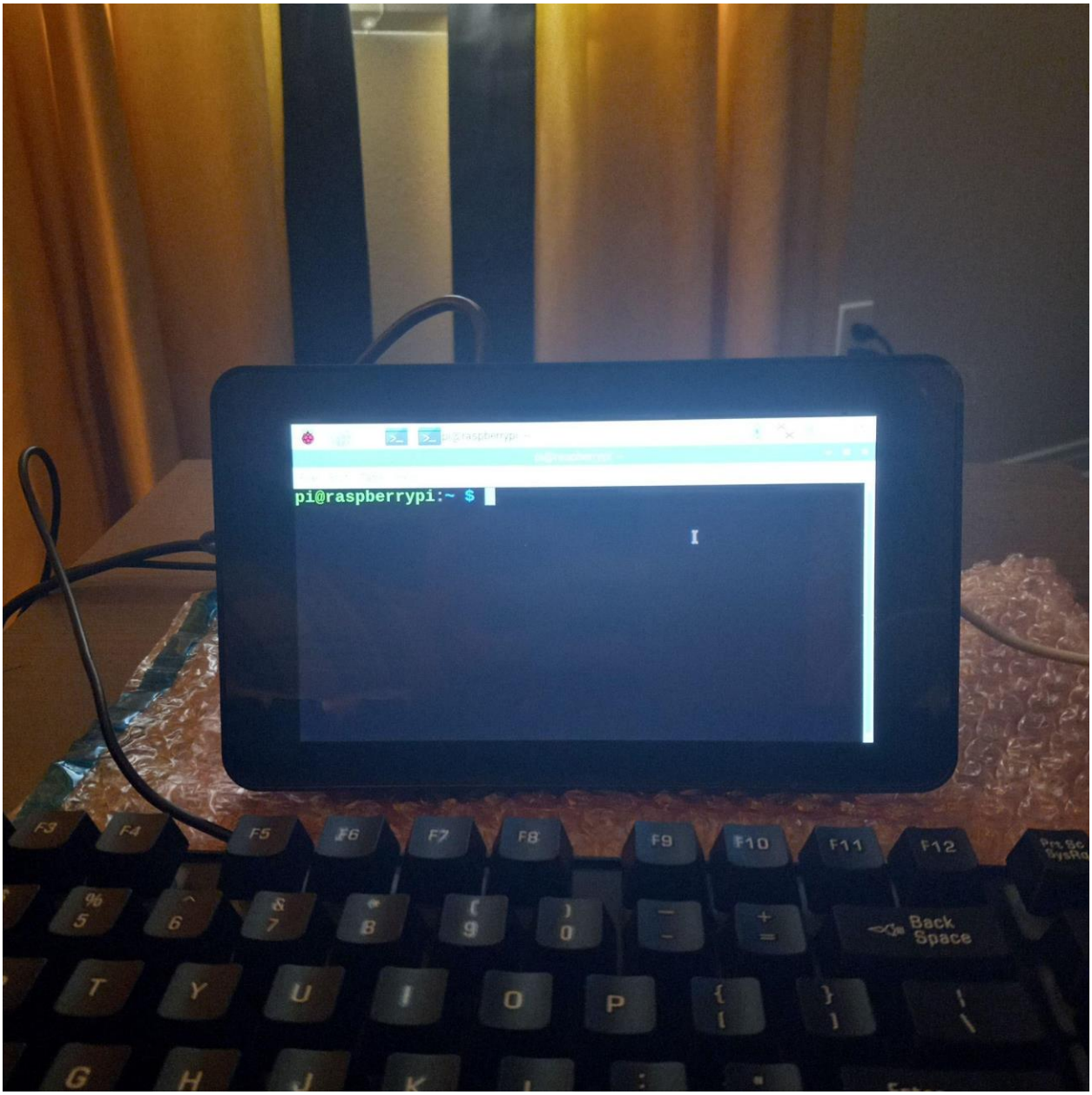


23. To apply the changes, I have saved on the boot configuration file, I run the command: **reboot**, that will reboot the raspberry pi and apply the changes.



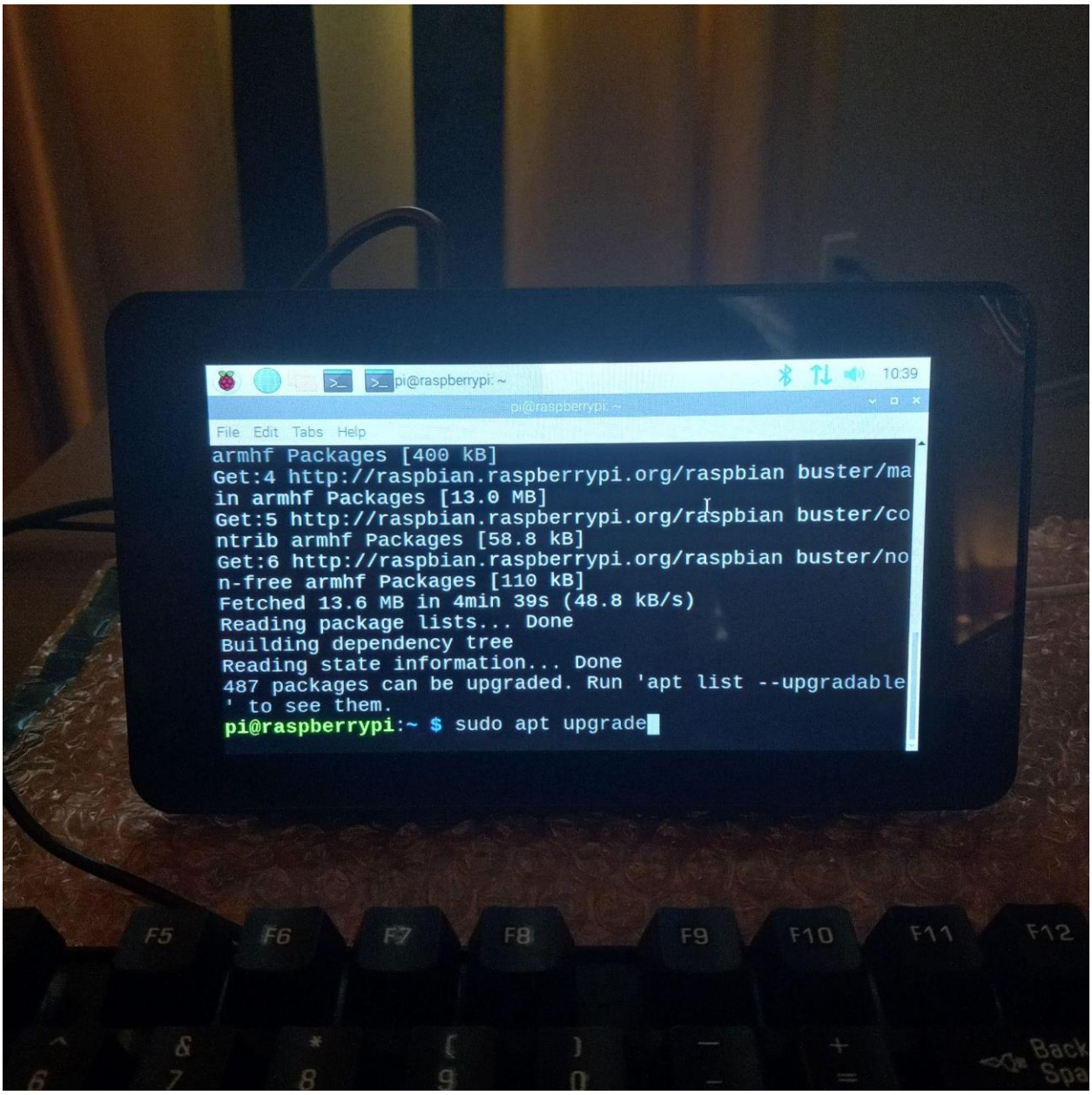
Part 2: Creating a DNS server

24. We finally have our raspberry pi standing and in the proper display rotation, and now we can start executing commands to create a **DNS server.**

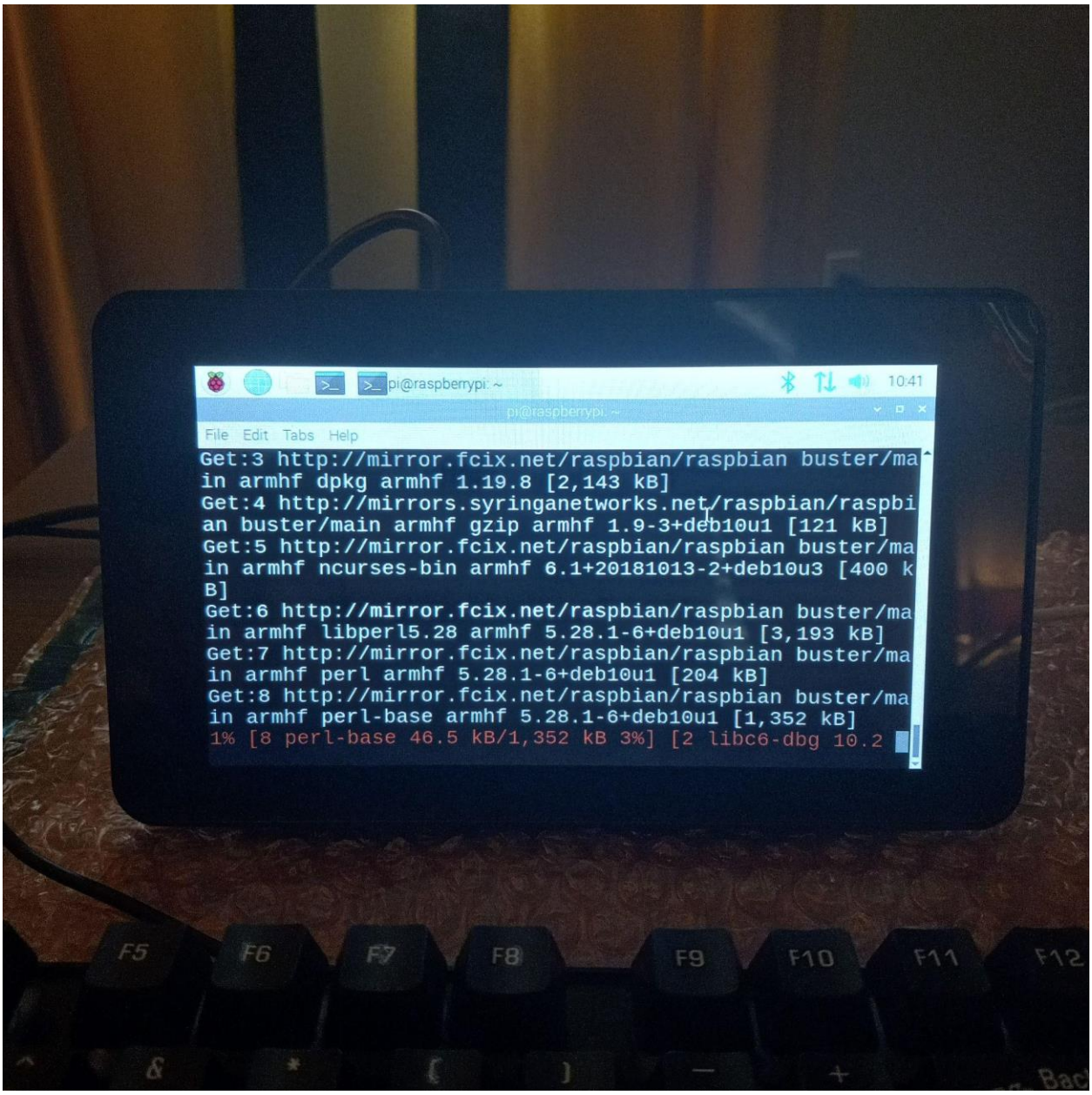


25. The command executed here is: **sudo apt update**, this command will update the local package lists of accessible software packages from the system's repositories. I press **y** and **enter** to continue with our update.

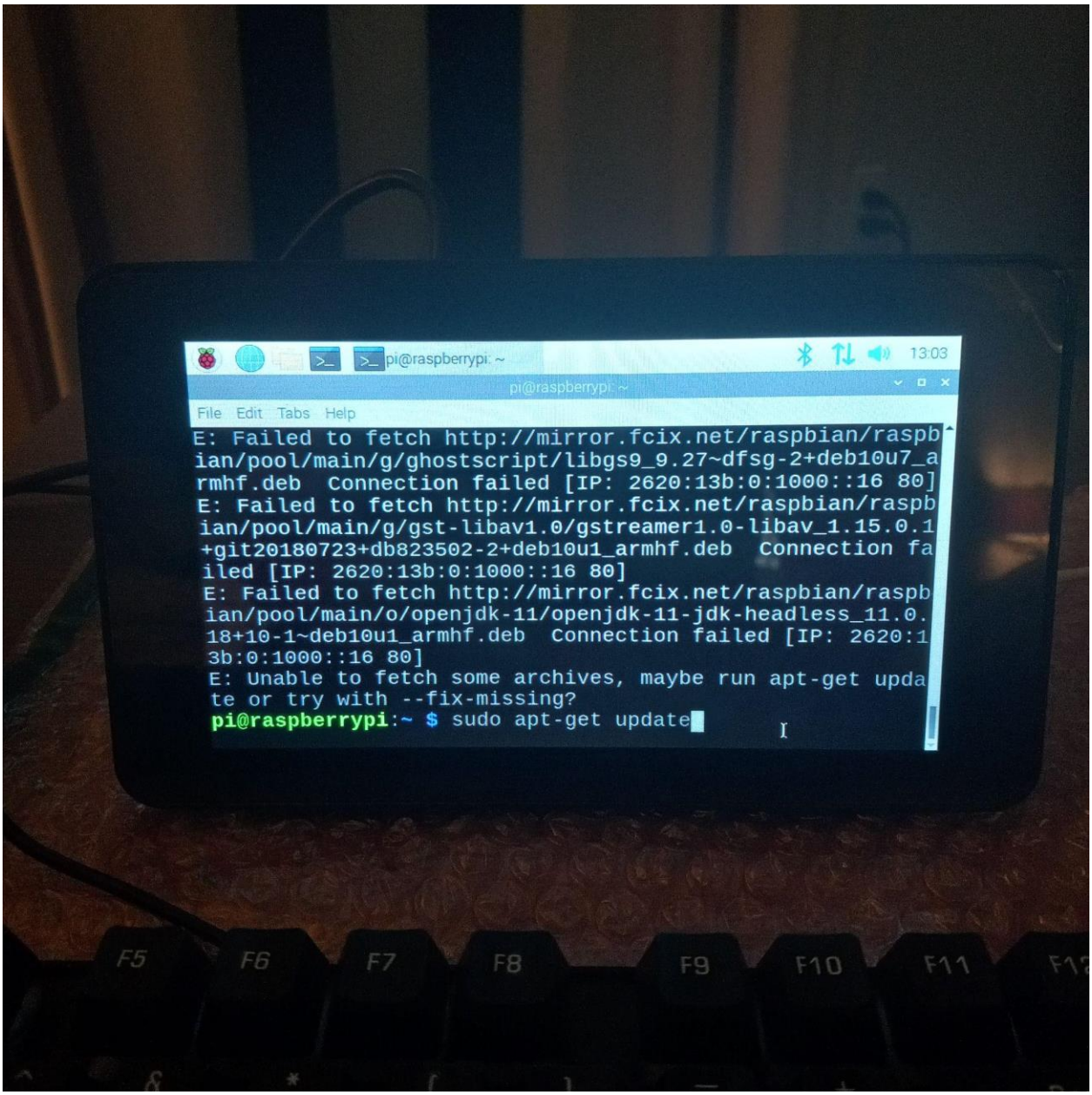
26. After our previous command is finished, I run the next command called: **sudo apt upgrade**, this command will upgrade the system's installed software packages to the most recent versions.



