

## Adding Tune Start API to RF2K-S

This modifies your amplifier software off of the mainstream release. You do this mod at your own risk. I have tested it and found no issues however I accept no responsibility if issues develop. You have been warned. The PgTg plug-in will still work if you choose not to install this modified rest server, you just won't be able to tune from SmartSDR.

If you do an update via the updater service in the RF2K-S, this mod will be overwritten and will have to be repeated.

A copy of the modified restServer.py module is included in this distribution zip file. It is from the latest RF-Power version (as of April 2026) which is GUI186. It appears to be backwards compatible as the only change in GUI186 was housekeeping and not impactful.

I strongly recommend you make a backup of your SDCARD before installing this modified rest server module. See K9UR's excellent YouTube video for instructions on how to do this. You will need to use VNC viewer to remote into the amp (you probably already have that). You will also need another SDCARD and a USB SDCARD reader to plug into the rear USB

<https://www.youtube.com/watch?v=R8rDE0ZrU6o>

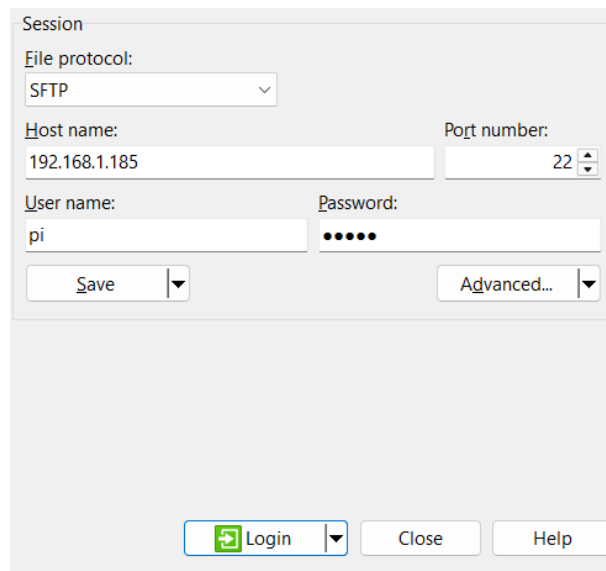
You also must make a backup copy of the restServer.py file on the SDCARD before making any changes to the real file. That is another level of backup you have, if something goes wrong, you can also copy the original file right back where it was. Two ways then to get this back if it goes sideways, one is to copy your original file back, the other is to put in your backup SDCARD. The installer will also place a copy of the restServer.py and this instruction file in the C:\ProgramData\PgTg\RfkitAmpTuner folder if you would prefer to install it from there instead of extracting it again, either way will work.

1. Extract the modified rest.Server.py file from this zip file to somewhere on your PC (desktop, documents, etc.) or plan on copying it from the hidden folder C:\ProgramData\PgTg\RfkitAmpTuner
2. Install WinSCP on your PC if you don't already have it.

<https://winscp.net/eng/download.php>

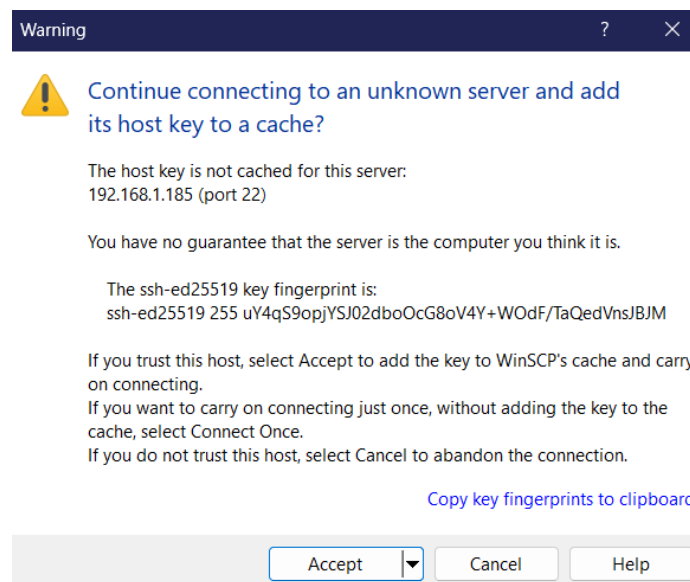
3. Run WinSCP.
4. Log into your amplifier with WinSCP. You will need the IP address of the amp, and the username is pi and the password is rfkit. Fill in the IP address of your

amp in the Host name field, pi in the User name field, rkit in the Password field and click Login



