JS8 and JS8Call

Telemetry and Messaging

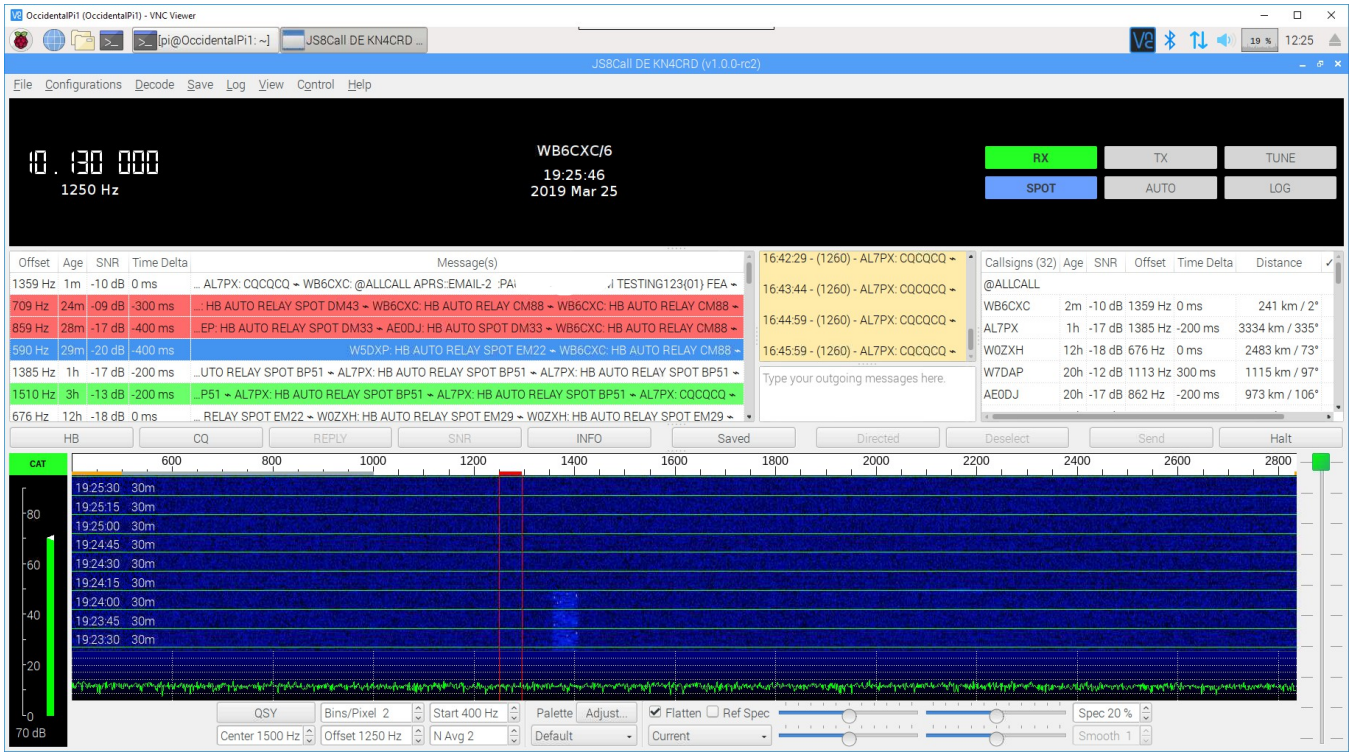
A JS8 to APRS Gateway Receiver

Paul Elliott / WB6CXC

HF Telemetry

- Drift-buoy project needs a good way to transmit data to shore-based server.
- 40 meters, 30 meters, and 20 meters are appropriate bands for worldwide paths.
- 30m HF APRS has necessary features, but coding is far from optimum for error-prone weak-signal conditions.
- HF WSPR has good low-level characteristics and a good worldwide receiving / reporting infrastructure. It is being used for some telemetry but with a very limited and inflexible data format.
- FT8 has good low-level characteristics but a poor receiving / reporting infrastructure. It also has very limited data capability.
- **JS8Call** is a new mode, derived from FT8
  - It includes Forward Error Correction and is optimized for weak-signal conditions.)
  - It provides a flexible data format and APRS interface.

JS8Call



- Created by Jordan Sherer / KN4CRD
- Runs on Windows, Mac OSX, Raspberry Pi, Desktop linux
- The program is now in general release, see www.js8call.com

JS8Call

- JS8Call was previously named FT8Call.
- Proposed July 2017, first program release July 2018
- JS8Call uses a custom FT8 modulation called JS8 (Jordan Sherer designed 8-FSK modulation). This is the base RF transport.
- JS8Call has a “directed calling” protocol laid on top of the base RF transport to support free-form and directed message passing.
- Uses a keyboard messaging style interface.
- Provides API for remote and programmatic interface.
- **Supports messaging to APRS**
  - **Position report, telemetry (etc), text msg, email.**

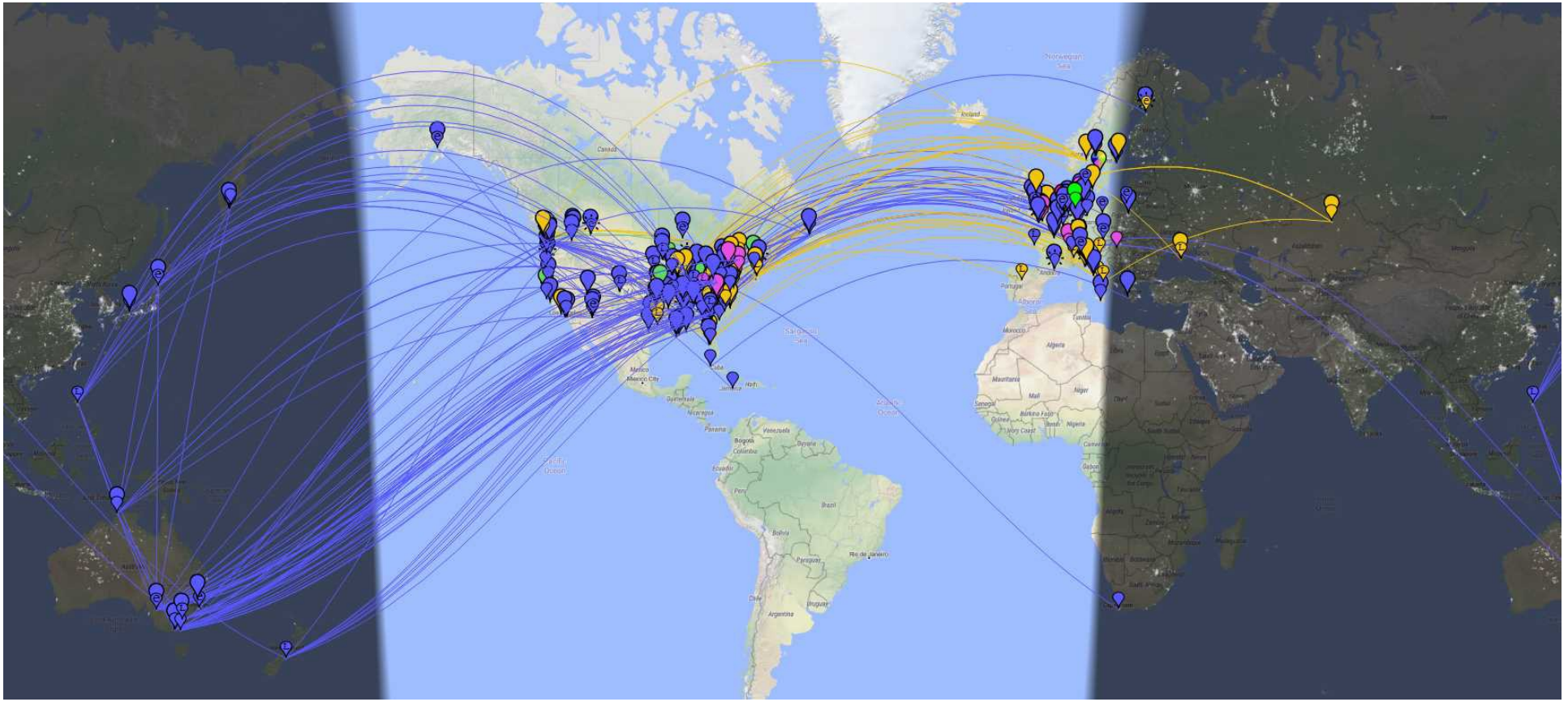
JS8Call Features

- Lots of good features for actual real-time free-form QSOs and general communications.
- Has mailbox capability so messages can be stored and automatically delivered.
- Messages can be relayed from station to station.
- Stations can be queried for their “heard” list, making relay routes easier to manage.
  - No automatic routing (yet)
- Periodic “Heartbeat” transmissions, and auto-ACKs help provide network status information.