27. After pressing **y** and **enter**, I will wait for the upgrade to finish.

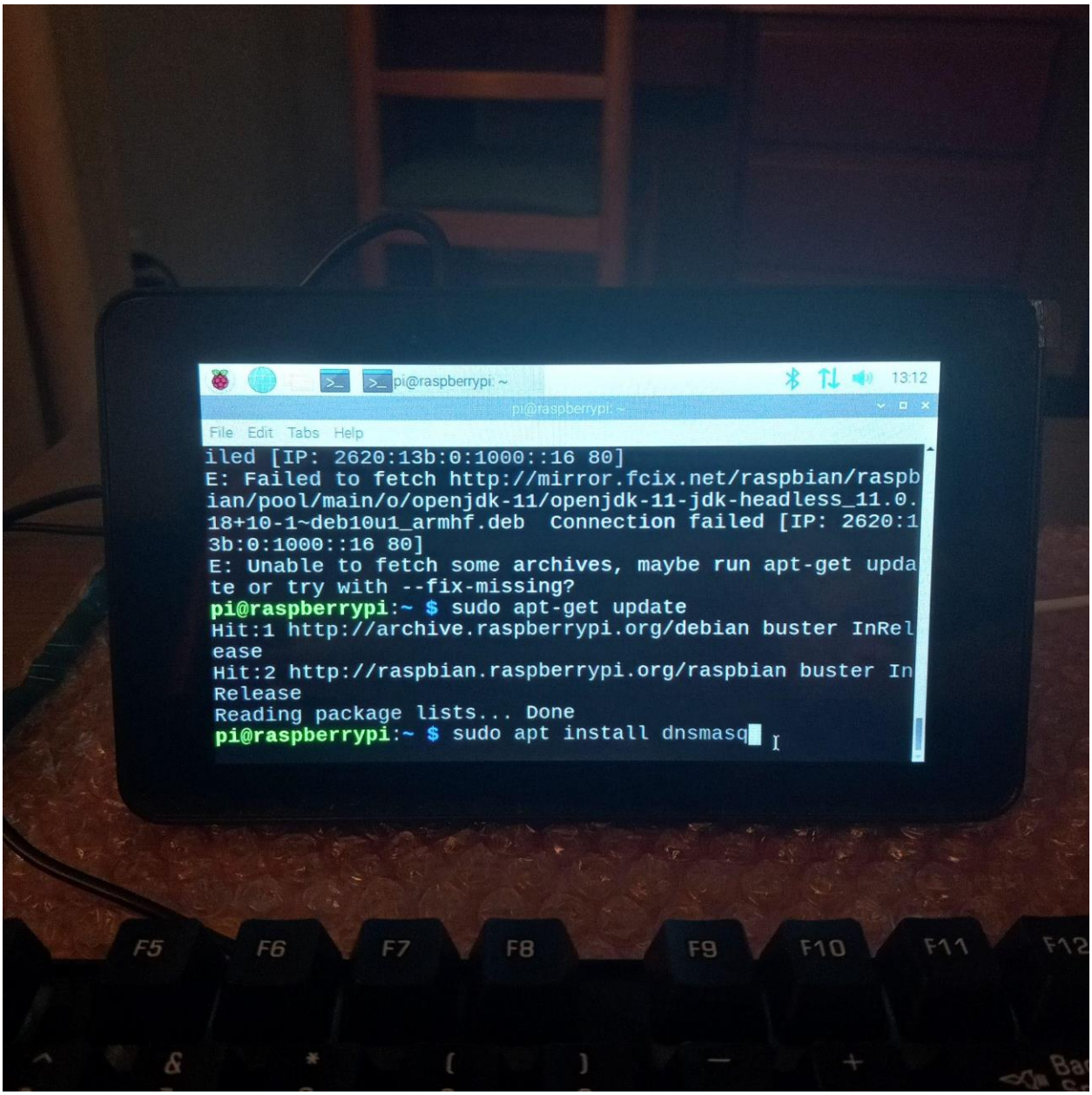


28. The upgrade is now finished. However, some archives are missing so we need to download those archives by running the command: **sudo apt-get update**, and that will get our missing archives.

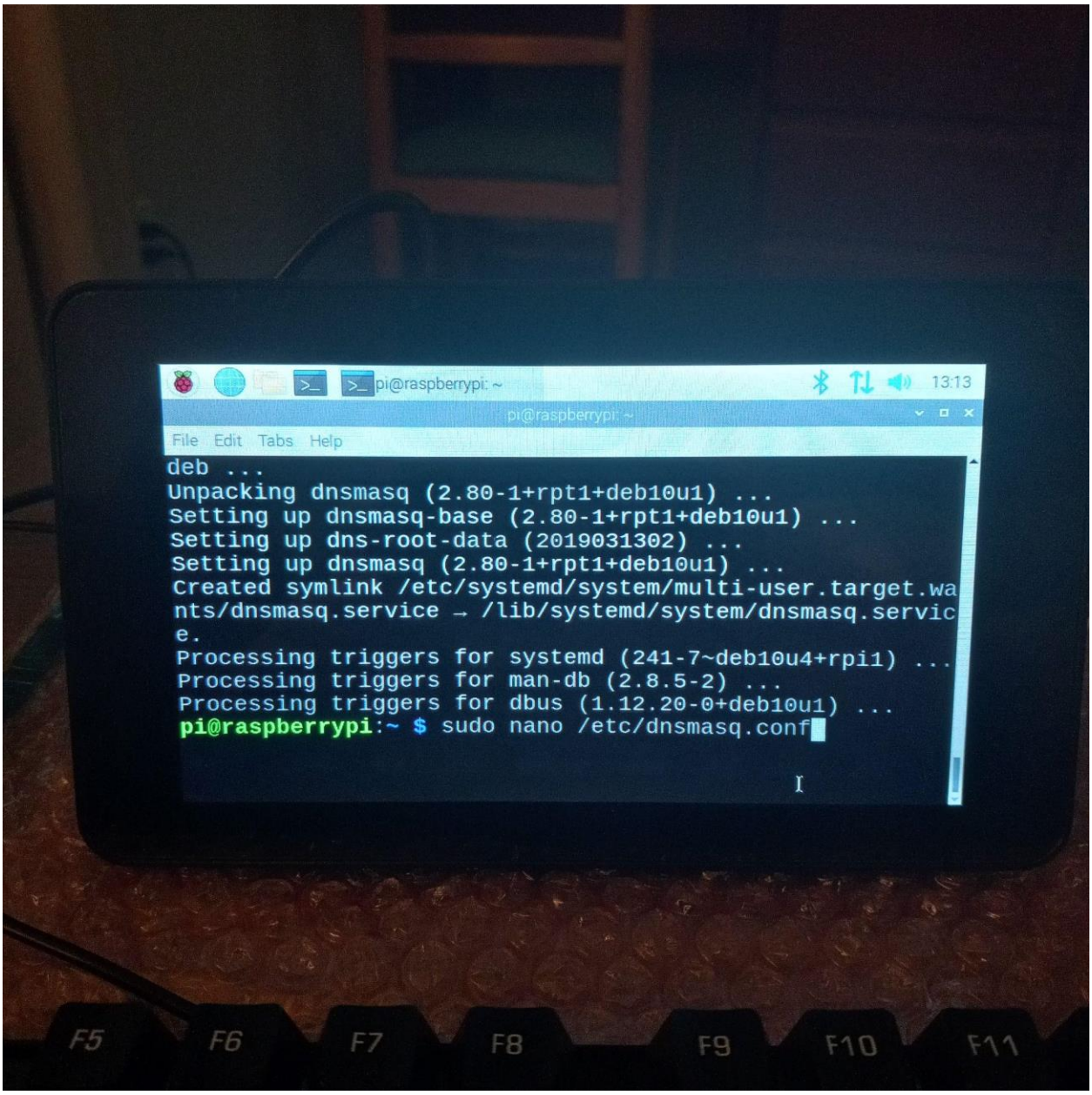


29. The next step is installing dnsmasq, since we need that software package to create our **DNS server**, so I executed the command: **sudo apt install dnsmasq**.

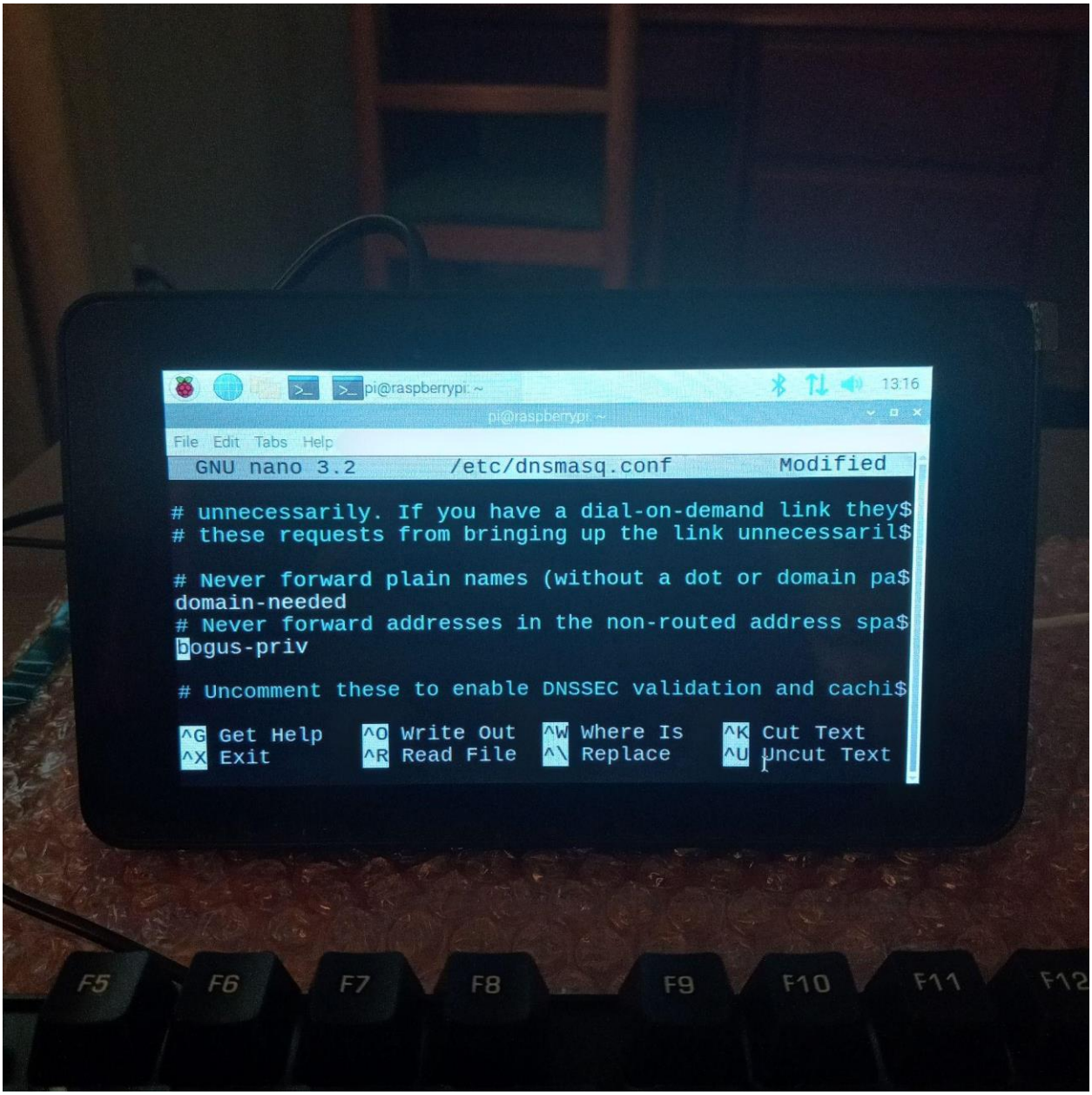
Note: **dnsmasq** is a lightweight and simple DNS (Domain Name System) forwarder and DHCP (Dynamic Host Configuration Protocol) server for Linux-based computers.



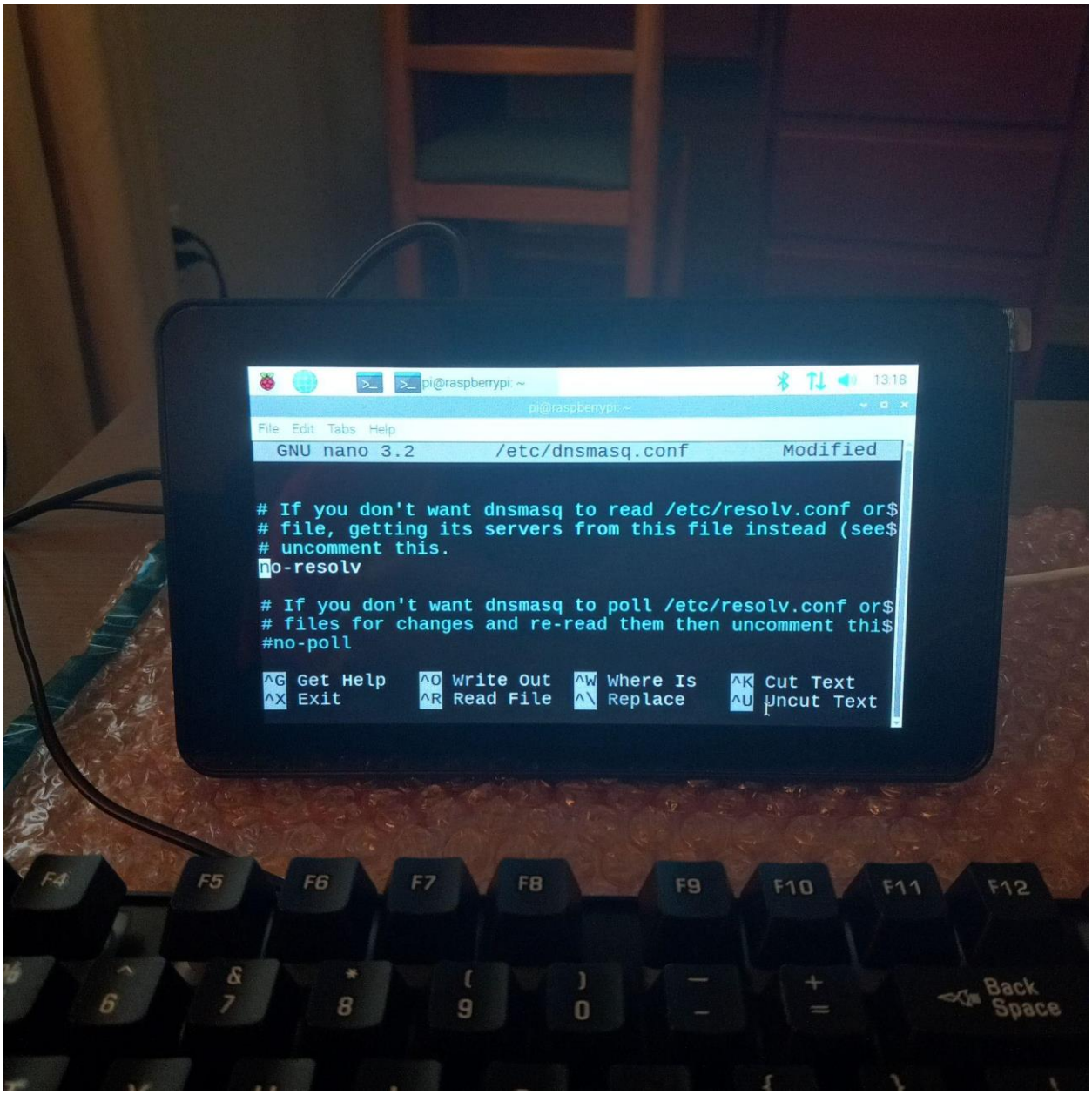
30. After the installation of dnsmasq is done, I can now run the command: **sudo nano /etc/dnsmasq.conf**, this command will open the dnsmasq configuration file, so we can edit or add data to the configuration file in order to create a **DNS server.**



31. I have now opened the dnsmasq configuration file, and now it is time to add the following: **domain-needed**, **bogus-priv**, **no-resolv**, **server=8.8.8.8**, **server=8.8.4.4**, and replace **cache-size=1000** instead of **cache-size=150**. In the image below, adding **domain-needed** will set the DNS server so that it will not send domain names or names without a dot (.) to upstream servers.