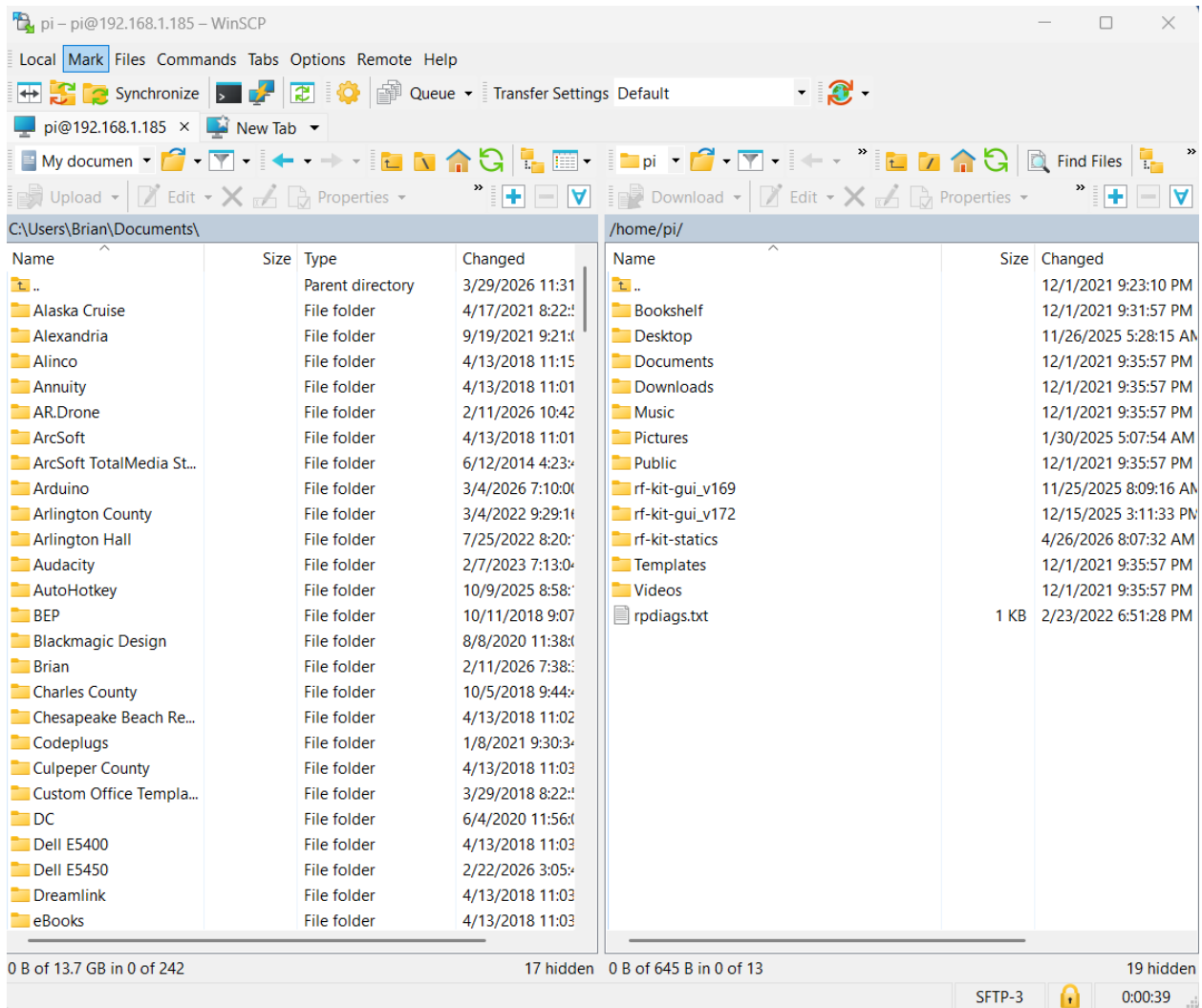
The image shows the WinSCP Session dialog box. It has a title bar 'Session'. Inside, there are fields for 'File protocol:' (set to SFTP), 'Host name:' (192.168.1.185), 'Port number:' (22), 'User name:' (pi), and 'Password:' (masked with dots). At the bottom, there are buttons for 'Save', 'Advanced...', 'Login' (with a green arrow icon), 'Close', and 'Help'.

5. The first time you login to your amp with WinSCP, you will see a warning about it being an unknown server and asking you to add its key to the cache. Click Accept to allow this, it is a one-time thing.



The image shows a 'Warning' dialog box with a yellow warning icon. The text inside says: 'Continue connecting to an unknown server and add its host key to a cache?'. It then states: 'The host key is not cached for this server: 192.168.1.185 (port 22)'. It continues: 'You have no guarantee that the server is the computer you think it is.' and 'The ssh-ed25519 key fingerprint is: ssh-ed25519 255 uY4qS9opjYSJ02dboOcG8oV4Y+WOf/TaQedVnsJBjM'. It provides instructions: 'If you trust this host, select Accept to add the key to WinSCP's cache and carry on connecting.', 'If you want to carry on connecting just once, without adding the key to the cache, select Connect Once.', and 'If you do not trust this host, select Cancel to abandon the connection.' There is a link 'Copy key fingerprints to clipboard'. At the bottom, there are buttons for 'Accept', 'Cancel', and 'Help'.

6. Now you will see the amp's /home/pi folder on the right column, and probably your documents folder on the left side.



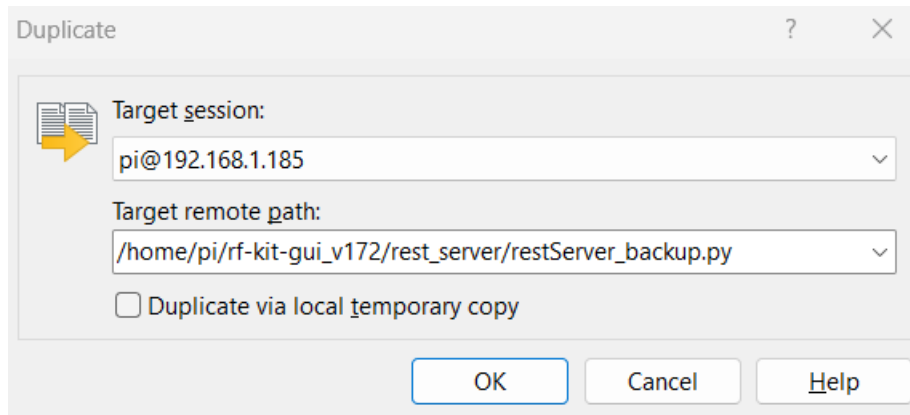
- On the right side, double click on the rf-kit-gui folder that has the highest version number, this will be the version your amp is running. In my case, it is rf-kit-gui-172.

/home/pi/rf-kit-gui_v172/		
Name	Size	Changed
..		4/23/2026 9:11:13 AM
__pycache__		11/25/2025 8:08:48
bottle		3/22/2026 3:19:23 PM
canBus		11/25/2025 8:08:48
debug		11/25/2025 8:08:47
hamlib		11/15/2020 5:49:17
keyboard		11/25/2025 8:08:48
main_screen		11/25/2025 8:08:48
menu		11/25/2025 8:08:48
network		11/25/2025 8:08:48
operational_interface		11/25/2025 8:08:48
resources		1/31/2025 12:12:14
rest_server		4/15/2026 1:42:24 PM
bootConfig.py	1 KB	4/12/2024 12:03:28
config.py	7 KB	5/7/2024 12:15:52 PM
contestDisplay.py	3 KB	4/12/2024 12:03:28
cursorSetting.py	2 KB	4/12/2024 12:03:28
customlogging.py	1 KB	5/7/2024 12:15:52 PM
data.py	14 KB	7/9/2025 12:19:47 PM
dimensions.py	1 KB	2/2/2025 11:14:45 AM
display.py	1 KB	8/29/2023 7:55:05 AM
displaySetting.py	2 KB	8/29/2023 7:55:05 AM
enums.py	1 KB	2/2/2025 3:04:01 PM
firmware251.bin	106 KB	7/6/2025 1:33:16 PM
interface.py	16 KB	6/4/2025 2:38:58 PM
localUpdateScreen.py	4 KB	6/4/2025 2:38:58 PM

