

2020 ARRL/TAPR

Digital Communications Conference



Virtual DCC

September 11-13, 2020



**2020 ARRL/TAPR
Digital Communications Conference**

Virtual DCC
September 11–13, 2020



TAPR, 1 Glen Ave., Wolcott, CT 06716 USA www.tapr.org
ARRL, 225 Main St., Newington, CT 06111 USA www.arrl.org

Copyright © 2020

The American Radio Relay League, Inc.

Copyright secured under the Pan-American Convention

International Copyright secured

All rights reserved.

No part of this work may be reproduced in any form except by written permission of the publisher. All rights of translation reserved.

Printed in USA.

Quedan reservados todos los derechos.

ISBN: 978-1-62595-125-0

First Edition

**ARRL and TAPR – 39th Annual
Digital Communication Conference
September 11-12, 2020 – Virtual**

Greetings!

Welcome to the 39th annual ARRL and TAPR Digital Communications Conference (DCC).

This year we are trying something completely different. We are conducting the DCC online and virtual for the very first time.

We are excited to produce an online and virtual conference. It opens up so many opportunities. The most exciting one is the ability for presenters from all around the world to participate. One of the realities is that everyone will be in different time zone. For some, they will be attending late in the evening or early in the morning. We sincerely appreciate their efforts.

The year 2020 will go down in history as a very strange one indeed. The COVID-19 virus has upended everyone's lives, all around the world. Several conferences were moved to online meetings and webinars. We have been watching and learning from each of these so that we can bring you an exciting and informative technical conference.

In your hands (or perhaps on the computer screen) is the proceedings for this year's DCC. It is grand compilation of several experimenter's hard work. If one thing the COVID-19 virus has given us, that is more time operating, building, and experimenting. I think that shows in the over 230 pages of conference proceedings.

We hope you enjoy this year's virtual DCC. Let us know what you think. It may be a trend we see going forward into the future.

73,

Steven Bible, N7HPR
President, TAPR

Table of Contents

Welcome; Steven Bible, N7HPR, President, TAPR.....	iii
Timing and Location Performance of Recent u-blox GNSS Receiver Modules; John Ackermann, N8UR.....	1
APRS Performance and Limits; Marco Bersani, IK2PIH	45
Improved Layer 2 Protocol; Nino Carrillo, KK4HEJ.....	77
HF Propagation Measurement Techniques and Analyses; Steve Cerwin, WA5FRF.....	90
QMesh: A Synchronized, Flooded Mesh Network Protocol for Voice; Dan Fay, PhD, KG5VBY.....	113
A Comparison of Affordable, Self-Assembled Software-Defined Radio Receivers Using Quadrature Sampling Down-Conversion; Caleb Froelich; Dr. Rob Frohne, KL7NA; Konrad McClure; Joshua Silver, and Jordyn Watkins, KN6FFS.....	130
Forward Error Correction and Pictures from Mars; David Garner, N6WY	163
Aids to the Presentation and Analysis of WSPR Spots: TimescaleDB database and Grafana; Gwyn Griffiths, G2ZIL, and Rob Robinet, AI6VN.....	177
Packet Compressed Sensing Imaging (PCSI): Robust Image Transmission over Noisy Channels; Scott Howard; Grant Barthelmes; Cara Ravasio, Lisa Huang, Benjamin Poag, and Varun Mannam.....	185
Digital Signal Processing: I2S in ESP32; Anthony Le Cren, KF4GOH	200
Continued Lessons from the RF-Seismograph; Alex Schwarz, VE7DXW.....	207
How to Kill Packet-Radio & APRS? Come to Serbia! (Part 3); Miroslav “Misko” Skoric, YT7MPB	214
Build a Mobile Mesh Tower Fleet; Erik Westgard, NY9D	223
Current Status Report of FX.25 KISS TNC Development; Kazuhisa Yokota, JN1DFF and Masaaki Yonezawa, JE1WAZ	231

Timing and Location Performance of Recent u-blox GNSS Receiver Modules

by John Ackermann N8UR¹

1. INTRODUCTION

In the past few years several companies have introduced small GNSS² modules intended for OEM timing and positioning applications. u-blox³ AG, a Swiss corporation, is perhaps the most well-known of these and has introduced several generations of receivers with increasing capabilities. This paper presents experimental data on the seven u-blox modules listed in Table 1.⁴ More detailed information about the capabilities of the units is contained in Appendix 1.

Model	Comments
LEA-M8F	Frequency and timing series (disciplined frequency source)
NEO-M8N	Navigation series (low cost)
NEO-M8P	Positioning series (internal RTK engine)
NEO-M8T	Timing series (dual timepulse)
NEO-M9N	Navigation series (low cost, still L1 only)
ZED-F9P	Positioning series (L1/L2, internal RTK engine)
ZED-F9T	Timing series (L1/L2, dual timepulse)

Table 1: u-blox modules tested.

The focus of this paper is on the receivers' timing performance, and primarily the stability and other characteristics of its hardware time pulse output. Section 2 describes the timekeeping performance of the receivers. Sections 3 through 6 explore other aspects of timekeeping performance. Section 7 briefly explores how the performance of these receivers can allow a different design philosophy for GPS disciplined oscillators ("GPSDOs"). Finally, Section 8 provides a limited overview of the receivers' positioning performance.

1 The author has no relationship with u-blox AG.

2 These are "Global Navigation Satellite System" or "GNSS" receivers because they support satellite constellations in addition to the United States' Global Positioning System. However, in this paper the term "GPS" is used because, as of this writing, in most timing applications only the GPS constellation is used and the U.S.N.O. master clock serves as the reference. For precise positioning applications, both GPS and to a lesser extent the Russian GLONASS systems are used. The Chinese BeiDou and European Galileo systems are just starting to reach operational use.

3 Based on the corporate web site, the correct name is "u-blox" with hyphen and without capitalization.

4 The author acknowledges and greatly appreciates support from NSF Grants AGS- 2002278 and AGS- 1916690, The University of Scranton, and the New Jersey Institute of Technology Center for Solar-Terrestrial Research.

1.2 Testing methodology

For timing purposes, the key criterion is the quality of the receiver's hardware output that provides an electrical pulse aligned to GPS time. This is usually, but not always 1 pulse per second ("PPS"). U-blox calls this the TIMEPULSE output. The seven receivers were simultaneously and individually evaluated by using a counter to measure the offset in time between their TIMEPULSE outputs and a local PPS signal derived from a stable atomic clock. The variations in the second-by-second values recorded show the noise (sometimes referred to as "jitter") of the timing signal, expressed as quantities of time.⁵

All seven GPS modules, as well as a CNS Systems CNS-Clock II GPS time receiver⁶ used for sanity-checking, were fed from the same dual-frequency GPS antenna (Trimble Zephyr Geodetic⁷) via distribution amplifiers. Their TIMEPULSE outputs were measured simultaneously using a multi-channel counter (TAPR multi-TICC⁸) which has eight independent input channels with resolution of about 60 picoseconds. A high-performance Cesium frequency standard (HP 5071A⁹) served as the reference clock driving the multi-TICC counter. The measurement campaign lasted just under six days and recorded more than 500,000 samples from each of the eight receivers.

The series of timestamps recorded from each receiver's TIMEPULSE output constitutes a record of the time offset, or phase, of that output relative to the reference clock. The timestamp data was captured to text files and processed with the widely-used TimeLab¹⁰ time and frequency analysis software written by John Miles. The figures presented below, unless otherwise stated, were rendered by TimeLab after the applicable phase records were loaded and processed.

For some of the measurements reported in this paper, additional data collection runs were used to enable testing of various configurations.

Unless otherwise stated, for all tests the receivers were set to their default configurations save only for choosing the 0-D (timing) solution mode where available. No compensation was made for cable or other systematic delays, and no attempt was made to measure absolute time accuracy. Receivers that allowed entry of fixed observation coordinates were set to ECEF¹¹ X, Y, and Z values previously derived from post-processed Precise Point Positioning measurements made with a Trimble NetRS¹² receiver using the same antenna as the one used for these measurements.

5 For example, PPS noise might be described as "20 nanoseconds RMS". Somewhat confusingly, it is common to refer to the *amplitude* of noise; it is important to realize that this refers to the magnitude of time variations and not voltage or strength.

6 <https://www.cnssys.com/>

7 <https://kb.unavco.org/kb/article/the-design-and-performance-of-the-zephyr-geodetic-antenna-trimble-publication-241.html>. Note that this antenna is not optimized for GLONASS or BeiDou frequencies; however it appears to receive GLONASS signals reasonably well, and BeiDou performance was not tested.

8 https://github.com/TAPR/TICC/blob/master/multi-ticc/multi-TICC_App_Note_2020-01.pdf

9 Currently sold by Micro-Semi division of MicroChip: <https://www.microsemi.com/product-directory/cesium-frequency-references/4115-5071a-cesium-primary-frequency-standard>

10 <http://www.ke5fx.com/timelab/readme.htm>

11 "Earth-Centered, Earth Fixed" coordinates in meters

12 <https://kb.unavco.org/kb/article/trimble-netrs-resource-page-471.html>

1.3 Interpreting the Results

The time pulse output of a GPS receiver typically runs at a one pulse per second rate, and over the long term tracks the master clock of the GPS satellite constellation, which in turn is traceable to the US. Naval Observatory and National Institute of Standards and Technology time and frequency standards. In other words, a GPS receiver with a timing output provides a replica of the official time, and because frequency can be derived from a series of time measurements, of standard frequency as well. However, in the short term these pulses contain noise, or “jitter,” that results from both limitations in the quality of the GPS solution obtainable, and limitations in the hardware capability of the module. The characteristics of that noise determine the short-term timing capabilities of the unit.

It is possible to plot the PPS values on a graph with the Y axis showing the value and the X axis showing the elapsed time – what is called a “strip chart” recording. This technique gives a qualitative view of the data, and from it one can obtain a sense of performance. However, such a presentation provides mainly a qualitative view and does not lend itself to any but the most basic quantitative evaluation. It is often helpful to look at both the full phase record, as well as a close-up view that shows short-term (second-by-second) changes. This can reveal performance characteristics that are hidden by the limited resolution of a plot showing thousands of data points.

The noise of a clock or oscillator¹³ can be analyzed statistically to help understand the stability of the clock’s tick over varying periods of time. For example, the standard deviation of the series of PPS values could be used to get an idea of their spread.

However, for reasons beyond the scope of this paper, standard deviation is not the best tool to analyze the noise processes at work in clocks. A related statistic called the Allan Deviation (“ADEV”) is designed specifically for frequency stability analysis and better serves the purpose.¹⁴ In very general terms, ADEV can be considered as the likely variation between any two measurements taken at a specified interval (the interval is denoted as “tau”). For example, stating that “the “Allan Deviation of the PPS output is 2.3×10^{-10} at tau = 10 seconds” means that the values of any measurements taken of this PPS source at intervals of 10 seconds will mainly be within a range of 23 nanoseconds. A table of ADEV vs. tau describes the stability of the oscillator over varying time periods.

It is convenient to plot ADEV versus tau on a graph with ADEV on the Y axis, and tau on the X axis, both in log format. One advantage of this representation is that the slope of the plot reveals the primary noise process at work in that range of tau, as shown in Figure 1.

13 While “clock” and “oscillator” have different formal definitions (a clock consists of an oscillator plus additional components), this paper follows common informal practice and uses the two terms synonymously.

14 A quite good tutorial on ADEV is at https://en.wikipedia.org/wiki/Allan_variance

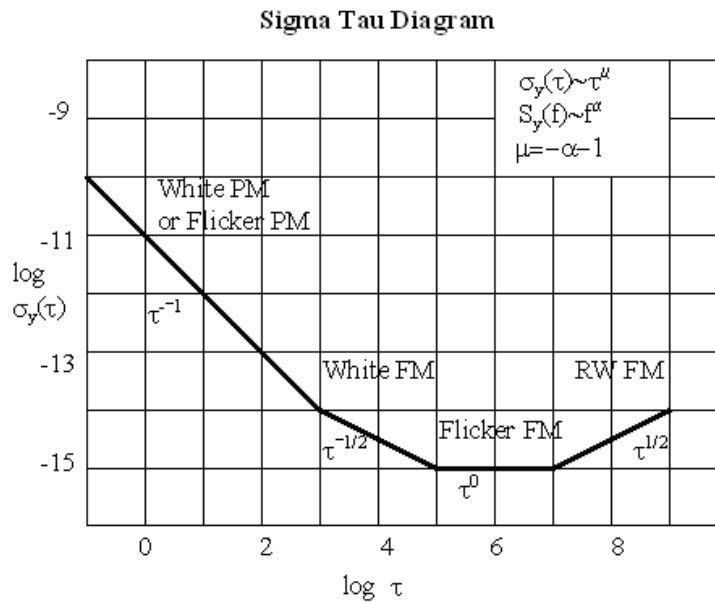


Figure 1: Noise Processes Shown as Allan Deviation Slope¹⁵

The PPS output of most GPS receivers exhibits white phase modulation bounded in absolute amplitude except for outliers, so the ADEV normally improves by one order of magnitude for each order of magnitude increase in tau. This is shown by a slope of minus 1 on the ADEV plot. In other words, longer averaging continues indefinitely to reduce noise and increase frequency stability (and thereby allow a more precise measurement).

In the discussion below, both strip-chart phase records and ADEV plots are used for illustration. Often both the phase record of the full data run, as well as a zoomed-in portion showing second-by-second variations, are provided.

As a technical note, the phase record will show any offset in frequency between the reference clock and the device under test as an upward or downward trend whose slope can be converted to a fractional frequency offset value. Since no two clocks ever run at exactly the same frequency, if for no reason other than quantum uncertainties, such an offset will always appear in the phase record of two independent sources. The offset and slope between the Cesium frequency reference and GPS constellation clock can be seen in Figure 3 below. For the purposes of this paper, the frequency offset is not relevant to the receiver performance, and therefore all figures showing phase plots, other than Figures 3 and 34, have had this offset removed to present a flat phase trend. Note that Allan Deviation measurements are not sensitive to frequency offset, so there is no need to account for it in ADEV plots.

¹⁵ Figure courtesy of W. J. Riley

2. RESULTS: TIMING PERFORMANCE

To set the stage, the following figures show the timing performance of all seven receivers in a single set of plots.



Figure 2: Allan Deviation of all seven u-blox receivers.

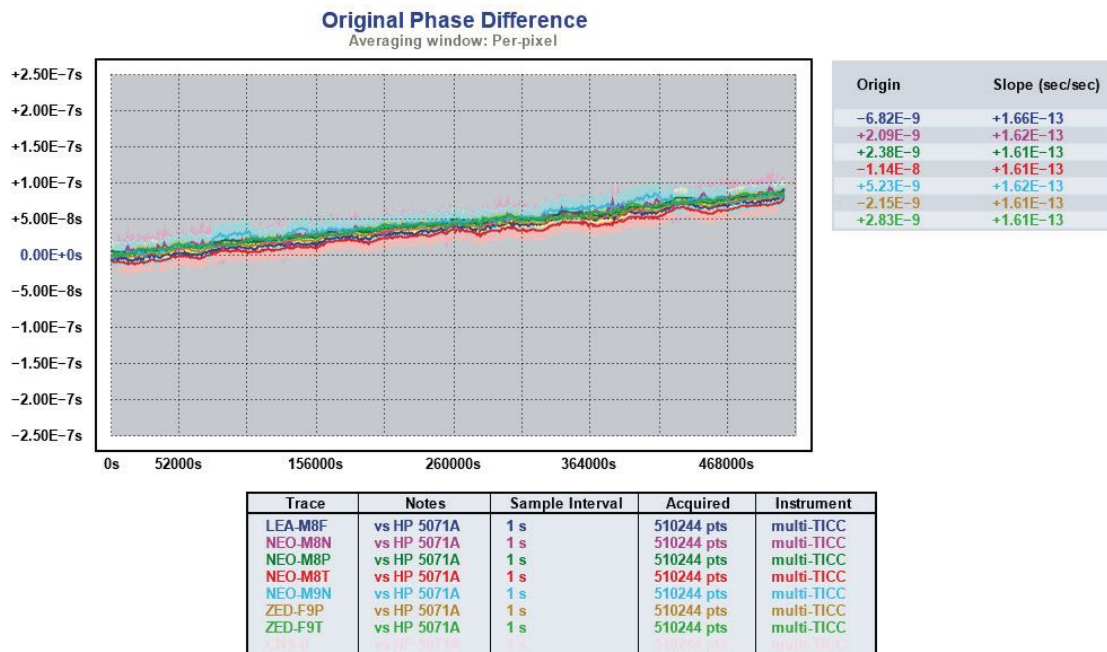


Figure 3: Raw Phase Difference

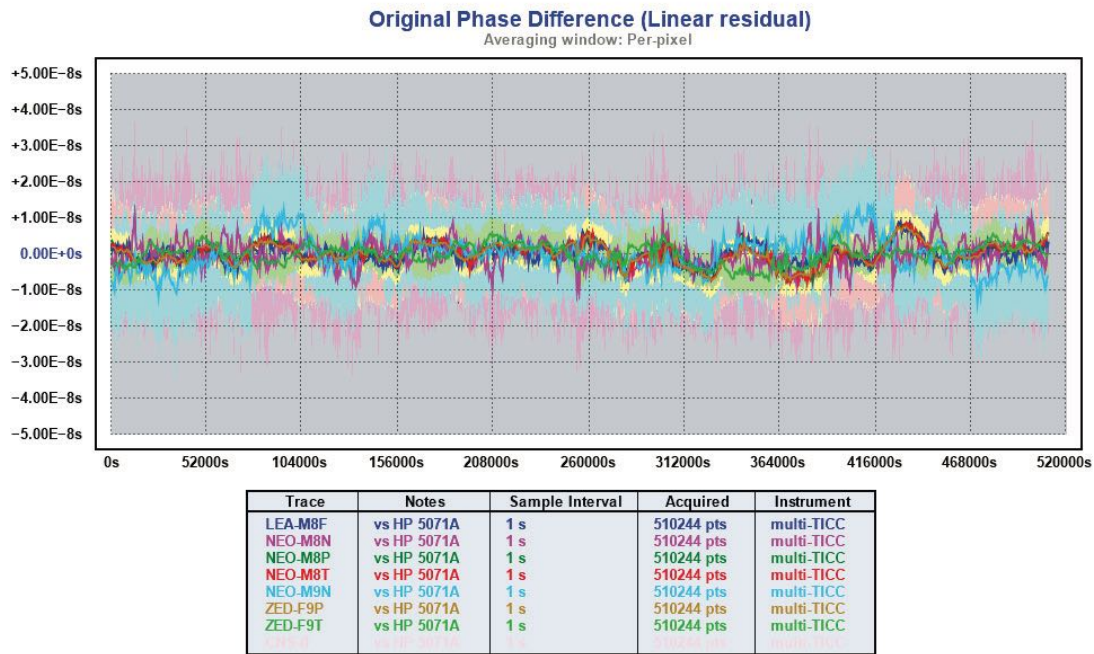


Figure 4: Phase With Offset Removed

Figure 2 shows the Allan Deviation of the seven u-blox receivers on a single plot. The results fall into two main groups: the single-frequency receivers with ADEV around 1×10^{-8} at 1 second tau, and the dual-frequency receivers, as well as the LEA-M8F, with 1 second ADEVs near 4×10^{-9} .

Figure 3 shows the phase of the receivers compared to the Cesium reference. The upward slope indicates that the reference was about 1.6×10^{-13} low in frequency compared to the GPS constellation.¹⁶ Figure 4 shows the same data with that slope removed.

Because it is difficult so see the performance of the individual receivers in these composite plots, the following sections describe and plot results for subgroups of the receivers.

¹⁶ The HP 5071A/HP specification is for frequency accuracy of better than 5×10^{-13} ; this measurement shows that this unit is operating well within that specification.

2.1 “N” Series Receivers

The NEO-M8N and NEO-M9N are low cost modules intended for navigation. They are not optimal for timing purposes because they do not allow a “0D” timing solution configuration (see Section 4 below), and they do not report the quantization error (see Section 3 below) to allow for software correction.

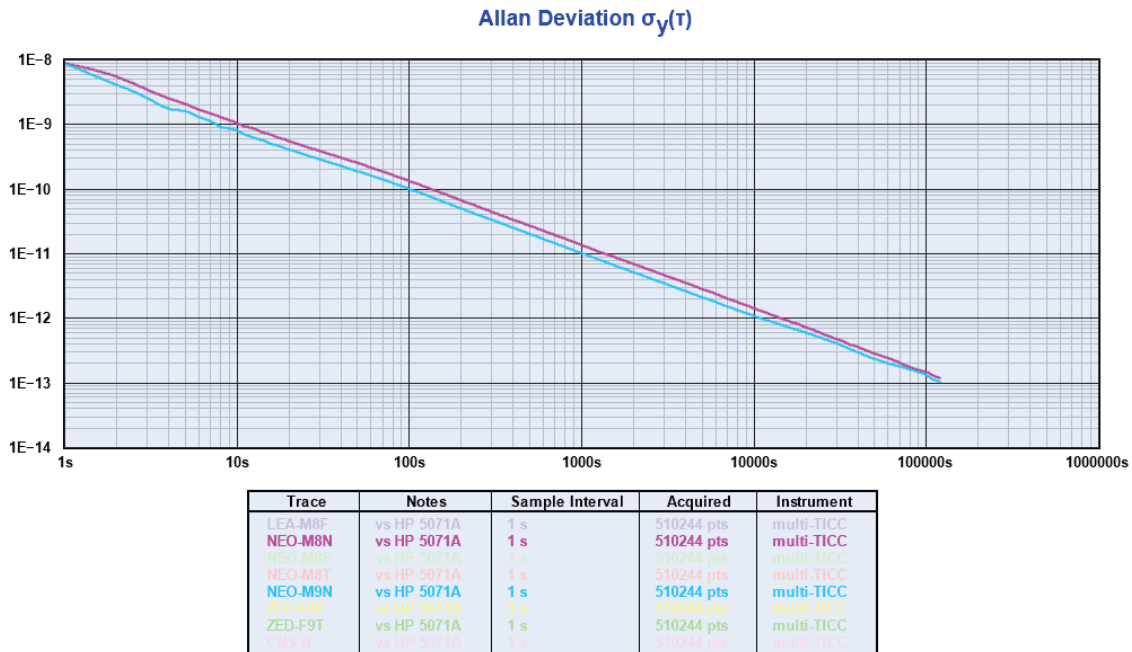


Figure 5: Allan Deviation of M8N and M9N Receivers

Figure 5 shows that the newer M9N receiver offers slightly better ADEV at most tau than the older M8N, but the difference is not substantial.

Figure 6 shows phase plots of the two receivers over the full measurement, and Figure 7 shows an approximately 45 second interval of that data. Figure 7 shows that the M8N has a sawtooth characteristic with a period of just under 10 seconds,¹⁷ with a peak-to-peak range of about 17 nanoseconds. The M9N shows a different pattern, with noticeable second-to-second variation of about 7 nanoseconds and larger steps of about 14 nanoseconds at an approximate 9 second interval. The smaller average noise amplitude explains the slight ADEV advantage of the M9N unit.

¹⁷ There is no reason to believe that this period is consistent across receivers or measurement runs. In particular, the period may be temperature dependent.

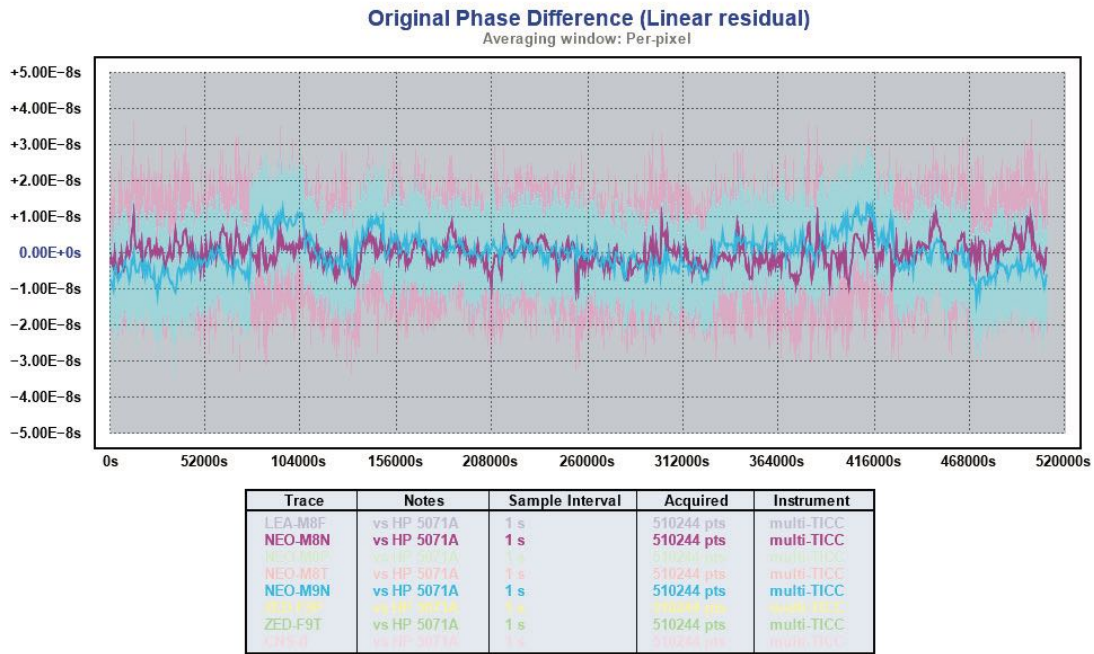


Figure 6: Phase Plot of M8N and M9N Receivers

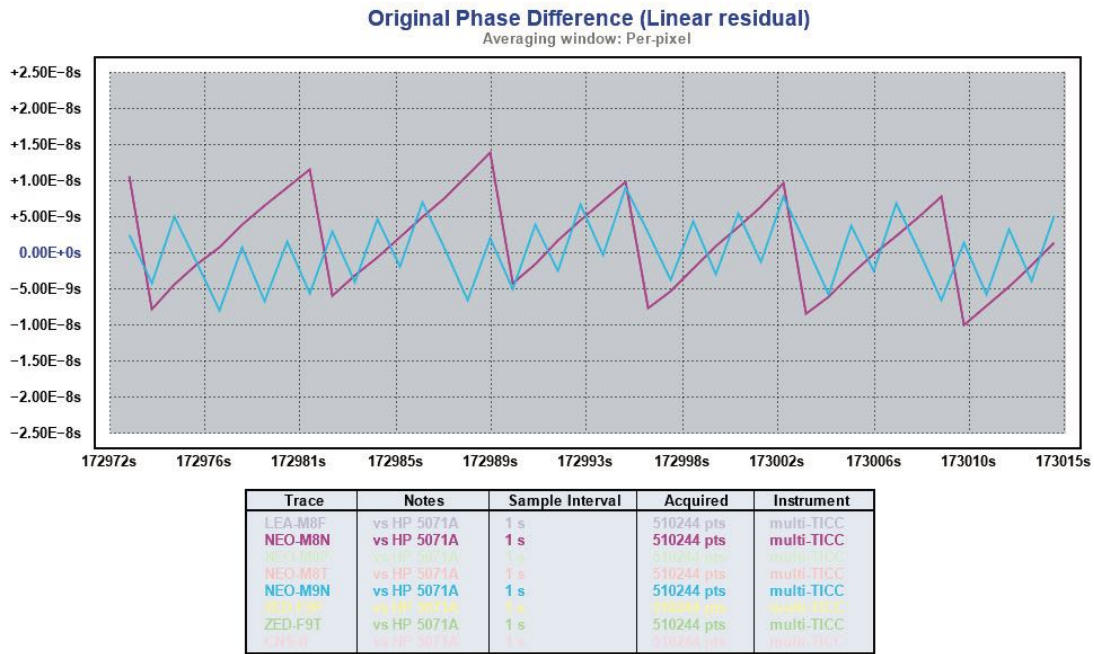


Figure 7: Zoomed Phase of M8N and M9N Receivers

2.2 M8P and M8T Series

The NEO-M8P (Positioning) and NEO-M8T (Timing) are more capable receivers than the “N” series, albeit still single-frequency. They are very similar, and while their raw timekeeping performance is only moderately better than the “N” series receivers, their additional capabilities allow better ultimate results. Both receivers can output the raw observation data (pseudorange, carrier phase, and doppler) required for RTK or PPP processing, and both provide quantization error correction (see Section 3 below), and a “0D” or “timing” solution mode that improves timing performance (see Section 4 below).

The differences between the “P” and “T” versions lie in the additional capabilities each provides. In particular, the M8P has an inbuilt RTK engine for real time kinematic positioning,¹⁸ while the M8T does not include the RTK engine but provides two TIMEPULSE outputs and two EXTINT inputs versus one of each on the M8P. The M8T is somewhat less expensive than the M8P, and that it makes it an attractive choice where internal RTK processing is not required.

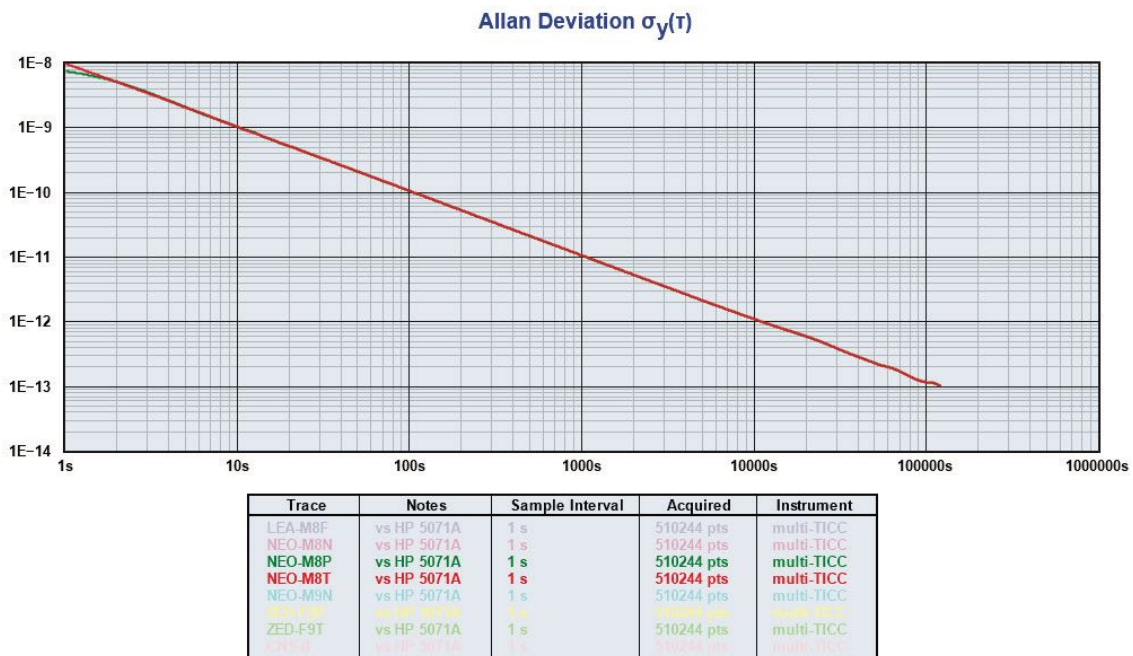


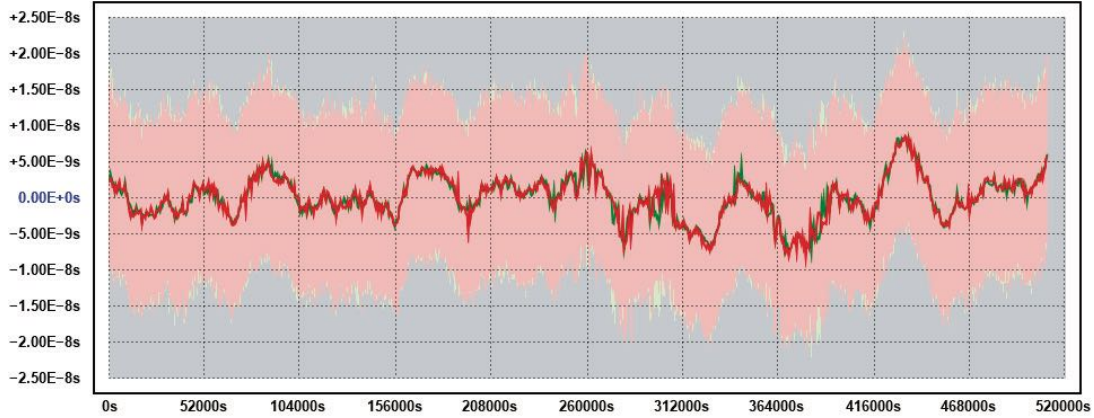
Figure 8: Allan Deviation of M8P and M8T Receivers

Figures 8 and 9 show ADEV and phase outputs for both the M8P (green) and M8T (red). These plots of the two receivers are virtually identical, with a very tiny edge to the M8P right at 1 second tau. The large-scale phase variations track very closely between the two units. Figure 10 zooms in the phase view. The peak-to-peak amplitude of both receivers is very similar, but the M8T sawtooth rate is about double that of the M8P and seems to operate in a “two-step” fashion (easier to see in the figure than to describe). The reason for this is currently unknown.

¹⁸ The RTK capability uses a second communications port to receive corrections in RTCM format, which are then applied to the navigation solutions output on the primary port.

Original Phase Difference (Linear residual)

Averaging window: Per-pixel

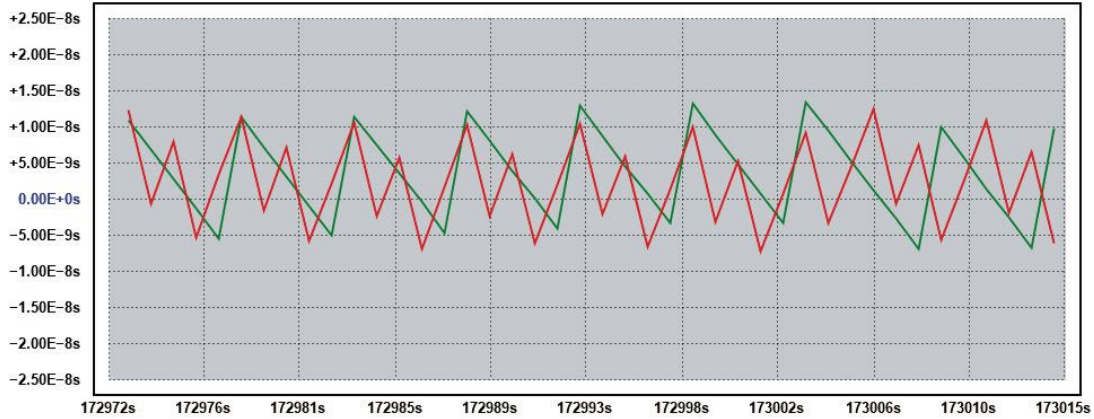


Trace	Notes	Sample Interval	Acquired	Instrument
LEA-M8F	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8N	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8P	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8T	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8N	vs HP 5071A	1 s	510244 pts	multi-TICC
ZED-F9P	vs HP 5071A	1 s	510244 pts	multi-TICC
ZED-F9T	vs HP 5071A	1 s	510244 pts	multi-TICC

Figure 9: Raw Phase of M8P and M8T Receivers

Original Phase Difference (Linear residual)

Averaging window: Per-pixel



Trace	Notes	Sample Interval	Acquired	Instrument
LEA-M8F	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8N	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8P	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8T	vs HP 5071A	1 s	510244 pts	multi-TICC
NEO-M8N	vs HP 5071A	1 s	510244 pts	multi-TICC
ZED-F9P	vs HP 5071A	1 s	510244 pts	multi-TICC
ZED-F9T	vs HP 5071A	1 s	510244 pts	multi-TICC

Figure 10: Zoomed Phase of M8P and M8T Receivers

2.3 F9P and F9T Series

The recently released ZED-F9P and ZED-F9T receivers represent a first in low-cost GPS technology, providing dual frequency operation along with other performance improvements. As in the “8” series, the ZED-F9P is optimized for positioning and has an inbuilt RTK processing engine, while the ZED-F9T is optimized for timing and costs less. Both provide raw observation data information for external processing.

Use of two reception frequencies (L1 = 1575 MHz; L2 = 1242 MHz) allows the receiver to calculate ionospheric/atmospheric delays and compensate for them, improving the accuracy of the fix. Survey/geodetic grade GPS timing receivers all utilize this strategy, and this capability should allow a significant performance increase over single-frequency receivers. It is also possible from two-frequency observations to calculate Total Electron Content values for the atmosphere near the receiver, a quantity of interest to space scientists and propagation predictors.

The dual-frequency capability of the ZED-F9 receivers has one qualification that is important to note: the only L2 modulation supported for the GPS constellation is the L2C signal that has been included in new GPS satellites launched since 2005.¹⁹ For older satellites, the F9 receivers will revert to single-frequency operation. A few of the older satellites are still operational and they act to “dilute” the F9 performance compared to other dual-frequency receivers that can make use of additional L2 signals.

As a result, an F9 series receiver listening only to the GPS constellation may have marginally greater ambiguity in its results compared to a survey-grade receiver. As time goes on and new satellites replace the oldest ones, the importance of this limitation will diminish. In the short term, enabling the GLONASS constellation will increase the number of L2 satellites the F9 can use, and improve its positioning performance. However, for timing purposes it is recommended to use only one constellation,²⁰ so in that case using GPS and disabling GLONASS is still the best option.

19 Early dual-frequency receivers used the L2P(Y) code which is broadcast by all GPS satellites. Though the code is encrypted and theoretically available only to military and other authorized users, means have been found to use the P code without decryption.