JS8Call Suitability

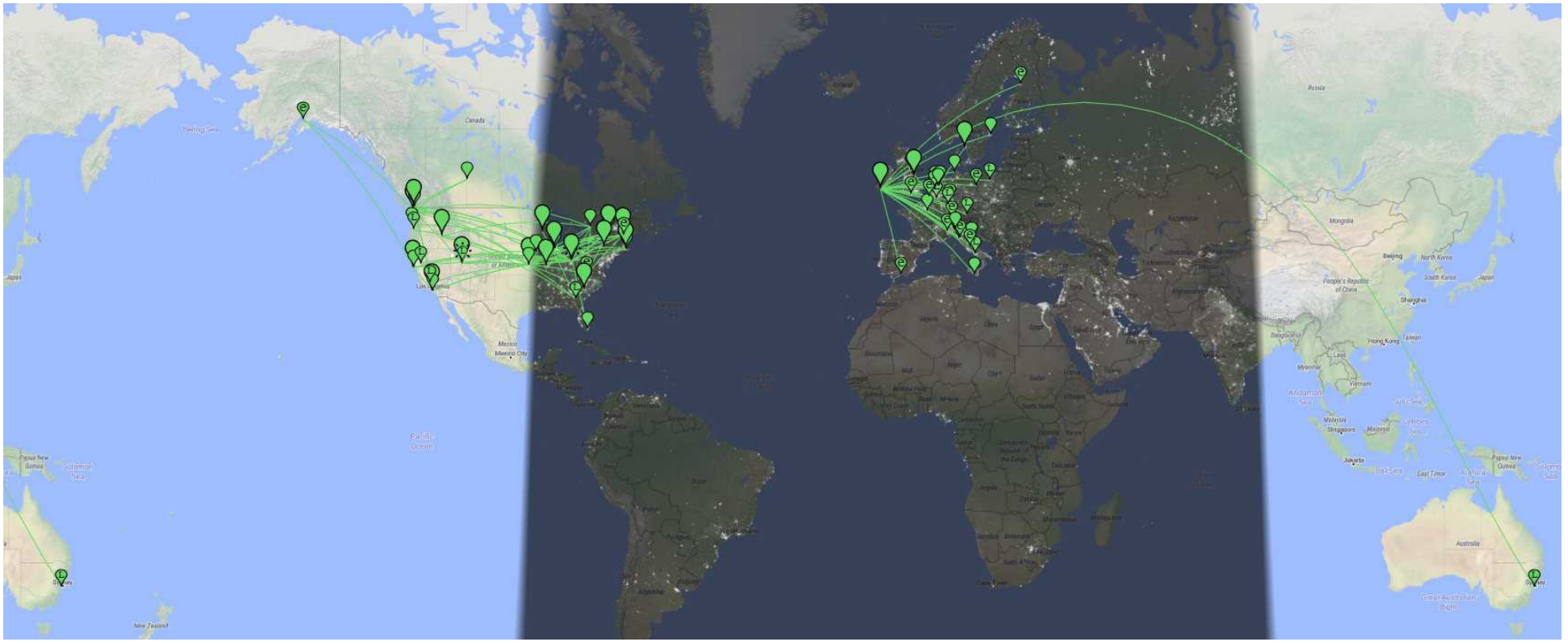
- JS8Call was created for human keyboard-keyboard contacts, but it also has the necessary fundamental characteristics to provide a reliable method for low-speed digital communications of arbitrary data.
- But can it provide a receiving infrastructure? Will it wither on the vine as many previous digital modes have?
- FT8 is the most popular HF digital mode in use now, and it appears that many FT8 users are trying JS8. More than 10,000 hams have downloaded the latest version of the JS8Call program.
- A mix of occasional operators and 24/7 gateway stations should provide good coverage. Not all operators will be relaying APRS, but some will.

JS8 Activity



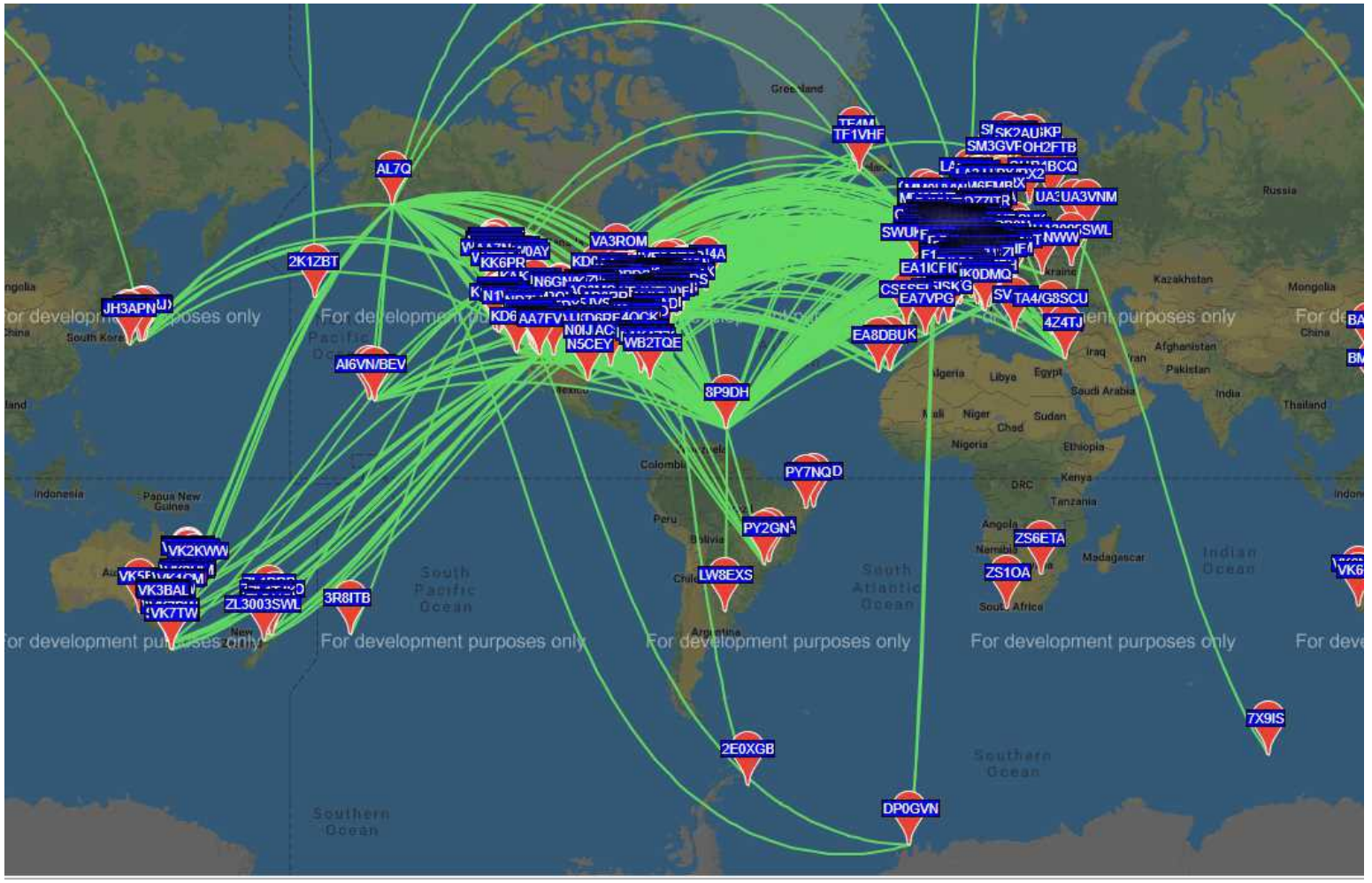
- All bands, 24-hours
- 40 meters most active, then 20 meters

30 Meter JS8 Activity



- 24-hours
- Not a whole lot of activity, but still useful

30 Meter WSPR Activity



- 24-hours
- We can see that propagation isn't the issue

Building a JS8 Gateway Station

- More 24/7 Stations!
- TX/RX is great, but RX-only is still useful
- Goal: Cheap, Easy, Good
  - Pick any two?
- 30 Meters (10.130 MHz USB)
- Cheap / Easy / (Good Enough) Rcv-Only
- I now have three 24/7 Cheap/Easy receive-only gateway stations on 30 meters: Friday Harbor WA, Occidental CA, and Anjala Finland

Receive-Only
Cheap / Easy / (Good Enough)

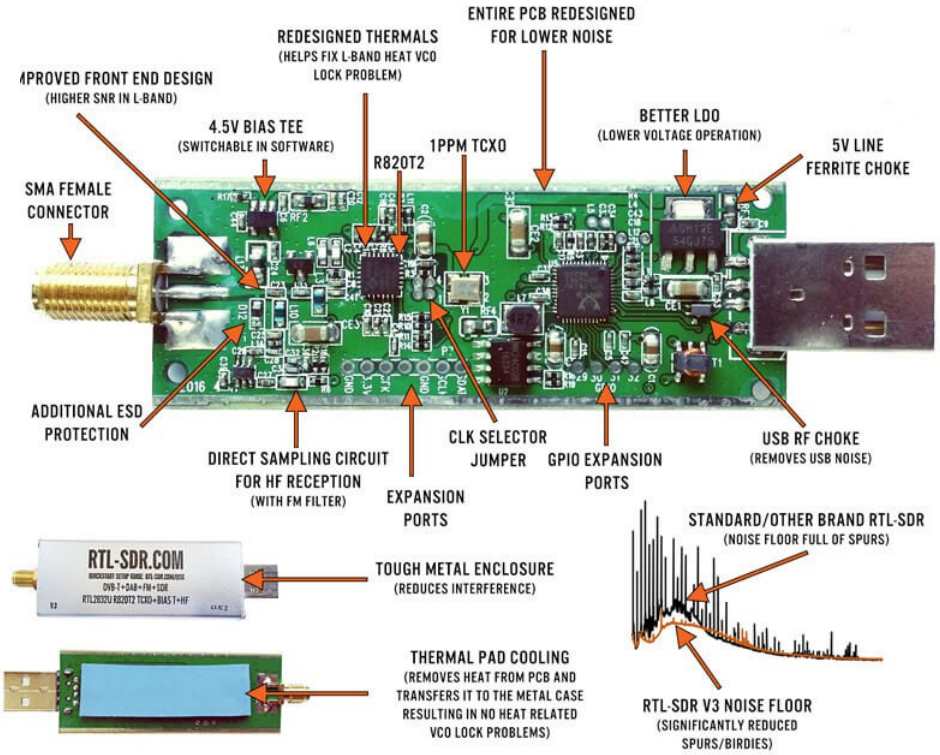
- RTL-SDR Blog Version 3: $22
- Raspberry Pi 3B: $34
- 8G Micro SD Card: $4
- RF Preamp: $11
- 10 MHz Front-End Filter: Homebrew, $5
  - Including preamp circuit: $7
- USB Power Adaptor: $10
- SMA Connectors, adaptors: $10
- **Total: $96**
  - Not including antenna and coax