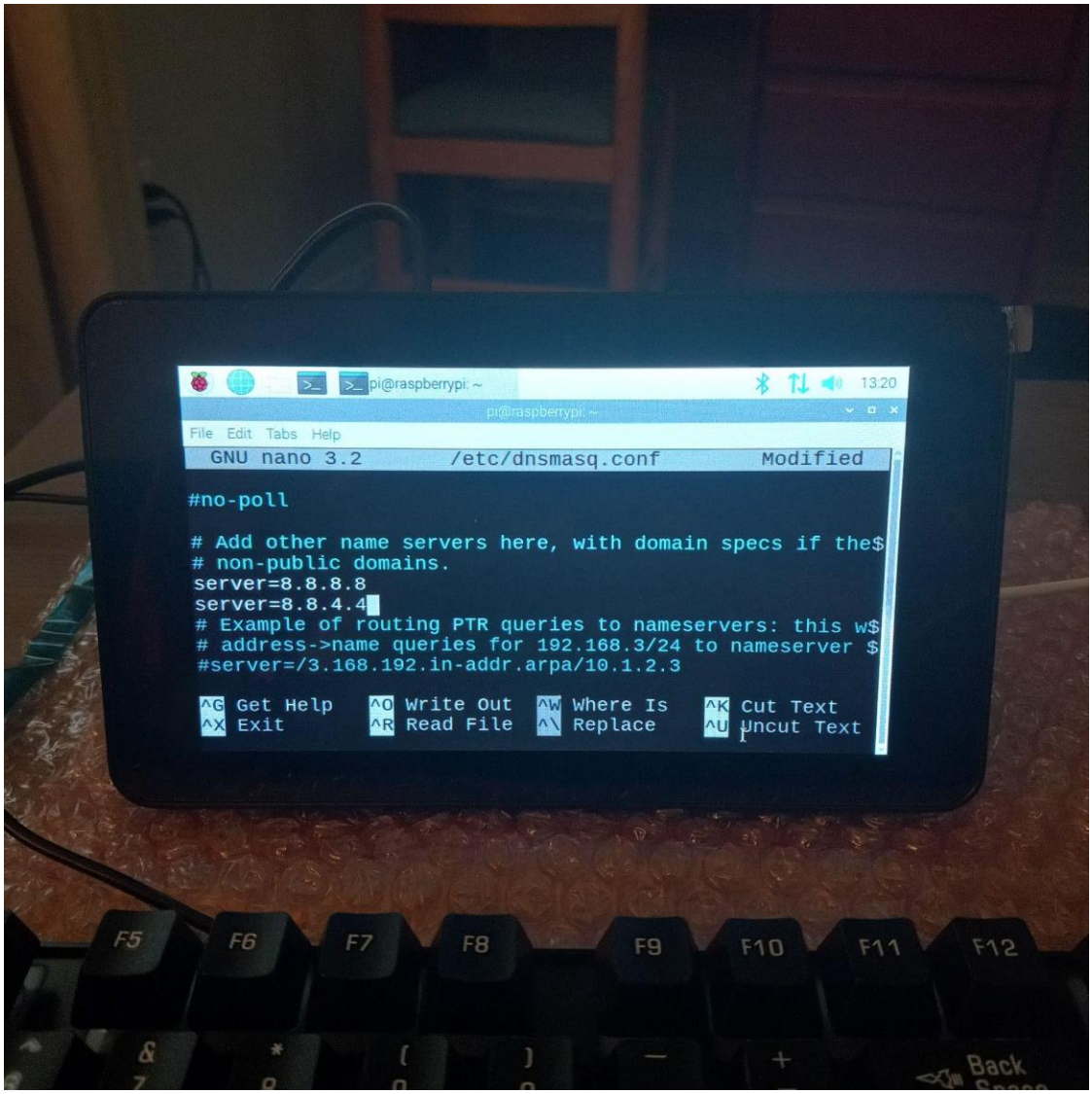


32. (Continuation of 31, opening/editing the dnsmasq configuration file) **no-resolv** prevents the DNS server from sending upstream DNS servers local IP range reverse-lookup requests.

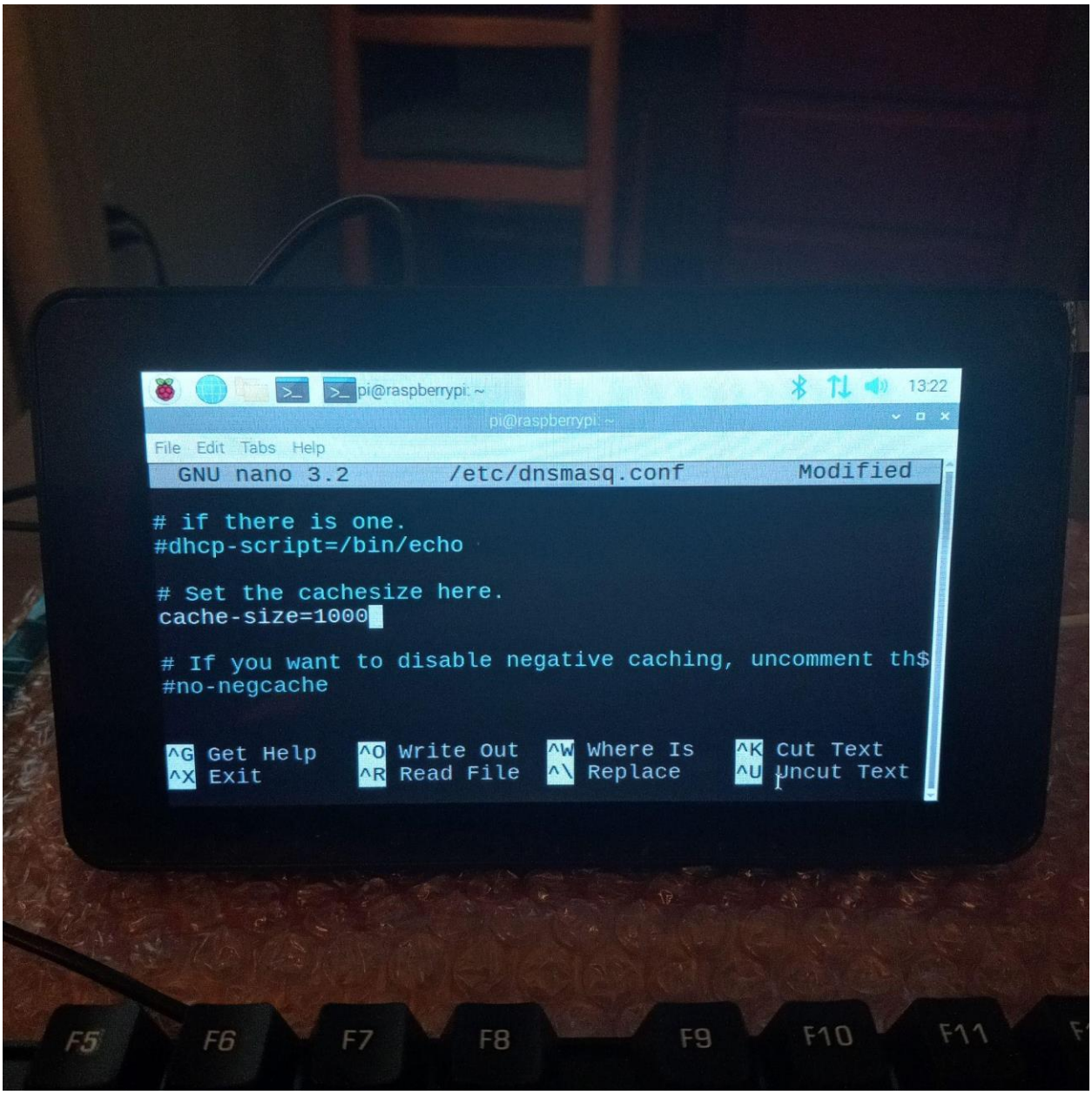


33. (Continuation of 31, opening/editing the dnsmasq configuration file) Next, I have added **server=8.8.8.8**, **server=8.8.4.4** to the dnsmasq configuration file

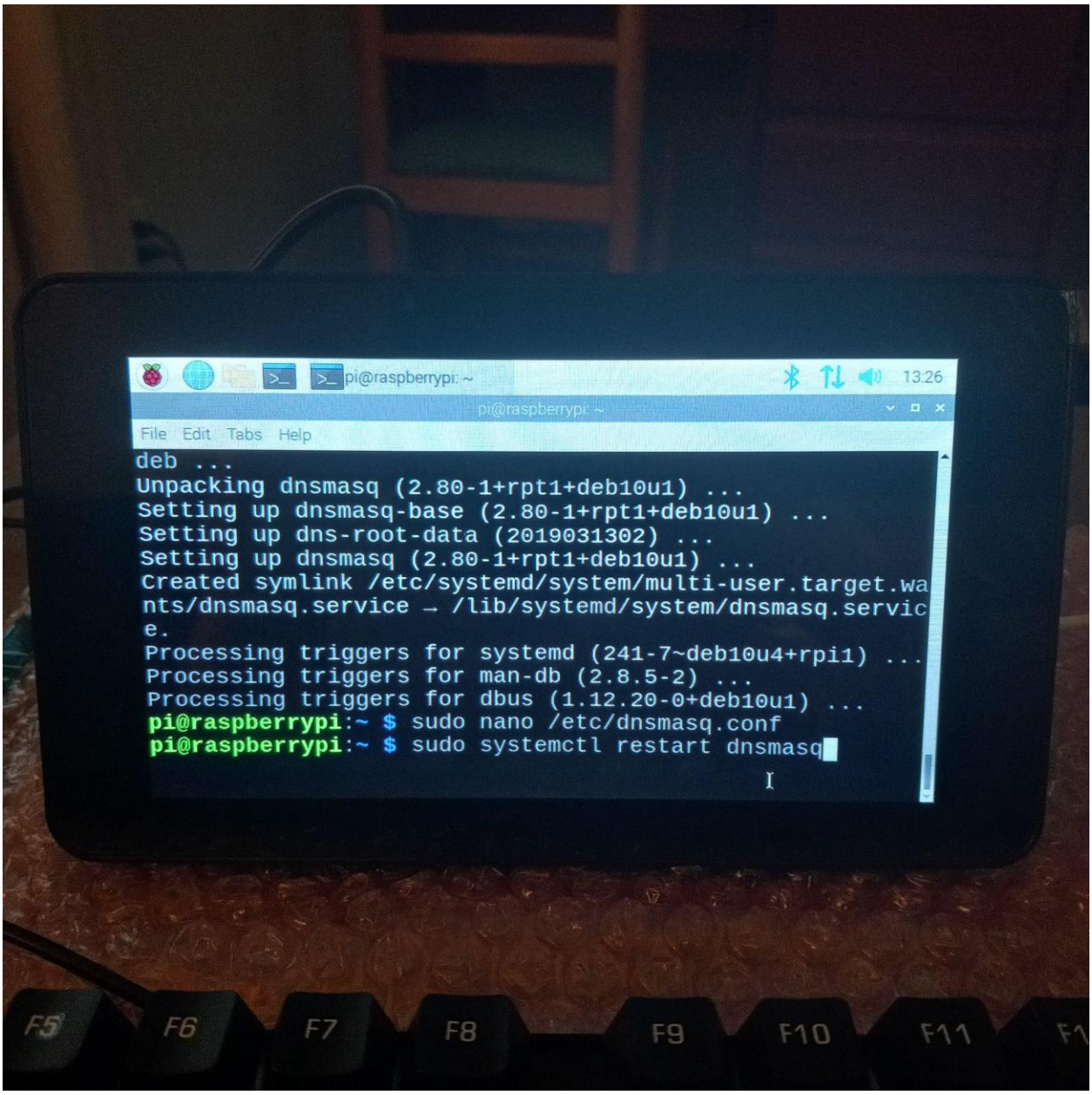
Note: **server=8.8.8.8** instructs dnsmasq to use the public DNS server run by Google at the IP address 8.8.8.8 as one of the DNS servers for DNS forwarding. **server=8.8.4.4**: This line instructs dnsmasq to use Google's public DNS server with the IP address 8.8.4.4 for as another DNS forwarding server.



34. So, I press the key **CTRL+W**, and type **#cache-size** so we find that line we need to uncomment and change the cache size from **150** to **1000**. Increasing the cache size will improve DNS performance and can store more DNS responses locally.