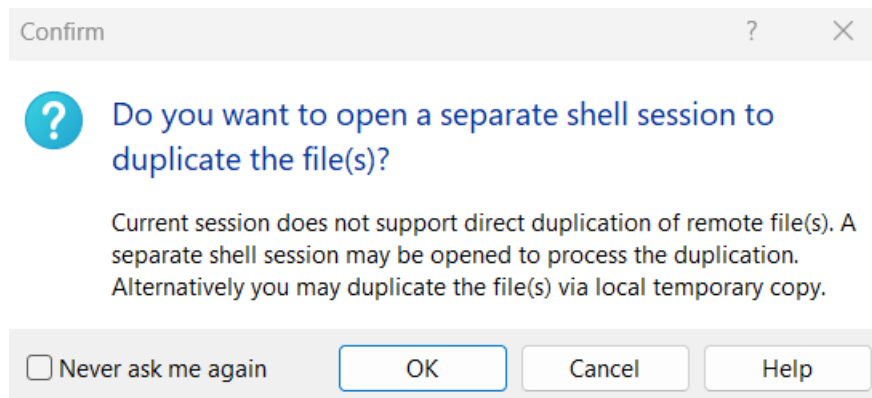
8. Now double-click on the rest\_server folder.

/home/pi/rf-kit-gui_v172/rest_server/		
Name	Size	Changed
..		12/15/2025 3:11:33 PM
__pycache__		4/23/2026 9:11:16 AM
restServer.py	16 KB	4/23/2026 9:10:54 AM

9. You will see your existing restServer.py module. Right-click on it, and click Duplicate. Change the name for the duplicate file (Target remote path) to restServer\_backup.py by putting your mouse cursor just after the r in Server and typing in \_backup so it looks like this, then click OK:



10. You will see a message about creating a separate shell to complete the file duplication, this is normal, click OK and you should see your duplicated file on the right side of WinSCP.

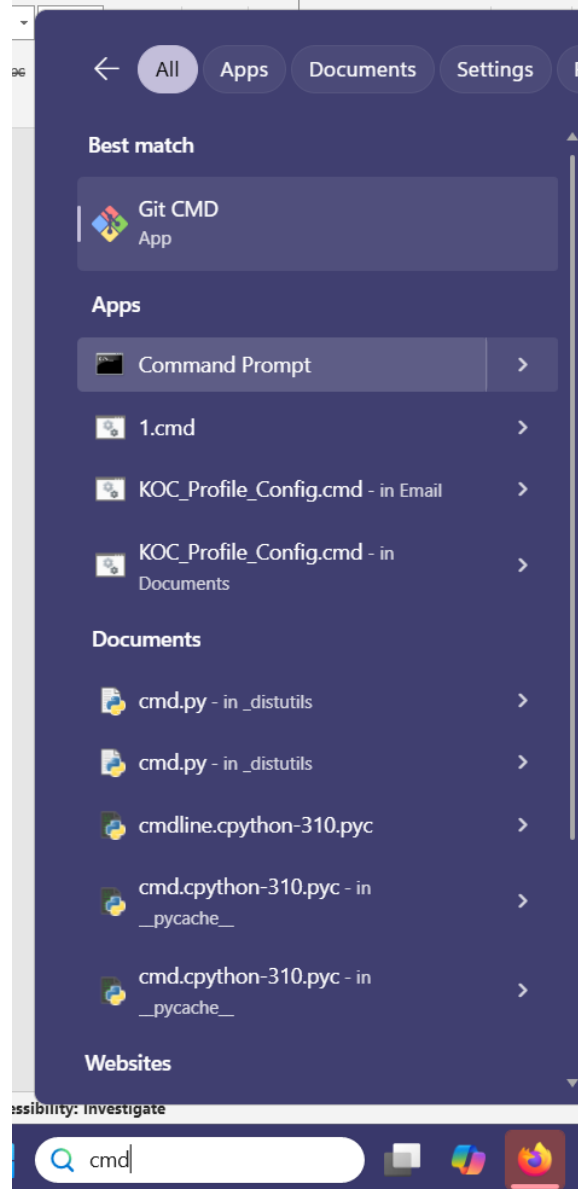


/home/pi/rf-kit-gui_v172/rest_server/		
Name	Size	Changed
..		12/15/2025 3:11:33 PM
__pycache__		4/23/2026 9:11:16 AM
restServer.py	16 KB	4/23/2026 9:10:54 AM
restServer_backup.py	13 KB	3/22/2026 8:59:42 AM

11. Now, on the left side of WinSCP, browse to wherever you extracted the modified restServer.py file that was included on the plugin distribution zip file (or browse to C:\ProgramData\PgTg\RfkitAmpTuner). Your location won't be the same as mine shown below, you want to browse to wherever you extracted that file. If you forgot that step, no problem, leave WinSCP open and find the zip file you downloaded and extract restServer.py to somewhere on your PC now, then browse to it on the left side of WinSCP.

C:\...\Visual Studio 2026\Projects\RfKitAmpTuner Plugin\Installers\Version 1.1.4\			
Name	Size	Type	Changed
..		Parent directory	4/26/2026 7:54:37 A
licdata.txt	1 KB	Text Document	4/8/2026 7:26:06 A
nsProcess.dll	5 KB	Application extensi...	6/28/2011 3:48:48 A
nsProcess.nsh	1 KB	NSIS Header File	6/28/2011 3:54:33 A
nsProcessW.dll	5 KB	Application extensi...	6/28/2011 3:48:44 A
PgTgBridge_RfKit_Plu...	100 KB	Application	4/26/2026 7:54:15 A
PgTgBridge_RfKit_Plu...	88 KB	Compressed (zipp...	4/26/2026 7:54:37 A
restServer.py	14 KB	Python File	4/26/2026 6:35:34 A
RfKit PgTg Plugin.nsi	5 KB	NSI File	4/26/2026 6:36:33 A
RfKitAmpTuner read...	7 KB	Text Document	4/26/2026 6:51:33 A
RfKitAmpTuner.dll	106 KB	Application extensi...	4/26/2026 7:52:25 A

12. Once you see the file on the left, you want to drag it with your mouse from the left pane of WinSCP (local) to the right pane of WinSCP (remote). The right pane should still be at /home/pi/rf-kit-gui-172/rest\_server/ (the version number will match your amp; I have 172 on mine).
13. That should complete the installation of the modified rest server module.
14. Reboot the amp so the it starts to use the modified rest server.
15. You can test the rest server using the Windows CURL command, from a command window.
16. In Windows, search for cmd and press Enter to open a command window.