$22 Software Defined Receiver

RTL-SDR BLOG v3

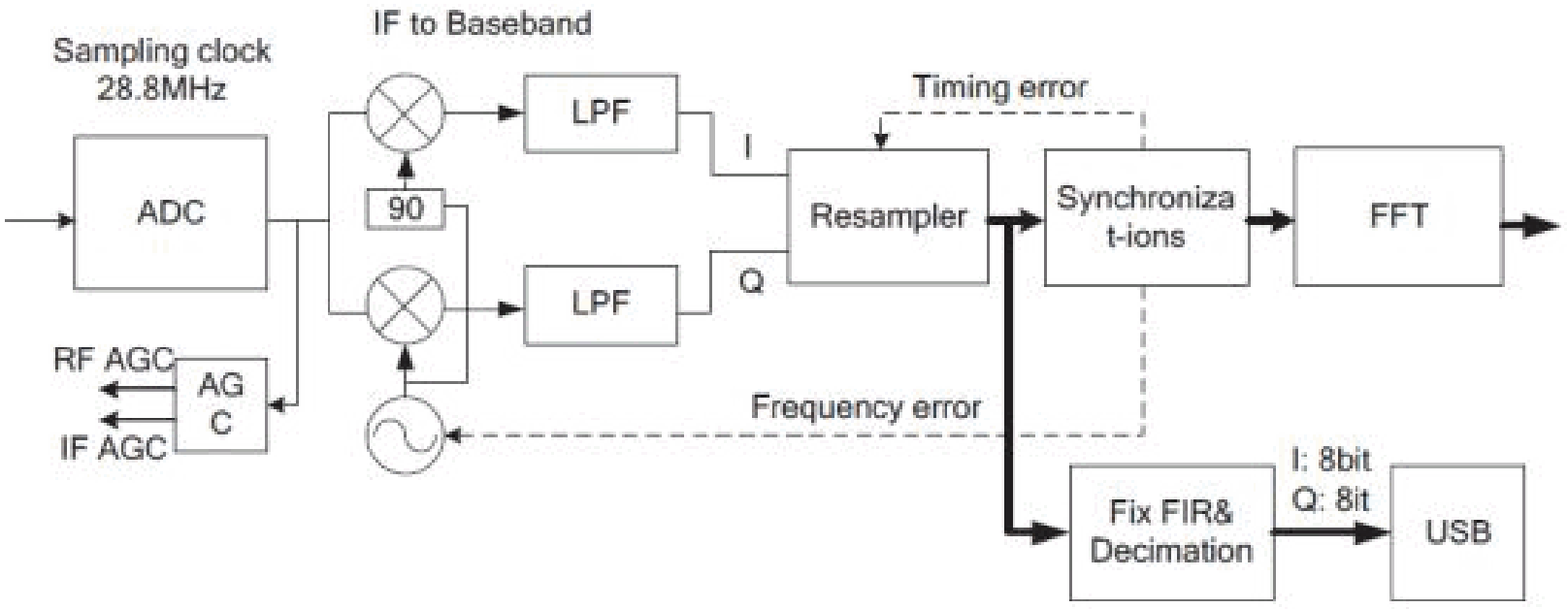
- 1PPM TCXO
- Software-switchable bias tee (for external preamp)
- Direct-sampling option for limited HF operation
  - 28.8 MHz sample clock and non-quadrature output make external filtering mandatory, due to Nyquist aliasing above 14.4 MHz
- 8-bit A/D converter limits receiver dynamic range
- Actually works quite well for 30-meter (10 MHz) band
- https://www.amazon.com/RTL-SDR-Blog-RTL2832U-Software-Defined/dp/B0129EBDS2

WHAT MAKES OUR RTL-SDR V3 BETTER THAN OTHERS?

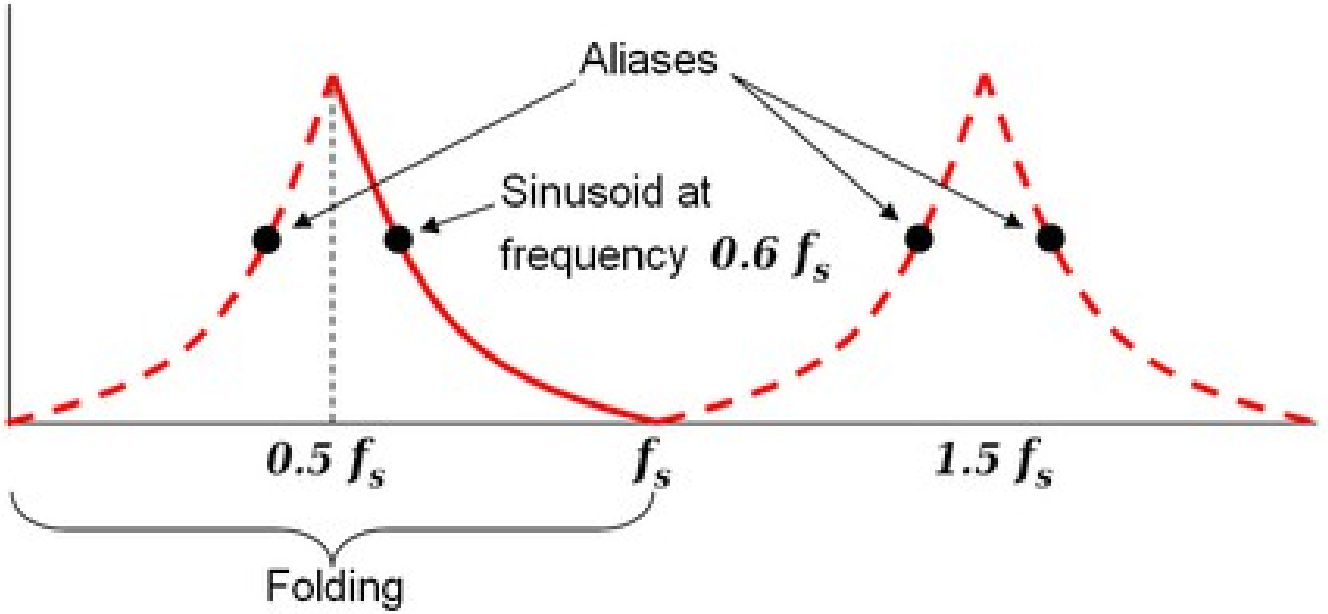


SDR Sampling and Conversion

RTL2832 schematic (A/D converter, digital processor, USB slave interface)

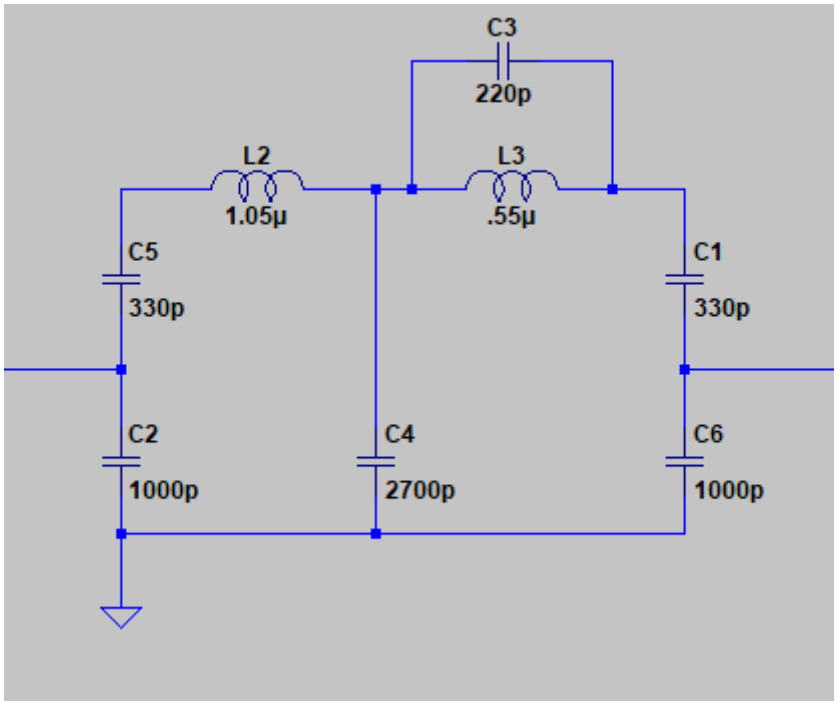


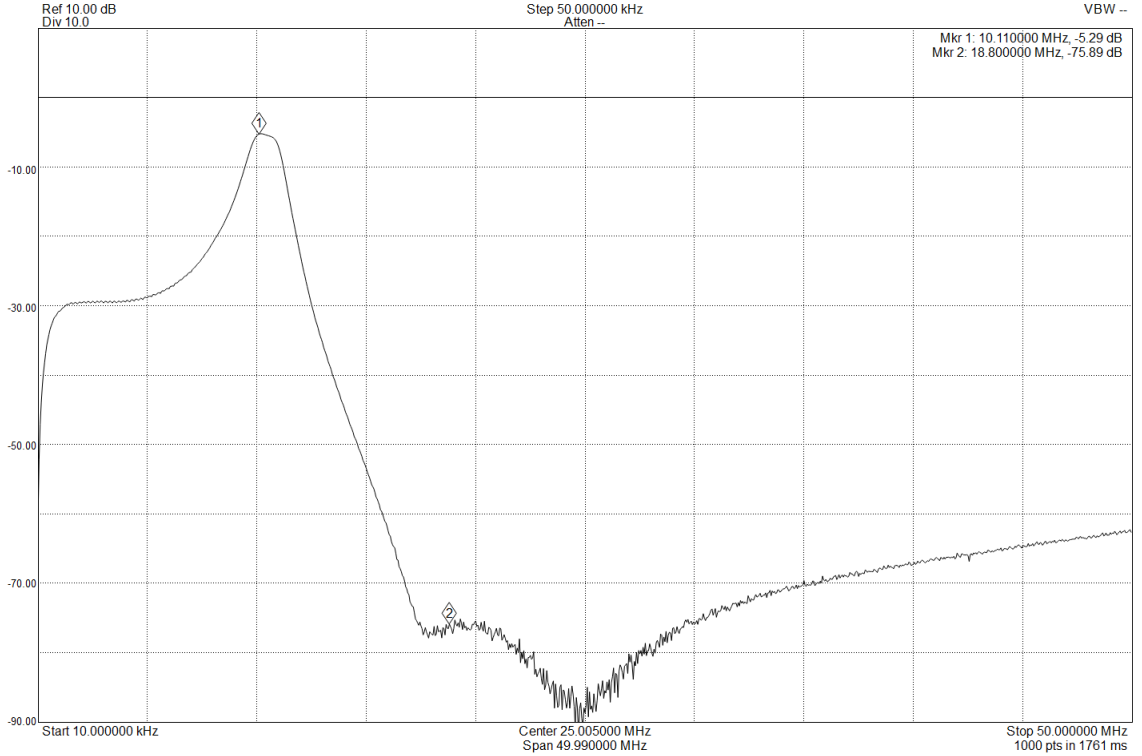
Spurious Responses due to Aliasing



- 28.8 MHz sample clock, 14.4 MHz Nyquist frequency
- Tuned to 10.0 MHz:
  - Alias at 18.8 MHz, 38.8 MHz, 47.6 MHz (etc.)
- Tuned to 14.0 MHz:
  - Alias at 14.8 MHz, 42.8 MHz, 43.6 MHz (etc.)
- RTL-SDR has no useful filtering at these frequencies