20 Because the two constellations are referenced to different master clocks, and mixing them increases timing ambiguity.

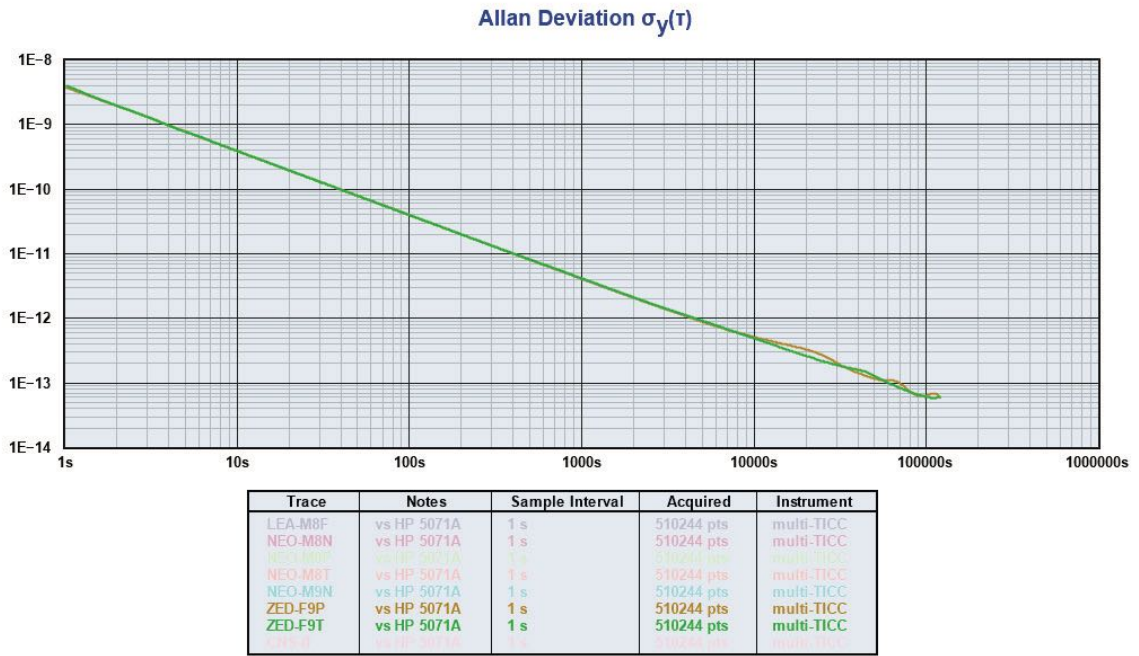
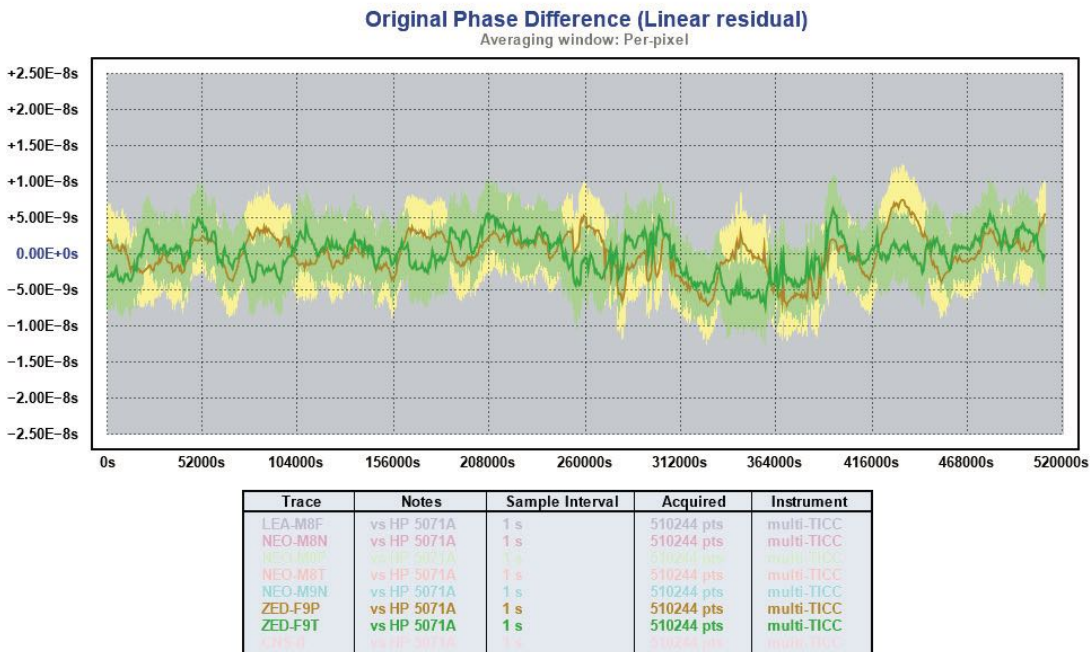


Figure 11: Allan Deviation of ZED-F9P and ZED-F9T Receivers



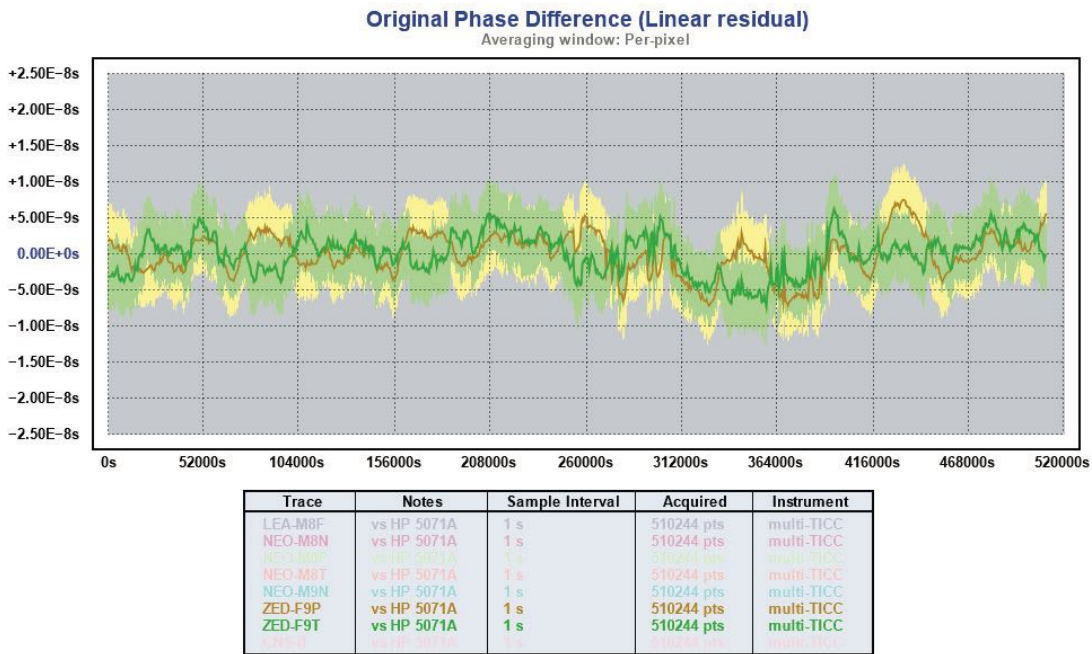


Figure 12: Raw Phase of ZED-F9P and ZED-F9T Receivers

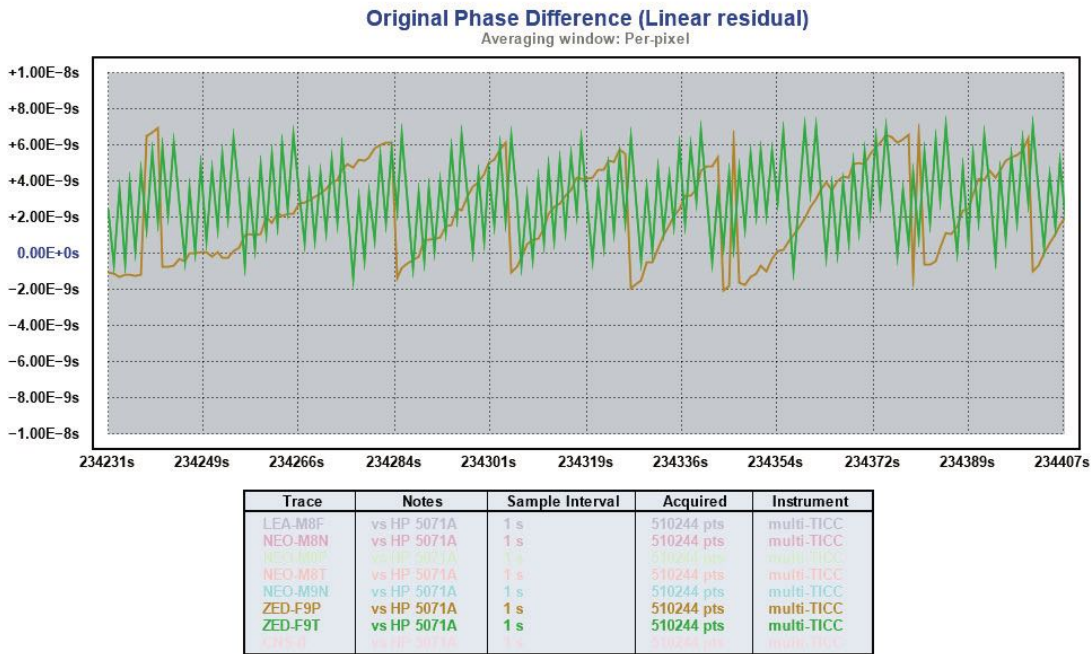


Figure 13: Zoomed Phase of ZED-F9P and ZED-F9T Receivers

Figures 11 and 12 plot ADEV and phase for the F9P and the F9T. As with the M8P and M8T, the two variants show essentially identical performance. It is interesting to note, but probably not meaningful, that in the long term the phase offsets of the two receivers seem to be 180

degrees out of phase – when one exhibits a positive offset, the other tends to show a negative offset.

Figure 13 shows an expanded phase view, and demonstrates markedly different behavior compared to the M8 series. In particular, the two receivers have very similar peak-to-peak noise of about 7 nanoseconds, but their processing appears to handle the TIMEPULSE output steering very differently. The F9T shows a multiple sawtooth pattern with second-by-second jitter of about 3 nanoseconds superimposed on an approximately 8 nanosecond sawtooth with a period of several seconds, while the F9P appears to show only a single sawtooth that occurs less frequently but with larger steps. As in the case of the 8-series receivers, the reason for this apparent difference is unknown.

2.4 Timing/Positioning Receiver Comparison By Series

Given the similar performance within each usage category (timing and positioning), it is useful to view the 8 and 9 series timing and positioning receivers as two groups, eliminating the lower-performing “N” series receivers and the interesting but quite different LEA-M8F from the visible plots.

For convenience, Figures 14 through 16 compare the four higher-end receivers (NEO-M8P, NEO-M8T, ZED-F9P, ZED-F9T) on a common scale. As expected, the dual-frequency receivers show lower jitter than their single-frequency equivalents, translating to lower Allan Deviation at a given measurement interval.



Figure 14: Positioning/Timing Receiver Allan Deviation Comparison

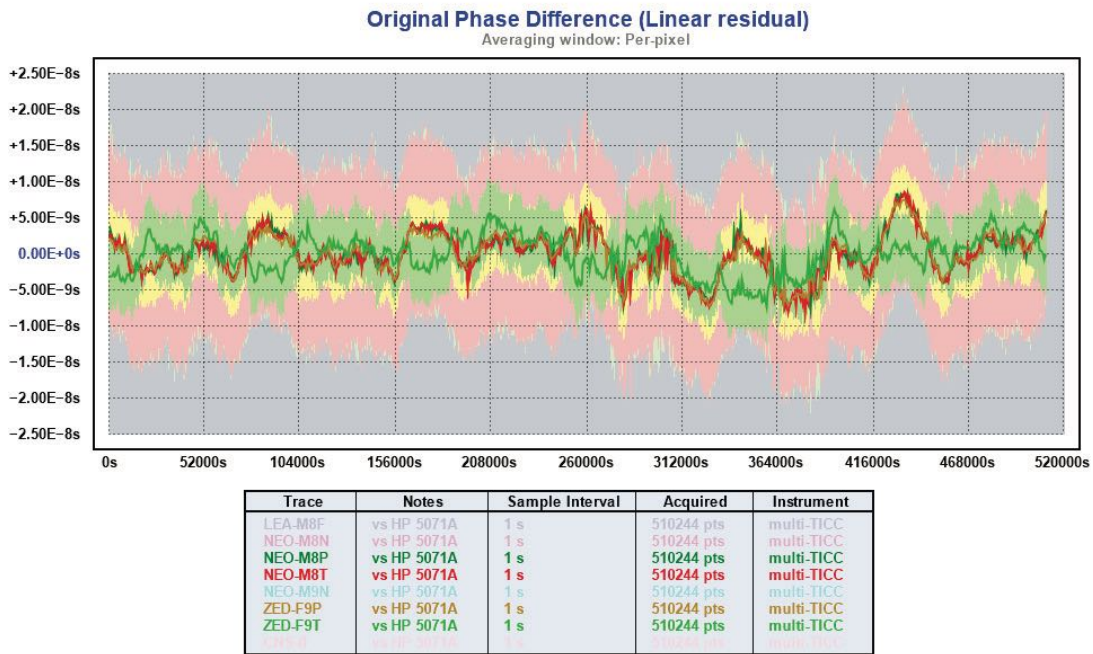


Figure 15: Positioning/Timing Receiver Phase Comparison

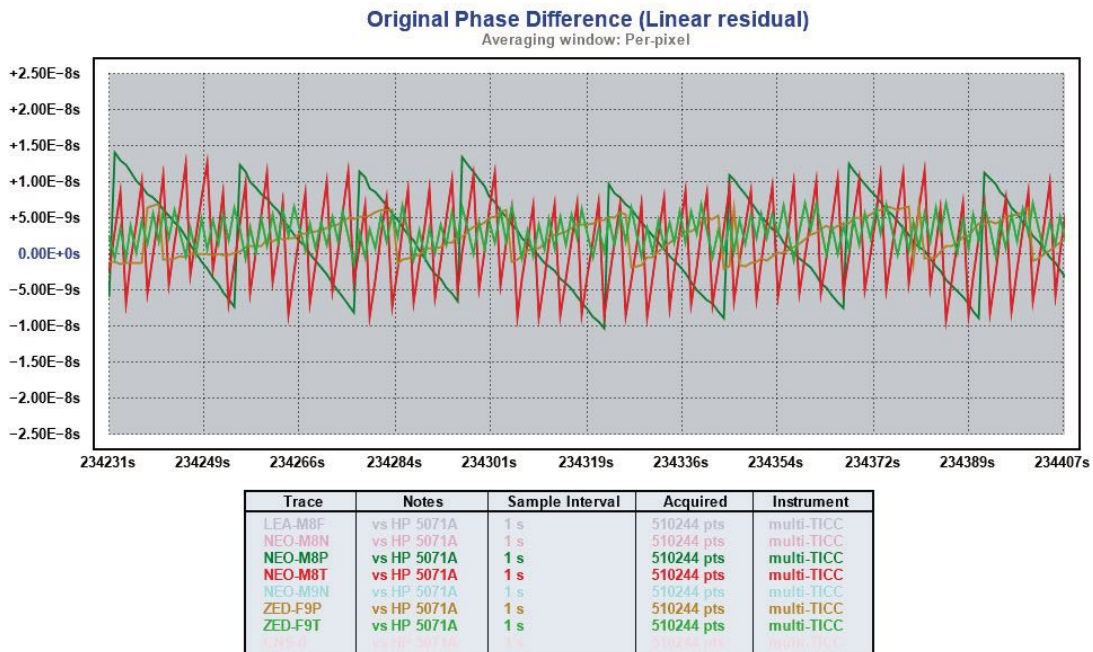


Figure 16: Positioning/Timing Receiver Zoomed Phase Comparison

The zoomed phase shows the two “P” receivers exhibiting the same simple sawtooth behavior over multiple seconds, while the two “T” models show a shorter-term sawtooth that is not visible on the “P” unit plots. It is unknown, but worth exploring in future, whether those differences are coincidental or result in differences in implementation.

2.5 LEA-M8F

The LEA-M8F has a different architecture than the other u-blox receivers. It includes a “frequency and time” subsystem that incorporates a 30.72 MHz TCXO which is kept disciplined to GPS. The TIMEPULSE is derived from that signal, and u-blox claims that it is essentially “jitter free”. (Of course, the truth of this statement depends on the user’s definition of jitter!)

In a sense, the LEA-M8F is a GPS disciplined oscillator (“GPSDO”), with its internal TCXO steered to the GPS timebase. In addition, it can be configured to steer an external oscillator. However, critical control loop parameters, such as time constant, are not made available to the user, so there is little opportunity to optimize its GPSDO performance, and in particular to take advantage of an oscillator with better short-term performance than the internal unit.

For this receiver, the hardware TIMEPULSE exhibits much less noise both in terms of short term jitter and GPS noise than other receivers in the M8 series – in fact, its performance is quite similar to the ZED-F9. However, the LEA-M8F does **not** include a usable quantization error message as discussed in Section 3 below), so the TIMEPULSE quality cannot be improved further by software correction.

The LEA-M8F makes its 30.72 MHz TCXO output available. This report includes measurements of that signal as well as the TIMEPULSE output.

2.5.1 LEA-M8F Timing Performance

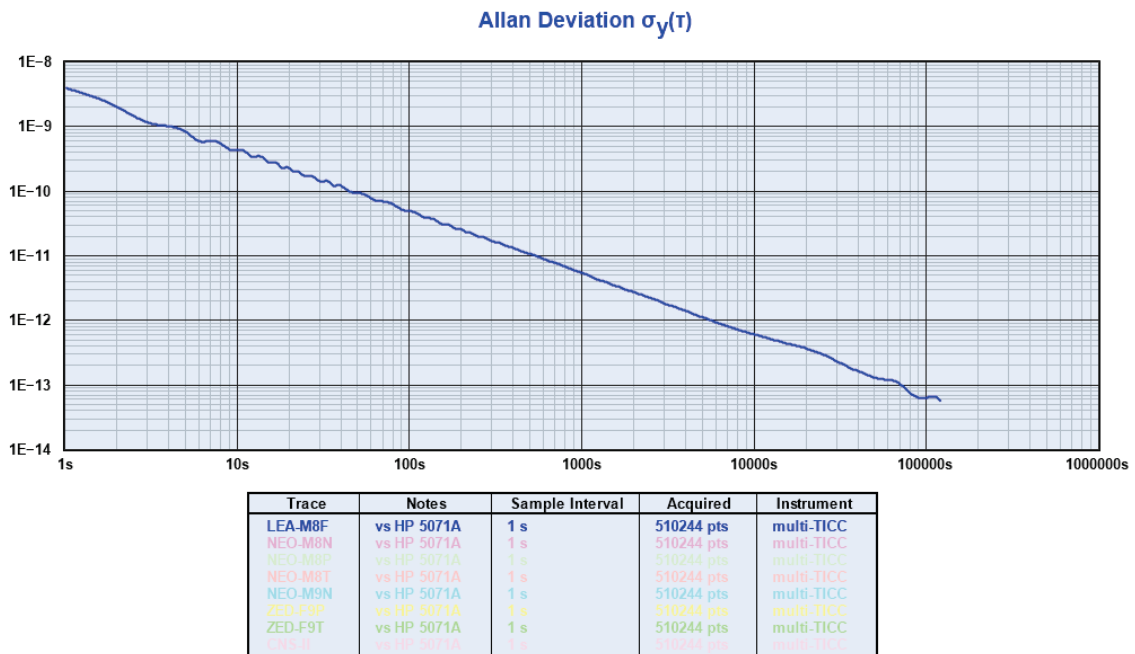


Figure 17: Allan Deviation of LEA-M8F Receiver

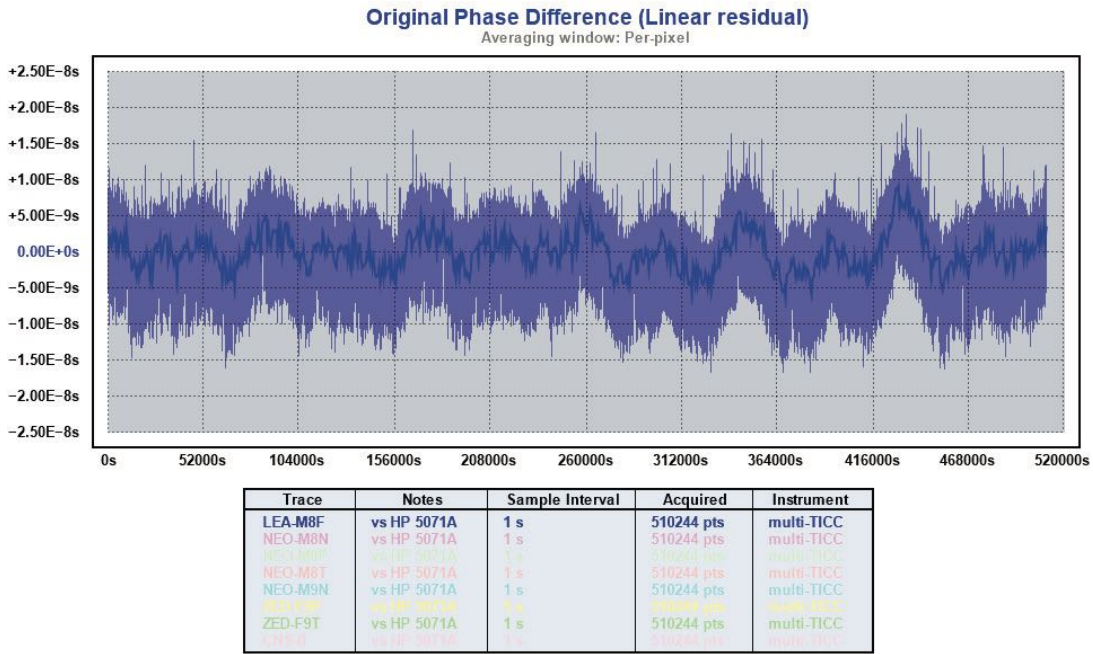


Figure 18: Raw Phase of LEA-M8F Receiver

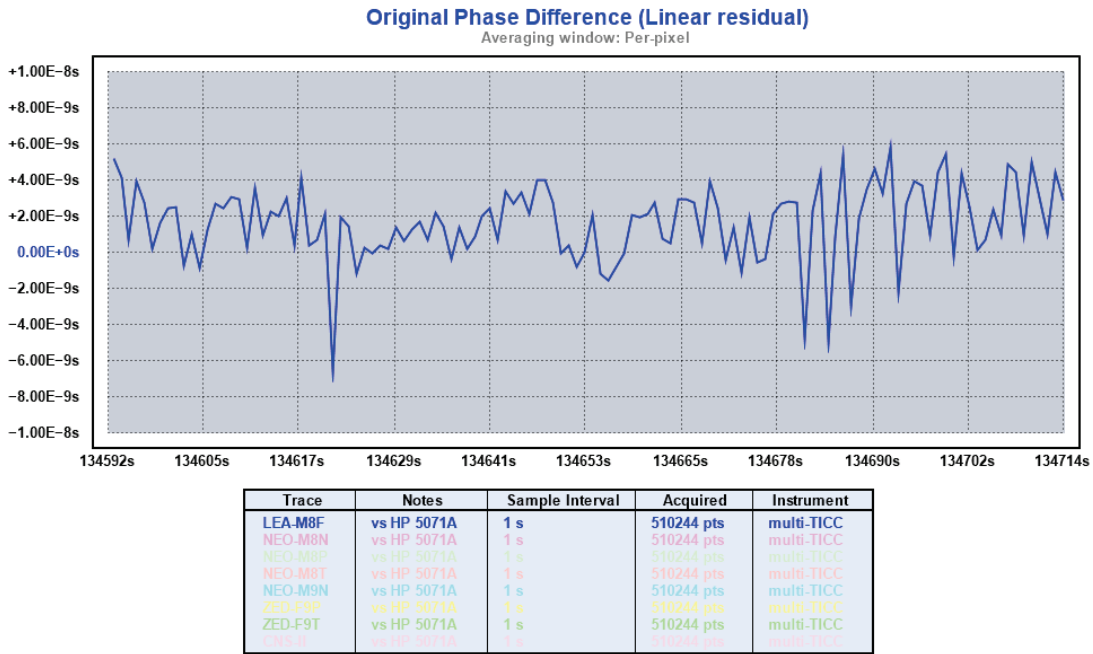


Figure 19: Zoomed Phase of LEA-M8F Receiver

Figure 17 and 18 show ADEV and phase performance for the LEA-M8F receiver, and Figure 19 shows a zoomed segment of phase data. The ADEV performance of the LEA-M8F is quite similar to the raw TIMEPULSE performance of the ZED-F9 series. Notable in the LEA-M8F ADEV plot is a slight ripple that is not present in the other receivers. Also note the occasional

excursions of about 8 nanoseconds apparent in Figure 19. These seem to be the main limiting factor in the unit's ADEV performance.

2.5.2 LEA-M8F Frequency Performance

Characterizing the LEA-M8F's 30.72 MHz steered oscillator output requires different techniques than those used for a PPS signal. A Miles Labs TimePod 5330A phase noise test set²¹, was used to characterize the Allan Deviation and phase noise of the M8F compared to the same 5071A Cesium frequency reference used for the other measurements in this paper. Using the same TimeLab software as before, it is also possible to show frequency as well as phase data, though it is not possible to zoom sufficiently to view cycle-by-cycle performance at this input frequency. The measurement system also allows plotting of RF phase noise data.

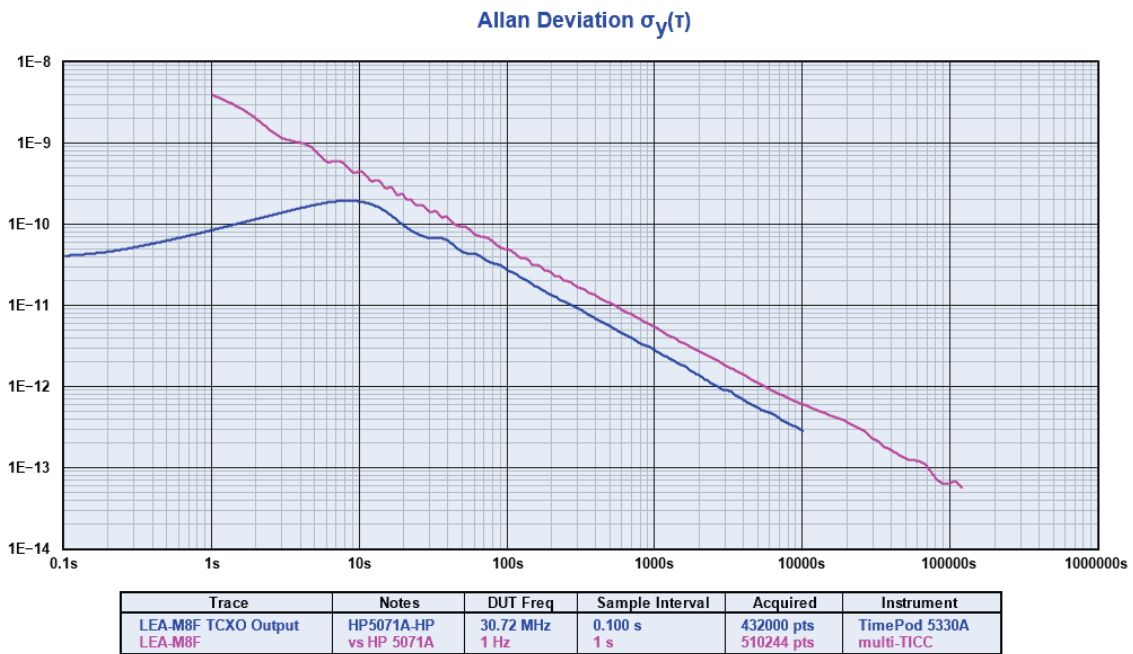


Figure 20: Allan Deviation of LEA-M8F Receiver

21 This unit is equivalent to the Micro-Semi 3120A (<https://www.microsemi.com/product-directory/phase-noise-and-allen-deviation-testers/4131-3120a>).

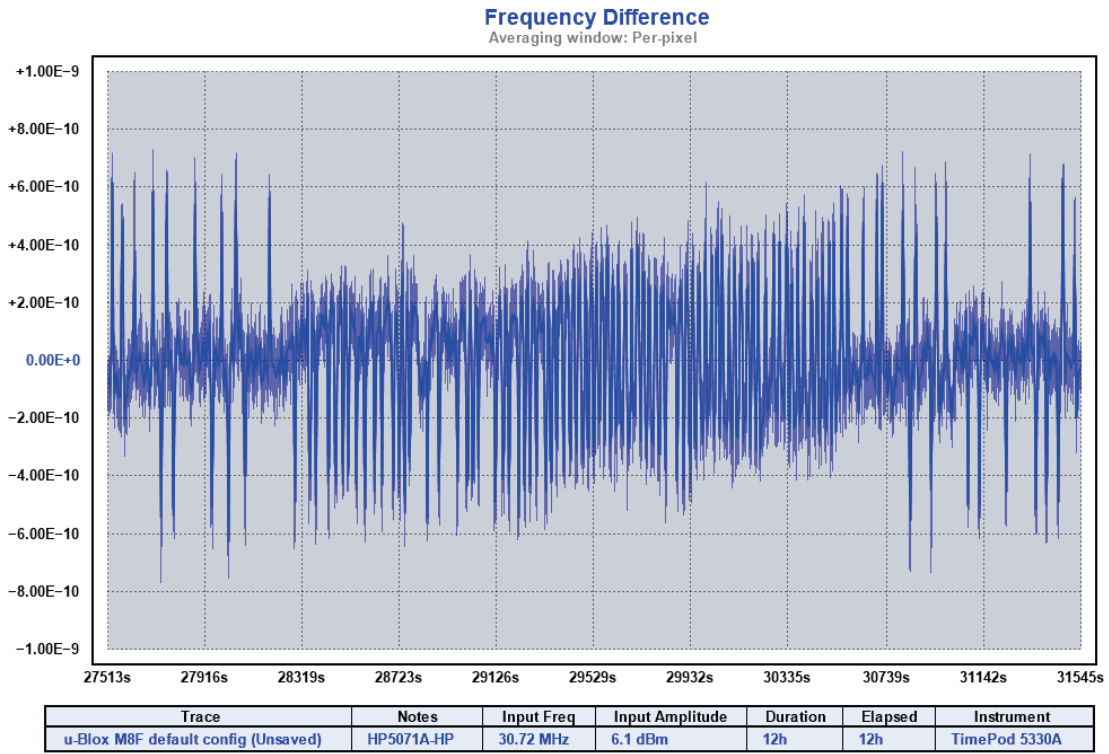


Figure 21: LEA-M8F Relative Frequency vs. Cesium

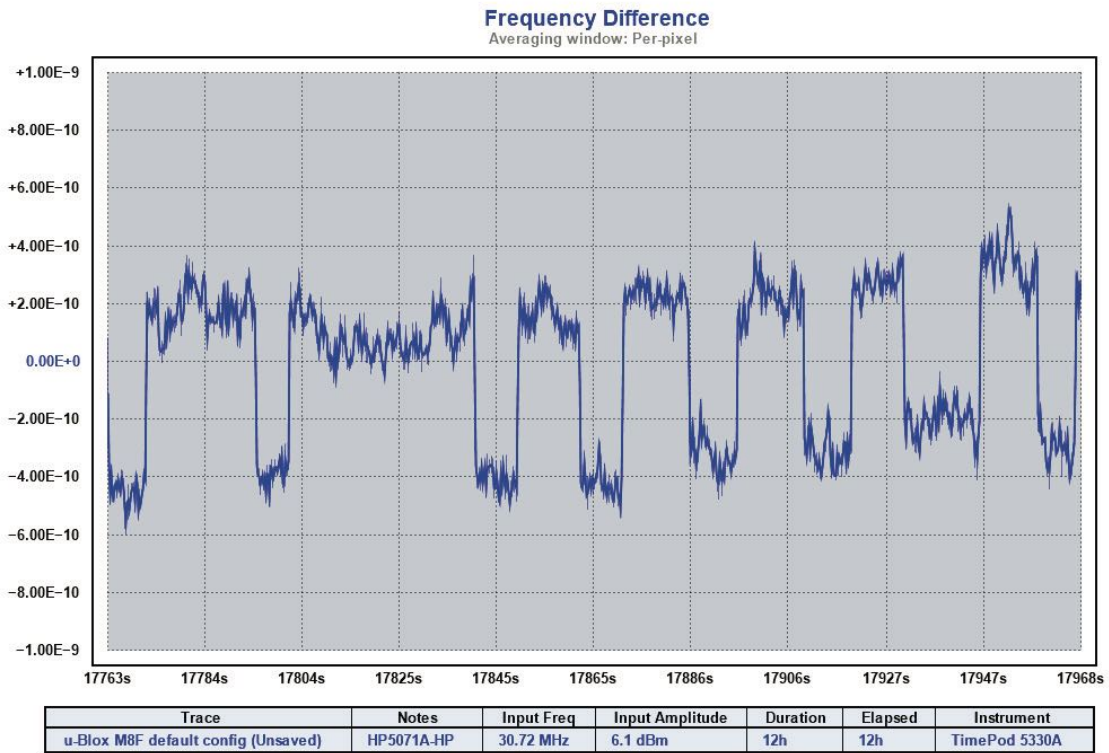


Figure 22: Zoomed LEA-M8F Relative Frequency vs. Cesium

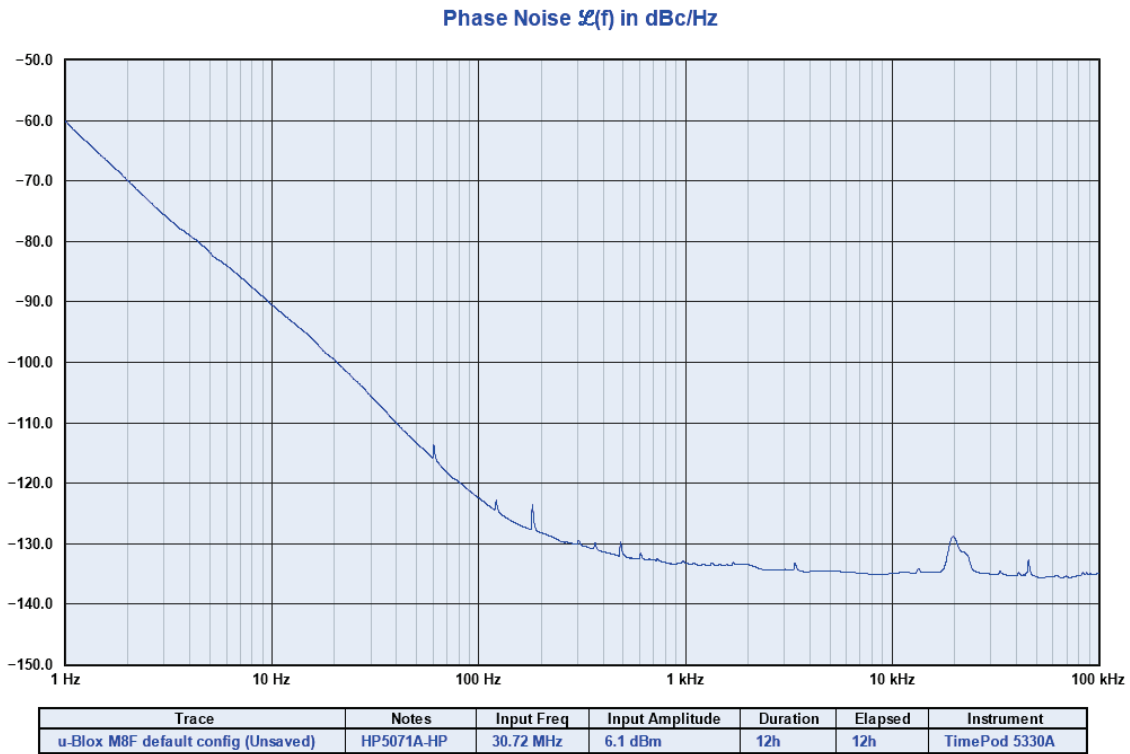


Figure 23: LEA-M8F Phase Noise

The blue line in Figure 20 plots the ADEV of the LEA-M8F steered oscillator output, while the violet line shows the 1 PPS TIMEPULSE output. The 30.72 MHz ADEV is significantly better than the TIMEPULSE output at short tau (<10 seconds) and slightly better at longer tau.

Figure 21 plots relative frequency difference of the LEA-M8F compared to the reference frequency standard over several thousand seconds. Note that the frequency record shows significant variability with numerous spikes of around 6×10^{-10} at varying intervals as well as a series of smaller excursions.

Figure 22 shows an expanded view of the frequency difference data from the LEA-M8F. This reveals that the output dithers approximately $\pm 3 \times 10^{-10}$ around the nominal frequency over periods of about 10 to 40 seconds. It thus appears that a form of pulse width modulation is used to obtain a nominal frequency that is “correct” on average. In fact, however, at any instant the frequency is either plus or minus about 0.3 parts per billion (“PPB”) from nominal. Whether this level of frequency jitter is acceptable in an RF application will depend upon the application. The frequency of this modulation indicates that the steering control loop time constant is quite short; its exact parameters are not known.

Figure 23 shows the phase noise of the 30.72 MHz oscillator from the LEA-M8F. The signal is relatively spur-free, but may be of marginal quality for high-performance RF applications, where a phase noise floor below -130 dBc/Hz is usually desirable.

Conclusion: The LEA-M8F is a unique design that might be useful in some situations where a direct RF signal output of reasonable spectral quality is desired. However, it does not yield results as good as those obtained by traditional “GPSDO” designs, and lack of ability to optimize control parameters limits its usefulness with external oscillators. The dithering used to obtain fine frequency control may be problematic in some applications.

3. SAWTOOTH OR QUANTIZATION ERROR CORRECTION

When a GPS receiver calculates its navigation solution, it can determine with great precision when the top of the second will occur. That knowledge is used to trigger the hardware PPS signal. But the hardware signal must come from somewhere, and it is usually derived from, and is synchronous with, the crystal oscillator that runs the receiver. If the clock runs at, for example, 25 MHz that means there is a timing pulse available every 40 nanoseconds and the receiver can provide an output pulse that is within 20 nanoseconds either side (*i.e.*, early or late) of the calculated time marker.²² If the clock frequency is increased, the PPS granularity is reduced and the time pulse can have even smaller ambiguity. A higher clock rate is the probable explanation for the ZED-F9 series receivers' smaller jitter and better short-term ADEV than the M8 series.

But no clock is ever exactly on frequency, and the receiver has to compensate for the difference between the hardware clock and the GPS constellation clock. It does so by adding or dropping hardware clock ticks between subsequent PPS output pulses to keep the hardware signal aligned as closely as possible to the software-determined second marker. In other words, if the clock runs at 25 MHz instead of counting 25 million ticks between PPS outputs, the receiver might count 25,000,001 or 24,999,999. The result of these changes causes an effect that looks in short-term plots of GPS phase like a sawtooth pattern. u-blox calls this effect “quantization error” and it is visible in the sawtooth on the left and right sides of Figure 24.

