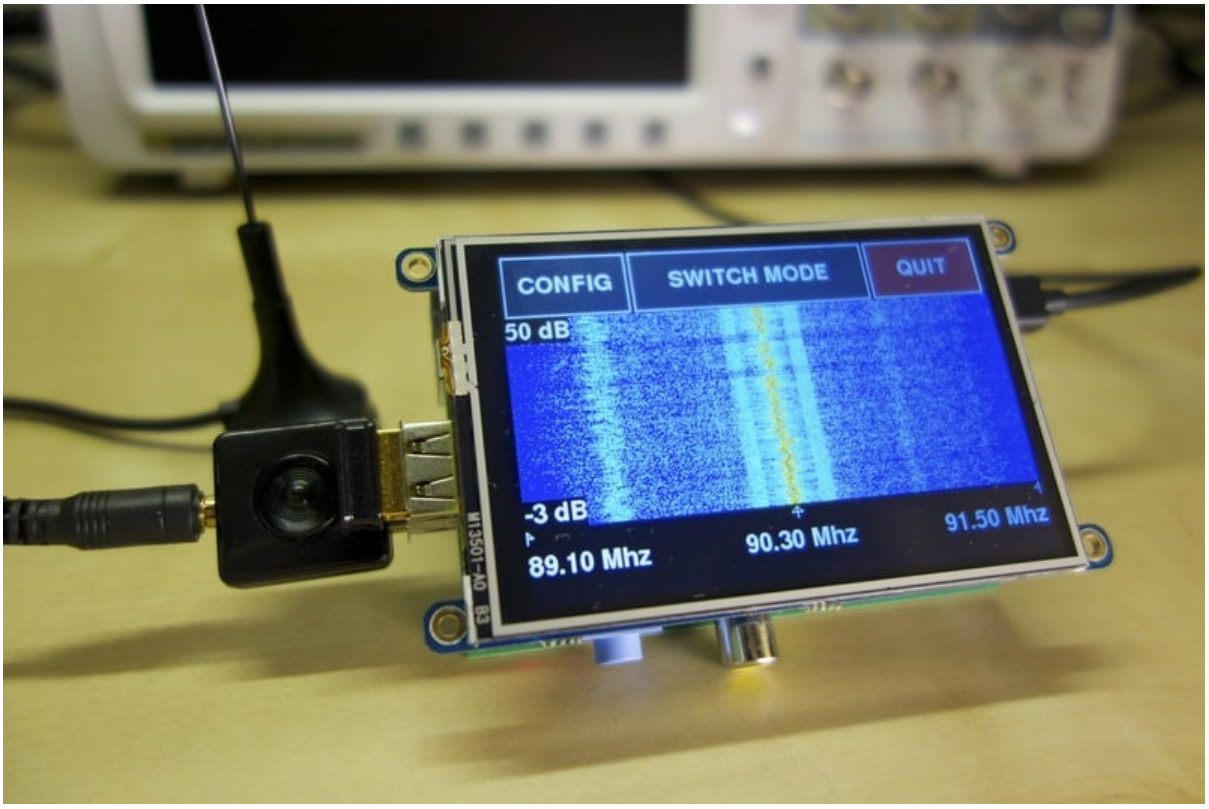
Freq Show: Raspberry Pi RTL-SDR Scanner

Created by Tony DiCola

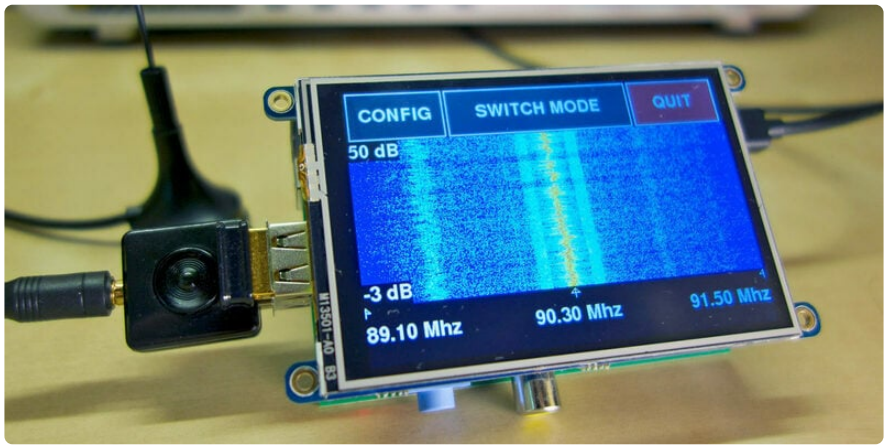


https://learn.adafruit.com/freq-show-raspberry-pi-rtl-sdr-scanner

Last updated on 2025-03-07 02:49:19 PM EST

Table of Contents

Overview



Have you ever wondered what's in the radio waves zipping invisibly around you every day? Software-defined radio (SDR) (https://adafru.it/e1Y) is a great tool to explore radio signals using a computer and inexpensive radio tuner. With SDR you can examine many radio signals such as FM radio (https://adafru.it/e1Z), television (https://adafru.it/e20), emergency & weather radio (https://adafru.it/e21), citizen band (CB) (https://adafru.it/e22), wireless protocols and much more (https://adafru.it/e23).

