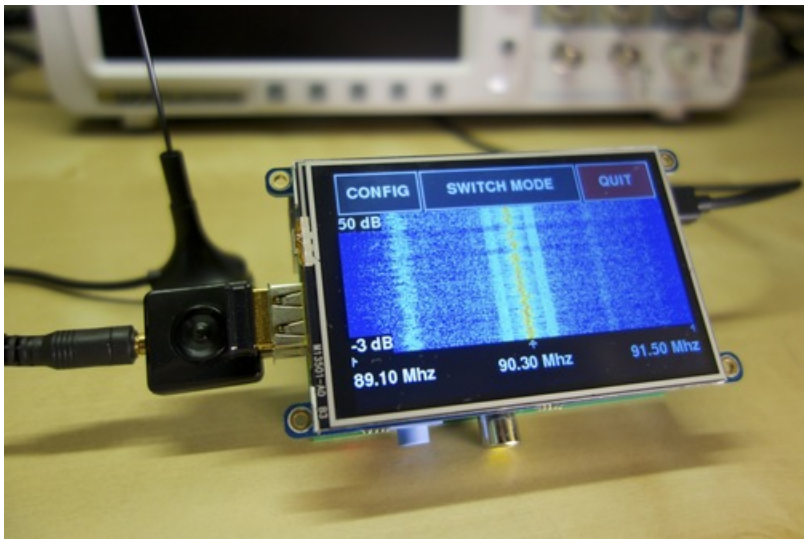
adafruit learning system

Freq Show: Raspberry Pi RTL-SDR Scanner

Created by Tony DiCola



Last updated on 2018-08-22 03:44:22 PM UTC

Guide 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 examime 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), 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 (https://adafru.it/e25). 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 (https://adafru.it/e27), and RTL-SDR (https://adafru.it/e25) radio dongle. With the Raspberry Pi Freq Show RTL-SDR scanner you can visualize the invisible world of radio!

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, the model B+ (http://adafru.it/1914), model B (http://adafru.it/998), or model A will 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)
    - Right now the software is optimized to run on the 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.

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 (https://adafru.it/dDF) and rechargeable battery (http://adafru.it/328), or a big portable phone charger (http://adafru.it/1565) to power the project on the go.

Once you have the parts assembling the project is very easy.

Follow the 3.5" PiTFT guide (https://adafru.it/jse) to learn how to install and use the PiTFT. Make sure to start with the easy install PiTFT image (https://adafru.it/nZA) so you can get up and running as quickly as possible. Don't go forward until you have the PiTFT working with the Raspberry Pi!

Finally plug in the RTL-SDR dongle to a USB port on the Raspberry Pi.

That's it, your hardware is ready to go! Continue on to learn how to install the software for this project.

Installation

Dependencies

Before you can install the software you'll need to install a few depdendencies. Make sure you're running the Raspbian operating system and it has been configured to work with the PiTFT. You will also need to make sure the Pi has access to the internet through a wired or wireless networking connection.

Connect to a terminal on the Pi and execute the following commands to install the dependencies:

```
sudo apt-get update
sudo apt-get install cmake build-essential python-pip libusb-1.0-0-dev python-numpy git pandoc
```

Don't worry if you see error messages that certain packages are already installed, you can ignore those and move on.

Next download, compile, and install the RTL-SDR library (https://adafru.it/e2g) by executing:

```
cd ~
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON -DDETACH_KERNEL_DRIVER=ON
make
sudo make install
sudo ldconfig
```

Finally install the RTL-SDR Python wrapper (https://adafru.it/e2i) by executing:

```
sudo pip install pyrtlsdr
```

At this point all the dependencies should be installed and you're ready to download and install the project software.

> Raspbian Jessie Warning!! If you're using Raspbian Jessie (which if you're using the current PiTFT easy install images then yes you're likely using Jessie) you MUST follow the steps below to make the touchscreen work.