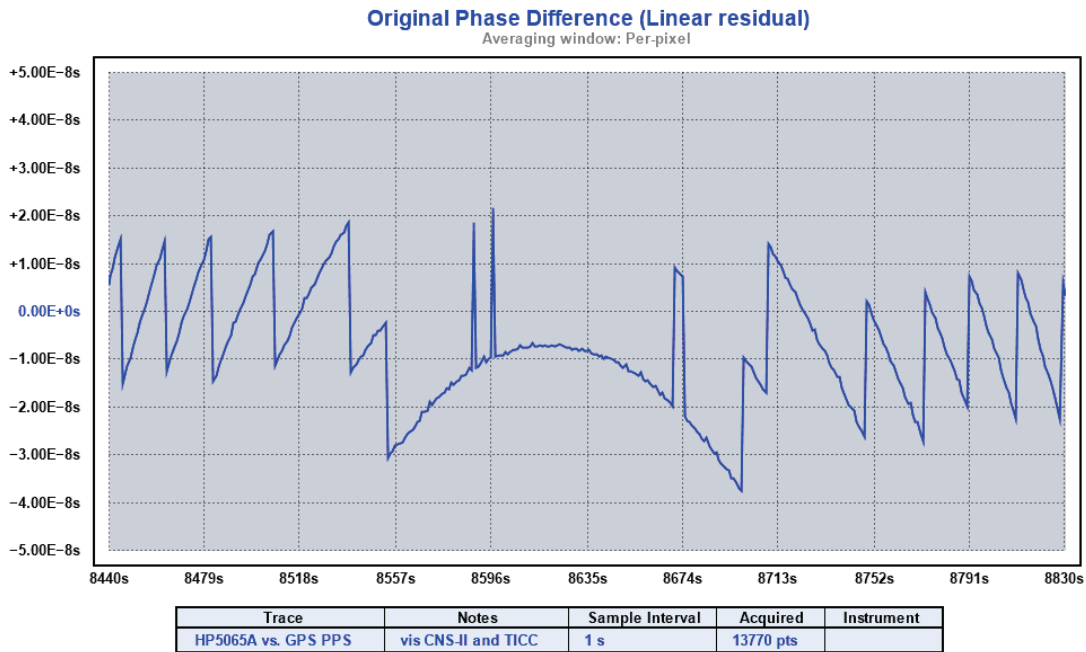


Figure 24: GPS PPS “Hanging Bridge”

²² That is theoretical accuracy; other factors may add fixed or variable offsets to the actual output pulse timing.

This jitter adds noise and worsens the ADEV of the PPS output, especially at short tau. If the frequency offset remains constant the jitter pattern will remain a sawtooth, and increased averaging can reduce or eliminate its effects so it does not impair long-term ADEV.

Over time, though, the oscillator will drift and have other instabilities such as those caused by temperature changes or frequency drift, and those changes can cause the phase relationship between the hardware signal and the computed time mark to change. A slow drift can result in the sawtooth turning into interesting patterns such as the “hanging bridge” shown in the middle of Figure 24. Patterns such as these result in short-term offsets above or below the mean PPS frequency, which reduce the overall quality of the timing result and worsen ADEV.

These effects can be mitigated in several ways. Higher quality internal oscillators (for example, temperature-controlled crystal oscillators, or “TCXOs”) can slow down frequency changes, making the sawtooth cycle longer (which may or may not be a good thing, if the result is to spend more time in a “hanging bridge” condition), and using higher internal frequencies improves resolution. For example, using a 100 MHz clock reduces the granularity to 10 nanoseconds.

While the sawtooth effect is very difficult to eliminate in hardware, software can be used to mitigate it. Many modern GPS units make available a message in the output data stream which predicts the offset of the upcoming hardware pulse from the GPS top-of-second. In the u-blox receivers, this is called a “quantization error” or “qErr” message and it is very effective to reduce the effect of hardware errors in cases where it is feasible to apply a software correction to the hardware PPS results.



Figure 25: Quantization Error and Correction

Figure 25 zooms in on the phase plot for an M8T with two traces added, one showing the value of the quantization error (“qErr”) value received via the receiver serial port (in pink), and the second showing the result of subtracting the qErr value from the phase value (in green). The qErr data almost exactly offsets the raw PPS jitter. Subtracting the two yields the green line, in which almost all jitter has been removed, showing the effectiveness of this technique (and making visible via the upward slope a small offset between GPS and local reference at that point in time).

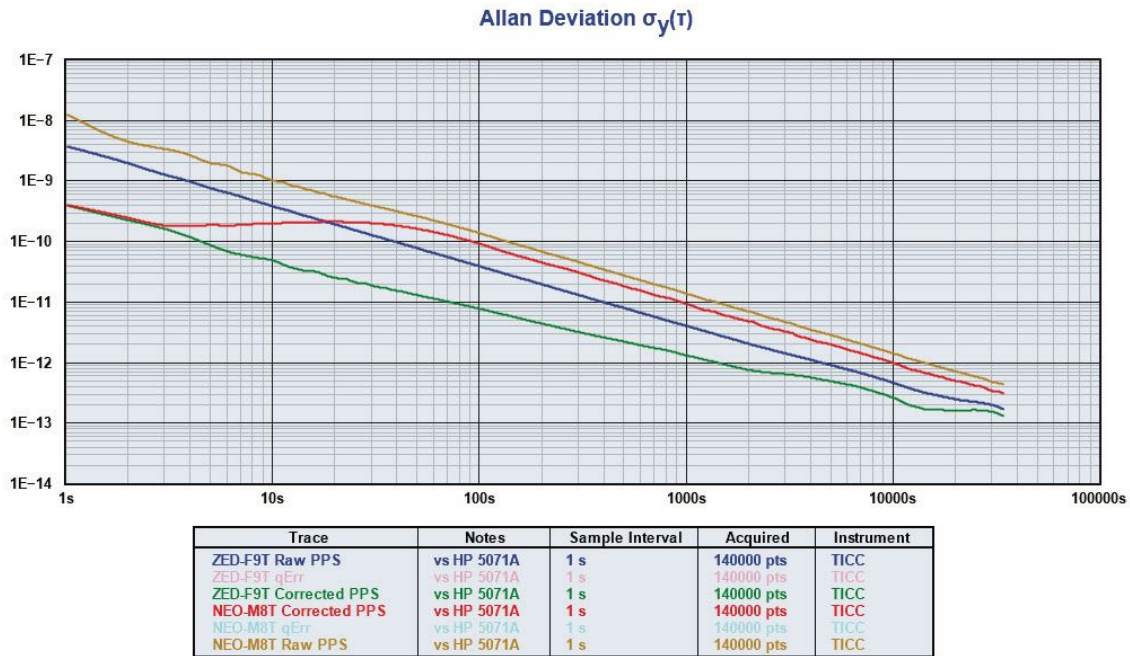


Figure 26: Allan Deviation After qErr Correction

Figure 26 shows before-and-after ADEV performance of both the NEO-M8T and ZED-F9T when applying sawtooth correction. Note that the M8T shows the most dramatic improvement at short tau, and beyond 100 seconds a much more modest one. Conversely, the F9T shows significant improvement out to much longer tau. Overall, the dual-frequency receivers perform about an order of magnitude better than the single-frequency ones.

Looking more closely at the ADEV results, at 3 seconds and below, the two receivers yield almost identical corrected results. It is believed that within this regime the more predictable resolution-limited jitter dominates, and both receivers can correct for this very precisely. At longer tau, however, GPS noise as well as hardware limitations enter the picture, and because the F9T uses its dual-frequency capability to reduce GPS noise, the quantization error remains visible at longer taus, and the effects of the error correction also remain apparent.

Conclusion: Quantization error correction is a powerful tool to improve the short-term jitter performance of GPS receivers. It is at least as beneficial for dual-frequency designs as for single-frequency ones.

4. FIXED POSITION (“0-D”) VS. 3-D NAVIGATION TIMING RESULTS

One difference between standard GPS receivers and units specialized for timing purposes is the ability to enter into a “timing” or “0D” navigation mode, where the receiver’s location is fixed and only the time is calculated in the solution. The receiver location can be determined either autonomously via a site-survey function that averages position readings over a period of time, or by manual entry of the known location into the receiver parameters.

The 0D navigation mode should produce better timing results because there is only one variable being solved for, rather than four (three physical dimensions plus time) as in a normal position fix. How much difference in performance does it actually make?

As shown in Figures 27 through 29, using the 0D mode does improve the Allan Deviation of the TIMEPULSE output for tau greater than about 30 seconds, though there is little difference in the short term. The dual frequency ZED-F9T receiver shows much greater benefit than the single-frequency NEO-M8T. The phase plots show that 0D mode subjectively wanders less over longer time intervals.



Figure 27: Allan Deviation of Timing vs. 3D Navigation Solution

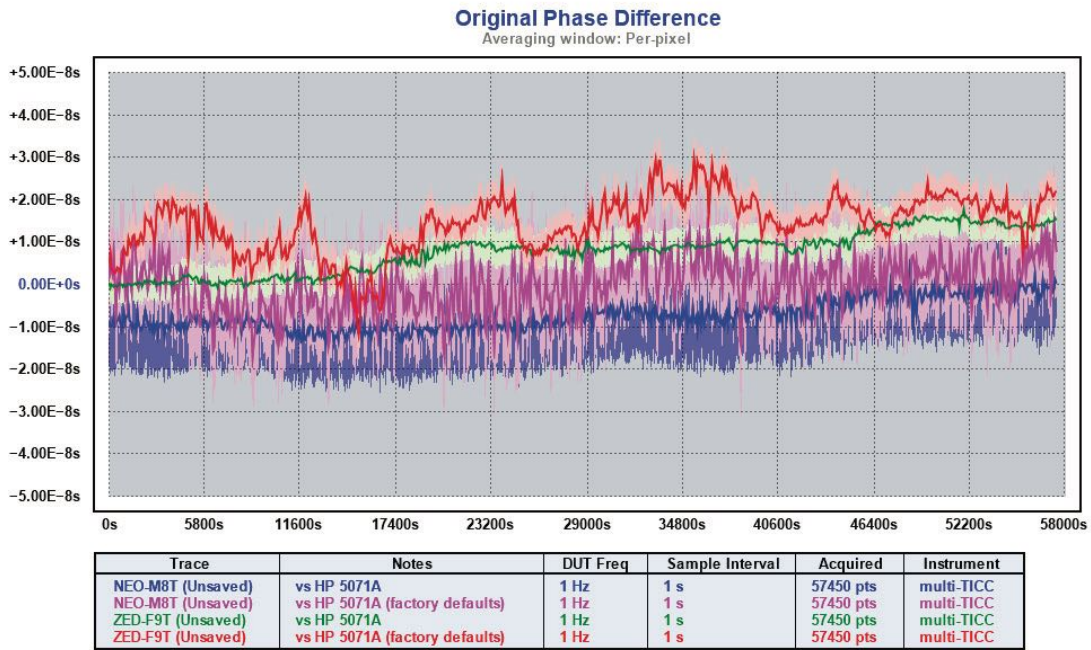


Figure 28: Raw Phase of Timing vs. 3D Navigation Solution

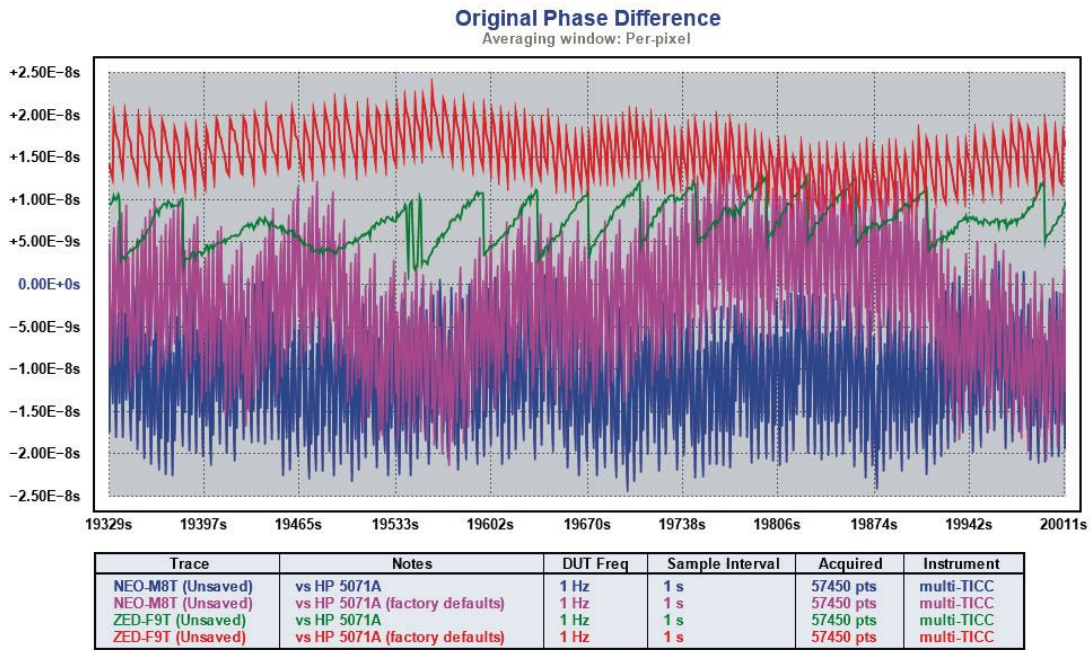


Figure 29: Zoomed Phase of Timing vs. 3D Navigation Solution

Conclusion: Use of “timing” or “0D” navigation solution mode provides improved timing performance, particularly over longer measurement intervals.

5. RESULTS AT HIGHER TIMEPULSE RATES

The u-blox receivers compared here all allow the TIMEPULSE output to be set to much higher rates than one pulse per second, typically 10 MHz or greater. Some receivers also allow a higher navigation solution rate than one per second (8 Hz for the NEO-M9N and 25 Hz for the dual-frequency ZED-F9T). Accordingly, the inexpensive NEO-M8N, its replacement, the NEO-M9N, and high-end ZED-F9T were tested to determine how they perform at a 10 MHz TIMEPULSE rate, and also at higher navigation solution rates where supported. The test configuration was the same as that used for the LEA-M8F measurements reported in Section 2.7.

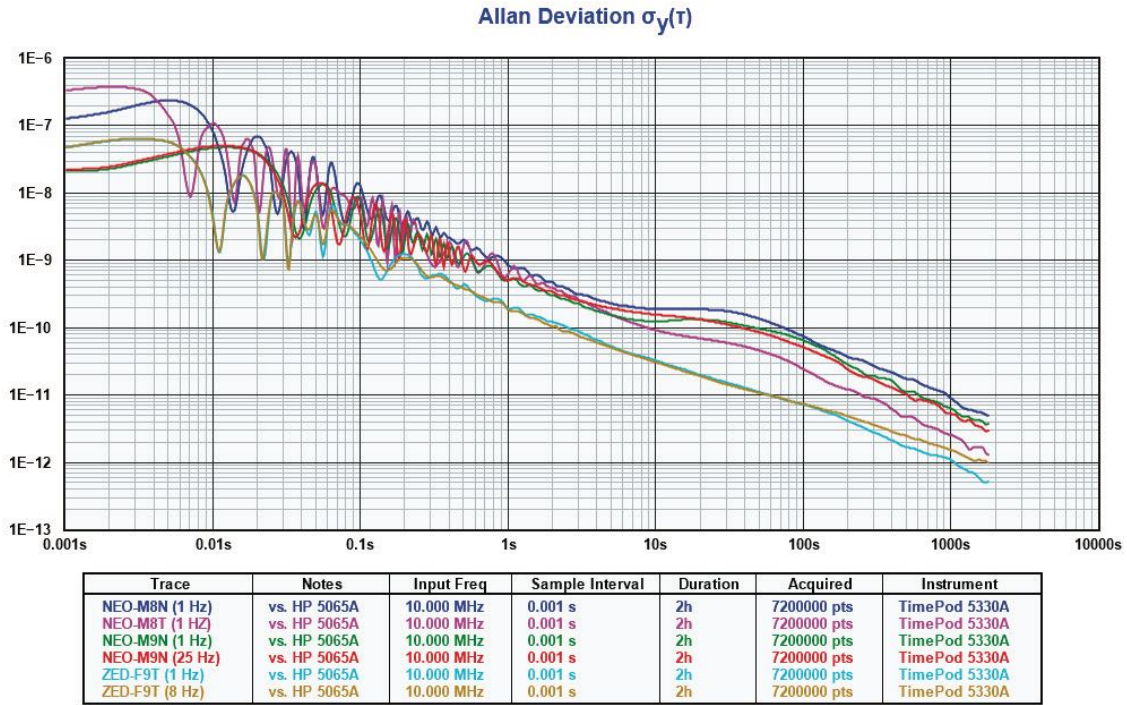


Figure 30: Allan Deviation at 10 MHz TIMEPULSE Rate

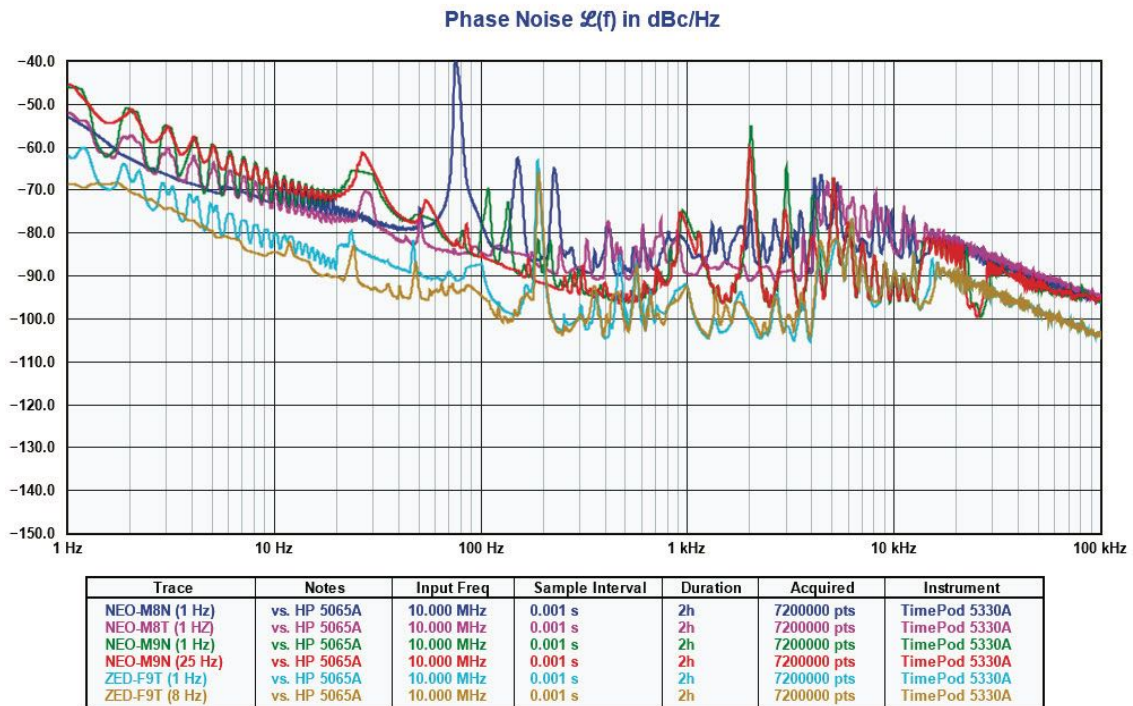


Figure 31: Phase Noise at 10 MHz TIMEPULSE Rate

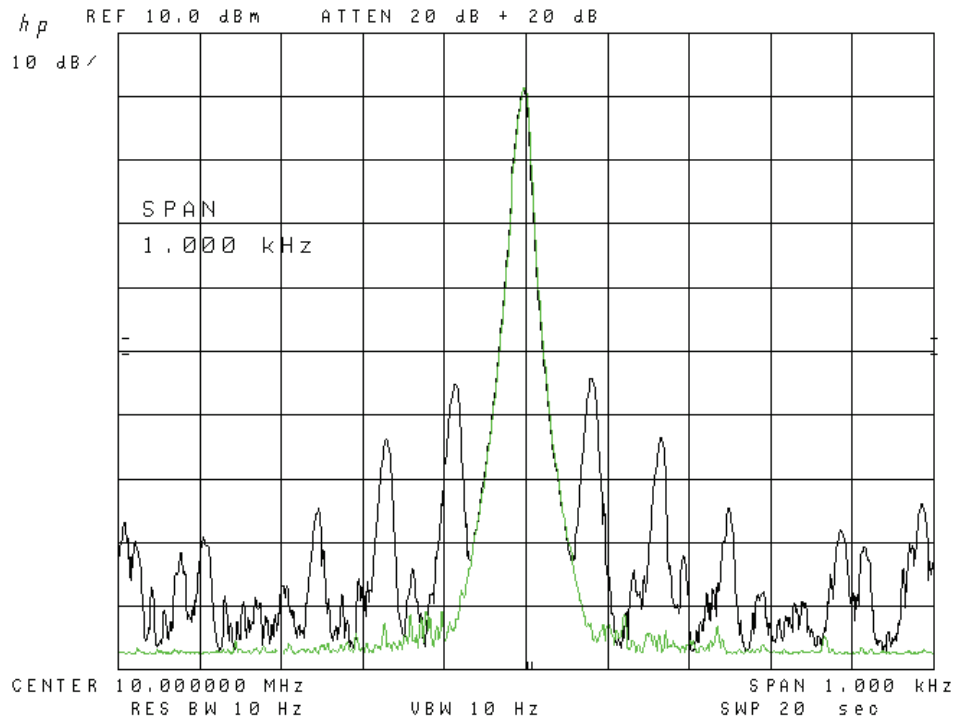


Figure 32: Spectrum Analyzer View of ZED-F9T (Black) and HP 8642A (Green) at 10 MHz, 1 kHz Span

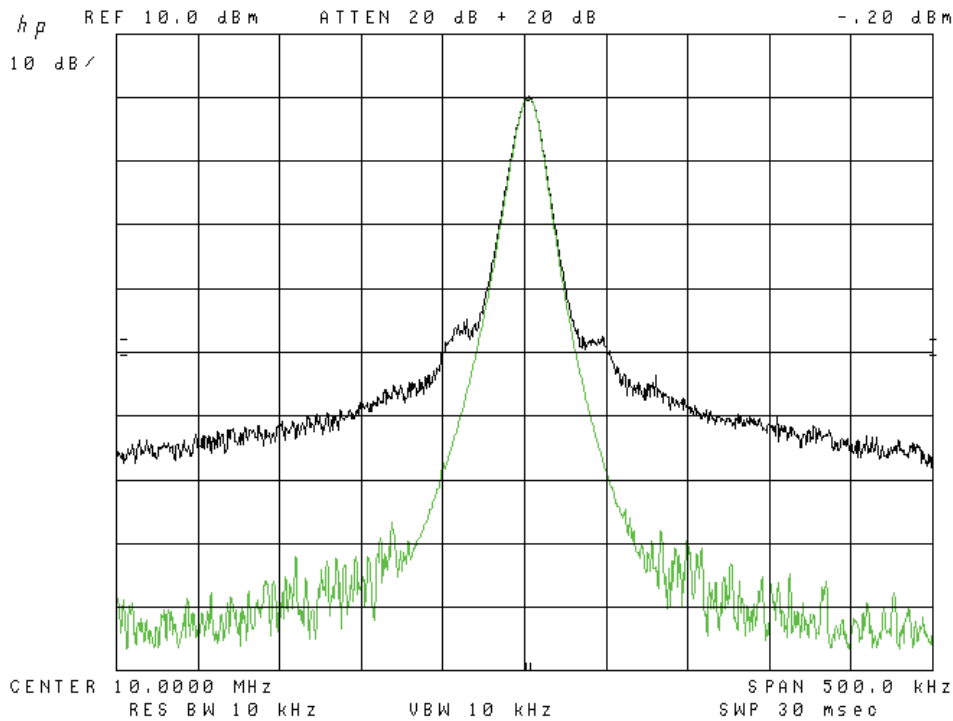


Figure 33: Spectrum Analyzer View of ZED-F9T (Black) and HP 8642A (Green) at 10 MHz, 500 kHz Span

Figure 30 shows the ADEV of the receivers when set to 10 MHz output and 1 Hz navigation solution rates, as well as results for the higher rates supported in the 9-series units.²³ In general, the ADEV of a 10 MHz TIMEPULSE is somewhat better than at one pulse per second. However, the “ringing” oscillations at tau less than 1 second are warnings that not all is well, particularly since a signal at 10 MHz is likely to be used in an RF system where signal purity and spur-free performance is important.

A phase-noise view of the data, as shown in Figure 31, reveals that there is indeed cause for concern. In particular, not only is the performance at close offsets substandard compared to most RF signal sources, but the noise floor is very high at about -105 dBc/Hz at 100 kHz offset. These signals are clearly *not* suitable for use in most RF applications. Figures 32 and 33 show screen captures from an HP 8568B spectrum analyzer comparing the ZED-F9T 10 MHz TIMEPULSE output (white) with a laboratory signal generator (violet). Figure 31 is a close-up showing a 1 kHz span, while Figure 32 shows 500 kHz.

Conclusion: While it would be unreasonable to expect that any GPS receiver signal evaluated this way to be as clean as a laboratory-grade generator, the TIMEPULSE noise is 30 to 40 dB higher and is probably unusable for RF applications.

²³ Note that the X axis begins at 0.001 second in this plot, where the previous plots have all begun at 1 second. This is because the most interesting results appear in this short-tau regime, where frequency stability and phase noise measurements start to cross over.

6. THE “EXTERNAL INTERRUPT” INPUT

Several of the u-blox receivers have one or more “EXT INT” (external interrupt) inputs that can be used to timestamp an external signal against GPS time. This input was tested using a 1 PPS signal derived from the 5071A Cesium frequency standard, logging the output of the u-blox “UBX-TIM-TM2” binary message to disk for later processing.

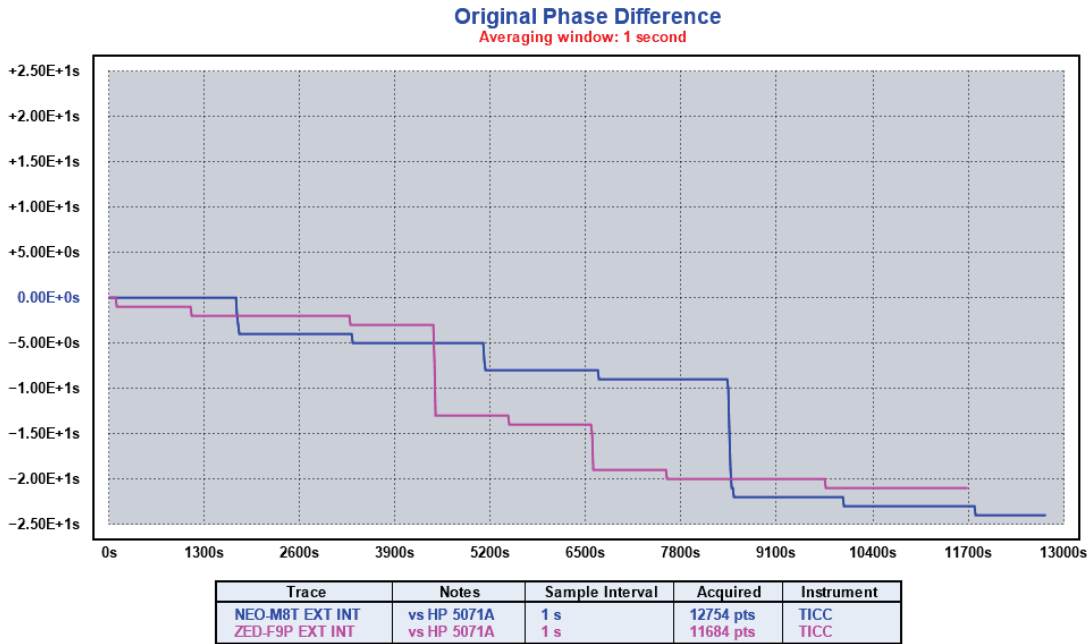


Figure 34: Raw Phase of EXT INT Inputs

Figure 34 shows the phase records of the NEO-M8T (blue) and ZED-F9P (magenta)²⁴ with the EXTINT feature activated and driven by the PPS signal from an HP 5071A Cesium standard. There is clearly a problem here; the stairsteps in the phase record show jumps of several seconds.

It appears that the receivers in this mode sometimes both duplicate and miss input pulses. Here is a portion of the data logged from the UBX-TIM-TM2 message:

```

331370.000000418 10 456
331371.000000420 10 457
331371.000000420 10 457          # duplicate
                                   # 33172 missing

331373.000000418 10 459
331373.000000418 10 459          # duplicate
331374.000000420 10 460

```

²⁴ It would have been more consistent to use the ZED-F9T timing receiver for this test, but the evaluation board for that unit does not make the EXT INT input accessible.

These repeated and missing data points appear with considerable frequency in both the NEO-M8T and ZED-F9P results. If there were only duplicates, it would be easy to filter them out, but the missing data points are very difficult to handle and ultimately make it impossible to create a complete phase record.

Figures 35 through 37 show results after selecting a data segment in which none of these glitches occurred. Using the glitch-free data, the results look broadly similar to those of the TIMEPULSE output, which is what would be expected.

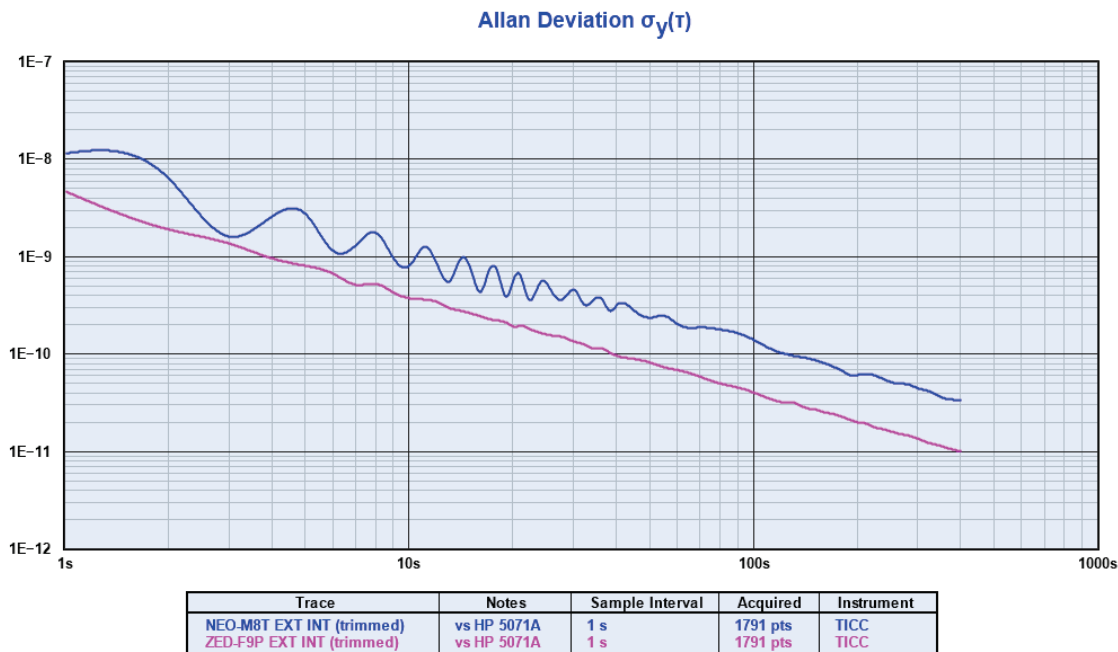


Figure 35: Allan Deviation of EXT INT In Glitch-Free Segment

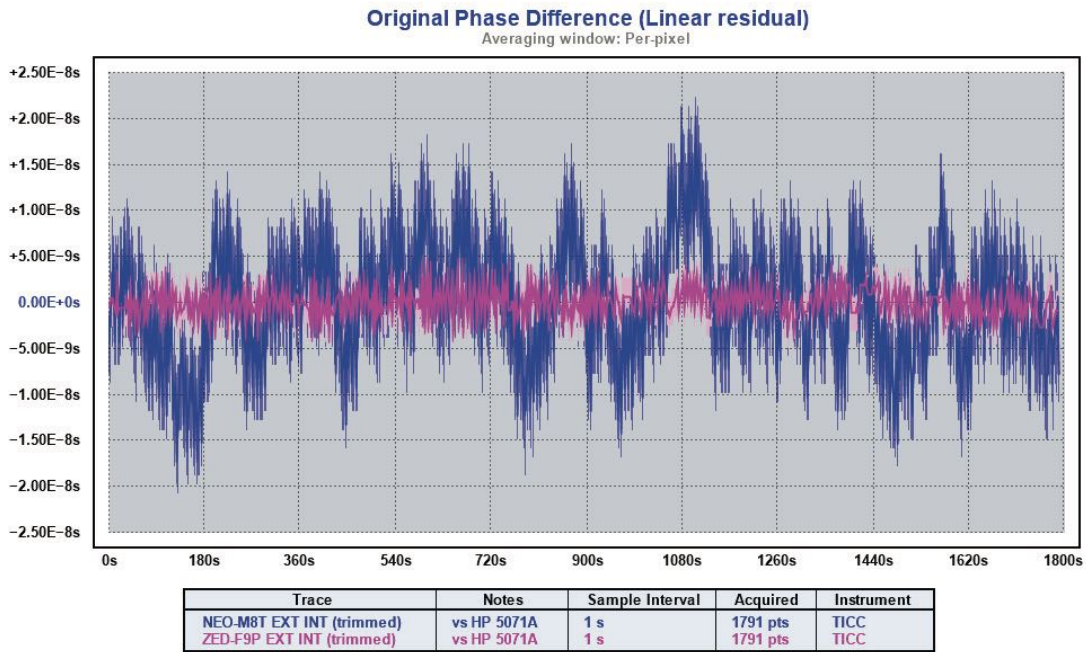


Figure 36: Raw Phase of EXT INT In Glitch-Free Segment

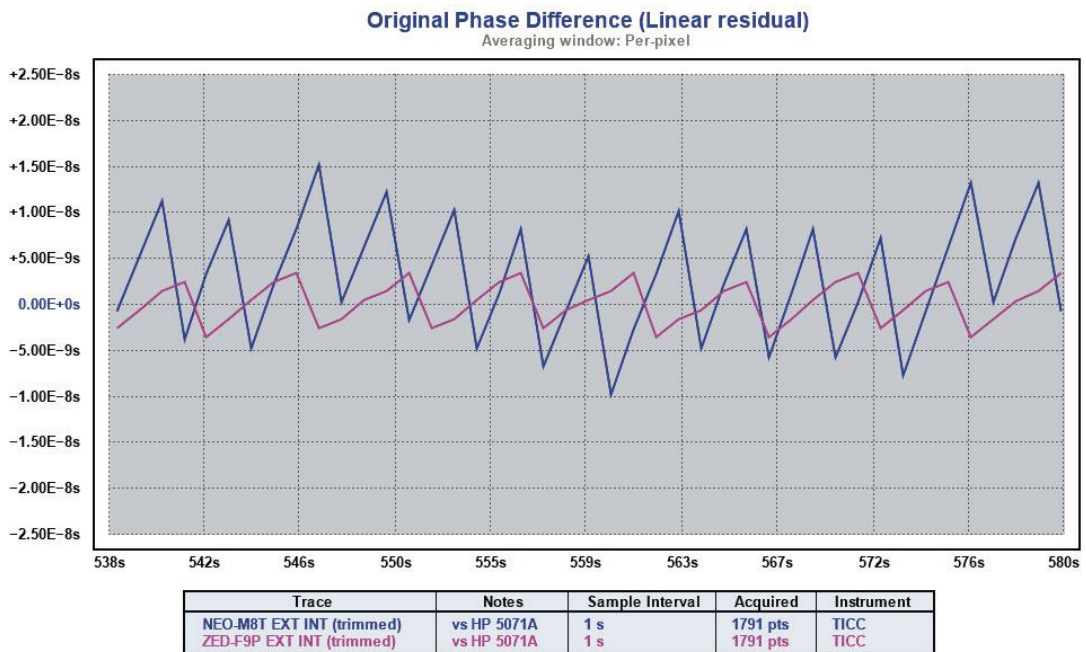


Figure 37: Zoomed Raw Phase of EXT INT In Glitch-Free Segment

Conclusion: If it were not for these issues, the performance of the EXT INT measurement appears similar to that of the TIMEPULSE output and could be useful for timestamping, as shown by figures 35 through 37. Investigation into the causes of the glitches continues.

7. IMPLICATIONS FOR GPS DISCIPLINED OSCILLATORS

A GPS Disciplined Oscillator (“GPSDO”) uses the pulse-per-second output of a GPS receiver to steer the frequency of a crystal oscillator. As shown in this paper, the GPS PPS signal has noise limiting its short-term performance, but over the long term that signal tracks the ensemble of atomic clocks used to control the GPS satellite constellation.

A free-running crystal oscillator can have very low noise in the short term, but it needs to be adjusted to its nominal frequency, and suffers from temperature sensitivity and long-term frequency drift due to aging, both of which limit its ultimate accuracy and stability.



Figure 38: Allan Deviation Intercept of GPS and OCXO²⁵

Figure 38 shows how the performance of a crystal oscillator is better at short measurement intervals, while the GPS is more stable over longer tau. The purpose of the GPSDO is to obtain the best of both worlds by steering the crystal oscillator frequency to follow the GPS PPS frequency. In effect, the oscillator acts as a “flywheel” to smooth the GPS noise.

This is normally accomplished by dividing the crystal oscillator output to create a PPS signal, comparing the timing of that signal to the GPS PPS, and using a phase lock loop to steer the crystal so that its PPS signal tracks that of the GPS receiver. The bandwidth, or time constant, of the loop controls where each input signal is dominant in the output, and the goal is to set the bandwidth so that the transition occurs at the intercept point to maintain the lowest ADEV

²⁵ Data courtesy of Tom Van Baak

over the entire range of tau. With a high quality oscillator and GPS as shown in Figure 38, the time constant is typically in the range of several hundred to two thousand or more seconds.



Figure 39: GPSDO Realization of Intercept Point²⁶

The green trace in Figure 39 shows the actual performance of a GPSDO overlaid on plots of the free-running oscillator, and GPS PPS, performance. This high-quality GPSDO shows the results obtainable when the underlying signals sources are good, and the control loop is properly tuned.