17. For the following curl commands, it is best to copy and paste them from this document into a command window. The syntax has to be exact and is unforgiving of typing mistakes. Also be sure to substitute the IP address of YOUR amplifier into these commands. The port is always 8080.
18. Copy and paste in this command to verify you can talk to the amp, and you should get a response with your device name and software version. If you don't, figure out why you can't reach your amp:

```
curl -X GET http://192.168.1.185:8080/info
```

```
Command Prompt
Microsoft Windows [Version 10.0.26200.8246]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Brian>curl -X GET http://192.168.1.185:8080/info
{"device": "RF2K-S", "software_version": {"GUI": 172, "controller": 251}, "custom_device_name": "N30C"}
C:\Users\Brian>
```

19. Now try a curl command to the new REST API endpoint that is in the modified file you installed. If you are not transmitting a tune level carrier when you type this in, the tune won't actually occur and you will get a message that shows the current tuning values:

```
curl -X PUT -H "Accept: application/json" -H "Content-Type: application/json" -d '{"tuner_mode": "START"}' http://192.168.1.185:8080/tuner
```

```
C:\Users\Brian>curl -X PUT -H "Accept: application/json" -H "Content-Type: application/json" -d '{"tuner_mode": "START"}' http://192.168.1.185:8080/tuner
{"mode": "AUTO", "setup": "CL", "L": {"value": 310, "unit": "nH"}, "C": {"value": 530, "unit": "pF"}, "tuned_frequency": {"value": 7041, "unit": "kHz"}, "segment_size": {"value": 25, "unit": "kHz"}}
C:\Users\Brian>
```

20. When the tuner is actually tuning (tuner enabled on that band and tune carrier present, the response to the curl command will look like this (AUTO\_TUNING\_FROM\_AUTO):

```
C:\Users\Brian>curl -X PUT -H "Accept: application/json" -H "Content-Type: application/json" -d '{"tuner_mode": "START"}' http://192.168.1.185:8080/tuner
{"mode": "AUTO_TUNING_FROM_AUTO", "setup": "CL", "L": {"value": 310, "unit": "nH"}, "C": {"value": 551, "unit": "pF"}, "tuned_frequency": {"value": 7041, "unit": "kHz"}, "segment_size": {"value": 25, "unit": "kHz"}}
C:\Users\Brian>
```

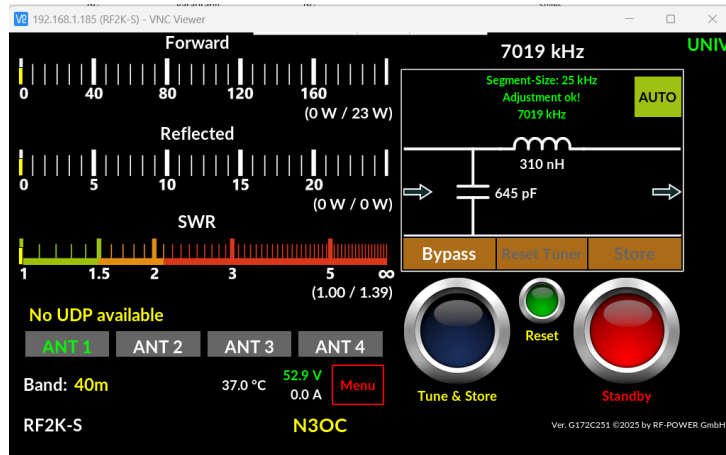
21. It is not necessary to go through all these curl commands with an actual tuning cycle, the important one is step 19 where you hit the tuner endpoint with an HTTP PUT, if you get an answer to that you should be good, you can re-boot the amp and restart PgTg and you should be good. Once it all is loaded and settles down, try a tune cycle from SmartSDR.

If you prefer not to use WinSCP, you can modify your existing restServer.py module right on the amp using tools that are already loaded into the Raspberry Pi.

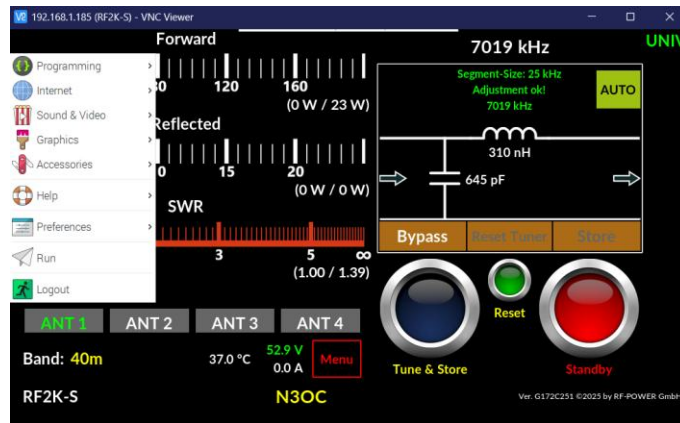
You need to be able to make a VNC connection from your PC to the RF2K-S amp in order to do this, and you will make all the changes from that screen. Alternatively, you can use WinSCP to connect to the amp file system and manipulate the files that way.