If you're using Raspbian Jessie there's one more critical step to follow to fix an issue with incompatibilities between the SDL library and touchscreen support. See the FAQ page here for a script to create and run so that an older SDL library can be installed (https://adafru.it/l6e). You **must** run this script to make sure FreqShow works on Raspbian Jessie (if you don't run it then mouse/touchscreen input will be erratic and not work at all).

Installation

Installing the software for this project is as easy as downloading its Python source code (https://adafru.it/e2j). At a terminal on the Pi run the following commands to download the source:

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

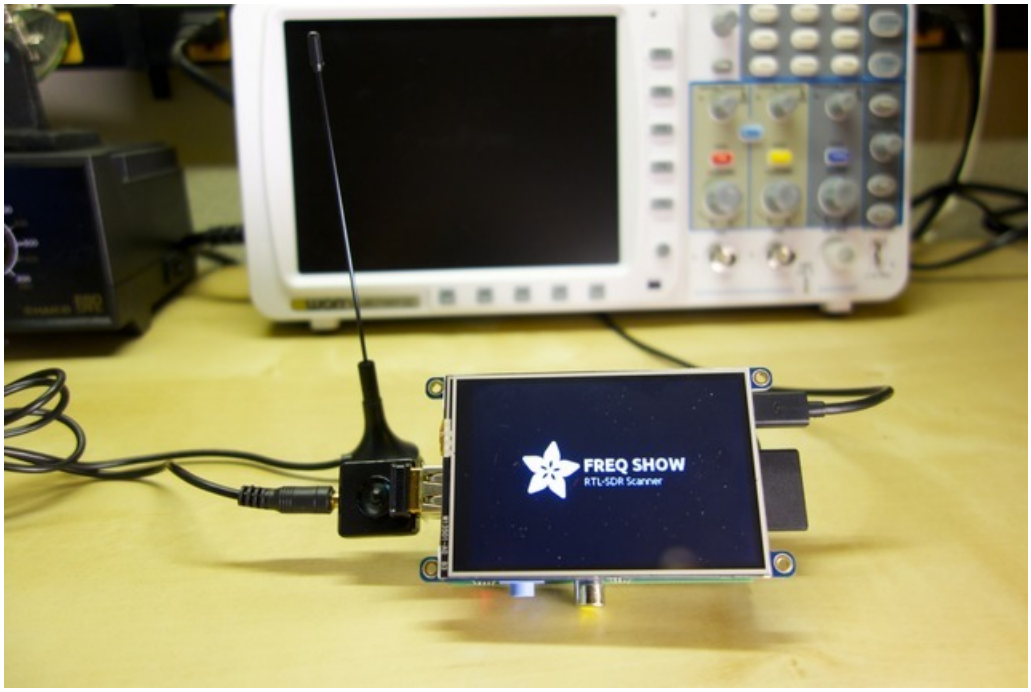
That's it, at this point the software should be downloaded and ready to run! Continue on to the next page to learn how to run the software.

Usage

To start the program connect to a terminal session on the Raspberry Pi and navigate to the FreqShow folder (if you aren't there already), then run:

```
sudo python freqshow.py
```