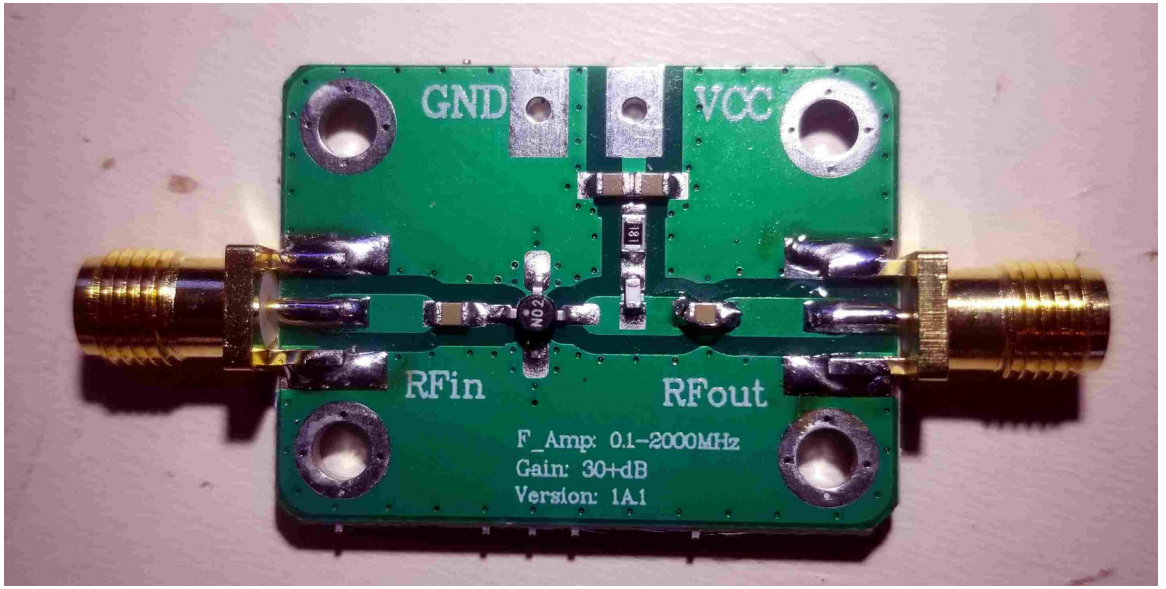
Front-End Filter





- 10 MHz bandpass filter with 18 MHz Notch
- 6dB loss due to design and component Q
- -70dB at first alias frequency
- Values and design may be different than shown

$11.00 RF Preamp



- 100 Khz – 2 GHz, 30 dB gain
- With jumper (or resistor) bridging output capacitor, RTL-SDR can provide power via bias-T
- Preamp makes up for loss in front-end filter
- https://www.amazon.com/gp/product/B01N2NJSGV

DSP on Raspberry Pi

CSDR

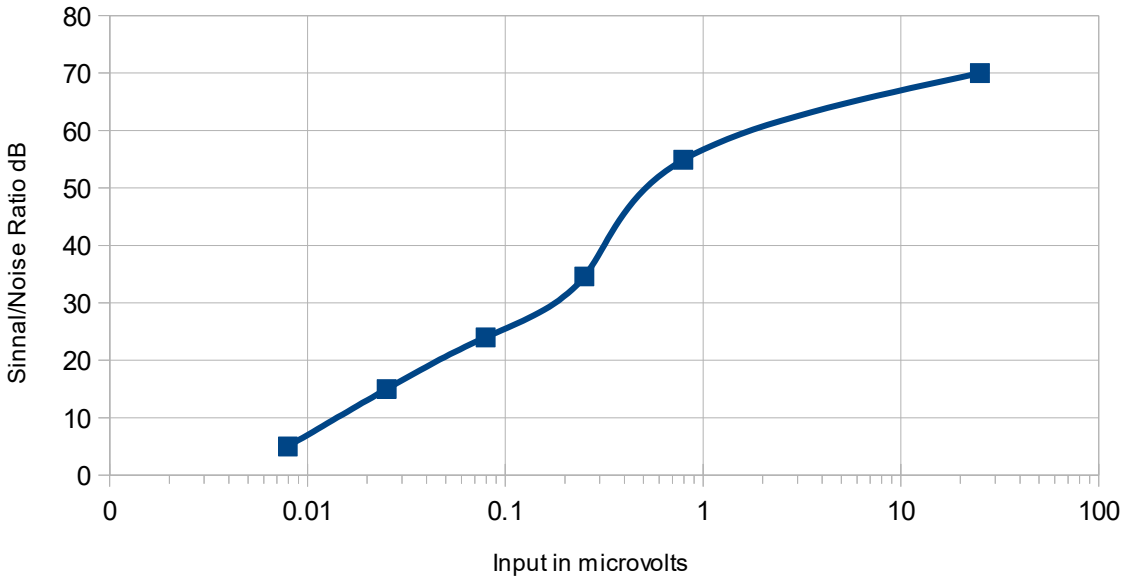
- **csdr** is a command line tool to carry out DSP tasks for Software Defined Radio.
- It can be used to build simple signal processing flow graphs, right from the command line.
- https://github.com/simonyiszk/csdr
- Need to play with time synchronization to compensate for delay in DSP pipeline
  - Using “Chrony” for this

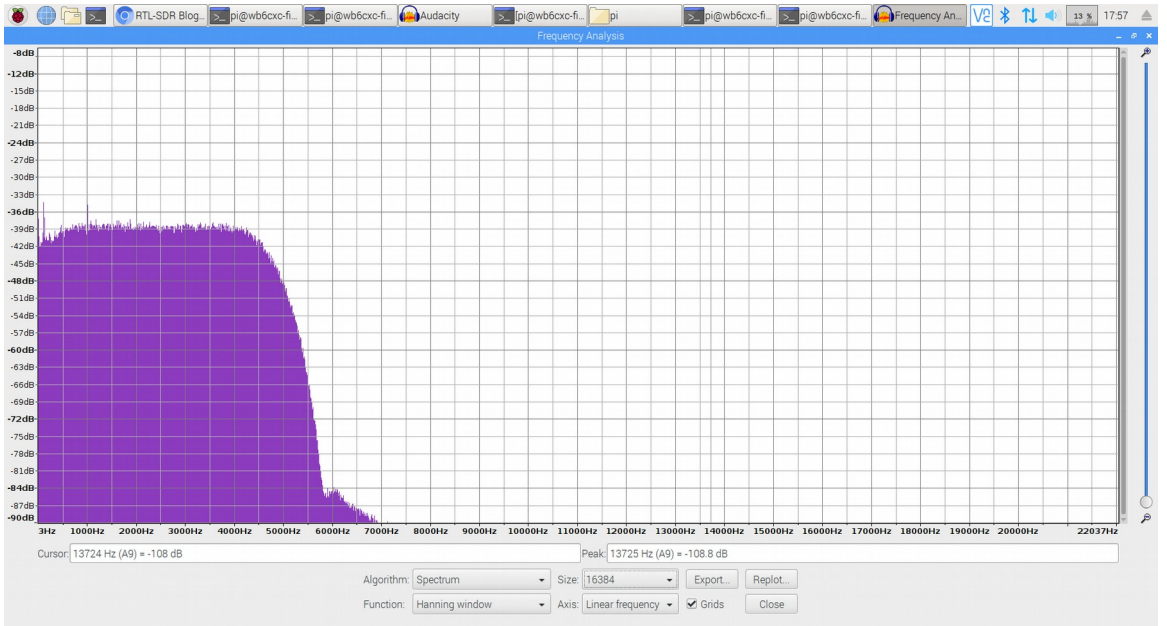
Configuration of RTL-SDR v3 and DSP SSB Receiver

```
#!/bin/bash
if [ $# -eq 1 ]
then
    freq=$1
    echo "frequency = $freq"
    rtl_biast -b 1
    rtl_sdr -s 1200000 -f `python -c "print float($freq + 100000)"` -D 2 - |
    csdr convert_u8_f  |
    csdr shift_addition_cc 0.08333333333333 |
    csdr fir_decimate_cc 25 0.05 HAMMING |
    csdr bandpass_fir_fft_cc 0 0.5 0.05 |
    csdr realpart_cf |
    csdr agc_ff |
    csdr limit_ff |
    csdr convert_f_s16 |
    aplay -v -r 48000 -f S16_LE -
else
    echo "rtl-sdr-usb freq_in_Hz"
fi
```