1. Connect to the RF2K-S with VNC viewer. You should see the normal amplifier screen.

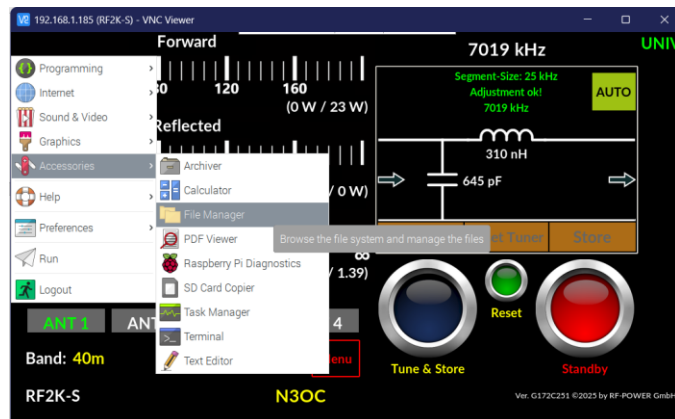




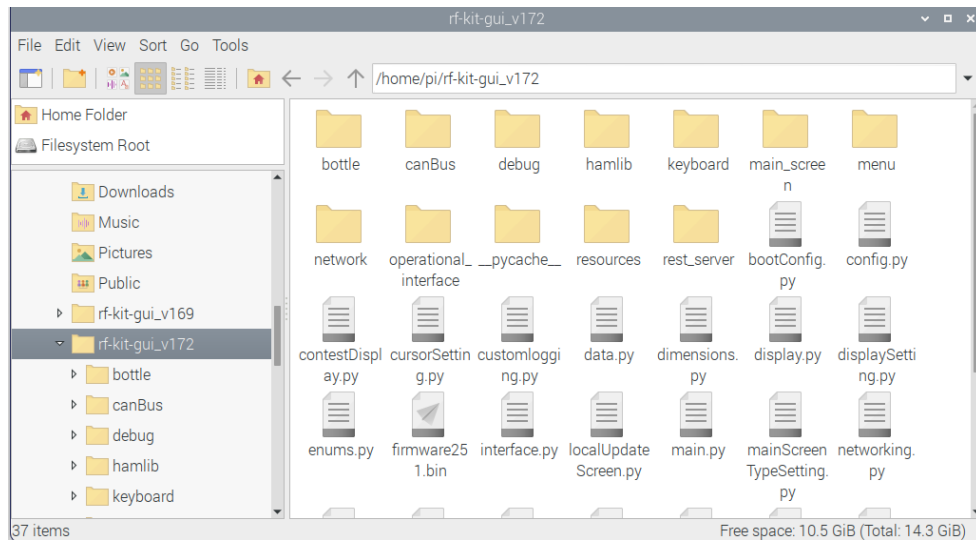
2. Press CTRL-ESC on your PC to get to the raspberry pi programs menu.



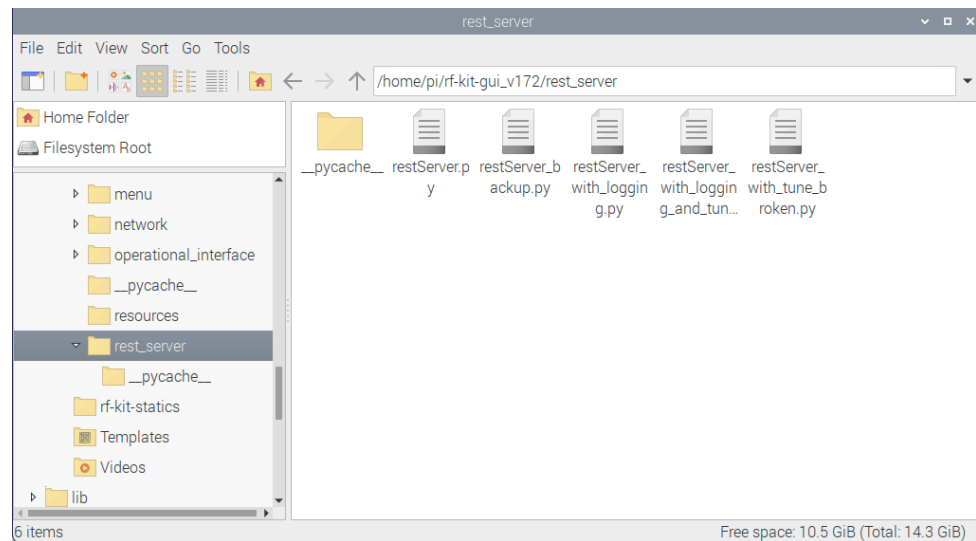
3. Click on Accessories, then File Manager. The File Manager will open on top of the amplifier screen.



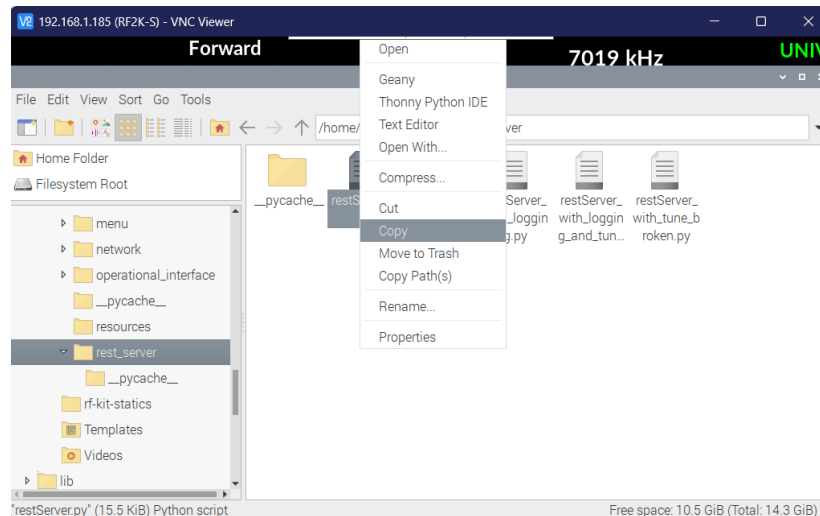
4. Double-click on the folder that matches the software version of your RF2K-S. This is usually the one with the higher number. Mine is rf-kit-gui-v172. This will take you inside the folder where the currently running version is kept.



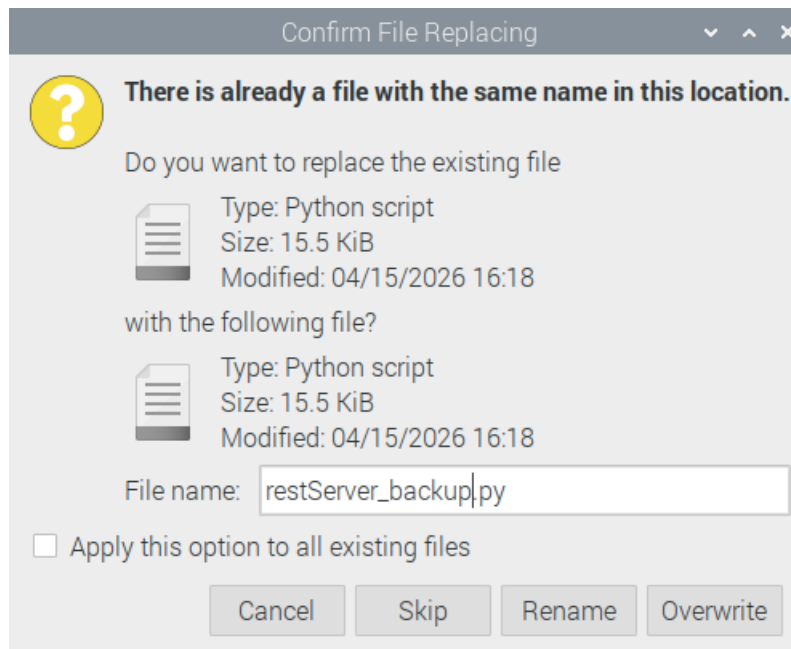
5. Double-click on the rest-server folder. This is the folder where the file we need to change is located.