If the GPS PPS noise is lowered, the loop time constant can be decreased, causing the GPS to dominate at shorter tau. This means that the long term performance of the crystal oscillator becomes less important, and may allow use of a lower cost oscillator so long as its short-term stability is acceptable for the application.

²⁶ Data courtesy of Tom Van Baak

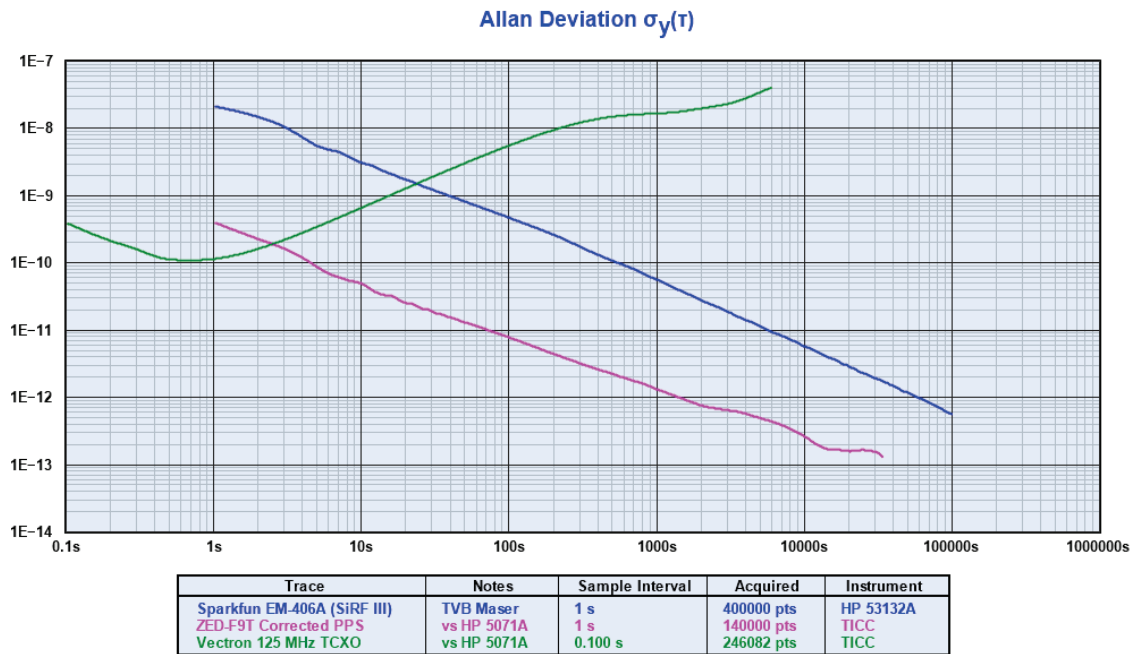


Figure 40: Allan Intercept of Low-Performance Oscillator

Figure 40 shows a less stable, less expensive, oscillator,²⁷ and also shows the impact of using a lower noise PPS source for control. With a mediocre GPS, the intercept point occurs at about 25 seconds, and the worst-case Allan Deviation is about 1.5×10^{-9} . By using the ZED-F9T receiver and applying sawtooth correction, the intercept point occurs at less than three seconds, and the worst-case Allan Deviation is about 2×10^{-10} – nearly an order of magnitude better.

Conclusion: Use of a dual-frequency GPS receiver together with application of sawtooth error correction may allow a GPS to be implemented with a lower-cost oscillator.

²⁷ This is a VHF temperature controlled crystal oscillator (TCXO) running at 125 MHz which is optimized more for low phase noise than for frequency stability.

8. POSITIONING PERFORMANCE

Although the focus of this paper is on timing, it is appropriate to describe the positioning performance of these units. All provide both standard NMEA output data, as well as proprietary binary format messages. Some have a “high precision” NMEA output that provides seven decimal places of latitude and longitude precision compared to four places in the standard mode. Some have an RTK processing engine that allows real time corrections to be applied to the output solution. Most support SBAS (satellite-base augmentation system) corrections with no external hardware, though oddly the two “P” series positioning receivers do not. The “8” series receivers support concurrent GPS and GLONASS operation, while the “9” series receivers allow concurrent reception of four constellations (GPS, GLONASS, BeiDou and Galileo).

A full exploration of GPS positioning performance capabilities and tools could occupy many pages. The following is not intended to be a comprehensive report and should be viewed instead as a limited comparison of performance in selected measurements.

8.1 Autonomous Positioning

As an initial test, all seven units were connected to the same antenna²⁸ and set to their factory default condition, except that NMEA high precision (seven decimal place latitude and longitude resolution) was enabled where available and SBAS and QZSS augmentation were turned off. The 8-series receivers used the GPS and GLONASS constellations, while the 9-series receivers had all four constellations enabled.

The NMEA “GGA” sentence (“GPS Fix Data”) from each receiver was simultaneously logged to a disk file for 12 hours.²⁹ That data was then processed by the “gpsprof” program that is provided with the gpsd software suite³⁰ to generate a scatter plot and Circular Error Probability statistics for each receiver.

Circular Error Probable (“CEP”) is a generally accepted metric for GPS positioning performance.³¹ It is roughly the radius of a circle containing a specified percentage of all data positions logged. 95 percent is most commonly used value. The gpsprof program also calculates a similar value for Elevation Probability (EP). Figure 41 shows the scatter plots and CEP and EP data for each receiver.

28 As noted earlier, the Trimble Zephyr Geodetic antenna used is capable of L1/L2 operation, but is not optimized for the carrier frequencies used by GLONASS and BeiDou. However, all the receivers were able to lock multiple GLONASS satellites with good carrier-to-noise ratios.

29 To obtain the best average position, it would be better to collect data for 24 or 48 hours, but time did not permit that long a measurement.

30 <https://gpsd.gitlab.io/gpsd/index.html>

31 See, e.g., <https://www.gpsworld.com/gps-accuracy-lies-damn-lies-and-statistics/>;
<http://www.dtic.mil/dtic/tr/fulltext/u2/a199190.pdf>

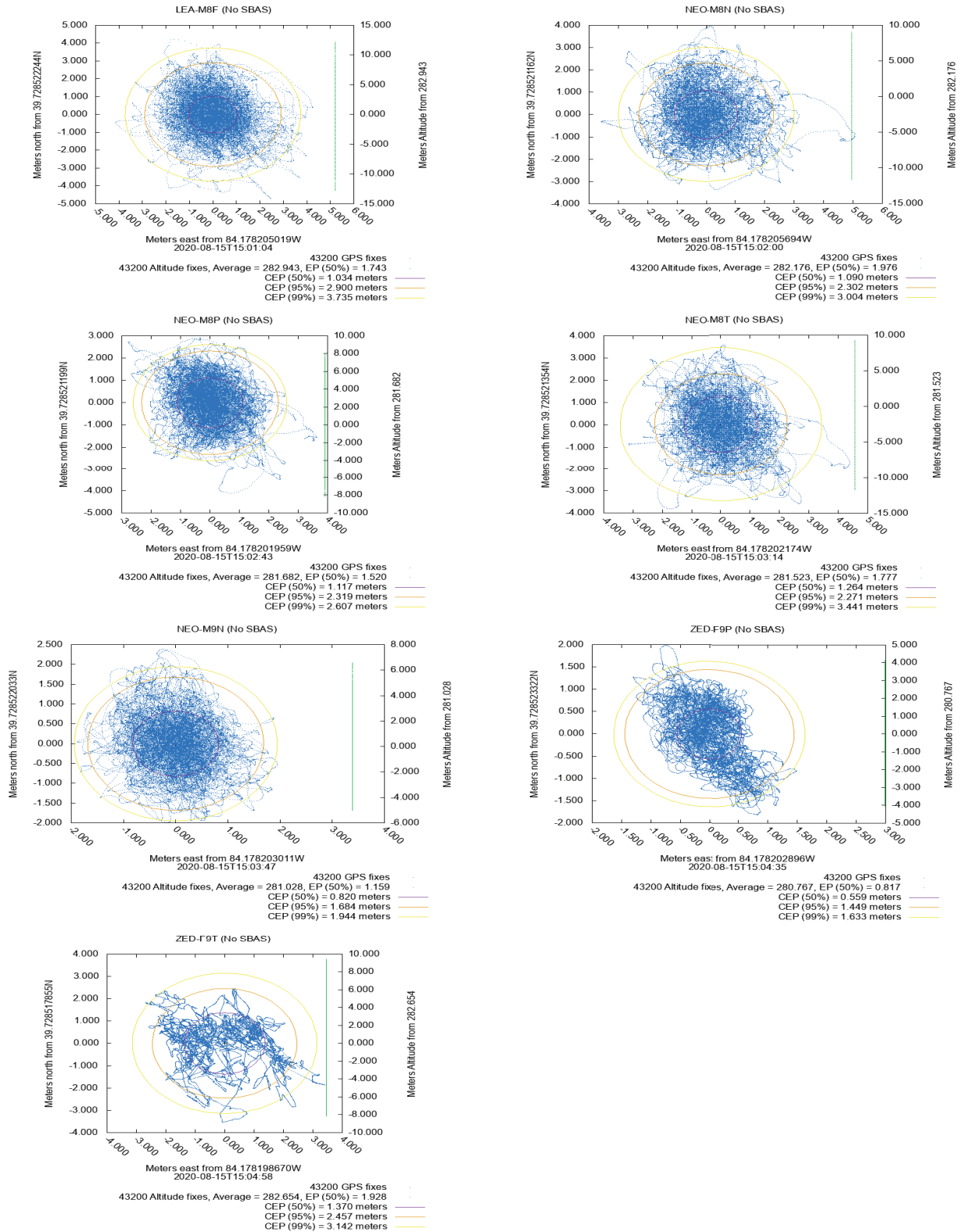


Figure 41: Circular Error Probability of Receiver Modules (via gpsprof)

Receiver	CEP – 50%	CEP – 95%	CEP – 99%	EP – 50%
LEA-M8F	1.034	2.900	3.735	1.743
NEO-M8N	1.090	2.302	3.004	1.976
NEO-M8P	1.117	2.319	2.607	1.520
NEO-M8T	1.264	2.271	3.441	1.777
NEO-M9N	0.820	1.684	1.944	1.159
ZED-F9P	0.559	1.449	1.633	0.817
ZED-F9T	1.370	2.457	3.142	1.928

Table 3: CEP/EP Values (via *gpsprof*)

For ease of comparison, Table 3 consolidates the CEP and EP results. It indicates that all the “8” series receivers perform similarly. The “9” series results are surprising. The single-frequency M9N performs significantly better than its M8N predecessor, and the ZED-F9P performs best of all the units, indicating that it uses a dual-frequency solution in its NMEA output. But the ZED-F9T, which one would think to have similar performance as the F9P, performs like the “8” series, and its scatter plot in Figure 42 looks quite different from that of the other receivers. Perhaps its timing firmware has not been updated to use a dual-frequency positioning solution, or there is some other issue.

Receiver	Latitude ³²	Longitude ³³	Altitude
LEA-M8F	39.xx8522244	-84.xx8205019	282.943
NEO-M8N	39.xx8521162	-84.xx8205694	282.176
NEO-M8P	39.xx8521199	-84.xx8201959	281.682
NEO-M8T	39.xx8521354	-84.xx8202174	281.523
NEO-M9N	39.xx8522033	-84.xx8203011	281.028
ZED-F9P	39.xx8523322	-84.xx8202896	280.767
ZED-F9T	39.xx8517855	-84.xx8198670	282.654
REFERENCE ³⁴	39.xx8519939	-84.xx8203511	281.521
AVERAGE	39.xx8521309	-84.xx8202774	281.825
MAX – MIN (m)	0.607	0.602	2.176
AVG – REF (m)	0.152	-0.063	0.304
MIN – REF (m)	-0.231	-0.415	-0.754
MAX – REF (m)	0.376	0.187	1.133

Table 4: Range of Average Positions

32 Values obfuscated to complicate ICBM targeting. At this latitude, 111029.3831 meters/degree.

33 Values obfuscated to complicate ICBM targeting. At this longitude, 85731.6808 meters/degree.

34 Reference position determined from 48 hour data collection using Trimble NetRS receiver, post-processed using IGS final data via NRCAN (<https://webapp.geod.nrcan.gc.ca/geod/tools-outils/ppp.php>)

As shown in Table 4, the average positions calculated by gpsprof for the seven receivers shows a range in both latitude and longitude of about 0.6 meter, and a range of about 2.2 meters in altitude. The average of the seven receivers' average positions was 15 centimeters for latitude, six centimeters for longitude, and 30 centimeters for altitude away from the antenna location as determined by post-processing of results from a geodetic receiver.

8.2 RTK Positioning (NEO-M8P and ZED-F9P)

The “positioning” series receivers (NEO-M8P and ZED-F9P) have an internal Real Time Kinematic (RTK) processing engine. By providing an input stream of RTCM³⁵ messages from a base or reference station, the receivers will apply corrections that can result in centimeter or even better positioning accuracy.

As a test, the NEO-M8P and ZED-F9P were configured to use RTCM corrections fed via the u-blox “u-center” software³⁶ which has an NTRIP client capability.³⁷ Correction data was obtained from the Ohio Department of Transportation virtual reference station network³⁸ which has a reference station a few kilometers from the author’s location. Data was collected from both receivers simultaneously. Due to time constraints and configuration issues it was only possible to collect data for the NEO-M8P with an RTK “FIX” solution for just under 8 hours. The ZED-F9P data run was more successful, but for comparability only data matching the time period collected from the NEO-M8P was analyzed.

As is shown in Table 4, the RTK processing engines work, providing significantly smaller CEP and EP than the autonomous measurements. The fact that the single- and dual-frequency receivers performed almost identically is surprising, but this may be a limitation of the reference network used. (The author currently has little experience configuring RTK systems.)

Receiver	CEP – 50%	CEP – 95%	CEP – 99%	EP – 50%
NEO-M8P	0.010	0.029	0.035	0.021
ZED-F9P	0.013	0.025	0.033	0.016

Table 4: Internal RTK Engine CEP/EP

The offsets in position shown in Table 5 are not what would be expected. Again, configuration issues may affect these results and further work needs to be done. Figure 43 is a scatter plot showing the spread of the data sets.

35 Real Time Correction Message protocol

36 <https://www.u-blox.com/en/product/u-center>

37 https://en.wikipedia.org/wiki/Networked_Transport_of_RTCM_via_Internet_Protocol

38 <http://www.dot.state.oh.us/Divisions/Engineering/CaddMapping/Survey/VRS/Pages/default.aspx>

Receiver	Latitude ³⁹	Longitude ⁴⁰	Altitude
NEO-M8P	39.xx8511818	-84.xx8194318	282.942
ZED-F9PN	39.xx8504462	-84.xx8179344	281.912
REFERENCE ⁴¹	39.xx8519939	-84.xx8203511	281.927
RANGE (m)	0.817	1.284	0.030
ZED-F9P – REF (m)	-1.718	-2.072	0.392
NED-M8p – REF (m)	-0.092	-0.788	0.421

Table 5: Internal RTK Engine Range of Positions

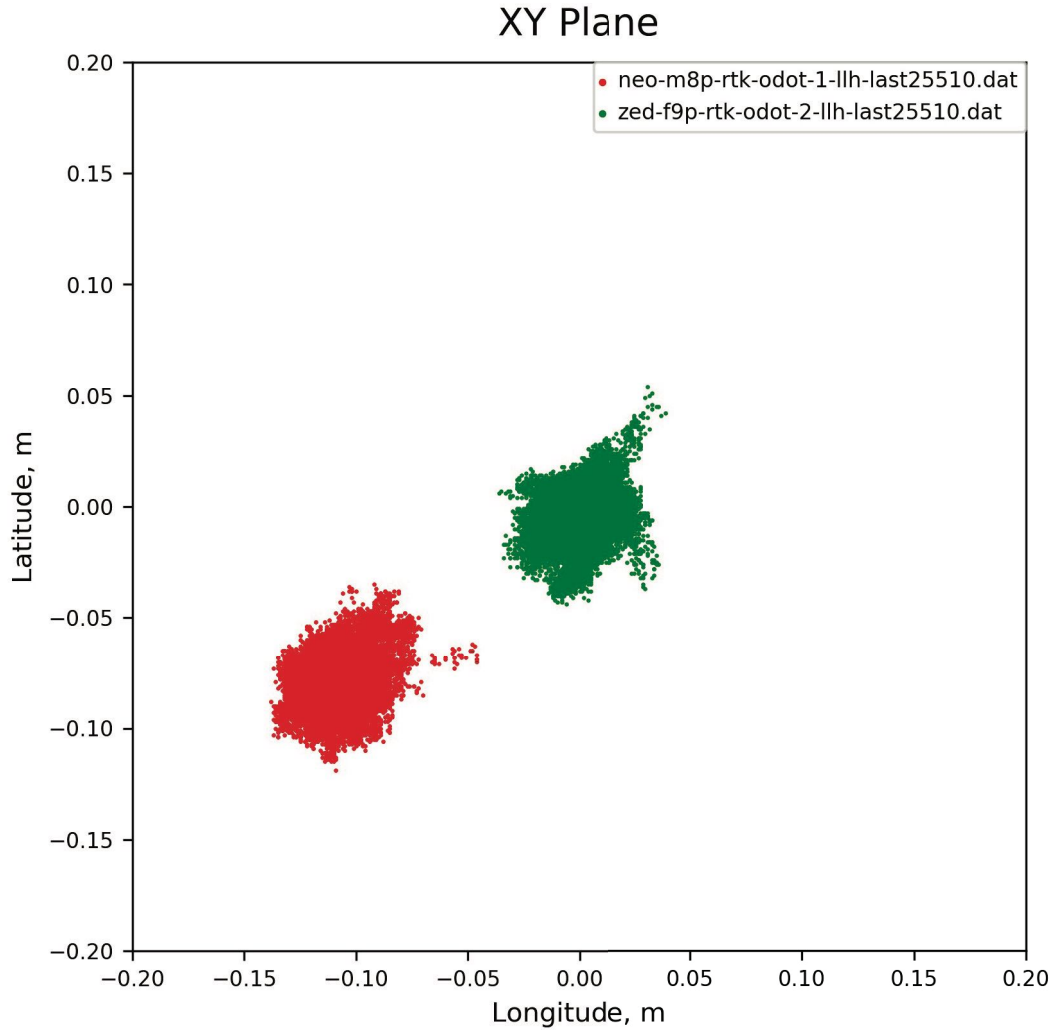


Figure 43: RTK Scatter Plot

39 Values obfuscated to complicate ICBM targeting. At this latitude, 111029.3831 meters/degree.

40 Values obfuscated to complicate ICBM targeting. At this latitude, 85731.6808 meters/degree.

41 Reference position determined from 48 hour data collection using Trimble NetRS receiver, post-processed using IGS final data via NRCan (<https://webapp.geod.nrcan.gc.ca/geod/tools-outils/ppp.php>)

8.3 Post-Processing

The greatest absolute positioning accuracy is generally achieved by post-processing raw satellite observation data using either a double-difference method employing corrections from known reference stations (such as is done by the Online Positioning User Service, or OPUS, web site operated by NOAA in the United States⁴²) or the Precise Point Positioning method that applies correction information based on observed satellite orbital and clock information (such as is done by the service offered by Natural Resource Canada at <https://webapp.geod.nrcan.gc.ca/geod/tools-outils/ppp.php?locale=en>).

Raw data was recorded simultaneously from the NEO-M8P and ZED-F9P using GPS only, and from the ZED-F9T using both GPS and GLONASS. After allowing several weeks for the final IGS correction data to become available, the data from each receiver was submitted to the NRCAN web site. For reference, data from a Trimble NetRS (dual frequency, GPS only) receiver was also submitted to NRCAN for processing.

	NetRS (GPS)		NEO-M8P (GPS)		ZED-F9P (GPS)		ZED-F9T (GPS+GLONASS)	
	24 Hour	Sigma (95%)	24 Hour	Sigma (95%)	24 Hour	Sigma (95%)	24 Hour	Sigma (95%)
LAT ITRF2014	39 xx 42.67100	0.0068	39 xx 42.66852	0.3601	39 xx 42.67067	0.0090	39 xx 42.67086	0.0048
LON ITRF2014	-84 xx 41.53109	0.0124	-84 xx 41.53533	0.4131	-84 xx 41.53164	0.0160	-84 xx 41.53226	0.0084
EL HGT ITRF2014	247.101	0.0247	247.21	0.6522	247.1254	0.0370	247.1548	0.0217

Table 6: PPP Results from NRCAN^{43,44}

Table 5 shows the results. As can be seen, the ZED-F9P and ZED-F9T performed comparably to the NetRS. In fact, the ZED-F9T was slightly better. There is a reason for this.

As noted in Section 2.3, the ZED-F9 series receivers are dual-frequency, but they decode only the L2C signal that is present on GPS satellites launched since 2005. There are still some satellites in operation that do not support L2C, and the ZED-F9 receivers receive only the L1 signal from those. The much older NetRS receiver, on the other hand, receives the legacy L2P(Y) as well as the L2C signal.

As a result, when constrained to the GPS constellation, the ZED-F9 receivers have fewer available L2 satellites to use, perhaps increasing dilution of precision, and that slightly reduces their positioning accuracy. By adding the GLONASS constellation, the receiver has more usable satellites in view, and its performance is thereby somewhat improved, as is the case for this ZED-F9T test.⁴⁵ Since the NetRS receiver does not support GLONASS, it now loses out to the ZED-F9's ability to use satellites from both constellations. This discrepancy will diminish as more L2C-capable satellites are launched.

42 <https://www.ngs.noaa.gov/OPUS/>

43 Values obfuscated to complicate ICBM targeting.

44 These elevations are quite different from those shown in other tables and plots. Those are referenced to the NAD88 datum which more-or-less represents mean sea level, while ITRF2014 is based on the ellipsoid. As is shown, the difference can be tens of meters. GPS elevation measurements are not straight-forward.

45 Note that for timing as opposed to positioning applications, it is better to use a single satellite constellation in order to avoid multiple and slightly different clock systems.

9. CONCLUSION

The u-blox ZED-F9 series receivers represent a new generation of affordable dual-frequency GNSS hardware. The dual-frequency approach allows significant improvements in both timekeeping, with improved short and long term frequency stability, and positioning.

The quantization error or “sawtooth” correction capability of some u-blox receivers allows significant reduction in short term timing noise, and the OD navigation mode improves long-term timing performance..

While the TIMEPULSE output of some receivers may be set to RF frequencies, the spectral purity of the output at those frequencies does not lend itself for use as a clock source in radio frequency systems.

The external interrupt (“EXT INT”) input of some receivers does not seem to be suitable for use as a precision timestamping counter due to unexplained glitches in the data output. The cause (and cure) of these glitches is as yet unknown.

The autonomous positioning performance of the receivers is generally similar, with the NEO-M9N and ZED-F9P having better CEP than the other units. The positioning receivers (NEO-M8P and ZED-F9P) can provide centimeter level results using their internal RTK engines and a nearby reference station. The dual-frequency receivers (ZED-F9P and ZED-F9T) can provide post-processed results similar to those of survey receivers.

APPENDIX 1
Receiver Models/Firmware Versions Tested

MODEL	FIRMWARE	PROT	HiPREC	qERR	SBAS	RAW	RTK	L2	PRICE
LEA-M8F	2.01	16.00	NO	NO	YES	NO	NO	NO	\$99.00
NEO-M8N	SPG-3.01	18.00	NO	NO	YES	NO	NO	NO	\$25.00
NEO-M8P-2	HPG 1.40REF	20.30	YES	YES	NO	YES	YES	NO	\$149.00
NEO-M8T	TIM 1.10	22.00	NO	YES	YES	YES	NO	NO	\$89.00
NEO-M9N	SPG 4.00	32.00	YES	NO	YES	NO	NO	NO	\$25.00
ZED-F9P ⁴⁶	HPG 1.11	27.10	YES	YES	NO	YES	YES	YES	\$199.00
ZED-F9T	TIM 2.01	29.00	YES	YES	YES	YES	NO	YES	\$199.00 ⁴⁷

FIRMWARE	=	Firmware version of unit tested
PROT	=	Protocol version
HiPrec	=	Seven-decimal-place NMEA output option
qERR	=	Availability of Quantization Error correction message
SBAS	=	SBAS reception/correction
RAW	=	Raw (pseudorange, doppler, carrier phase) output message
RTK	=	Internal RTK processing engine
L2	=	Dual Frequency L1/L2 receiver (L2C only)
PRICE	=	Quantity 1 price per u-blox.com visited 15 August 2020

⁴⁶ HPG v. 1.13 firmware, which adds SBAS support and has other improvements, became available in June, 2020. The tests conducted for this paper began before this update was available.

⁴⁷ In quantity volumes, the F9T unit becomes less expensive than the F9P.

Marco Bersani

APRS
PERFORMANCE AND LIMITS

Rev. 1.01 Aug. 2020

Table of contents

CHAPTER 1: Introduction	4
1.1 What is APRS.....	4
1.2 APRS digipeating	4
1.3 Operation of the APRS radio network.....	5
CHAPTER 2: Access to the shared channel	7
2.1 APRS frames and transmission channel capacity.....	7
2.2 Performance and limitations of the ALOHA protocol	8
2.3 CSMA: listen before transmitting.....	11
CHAPTER 3: APRS networks with a single digipeater	13
3.1 An APRS network with a single isofrequency digipeater	13
3.2 More efficiency: one digipeater, multiple uplink channels	17
3.3 APRS network with an analog repeater?.....	19
CHAPTER 4: Performance and limitations of the current APRS network	20
4.1 Impact of multiple digipeaters on an isofrequency network.....	20
4.2 Multiple Digipeating	23
CHAPTER 5: New possible arrangements	26
5.1 Foreword.....	26
5.2 A minimally invasive intervention: add a listening channel.....	26
5.3 Greater efficiency: a network articulated into two levels	27
5.4 IGATES: Widespread listening and broadcastings	28
CONCLUSIONS	31
Bibliography	32

Copyright (c) 2020 Marco Bersani

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation

Contacts: Marco Bersani, IK2PIH, bermarco72@gmail.com

Author's note

The APRS network has great potential on paper: the ability to track mobile stations, send weather data, telemetry and messages, all with the old equipment for the packet radio at 1200 baud AFSK, isofrequency and almost in real time.

Almost all users, however, have experienced difficulty using it, due to the congestion that occurs as soon as a fairly limited number of active stations send reports and messages.

In recent years, several strategies have been carried out to improve the performance of the existing network, by optimizing the network protocol with the introduction of new rules (new paradigm), by inviting stations to transmit at the bare minimum, by discouraging multiple digipeating and by rationalizing the distribution of digipeaters through the territory; however, it's not there yet a quantitative analysis of the efficiency of the system architecture that could serve as a guide to identify the main critical issues and to formulate coherent reorganization proposals.

It does not help the collection of statistics of the packets received and retransmitted by the digipeaters because collision information, essential for performance evaluation, is missing.

Unfortunately, the theoretical study of the APRS network is difficult for a number of reasons, including the generation of not completely random traffic, the variety and heterogeneity of hardware and software used, the coexistence of different mechanisms for accessing the shared channel, partly CSMA and partly ALOHA, the typical capture phenomenon of FM modulation (the strongest signals obscure the weakest signals), the noise of the radio channel.

Heavy simplifications are necessary to build a model that is simple enough to be treated with elementary means; the model proposed in the following pages does not pretend to accurately describe the functioning of the APRS network but wants to be a useful tool to evaluate its performance and to formulate reflections on possible new arrangements.

Note to the English version

I translated this book into English from Italian by myself, using Google translator and my limited knowledge of English (I ended studying English many years ago), so I am aware that this book needs a better translation and several adjustments. I hope it is understandable anyway and I apologize for the mistakes I definitely made. But above all, I hope readers will find the reflections I wanted to share with them useful and interesting.

The author

CHAPTER 1

Introduction

1.1 What is APRS

The APRS (Automatic Packet Reporting System) is a radio localization system developed by a radio amateur, Bob Bruninga, in the early nineties of the last century.

It was created as an experimental civil protection aid project in accordance with the obligations / duties of the radio amateur to make himself available together with his equipment in case of insufficient normal civil communications.

It is based on the transmission of digital packet radio signals and allows the dissemination of information on position, speed, direction and operating status of amateur radio stations, with the possibility of displaying data in real time on maps in the form of icons.

It also allows to send weather data, telemetry, information on emergency situations (road accidents, civil warnings and similar) and to exchange very short text messages.

To efficiently disseminate information to all stations on the network despite the reduced communication speed (1200 bps) it uses a "one to many" type protocol on a single radio channel in unconnected mode (UI frames of the AX.25 protocol) without the certainty that the recipient receives what has been sent.

Network coverage depends on the number of digital repeaters ("digipeaters") present in the area, which automatically forward local signals to the most distant stations.

Furthermore, if there is an IGATE internet access station in the area, the signals also reach the web and are visible all over the world.

Mobile stations, if equipped with a GPS receiver connected to the radio, can be located on the maps during their march.

To access the network via radio, a transceiver on the 144 MHz frequency is required (but there are frequencies destined for APRS also in short waves) connected to a PC with sound card or to an external TNC (modem); there are commercially available transceivers with built-in TNC.

To operate exclusively from the Internet (without the use of the transceiver), a PC or mobile phone with special software and an internet connection is sufficient.

And, of course, a valid radio amateur callsign is required, usually issued after passing an exam.

The reduced transmission speed can cause congestion on the radio channel if the stations are too numerous and / or configured with unnecessary or too frequent transmissions. To operate APRS via radio you should be aware of the functionality and limitations of the system.

If you access via the internet, however, you do not have these limitations and in a few seconds you can easily receive data from hundred of stations located in a wide area around you.

1.2 APRS digipeating

"Digipeater" is short for "digital repeater", a repeater for packet data instead of voice. Unlike a standard analog voice repeater, which receives on one frequency and retransmits everything it hears (useful information and noise) simultaneously on another frequency, the digipeater receives a packet of data, records it in an internal memory and then, a moment later, retransmits it, regenerated, usually on the same frequency.

Digital regeneration of the signal at each repetition is indispensable: signals that are not completely clean, suitable for the voice, may not be suitable for data transmission; disturbances not noticeable on the voice could be fatal for a packet data transmission.

In general, the appropriate signal levels for voice are not suitable for data packets, because in data transmission an entire packet must be received perfectly to retrieve any information contained therein. This is the reason why in digital transmissions it is preferred to regenerate the signals every time they are repeated rather than repeating the analog signal that carries the digital information, even if this involves a certain delay in the propagation of digital signals through repeater chains.

Digipeating is also much more critical for APRS than for conventional packet radio because APRS involves transmitting packet data to and from moving vehicles, while conventional packet radio is mainly used between fixed locations, usually with better antennas and more power.

Furthermore, with APRS there is no handshaking process: the sender transmits the packets and hopes that the recipients will receive them without errors. The receiving stations simply ignore the bad packets. This is the price of the one-to-many transmission nature of APRS compared to the connected nature of traditional packet radio.

Lastly, each transmission made on the same channel occupies not only the time taken by the original user to send it, but two, three additional time slots depending on the number of retransmissions required.

The indiscriminate use of three, four or more repetitions can significantly reduce the channel capacity.

1.3 Operation of the APRS radio network

The APRS network is made up of digipeaters located in favorable positions (wide radio coverage), all operating on the same frequency and identified by a radio amateur callsign and a generic alias, the same for all digipeaters.

APRS stations can transmit their packets locally, without requiring repetition by the digipeaters (even if this rarely happens) or request their packets to be repeated by one more digipeaters; in this case they can explicitly specify the sequence of digipeaters they want to use (this circumstance does not occur frequently) or, more often, simply require their packets to be repeated one or more times, referring to the digipeaters with their generic aliases (generic digipeating).

Generic digipeating is a very effective way for spreading APRS packets because stations that ask to be repeated do not have to know the structure of the network (this is particularly useful for mobile stations) and in the case of multiple repetitions the APRS packets can be diffused in all directions (UI flooding phenomenon).

Traditionally the digipeaters of the APRS network were divided into two categories: digipeaters with WIDE aliases, located in favorable geographical locations with wide radio coverage, and stations with RELAY aliases, often the same clients of the network.

The traditional "RELAY, WIDE" transmission path required the help of nearby cooperating home stations as a first step in the APRS network.

Usually WIDE digipeaters also responded to the "RELAY" alias, so a WIDE digipeater could also serve as the first step.

If you wanted to be repeated several times, you had to specify paths like "RELAY, WIDE, WIDE". Fixed stations that were able to be heard directly by WIDE digipeaters could specify WIDE or WIDE,WIDE type routes, avoiding reliance on RELAY stations, with benefit for local traffic.

As in the conventional packet radio, each digipeater in the chain "canceled" the callsign to which it had answered by adding an asterisk.

This type of path worked with any type of TNC used as a digipeater, (a dedicated firmware was not necessary) but it increased the size of the packets and, since the TNCs were not able to recognize the packets they had already repeated, caused unnecessary duplication of the packets if more than two WIDE repeats were performed.

With the growing popularity of APRS, channel congestion increased significantly.

To solve these problems since the second half of the 2000s (in Europe since 2008), a completely new convention was introduced.

The "New paradigm" path convention completely discards the use of "RELAY" and "WIDE" and uses only WIDEn-N type paths, already foreseen in the original specifications of the APRS protocol. In addition, digipeaters currently have internal firmware that can detect duplicate packets and avoid retransmitting them - but only if the path is a WIDEn-N path - and you can set them to ignore (or truncate) too long paths.

This significantly reduces channel congestion caused by duplicate packets and blocks out-of-area noise caused by the abuse of excessive repetitions.

To guarantee mobile stations the support of fixed stations in poorly covered areas, as happened with the RELAY stations, the new paradigm provides for the possibility of configuring home stations in a favorable position as "fill-in digipeaters" enabling digipeating with alias "WIDE1-1".

Mobile stations can thus use paths of the type "WIDE1-1, WIDE2-1" relying indifferently for the first jump on real digipeaters or on "fill-in digipeaters".

The "fill-in digipeaters", however, must be activated with caution and only if in the area access to the WIDE digipeaters by mobile / portable stations is difficult: the indiscriminate activation of these repeater stations causes in fact unnecessary duplication of packets and consequent reduction of the radio channel capacity.

Finally, the diffusion of the internet makes it possible today another particularly efficient mode of operation of the APRS network which uses gateways between the radio network and the web ("IGATES" stations).

Numerous IGATES listening stations are currently operating throughout the territory, forwarding the packets they receive to a network of servers on the internet (APRS-IS).

By connecting to the APRS-IS network via client software or special websites (for example aprs.fi) it is possible to view on maps APRS traffic from all over the world.

There are also IGATES stations operating in reverse, transferring part of the information from the APRS-IS servers to the radio channel; their use, however, is discouraged because the capacity of the radio channel is limited and is easily saturated by the considerable amount of information - even filtered - coming from the internet.

CHAPTER 2

Access to the shared channel

2.1 APRS frames and transmission channel capacity

The transmission of APRS data takes place on a single radio channel of the two meters band with AFSK modulation at 1200 bps, obtained by injecting into the microphone input of an FM transmitter an audio signal generated by a Bell202 modem.

It is a suboptimal solution from different points of view (low spectral efficiency, low transmission speed, sensitivity to disturbances), motivated by the low cost availability of the Bell modem in the years in which the standard was introduced (early '80s of the last century) and the possibility to transmit data with equipment designed for voice communications.

To estimate how many frames it is possible to transmit on such a shared channel, it is first necessary to evaluate the average length of an APRS frame (packet).

The APRS protocol uses only AX25 unnumbered information frames and divides the UI packet into seven sections as described in the following table:

1	Flag	1 byte	01111110 (hex 7E), delimits the start of the packet
2	Address	21 bytes	Source (7bytes), destination (7bytes), digipeater (7bytes)
3	Control	1 byte	hex 03 (frame UI)
4	Protocol ID	1 byte	hex F0 (no layer 3 protocol)
5	Information	128 bytes	The length of the field depends on the type of information, usually 80-90 bytes but can go up to 209 bytes.
6	Frame Check Sequence	2 bytes	CRC, 16-bit number calculated from source and destination, used to verify the integrity of the packet.
7	Flag	1 byte	01111110 (hex 7E), delimits the end of the packet

Table: structure of an APRS frame

The overall length of a frame is on average equal to 155 bytes, to which must be added 45 bytes of flags (hex 7E) transmitted at the beginning to allow time for the devices to switch to transmission, assuming a transmission delay (txdelay) of 0,3s at the speed of 1200bps.

In total, the length of an APRS packet (frame) is on average 200 bytes; at the speed of 1200 bps each frame takes on average 1.33 seconds to transmit.

In APRS technology, the capacity of the transmission channel refers to the “network cycle”, that is, to the minimum time interval arbitrarily established in which all stations transmit a packet at least once. Since APRS is - or should be - a real-time communication system, the network cycle should not last too long, to allow you to have a complete display of the situation in a short time.

Numerous documents agree on reasonable network cycle values between 10 and 30 minutes, typically 20 minutes. In 20 minutes at the speed of 1200bps (150 bytes / sec) it is theoretically possible to transmit up to 900 frames with a total length of 200 bytes each, for a total of 180000 bytes.

transmission speed		duration of a network cycle	Length of a frame	Total capacity of a network cycle	
bps	B/s	min	B	B	frames
1200	150	20	200	180000	900

Table: capacity of the APRS channel

2.2 Performance and limitations of the ALOHA protocol

Whenever multiple terminals of a computer network access a shared transmission medium, be it a cable or a radio channel, it is necessary to establish communication rules to prevent multiple terminals from transmitting data simultaneously giving rise to collisions with loss of information; the simplest method of accessing the shared channel is called "ALOHA". The "ALOHA" protocol was developed in 1970 by prof. Norman Abramson of the University of Hawaii to connect the computers of the various University sites scattered around the islands to a central server via a radio network.

The original version of the protocol, called "pure ALOHA", provides that as soon as a terminal on the network has a packet to transmit, it is sent without worrying about checking if the channel is idle or busy. The inevitable collisions are accepted, possibly providing for a retransmission of the transmitted packets not confirmed by the receiver. Assuming that the packets are all the same length, the communication vulnerability period is equal to twice the duration of a packet because a partial overlapping of the packets (at the limit of a single bit) is sufficient to lose their entire content.