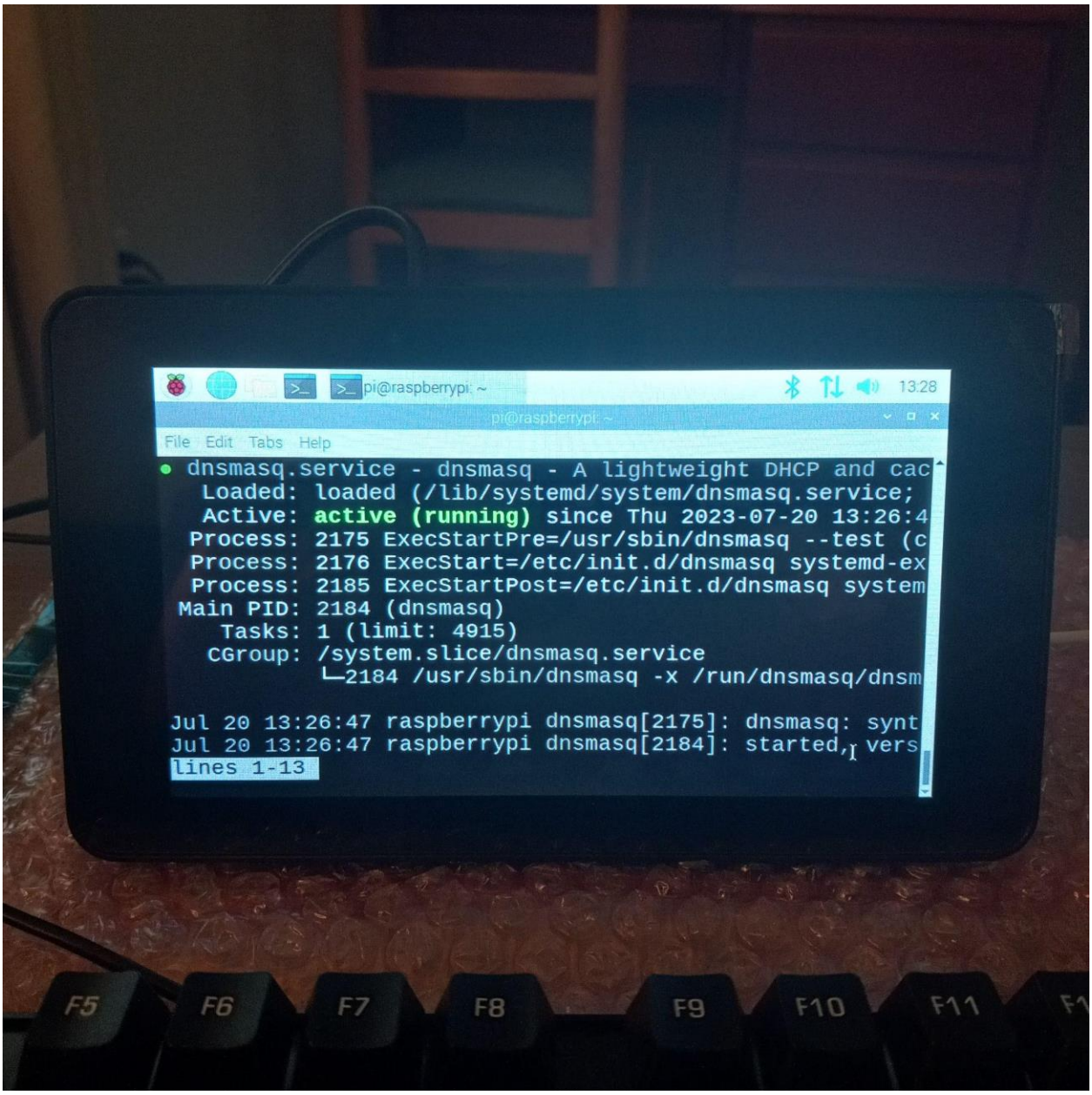


35. After saving those changes the next step is to apply them by restarting dnsmasq, so the next command I execute is **sudo systemctl restart dnsmasq**, restarting dnsmasq will apply those changes.

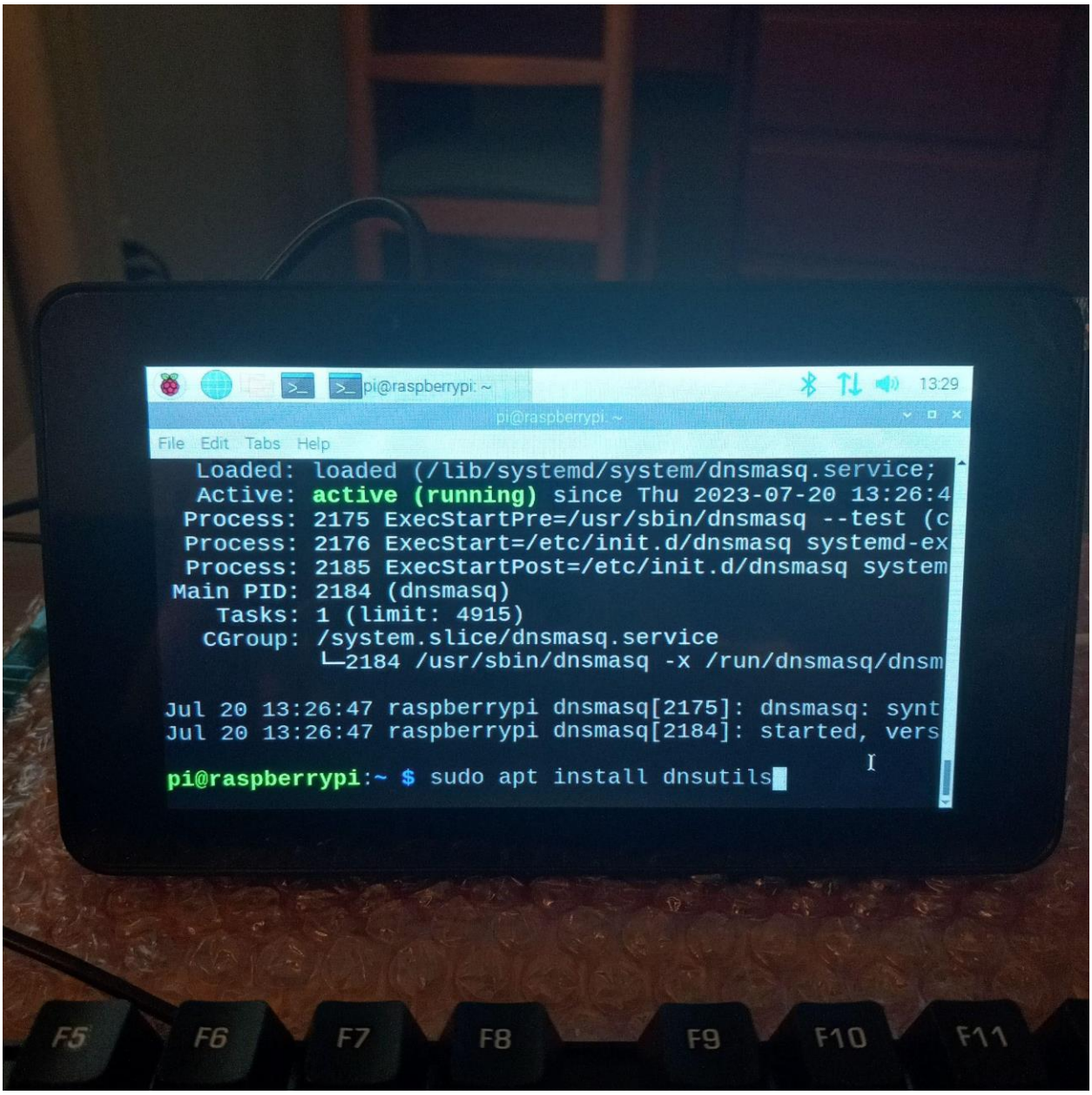


36. After restarting dnsmasq, I executed the command: **sudo systemctl status dnsmasq**, this command will check the status of dnsmasq, and as we can see from the photo below it is currently active/running

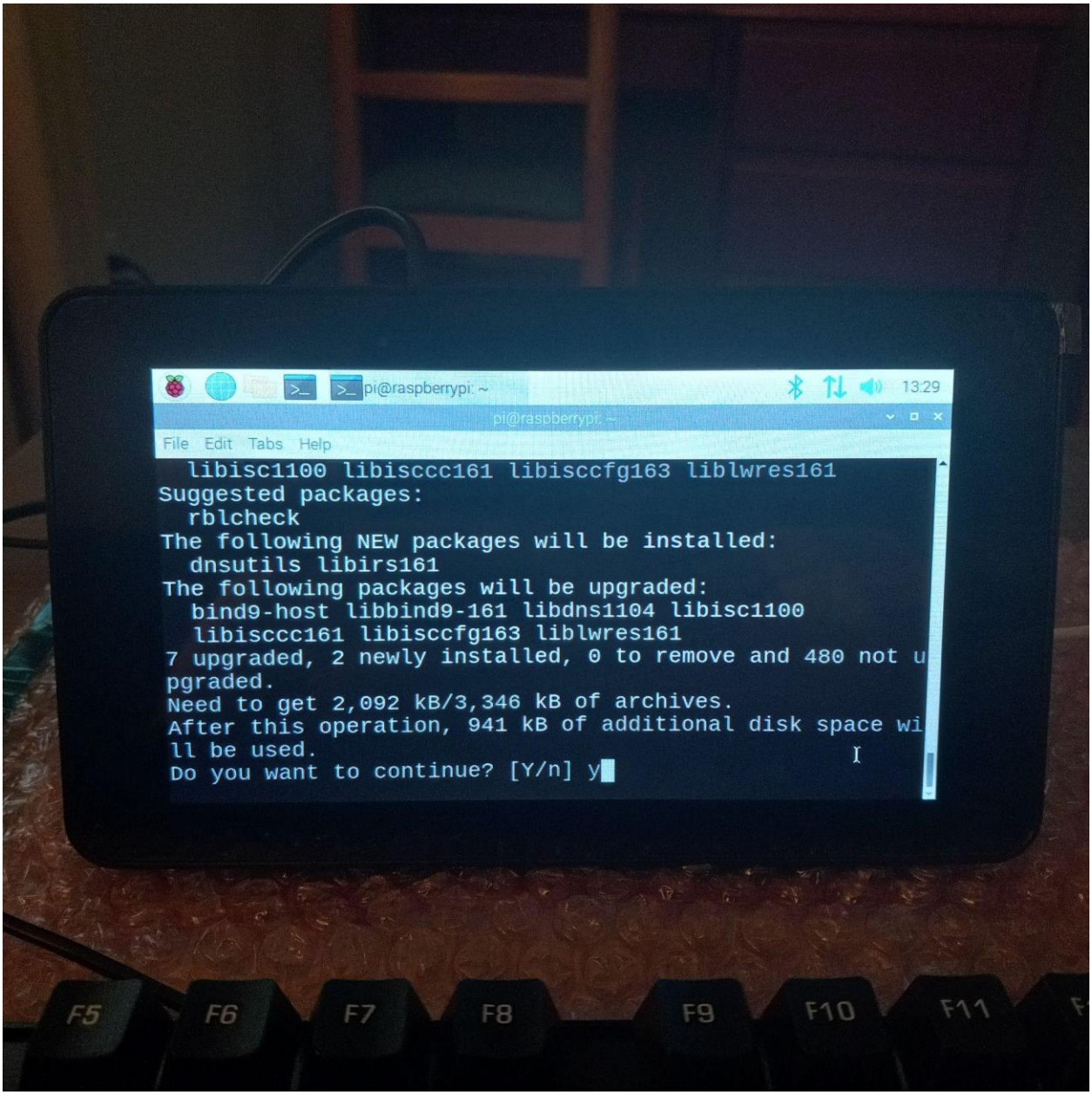
    Note: I forgot to take a photo before executing the command: **sudo systemctl status dnsmasq** which is not seen the image below.



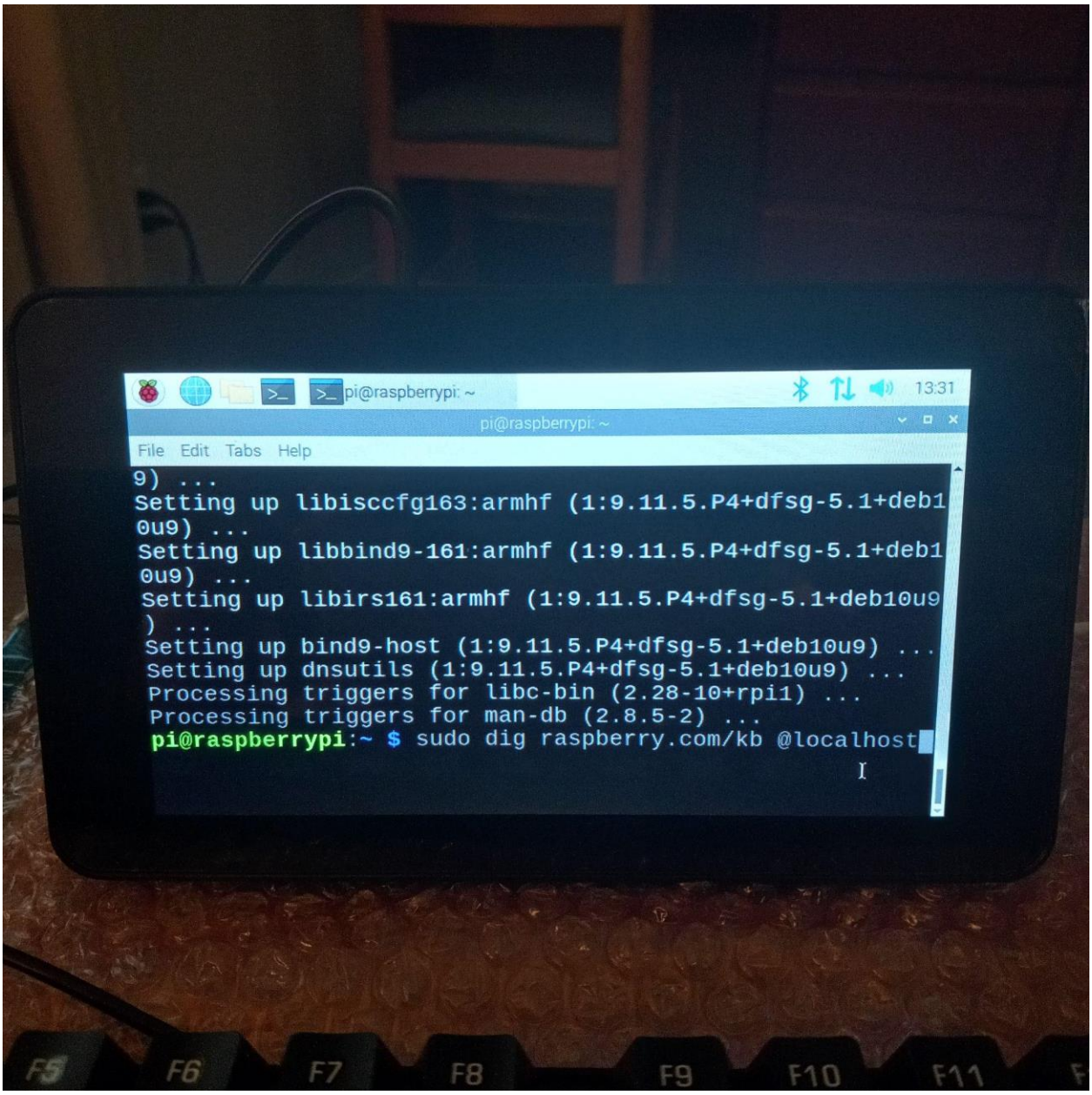
37. The next thing is to install **dnsutils**, so I run the command: **sudo apt install dnsutils**. Since we will use the **dig** command later.