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

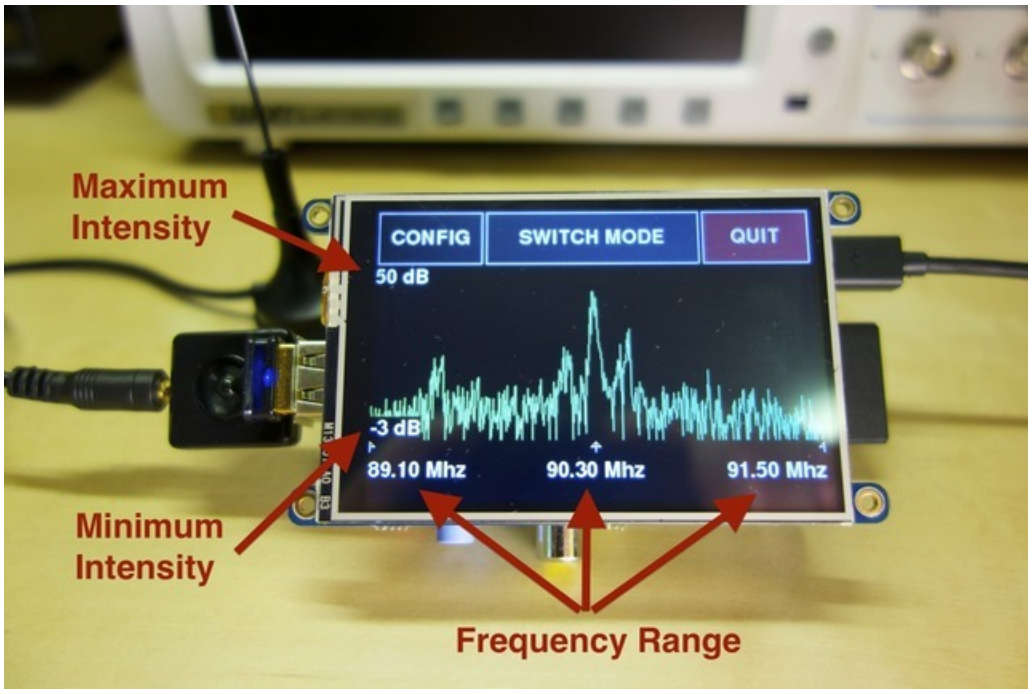


If the program fails to load go back to the installation steps (https://adafru.it/oBw) and carefully check all the dependencies are installed, then try again.

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 frequncy graph, and 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:



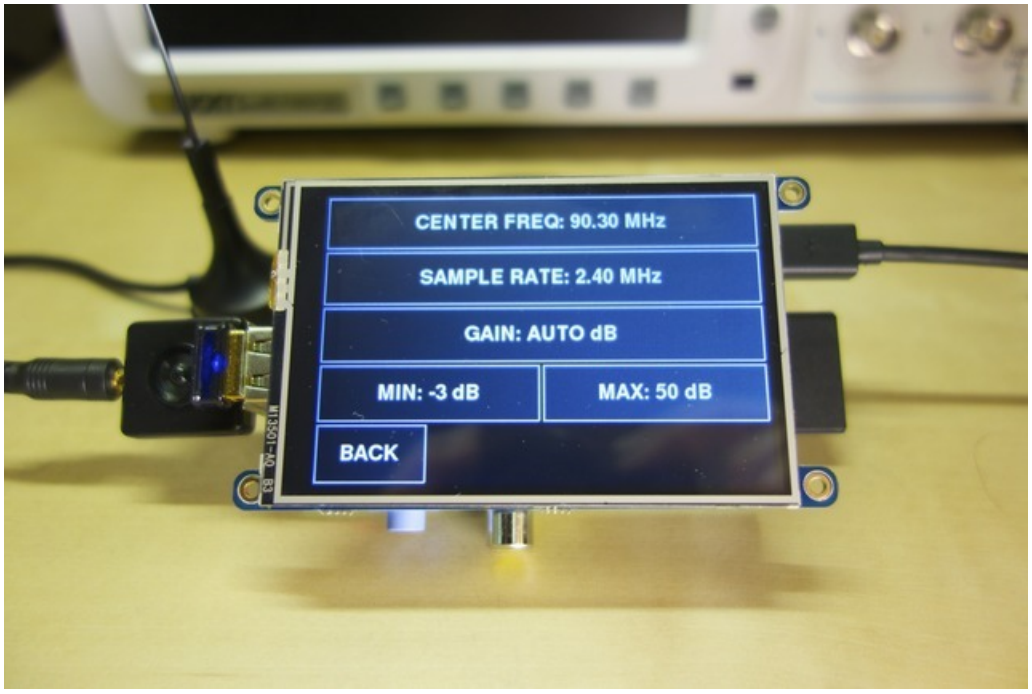
To the left of the graph the minimum and maximum intensity values are displayed in decibels. The top value is the maximum intensity, in this case 50 dB, which means a point at the very top of the graph has an intensity of 50 dB. Similarly the bottom value is the minimum intensity, -3 dB, so points at 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.