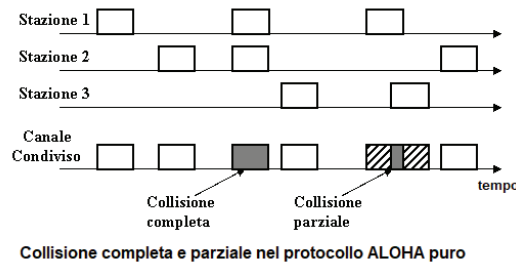


Figure: Pure ALOHA - collisions

A more advanced version of the protocol, called "slotted ALOHA", provides that the transmission takes place at precise time intervals ("slots") lasting the duration of a packet.

Each station is bound to begin its transmission at the beginning of a time slot.

In this case, the communication vulnerability period is equal to the duration of a single packet because the packets completely overlap or no collision occurs.

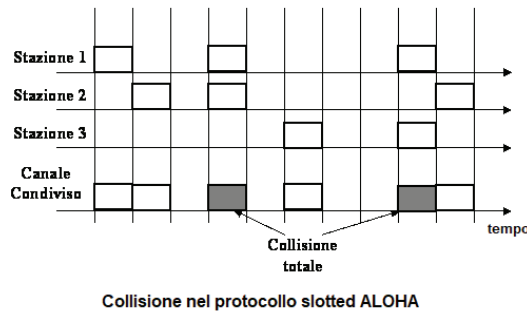


Figure: Slotted ALOHA – collision

This halves the likelihood of collisions and increases the efficiency of communication.

The realization of a "slotted ALOHA" shared access, however, requires a synchronization between stations, impossible to achieve with the hardware available for the amateur packet radio; for this reason, in the following we aim to evaluate how the pure ALOHA protocol influences the performance of communications on a shared channel without taking into consideration the slotted ALOHA access.

So let's imagine having a shared communication channel in which numerous stations send packets all of the same length randomly with the same signal level, without worrying about listening if the channel is idle ("pure ALOHA" access protocol to the shared medium).

Let's indicate with G the fraction of the channel capacity committed by traffic (normalized traffic) and with S the quantity of information successfully transmitted referring to the overall channel capacity (normalized throughput).

Theoretical studies show that, for such access to the shared channel, the normalized throughput S depends on the normalized traffic G according to the relationship

$$S = G e^{-2G}$$

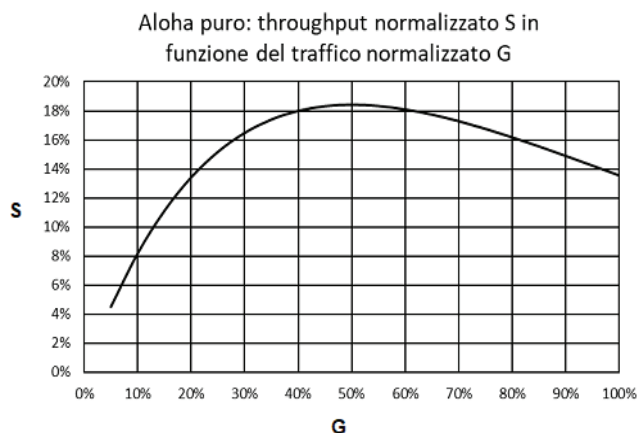


Figure: Pure ALOHA - throughput S as a function of traffic G

The maximum throughput is 18.4% of the total channel capacity at 50% of channel use ($G = 0.5$); in these conditions the maximum percentage of packets received successfully is equal to 36.8% of the total traffic generated. If the traffic exceeds 50% of the overall transmission channel capacity, not only does the throughput decrease, but the network becomes unstable and the communications collapse. To increase the probability of successfully transmitting packets it is necessary to reduce the load on the channel.

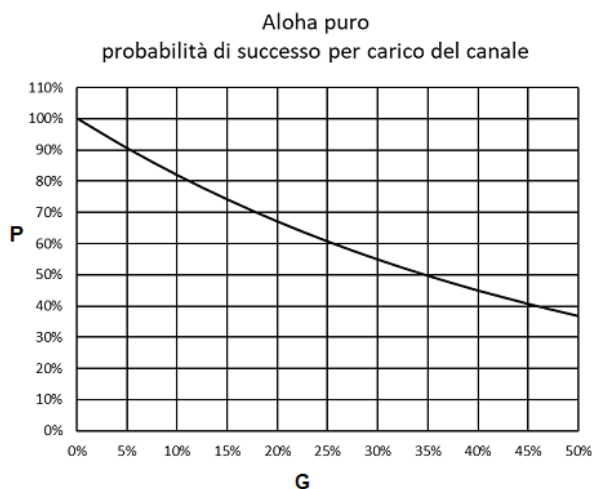


Figure: Pure ALOHA - Chance of success as a function of traffic G

In the packet radio context, the situation we have described above occurs when a certain number of terminals, which do not listen to each other (and therefore their collision control mechanism is ineffective), randomly transmit frames to a privileged station, which listens to all terminals.

Typically this occurred for conventional packet radio satellites that hosted a BBS (e.g. pacsat-AO16). These satellites had uplink channels dedicated to listening to terrestrial stations - which due to the great mutual distance did not listen to each other and could not operate any collision check - and a broadcast-type downlink channel.

Referring to the overall capacity of an APRS network cycle calculated previously (900 frames / cycle) it is possible to estimate the number of frames successfully receivable from the privileged station according to the load of the channel in a situation of this type:

Load	Throughput	Collisions	Idle	Chance of success	Frames sent	Frames successfully received
G	S	C=G-S	I=1-G	P = S/G	fr/cycle	fr/cycle
0%	0,0%	0,0%	100,0%	100,0%	0	0
5%	4,5%	0,5%	95,0%	90,5%	45	41
10%	8,2%	1,8%	90,0%	81,9%	90	74
15%	11,1%	3,9%	85,0%	74,1%	135	100
20%	13,4%	6,6%	80,0%	67,0%	180	121
25%	15,2%	9,8%	75,0%	60,7%	225	136
30%	16,5%	13,5%	70,0%	54,9%	270	148
35%	17,4%	17,6%	65,0%	49,7%	315	156
40%	18,0%	22,0%	60,0%	44,9%	360	162
45%	18,3%	26,7%	55,0%	40,7%	405	165
50%	18,4%	31,6%	50,0%	36,8%	450	166

We observe that in the situation of maximum possible channel load ($G = 50\%$) we will be able to transmit 450 frames per network cycle, of which however only 36.8% (166 frames) will be correctly received by the privileged station and almost 2/3 lost in collisions, making communication unreliable. By reducing the network load to 15% of the channel capacity, corresponding to 135 frames transmitted per network cycle, 100 frames will be received successfully. The probability of success of communication to the privileged station will now be 74.1%.

Assuming that the mobile APRS stations on the network are about half of the fixed stations, a total of 57 stations (19 mobile that transmit a frame every 5 minutes and 40 fixed that transmit a frame every 20 minutes) will be able to transmit with good probability of success to the privileged listening station.

The situation described, which does not currently occur in the APRS network, is a useful starting point for evaluating the performance of more complex situations that we will discuss later.

2.3 CSMA: listen before transmitting

CSMA (Carrier Sense Multiple Access) is a method of access control to the shared channel of the distributed type in which the channel traffic carries information on the control of its flow.

It was introduced in 1971 for use in packet radio channels and subsequently was also used in wired networks, in particular in ethernet networks in the CSMA / CD version.

You listen to the channel before transmitting; if the channel is idle, an algorithm is executed that decides when to transmit.

There are three versions of the protocol that are distinguished by the decision algorithm:

- non-persistent CSMA: listen to the channel before transmitting; if it is idle, transmit, if it is busy, wait for a random interval and try again.
- p-persistent CSMA: listen to the channel before transmitting; if it is busy, wait, if it is idle, transmit with probability "p".
- 1-persistent CSMA: listen to the channel before transmitting; if it is idle, transmit, if it is busy, wait for it to be idle and transmit.

In the TNCs used for packet radio the 1-persistent version is common, used in particular for digipeating, and is also supported the more elaborate p-persistent version which, although more efficient, is rarely used because in order to function correctly it requires that all stations set the same parameters that define the transmission probability.

An important parameter that affects the performance of all versions of the CSMA protocol is the collision window "a", caused by the delay in information propagation, defined by the relationship:

$$a = tp / tf$$

in which "tp" is the information propagation time and "tf" the duration of a packet.

Referring to the 1-persistent version, in the literature it is found that the throughput S depends only on the normalized traffic G and the amplitude of the collision window, and is given by:

$$S = \frac{G e^{-G(1+2a)} [1 + G + aG (1 + G + \frac{aG}{2})]}{G(1 + 2a) - (1 - e^{-aG}) + (1 + aG) e^{-G(1+a)}}$$

If the window "a" is sufficiently small (propagation time much less than the duration of the packet, a = 0) the previous relationship is simplified in:

$$S = \frac{G + G^2}{1 + G e^G}$$

As the amplitude of the collision window "a" increases, the performance decreases until it becomes comparable with the performance of the pure ALOHA protocol for a > 0.5

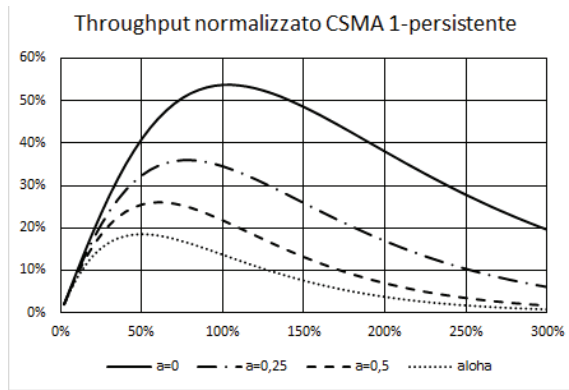


Figure: throughput S as a function of traffic G for different values of "a" and comparison with ALOHA

G	a=0	a=0,25	a=0,5	a=0,75	Pure ALOHA
2%	2,0%	2,0%	2,0%	1,9%	1,9%
10%	9,9%	9,4%	9,0%	8,5%	8,2%
20%	19,3%	17,5%	15,9%	14,4%	13,4%
30%	27,8%	24,0%	20,8%	18,0%	16,5%
40%	35,1%	28,9%	23,9%	19,8%	18,0%
50%	41,1%	32,4%	25,6%	20,2%	18,4%
60%	45,9%	34,6%	26,1%	19,7%	18,1%
70%	49,4%	35,7%	25,8%	18,6%	17,3%
80%	51,8%	36,0%	24,8%	17,0%	16,2%
90%	53,2%	35,5%	23,4%	15,4%	14,9%
100%	53,8%	34,5%	21,8%	13,6%	13,5%

Table: throughput S as a function of traffic G for different values of "a" and comparison with ALOHA

In the case of 1200 bps amateur radio packet communications that use FM voice communication devices, the average duration of the transmission of a previously estimated packet is equal to 1.33s, while the information propagation time is dominated by the switching time of the apparatuses and can be estimated at 0.3s; the collision window will then be

$$a = 0,3 / 1,33 = 0,23$$

resulting in a maximum throughput of 36% for total traffic offered $G = 80\%$.

The probability of successful transmission is not very high ($P = 36/80 = 45\%$); to increase the reliability of communications to 72% it is necessary to halve the traffic offered ($G = 40\%$) with a significant reduction in throughput ($S = 28.9\%$).

Despite this, in a network cycle of 20 minutes, 360 frames can be transmitted with good chance of success, equal to 144 stations, 72 mobile that transmit a frame every 5 minutes and 72 fixed that transmit a frame every 20 minutes.

The stations, however, should all listen to each other, which is highly unlikely given their radio horizon (on the plain, on average 5 km for a mobile station, 10 km for a fixed station).

CHAPTER 3

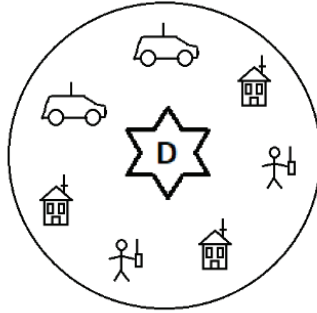
APRS networks with a single digipeater

3.1 An APRS network with a single isofrequency digipeater

Now let's imagine that there is only one digipeater on the network that listens to all stations with the same signal level (exposed terminal) and repeats the correctly received packets.

The stations on the network do not listen to each other (hidden terminals) but all listen to the digipeater, and transmit their packets randomly when they detect the channel to be idle, that is when the digipeater does not transmit.

On the contrary, when the digipeater is transmitting, no packets are emitted by the terminals.



We can think the network cycle as logically divided into two time slots, one used for the generation of traffic by network terminals with pure ALOHA protocol, which we could call “uplink slot”, and one used for the repetition of the valid packets received by the digipeater with 1-persistent CSMA protocol, which we could call the “downlink slot”.

Suppose for simplicity that the collision window of the CSMA 1-persistent protocol is very small, so that at low network loads all the throughput of the uplink slot is repeated without collisions.

Referring to the uplink slot and indicating with G the normalized traffic, we will have that throughput " S ", collisions " C ", inactivity factor " I " and transmissions chance of success " P " can be calculated with the previous expressions referring to a pure ALOHA channel.

But since all the throughput of the uplink slot is repeated by the digipeater, the overall capacity " K " of the channel referred to the uplink slot is

$$K = 1 + S = 1 + G e^{-2G}$$

To calculate the traffic expressions of the uplink slot G_k , the throughput S_k , the collisions C_k and the fraction of idle channel I_k according to the traffic in the uplink slot G normalized with respect to the overall capacity of the channel, it is sufficient to renormalize G , S , C and I compared to K

$$\begin{aligned} G_k &= G / K = G / (1 + G e^{-2G}) \\ S_k &= S / K = G e^{-2G} / (1 + G e^{-2G}) \\ C_k &= G (1 - e^{-2G}) / (1 + G e^{-2G}) \\ I_k &= (1 - G) / (1 + G e^{-2G}) \end{aligned}$$

Let us represent also in this case the normalized throughput S_k with respect to the normalized traffic G_k and compare it with the throughput graph of the previous case (shared channel with pure ALOHA protocol access without digipeater):

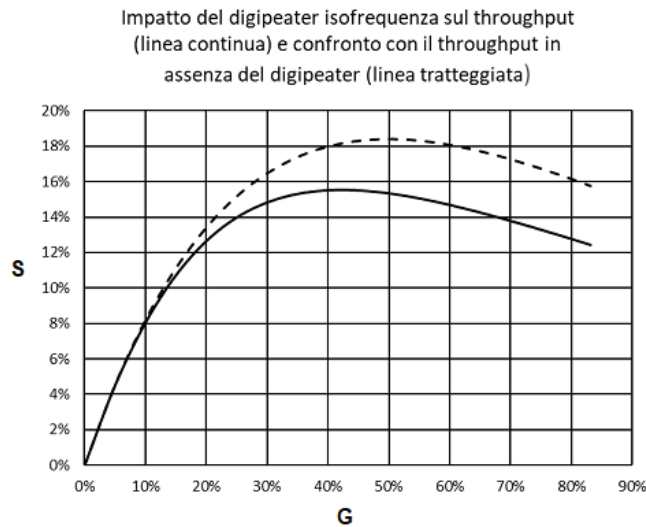


Figure: impact of the isofrequency digipeater on throughput (solid line) and comparison with throughput without digipeater (dashed line)

We observe that, as expected, the throughput is always lower than in the case of pure ALOHA in the absence of digipeater, and the maximum throughput $S_{k, \max} = 15,5\%$ occurs at a lower channel load ($G_k = 42.2\%$), being a part of the channel capacity engaged in digipeating.

With the same normalized traffic, the chance of transmission success also decreases, especially when the traffic in the channel is heavy.

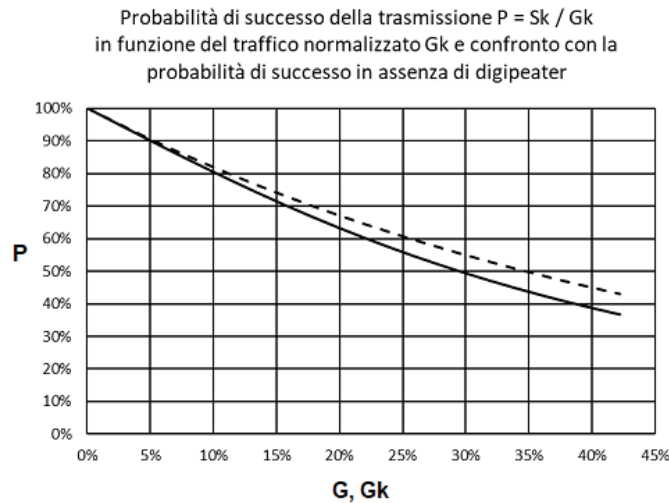


Figure: Chance of success $P=S_k / G_k$ according to the normalized traffic G_k and comparison with the chance of success without a digipeater

Uplink slot load	Normalized traffic	Normalized throughput	Normalized Collisions	Channel fraction used for digipeating	Channel fraction not used	Chance of success
G	Gk	Sk	Ck	Rk	lk	$P = Sk/Gk$
0%	0,0%	0,0%	0,0%	0,0%	100,0%	100,0%
5%	4,8%	4,3%	0,5%	4,3%	90,9%	90,5%
10%	9,2%	7,6%	1,7%	7,6%	83,2%	81,9%
15%	13,5%	10,0%	3,5%	10,0%	76,5%	74,1%
20%	17,6%	11,8%	5,8%	11,8%	70,5%	67,0%
25%	21,7%	13,2%	8,5%	13,2%	65,1%	60,7%
30%	25,8%	14,1%	11,6%	14,1%	60,1%	54,9%
35%	29,8%	14,8%	15,0%	14,8%	55,4%	49,7%
40%	33,9%	15,2%	18,7%	15,2%	50,9%	44,9%
45%	38,0%	15,5%	22,6%	15,5%	46,5%	40,7%
50%	42,2%	15,5%	26,7%	15,5%	42,2%	36,8%

Table: Distribution of channel use according to the traffic offered and chance of transmission success - network with a single isofrequency digipeater

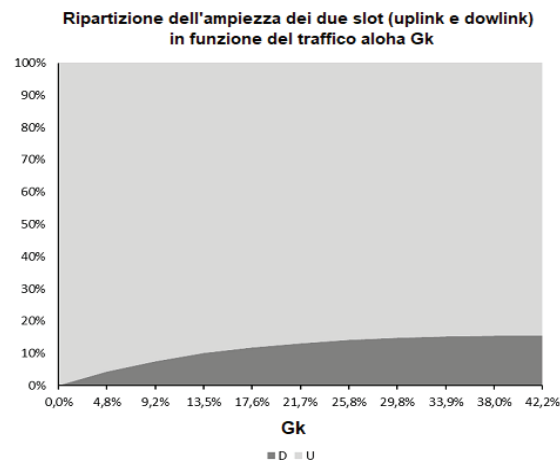


Figure: Distribution of the uplink and downlink slot according to aloha traffic Gk

Normalized load	Width of the downlink slot	Width of the uplink slot
Gk	D	U
0,0%	0,0%	100,0%
4,8%	4,3%	95,7%
9,2%	7,6%	92,4%
13,5%	10,0%	90,0%
17,6%	11,8%	88,2%
21,7%	13,2%	86,8%
25,8%	14,1%	85,9%
29,8%	14,8%	85,2%
33,9%	15,2%	84,8%
38,0%	15,5%	84,5%
42,2%	15,5%	84,5%

When the load of the network is reduced, however, the introduction of the digipeater on the channel slightly worsens throughput and chance of success compared to the situation without a digipeater, because decreasing the offered traffic reduces also the use of the channel for digipeating, and consequently increases the width of the uplink slot.

Referring to the overall capacity of the network cycle previously calculated (900 frames / cycle) it is possible to estimate also in this case the number of frames received successfully according to the network load.

Normalized traffic	Chance of success	Frames transmitted per network cycle	Frames received per network cycle
Gk	$P = S_k/G_k$	fr/cycle	fr/cycle
0,0%	100,0%	0	0
4,8%	90,5%	43	39
9,2%	81,9%	83	68
13,5%	74,1%	121	90
17,6%	67,0%	159	106
21,7%	60,7%	195	119
25,8%	54,9%	232	127
29,8%	49,7%	268	133
33,9%	44,9%	305	137
38,0%	40,7%	342	139
42,2%	36,8%	380	140

Table: APRS frames successfully received according on the traffic offered - network with a single isofrequency digipeater

In the situation of maximum possible channel load ($G_k = 42.2\%$) we will be able to transmit 380 frames per network cycle of which, however, only 36.8% (140 frames) will be correctly received and repeated by the digipeater and almost 2 / 3 lost in collisions, making the communication unreliable. By reducing the network load to 13.5%, corresponding to 121 frames transmitted by the APRS terminals per network cycle, as many as 90 frames will be successfully received and repeated, increasing the reliability of communication to 74.1% (only 1/4 of the packets transmitted will be lost in collisions).

Assuming also in this case that mobile stations in the network are about half of the fixed stations, we will be able to serve 53 stations with high probability of success, 17 mobile stations that transmit a frame every 5 minutes and 36 fixed stations that transmit a frame every 20 minutes, performances that do not differ much from those of an ALOHA channel without digipeater.

Therefore, in the case of an APRS network with a single digipeater, it is not justified to use separate frequencies for the uplink and downlink, because the throughput improvements would be moderate at the cost of an increase in terminals and digipeater complexity (they should be able to operate simultaneously on two different frequencies and should be equipped with two TNCs).

3.2 More efficiency: one digipeater, multiple uplink channels

A classic solution to increase the throughput of a single digipeater network is to use multiple uplink channels - for example in UHF -, to which the terminals access with the ALOHA protocol, and a separate downlink channel - for example in VHF - received by all stations, on which the digipeater repeats the overall throughput of the uplink channels, possibly in a redundant way and / or integrated by other information on the state of the network.

Since the maximum throughput for each uplink channel is 18.4% of its overall capacity, a single downlink channel could theoretically serve up to 5 uplink channels at the same speed, ($100 / 18.4 = 5.43$) and the maximum throughput on the downlink channel could reach 92% of the channel capacity ($18.4\% * 5 = 92\%$).

In order to transmit information redundantly, however, it seems appropriate to reduce the maximum number of uplink channels to 4, with a corresponding overall maximum throughput of 73.6% ($18.4\% * 4 = 73.6\%$).

Also in this case, to ensure a high probability of communication success, you must significantly reduce the load for each uplink channel:

Traffic on 1 uplink channel	Through-put for 1 uplink channel	Total through-put (4 channels)	Chance of success	Frames sent	Frames received
G	S	4 S	P	fr/cycle	fr/cycle
5,0%	4,5%	18,1%	90,5%	180	163
10,0%	8,2%	32,7%	81,9%	360	295
15,0%	11,1%	44,4%	74,1%	540	400
20,0%	13,4%	53,6%	67,0%	720	483
25,0%	15,2%	60,7%	60,7%	900	546
30,0%	16,5%	65,9%	54,9%	1080	593
35,0%	17,4%	69,5%	49,7%	1260	626
40,0%	18,0%	71,9%	44,9%	1440	647
45,0%	18,3%	73,2%	40,7%	1620	659
50,0%	18,4%	73,6%	36,8%	1800	662

Table: APRS frames successfully received according to the traffic offered and chance of success – single digipeater network with multiple uplink channels and a separate downlink channel

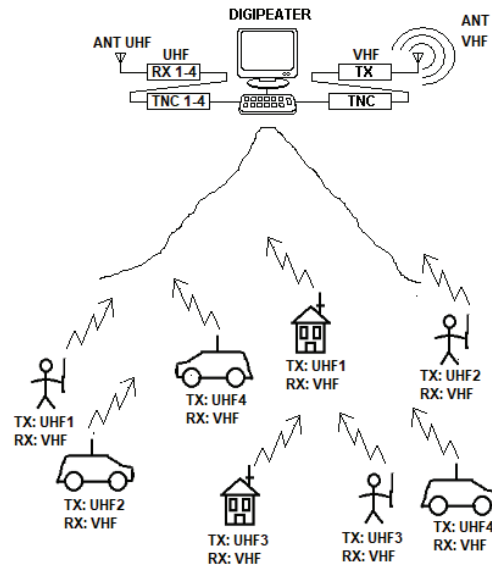
For a load of the single uplink channel of 15%, corresponding to 135 frames transmitted per network cycle on each uplink channel (540 total frames), 400 frames will be flawlessly received with a high probability of success (74.1%).

Assuming also in this case that the mobile stations in the network are about half of the fixed stations, we will be able to transmit with a high chance of success 228 stations, 78 mobile that transmit a frame every 5 minutes and 150 fixed that transmit a frame every 20 minutes, provided that the traffic is distributed equally across the four uplink channels.

To evenly distribute the stations among the four channels, rules could be established based -for example- on the letters of the callsign, on the nature of the stations (distinguishing between fixed and mobile stations) or it could be the same digipeater, depending on traffic conditions, to suggest some stations to move to another less busy channel.

Such a configuration could successfully serve a very large or very congested geographical area using a single digipeater, at the expense of an increase in complexity and cost of the terminal stations, which

should be equipped with two apparatuses, one for UHF transmission and one for reception in VHF (or a dual band transceiver), and two TNCs. The digipeater should be much more complex, equipped with: four VHF receivers tuned to the four uplink channels, four TNCs for reception, a VHF transmitter, a TNC for repeating received packets and dedicated software.



To reduce at least the complexity of the terminal stations, it is possible to think of compromise solutions that, in exchange for lower performance, allow network clients for the use of single band equipment and / or a single TNC.

Referring to the configuration just described, in fact, it is not strictly necessary that APRS terminals use two separate TNCs, one for transmission and one for reception: taking advantage that the transmission of each terminal station is bursty, a single TNC could be used by disabling the CSMA access control and agreeing not to listen during the short time interval in which the transmission of the APRS packet takes place (1.33 seconds).

The reception of the information could however be guaranteed by the redundancy of the digipeater transmission.

The uplink channels and the downlink channel could then be allocated on the same band, for example in VHF, so that the terminal stations could use single-band equipment.

In particular, by using only two uplink channels shifted by +600kHz and -600kHz with respect to the frequency of the downlink channel, you could exploit the standard offsets of the analog equipment currently used by the radio amateurs.

In this way the performance would be about half of the previously calculated one but no modification of the terminal stations would be required, only the activation of the standard offsets for accessing the analog voice repeaters.

Also in this case, rules could be established to distribute the stations among the two uplink channels: the possibility to allocate one frequency for the fixed stations and the other for the mobile stations appears interesting.

However, this solution would force to install resonant cavities on the digipeater to obtain the necessary selectivity, further increasing its complexity, cost and size.

3.3 APRS network with an analog repeater?

The AFSK modulation, very inefficient and used in low speed packet radio communications (1200 bps) to allow the reuse of normal analog FM radio equipment for voice communications, allows at least in theory the use of a conventional analog repeater for repetition of packets.

The main advantage of this solution is that, in such a similar configuration, the problem of the hidden terminal no longer exists: given that all the stations listen to each other in real time via the analog repeater, the CSMA mechanism of shared channel access (carrier detection) installed in the terminal TNCs is effective. Another advantage is the absence of delay in the propagation of the information: the network terminals immediately listen to the repeated packets as they are generated, having the feeling to be connected to a network which, despite the low transmission rate, responds promptly, making it possible to exchange messages and bulletins in real time.

The idea is almost as old as packet radio: an old document by D. Engle, KE6ZE, entitled "Packet radio timing considerations", examined the performance of three different connection modes, direct, through an analog repeater and through a conventional digipeater, concluding that the use of an analog repeater is more efficient than conventional digipeating.

The author conducted experiments on a standard voice repeater with a transmission delay of 500ms. A few years later (1987) R. Finch and S.Avent, in another document "A duplex packet radio repeater approach to layer one efficiency", resumed and perfected the idea, highlighting how a sufficiently fast turn-around was the key to an essentially transparent operation, as well as the cleanest possible audio chain to reduce the number of errors, proposing the use of special repeaters instead of the reuse of FM voice repeaters, they say - and not wrongly - inadequate.

If we imagine in fact to use conventional FM repeaters, the switching / activation times of the transmitters are at least double compared to direct communication, and the size of the collision window of the CSMA protocol is at least double compared to that calculated in the previous chapter ($a = 0.46$ instead of $a = 0.23$). For such an amplitude of the collision window the theoretical calculations indicate a maximum throughput $S = 26\%$ for normalized traffic $G = 60\%$ (reliability 43.3%), performance not much higher than that of an ALOHA channel; if we add that the AFSK signal degrades passing through the repeater - and therefore increases the probability of errors - the gap between the two solutions is further reduced. At low loads, necessary to have more reliable communications, the gap between the solutions becomes negligible:

	Offered traffic G	Throughput S	Chance of success
1-persist CSMA($a=0,5$, analog repeater) (*)	20%	15,9%	79%
ALOHA (up/downlink on separate channels)	20%	13,4%	67%
ALOHA + isofrequency digipeating	20%	12,5%	64%

(*) the degradation of the signal introduced by the analog repeater is neglected

In conclusion, the use of an analog repeater can be a valid solution for the creation of an APRS network, provided that the switching times of the repeater are very short and that the repeater distorts the signal as little as possible.

The greatest advantages are with heavy traffic; when the network is lightly loaded, the performance is not much higher than conventional solutions.

CHAPTER 4
Performance and limitations of the current APRS network

4.1 Impact of multiple digipeaters on an isofrequency network

The results we reached in the previous chapter refer to a network in which terminals do not listen to each other and access randomly the shared radio channel and in which operates only one digipeater which listens to all the stations.

Let us now try to evaluate the impact on the local throughput of the presence of adjacent digipeaters operating on the same frequency that listen to each other, making multiple digipeating possible (which however is a discouraged practice).

For simplicity we consider the case in which each digipeater listens and serves only the terminals of its area, that multiple digipeating does not occur and that the use of bandwidth for the repetition of the packets by the digipeaters is negligible compared to the overall capacity of the channel, confusing for each digipeater the uplink slot with the entire channel.

By indicating with G the total traffic in the uplink slot of each digipeater, the overall throughput S in the uplink slot of each digipeater will be equal to

$$S = G e^{-2G}$$

But of all the throughput of the uplink slot we are concerned only with the useful fraction produced by the local traffic; if we imagine that the proportion between useful fraction of throughput " S_L " and overall throughput " S " is the same as that between local useful traffic " G_L " and overall traffic " G " we will have that:

$$S_L = S (G_L / G)$$

By indicating with N the number of adjacent interfering digipeaters, with " S_L " the traffic repeated by them - which, since multiple digipeating does not occur, is configured as an ALOHA disturbance in the local uplink slot - and with G_L the normalized traffic in the uplink slot produced from local stations only, we will have that total traffic " G " in the uplink slot will be

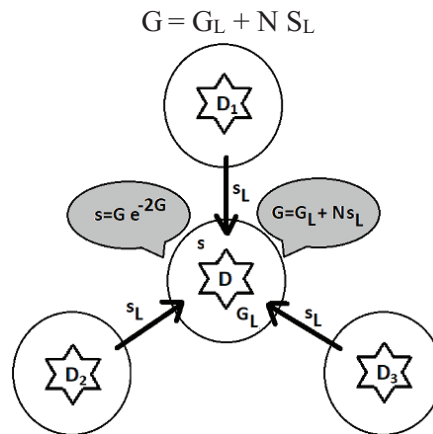


Figure: impact of adjacent digipeaters on the uplink slot of the central digipeater: only the G_L fraction of the traffic of the upload slot is generated by the local terminals, the rest is disturbance (Ns_L); throughput depends on overall traffic G (local traffic + noise).

By combining the two previous relationships we will have:

$$S_L = (G_L / G) G e^{-2G}$$

Simplifying for G and expressing GL as a function of G:

$$S_L = (G - N S_L) e^{-2G}$$

and developing the calculations we will get

$$S_L / (G - N S_L) = e^{-2G}$$

$$(G - N S_L) / S_L = e^{2G}$$

$$(G / S_L) - N = e^{2G}$$

$$G / S_L = e^{2G} + N$$

$$S_L / G = 1 / (e^{2G} + N)$$

$$S_L = G / (e^{2G} + N)$$

The relation obtained allows to calculate the useful throughput fraction SL as a function of traffic G and has significance for values of $G < 0.5$.

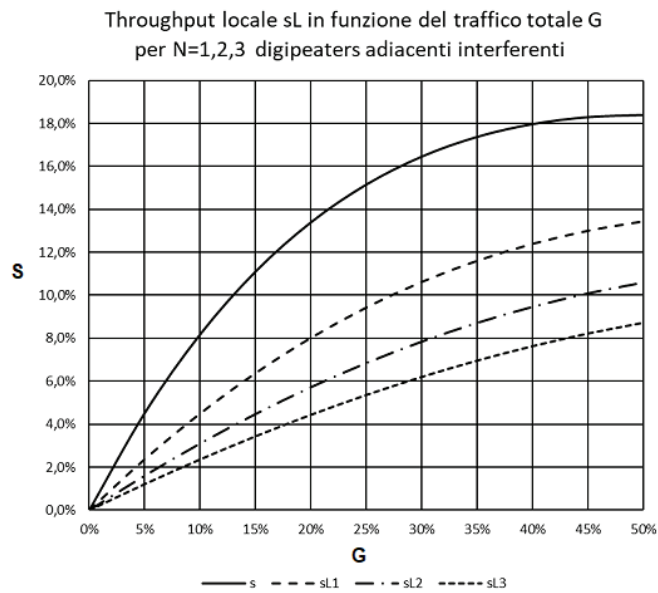


Figure: Local (useful) throughput sL according to the overall traffic G for N=1,2,3 interfering digipeaters

Total traffic uplink slot	Total Throughput	Local throughput (useful)			Chance of success
		1 Interfering digi	2 Interfering digis	3 Interfering digis	
G	S	SL1	SL2	SL3	P
0%	0,0%	0,0%	0,0%	0,0%	100,0%
5%	4,5%	2,4%	1,6%	1,2%	90,5%
10%	8,2%	4,5%	3,1%	2,4%	81,9%
15%	11,1%	6,4%	4,5%	3,4%	74,1%
20%	13,4%	8,0%	5,7%	4,5%	67,0%
25%	15,2%	9,4%	6,9%	5,4%	60,7%
30%	16,5%	10,6%	7,8%	6,2%	54,9%
35%	17,4%	11,6%	8,7%	7,0%	49,7%
40%	18,0%	12,4%	9,5%	7,7%	44,9%
45%	18,3%	13,0%	10,1%	8,2%	40,7%
50%	18,4%	13,4%	10,6%	8,7%	36,8%

Table: Local throughput as a function of total offered traffic with 1,2,3 interfering digipeaters

Remembering that $SL = S (GL / G)$, we can derive the corresponding local traffic $GL = (SL / S) G$, which allows us to estimate the maximum number of useful frames that can be transmitted in the local uplink slot per network cycle, and therefore the maximum number of stations that each digipeater can serve.

Total traffic uplink slot	Local throughput (useful)			Local frames transmitted per network cycle		
	1 Interfering digi	2 Interfering digis	3 Interfering digis	1 Interfering digi	2 Interfering digis	3 Interfering digis
G	GL1	GL2	GL3	fr/cycle	fr/cycle	fr/cycle
0%	0,0%	0,0%	0,0%	0	0	0
5%	2,6%	1,8%	1,3%	24	16	12
10%	5,5%	3,8%	2,9%	49	34	26
15%	8,6%	6,0%	4,7%	78	54	42
20%	12,0%	8,5%	6,6%	108	77	60
25%	15,6%	11,3%	8,9%	140	102	80
30%	19,4%	14,3%	11,3%	174	129	102
35%	23,4%	17,6%	14,1%	210	158	127
40%	27,6%	21,1%	17,0%	248	190	153
45%	32,0%	24,8%	20,3%	288	223	182
50%	36,6%	28,8%	23,8%	329	259	214

Table: locally transmissible frames with 1,2,3 interfering digipeaters

Referring to the case of three adjacent interfering digipeaters, in the situation of maximum possible load of the uplink slot ($G = 50\%$) we will be able to transmit 214 frames per network cycle, of which however only 36.8% (79 frames) will be correctly received and repeated by the local digipeater and almost 2/3 lost in collisions, making communication unreliable.

By reducing the load on the network in the uplink slot to 15%, corresponding to 42 frames transmitted by the terminals in the field per network cycle, the reliability of communication will increase to 74.1% and it will be possible to successfully receive and repeat 31 frames per cycle 20-minute network, i.e. 1 frame every 39 seconds. Assuming also in this case that the mobile stations in the network are about half of the fixed stations, each digipeater in a 20-minute network cycle can serve with high chance of success only 21 stations, 7 mobile stations that transmit a frame every 5 minutes and 14 fixed stations that transmit a frame every 20 minutes. We observe how the presence of adjacent digipeaters on the same channel heavily penalizes local throughput.

4.2 Multiple Digipeating

Now suppose that, in the scenario outlined, a station - which constitutes an exception - sends a frame requesting to be repeated by multiple digipeaters. We ask ourselves: what is the probability of the packet reaching at its destination by successfully spreading over the network?

Let's initially consider the case of the propagation of the packet in a chain of adjacent repeaters, each serving its own "cell". A packet propagating in a chain of N adjacent repeaters will be subject to ALOHA statistics N times; in particular, the chance that it is correctly repeated through the chain will be the product of the probabilities that it has to successfully pass through every single digipeater.