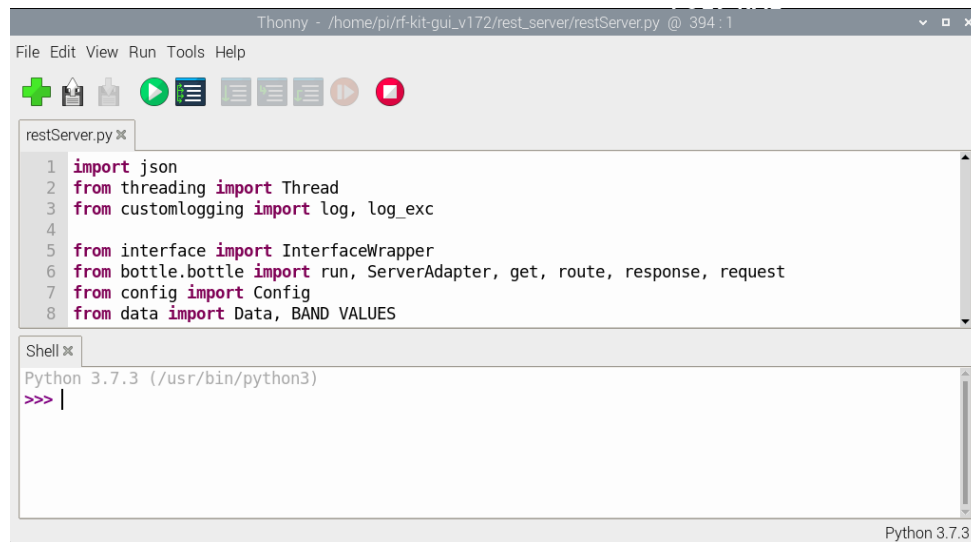
6. Right-click on the restServer.py file and click Copy.



7. Now right-click on a blank area in the file manager and click Paste. You will get a warning message that the file name already exists. Change the file name to `restServer_backup.py` and click Rename.



8. You will now see both the original file and the backup in the file manager. If anything goes wrong, you can rename the backup file back to `restServer.py` and be right back where you started.
9. Right-click the `restServer.py` file and click Thonny Python IDE to open the file for editing.



Thonny - /home/pi/rf-kit-gui\_v172/rest\_server/restServer.py @ 394 : 1

File Edit View Run Tools Help

restServer.py

```
1 import json
2 from threading import Thread
3 from customlogging import log, log_exc
4
5 from interface import InterfaceWrapper
6 from bottle.bottle import run, ServerAdapter, get, route, response, request
7 from config import Config
8 from data import Data, BAND_VALUES
```

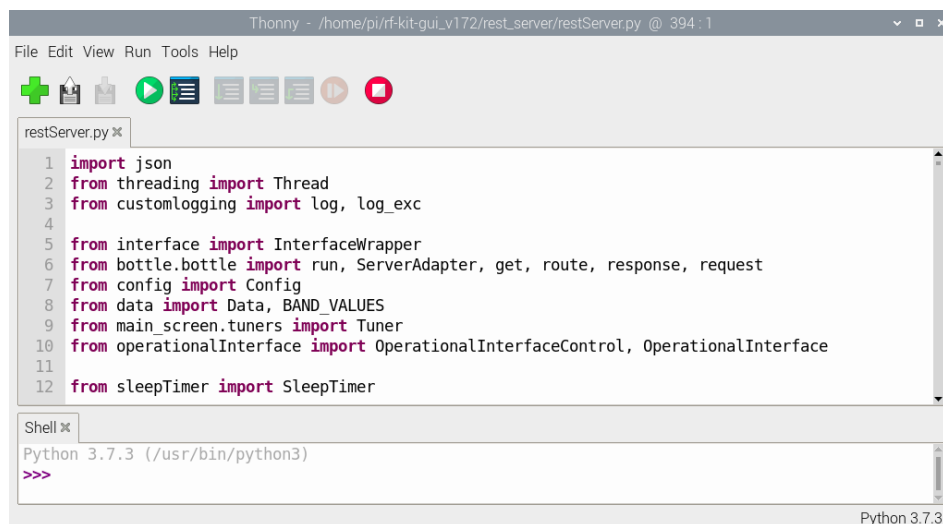
Shell

Python 3.7.3 (/usr/bin/python3)

```
>>> |
```

Python 3.7.3

10. We are going to work in the top box only, so if you want you can scroll the bottom of the top window down to make it larger.



Thonny - /home/pi/rf-kit-gui\_v172/rest\_server/restServer.py @ 394 : 1

File Edit View Run Tools Help

restServer.py

```
1 import json
2 from threading import Thread
3 from customlogging import log, log_exc
4
5 from interface import InterfaceWrapper
6 from bottle.bottle import run, ServerAdapter, get, route, response, request
7 from config import Config
8 from data import Data, BAND_VALUES
9 from main_screen.tuners import Tuner
10 from operationalInterface import OperationalInterfaceControl, OperationalInterface
11
12 from sleepTimer import SleepTimer
```