38. It prompts us to accept or decline, so I press **y**, and **enter** to continue with the installation.

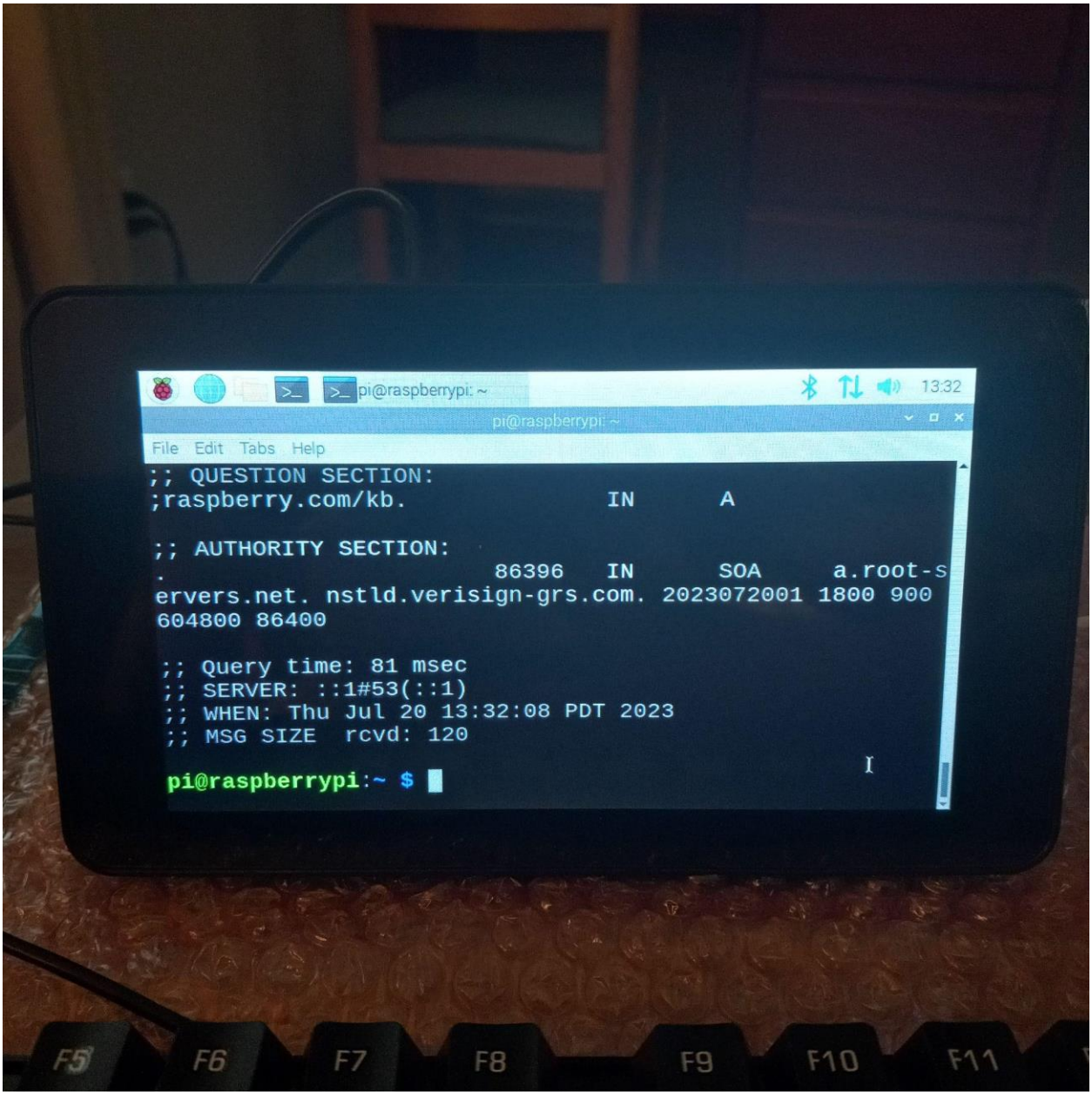


39. Now I execute the command: **sudo dig raspberry.com/kb @localhost**, because we want to query our DNS server, so we can retrieve several types of DNS information, and see its query time.

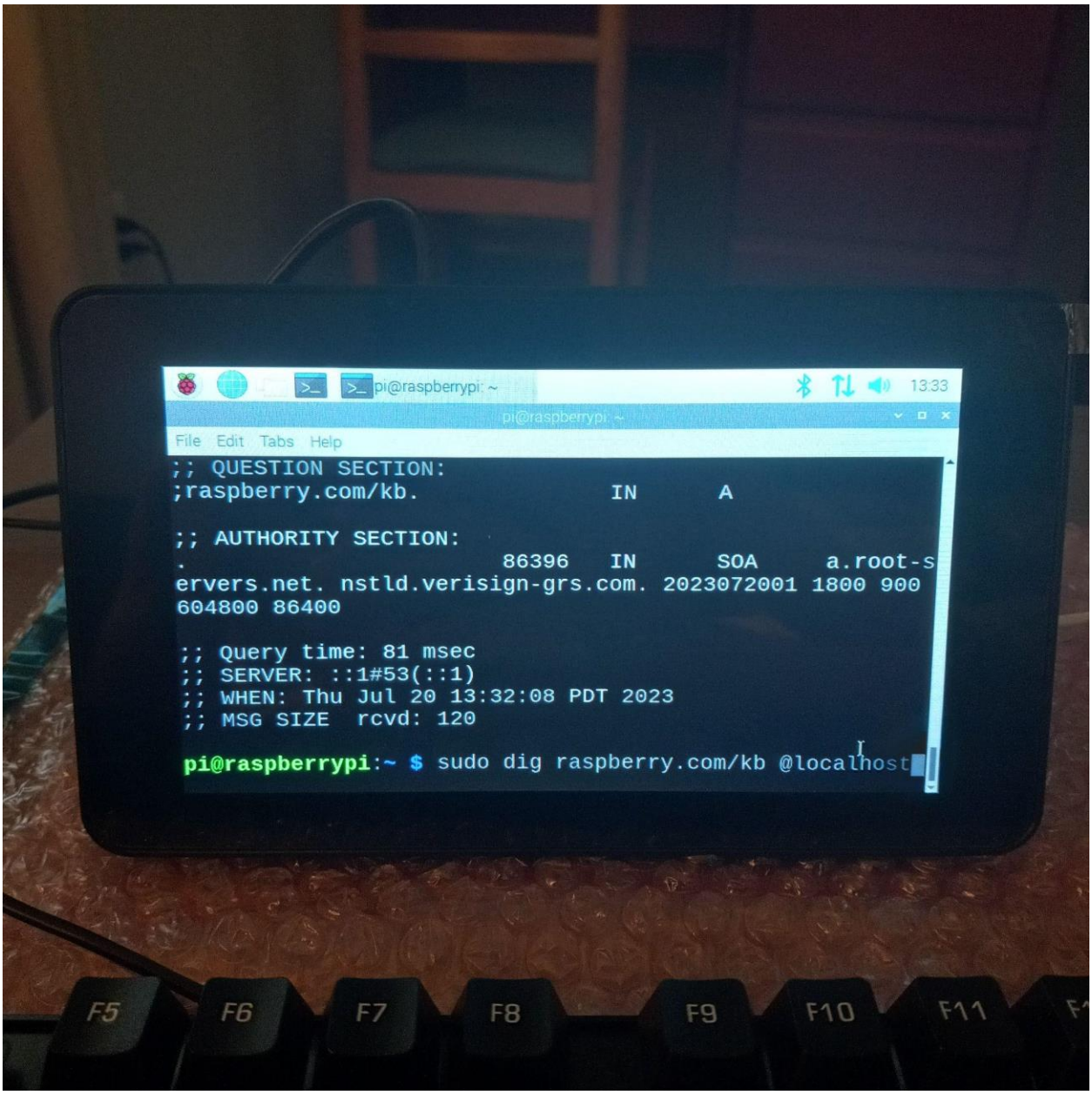


40. Now we can see the query time of our DNS server, the query time is **81 msec**.

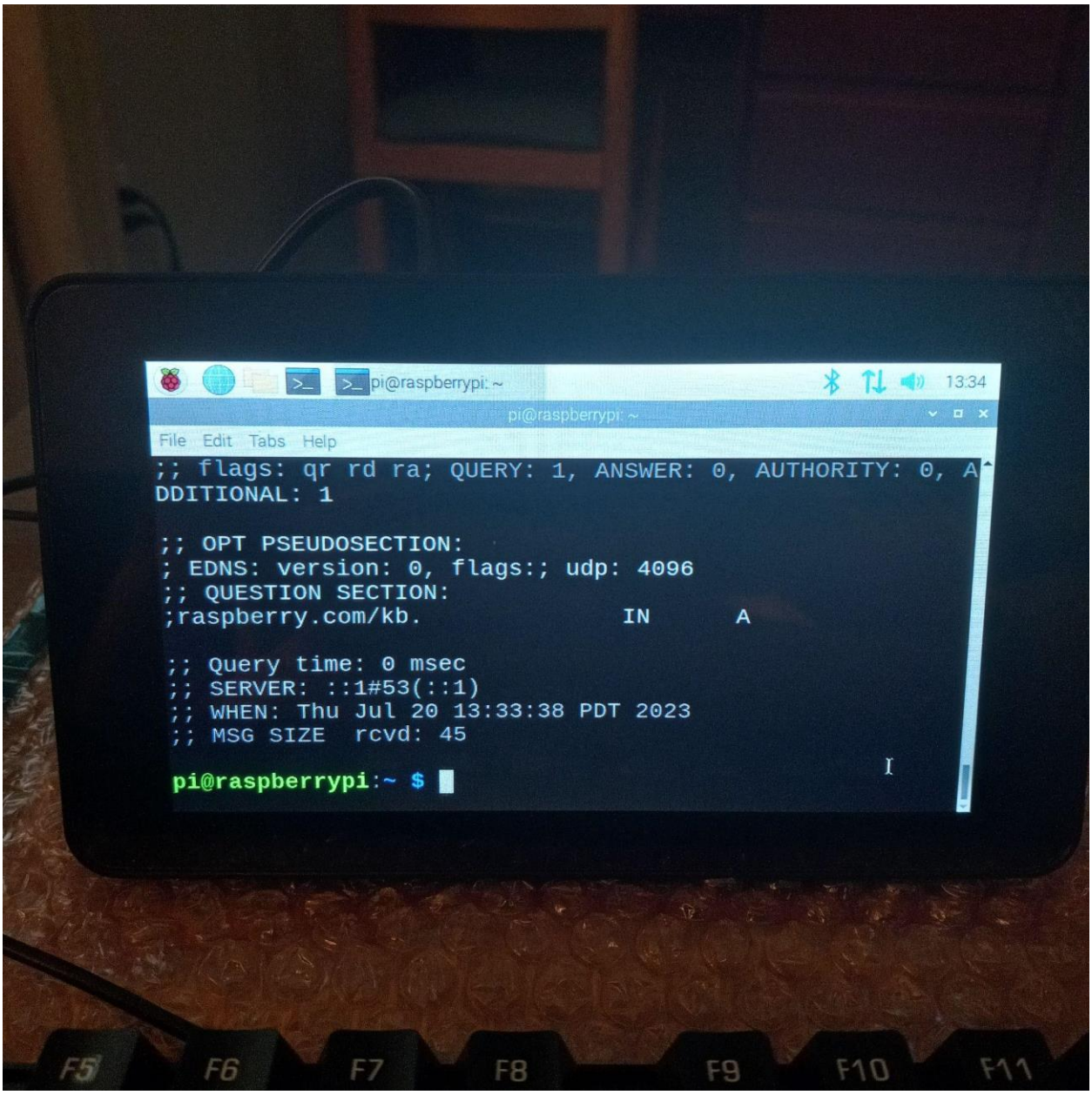
Note: **msec** stands for milliseconds.



41. So, after seeing the query time being **81 msec**, I rerun our previous command: **sudo dig raspberry.com/kb @localhost,** to see the query time if it has changed.

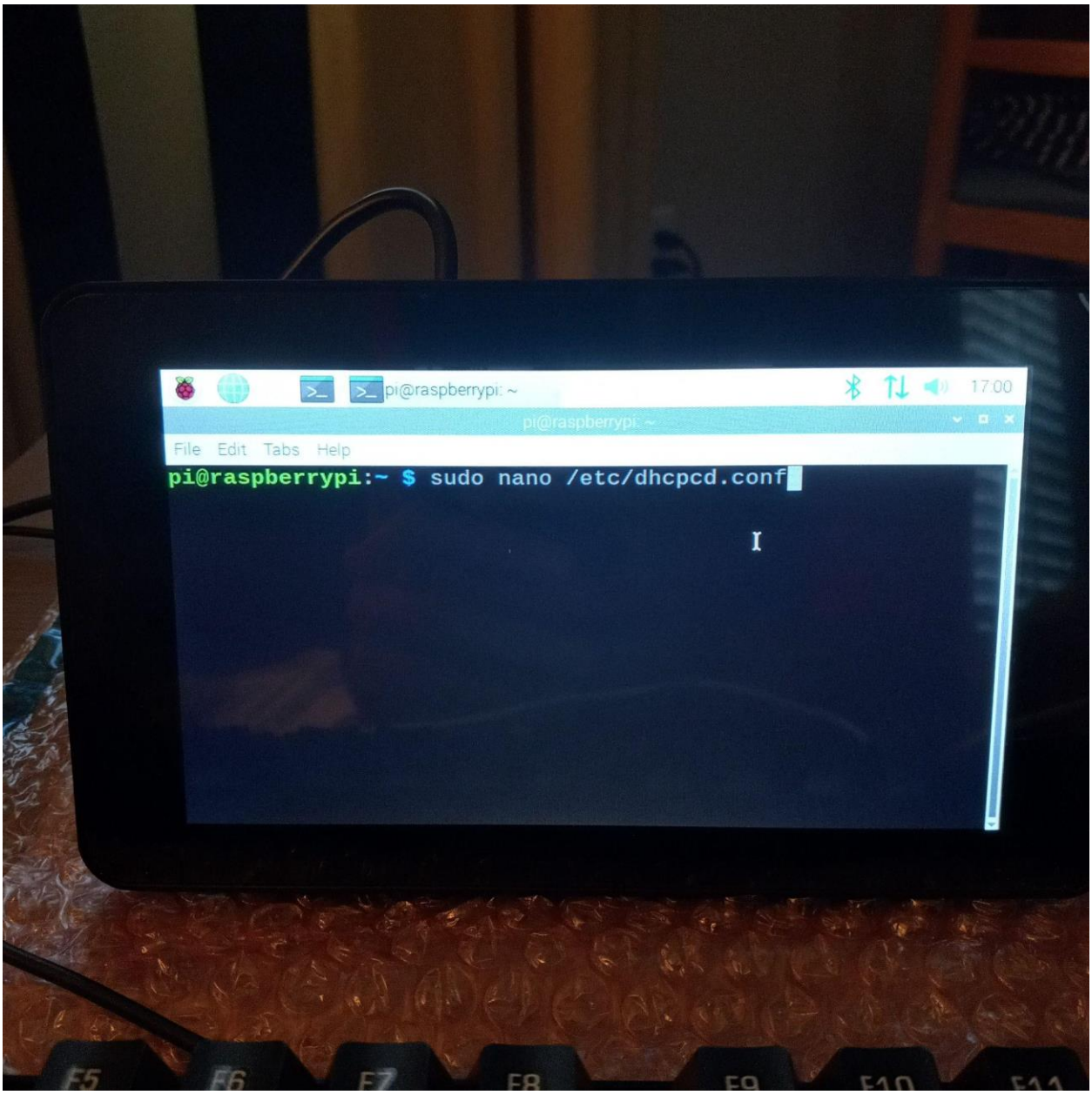


42. Now we can see the query time has changed from **81 msec** to **0 msec**.



Part 3: Creating a DHCP server

43. So, I am going to be creating a DHCP server out of the raspberry pi, so let's begin with opening and editing the **dhcpcd.conf** file, so here I run the command: **sudo nano /etc/dhcpcd.conf**