Although dedicated SDR hardware like the HackRF (https://adafru.it/e24) allow you to tune an immense range of the radio spectrum, you can easily get started with SDR using a Raspberry Pi and inexpensive RTL-SDR tuner (http://adafru.it/1497). Inspired by the HackRF PortaPack (https://adafru.it/e26), this project will show you how to build a small portable SDR scanner using a Raspberry Pi, PiTFT (http://adafru.it/2097), and RTL-SDR (http://adafru.it/1497) radio dongle. With the Raspberry Pi Freq Show RTL-SDR scanner you can visualize the invisible world of radio!

UNDERSTAND: FREQSHOW DOES NOT PLAY AUDIO. It graphs the frequency and amplitude of RF signals, which can be useful for troubleshooting and developing a basic understanding of radio and wireless protocols, but it does not decode nor demodulate them.

Before you get started it will help to familiarize yourself with a few other guides for more background information:

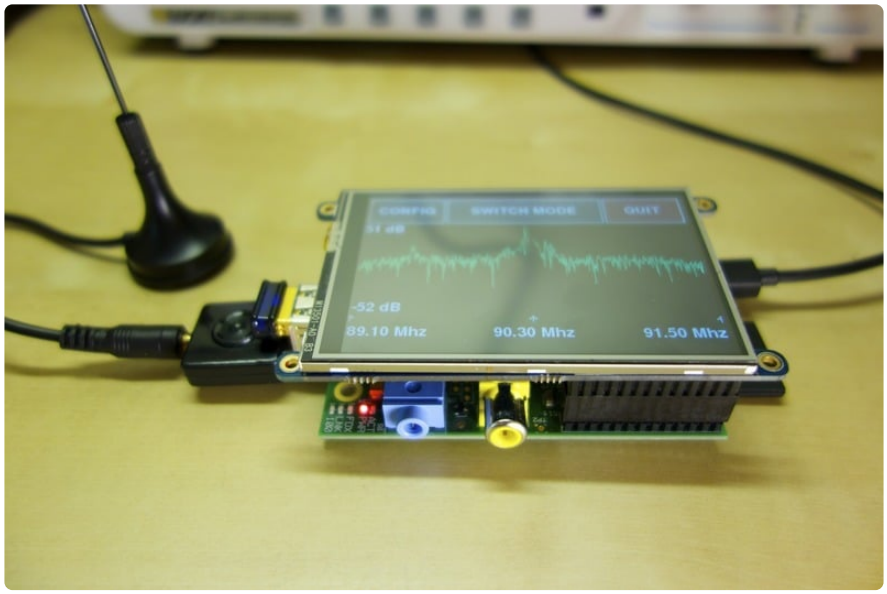
- Circuit Playground: F Is For Frequency (https://adafru.it/oBt)
- Getting Started With RTL-SDR and SDR# (https://adafru.it/oBu)
- FFT: Fun with Fourier Transforms (https://adafru.it/oBv)
- PiTFT 3.5" Touch Screen For Raspberry Pi (https://adafru.it/jse)
- Raspberry Pi: Using SSH (https://adafru.it/jsE)

Also for some inspiration on what you can do with SDR, check out these excellent presentations from previous DEF CON conferences (https://adafru.it/e2c):

- All Your RFz Are Belong To Me: Hacking The Wireless World With Software-Defined Radio (https://adafru.it/e2d)
- Noise Floor: Exploring Unintentional Radio Emissions (https://adafru.it/e2e)
- Hacker + Airplane = No Good Can Come Of This (https://adafru.it/e2f)

Before using SDR and scanning tools be sure to check the laws for your country. In some countries, like the US, there are frequencies for cell phones and other communication that you cannot legally tune: http://en.wikipedia.org/wiki/Scanner_(radio)#Legislation

Hardware



To build this project you'll need the following hardware:

- Raspberry Pi computer. Most any model will do. This project uses the GPIO header and one USB-A port...so anything but Compute Module or headerless Zero units can work.
- RTL-SDR (http://adafru.it/1497) software radio USB dongle
    - Note that you can use any tuner supported by the RTL-SDR library (https://adafru.it/e2g).

- 3.5" PiTFT display (https://adafru.it/jse)
    - The software is designed to run on a 480x320 3.5" PiTFT display. If you want to get your hands dirty in the code it should be possible to port it to smaller displays like the 320x240 2.8" PiTFT by changing font sizes and other dimensions. But as-is, use the larger 480x320 screen.
    - Many users have reported success using the official 7" Raspberry Pi Touch Display instead of the PiTFT. We'll provide alternate steps to try.