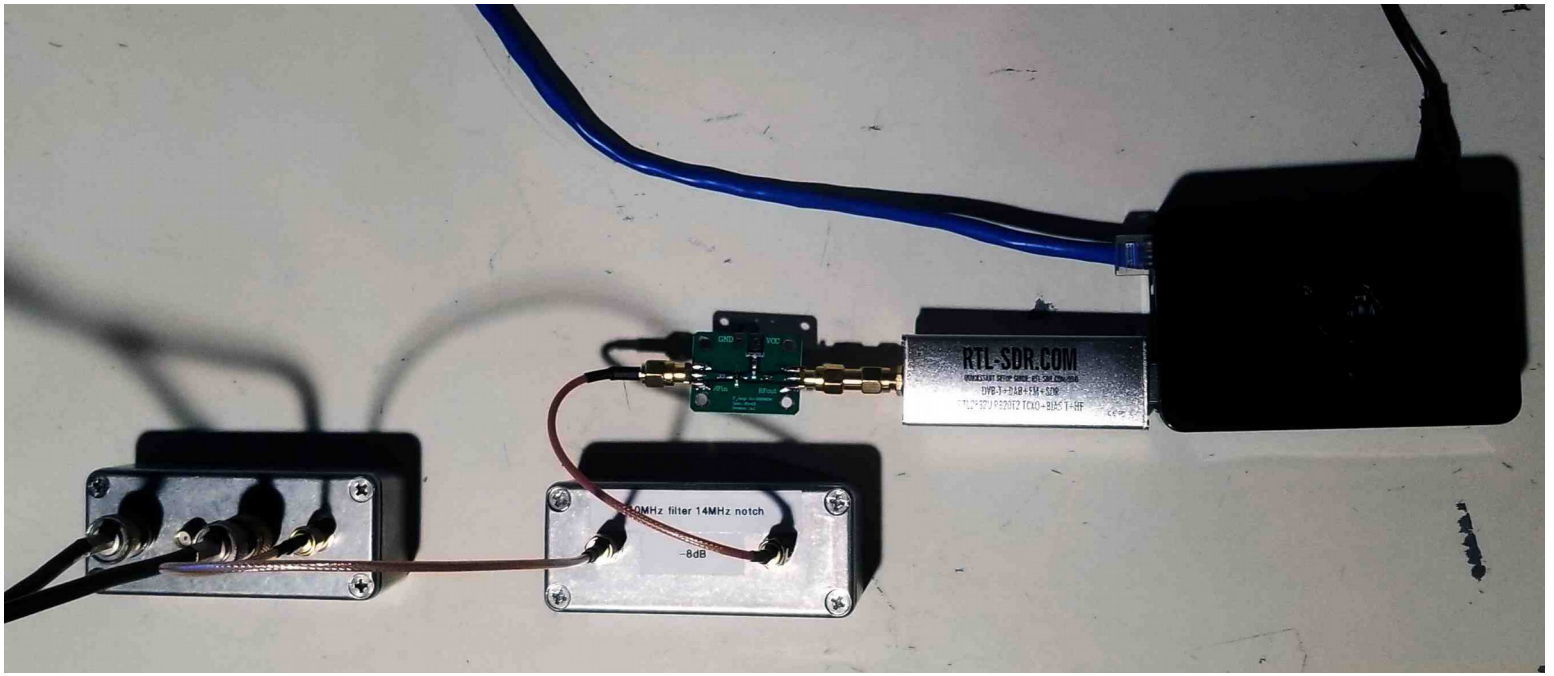
Receiver Performance

- Receiver tuned to 10.000 MHz, signal at 10.001 MHz giving 1KHz beat note
- Noise floor around 0.005 uV (useful with low-gain antenna)
- Using “Audacity” program for SNR analysis
- Still need to do strong-signal overload (IMD) measurements





Raspberry Pi JS8 Receive Gateway



- RPi running SDR software and JS8Call
- Reports received signals to pskreporter.info
- Forwards APRS messages to APRS-IS
- Uses 15% - 30% CPU cycles of Rpi 3 B
  - No heatsink required
- That box on the left is a passive antenna splitter for A/B receiver testing

Paper-Clip Transmitting Antenna

- Receiver about 100 yards from transmitter.
- <1W output (if matched)
- Sending JS8 APRS email:

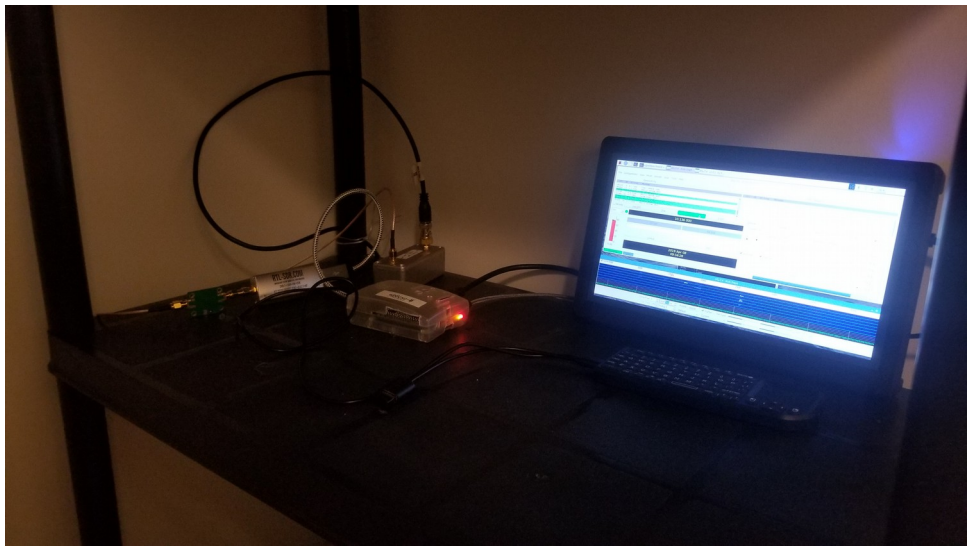
APRS::EMAIL-2 :ME HELLO WORLD{02}

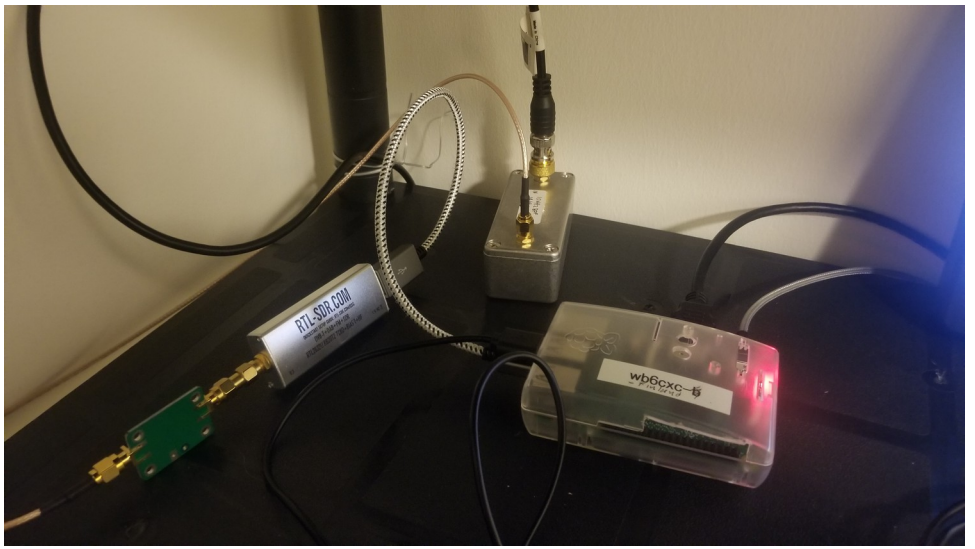
- “ME” is a shortcut for my email address
- This takes four JS8 frames to send (4 x 15 seconds)
- http://www.aprs-is.net/email.aspx

WB6CXC/FIN

Rcv-only gateway in Finland

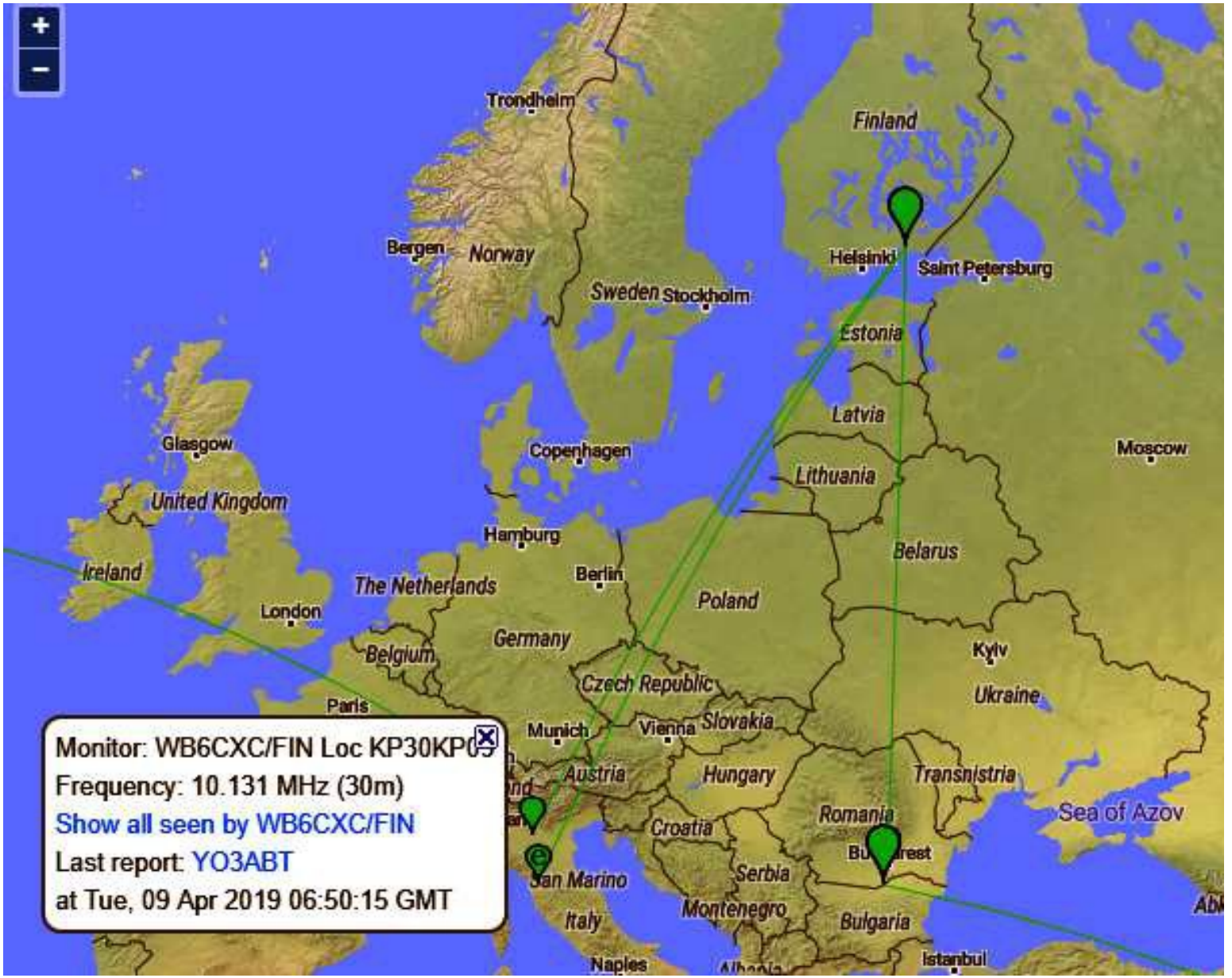
- Summer house in Finland, about 130km ENE of Helsinki
- Station in upstairs utility closet
- Indoor antenna, 30m dipole tacked along ceiling trim
- Motorcycle battery backup





WB6CXC/FIN

Rcv-only gateway in Finland



What Next?