44. So, we have opened the configuration file, now it's time to set a static ip address to our raspberry pi by adding the following lines:

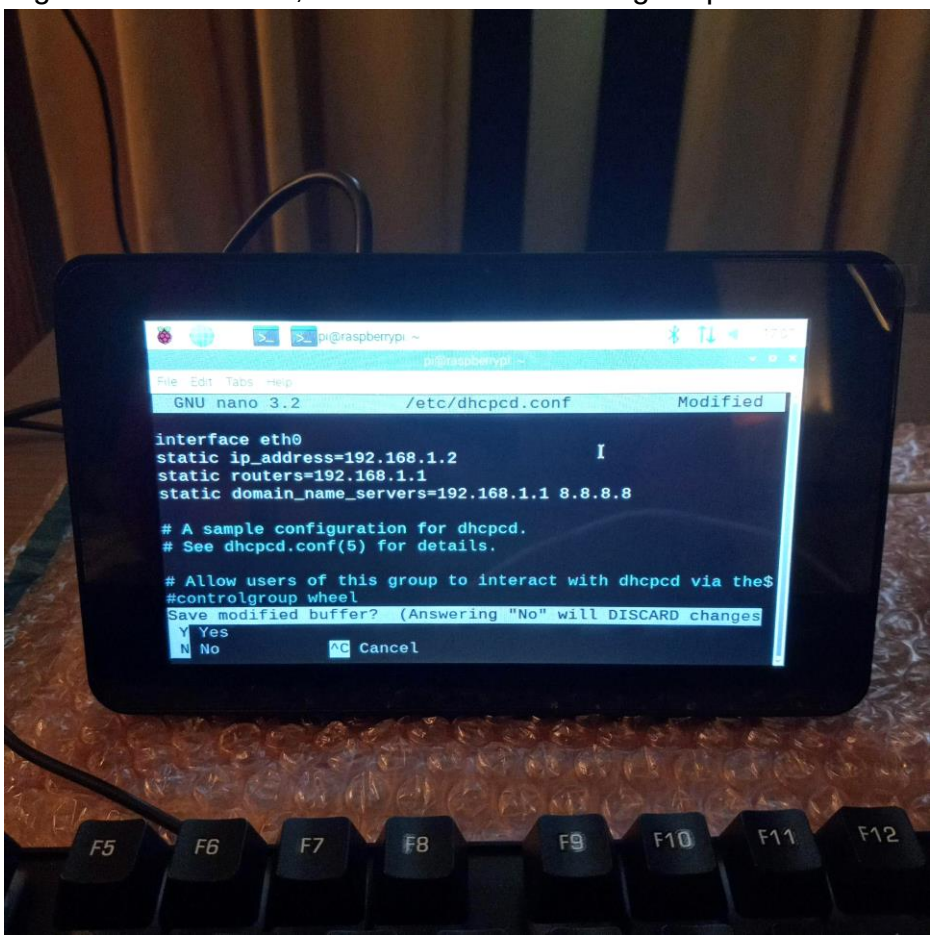
    interface eth0 - This line defines the network interface it is setting; in this example, it is eth0, the Ethernet interface.

    static ip_address=192.168.1.2 - Static ip_address=192.168.1.2 - This

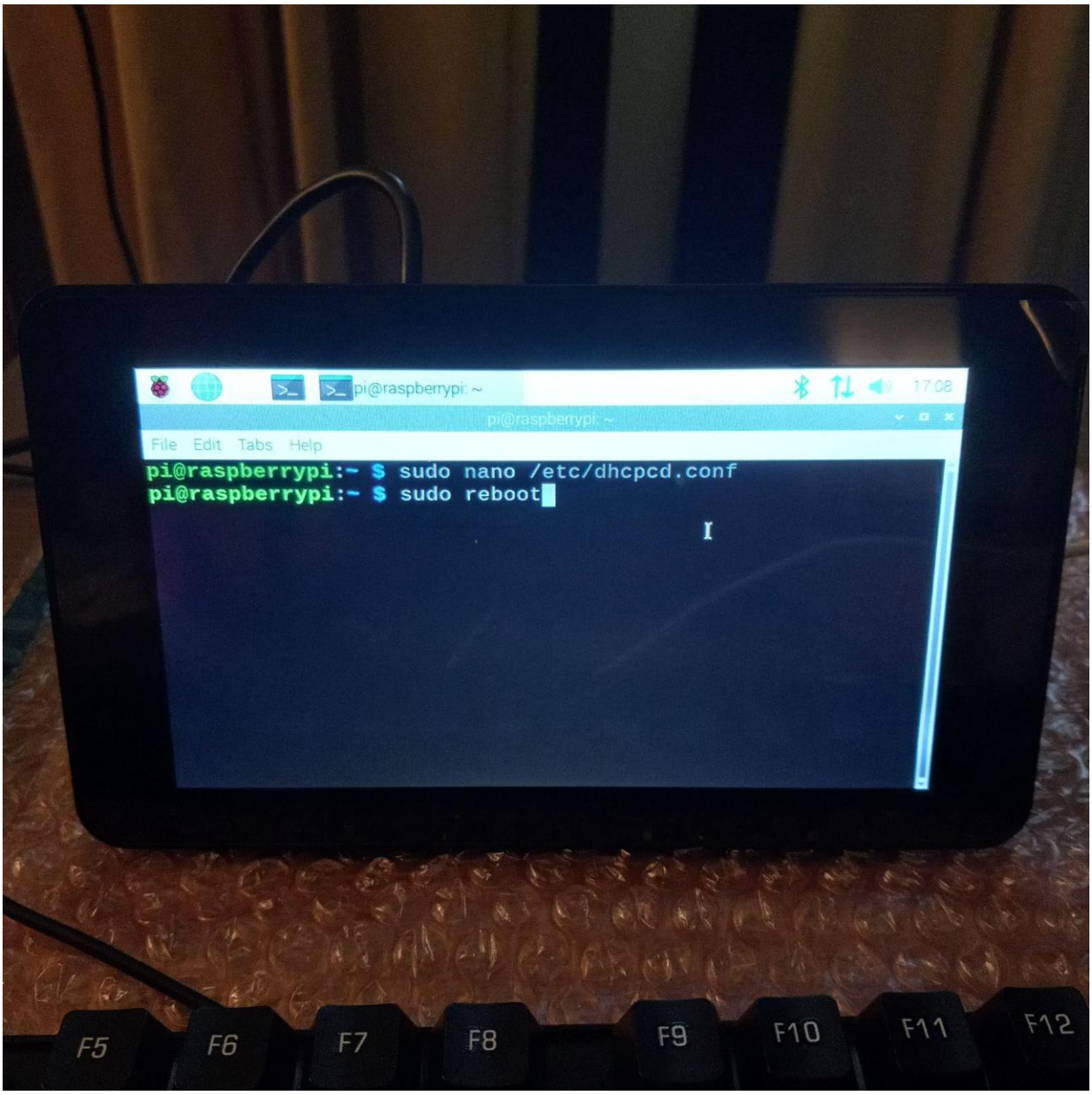
line configures the eth0 interface with a static IP address. A static IP address implies that the Raspberry Pi will always be allocated the same IP address on the network.

static routers=192.168.1.1 - This line makes 192.168.1.1 the default gateway (router) for the eth0 device.

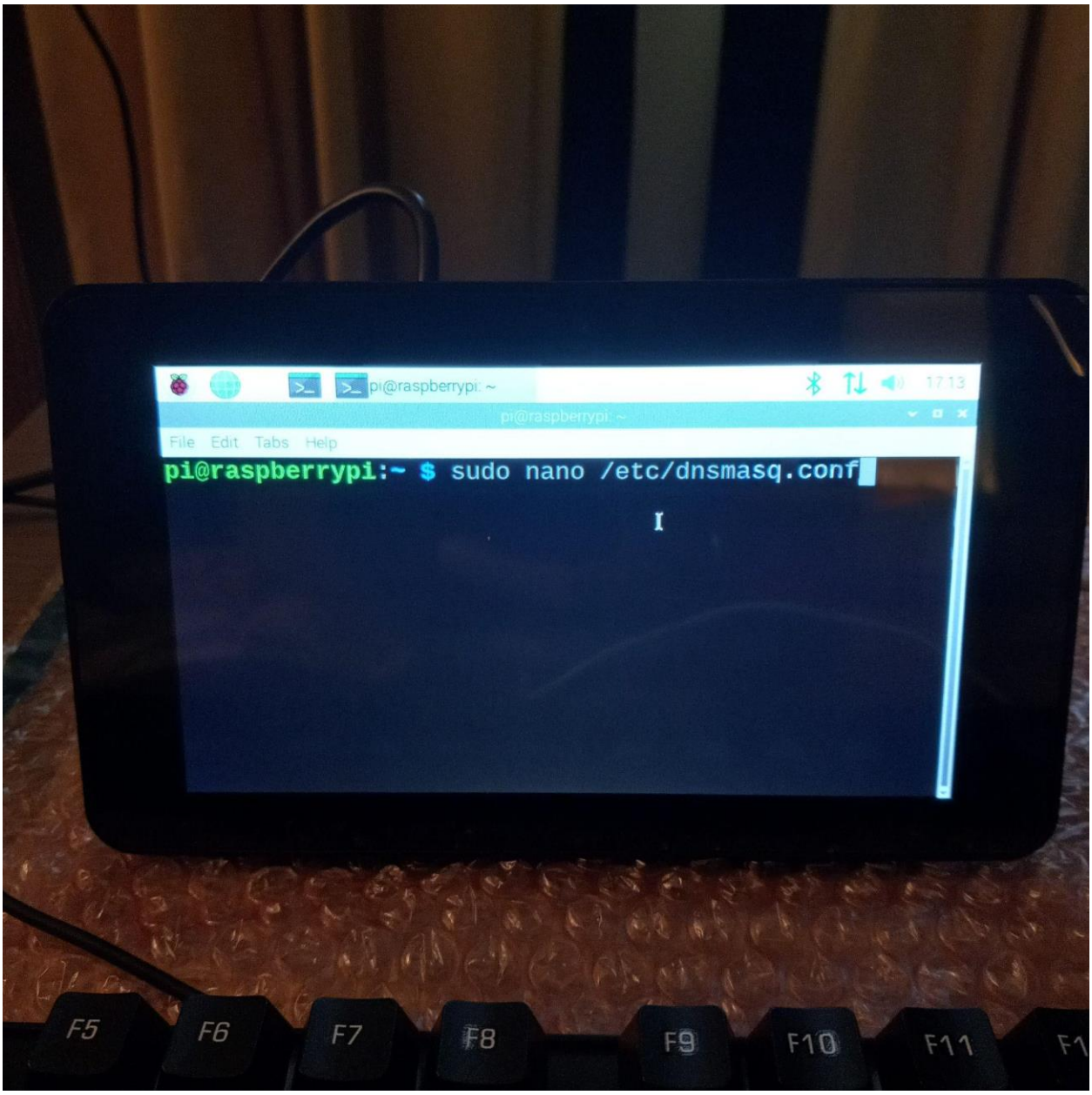
static domain_name_servers=192.168.1.1 8.8.8.8 - This line configures the DNS servers on the Raspberry Pi. DNS servers are responsible for transforming domain names into IP addresses. The first is our local router acting as a DNS server, and the second is Google's public DNS server



45. The next step is to reboot our raspberry pi to apply those changes I made, I run the command: **sudo reboot** to reboot our raspberry pi.



46. After rebooting the raspberry pi, the next step is to open and edit the dnsmasq.conf file, using the command: **sudo nano /etc/dnsmasq.conf**



47. I have added the following lines:

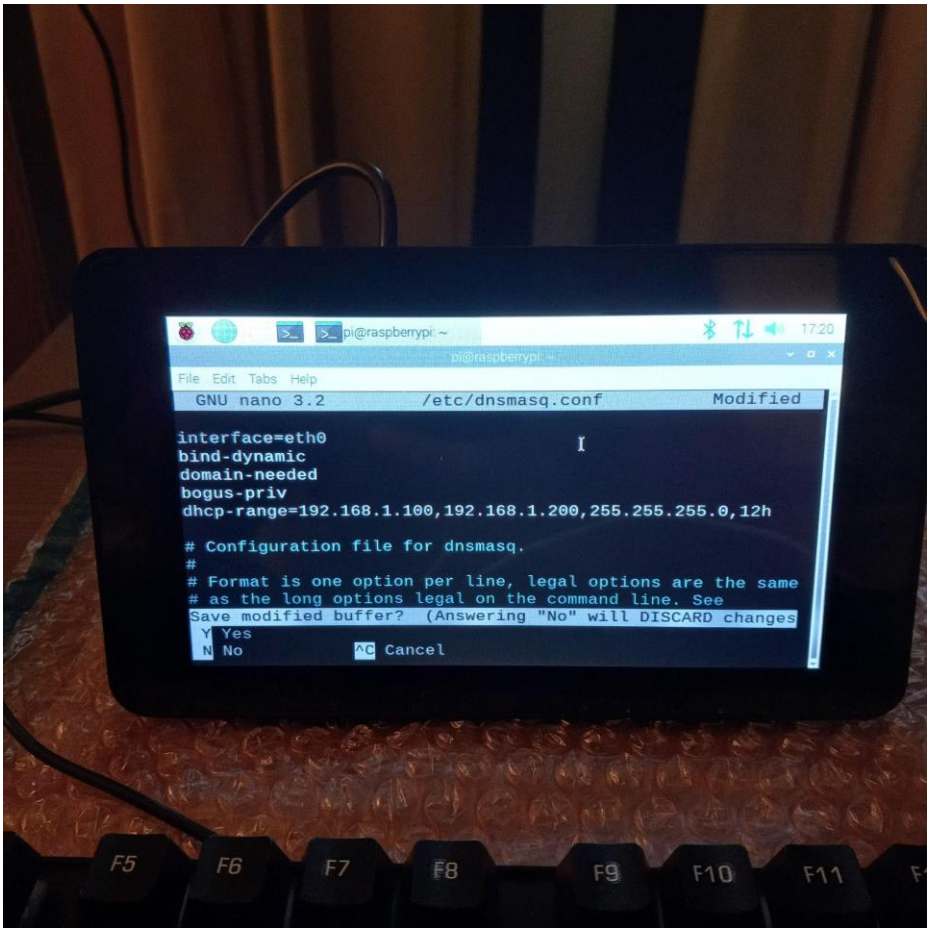
    interface=eth0 - This line defines the network interface dnsmasq will use to listen for DHCP requests, we can also use **wlan** for Wi-Fi.

bind-dynamic - This option instructs dnsmasq to associate dynamically allocated DHCP leases with the interface indicated by interface. It indicates that client IP addresses will be directly tied to the eth0 interface.