You might also consider some extra parts to extend the project:

- MCX jack to BNC (http://adafru.it/1531) or SMA (http://adafru.it/1532)adapters to plug in an external antenna.
- Powerboost charger (http://adafru.it/1944) and rechargeable battery (http://adafru.it/328), or a big mobile charger (http://adafru.it/1565) to power the project on the go.

Installation

The general steps for installing the Freq Show software are:

- Install Raspberry Pi Operating System
    - This guide was written and tested using the following OS images:
        - 2024-11-19-raspios-bookworm-arm64.img.xz
        - 2024-11-19-raspios-bookworm-armhf.img.xz
- Enable support for the PiTFT
    - This is done by enabling a device tree overlay via config.txt.
- Install required system software dependencies
    - git is needed to clone (download) the Freq Show code
    - librtlsdr is needed to support communicating with the TV tuner dongle
- Set up a Python Virtual Environment
    - Also install additional required Python libraries
    - More info on Python Virtual Environments here (https://adafru.it/1a9O)
- Install the Freq Show software

Let's go through each step.

Install Raspberry Pi OS

It should work to use any of the current **Desktop** versions (either 32bit or 64bit) of the official Raspberry Pi Operating System. There is nothing special about the initial operating system setup. It should work to follow Raspberry Pi's instructions here:

Raspberry Pi Operating System Install

https://adafru.it/1a66

Configuring the Pi to connect to your local network can be done when writing the OS image to the SD card using the **rpi-imager** software. See the OS customization section in the instructions linked above.

Before proceeding, make sure the Raspberry Pi is booting OK and is connecting to your local network and the internet. The following setup commands can be run by either SSHing into the Pi or by running them directly on the Pi via a terminal window.

> Before proceeding, make sure the Raspberry Pi is booting OK, is connecting to your local network and the internet, and that you can access it via SSH or a terminal window.

Enable Support for PiTFT

To enable support for the 3.5" PiTFT we add a device tree overlay entry to the config.txt file. Enabling the SPI and I2C interfaces can be done at the same time.

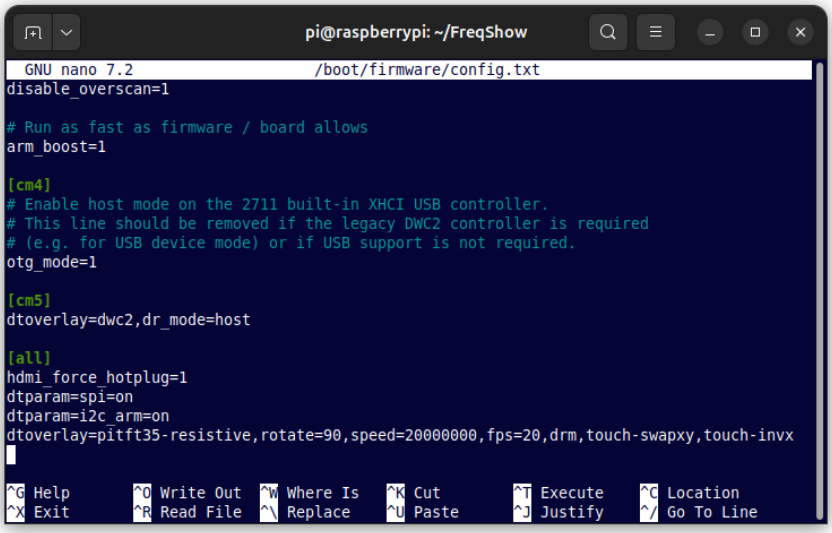
Use the text editor **nano** to open the **/boot/firmware/config.txt** file for editing:

```
sudo nano /boot/firmware/config.txt
```

Scroll to the bottom of the file and add these lines:

```
hdmi_force_hotplug=1
dtparam=spi=on
dtparam=i2c_arm=on
dtoverlay=pitft35-resistive,rotate=90,speed=20000000,fps=20,drm,touch-swapxy,touch-invx
```

It should look something like this:

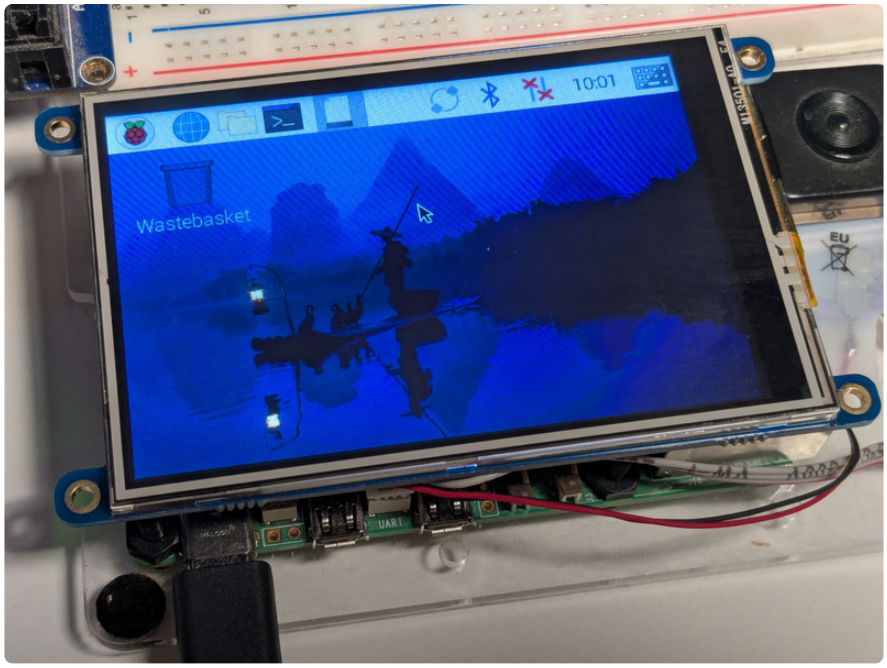


Then hit <**CTRL**><**X**> and say "**yes**" to save the changes and exit nano.

Next, reboot the pi:

```
sudo reboot
```

And make sure the Pi Desktop shows up on the PiTFT after rebooting.



Install System Dependencies

We need a few additional system level software components. These are operating system libraries, not Python libraries, so we use apt-get in the usual way. Connect to a terminal, either via SSH or directly, on the Pi and execute the following commands to install these dependencies:

```
sudo apt-get update
sudo apt-get install -y git librtlsdr0
```

Set Up Virtual Environment

Next we'll create a Python Virtual Environment to use with the Freq Show software. In addition to creating the virtual environment, we'll install some additional Python libraries needed by the Freq Show software.

To create the virtual environment, run the following commands in a terminal:

```
python -m venv env
```

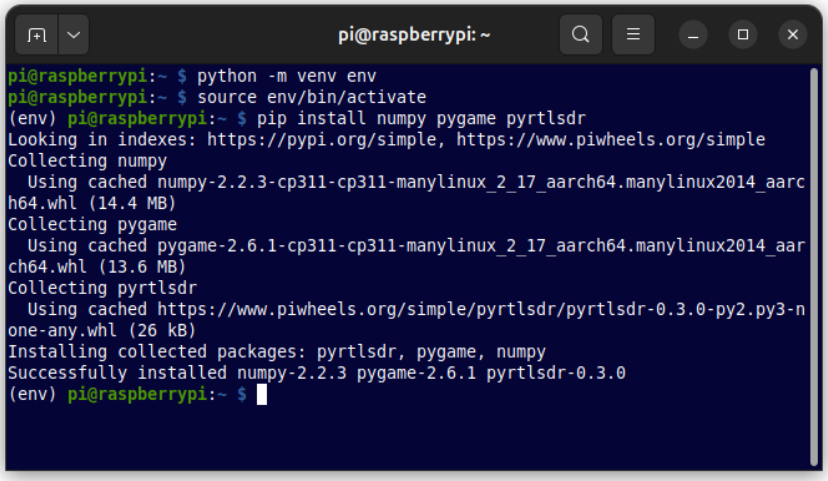
And now activate the virtual environment:

```
source env/bin/activate
```

The prompt should change to show the virtual environment name "env". Now we can pip install the additional Python libraries needed:

```
pip install numpy pygame pyrtlsdr
```

Running the above commands should looking something like this:



The actual output from running pip may vary depending on your setup, location, etc. But it should result in some indication that the libraries are already installed and/or were successfully installed.

Install Freq Show Software

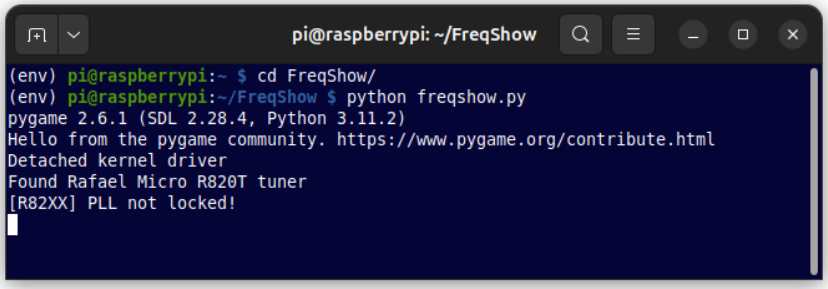
The Freq Show software is kept in a Github Repository (https://adafru.it/e2j). We'll download a local copy of this code using git to clone the repository. To do so, in a terminal, run the following commands:

```
cd ~
git clone https://github.com/adafruit/FreqShow.git
cd FreqShow
```