- This gateway receiver design could be used below 10 MHz with the appropriate front-end filter
- 14 MHz is uncomfortably close to the 14.4 MHz Nyquist frequency, would require a very fancy anti-aliasing filter
- 18 MHz and 21 MHz operation should be practical, will have sideband inversion (which can be fixed in the demodulation software)
- A full transceiver design will probably not use a SDR receiver, but instead a use hybrid analog / digital approach

Gateway Transceiver

- Receiver
  - Using “Tayloe Quadrature Sampling Mixer”
  - Analog low-pass filters with matched gain and delay
  - Two-channel A-D Converter
  - Software SSB demodulation similar to SDR gateway
- Transmitter
  - Direct digital generation of 8-FSK JS8 signal
  - 10W Class-E power amplifier, filters
- A single clock generator chip can provide receiver and transmitter clocks

Links

- http://js8call.com/
- https://www.rtl-sdr.com/tutorial-setting-up-a-low-cost-qrp-ft8-jt9-wspr-etc-monitoring-station-with-an-rtl-sdr-v3-and-raspberry-pi-3/
- https://github.com/simonyiszk/csdr
- http://www.aprs-is.net/email.aspx
- https://www.amazon.com/RTL-SDR-Blog-RTL2832U-Software-Defined/dp/B0129EBDS2
- https://www.amazon.com/gp/product/B01N2NJSGV

JS8 and JS8Call

Telemetry and Messaging

A JS8 to APRS Gateway Receiver

Paul Elliott / WB6CXC

In my previous presentation I mentioned the telemetry drift-buoy I wanted to design and set free to roam the oceans of the world (and I still want to!) The buoy will have a low-power (<10W) transmitter, sending data in the HF ham bands, probably 30 meters (10 MHz.)

In order for this to work, there needs to be a network of receivers that will pick up these transmissions and forward the data back to me, or at least to a place where it can be retrieved.

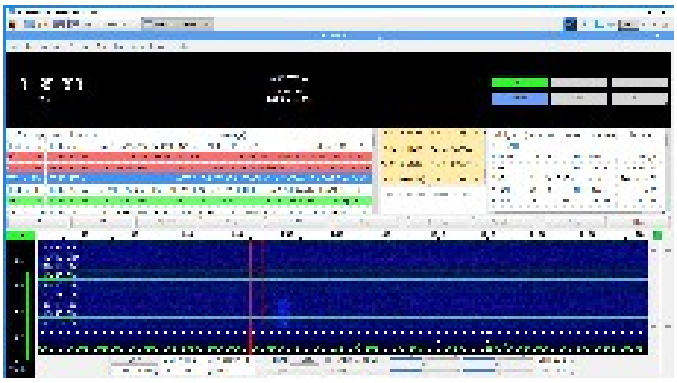
HF Telemetry

- Drift-buoy project needs a good way to transmit data to shore-based server.
- 40 meters, 30 meters, and 20 meters are appropriate bands for worldwide paths.
- 30m HF APRS has necessary features, but coding is far from optimum for error-prone weak-signal conditions.
- HF WSPR has good low-level characteristics and a good worldwide receiving / reporting infrastructure. It is being used for some telemetry but with a very limited and inflexible data format.
- FT8 has good low-level characteristics but a poor receiving / reporting infrastructure. It also has very limited data capability.
- **JS8Call** is a new mode, derived from FT8
  - It includes Forward Error Correction and is optimized for weak-signal conditions.)
  - It provides a flexible data format and APRS interface.

HF APRS, and WSPR were possible candidates, but each had serious drawbacks.

JS8 / JS8-Call has now come on the scene in a big way, and looks like a winner.

JS8Call



- Created by Jordan Sherer / KN4CRD
- Runs on Windows, Mac OSX, Raspberry Pi, Desktop linux
- The program is now in general release, see www.js8call.com

JS8Call

- JS8Call was previously named FT8Call.
- Proposed July 2017, first program release July 2018
- JS8Call uses a custom FT8 modulation called JS8 (Jordan Sherer designed 8-FSK modulation). This is the base RF transport.
- JS8Call has a “directed calling” protocol laid on top of the base RF transport to support free-form and directed message passing.
- Uses a keyboard messaging style interface.
- Provides API for remote and programmatic interface.
- **Supports messaging to APRS**
  - **Position report, telemetry (etc), text msg, email.**

While JS8 modulation and coding is very similar to FT8, the two modes are not compatible. This is by design, since FT8 is built around fixed fields, designed for “contest style” QSOs, where a minimum set of standardized fields can be carried over a single 15-second frame.

JS8 instead provides a few fixed fields, but the bulk of the message space is available for free-format text, and allows multiple frames to be chained in order to send messages of an arbitrary size.

Using JS8 to send APRS email:
@ALLCALL APRS::EMAIL-2 :ME msg body{xx}

Fixed-length fields are used for APRS header

JS8Call Features

- Lots of good features for actual real-time free-form QSOs and general communications.
- Has mailbox capability so messages can be stored and automatically delivered.
- Messages can be relayed from station to station.
- Stations can be queried for their “heard” list, making relay routes easier to manage.
  - No automatic routing (yet)
- Periodic “Heartbeat” transmissions, and auto-ACKs help provide network status information.

Of course you can always just call CQ and talk (type) to each other. Most of the JS8 activity is just this.

JS8Call Suitability

- JS8Call was created for human keyboard-keyboard contacts, but it also has the necessary fundamental characteristics to provide a reliable method for low-speed digital communications of arbitrary data.
- But can it provide a receiving infrastructure? Will it wither on the vine as many previous digital modes have?
- FT8 is the most popular HF digital mode in use now, and it appears that many FT8 users are trying JS8. More than 10,000 hams have downloaded the latest version of the JS8Call program.
- A mix of occasional operators and 24/7 gateway stations should provide good coverage. Not all operators will be relaying APRS, but some will.

JS8-Call uses built-in Huffman compression for plain upper-case text, optimized for standard English letter frequency as used in ham QSOs. There is also some sort of “phrase” compression – the details are unclear (actually, it's all revealed in the public-domain code, I just haven't studied it.)

The Huffman compression means that random data (such as telemetry values) will have to be mapped into the 44-character Huffman table, resulting in a slight expansion rather than compression. Mapping random data into the first 38 table values looks to be optimum, about an 8% expansion.

Huffman compression uses fewer bits for more commonly used characters. Much like Morse code.

JS8 Activity

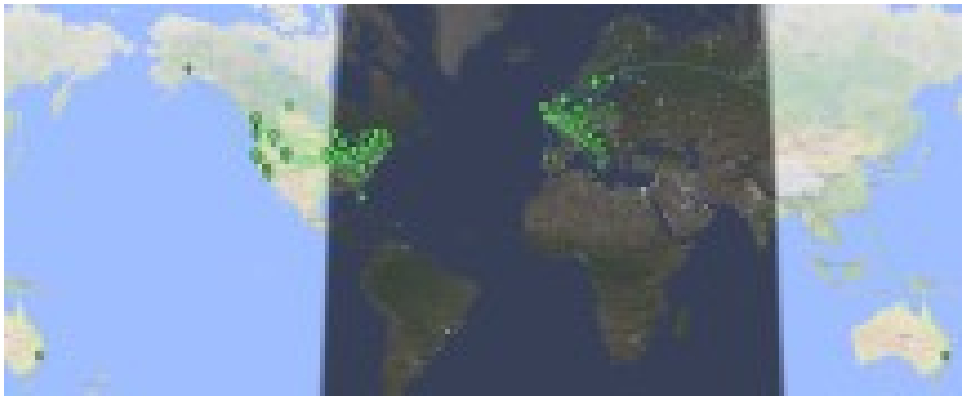
- All bands, 24-hours
- 40 meters most active, then 20 meters

Looking at www.pskreporter.info, we can see a recent day's worth of JS8 activity. Not all stations report to pskreporter, but many do.

I don't know how many also forward to APRS, but I assume it's a small percentage since few use this service and a passcode is needed to do this. The passcode is easy to get, but it is still an additional step

The APRS passcode does not provide for security or authentication. Other means are required for this. For telemetry and remote command, some method of authentication should used.

30 Meter JS8 Activity



- 24-hours
- Not a whole lot of activity, but still useful

30 Meter WSPR Activity

- 24-hours
- We can see that propagation isn't the issue

Building a JS8 Gateway Station

- More 24/7 Stations!
- TX/RX is great, but RX-only is still useful
- Goal: Cheap, Easy, Good
  - Pick any two?
- 30 Meters (10.130 MHz USB)
- Cheap / Easy / (Good Enough) Rcv-Only
- I now have three 24/7 Cheap/Easy receive-only gateway stations on 30 meters: Friday Harbor WA, Occidental CA, and Anjala Finland

So having more JS8 30-meter activity would be good. For my telemetry needs, having more receivers that gateway to APRS is the big thing, but without people transmitting, few will put up receivers.