Assuming that all repeaters are identical and are in the same load conditions G , and indicating with P the probability that a packet is correctly repeated by a digipeater - which depends on the load G of the cell itself -, we will have the probability P_N that the packet has to successfully pass through the whole chain of digipeaters will be

$$P_N(G) = [P(G)]^N$$

Total traffic G	Chance of success of the transmission			
	1 repetition (local)	2 Repe-titions	3 Repet-itions	4 Repe-titions
0%	100,0%	100,0%	100,0%	100,0%
5%	90,5%	81,9%	74,1%	67,0%
10%	81,9%	67,0%	54,9%	44,9%
15%	74,1%	54,9%	40,7%	30,1%
20%	67,0%	44,9%	30,1%	20,2%
25%	60,7%	36,8%	22,3%	13,5%
30%	54,9%	30,1%	16,5%	9,1%
35%	49,7%	24,7%	12,2%	6,1%
40%	44,9%	20,2%	9,1%	4,1%
45%	40,7%	16,5%	6,7%	2,7%
50%	36,8%	13,5%	5,0%	1,8%

Table: Chance of successful transmission in the case of a sequence of multiple repetitions

Probabilità di successo della comunicazione
con digipeating multiplo in funzione del
traffico G nelle celle

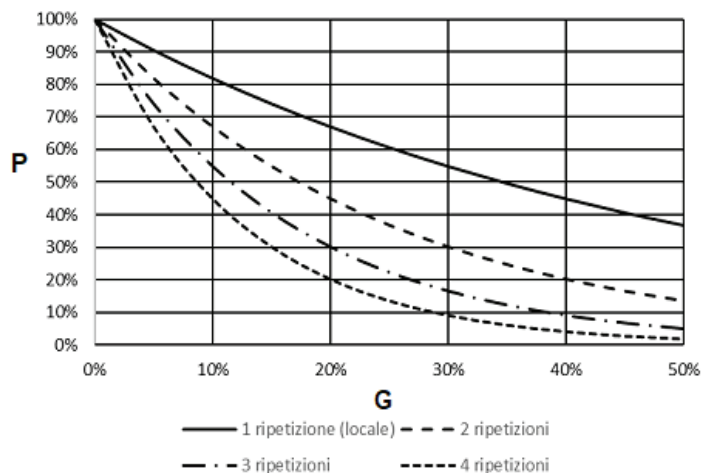
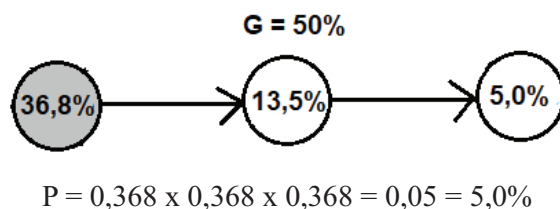


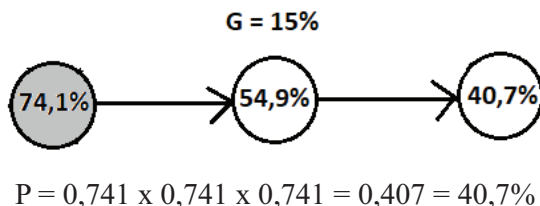
Figure: Chance of successful transmission in the case of a sequence of multiple repetitions according to the traffic G in the cells

The probability that the packet arrives at its destination therefore depends therefore both on the load of the cells “G” and on the number of N cells that the packet must go through to arrive at its destination.

In the situation of maximum possible load of the cells (G = 50%, corresponding to 23.8% of useful traffic considering for each digipeater the disturbance caused by three other interfering digipeaters) the packet can reach the adjacent cell with a probability of 13.5% , and the next with probability 5%.



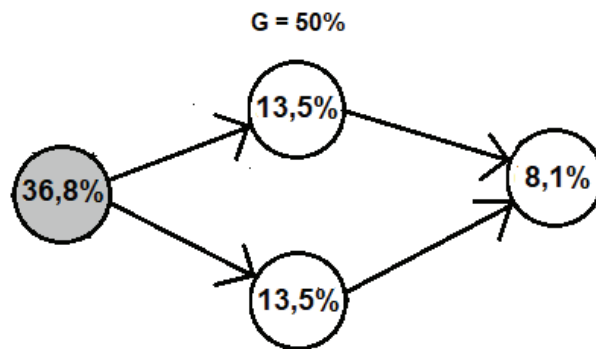
By reducing the load of the network in the cells to 15%, (corresponding to 4.7% of useful traffic) the chance to reach the adjacent cell will increase to 54.9%, and the immediately following one will be 40.7%.



If, to reach the recipient, the repeated packet travels several paths, the probability of success is greater because communication becomes redundant; remembering that the reliability of a redundant system R is given by

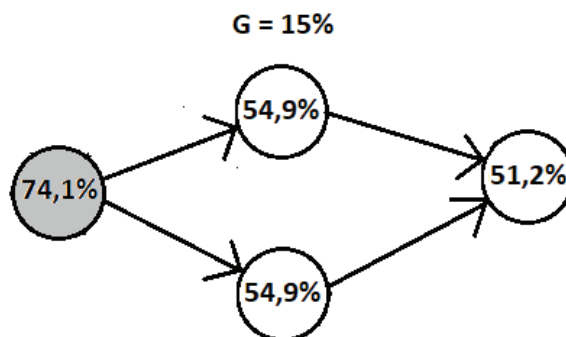
$$R = 1 - (1 - R_1) (1 - R_2) \dots (1 - R_N) ,$$

in the maximum load situation of the cells ($G = 50\%$, $GL = 23.8\%$ with 3 interfering digipeaters), referring to the figure below and remembering that the digipeaters are all identical and loaded the same way, the overall reliability (chance of success) of the communication from the leftmost cell to the rightmost cell (from sender to recipient) can be calculated as follows:



$$\begin{aligned}
 P &= 0,368 \times [1 - (1 - 0,368)^2] \times 0,368 = \\
 &= 0,368 \times 0,601 \times 0,368 = \\
 &= 0,081 = 8,1\%
 \end{aligned}$$

In the same situation, reducing the overall load of the cells to 15% ($GL = 4.7\%$ with 3 interfering digipeaters) the overall reliability of communication will increase to 51.2%.



$$\begin{aligned}
 P &= 0,741 \times [1 - (1 - 0,741)^2] \times 0,741 = \\
 &= 0,741 \times 0,933 \times 0,741 = \\
 &= 0,512 = 51,2\%
 \end{aligned}$$

We observe, as we expected again, that multiple digipeating has a fair chance of success only if the network load is very low.

CHAPTER 5

New possible arrangements

5.1 Foreword

The simplified analysis of the functioning of the APRS network made in the previous chapters suggests that the weak point of the current network is the interference that the uplink slot of each cell undergoes from adjacent digipeaters. Researching new configurations that do not suffer from this problem, we cannot ignore the need for reuse of existing equipment (radio and TNC), possibly without the need to make changes: it is very difficult to think that users will invest time and economic resources in a technology outdated as is the packet radio. The solutions proposed in this chapter try to circumvent the problem by giving up the constraint of a completely isofrequency network without increasing too much - or not increasing at all - the complexity of terminal stations and repeaters.

5.2 A minimally invasive intervention: add a listening channel

A non-invasive intervention would consist in equipping the existing digipeaters with a UHF only listening port, to be used as an additional uplink port, leaving the operation of the isofrequency network in VHF unchanged.

Terminal stations could access the network in two ways:

- isofrequency in VHF, using a VHF radio and a TNC as now;
- transmitting in UHF and receiving in VHF, using a dual band transceiver and a single TNC, disabling the CSMA access mechanism to the shared channel.

In this way, the traffic repeated by the digipeaters would no longer congest the UHF uplink channel, which for each cell could always be used to its maximum capacity for ALOHA access ($G = 50\%$, $S = 18\%$). Stations that did not wish (or could not) adapt to the new access method would continue to access the APRS network isofrequency in VHF as before.

This solution is interesting because it would allow a gradual transition to the new configuration.

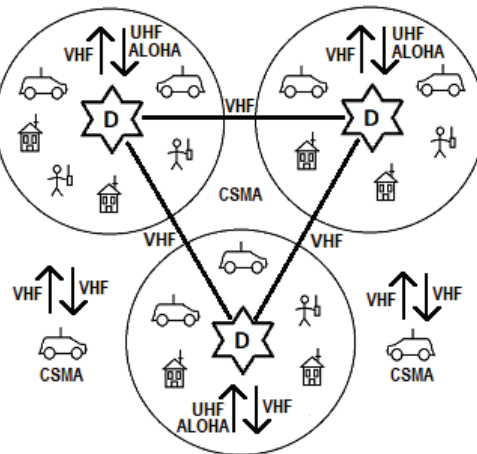


Figure: terminal stations transmit in UHF, digipeating occurs in VHF. Stations wishing to access the network as before, isofrequency in VHF, can continue to do so.

However, it should be emphasized that the improvement of the uplink performance, however strategic given the weak power of the terminal stations, causes a greater saturation of the VHF network with negative outcomes on communications between the digipeaters.

In addition, stations that transmit in UHF must accept the loss of information caused by the fact that they cannot listen to the downlink channel in VHF when they transmit.

However, as we said before, this loss is very limited and is not a serious problem because the transmission of packets is bursty.

5.3 Greater efficiency: a network articulated into two levels

A more efficient solution is to articulate the network into two different hierarchical levels, a lower level (LAN) in which local traffic is processed, and a higher level in which interconnection between cells (WAN) takes place.

Operationally, the network can be fragmented into many independent cells operating at different frequencies (for example in UHF), each served by its own digipeater.

The interconnection between the cells (WAN), which makes multiple digipeating possible, can take place on a separate frequency, for example in VHF.

The terminal stations would operate isofrequency in UHF with the usual hardware and software; only the digipeater would be more complex, having to be equipped with two TNCs, one for the management of the belonging cell, combined with a UHF transceiver, the other to communicate with the other digipeaters, combined with a VHF transceiver.

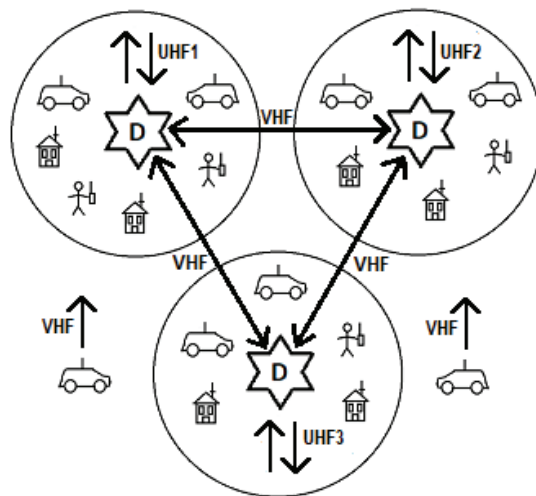


Figure: the network, completely via radio, divided into independent UHF cells and interconnected in VHF. Some stations, which only want to be tracked, can access the network in VHF.

In this configuration, traffic between digipeaters (data transfer between different cells) does not interfere with local traffic either in uplink or downlink: the APRS stations listen only to their own digipeater and are not disturbed either by the stations or by the digipeaters of the neighboring cells.

Within each cell, throughput can reach 15.5% of the transmission channel capacity for a traffic equal to 42.2% of the total channel capacity.

Adjacent cells operate on different UHF frequencies so as not to interfere with each other, but the reuse of frequencies can occur for cells far enough away that they do not listen to each other.

At the upper hierarchical level (WAN), traffic between digipeaters is reduced to only the packets for which multiple digipeating is required, with great benefit for the entire network (both throughput and reliability of communications).

The diffusion of information within the network takes place as follows: packets transmitted with the WIDE1-1 path remain confined within their cell, packets for which a wider diffusion is required are instead repeated in VHF by progressively decreasing the counter of repetitions and retransmitted in UHF in the transit cells, until the counter is reset.

Mobile stations that move from one cell to another and only need to be tracked (without having to be contacted) can forward their signal directly in VHF to the upper hierarchical level; in this way they do not need to frequently retune and have the additional advantage of finding a much less busy VHF channel because it is free from the local traffic managed in UHF within the individual cells.

Once they reach their destination, they can tune into the UHF frequency of the arrival cell and participate in two-way communications.

5.4 IGATES: Widespread listening and broadcasting

As we observed in the introduction, the internet currently plays an important role in routing APRS packets. For strategic reasons, however, the network remains potentially able to operate exclusively via radio by relying on digipeaters.

If you give up potential independence from the web and agree to use exclusively the internet for the collection and routing of the packets, you can easily create a particularly efficient configuration of the APRS network with the available hardware that we could define as "widespread listening and broadcasting".

The reception of the packets transmitted by the APRS stations would be carried out by numerous exclusively receiving IGATE stations scattered throughout the territory, which would create a widespread listening isofrequency network.

The APRS-IS network would act as a collector for all packets received, routing the data to other exclusively transmitting IGATE stations which would broadcast the packets of local interest on a different frequency.

In this configuration isofrequency digipeaters, which as we have seen are the main cause of the degradation of the performance of the APRS network, would not be used at all.

The advantages would be numerous:

- since the digipeating (nor the broadcasting in general) of the signals on the listening frequency does not take place, the listening channel would be much less busy, with considerable benefit for the overall throughput;
- by distributing IGATE receiving stations with a small coverage area in a capillary way, it would be possible to reduce the number of stations listened by each receiving IGATE and therefore the collisions;
- packets colliding for a receiving IGATE could be correctly received by neighboring receiving IGATES (phenomenon of reception diversity).

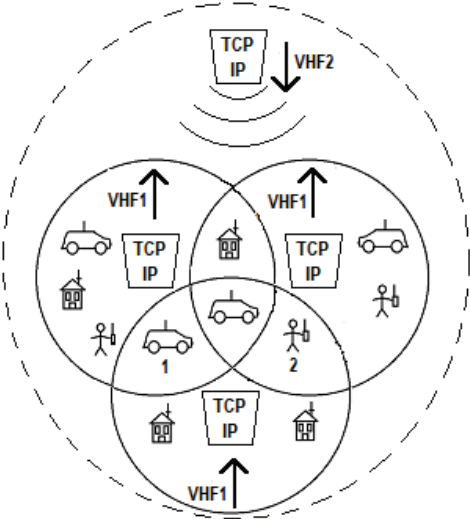


Figure: stations 1 and 2, which transmit simultaneously, collide for the receiver IGATE below but not for the other two receiver IGATES. The transmitting IGATE radiates its signal in the area of the three cells.

The transmitting IGATES, located in privileged geographical positions and different from the receiving IGATES, could operate on a different frequency in the same band as the receiving IGATES.

They would only transmit the traffic of the underlying IGATES receiving cells, plus the packets for which multiple digipeating is required, effectively replacing the digipeaters, with the difference that the transfer of the packets between transmitting cells would take place via the internet and not via radio. To share the transmission channel, neighboring transmitting IGATES would use the CSMA 1-persistent protocol, distant IGATES would simply not listen to each other.

Being able to operate IGATES transmitters and receivers on the same band would allow terminal stations to use the current equipment by simply setting an offset between transmission and reception, without the need to use dual band equipment.

Also in this case terminal stations would accept the small loss of information in the short time interval during which they transmit.

Of course the traffic could also always be viewed on the internet as it happens now.

Even higher performances could be obtained by operating the transmitting IGATES on different frequencies eliminating the contention access problems on the downlink channel: in this way each transmitting IGATE could exploit the capacity of its channel at 100%, being able to transmit up to 900 frames per network cycle of 20 minutes at a speed of 1200 bps with the maximum reliability allowed by the noisy radio channel, corresponding for example to 115 mobile stations that transmit an APRS packet every 5 minutes and 440 fixed stations that transmit an APRS packet every 20 minutes.

Such transmitting IGATES could easily serve very large areas or a large number of APRS stations. Fixed stations should tune into the downlink frequency of their area; mobile stations could simply be tracked while they are in motion and, only when they reach their destination, tune their receivers into the downlink frequency of the arrival location to communicate in a bidirectional way.

Such solutions, certainly feasible in theory, pose however problems from a practical point of view regarding the availability of internet connections in isolated places, and the cost, because they require a very high number of internet connections.

A less expensive solution would be to make bidirectional IGATE cells with an intermediate coverage area, partly sacrificing the benefits in terms of the reduced number of collisions that derive from small listening cells.

These medium-sized cells, located in privileged locations for the coverage of a limited geographical area, but not remote, (aqueduct towers, bell towers, high buildings, any private house with a panoramic view), could easily have internet access, and using the same connection for traffic in both directions would halve the number of connections needed, and therefore the overall cost.

The cells would operate on separate frequencies for reception and transmission as proposed in the previous model but could hardly operate on the same band due to the interference that the transmitter would exert on the receiver at the IGATE.

Terminal stations should therefore use dual band equipment. The interconnection between the cells, of course, would take place via the internet.

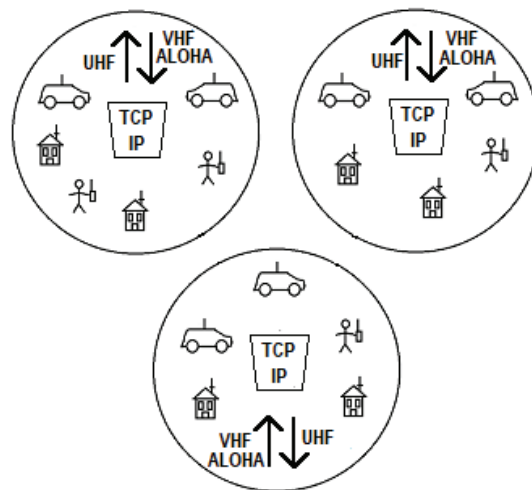


Figure: the network, divided into cells served by bidirectional IGATES and interconnected via the internet. Terminal stations transmit in VHF as soon as they have a packet to send and receive the data stream of their IGATE.

CONCLUSIONS

The analysis of the simple model proposed highlights the limits of the ALOHA protocol: random access to the shared channel heavily affects the maximum throughput, which cannot in any case exceed 18.4% of the total traffic at the maximum load conditions (50% of the channel capacity), and the probability of communication success under maximum load conditions is limited (36.8%).

Communication reliability improves when the traffic is low but the percentage of channel inactivity becomes very high, with a consequent significant waste of bandwidth.

The introduction of a digipeater in the network does not greatly reduce the overall performance in conditions of maximum load (maximum throughput 15.5% at a traffic equal to 42.2% of the channel capacity), and even less when the network is lightly loaded.

Instead, the presence of multiple digipeaters on the same frequency, necessary for multiple digipeating, heavily penalizes the performance of the individual cells and of the network as a whole, limiting the maximum number of stations, the throughput and the reliability of communications: the network can guarantee reliable communications only when the APRS stations are very limited in number.

Multiple repetition of the same packet along different paths increases the reliability of communication at the expense of an increase in network congestion.

To improve performance it is necessary to renounce the constraint of a completely isofrequency network: it is possible to equip the existing digipeaters with an additional uplink port leaving the operation of the current isofrequency network unchanged, but the most efficient solution that only uses radio communications is to fragment the network into cells operating on different frequencies and to articulate the network on two distinct hierarchical levels, local and interconnection between cells.

If you agree to use exclusively the internet for the collection and routing of packets, giving up the digipeaters, you can easily create particularly performing configurations with existing hardware, but the APRS network becomes web dependent and vulnerable in case of collapse of the normal communication channels, failing the primary function of aid to civil protection for which it was initially conceived.

The configurations identified in the previous pages are just some of the possible solutions; in any case, any solution must allow the reuse of existing equipment in order to be accepted by users.

Finally, one might wonder if it still makes sense to deal with an outdated technology such as APRS - and packet radio in general - certainly very limited by current standards; if this question is legitimate as far as performance is concerned, the fact remains that APRS and packet radio are still nowadays a valid tool for experimentation and personal education.

Bibliography

- W. Beech, D.Nielsen, J.Taylor “AX.25 Link Access Protocol for Amateur Packet Radio”, TAPR 1988
- R.Finch, S.Avent, “A duplex packet radio repeater approach to layer one efficiency”, 1987
- H.P. Van Tonder “Channel Throughput Enhancement of an Automatic Position Reporting System Network with MAC Layer Protocol Optimization”
- H.P. Van Tonder , “Improving Automatic Position Reporting System (APRS) Throughput and Reliability”, University of Stellenbosch, Dec 2004
- P. Loveall, “How APRS Works”, 2005 (PPT)
- B. Zielinski, “Effective throughput of AX.25 protocol”, Bulletin of the Polish Academy of sciences, Vol. 61, No. 3, 2013
- Maher H. Heal, “A Comment on the Throughput of Non-persistent CSMA”, Abu Dhabi University
- Paulette Altmaier, “A Short Tutorial on CSMA”, May 1991
- “K.W. Finnegan, “Examining Ambiguities in the Automatic Packet Reporting System”, December 2014
- The APRS Working Group , “Aprs protocol reference v.1.0”, August 2000
- Kenwood corporation, “APRS”, 1999
- Claire Goursaud, Yuqi Mo. , “Random Unslotted Time-Frequency ALOHA: Theory and Application to IoT UNB Networks”, 23rd International Conference on Telecommunications (ICT) , May 2016, Thessaloniki, Greece
- Sandro Petrizzelli, “Appunti di reti di telecomunicazioni”, cap 5 - protocolli di linea (parte II)
- Renato Lo Cigno, “Reti - Livello Collegamento: Data-Link e Medium Access Control”, Università di Trento

Improved Layer 2 Protocol

Nino Carrillo KK4HEJ

IL2P Overview

IL2P is a Layer 2 packet format that incorporates Forward Error Correction (FEC), packet-synchronous scrambling, and efficient encoding of variable-length packets for narrow-band digital data links. IL2P builds on the extensive work done by others in the amateur radio field to improve the quality, speed, and flexibility of packet radio data networks. IL2P is inspired and informed by the FX.25 draft standard, but departs from on-air backwards compatibility with AX.25 in order to implement a more capable standard. Several of the IL2P Design Goals stem directly from recommendations made by the authors of FX.25 in their draft specification document.

Initial implementations of IL2P target compatibility with the standard AX.25 KISS interface to transfer data to and from a local host device. Many popular host applications (like linBPQ and APRS servers) expect TNCs to speak AX.25 KISS. Therefore, the first hardware implementation of IL2P in existence translates AX.25 KISS frames into IL2P for broadcast on-air, and converts them back to AX.25 KISS frames at the receive side to send them to the host.

Cost of custom-made printed circuit boards and fast embedded digital signal processors are significantly lower today than in 2006, when the FX.25 draft standard was published. It now is possible to implement a KISS TNC in low-power embedded firmware that can encode and decode IL2P packets in real time, while listening for legacy AX.25 packets, and performing 1200 baud AFSK or 9600 baud GFSK modulation and demodulation on a datastream. It is the author's hope that these hybrid firmware TNCs, which can offer legacy AX.25 compatibility in parallel with IL2P capabilities at lower cost than traditional hardware TNCs, accelerate the adoption of this improved standard.

Design Goals

- Incorporate forward-error-correction
- Eliminate bit-stuffing
- Streamline the AX.25 header format
- Improve packet detection in absence of DCD and for open-squelch receive
- Produce a bitstream suitable for modulation on various physical layers
- Avoid bit-error-amplifying methods (differential encoding and free-running LFSRs)
- Increase efficiency and simplicity over FX.25

Interface to Physical Layer

IL2P can be applied to various modulation methods including Audio Frequency Shift Keying (AFSK), Gaussian Frequency Shift Keying (GFSK), and any others that support binary symbols. A '1' bit in IL2P

is sent as an AFSK "mark" tone (1200 Hz), while a '0' bit is sent as an AFSK "space" tone (2200Hz). When using 9600 GFSK, a '1' bit is sent as positive FM carrier deviation (appears as a positive voltage pulse on the TNC's TXA line), and a '0' bit is sent as negative FM carrier deviation. Unlike Bell 202 Non-Return-to-Zero Inverted (NRZI) AFSK and G3RUH 9600, IL2P **does not** use differential encoding.

Technical Details

Reed Solomon Forward Error Correction

Reed-Solomon (RS) forward-error-correction is used to detect and correct errors in the header and payload blocks. The IL2P RS encoder processes header and payload data *after* it has been scrambled, to eliminate the error-amplifying characteristics of multiplicative LFSRs. RS codes have maximum block lengths defined by their underlying Galois Field (GF) size. IL2P uses an 8-bit field to match the size of a byte. The Galois Field is defined by reducing polynomial $x^8+x^4+x^3+x^2+1$. The maximum RS block size is 255 bytes, including parity. In order to support packets larger than the RS block size, large packets are segmented by the encoder into nearly-equal sized blocks before RS encoding into a contiguous IL2P packet.

Variable parity lengths of 2, 4, 6, 8, or 16 symbols (bytes) are used depending on the size of the payload block and selected FEC strength. This allows for increased efficiency for short packets, and provides a consistent symbol-error capability independent of packet length. Variable code shortening is used to eliminate block padding, enabled by a payload byte count subfield in the header.

The RS encoder uses zero as its first root.

IL2P does not use a Cyclic Redundancy Check (CRC) or Frame Check Sequence (FCS). Instead, validity of received data is verified through successful decoding of the RS blocks.

Data Scrambling

IL2P employs packet-synchronous multiplicative scrambling to reduce transmit signal occupied bandwidth, ensure sufficient zero crossings for the receive data-clock PLL, and DC-balance the transmit bitstream. The scrambling is carried out by a linear-feedback-shift-register (LFSR), using feedback polynomial x^9+x^4+1 , which is maximal. This polynomial is significantly lower-order than that used in G3RUH 9600 modems. Selection of a lower order ensures the longest runs of continuous 1 or 0 bits will be shorter, which aids receive data-clock stability.

Packet-Synchronized LFSR

The LFSR is reset to initial conditions at the start of every packet. Scrambling begins at the first bit after the Sync Word. The Preamble and Sync Word are not scrambled. During receive, prior to Sync Word detection, the LFSR is not engaged. The LFSR state is unaltered between blocks inside a packet, scrambling or unscrambling continues with the state left at the end of the last block.

Scrambling Inside RS Code Block

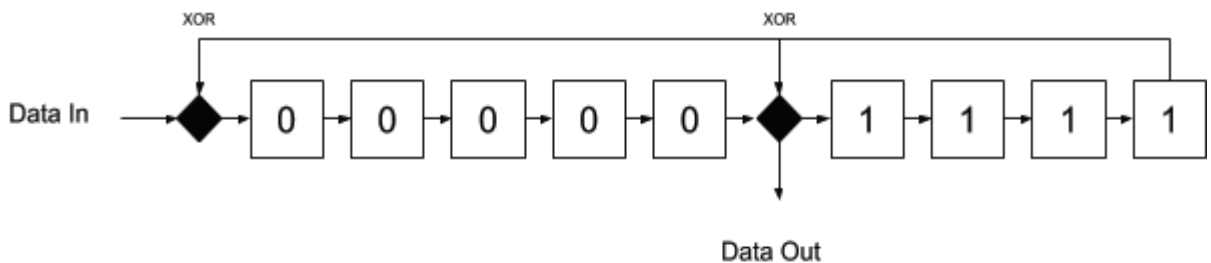
IL2P LFSR encoding is applied inside the RS code block to eliminate the bit-error-amplifying characteristics of LFSR processing. A free-running LFSR (such as in the receive circuitry of the G3RUH modem) propagates bit errors at a multiple of the number of feedback polynomial coefficients (or taps on the LFSR). For example, when a single bit-error passes through a free-running LFSR defined by X^9+X^4+1 (or any other 3-term polynomial), 3 erroneous bits will appear on the output as they are XOR'd through the feedback taps of the shift register. This is of little concern in legacy AX.25 on-air protocols, because even a single bit error anywhere in the packet will cause the packet to be rejected.

RS codes correct errors on a symbol-by-symbol basis (byte-by-byte for IL2P). In order to prevent the LFSR spreading a single bit error from one RS symbol to another, the IL2P packet encoder applies RS encoding *after* the data has been scrambled, and the receiver applies RS decoding *before* the data is unscrambled. This allows bit errors to be corrected by the RS decoder before passing through the receive LFSR. The RS parity symbols themselves are not passed through an LFSR, they are appended to the RS block exactly as computed.

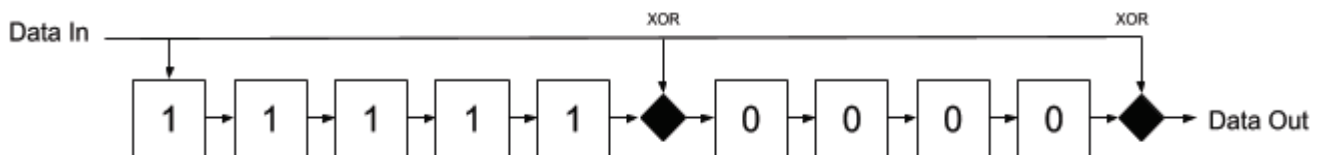
Extracting All Data from LFSR Memory

Efficient LFSR algorithms can be constructed by arranging an LFSR in Galois configuration. Galois configured LFSRs have bit delay, which means it takes some number of bit cycles after a bit of information enters the LFSR for it to appear in its scrambled form on the output. Because of this, the output of the LFSR is taken after its bit delay has elapsed (5 bits in this case), and flushed at the end of the data block to extract all information bits from its memory. The LFSR schematics given below represent Galois configuration of the IL2P scrambling polynomial.

Transmit LFSR Schematic and Initial Conditions



Receive LFSR Schematic and Initial Conditions



Packet Structure

IL2P Packet Format				
Preamble	Sync Word	Control & Addressing	Header Parity	Payload Blocks & Parity
variable	3 bytes	13 bytes	2 bytes	0-1081 bytes

All bytes are sent **Most Significant Bit first**.

Preamble

The IL2P recommended Preamble is variable length, and consists of some number of 0x55 bytes (01010101), which provides the receive data slicer frequent bit transitions to establish a lock on the transmitted data-clock before information appears. When sent back-to-back, the Preamble of subsequent packets is omitted. There is no terminating symbol. All IL2P packets are terminated by byte count, which is stored in the header.

Sync Word

The IL2P Sync Word is 0xF15E48. This 24 bit sequence has an equal number of 1's and 0's and identifies the start of all IL2P Packets. Recommended Sync Word match tolerance at the receiver is 1 bit, meaning the receiver will declare a match if 23 out of the last 24 bits received match the Sync Word (any single bit flipped). This intended to ensure Sync Word detection on noisy links, at the cost of increasing the Sync Word match space up to 25 possible matches out of 2^{24} possible bit sequences. In a 9600 bit/sec application with open squelch and ignoring DCD, the expected average interval time between false matches is about 69 seconds (bit rate * $2^{24} / 25$). False matches are rejected by the receiver after the header fails RS decoding.

FEC Level

A one-bit subfield in the header identifies the amount of FEC parity bytes applied to the packet. A zero value indicates variable FEC up to 8 bytes per block (referred to as Baseline FEC in this document). A one value indicates constant FEC of 16 bytes per block (referred to as Max FEC).

IL2P Header Types

IL2P defines 2 possible header mappings, encoded in a 1-bit header subfield. A zero value indicates transparent encapsulation. A one value indicates translated encapsulation. Both mappings include a 10-bit payload count, enabling packet sizes up to 1023 payload bytes after the header. This count does not include parity bytes attached to the payload.

IL2P Type 0 Header

Type 0 headers are used for transparent encapsulation of data - the entire encapsulated packet appears in the payload of the IL2P packet. Therefore, the header only includes the 10 bit PAYLOAD BYTE COUNT subfield as described in IL2P Type 1 Header. Type 0 encapsulation occurs when a KISS frame is presented to the IL2P encoder that cannot be translated. Some examples of non-translatable KISS frames include MIC-E encoded APRS data (callsign characters can't translate to SIXBIT), Extended mode AX.25 frames (modulo-127 window sizes), and unrecognized AX.25 PID codes. These frames are placed entirely in the IL2P payload, so they still benefit from forward-error-correction.

IL2P Type 1 Header

Type 1 headers contain a compressed and translated AX.25 header. The majority of common AX.25 traffic is compatible with Type 1 translation. The Control and Addressing section of the header contains everything normally found in an AX.25 header, with some modifications. IL2P stores destination and source callsigns using six bits per character in DEC SIXBIT coding (take the ASCII code for a printable character and subtract 0x20). IL2P also compresses the Protocol ID field to 4 bits rather than 8.

Control and Addressing Field Map for IL2P Type 1 Header													
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12
Bit 0													SRC SSID
Bit 1													
Bit 2	DEST	DEST	DEST	DEST	DEST	DEST	SRC	SRC	SRC	SRC	SRC	SRC	
Bit 3	C/S 1	C/S 2	C/S 3	C/S 4	C/S 5	C/S 6	C/S 1	C/S 2	C/S 3	C/S 4	C/S 5	C/S 6	
Bit 4													
Bit 5													
Bit 6	UI	PID				CONTROL							
Bit 7	FEC LEVEL	HDR TYPE	PAYLOAD BYTE COUNT										
Subfields spanning Bit 6 and Bit 7 have MSB on the left. SSID are four-bit subfields. Callsigns are packed in DEC SIXBIT encoding.													

Type 1 Header Control and Addressing Subfields

The Type 1 Header is composed of several fields found in the AX.25 header, though they are translated and compressed into an IL2P format. Type 1 Headers do not support AX.25 repeater callsign addressing, Modulo-127 extended mode window sequence numbers, nor any callsign characters that cannot translate to DEC SIXBIT. If these cases are encountered during IL2P packet encoding, the encoder switches to Type 0 Transparent Encapsulation.

Payload Byte Count Subfield

The Payload Byte Count is stored in the header as a 10-bit subfield (possible values 0-1023). The count represents the total number of data bytes stored in all payload blocks following the header. The count excludes the header, and all parity symbols appended to payload blocks. See the Payload Blocks section of this document for a description of how payload parity symbols are appended to payload blocks.

UI Subfield

AX.25 specifies 3 types of frames: Information, Supervisory, and Unnumbered. Each has different uses for the AX.25 Control field, and only some have a PID field. All AX.25 Information frames have a PID field. AX.25 Supervisory frames do not have a PID field. AX.25 Unnumbered frames do not have a PID field, unless their Control field is set to the Unnumbered Information (UI) opcode. The IL2P Type 1 Header UI subfield is 1 bit and is set only for AX.25 Unnumbered Information frames to signal that the PID field exists for a U-Frame.

PID Subfield

In Type 1 header mapping, IL2P maps the AX.25 8-bit PID field into a 4-bit IL2P subfield. The IL2P PID subfield is also used to identify the AX.25 frame type, which informs the encoding and decoding of the IL2P Control subfield.

IL2P AX.25 PID Code Mapping		
IL2P PID	Translation	AX.25 PID
0x0	AX.25 Supervisory Frame (No PID byte)	Omit PID
0x1	AX.25 Unnumbered Frame (No PID byte, except UI)	Omit PID
0x2	AX.25 Layer 3	yy10yyyy or yy01yyyy
0x3	ISO 8208 / CCIT X.25 PLP	0x01
0x4	Compressed TCP/IP	0x06
0x5	Uncompressed TCP/IP	0x07
0x6	Segmentation fragment	0x08
0x7	Future	
0x8	Future	
0x9	Future	
0xA	Future	
0xB	ARPA Internet Protocol	0xCC
0xC	ARPA Address Resolution	0xCD
0xD	FlexNet	0xCE
0xE	TheNET	0xCF
0xF	No L3	0xF0

Control Subfield

The Control Subfield contains 7 bits, and its mapping depends on the translated AX.25 frame type.

Translated AX.25 I-Frame Control Subfield

All AX.25 I-Frames are considered commands. Therefore, IL2P omits the Command (C) bit for translated I-Frames. This subfield contains a Poll/Final (P/F) bit, receive sequence N(R), transmit sequence N(S).

Translated AX.25 I-Frame Control Subfield Map						
Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P/F	N(R)			N(S)		

Translated AX.25 S -Frame Control Subfield

AX.25 S-Frames can be one of 4 opcodes. All include a receive sequence number N(R), and a C bit.

Translated AX.25 S-Frame Control Subfield Map						
	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	N(R)			C	OPCODE	
RR Receive Ready	N(R)			C	0	0
RNR Receive Not Ready	N(R)			C	0	1
REJ Reject	N(R)			C	1	0
SREJ Selective Reject	N(R)			C	1	1

Translated AX.25 U-Frame Control Subfield

AX.25 U-Frames contain an opcode, P/F bit, and C bit. Certain opcodes are always commands or responses, some can be either. There are no sequence numbers in U-Frames.

Translated AX.25 U-Frame Control Subfield Map								
		Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		P/F	OPCODE			C		
command	SABME set async balanced mode extended	Not supported, send as Transparent						
command	SABM set async balanced mode	P	0	0	0	1		
command	DISC disconnect	P	0	0	1	1		
response	DM disconnect mode	F	0	1	0	0		
response	UA unnumbered acknowledge	F	0	1	1	0		
response	FRMR frame reject	F	1	0	0	0		
either	UI unnumbered information	P/F	1	0	1	C/R		
either	XID exchange identification	P/F	1	1	0	C/R		
either	TEST	P/F	1	1	1	C/R		

Payload Blocks

Each payload block forms a contiguous RS code block once parity is added. RS codes can correct a number of erroneous symbols in a code block equal to half the number of parity symbols. So a code block with 2 parity symbols can recover one erroneous symbol anywhere in the code block.

Baseline FEC block lengths and parity counts in IL2P are designed to provide roughly 1.5% symbol-error-rate recovery in the payload blocks. The number of parity symbols added to each block

varies based on the size of the block. To achieve that, the following procedure is conducted by the transmitter to calculate the number of payload blocks and parity symbols required to compose the packet:

Baseline FEC Payload Block Size Computations

```

payload_block_count = Ceiling(payload_byte_count / 247)
small_block_size = Floor(payload_byte_count / payload_block_count)
large_block_size = small_block_size + 1
large_block_count = payload_byte_count - (payload_block_count * small_block_size)
small_block_count = payload_block_count - large_block_count
    
```

Large blocks are 1 byte bigger than small blocks. Not every packet requires large blocks, they exist to carry remainder bytes. If small_block_size divides evenly into payload_byte_count, then the packet can be encoded without large blocks. Large blocks, if they exist, are always placed closest to the header when the packet is assembled.

Worked examples:

IL2P Baseline FEC Payload Block Count Examples				
Payload Byte Count	100	236	512	1023
Small Block Size	100	236	170	204
Large Block Size	101	237	171	205
Large Block Count	0	0	2	3
Small Block Count	1	1	1	2

Baseline FEC Parity Symbol Count Computation

The number of parity symbols appended to each payload block is driven by small_block_size.

```
parity_symbols_per_block = (small_block_size / 32) + 2
```

The encoder will append 2, 4, 6, or 8 parity symbols per payload block. The maximum small_block_size for each parity symbol count is given below.