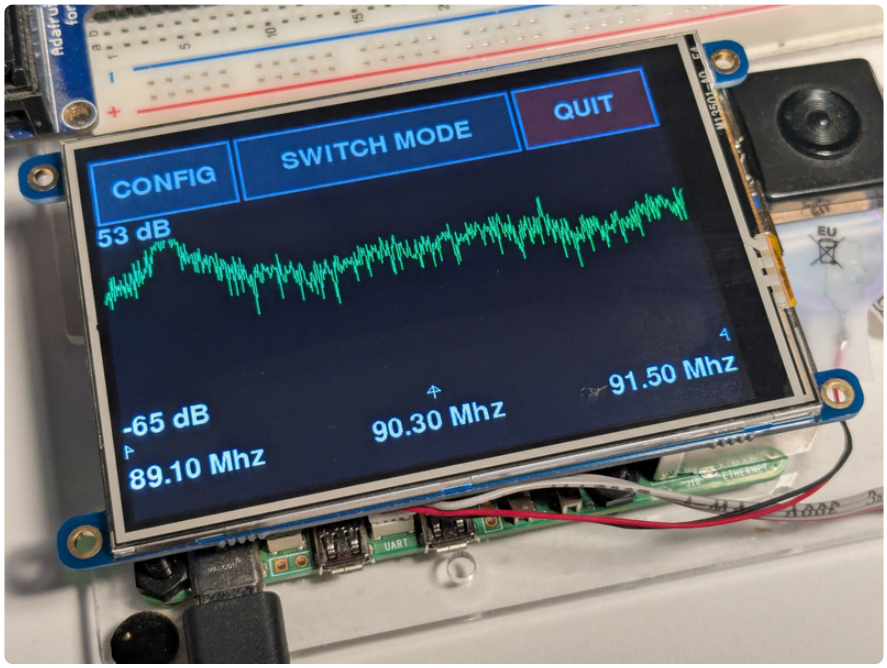
And now it should be possible to run the Freq Show software:

```
python freqshow.py
```

The output should look like this:



And the Freq Show screen should show on the PiTFT:



Now let's go into more details about using the Freq Show software.

Usage

Running the Freq Show software is a two step process:

- Enable the Python Virtual Environment
- Run the Freq Show software

Both of those steps were covered in the previous section during installation and setup. However, any time you are starting from a fresh reboot, both steps will need to be done again. These commands are run on the command line in a terminal window.

To enable the Python Virtual Environment, run the following command:

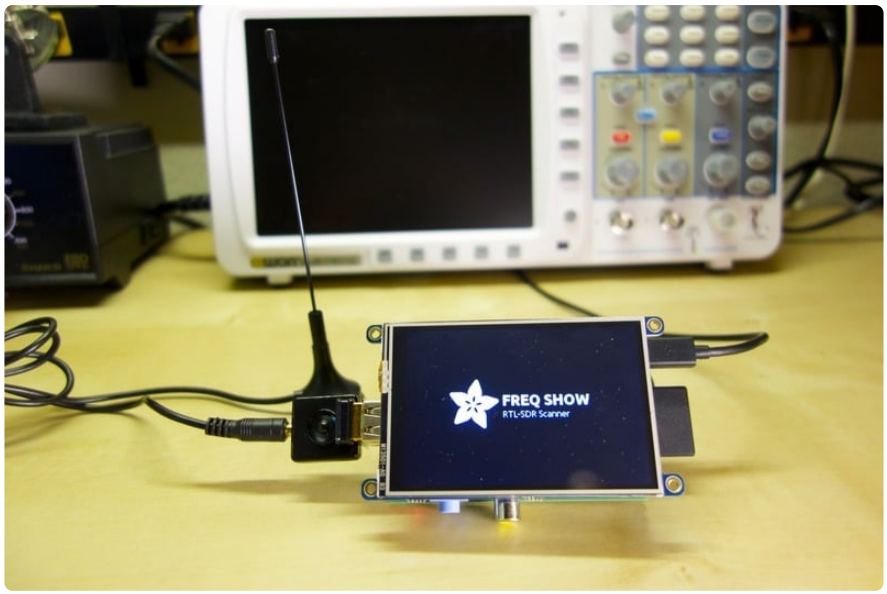
```
source env/bin/activate
```

The prompt should change to show the name "env". Then the Freq Show software can be run by changing to the directory with the software and launching the code with Python:

```
cd FreqShow
python freqshow.py
```

That should look something like this:

After a few moments a splash screen should display on the PiTFT as the program loads:

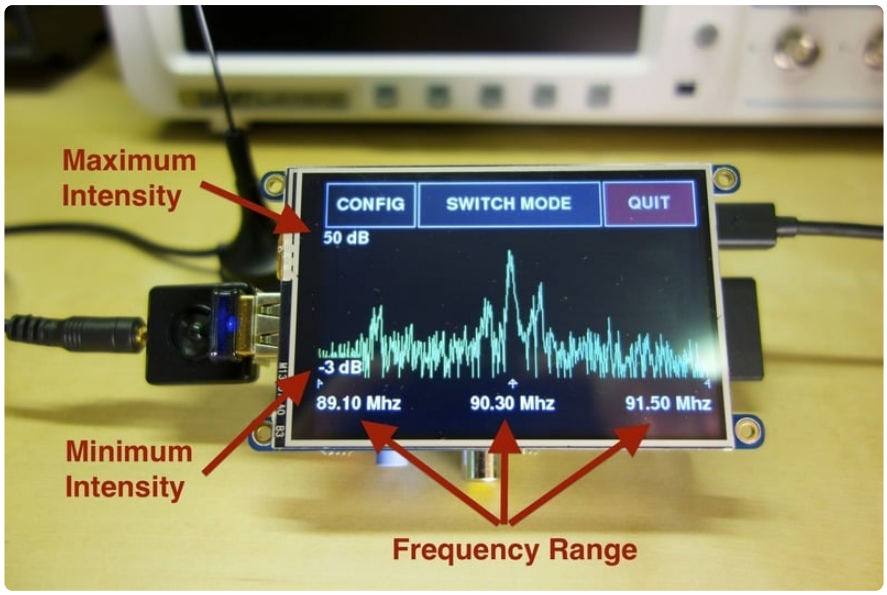


If the program fails to load, go back to the installation steps (https://adafru.it/oBw) and carefully check all the dependencies are installed, then try again.

Note that if you see an error like "Kernel driver is active, or device is claimed by second instance..." it could mean the TV tuner driver is still loaded by the kernel. Normally when the RTL-SDR code is compiled with the steps in this guide it should automatically unload the conflicting kernel driver, but if that fails you can manually stop the conflicting module from loading. See the "Avoid the Raspberry Pi to load a kernel module" step from this blog post (https://adafru.it/fIS) to see what kernel modules should be added to the **/etc/modprobe.d/raspi-blacklist.conf** file (make sure to reboot the Pi after editing that file).

Once the program loads, it will start by displaying an animated frequency graph and a menu of options. The graph displays the intensity of radio signals (measured in decibels (https://adafru.it/cM0)) across a range of frequencies. The taller a peak on the graph, the higher the intensity of the signal at that frequency.

Below is an overview of the important parts of the graph:



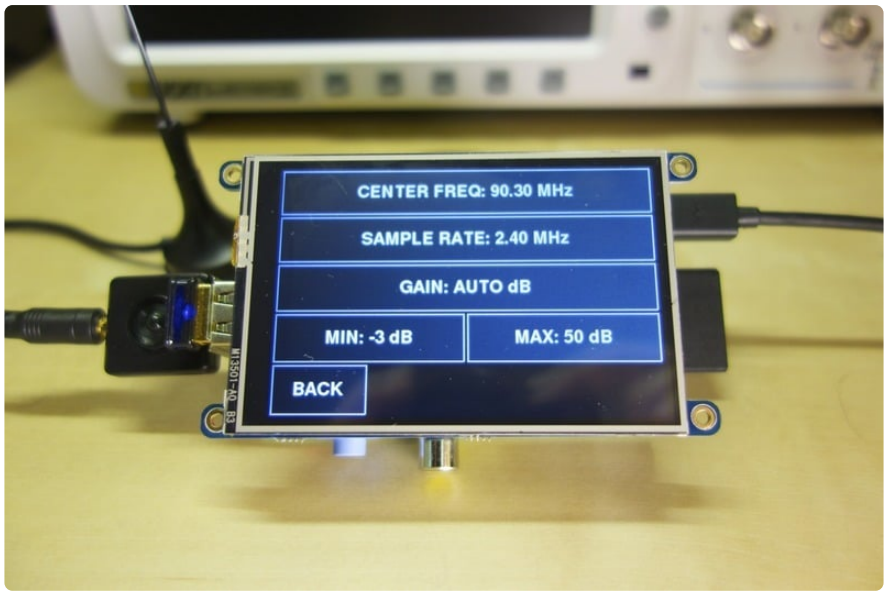
To the left of the graph, the minimum and maximum intensity values are displayed in decibels. The top value is the maximum intensity, in this case 50 dB, which means a point at the very top of the graph has an intensity of 50 dB. Similarly the bottom value is the minimum intensity, -3 dB, so points at the bottom of the graph have an intensity of -3 dB. Points between the top and bottom of the graph have an intensity in between the minimum and maximum, so for example a point in the middle of the graph would have an intensity around 26-27 dB.

At the bottom of the graph the range of frequencies are displayed. The minimum frequency is displayed at the bottom left, the center frequency in the middle, and the maximum frequency at the bottom right. In the image above you can see a total range of about 2 MHz of frequencies centered at 90.3 MHz.