A full gateway transceiver would be great, and it's easy to put one together using a ham transceiver and a computer. But few people want to tie up their main ham rig for a 24/7 gateway.

How to build a gateway that could be put into 24/7 service? Here I describe an inexpensive SDR receiver solution.

A low-cost full transceiver design is also slowly being developed.

Receive-Only
Cheap / Easy / (Good Enough)

- RTL-SDR Blog Version 3: $22
- Raspberry Pi 3B: $34
- 8G Micro SD Card: $4
- RF Preamp: $11
- 10 MHz Front-End Filter: Homebrew, $5
  - Including preamp circuit: $7
- USB Power Adaptor: $10
- SMA Connectors, adaptors: $10
- **Total: $96**
  - Not including antenna and coax

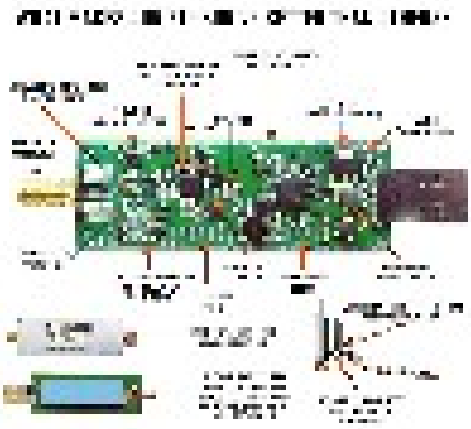
There are better-performing SDR receivers out there (SDRPlay, Funcube, others), but these cost >$100. I wanted to see if the much cheaper RTL-SDR unit might work acceptably.

As with my Raspberry Pi WSPR transmitter, the antenna is perhaps the most expensive part of the project. Fortunately, I have hundreds of feet of old #10 wire and coax, and a box of connectors. And trees.

$22 Software Defined Receiver

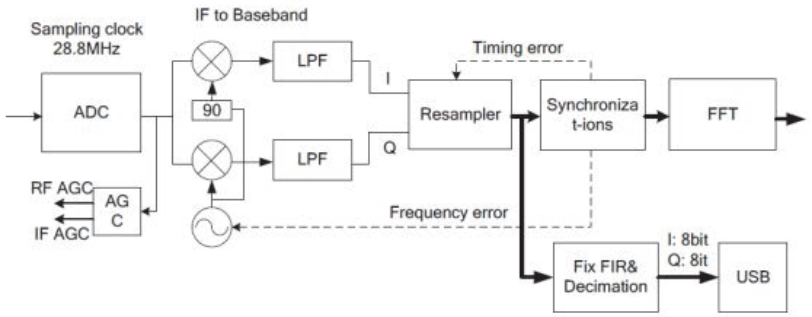
RTL-SDR BLOG v3

- 1PPM TCXO
- Software-switchable bias tee (for external preamp)
- Direct-sampling option for limited HF operation
  - 28.8 MHz sample clock and non-quadrature output make external filtering mandatory, due to Nyquist aliasing above 14.4 MHz
- 8-bit A/D converter limits receiver dynamic range
- Actually works quite well for 30-meter (10 MHz) band
- https://www.amazon.com/RTL-SDR-Blog-RTL2832U-Software-Defined/dp/B0129EBDS2



SDR Sampling and Conversion

RTL2832 schematic (A/D converter, digital processor, USB slave interface)

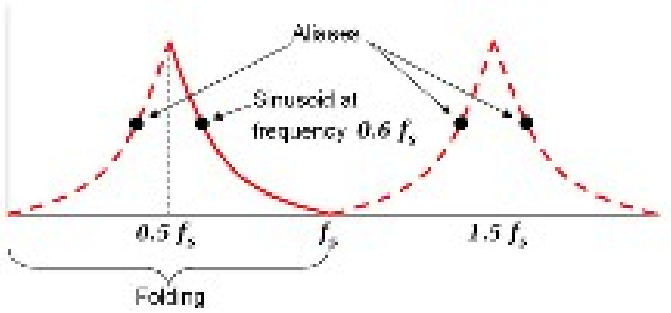


This is the analog to digital converter and post-conversion processing inside the back-end chip used in the RTL-SDR (and most other SDRs). There is also a front-end down-converter in these SDRs, but in the direct-sampling mode this downconverter is bypassed and the RF is routed directly to the A/D converter, which samples the input at 28.8 MHz.

The converter output is re-sampled by a quadrature mixer, and the I/Q paths are further filtered. They are then further down-sampled, filtered, and sent to the USB interface. USB data rate is set to 1.2 Mbytes/second.

The 28.8 MHz sampling gives rise to spurious responses (Nyquist)

Spurious Responses due to Aliasing



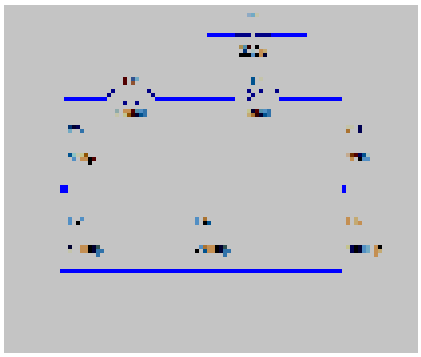
- 28.8 MHz sample clock, 14.4 MHz Nyquist frequency
- Tuned to 10.0 MHz:
  - Alias at 18.8 MHz, 38.8 MHz, 47.6 MHz (etc.)
- Tuned to 14.0 MHz:
  - Alias at 14.8 MHz, 42.8 MHz, 43.6 MHz (etc.)
- RTL-SDR has no useful filtering at these frequencies

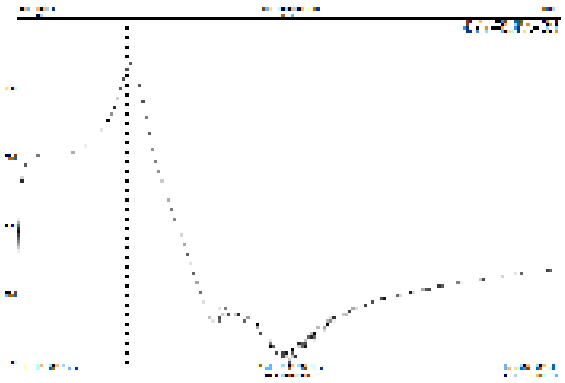
Alias frequencies are at (Fs * x) +/- Ftune, x = 1,2,3,...

Fs = sample clock frequency
Ftune = frequency receiver is tuned to

You can also use one of these alias frequencies as the desired signal – this is called “undersampling”

Front-End Filter





- 10 MHz bandpass filter with 18 MHz Notch
- 6dB loss due to design and component Q
- -70dB at first alias frequency
- Values and design may be different than shown

To reduce potential spurious responses, I built a front-end filter. I decided to give it a bandpass centered at 10.1 MHz, with a sharp null at the first image of 18.8 MHz. Given the fairly low dynamic range of the 8-bit A-D converter in the SDR, it seemed a good idea to at least somewhat filter out the lower frequencies as well.

Due to my construction techniques, and possible component resonances, the filter also showed a pronounced null at 25 MHz. It wasn't due to the chip inductors; changing these to hand-wound toroids didn't change a thing. No harm done, but the higher-frequency rejection probably suffers because of this.

The filter response plot may be from a design with one more section. I tried a lot of filter designs.

$11.00 RF Preamp

- 100 Khz – 2 GHz, 30 dB gain
- With jumper (or resistor) bridging output capacitor, RTL-SDR can provide power via bias-T
- Preamp makes up for loss in front-end filter
- https://www.amazon.com/gp/product/B01N2NJSGV

This is a Chinese board with a single Darlington-pair preamp. I jumpered the output capacitor so the SDR bias-t could directly power the preamp. Depending on the specific preamp used, a bypassed series resistor may be needed for bias-t powering.

The preamp could easily be included in the filter circuit. Parts-cost: a couple of dollars, more than offset by the fewer connectors required.

DSP on Raspberry Pi

CSDR

- **csdr** is a command line tool to carry out DSP tasks for Software Defined Radio.
- It can be used to build simple signal processing flow graphs, right from the command line.
- https://github.com/simonyiszk/csdr
- Need to play with time synchronization to compensate for delay in DSP pipeline
  - Using “Chrony” for this

Configuration of RTL-SDR v3 and DSP SSB Receiver

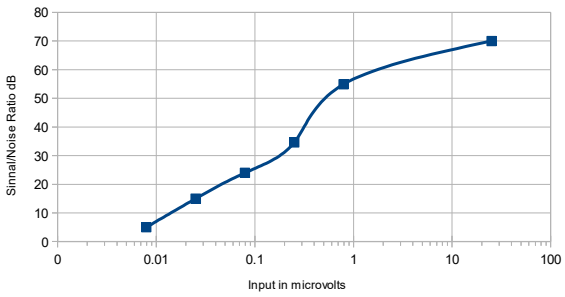
```
#!/bin/bash
if [ $# -eq 1 ]
then
   freq=$1
   echo "frequency = $freq"
   rtl_biast -b 1
   rtl_sdr -s 1200000 -f `python -c "print float($freq + 100000)"` -D 2 - |
   csdr convert_u8_f  |
   csdr shift_addition_cc 0.08333333333333 |
   csdr fir_decimate_cc 25 0.05 HAMMING |
   csdr bandpass_fir_fft_cc 0 0.5 0.05 |
   csdr realpart_cf |
   csdr agc_ff |
   csdr limit_ff |
   csdr convert_f_s16 |
   aplay -v -r 48000 -f S16_LE -
else
   echo "rtl-sdr-usb freq_in_Hz"
fi
```

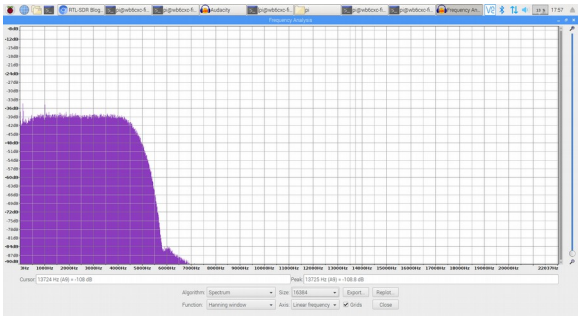
This is largely taken from a tutorial on the RTL-SDR website:
https://www.rtl-sdr.com/tutorial-setting-up-a-low-cost-qrp-ft8-jt9

The design in the tutorial used “ncat” to send the SDR output to multiple demodulating applications. I simplified it for single-channel use. Also, I had dropout problems with the audio piping using Pulseaudio. Instead, I use “aplay”. I also narrowed the digital bandpass filter of the audio SSB demodulator, giving it a 5 Khz cutoff.