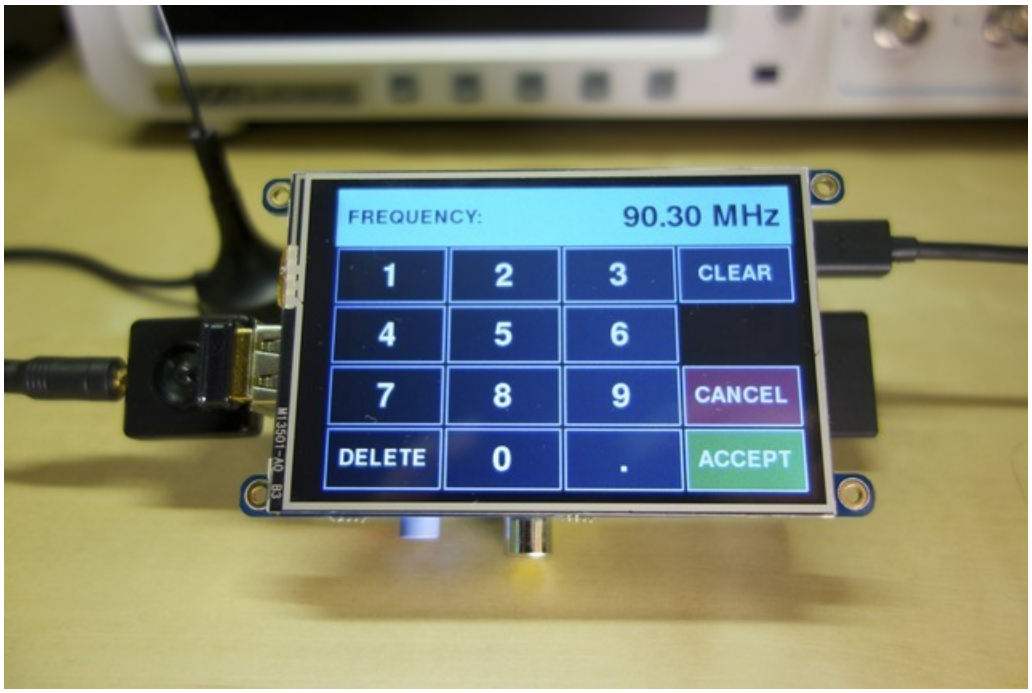
The frequency 90.3 MHz actually corresponds to an FM radio station (KEXP (https://adafru.it/e2l) in Seattle, WA). It's interesting to see the spikes of intensity near the center frequency--the graph is visualizing what an FM radio plays as audio!

Be aware FreqShow only displays the spectrum and does not currently turn the radio data into audio.

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**.

