F5OEO/rpitx

Originalartikel

Backup

<html> rpitx is a radio transmitter for Raspberry Pi (B, B+, PI2, PI3B,PI3B+,PIZero,PiZerow) that transmits RF directly to GPIO. It can handle frequencies from 5 KHz up to 1500 MHz. Before you transmit, know your laws. rpitx has not been tested for compliance with regulations governing transmission of radio signals. You are responsible for using your rpitx legally. Rpitx is now based on a general Radio Frequency library : https://github.com/F5OEO/librpitx Created by Evariste Courjaud F5OEO. Code is GPL Assuming a Raspbian Lite installation (stretch) : https://www.raspberrypi.org/downloads/raspbian/ <div class=„highlight highlight-source-shell“> <pre> git clone https://github.com/F5OEO/rpitx cd rpitx # make sure to have access to the internet to download packages # or download and install them manually (libsndfile1-dev and imagemagick) ./install.sh </pre> Plug a wire on GPIO 4, means Pin 7 of the GPIO header (header P1). This acts as the antenna. The optimal length of the wire depends the frequency you want to transmit on, but it works with a few centimeters for local testing. <h2>General</h2> rpitx is the main software to transmit. It allows to transmit from: IQ files *.iq (can be generated by external software like GNU Radio). Frequency/Time files *.ft (generally used to easily implement digital modes) Usage: <pre>rpitx [-i File Input][-m ModeInput] [-f frequency output] [-s Samplerate] [-l] [-p ppm] [-h] -m {IQ(FileInput is a Stereo Wav contains I on left Channel, Q on right channel)}

```
            {IQFLOAT(FileInput is a Raw float interlaced I,Q)}
            {RF(FileInput is a (double)Frequency,Time in nanoseconds}
            {RFA(FileInput is a (double)Frequency,(int)Time in
nanoseconds,(float)Amplitude}
             {VFO (constant frequency)}
```

-i path to File Input -f float frequency to output on GPIO_18 pin 12 in khz : (130 kHz to 750 MHz), -l loop mode for file input -p float frequency correction in parts per million (ppm), positive or negative, for calibration, default 0. -d int DMABurstSize (default 1000) but for very short message, could be decrease -c 1 Transmit on GPIO 4 (Pin 7) instead of GPIO 18 -h help (this help).

</pre> <h2>Modulation samples</h2> Some modulations are included in this repository and can be easily extended. These scripts create files which can be used by rpitx. Some output in IQ (like SSB) other in FT (like SSTV). <h3>Single Side Band modulation (SSB)</h3> pissb converts an audio file (Wav 48KHZ mono only!) to SSB (Upper Side Band right now) and outputs it to an IQ file. Assuming your audio file is in your current working directory: <div class=„highlight highlight-source-shell“> <pre> ./pissb audio48mono.wav ssbIQ.wav </pre> You could then transmit it on 50MHz (please set a correct frequency to

be legal) <div class=„highlight highlight-source-shell“> <pre> sudo ./rpitx -m IQ -i ssbIQ.wav -f 50000 -l </pre> A sample script

```
testssb.sh
```

is included. <h3>Broadcat Frequency Modulation (FM)</h3> pifm converts an audio file (Wav) to broadcast FM using Christophe Jacquet (F8FTK) PiFmRds project fork : https://github.com/F5OEO/PiFmRds See Readme from this project for instructions. <h3>Slow Scan Television (SSTV)</h3> pisstv converts an RGB picture to a SSTV .ft file. The SSTV module will transmit using the Martin M1 encoding mode. If you have a JPEG picture 320×256 you can convert it to an RGB picture with: <div class=„highlight highlight-source-shell“> <pre> imagemagick convert -depth 8 picture.jpg picture.rgb </pre> You can then transmit on 144.5Mhz (please set a correct frequency to be legal) : <div class=„highlight highlight-source-shell“> <pre> ./pisstv picture.rgb 144.5e6 </pre> A sample script

```
snapsstv.sh
```

grabs a picture from a PiCamera and then transmits it on 144.5 MHz. <h3>Fast Simple QSO (FSQ)</h3> pifsq allows to send a text with the new FSQ modulation It is still under development. A sample script

```
testfsq.sh
```

allows to send a text with FSQ <h3>Weak Signal Propagation Reporter (WSPR)</h3> wsprrpi allows to send wspr beacon https://en.wikipedia.org/wiki/WSPR_(amateur_radio_software) It uses a fork from James (https://github.com/JamesP6000/WsprryPi) project and adpated to librpitx for frequency ehancement and cleaner spectrum (to be confirmed) See https://github.com/F5OEO/WsprryPi for instructions <h3>Variable Frequency Offset (VFO)</h3> A VFO mode is provided to allows precise frequency resolution. For example to set a carrier on 100MHz (please set a correct frequency to be legal) <div class=„highlight highlight-source-shell“> <pre> sudo ./rpitx -m VFO -f 100000 </pre> All rights of the original authors reserved. Special thanks to Sylvain Azarian F4GKR for improving SSB modulation Inspired by http://pe1nnz.nl.eu.org/2013/05/direct-ssb-generation-on-pll.html http://www.icrobotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter https://github.com/richardghirst/PiBits/pull/18 http://www.bellard.org/dvbt/ </html>