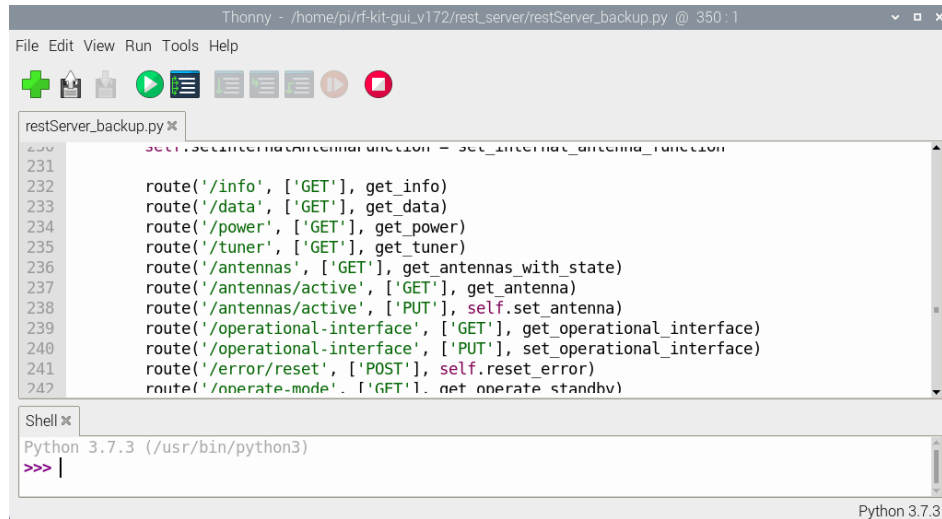
Shell

Python 3.7.3 (/usr/bin/python3)

```
>>>
```

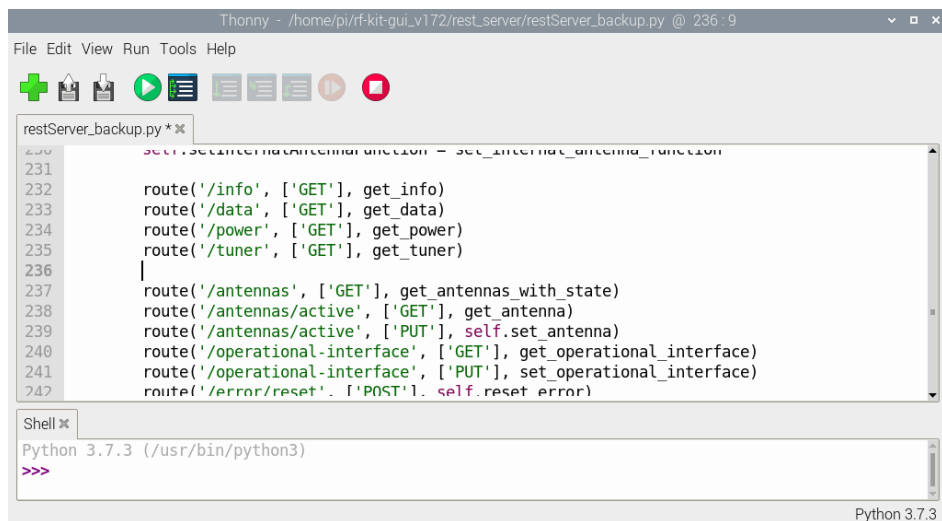
Python 3.7.3

11. Scroll down to approximately line 232 where the REST server routes (endpoints) are listed. Look for the one that says tuner, mine is on line 235.



```
Thonny - /home/pi/rf-kit-gui_v172/rest_server/restServer_backup.py @ 350 : 1
File Edit View Run Tools Help
restServer_backup.py
230 self.set_antenna_antenna_function = set_antenna_antenna_function
231
232 route('/info', ['GET'], get_info)
233 route('/data', ['GET'], get_data)
234 route('/power', ['GET'], get_power)
235 route('/tuner', ['GET'], get_tuner)
236
237 route('/antennas', ['GET'], get_antennas_with_state)
238 route('/antennas/active', ['GET'], get_antenna)
239 route('/antennas/active', ['PUT'], self.set_antenna)
240 route('/operational-interface', ['GET'], get_operational_interface)
241 route('/operational-interface', ['PUT'], set_operational_interface)
242 route('/error/reset', ['POST'], self.reset_error)
243 route('/operate-mode', ['GET'], get_operate_standby)
244
Shell
Python 3.7.3 (/usr/bin/python3)
>>>
```

12. Click at the end of line 235 (the one that says `route('/tuner', ['GET'], get_tuner)` to position your cursor at the end of that line and press Enter to start a new line.



```
Thonny - /home/pi/rf-kit-gui_v172/rest_server/restServer_backup.py @ 236 : 9
File Edit View Run Tools Help
restServer_backup.py
230 self.set_antenna_antenna_function = set_antenna_antenna_function
231
232 route('/info', ['GET'], get_info)
233 route('/data', ['GET'], get_data)
234 route('/power', ['GET'], get_power)
235 route('/tuner', ['GET'], get_tuner)
236
237 route('/antennas', ['GET'], get_antennas_with_state)
238 route('/antennas/active', ['GET'], get_antenna)
239 route('/antennas/active', ['PUT'], self.set_antenna)
240 route('/operational-interface', ['GET'], get_operational_interface)
241 route('/operational-interface', ['PUT'], set_operational_interface)
242 route('/error/reset', ['POST'], self.reset_error)
243
Shell
Python 3.7.3 (/usr/bin/python3)
>>>
```

13. Paste (or type) this line in to add a new REST API endpoint:

```
route('/tuner', ['PUT'], set_tuner_start
```

```

Thonny - /home/pi/rf-kit-gui_v172/rest_server/restServer.py @ 394:1
File Edit View Run Tools Help

restServer.py
265 self.antennaSetAllowedCheck = antenna_change_allowed_check
266 self.setInternalAntennaFunction = set_internal_antenna_function
267
268 route('/info', ['GET'], get_info)
269 route('/data', ['GET'], get_data)
270 route('/power', ['GET'], get_power)
271 route('/tuner', ['GET'], get_tuner)
272 route('/tuner', ['PUT'], set_tuner_start)
273 route('/antennas', ['GET'], get_antennas_with_state)
274 route('/antennas/active', ['GET'], get_antenna)
275 route('/antennas/active', ['PUT'], self.set_antenna)
276 route('/operational-interface', ['GET'], get_operational_interface)

Shell
Python 3.7.3 (/usr/bin/python3)
>>>
Python 3.7.3