Maximum small_block_size	
Parity Symbols per Block	Maximum small_block_size
2	61
4	123
6	185
8	247

Max FEC Payload Block Size Computations

Under the Max FEC scheme, the encoder will always append 16 parity symbols per payload block, regardless of block size. This provides a minimum of roughly 3% symbol-error-rate recovery in the payload blocks. Shorter packets benefit from higher error recovery capacity.

```
payload_block_count = Ceiling(payload_byte_count / 239)
small_block_size = Floor(payload_byte_count / payload_block_count)
large_block_size = small_block_size + 1
large_block_count = payload_byte_count - (payload_block_count * small_block_size)
small_block_count = payload_block_count - large_block_count
parity_symbols_per_block = 16
```

IL2P Transmit Encoding Procedure for AX.25 KISS Data

1. Place Sync Word in the first three bytes of output buffer
2. Extract all AX.25 header fields
3. Check AX.25 header fields for compatibility with Type 01 Header

If AX.25 Fields Type 1 Compatible

4. Compose IL2P Control & Addressing Field and place in output buffer
5. Initialize LFSR to initial conditions
6. Scramble the output buffer starting at the Control & Addressing Field
7. RS Encode output buffer starting at the Control & Addressing Field
8. Count payload bytes in AX.25 input data and perform Payload Block Size computations
9. Perform Parity Symbol Count computation
10. Scramble then RS encode each payload block (large blocks closest to header)
11. Send output buffer data to transmitter (AFSK or GFSK modulator)

If AX.25 Fields Not Type 1 Compatible Send As Type 0

4. Count all bytes in AX.25 input data and perform Payload Block Size computations
5. Perform Parity Symbol Count computation
6. Place PAYLOAD BYTE COUNT subfield in Control & Addressing Field (all other fields 0)
7. Scramble the output buffer starting at the Control & Addressing Field
8. RS Encode output buffer starting at the Control & Addressing Field
9. Scramble then RS encode each payload block (large blocks closest to header)
10. Send output buffer data to transmitter (AFSK or GFSK modulator)

IL2P Receive Decoding Procedure for KISS AX.25 Data

1. Search receive bitstream for Sync Word match

On Sync Word Match Within 1 Bit Tolerance

2. Collect next 15 bytes as IL2P Header
3. RS Decode IL2P Header

If RS Decode Successful

4. Initialize LFSR to initial conditions
5. Unscramble 13 byte Control & Addressing Field
6. Extract IL2P Control & Addressing Field and translate to AX.25 header in KISS buffer
7. Perform Payload Block Size computations on PAYLOAD BYTE COUNT
8. Perform Parity Symbol Count computation
9. Collect payload blocks from receive bitstream according to results of Step 7 and 8
10. RS decode and then unscramble each payload block
11. Place unscrambled data in KISS buffer and send to host
12. Return to Step 1

If RS Decode of Header or Any Payload Block Unsuccessful

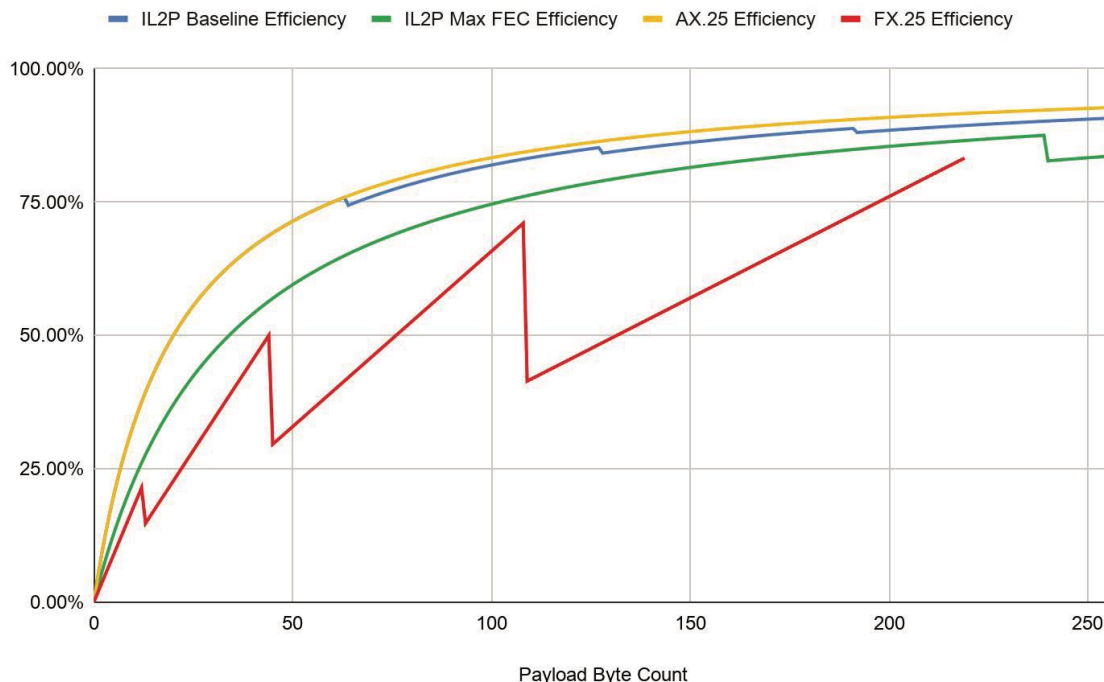
13. Discard packet
14. Return to Step 1

Comparative Protocol Efficiency Analysis

Protocol Efficiency in the graph below shows the percentage of payload bytes that make up the packet, excluding Preamble. The IL2P Header and Sync Word consume 18 bytes, so efficiency generally increases as packet size grows. The sawtooth bumps in the graph represent Payload Byte Counts where an additional code block is required to contain the payload.

For comparison, the efficiency of AX.25 and FX.25 (255,239) protocols are included on the graph. The FX.25 line is computed using the smallest block size compatible with the payload size. The costs of bit-stuffing incurred under AX.25 and FX.25 are ignored.

Protocol Efficiency vs Payload Byte Count



Example Encoded Packets

These examples are intended for use as verification samples to help individuals implementing their own IL2P encoders and decoders. Note that all AX.25 data samples below lack opening and closing flags, and are not bit-stuffed. All IL2P data samples below lack Sync Word.

AX.25 S-Frame

This frame sample only includes a 15 byte header, without PID field.

Destination Callsign: KA2DEW-2

Source Callsign: KK4HEJ-7

N(R): 5

P/F: 1

C: 1

Control Opcode: 00 (Receive Ready)

AX.25 data:

```
96 82 64 88 8a ae e4 96 96 68 90 8a 94 6f b1
```

IL2P Data Prior to Scrambling and RS Encoding:

```
2b a1 12 24 25 77 6b 2b 54 68 25 2a 27
```

IL2P Data After Scrambling and RS Encoding:

```
26 57 4d 57 f1 96 cc 85 42 e7 24 f7 2e 8a 97
```

AX.25 U-Frame

This is an AX.25 Unnumbered Information frame, such as APRS.

Destination Callsign: CQ -0

Source Callsign: KK4HEJ-15

P/F: 0

C: 0

Control Opcode: 3 Unnumbered Information

PID: 0xF0 No L3

AX.25 Data:

86 a2 40 40 40 40 60 96 96 68 90 8a 94 7f 03 f0

IL2P Data Prior to Scrambling and RS Encoding:

63 f1 40 40 40 00 6b 2b 54 28 25 2a 0f

IL2P Data After Scrambling and RS Encoding:

6a ea 9c c2 01 11 fc 14 1f da 6e f2 53 91 bd

AX.25 I-Frame

This is an AX.25 I-Frame with 9 bytes of information after the 16 byte header.

Destination Callsign: KA2DEW-2

Source Callsign: KK4HEJ-2

P/F: 1

C: 1

N(R): 5

N(S) 4

AX.25 PID: 0xCF TheNET

IL2P Payload Byte Count: 9

AX.25 Data:

96 82 64 88 8a ae e4 96 96 68 90 8a 94 65 b8 cf 30 31 32 33 34 35 36 37 38

IL2P Scrambled and Encoded Data:

26 13 6d 02 8c fe fb e8 aa 94 2d 6a 34 43 35 3c 69 9f 0c 75 5a 38 a1 7f f3 fc

References for Further Study

General background on Polynomial Codes, Error Detection, and Error Correction:

Widjaja, Indra and Leon-Garcia, Alberto. *Communication Networks*. New York: McGraw-Hill 2004 166-190. Print.

A good primer on Reed Solomon codes from the BBC:

<https://downloads.bbc.co.uk/rd/pubs/whp/whp-pdf-files/WHP031.pdf>

James Miller's G3RUH 9600 Modem:

<https://www.amsat.org/amsat/articles/g3ruh/109.html>

Another 9600 modem implementation by John Magliacane KD2BD:

<https://www.amsat.org/amsat/articles/kd2bd/9k6modem/9k6modem.html>

The AX.25 2.2 specification:

<http://www.ax25.net/AX25.2.2-Jul%2098-2.pdf>

The FX.25 draft specification:

http://www.stensat.org/docs/FX-25_01_06.pdf

Wikipedia DEC SIXBIT encoding:

https://en.wikipedia.org/wiki/Six-bit_character_code#DEC_six-bit_code

Wikipedia Linear Feedback Shift Registers:

https://en.wikipedia.org/wiki/Linear-feedback_shift_register

KISS Protocol

www.ax25.net/kiss.aspx

This document was written by Nino Carrillo, reachable at nino.carrillo@outlook.com.

Changes:

26 Jan 2020 v0.3: Updated dead link to AX25 specification.

1 Aug 2020 v0.4: Added Max FEC scheme (16 parity bytes per block), updated protocol efficiency graph.

HF Propagation Measurement Techniques and Analyses

Steve Cerwin WA5FRF

This paper presents methods and equipment for performing HF skywave propagation measurements using WWV standard time and frequency stations and selected results. Topics include single frequency and spectral analysis methods for measurement of ionospherically induced Doppler shifts, comparison of WWV frequency data taken simultaneously on multiple frequencies, observation of skywave mode splitting during the dawn and dusk day/night transitions, the use of precise WWV timing tick measurements to identify single and multipropagation modes, and techniques for separating WWV and WWVH during simultaneous reception. Instrumentation methods, data records, and propagation insights are given.

- I. Introduction
- II. Ionospherically Induced Frequency Variations
 - A. Instrumentation for Frequency Measurements
 - B. Measured Spectrograms
 1. Typical Diurnal Behavior at 5 MHz
 2. Simultaneous 2.5, 5, and 10 MHz records
 3. Mode Splitting During Sunrise and Sunset
 4. Diffuse Frequency Scatter and AM Sidebands
 5. The Ionosphere as an AM and FM Modulator
 6. Spectrogram vs. Single Frequency Measurement
- III. Precise Timing Measurements on WWV Per-second Ticks to Infer Mode
 - A. Possible Mode Splitting Mechanisms
 - B. Mode Order Determination through Time-of-Flight Measurement
 - C. Timing Tick Format
 - D. Timing Reference and Receiver Calibration
 - E. Estimated Arrival Times, Superposition of Multiple Modes, and Pulse Broadening
 - F. Measured Data over a Complete Morning Transition
 - G. Correlation with Ray Trace Predictions
 - H. Requirement for Automated Data Acquisition and Analysis
- IV. Separation of WWV and WWVH During Simultaneous Reception
 - A. Spectral Overlap from Simultaneous WWV and WWVH Reception
 - B. 500 and 600 Hz Tone Interlacing
 - C. Deinterlace Methods
 1. Interlaced Data using Only One Tone
 2. WWV or WWVH -only Data Obtained by Muting the Unwanted Tone on Alternate Minutes
 3. Fully Deinterlaced Processing Placing 100% Continuous Data in Separate Data Records
- V. Conclusion and Acknowledgements

I. Introduction

The author participated in the HamSCI effort to measure the impact of the 2017 North American total eclipse on LF and HF propagation by recording amplitude data on both 5 MHz WWV and 60 kHz WWVB. Subsequently, as a calibration procedure for an ARRL Frequency Measuring Test the author recorded detailed on-air WWV spectrograms.¹ In the process of monitoring the apparent frequency of 5 MHz WWV over a CO to TX path several interesting frequency effects were observed. Over several days of observation the general frequency behavior was observed to be turbulent at night and steady during the day. The transition from night to day at dawn was marked by a comparatively large positive frequency swing while the transition from day to night at dusk showed a complimentary negative swing. Additionally, the frequency track at sunrise was observed to diverge into multiple higher order tracks that progressed geometrically upwards in frequency. On some occasions the higher order modes were continuous throughout the night-day transition but on others some higher order modes would manifest abruptly midway through the transition. The evening negative swing occasionally broke into multiple tracks but generally was more docile than the dawn behavior.

This paper describes measurement techniques and experiments designed to learn more about these phenomena. Experimental procedures, data records, and possible inferences to ionospheric physics are given in the following sections.

II. Ionospherically Induced Frequency Variations

A. Instrumentation for Frequency Measurements

The ionospherically induced frequency shifts are very small compared to the HF carrier frequency. Nighttime turbulence may be only a few tenths of a Hz in amplitude and the peak swings at dawn and dusk can reach 1-3 Hz. A typical communications receiver with digital frequency readout has neither the display resolution nor the means to determine the precise frequency of a received carrier with the required accuracy. So the usual procedure is to place the receiver in Upper Side Band (USB) mode and tune it low in frequency so that the difference between the receiver and actual frequency comes out as a tone in the audio output. In essence, the process frequency-translates the carrier frequency from RF to audio. The translation preserves minute frequency variations and places them about a drastically reduced center frequency. This allows analysis on a computer based spectrum analyzer using a sound card. A common setup uses 1000 Hz as the tone frequency. The receiver is set for 4.999 MHz in USB mode for 5 MHz WWV. The spectrum analyzer program is set up with a 1000 Hz center frequency with a 10 Hz span. This allows resolution down to about 10 milliHertz (mHz).

Not all receivers are suitable for high precision frequency measurements because they do not possess the required long and short term frequency stability. The frequency specifications for accuracy and stability for many receivers is on the order of 5-10 HZ, which is fine for all normal

¹ Ionospheric Disturbances at Dawn, Dusk, and During the 2017 Eclipse. Steve Cerwin WA5FRF. QEX, Sept/Oct 2018

amateur radio modes but may not be good enough if the receiver does not settle down after a lengthy warm up. One way to tell if a receiver is usable is to watch the mid-day propagation of 5 MHz WWV since it usually shows flat frequency characteristics. Even good receivers require lengthy warm up times (> 6 hours) in a temperature stabilized room. But even then day-long records have dubious frequency track accuracy if precision to 0.1 Hz or better is desired. A far better approach for receivers that will accept an external reference is to stabilize the receiver with a GPS Disciplined Oscillator (GPSDO). These devices have become readily and inexpensively available. They put out a 10 MHz reference clock and a 1 pps reference with accuracy and stability that are comparable to WWV itself. The author's Icom IC-7610 and R8600 were stabilized in this way.

Another measurement issue is ensuring a high fidelity audio connection between the receiver and the computer sound card. Simply connecting the audio output of the radio to the audio in of the computer can lead to serious ground loop problems. Induced currents from the AC mains and computer switching power supplies can distort and even obliterate the desired signal. The solution is to break the connection with transformer coupling. All of the data presented here used a 600:600 ohm isolation transformer with impedance matching and level control circuitry. The primary side connected to the radio used a 560 ohm series resistor to set a nominal 600 ohm input impedance to the transformer. The secondary was connected across a 1k ohm pot with the wiper ported to the sound card input to provide a separate level control. Finally care must be taken not to overdrive the audio circuitry or harmonic and intermodulation distortion will result.

The spectral waterfall plots were captured using Spectrum Lab, a versatile spectrum analysis program available on the internet. The DSP parameters were selected for the task at hand and varied according to desired record length, update rate, spectral resolution, and desired analog dynamic range. Input levels to the sound card were set using the receiver volume control and the inline isolation circuit. Recording levels in Spectrum Lab were usually set up for a -10 to -60 dBfs range, for a 50 dB display range. A Hamming window function was used with a 75% overlap. Figure 1 shows the DSP parameters used for Fast, Moderate, and Slow data acquisition times. Columns 2-4 are user inputs entered into the program and the remaining columns show the resulting data acquisition parameters.

Acquisition Time	Sample rate	Decimate Divisor	FFT length	Sample Rate-kSPS	FFT Bin mHz	Noise BW mHz	Max Freq kHz	FFT Window Time	Update Rate-sec	Waterfall Increment
Fast	44100	8	65536	5.51222	84.1099	114.389	2.75611	11.89 sec	2.9725	5 min
Fast	44100	16	65536	2.75611	42.0549	57.1947	1.37806	23.78 sec	5.945	5 min
Fast	44100	8	131072	5.51222	42.0549	57.1947	2.75611	23.78 sec	5.945	5 min
Moderate	44100	12	131072	3.67482	28.0366	38.1298	1.83741	35.67 sec	8.9175	15 min
Moderate	44100	16	131072	2.75611	21.0275	28.5974	1.37806	47.56 sec	11.89	15 min
Slow	44100	16	262144	2.75611	10.5137	14.2987	1.37806	1.585 min	23.775	30 min
Slow	11025	4	524288	2.75625	5.25713	7.1497	1.37813	3.17 min	47.55	1 hr.

Figure 1 Spectrum Lab DSP Parameters Used in the Study

B. Measured Spectrograms

1. Typical Diurnal Behavior at 5 MHz

Figure 2 shows typical behavior of 5 MHz WWV over periods of time that include night, day, and the morning and evening transition periods. This data was taken with a thoroughly warmed up and stable receiver but without a GPSDO, so only relative and not absolute frequency is calibrated. The spectrogram is a waterfall display with oldest data at the bottom and newest at the top. The scale factor is 1 Hz/div horizontal and 1 hr./div vertical. At night the carrier frequency shows considerable turbulence with peak to peak values reaching 0.5 Hz. During the day, the turbulence disappears and the frequency track virtually straight-lines. The negative frequency swing as day transitions to night is clearly evident in the top half of the figure. This transition is relatively docile compared to wild frequency swings at dawn shown on the bottom half. The sunrise transition shows an intermittent mode with virtually no frequency shift, a continuous higher order frequency shift, and two additional higher order modes that manifest abruptly midway through the transition.

2. Simultaneous 2.5, 5, and 10 MHz records

Additional data were taken shortly after the HamSCI Festival of Frequency Measurements commemorating the WWV Centennial. Figures 3 through 5 show simultaneous recordings of 2.5, 5, and 10 MHz WWV. The data spans a time from 2100z on Oct. 10, 2019 through 1800z on Oct. 13, 2019. The receiver for 10 MHz was marginal on long term stability so some of the gradual drift shown in the spectrogram may have been receiver drift. As before horizontal scale is 1 Hz/div and vertical is 1 hr./div. A number of features correlate across frequency while others do not. All traces show positive frequency swings at dawn and negative swings at dusk, though to varying degrees and with varying characteristics. These Doppler shifts result from the velocity of path length changes as ionization layers descend at dawn and ascend at dusk. One interesting difference is that both 2.5 and 5 MHz show turbulent frequency behavior only at night whereas 10 MHz shows turbulence both day and night. This may have interesting implications for D layer absorption, reach into the upper ionosphere, and where the turbulence actually takes place. Another difference is that both 2.5 and 5 MHz show mode splitting at dawn whereas 10 MHz does not. This may have implications for multiple hop propagation and the inability of 10 MHz to support the high angle of propagation required for a multiple hop mode. The 10 MHz spectrogram also showed an interesting parallel frequency track from 0100z to 0900z on Oct. 13, 2019. The parallel track was displaced high in frequency by 1.5-2 Hz.

The fact that the nighttime frequency turbulence does not scale with operating frequency is interesting. Pure Doppler shifts or wave speed accelerations should scale with carrier frequency. So if all three frequencies were subjected to the same Doppler velocity the frequency shift at 5 MHz should be twice that at 2.5 MHz, and the shift at 10 MHz twice that. Yet the magnitude of the turbulence was about the same for all three frequencies. It appears that unless there is an offsetting effect that diminishes interaction with increasing frequency the cause for the turbulence may be something else.

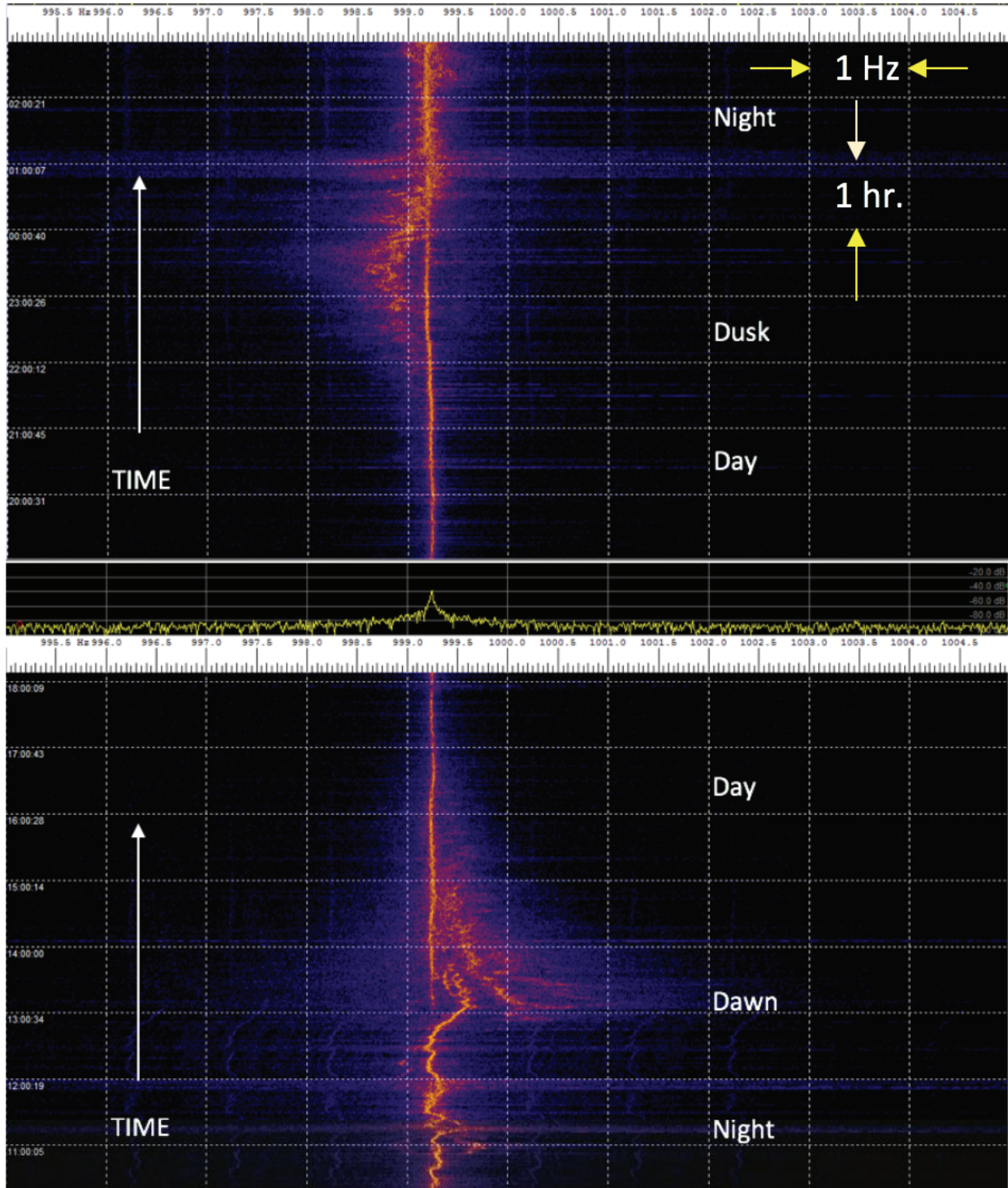


Figure 2 Typical 5 MHz WWV Spectrogram Showing Night, Day, and Transition Behavior

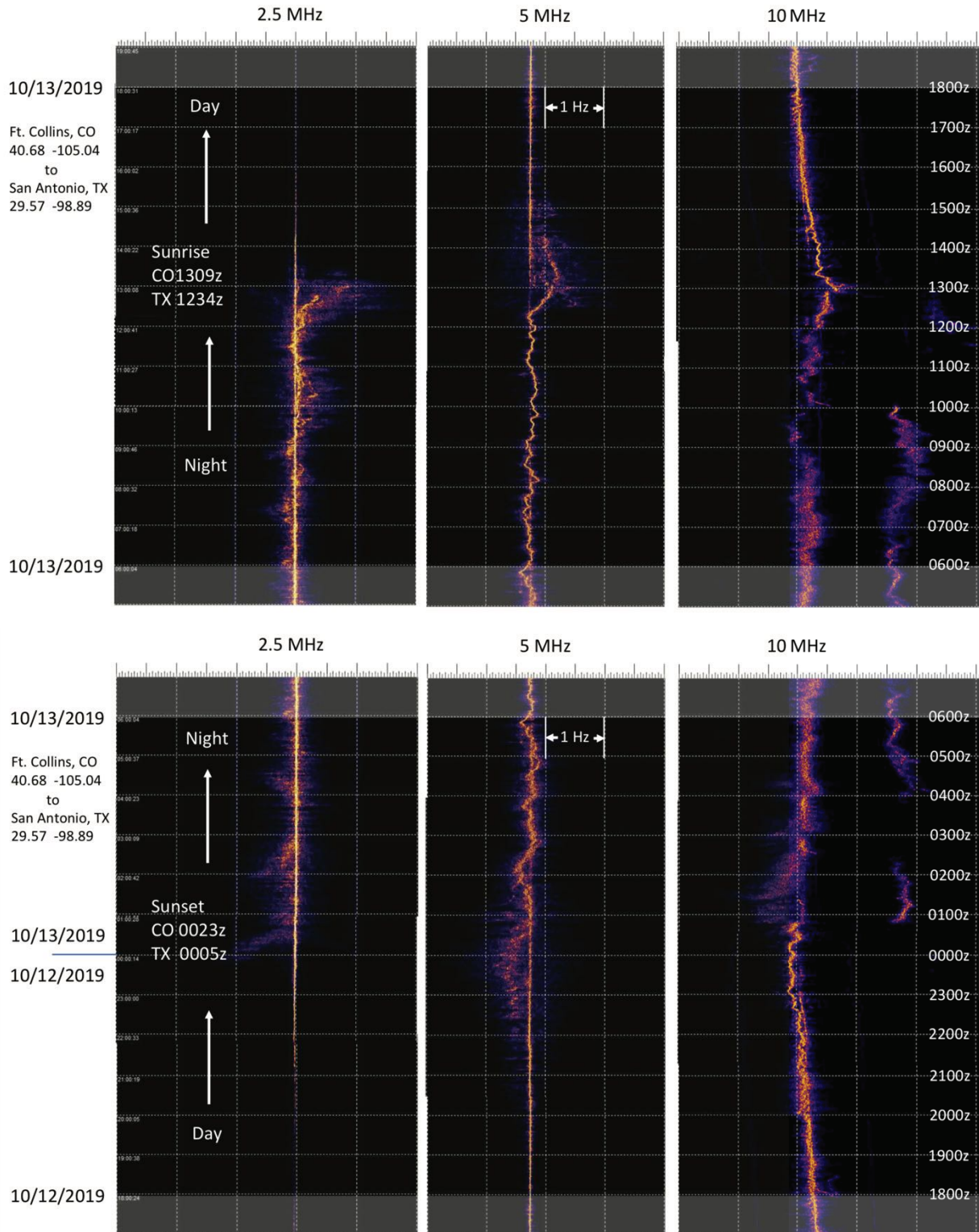


Figure 3 2.5, 5, and 10 MHz WWV for 18z-18z October 12-13, 2019

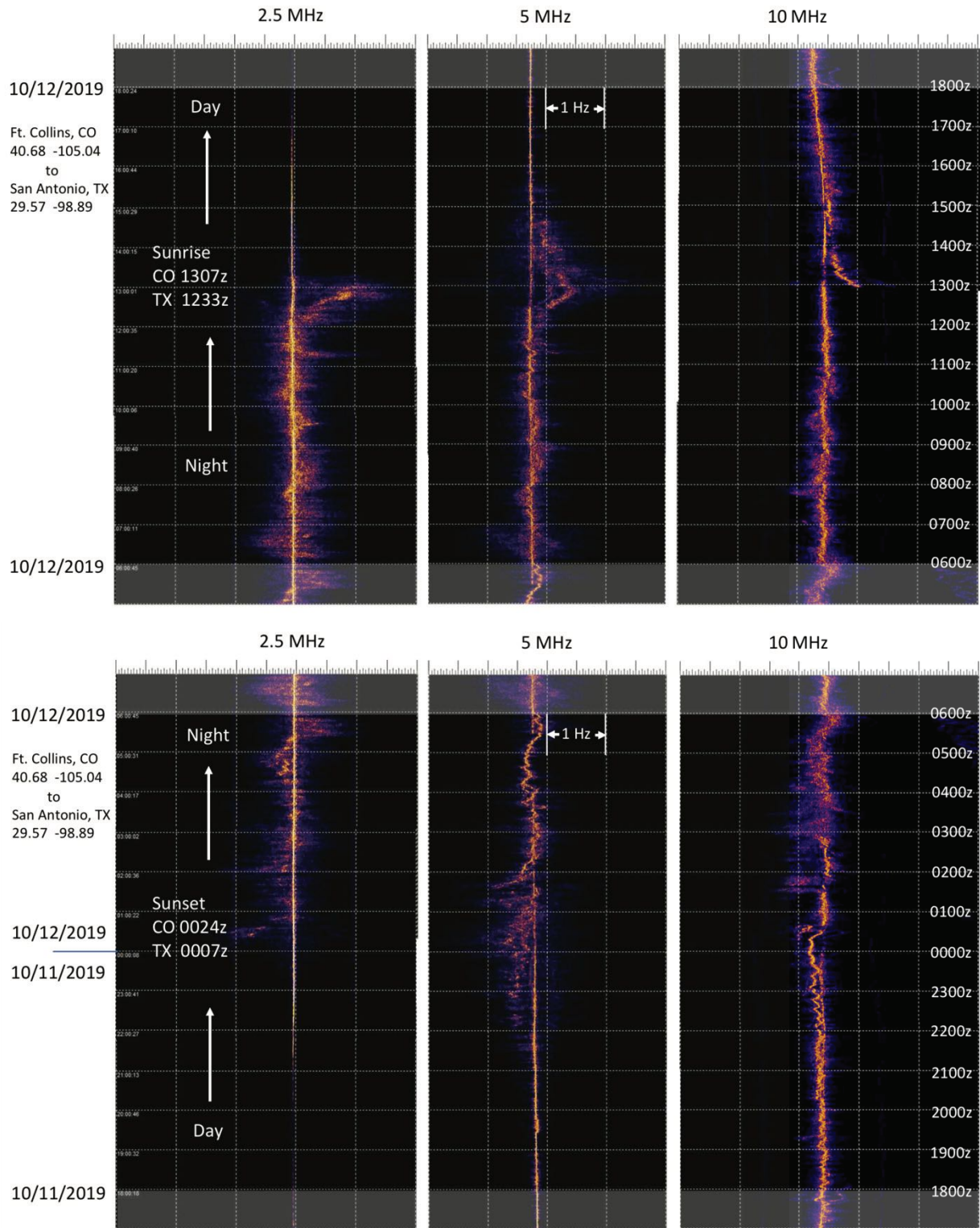
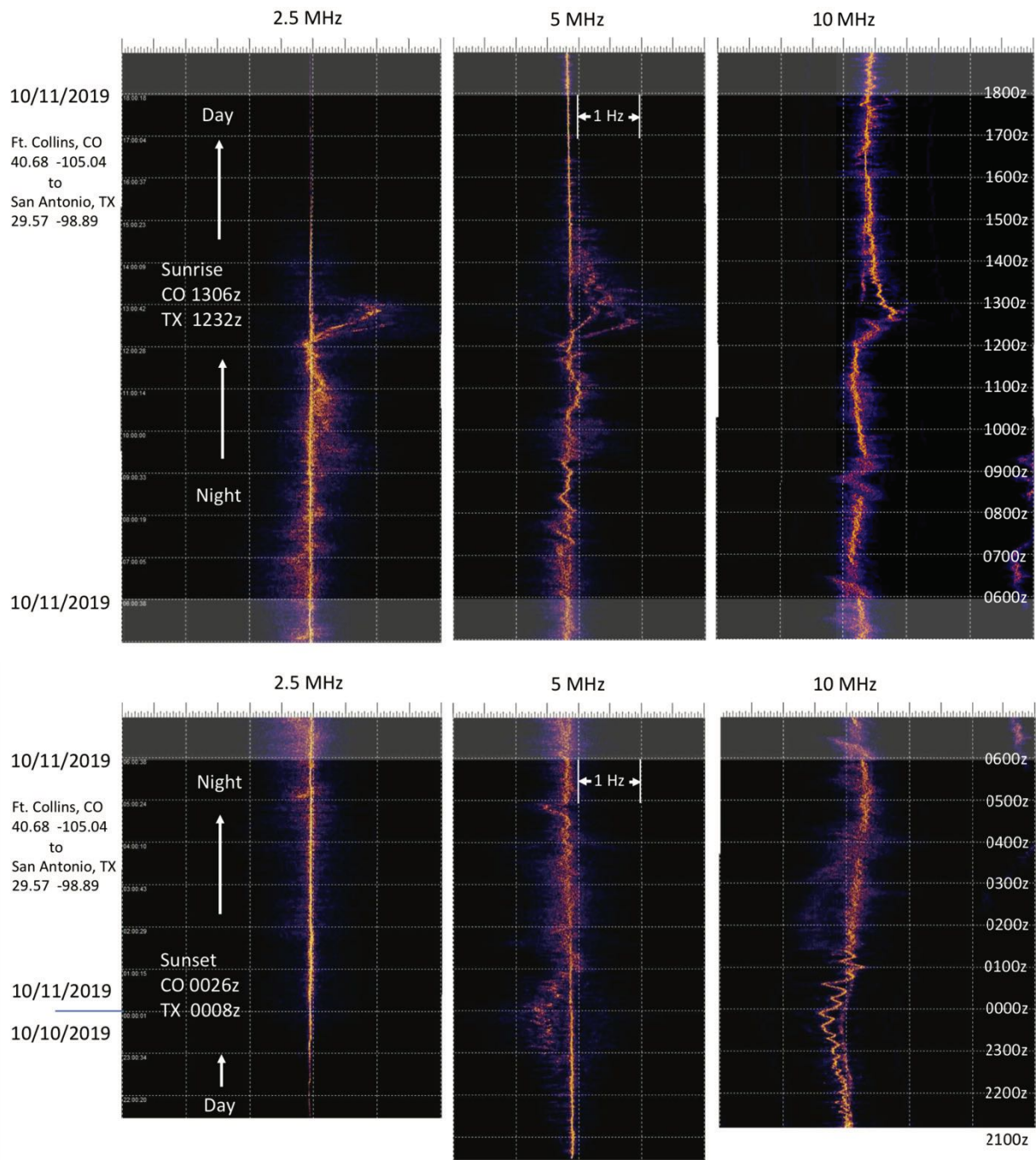


Figure 4 2.5, 5, and 10 MHz WWV for 18z-18z October 11-12, 2019



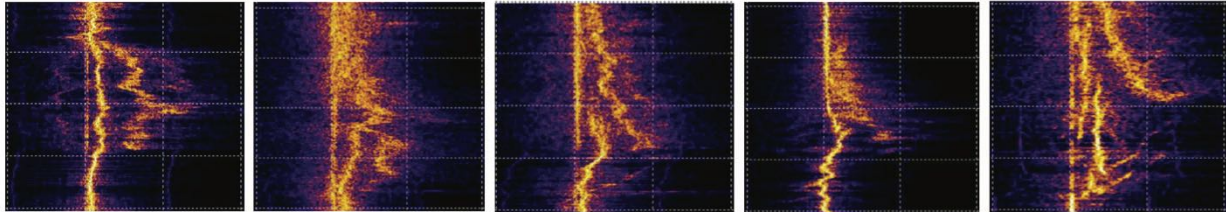
Beginning of Data Records

Figure 5 2.5, 5, and 10 MHz WWV for 18z-18z October 10-11, 2019

3. Mode Splitting During Sunrise and Sunset

While there are many interesting nuances and random events with propagation during the middle parts of night and day, the frequency swing and mode splitting behavior that occurs regularly during the dawn transition and occasionally at dusk can provide exceptional insight to the dynamic response of the ionosphere to changing solar illumination. Figure 6 shows additional records of the positive and negative frequency excursions that occur during sunrise and sundown. Horizontal scale is 1 Hz/div in all cases but the vertical time scale may vary.

Positive Frequency Excursions During Sunrise



Negative Frequency Excursions During Sundown

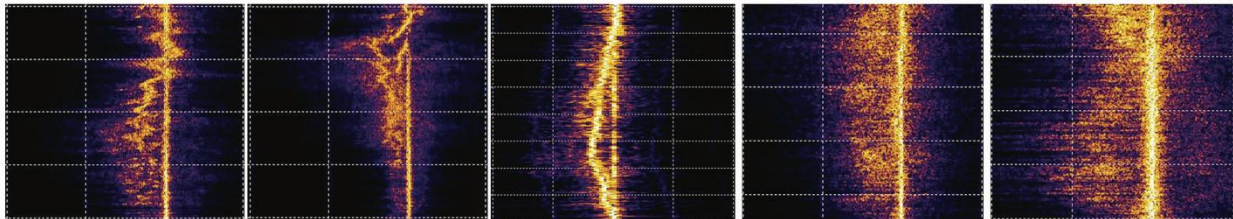


Figure 6 Positive and Negative Frequency Excursions on a WWV-WA5FRF Path During Sunrise and Sundown

In almost all cases there is a constant frequency track that shows little or no frequency displacement, though it can often disappear momentarily while additional higher order modes run amok. During sunrise there can be one to as many as four additional higher order modes. Some modes are present throughout the entire night-day transition showing a half-sine wave shape. Other modes are not present at the beginning of the transition but manifest abruptly part way through. Note that in many cases all of these modes are present at the same time, indicating simultaneous multipath propagation. The negative excursions at sundown show similar behavior but are less radical, probably because of moderation by recombination. Most of the time the evening transition shows only a single extra frequency excursion though occasionally multiple modes manifest.

4. Diffuse Frequency Scatter and AM Sidebands

Many times the modes will print a well defined track whose frequency can be measured as a function of time. But other times the carrier shows a great deal of temporal jitter that can be observed on an oscilloscope. During these times the frequency tracks can take on a fuzzy appearance in a noise band covering a broad frequency range. The two evening captures in the

lower right of Figure 6 are examples. Figure 7 shows an evening capture with an extremely diffuse spectrogram. The pattern is offset low in frequency like the usual evening negative swing but also shows a great deal of diffuse symmetry disposed about it. A non-shifted frequency track continues along the 1000 Hz line throughout the entire track indicating the presence of a non-shifted mode. A GPSDO was used to calibrate the receiver in this recording.