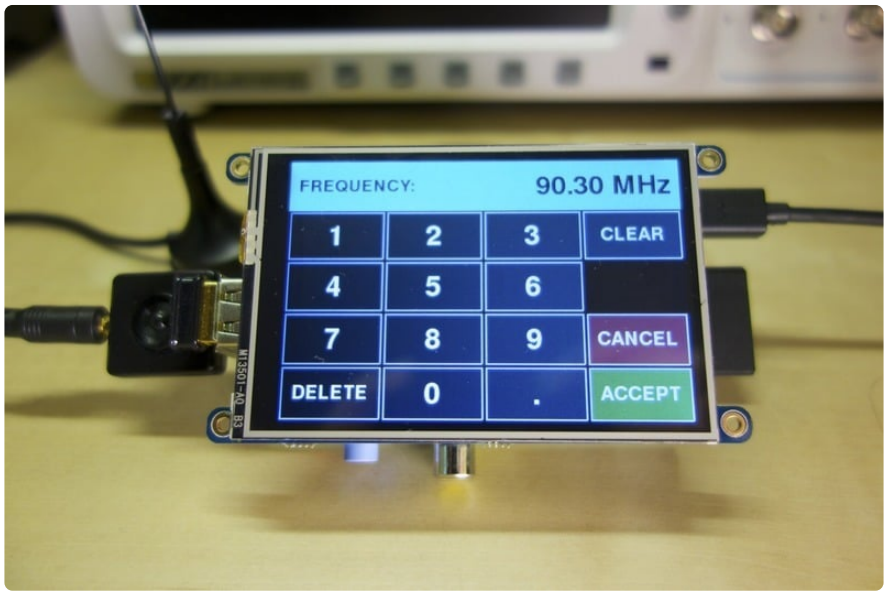
The frequency 90.3 MHz actually corresponds to an FM radio station (KEXP (https://adafru.it/e2l) in Seattle, WA). It's interesting to see the spikes of intensity near the center frequency--the graph is visualizing what an FM radio plays as audio!

REMINDER: FreqShow only DISPLAYS THE SPECTRUM and DOES NOT decode or demodulate the radio signal into audio or data!

You can change the center frequency and other settings by pressing the **CONFIG** button in the upper left corner. A list of settings such as the following should appear:



Press the **CENTER FREQ** button at the top to bring up a dialog to change the center frequency:



You can change the center frequency just like inputting numbers in a calculator. Press the **CLEAR** button to completely erase the current frequency value, enter new digits by pressing numbers (or the decimal point), and delete the last digit by pressing **DELETE**.

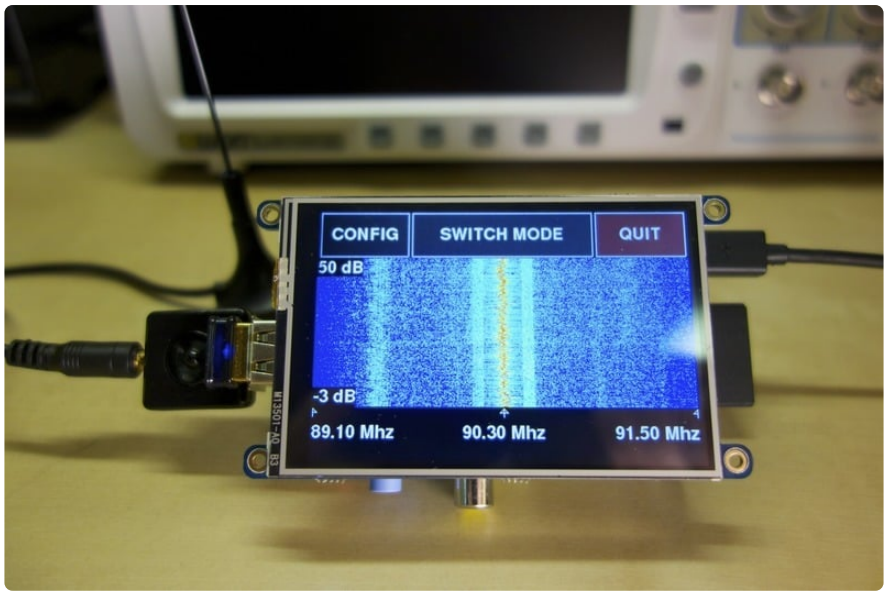
Once you've entered a new value, you can press **ACCEPT** to accept it and change the center frequency. If you don't want to change the frequency press **CANCEL** to dismiss the dialog without changing the value.

Try changing the center frequency to that of an FM radio station in your area. FM radio in the US is broadcast over 88 to 108 Mhz so pick a value within that range. After accepting a new center frequency you should be returned to the settings list.

Press **BACK** in the lower left to return to the frequency graph. Notice how the graph changes to show the intensity of the signal at the new center frequency.

Another useful way to visualize radio frequency data is with a spectrogram that shows the intensity of the signal over time. In the main graph view, press the **SWITCH MODE** button at the top of the screen. You should see the graph change to start building a waterfall plot that scrolls up from the bottom to the top of the screen.