Receiver Performance

- Receiver tuned to 10.000 MHz, signal at 10.001 MHz giving 1KHz beat note
- Noise floor around 0.005 uV (useful with low-gain antenna)
- Using “Audacity” program for SNR analysis
- Still need to do strong-signal overload (IMD) measurements



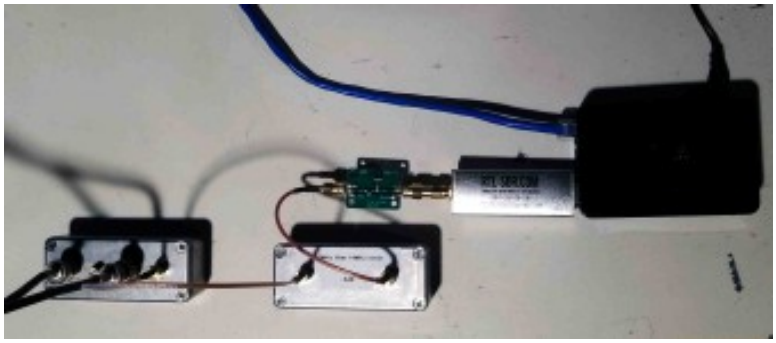


While this sensitivity sounds amazingly good, it's still far from the fundamental limits of a 50-Ohm resistor at room-temperature. Still, perhaps I don't really need the preamp. Early tests made me think the SDR was a bit “deaf”, but I should re-test this to verify. Excess gain does no good, and can cause overload problems.

I really want a “SNR” program so I can see real-time results, rather than sample the signal with Audacity and then use the FFT capability to see what I got.

I also want a dual-signal source for overload measurements. Shouldn't be too hard to put together.

Raspberry Pi JS8 Receive Gateway



- RPi running SDR software and JS8Call
- Reports received signals to pskreporter.info
- Forwards APRS messages to APRS-IS
- Uses 15% - 30% CPU cycles of Rpi 3 B
  - No heatsink required
- That box on the left is a passive antenna splitter for A/B receiver testing

With a Power Over Ethernet adaptor and a weatherproof case, this entire gateway could be located outside at the antenna, fed by an ethernet cable.

For receive-only operation, a short untuned antenna might be sufficient.

Some sort of transient protection on the RF and Ethernet connections would be a good idea.

Using “RealVNC server” on the RPi (included in the Raspbian image), and “RealVNC Client” on the remote computers. No monitor or keyboard needed at the Rpi. Monitor and control from anywhere. Up to five servers for free.

Paper-Clip Transmitting Antenna

- Receiver about 100 yards from transmitter.
- <1W output (if matched)
- Sending JS8 APRS email:

APRS::EMAIL-2 :ME HELLO WORLD{02}

- “ME” is a shortcut for my email address
- This takes four JS8 frames to send (4 x 15 seconds)
- http://www.aprs-is.net/email.aspx

Over the air testing of the gateway, using JS8Call on a Win10 PC, connected to an ICOM IC7200 or IC7300 transceiver, with power dialed down to minimum, into a mistuned antenna, or using paper-clip antenna.

Gateway receiver has 10m dipole at a small distance from transceiver antenna.

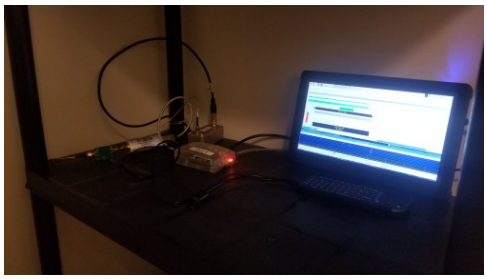
Also used antenna feeding a splitter, gateway . RPi on one port, Icom / Win computer on other. Ran both receivers for an hour, compared log results. Gateway did as well as Icom.

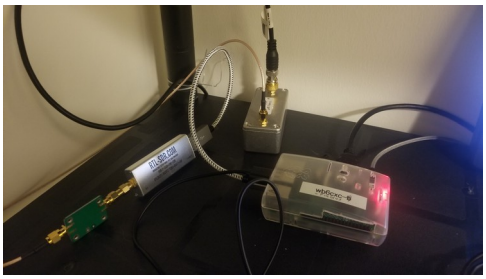
Some interesting differences in SNR reports, Icom better with strong signals, SDR better with weak ones. Each of my three receiver locations are in a quiet RF environment.

WB6CXC/FIN

Rcv-only gateway in Finland

- Summer house in Finland, about 130km ENE of Helsinki
- Station in upstairs utility closet
- Indoor antenna, 30m dipole tacked along ceiling trim
- Motorcycle battery backup





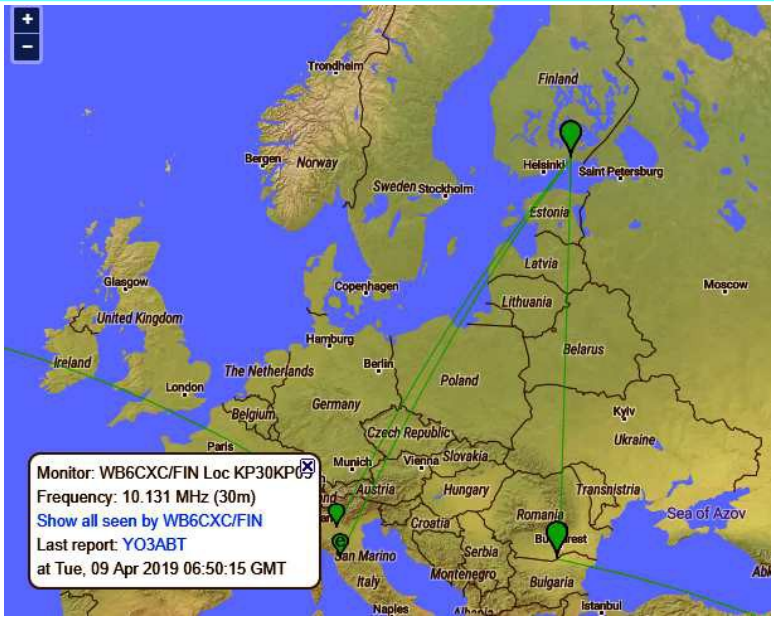
Installed gateway in Finland on May 8, 2019. Probably not a good spot for receiving a Pacific Ocean buoy signal, but I wanted to see how it worked anyway, and somebody might find it useful.

Antenna snakes back and forth, tacked up so it won't annoy my wife too much.

I had trouble with USB power to RPi, discovered that not all micro-USB cables and power adaptors are created equal. Found some that work, but eventually replaced plug-in power adaptor with motorcycle battery / charger / 12V USB adaptor.

WB6CXC/FIN

Rcv-only gateway in Finland



Receiver shows up in pskreporter, first stations received were in Italy and Romania

I can monitor and control this station from anywhere in the world using VNC and the internet.

“WB6CXC/FIN” is not a real callsign, just an arbitrary station identifier for pskreporter and APRS. This is not a transmitter so no official callsign or reciprocal license arrangement is needed.

What Next?

- This gateway receiver design could be used below 10 MHz with the appropriate front-end filter
- 14 MHz is uncomfortably close to the 14.4 MHz Nyquist frequency, would require a very fancy anti-aliasing filter
- 18 MHz and 21 MHz operation should be practical, will have sideband inversion (which can be fixed in the demodulation software)
- A full transceiver design will probably not use a SDR receiver, but instead a use hybrid analog / digital approach

For an extra $100, a better SDR would allow operation over the full range of ham bands.

The direct digital FSK synthesis method is probably the least expensive approach for QRP transmitter design. Some harmonic filtering is required, but if the synthesizer divider values are chosen carefully the signals are otherwise clean enough.

The Si5351 clock synth chip has three outputs, one used for the transmitter, and the remaining two used in the receiver circuit. This leads to an inexpensive transceiver with decent performance.

Gateway Transceiver

- Receiver
  - Using “Tayloe Quadrature Sampling Mixer”
  - Analog low-pass filters with matched gain and delay
  - Two-channel A-D Converter
  - Software SSB demodulation similar to SDR gateway
- Transmitter
  - Direct digital generation of 8-FSK JS8 signal
  - 10W Class-E power amplifier, filters
- A single clock generator chip can provide receiver and transmitter clocks

The Tayloe mixer is a high dynamic-range zero-IF (direct conversion) design, using analog transmission gates driven by quadrature clocks It has analog quadrature outputs (I/Q) which can be converted to USB or LSB outputs using analog or digital mixing techniques. This design would use digital methods. (Tayloe doesn't have to be zero-IF, but usually is.)

The chosen A-D converter samples at 192 Khz, so matched analog filters are required at the input to eliminate aliasing.

Power Over Ethernet may have difficulty feeding a 10W transmitter.

Links

- http://js8call.com/
- https://www.rtl-sdr.com/tutorial-setting-up-a-low-cost-qrp-ft8-jt9-wspr-etc-monitoring-station-with-an-rtl-sdr-v3-and-raspberry-pi-3/
- https://github.com/simonyiszk/csdr
- http://www.aprs-is.net/email.aspx
- https://www.amazon.com/RTL-SDR-Blog-RTL2832U-Software-Defined/dp/B0129EBDS2
- https://www.amazon.com/gp/product/B01N2NJSGV