```

14. Now we have to add the set\_tuner\_start function and we are done.

15. Scroll to approximately line 184 and look for the def get\_tuner function and scroll to the blank line at the end of that function, around line 214 and put your cursor there, and press Enter to add a blank line and move your cursor up one line with the up arrow, so your cursor is in the middle of 3 blank lines. Mine ended up at line 214, yours may be there or at least close to that.

```

Thonny - /home/pi/rf-kit-gui_v172/rest_server/restServer_backup.py @ 214:1
File Edit View Run Tools Help

restServer_backup.py
208 "C": value_unit(Data.getInstance().C, "pF"),
209 "tuned_frequency": value_unit(Data.getInstance().f_t // 1000, "kHz"),
210 "segment_size": value_unit(Data.getInstance().segmentSize, "kHz")
211 }
212 return tuner
213
214
215
216 def get_operate_standby():
217     operate_mode = "OPERATE" if Data.getInstance().inOperate else "STANDBY"
218     return {
219         "operate_mode": operate_mode

```

16. Copy the following function in and paste it to that location:

```

def set_tuner_start():
    handle_sleep()
    request_json = request.json
    tuner_mode = get_tuner().get("mode")
    if tuner_mode == "OFF" or tuner_mode == "BYPASS" or tuner_mode
    == "false":
        return '{"result": "' + tuner_mode + '"}'
    tuner_cmd = request_json.get("tuner_mode") if request_json is not
    None else None
    if tuner_cmd == "START":

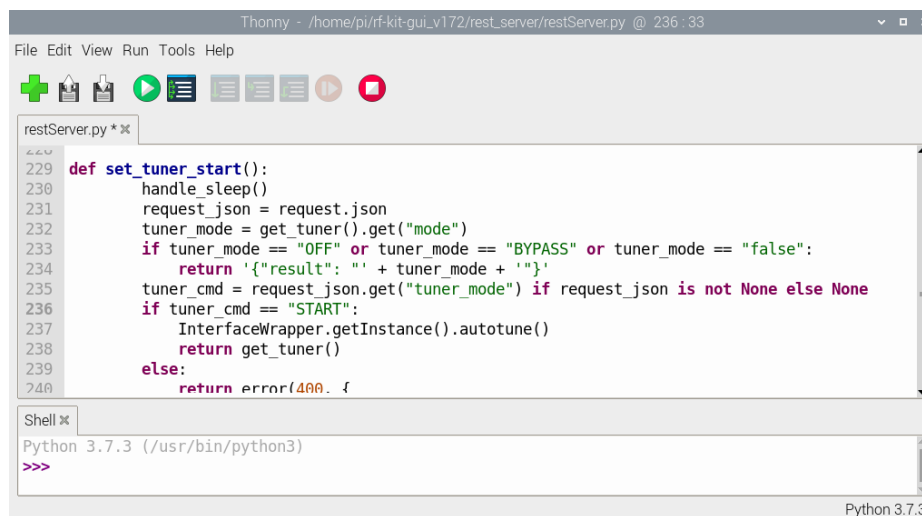
```

```

        InterfaceWrapper.getInstance().start_tuning()
        return get_tuner()
    else:
        return error(400, {
            "status": "400 Bad Request",
            "reason": "JSON request body with key 'tuner_mode' and value
of 'START' required"
        })

```

17. It should look like this when done. The indentation IS important. (Ignore the extra logger line I have in mine.)



The screenshot shows the Thonny IDE interface. The top bar indicates the file path: `Thonny - /home/pi/rf-kit-gui_v172/rest_server/restServer.py @ 236 : 33`. Below the menu bar (File, Edit, View, Run, Tools, Help) is a toolbar with icons for file operations and execution. The main editor window displays the `restServer.py` file with the following code:

```

229 def set_tuner_start():
230     handle_sleep()
231     request_json = request.json
232     tuner_mode = get_tuner().get("mode")
233     if tuner_mode == "OFF" or tuner_mode == "BYPASS" or tuner_mode == "false":
234         return '{"result": "' + tuner_mode + '"}'
235     tuner_cmd = request_json.get("tuner_mode") if request_json is not None else None
236     if tuner_cmd == "START":
237         InterfaceWrapper.getInstance().autotune()
238         return get_tuner()
239     else:
240         return error(400, {

```

Below the editor is a shell window titled "Shell" showing the Python 3.7.3 prompt:

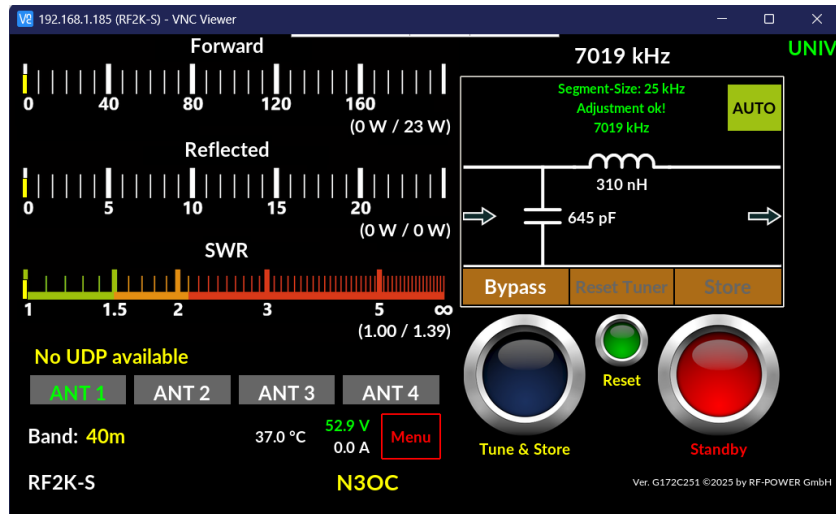
```

Python 3.7.3 (/usr/bin/python3)
>>>

```

The status bar at the bottom right indicates "Python 3.7.3".

18. Click the floppy disk icon with the down arrow (the one on the right) to save the file. Close Thonny IDE. Close the File Manager. You should now see the amp main screen.



19. Press CTRL-ESC again and click on Logout, then Reboot to reboot the amp.

This will put your changes into effect.

20. You can test the result with a CURL command in a command prompt, changing my IP address to the IP address of your amp:

```
curl -X PUT -H "Accept: application/json" -H "Content-Type: application/json" -d
{"tuner_mode\" : \"START\"}" "http://192.168.1.185:8080/tuner"
```

21. If the curl command gets a valid response, you should be good. If you see an error, you may have done something wrong. The amp won't actually tune with a curl command unless you are transmitting a carrier with tune level power, then you should hear the tuning cycle start.

22. You can also use WinSCP to move the file off of the amp to your PC, edit it there, and transfer it back if you have difficulty editing it right on the amp due to the reduced screen size.