For example this is a waterfall plot that has entirely filled the screen:



The waterfall plot displays intensity over time. Each row of the plot represents a point in time, and each pixel color in a row represents the intensity of the signal at that frequency. Pure blue pixels are the minimum intensity, pure red pixels are the maximum intensities, and a gradient of blue-cyan-yellow-red represents in between intensities.

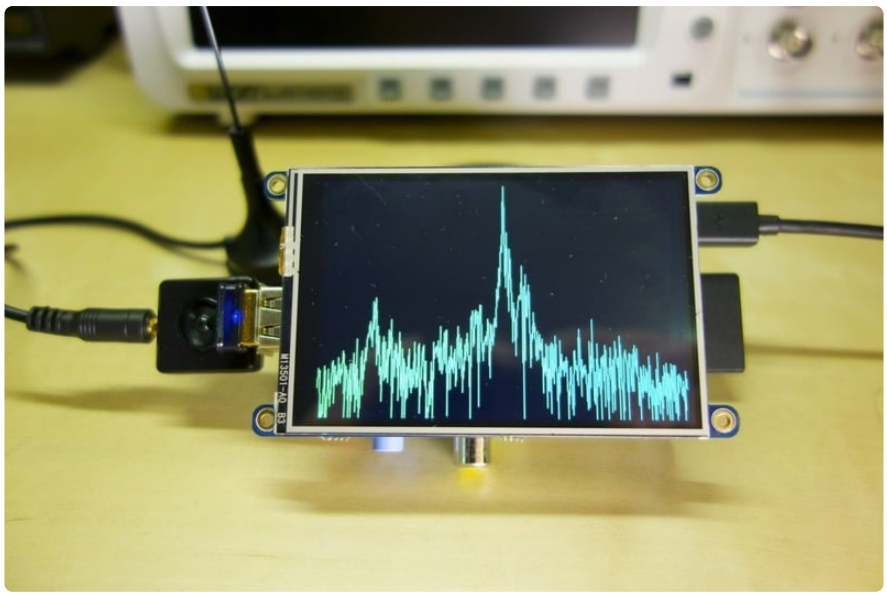
Notice in the picture above the highest intensity frequencies in the center are red, while the lowest intensity frequencies at the edges are blue. Smaller peaks are cyan and yellow.

About once a second, the graph will scroll up and a new measurement row will be added at the bottom. Over time the waterfall plot will fill the screen, with the most recent measurements at the bottom and oldest at the top. By looking at how the color of the waterfall plot changes, you can see how the intensity of a radio signal changes over time.

See the image below for the full range of intensity colors:

To switch back to the instantaneous frequency graph, press the **SWITCH MODE** button at the top of the screen again. You can press **SWITCH MODE** to swap between instantaneous and waterfall plot modes.

Another useful feature of the graph is a full screen view. Press anywhere on the graph image itself, like the middle of the screen. You should see the graph maximize itself in a full screen view with no buttons or labels. For example this is a full screen of the instantaneous mode:



The fullscreen view is great if you want to more clearly see the graph. You can switch back to the normal view by pressing the middle of the graph again.

Now go back to the settings menu by pressing the **CONFIG** button. Here's a quick overview of each setting:

- **Center Frequency**
  - This is the center frequency of the tuner and must be a value in megahertz. The exact range of allowed values will depend on the radio tuner, but for the RTL-SDR in the store the range is about 24 Mhz to 1,850 Mhz. **Note that it can be illegal to tune certain frequencies so check the laws in your country (https://adafru.it/e2m)!**
- **Sample Rate**
  - This controls how wide the range of frequencies are that will be displayed in the graph. The tuner doesn't support a large range of sample rates so your best bet is to stick with the default of 2.4 Mhz.

- **Gain**
    - The internal gain of the tuner can be adjusted with this setting. By increasing the gain you can remove noise and make a weak signal more easily visible. The tuner has a limited range of gains from about 1 to 50 dB and will try to use a value as close to the configured gain as possible. You can also choose **AUTO** as a gain option to have the tuner automatically adjust the gain for the best signal reception. I recommend sticking with auto gain for simplicity.
- **Min and Max**
    - These values at the bottom of the setting list control the minimum and maximum intensity values of the graph. Like gain they can be set to **AUTO**, in which case they will adjust themselves based on the lowest or highest intensity values ever seen. However if you want to restrict the graph to a certain range of values, you can set explicit intensity values in decibels. I've found a range of -3 dB to 50 dB is a good general range to set when using automatic gain.

Finally if you want to exit the tool, in the main graph view press the **QUIT** button in the upper right. A confirmation dialog will show which you can accept to exit, or cancel to return back to the tool.

That's all there is to using the Freq Show Raspberry Pi RTL-SDR scanner tool! If you run into issues with the tool or wish to contribute to it, check out the tool's home on GitHub (https://adafru.it/e2j).

Have fun exploring the world of radio waves around you!