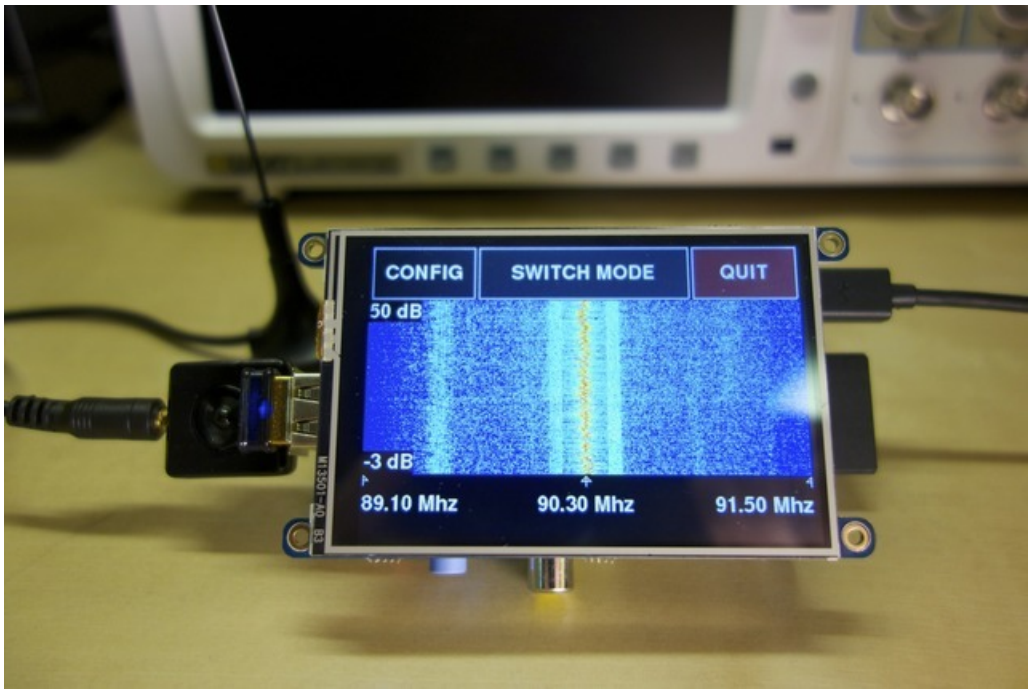
Once you've entered a new value you can press **ACCEPT** to accept it and change the center frequency. If you don't want to change the frequency press **CANCEL** to dismiss the dialog without changing the value.

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.

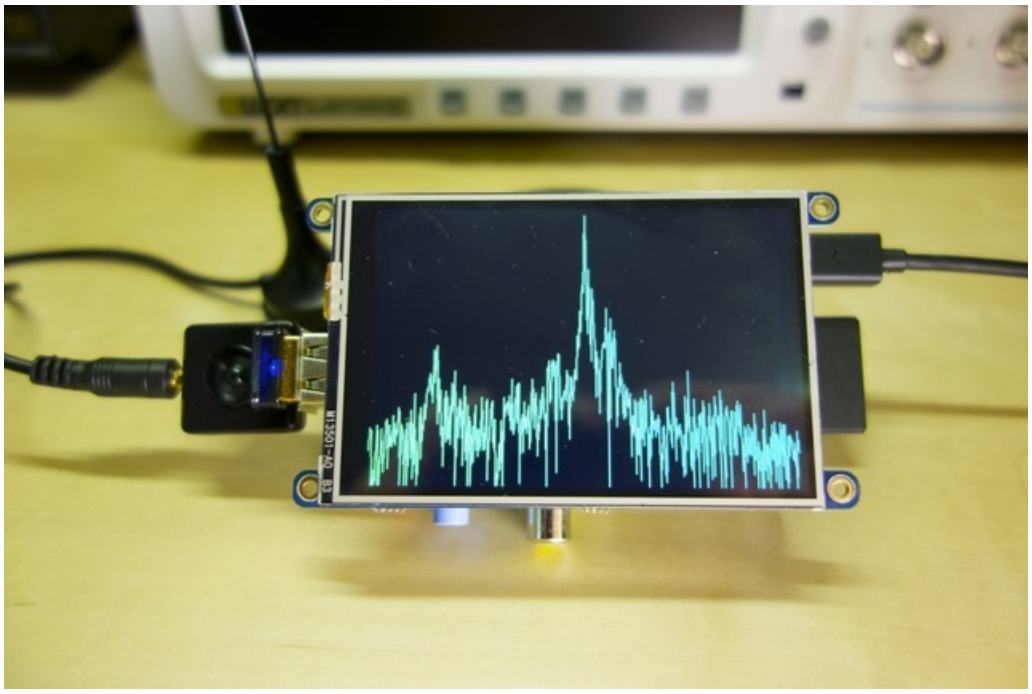
Notice in the picture above the highest intensity frequencies in the center are red, while the lowest intensity frequencies at the edges are blue. Smaller peaks are cyan and yellow.

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 watefall 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!