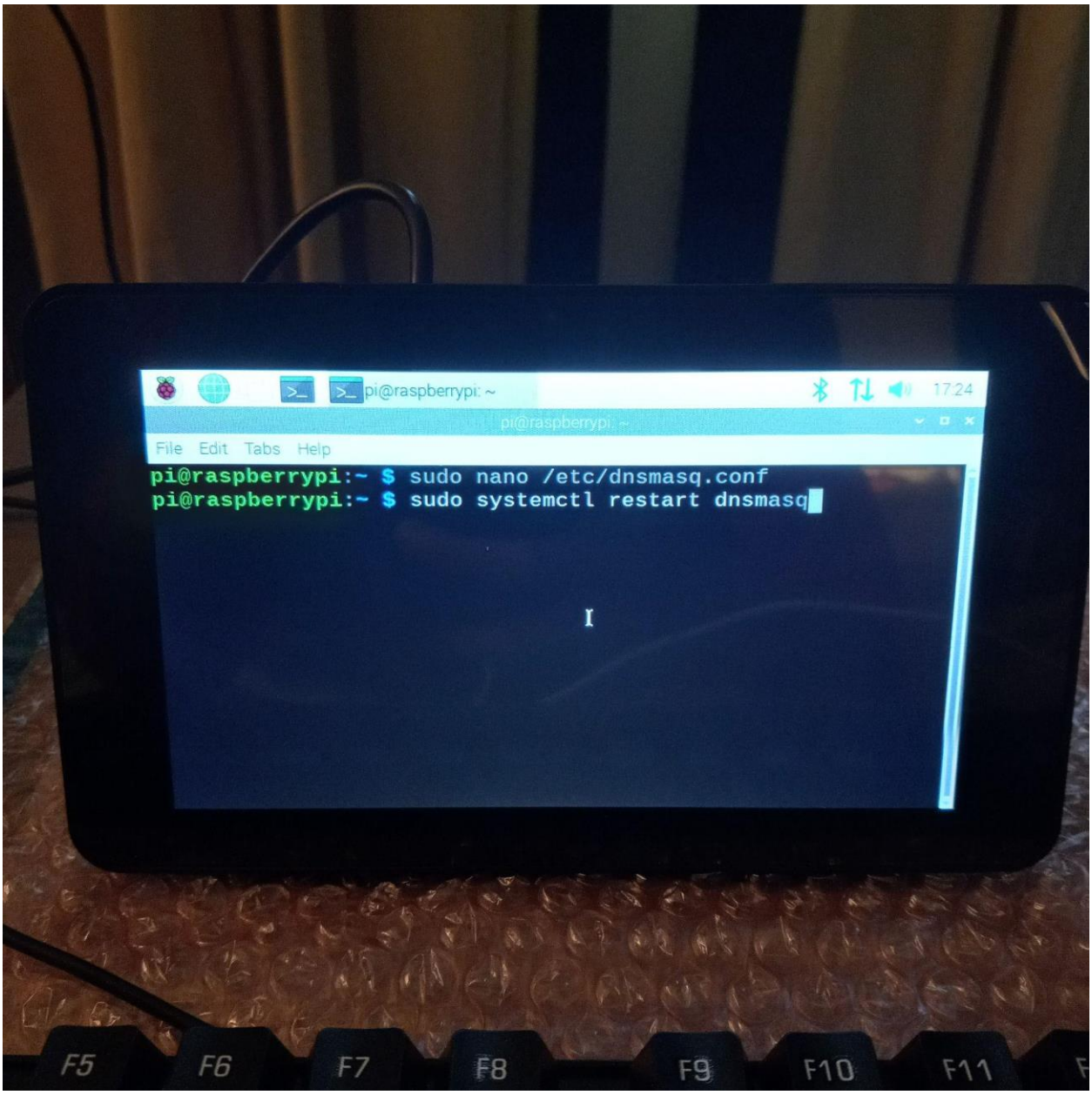
domain-needed - This option instructs dnsmasq not to transmit DNS requests for unresolved domains.

bogus-priv - This setting prevents requests for private IP ranges from being sent to upstream DNS servers.

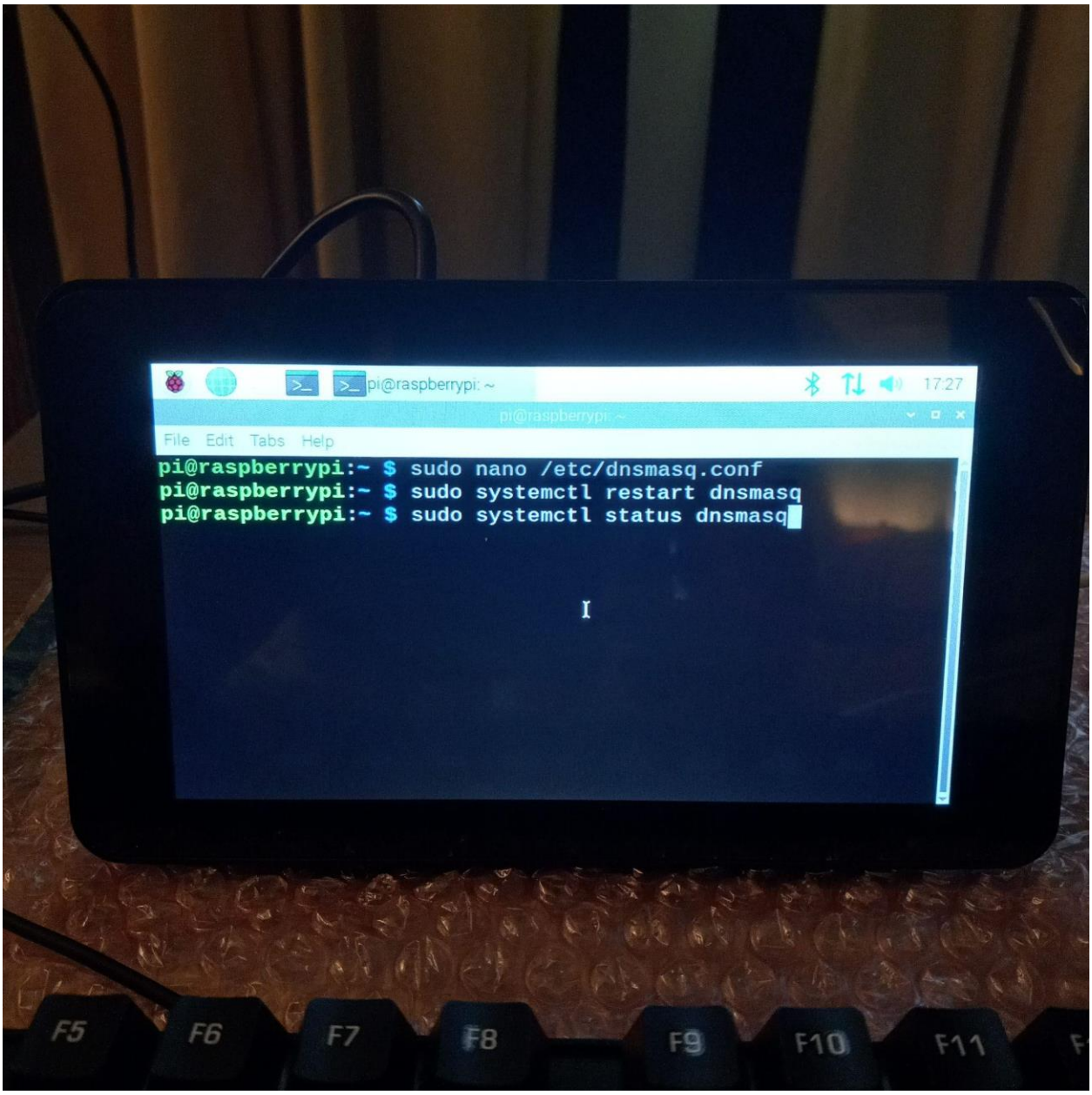
dhcp-range=192.168.1.100, 192.168.1.200, 255.255.255.0, 12h - This line specifies the IP address range that dnsmasq can issue to DHCP clients.



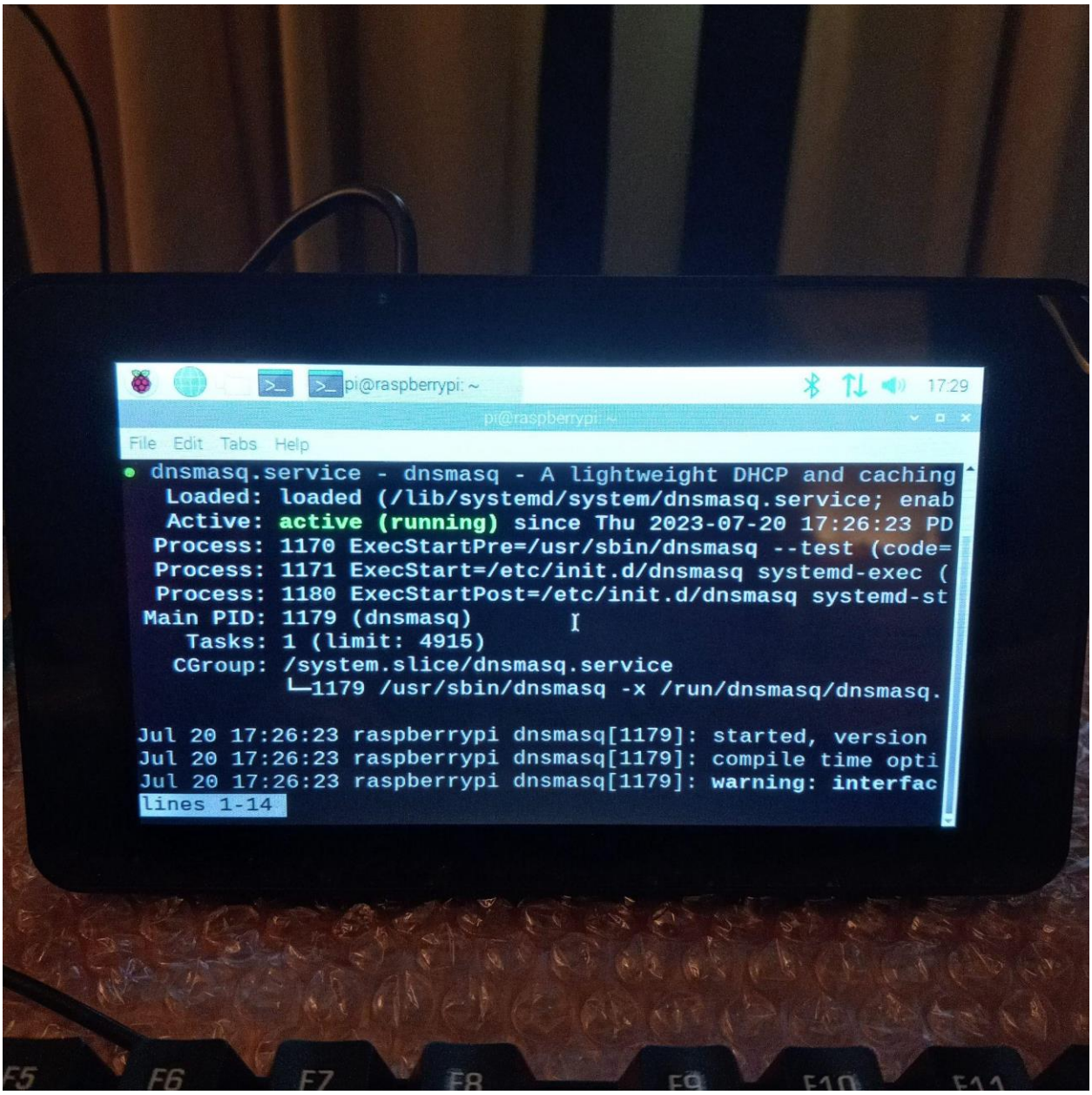
48. After saving those changes by pressing the keys **CTRL+X**, **y**, **enter**. The next step is to apply to those changes by restarting dnsmasq, by running the command: **sudo systemctl restart dnsmasq**.



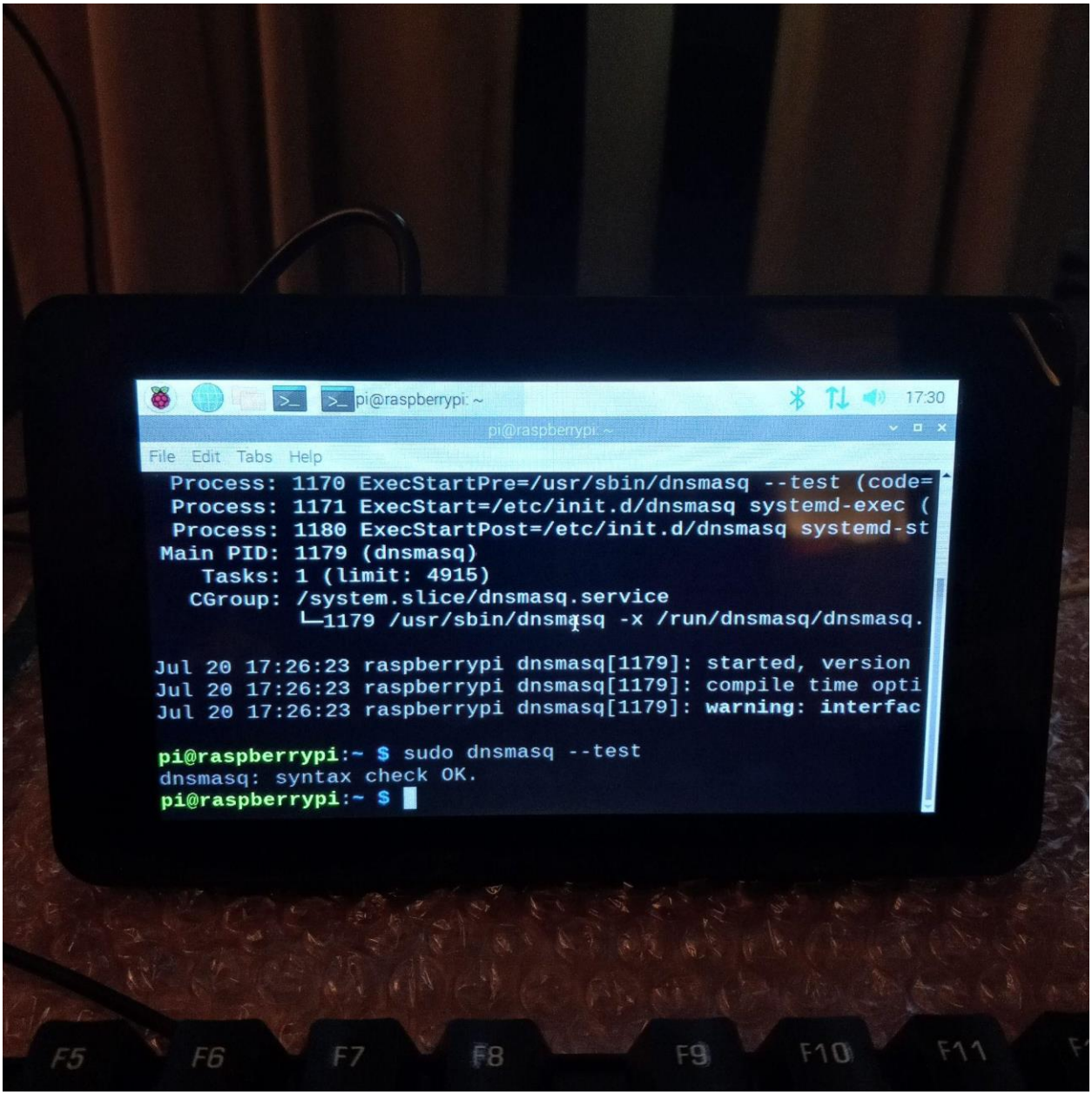
49. After restarting dnsmasq, the next step is to check its status and to do that we need to execute the command: **sudo systemctl status dnsmasq** and that will display its status.



50. After executing the previous command here, we can see that the dnsmasq is currently running and active, but to further verify we will run **4** more commands.

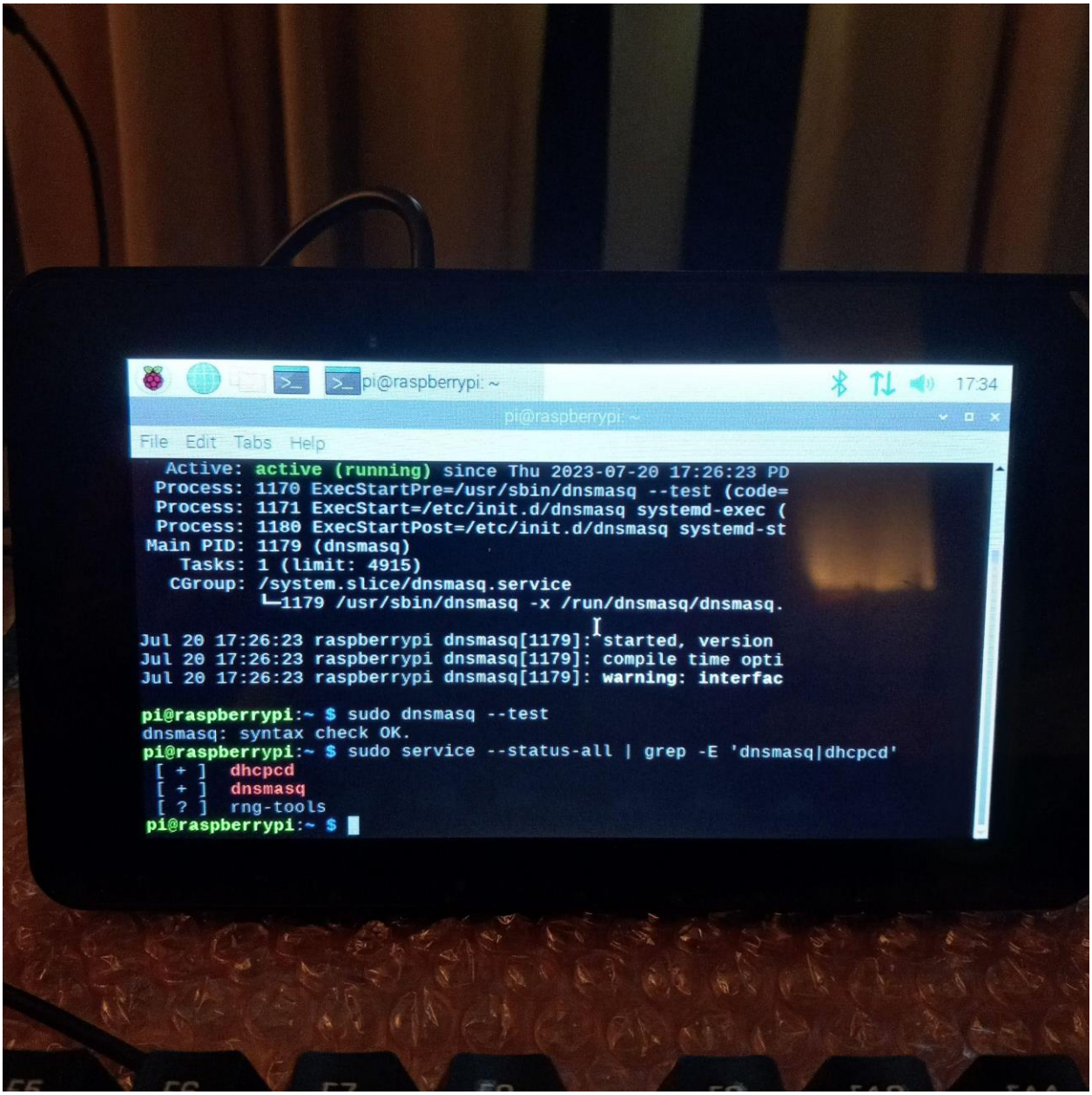


51. Next, we will execute the command: **sudo dnsmasq –test** This command tests the dnsmasq DHCP and DNS server. In test mode, dnsmasq will analyze the configuration file (dnsmasq.conf) and look for syntax mistakes or configuration problems.

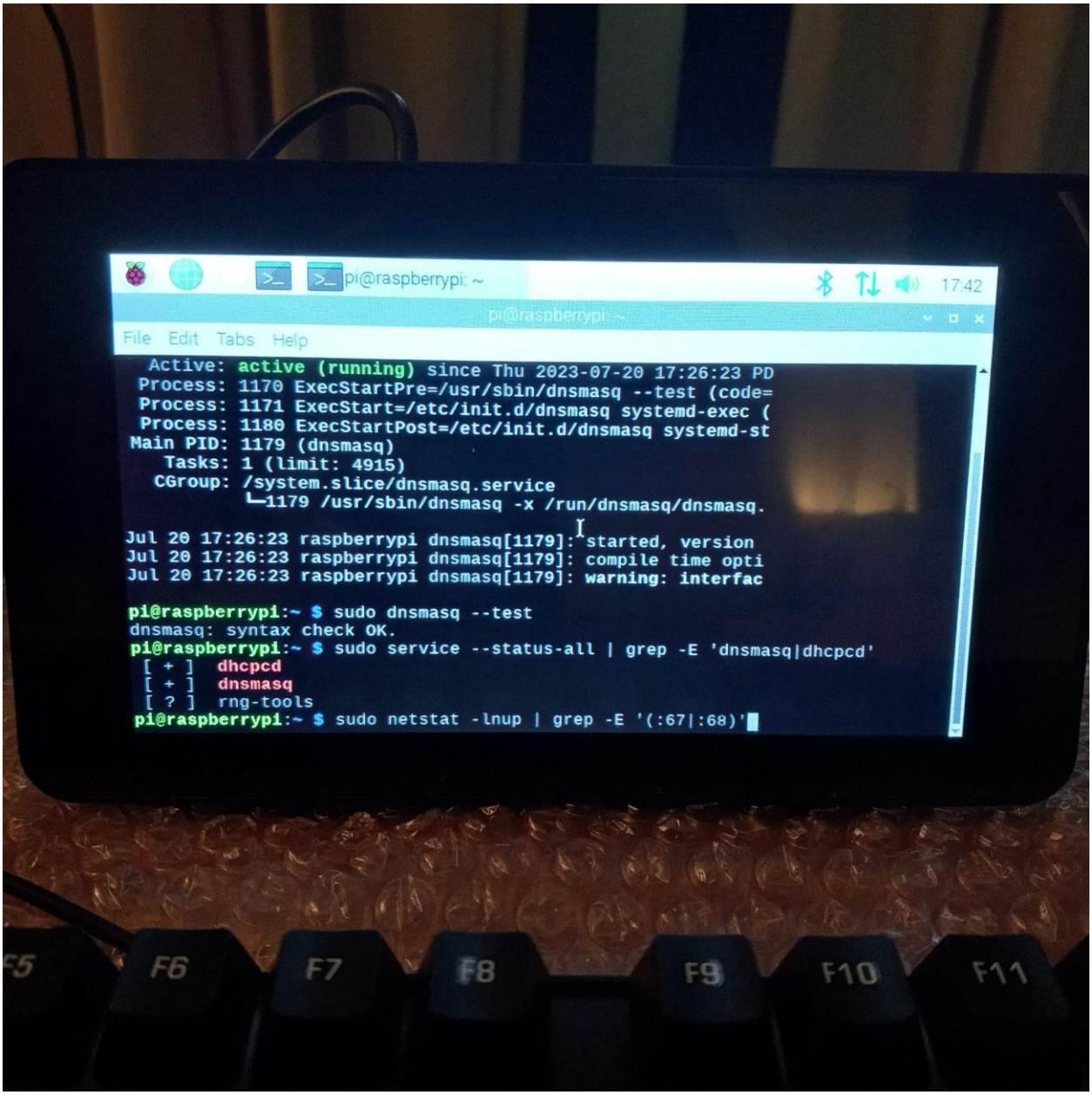


52. Next, we want to verify that our **dnsmasq** and **dhcpcd** services are up and running, so we will run the command: **sudo service –status-all | grep -E 'dnsmasq|dhcpcd.'** The command will display all services that

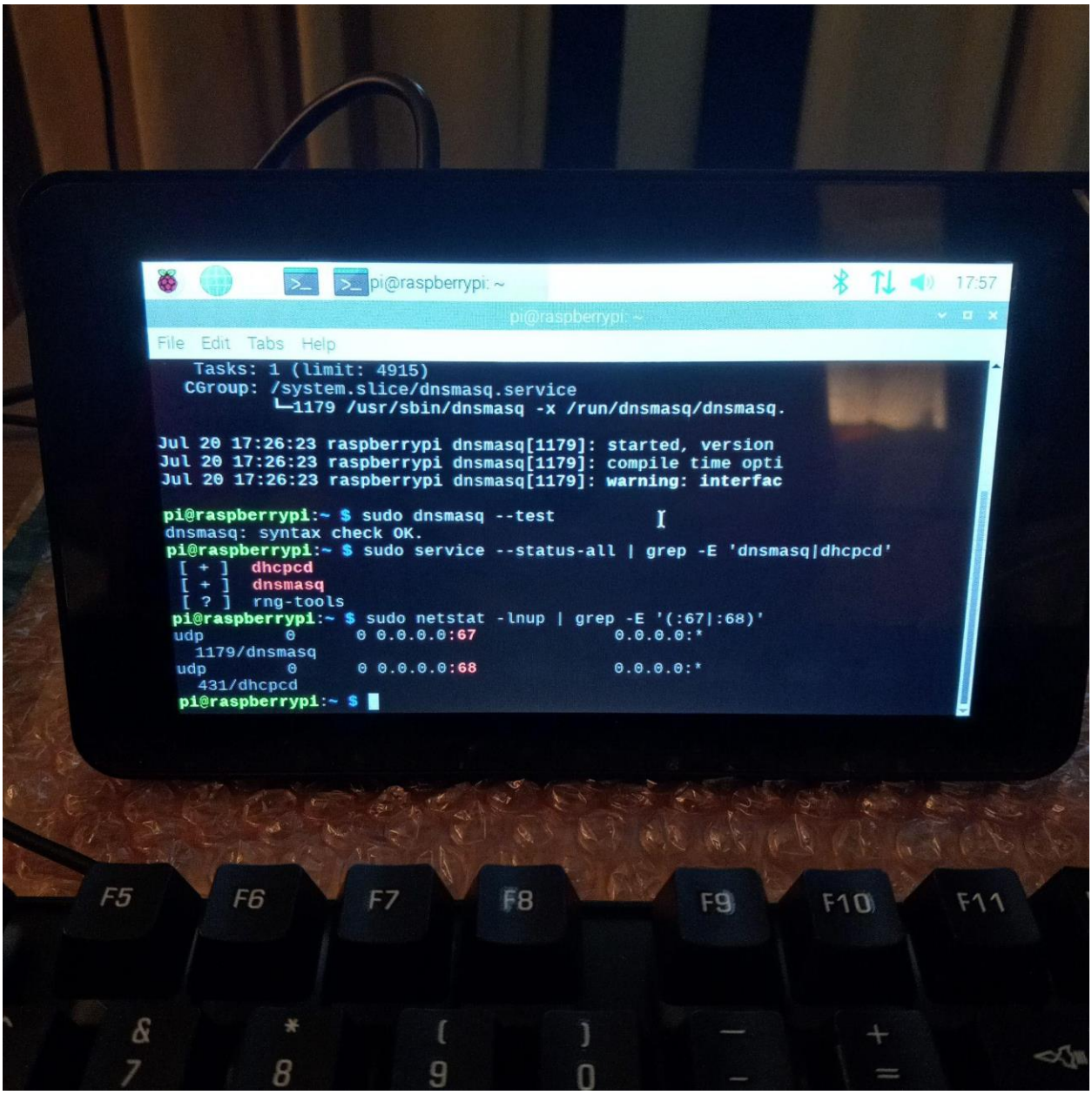
are currently running however we pipe that into a grep so we can search for dnsmasq and dhcpcd to see if they are running.



53. The last command we execute is: **sudo netstat -lnup | grep -E ‘(:67|:68)’** this command will check for active network sockets that are listening on UDP ports 67 and 68.



54. After running the command, it displays ports UDP 67 and 68 now that we can say that our raspberry pi is now successfully a DNS & DHCP server!


pi@raspberrypi: ~
17:57
pi@raspberrypi: ~
File Edit Tabs Help
Tasks: 1 (limit: 4915)
CGroup: /system.slice/dnsmasq.service
└─1179 /usr/sbin/dnsmasq -x /run/dnsmasq/dnsmasq.
Jul 20 17:26:23 raspberrypi dnsmasq[1179]: started, version
Jul 20 17:26:23 raspberrypi dnsmasq[1179]: compile time opti
Jul 20 17:26:23 raspberrypi dnsmasq[1179]: warning: interfac
pi@raspberrypi:~ $ sudo dnsmasq --test
dnsmasq: syntax check OK.
pi@raspberrypi:~ $ sudo service --status-all | grep -E 'dnsmasq|dhcpcd'
[+] dhcpcd
[+] dnsmasq
[?] rng-tools
pi@raspberrypi:~ $ sudo netstat -lnup | grep -E '(:67|:68)'
udp 0 0 0.0.0.0:67 0.0.0.0:*
1179/dnsmasq
udp 0 0 0.0.0.0:68 0.0.0.0:*
431/dhcpcd
pi@raspberrypi:~ $

References

Reference for DHCP: https://raspberrytips.com/dhcp-server-on-raspberry-pi/

Reference for DNS configuration: https://phoenixnap.com/kb/raspberry-pi-dns-server

Reference for Raspberry Pi hardware assembly: https://2securlee.com/read-think-learn

Special Thanks to

RCC Cybersecurity Club
RCC Cybersecurity Club Learning Center