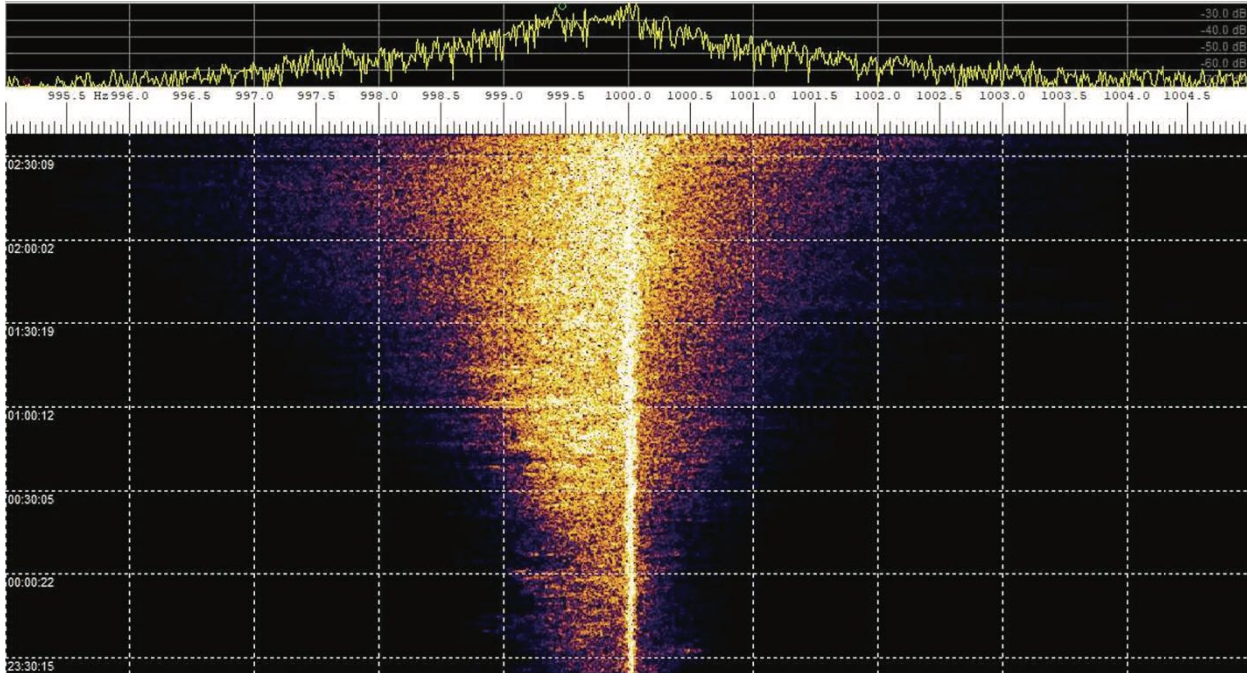


Figure 7 Diffuse Spectrogram Taken During Evening Transition

5. The Ionosphere as an AM and FM Modulator

Insights into ionospheric mechanisms can be gleaned by considering it as an AM and FM modulator. Comparison of the 1 kHz downconverted WWV carrier with a GPSDO stabilized 1 kHz tone from a function generator on a two-channel oscilloscope shows the frequency slewing at dawn and dusk, phase reversals, long term fading and often times rapid amplitude flutter. The combination of rapid amplitude flutter and the symmetry observed in the diffuse spectrograms suggested that at least some of this spectra may actually be AM sidebands resulting from the amplitude fluctuations.

The receiver's automatic gain control AGC is handy for maintaining constant average signal over the course of a long experiment. But it also suppresses amplitude effects if they are what is being studied. Worse, the strong per-second time ticks can pump the AGC, causing amplitude and spectral artifacts. For best data fidelity the receiver AGC should be disabled and signal level set with careful adjustment of the RF gain control.

6. Spectrogram vs. Single Frequency Measurements

There are two basic methods for measuring and displaying frequency data: single frequency and spectral waterfall, or spectrogram. Two popular computer programs within the amateur community for acquiring this type of data are FLDIGI and Spectrum Lab. A single frequency measurement system like FLDIGI is basically an FM demodulator that outputs a serial string of single numbers for the measured value at each time increment. Operational characteristics of the FM demodulator depends on how it is constructed. Characterization measurements on FLDIGI version 4.1.12 showed it to exhibit peak detection and capture effects, similar to the discriminators used in analog FM radios. Within each sample interval the program captures and outputs the frequency of the strongest signal in the passband.

The author built an FM demodulator based on a period counting variation of frequency counting methodology. It had the ability to acquire frequency data at a 100 SPS rate and also had a unique characteristic that allowed identification of the stronger of two multipath modes that were slightly different in frequency. It was useful in studying rapid selective fading when two slightly different frequency modes underwent time dependent interference. As an interference null approached a minimum when the two signals were 180-degrees out of phase the detector exhibited a momentary output swing in frequency. The direction of the swing revealed which frequency had the stronger signal. This behavior was verified in lab experiments using two sinewave generators with precisely controllable frequency and amplitude.

But either FM demodulator could output only a single frequency value per time slot: the value of the strongest signal present in the case of FLDIGI or the average of all signals present weighted by amplitude in the case of the period counter. On the other hand, a spectral analysis program like Spectrum Lab outputs an FFT at each time increment and displays them in waterfall format. It has the distinct advantage of identifying all frequency components present in each sample interval.

Figure 8 shows simultaneous records of 5 MHz WWV taken with all three detection methods during a sunrise transition. All three data representations are formatted to have the same vertical and horizontal scale factors. A comparison shows the obvious advantage of a spectrograph over a single frequency demodulator when mode splitting is present.

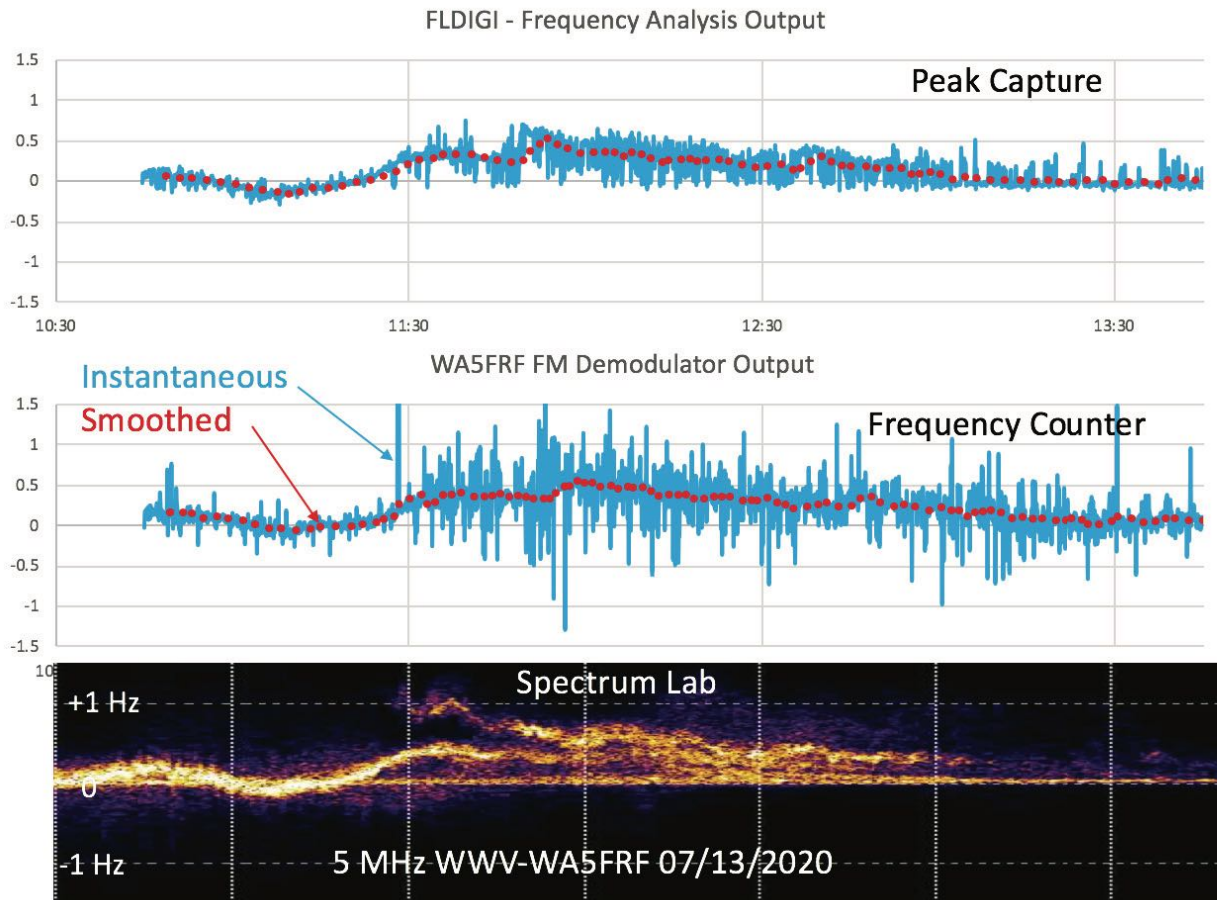


Figure 8 Comparison Between Single Frequency and Spectrogram Displays

The top and middle traces are from the single-valued demodulators. Both instantaneous and smoothed representations are shown. On the bottom is the spectrogram from Spectrum Lab reformatted into a horizontal representation with time amplitude scales matching the other two. The spectrogram shows the presence of three distinct frequency tracks that cannot be deduced from the other two representations. Prior to mode divergence they both followed the spectrogram with fidelity and comparatively low noise. After divergence at approximately 1130z the FLDIGI output showed data spikes corresponding to each of the three modes, depending on relative amplitude. The period counter began showing the bipolar noise spikes that occurred during local amplitude minima.

Each detection type has advantages depending on the task at hand. FLDIGI has become a very popular tool in the Frequency Measurement Test (FMT) community. The period counter proved useful in mode interference analyses that occur on rapid time scales. But for general study of ionospheric phenomenon the spectrogram format provides the most complete data and should be the method of choice for frequency analysis.

III. Precise Timing Measurements on WWV Per-second Ticks to Infer Mode

A. Possible Mode Splitting Mechanisms

The author had hypothesized at least some characteristics of the observed mode splitting on 5 MHz WWV at dawn might be caused by propagation paths using different modes. For a given propagation path between two fixed stations the signal may take one, two, or more hops to get from transmitter to receiver. Multiple hop modes between two fixed stations have longer path lengths because of the extra up-and-down zig-zagging. Path length increases with the number of hops. If all modes use the same ionization height for refraction and that height descends at a given rate, then the longer paths have a faster closure velocity simply because there is more distance change in the same amount of time. The higher closure velocity then creates more Doppler. If conditions permit refraction at the higher angles associated with the multihop modes then the result is simultaneous propagation with the higher order modes showing more Doppler shift according to number of hops. This premise is reinforced by the occurrence of high order modes that abruptly manifest part way through the night-day transition. During the first part of the transition there is insufficient ionization to support the angle required by the mode. When ionization reaches the level required to refract that angle, the mode appears.

Alternate mechanisms for mode splitting certainly exist and might include modes that enter and spend a lot of time within a layer (Pedersen Rays) and frequency shifts from accelerating and decelerating wave speeds. Bill Engelke AB4EJ suggested the possibility of multiple refractions from different layers with different ionization velocities. Of course all of these processes and more may be happening all at once. Additional complications at dawn include appearance or strengthening of the of the D and E layers, splitting of the F layer, and occasional simultaneous presence of WWV and WWVH carriers, each with their own Doppler shifts. The morning transition is a busy time. Sorting out possible contributions requires additional information that can be obtained by designing experiments to test for specific characteristics. It was towards this end that data from a timing experiment was sought.

B. Mode Order Determination through Time-of-Flight Measurement

In order to test the multiple hop idea a method was needed to confirm the presence of simultaneous multiple hop modes. Both measurement of vertical arrival angle and a time of flight measurement were considered. The time of flight measurement was deemed the better choice, and WWV provides convenient time markers via the per-second timing ticks. Since path length increases with number of hops, mode information can be inferred from time of flight. Information on reflection height rates of change, and therefore Doppler shifts, could then be combined with mode data to help analyze spectral recordings.

C. Timing Tick Format

WWV and WWVH transmit audio “ticks” once per second according to published schedules. The timing ticks last 5 mS and are 5 cycles of a 1 kHz sine wave for WWV and 6 cycles of a 1200 Hz

sine wave for WWVH. Figure 9 shows published timing tick data and on-air signals received at WA5FRF displayed on a two-channel oscilloscope. The oscilloscope was connected to the audio outputs of both a 5 MHz receiver (top trace) and a 10 MHz receiver (bottom) set up in AM mode. Horizontal scale factor was 5 mS/div. The top oscilloscope trace shows simultaneous reception of both the 5-cycle tick from WWV followed by the 6-cycle tick from WWVH. The bottom trace shows reception of WWVH only on 10 MHz. The great circle distance is 1350 km from WA5FRF to WWV and 6050 km to WWVH, resulting in a time difference of arrival of about 17 mS. With this station geography there was no problem separating the ticks from WWV and WWVH on the basis of arrival time.

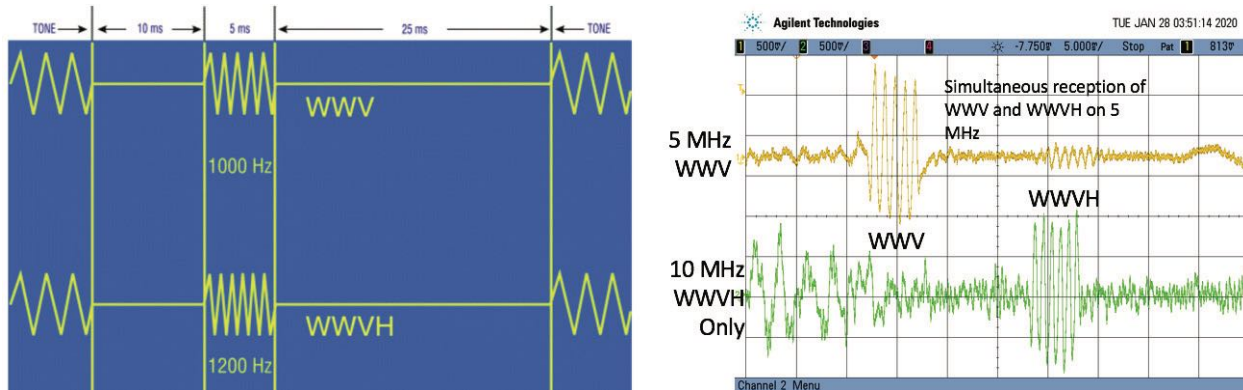


Figure 9 Published Timing Tick Specifications and On-air Received Signals

D. Timing Reference and Receiver Calibration

Three critical elements are required to make precise WWV time-of-flight measurements: an accurate sync pulse referenced to when the pulses are actually launched from WWV, an accurate measurement of the time delay through the receiver, and a consistent On-Time Marker or reference location on the received pulses. A modern GPS Disciplined Oscillator module outputs a 1-second pulse. The rising edge is accurately aligned with the moment of pulse launch from WWV and was used as the oscilloscope sync. The IC-7610 radio is an SDR and the propagation time through the receiver DSP is both lengthy and mode dependent. The propagation delay was measured by feeding a calibration signal into the receiver input and measuring the time delay to the audio output. The test signal was an AM burst generated by a pair of laboratory function generators set up to mimic the WWV timing tick. The delay was measured to be 4.0 mS in AM mode. The published on-time marker for the timing tick is the very beginning of the pulse. However determining the precise instant it actually starts on an oscilloscope display can be difficult, especially under noisy conditions. The middle of the first positive half-cycle of the burst was deemed a much more reliable point of reference and facilitated easy visual measurement using the cursors built into the scope. The first full half cycle of the timing tick was measured at $\frac{3}{4}$ of a cycle after pulse start. At 1 kHz the period of one cycle is 1.0 mS so this measurement reference is 0.75 μ S from the beginning of the pulse. Therefore a time measurement made from sync start to the middle of the first positive half-

cycle on the oscilloscope display required a 4.0 mS correction for receiver delay plus 0.75 mS for the measurement method for a total correction factor of 4.75 mS.

E. Estimated Arrival Times, Superposition of Multiple Modes, and Pulse Broadening

A simplified “flat earth” geometry was used to get a feel for expected times of flight over the 1350 km great circle path from WWV to WA5FRF. The left side of Figure 10 shows the geometry and formulas used to calculate the path distance for 1, 2, and 3 hop modes as a function of virtual reflection height. These distances were converted to time-of-flight and plotted as a function of reflection layer height in the graph on the right. Note that the slopes of the curves increase with mode order. So for a given rate of change in layer height the velocity of path change increases with mode order, predicting more Doppler shift for more hops.

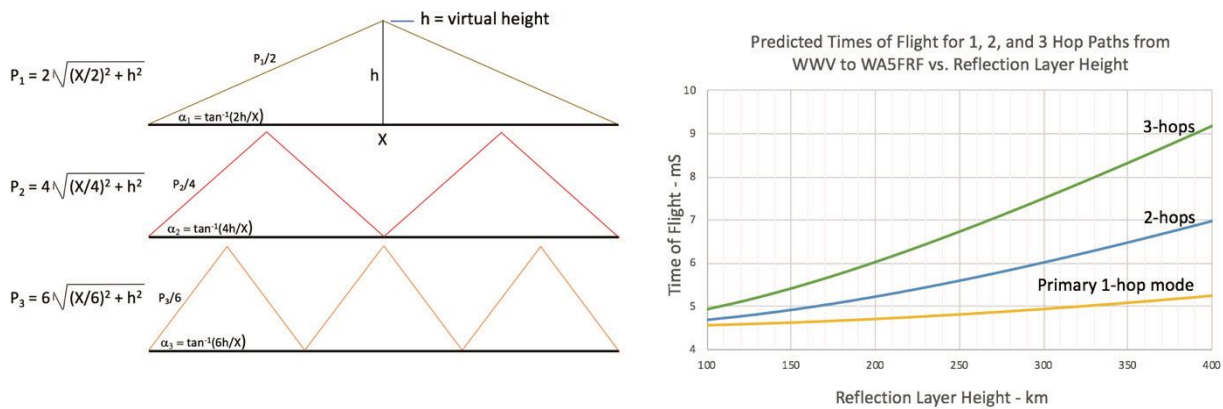


Figure 10 Idealized Geometry used to Estimate Time of Flight vs. Virtual Reflection Height and Predicted TOF for 1, 2, and 3 Hop Modes

At a virtual reflection height of 250 km, the times of flight for 1, 2, and 3 hop modes are 4.80, 5.60, and 6.73 mS respectively. A 2-hop mode would arrive 0.8 mS and a 3 hop mode would arrive 1.93 mS after the primary 1 hop pulse. But the timing tick from WWV is 5 mS long. So under conditions of simultaneous propagation the 2 and 3 hop modes would come down on top of the 1 hop pulse resulting in superposition. At the pulse frequency of 1 kHz, the delayed pulses would arrive about 1 and 2 cycles after the start of the primary pulse.

The superposition of two 5 mS pulses at 1 kHz was modeled in a Spice simulation as a function of phase difference. The upper half of Figure 11 shows results of the simulation for phase differences of 1, 2, and 3 mS, or 1, 2, and 3 complete cycles. For 1 mS delay the resultant consists of the first cycle of the first pulse followed by 4 cycles of the sum of the two, followed by the last cycle of the second pulse. Similarly a 2 mS delay results in 2 cycles from the first, 3 cycles of sum, and 2 cycles from the second. Finally, a 3 mS delay results in 3 cycles from the first, 2 cycles of sum, and 3 cycles from the second. The envelope amplitude shows individual pulse amplitudes as well as the sum. The overall result of the superposition is to lengthen the

resultant pulse by the arrival time difference. Time delays of 1, 2, and 3 mS produced resultant pulses containing 6, 7, and 8 cycles.

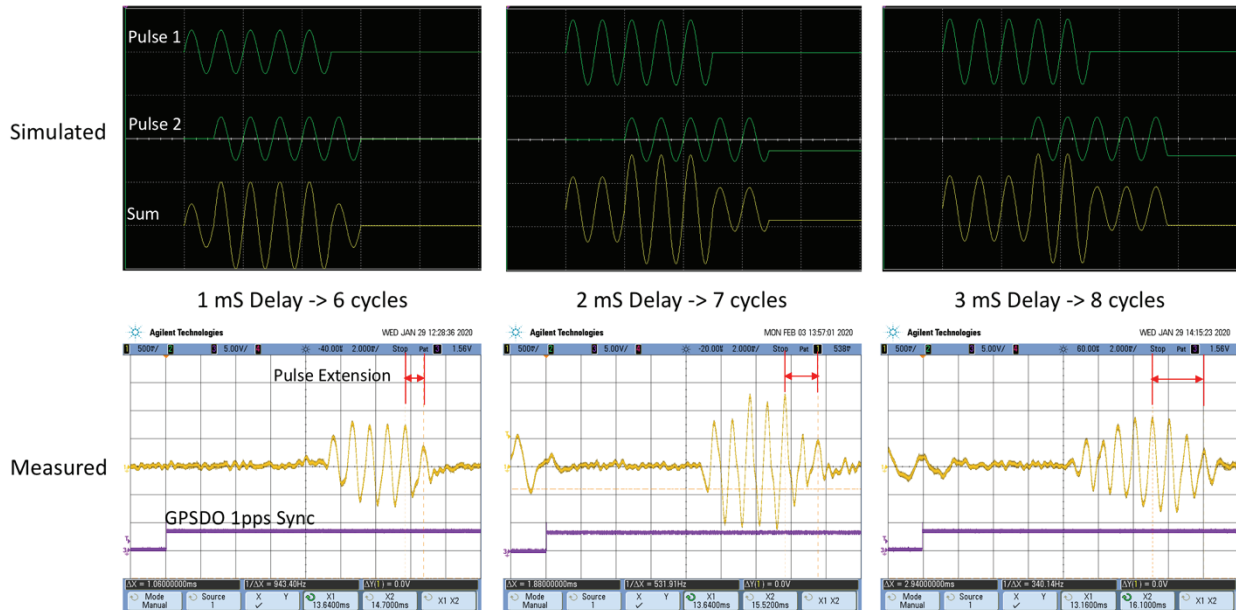


Figure 11 Simulated 5 mS Waveforms with 1, 2, and 3 mS Delayed Superpositions and Recorded On-air WWV Waveforms Showing Similar Characteristics

With this information in hand, it was time to go on the air to see if pulses with these characteristics were actually present during a dawn transition. They were, and oscilloscope recordings of 5 MHz WWV with these characteristics are shown beneath the corresponding simulations in Figure 11. Not shown in the figure are results of superpositions with time differences in-between integral full cycles. These produce more complex waveforms including a central null for the special case of a half-cycle difference. Some of these were observed on the air as well.

F. Measured Data over a Complete Morning Transition

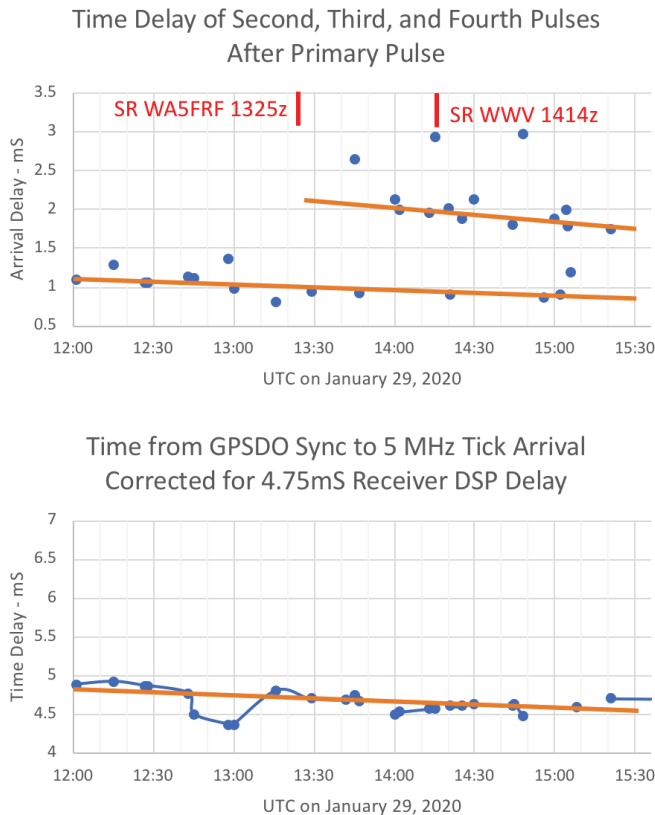


Figure 12 Cluster Plots of Tick Timing Data from January 29, 2020

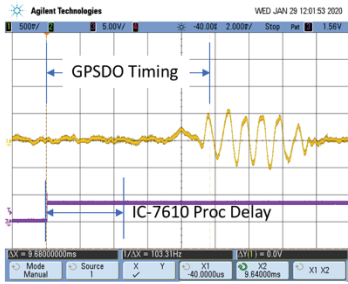
Data over a 4-hour morning transition was acquired on January 29, 2020. The data was acquired manually by enabling the oscilloscope to trigger on the next available trigger pulse from the GPSDO. Approximately 250 out of the roughly 14,000 possible timing ticks were captured. Primary pulse arrival time was measured at the beginning of the composite pulse. Time delay for a second pulse was obtained from the time extension at the end caused by superposition. Figure 12 shows a cluster plot of the data. The acquired data appeared to cluster in monotonic patterns that were consistent in time and slope with the predicted multihop data of Figure 10 for a specific layer height. Note the higher order modes did not manifest until midway through the transition. This is consistent with data observed on spectrograms.

The timing data supports existence of simultaneous multiple hops and even identified how many hops a particular pulse may have taken. Doppler calculations based on this very sparse data set showed promise in that they predicted frequencies that were present in spectrogram data. Pulse timing data should prove useful in in spectral analyses when use conjunction with other analytical methods and measurements.

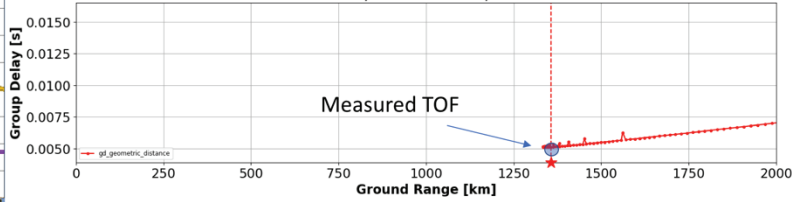
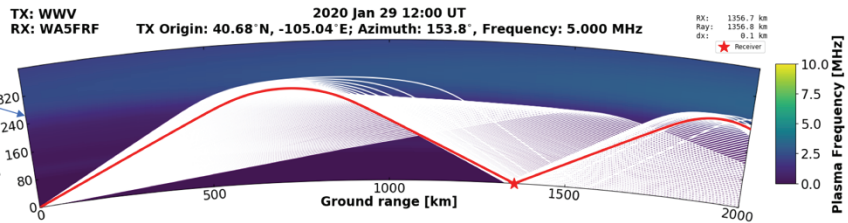
G. Correlation with Ray Trace Predictions

Theoretical ray trace data was provided by the HamSCI community to make comparisons with the measured timing data. Published propagation conditions for the day of the experiment were used. Nathaniel Frissell W2NAF provided PHaRLAP ray trace data for a single hop mode at 1200z and for a 2 hop mode at 1324z. Carl Luetzelschwab K9LA provided a Proplab Pro simulation for a 3 hop mode at 1430z. Corresponding data records were available at 1201z, 1329z, and 1444z. Figure 13 shows side by side comparisons between the ray trace predictions and the oscilloscope data records. The measured Times of Flight were consistent with those predicted by the ray trace programs.

Refraction height inferred from idealized geometry consistent PHaRLAP prediction.

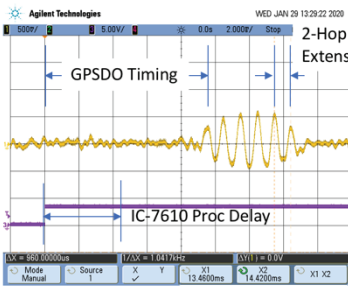


Measured Data

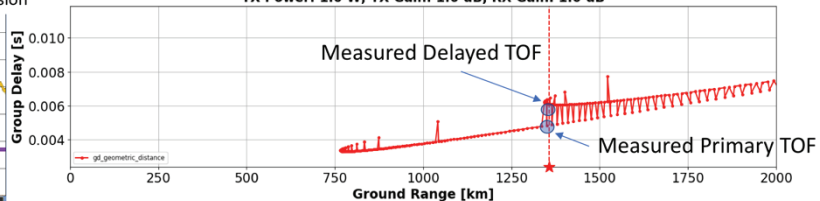
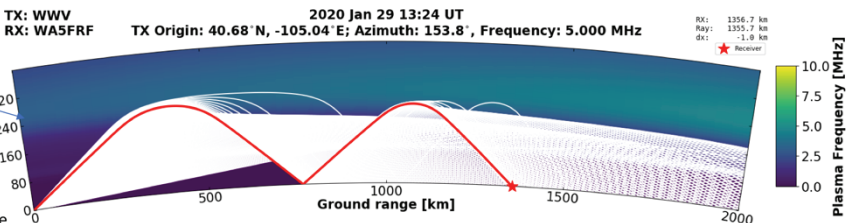


Predicted group delay near 5 mS consistent with measured arrival time of 4.93 mS for primary 1-hop mode at 1201z. $T = 9.68$ (GPSDO) - 4.75 (IC-7610) = 4.93 mS.

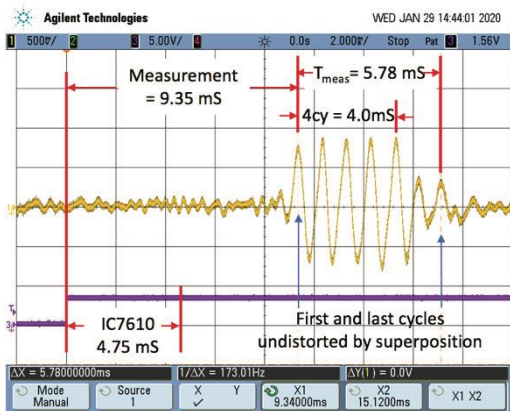
Refraction height inferred from idealized geometry consistent PHaRLAP prediction.



Measured Data

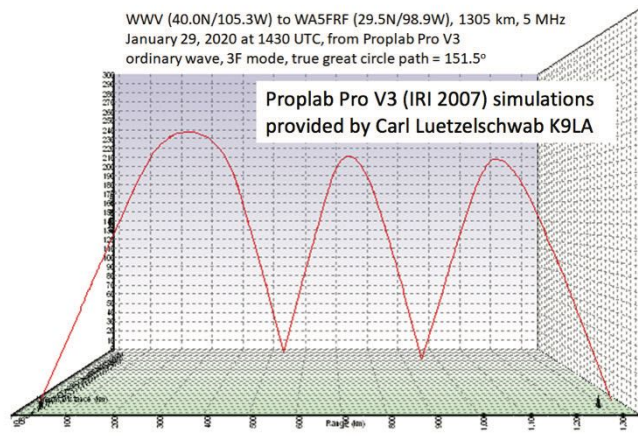


Predicted group delay near 6 mS consistent with measured 2-hop arrival time of 5.67 mS at 1329z. $T = 9.46$ (GPSDO) - 4.75 (IC-7610) + 0.96 (Extension) = 5.67 mS



Measured 3F TOF = Leading Edge - IC7610 + T_{meas} - 4 cy
 $= 9.35 - 4.75 + 5.78 - 4.0 = 6.38$ mS

WWV (40.0N/105.3W) to WA5FRF (29.5N/98.9W), 1305 km, 5 MHz
 January 29, 2020 at 1430 UTC, from Proplab Pro V3
 ordinary wave, 3F mode, true great circle path = 151.5°



ray trace	mode	elev angle	az angle	total dist in km	time of flight assuming speed of light
3D	3F	49	143.3	1938	6.46 msec

Figure 13 Measured Times of Flight were Consistent with Ray Trace Models

H. Requirement for Automated Data Acquisition and Analysis

This timing experiment is encouraging in that the results are consistent with multihop propagation as predicted by ray trace programs. It is supportive of the premise that at least some aspects of observed spectral mode splitting could come from this form of multipath and should provide a useful analysis tool when used in conjunction with other measurement types and theoretical predictions. The main problems with the experiment reported here are that the acquired data was sparse, manually acquired, and subject to human interpretation. What is needed is an automated data acquisition method that takes data on every available timing tick and then uses a computerized waveform analysis to extract the timing relationships of the overlapping pulses. Much more complete Doppler shift predictions could then be computed from measured path length changes as a function of time. This data then can be compared with an actual spectrogram to see how much of the observed data can be attributed to this effect and what requires additional analyses.

IV. Separation of WWV and WWVH During Simultaneous Reception

A. Spectral Overlap from Simultaneous WWV and WWVH Reception

Simultaneous reception of both Colorado WWV and Hawaii WWVH is a common occurrence at the author's South Texas QTH. Both stations have atomic clock accuracy and ionospheric frequency variations do not amount to more than a very few Hz so the carriers are precisely zero beat and they sound like one station to the ear. The easiest way to tell if you are receiving both is by listening to the time announcements at the top of the minute. The female voice from WWVH comes on first at about 20 seconds before the top of the minute followed by the male voice from WWV about 10 seconds later. Simultaneous reception of both carriers results in superposition of the spectra from both stations in a spectrogram, further complicating an already complicated spectra. Because the carriers use different parts of the ionosphere and come in from very disparate distances their spectra are completely different.

A full analysis of the spectra from WWV is very difficult if the spectra from WWVH cannot be removed. From South Texas separation could be accomplished with directional antennas since angular separation between Hawaii and Colorado is around 50 degrees. But for many stations directly east or west of Colorado the two stations are nearly inline, making an antenna solution virtually impossible. Fortunately the crafters of the WWV/WWVH system had the foresight to build in a solution to separate WWV from WWVH during simultaneous reception.

WWV and WWVH alternate use of 500 and 600 Hz tones in their broadcasts which possess the same frequency accuracy as the carrier. David Kazdan AD8Y recognized the possibility of WWVH interference and suggested use of these tones as a means to separate the two signals. Methods for achieving this separation through deinterlacing the tones were developed by the author and are presented in the following sections.

B. 500 and 600 Hz Tone Interlacing

WWV and WWVH alternately transmit standard tones of 500 and 600 Hz on most (but not all) minutes of each hour. Both stations use both tones in an interlacing scheme described at <https://tf.nist.gov/stations/iform.html>. The stations swap tones each minute in an even/odd minute order. WWV transmits the 500 Hz tone on even minutes and the 600 Hz tone on odd minutes. WWVH uses the complimentary process. Here is a summary:

Minute	WWV	WWVH
Even	500 Hz	600 Hz
Odd	600 Hz	500 Hz

C. Deinterlace Methods

Numerous separation schemes are possible using one or both tones. Three possible options with increasing complexity and capability are reported below. For these studies the author used a GPSDO stabilized Icom IC-7610, which has dual receivers. In the examples presented here the tone of interest was frequency translated to 1000 Hz so the spectrograms use the same 1000 Hz center frequency, 10 Hz span format used elsewhere in this paper.

1. Interlaced Data using Only One Tone

If a receiver and data recorder are set up to listen to only one tone, WWV and WWVH data appear sequentially in a single waterfall. The individual contributions are recognizable in the alternate-minute composite but only half of the available data for each station is displayed. An example of this method is shown in Figure 14.

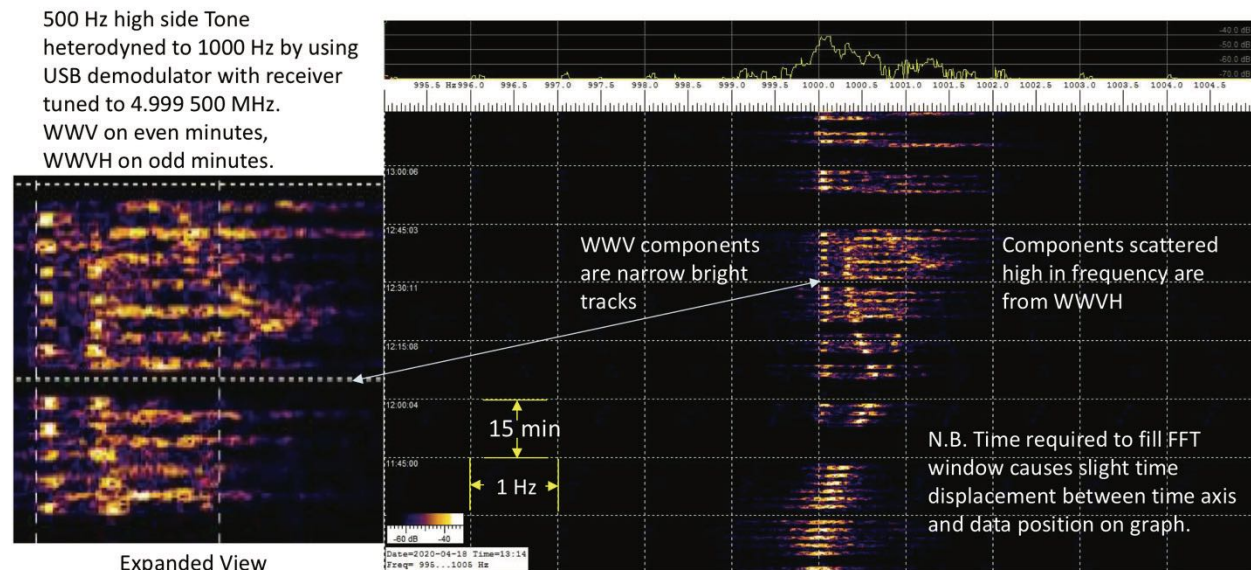


Figure 14 Spectrogram Using a Single Tone. WWV and WWVH Data are Separate but Interlaced

2. WWV or WWVH -only Data Obtained by Muting the Unwanted Tone on Alternate Minutes

This method uses one tone to display only one station at a time by muting the unwanted tone every other minute. It presents sparse data records with 50% of the data missing. The method can be implemented entirely in audio processing with no requirement to control the frequency of either the radio or the analysis program. This method also occurs naturally if only one of the two stations can be heard. An example of this data type is shown in Figure 15.

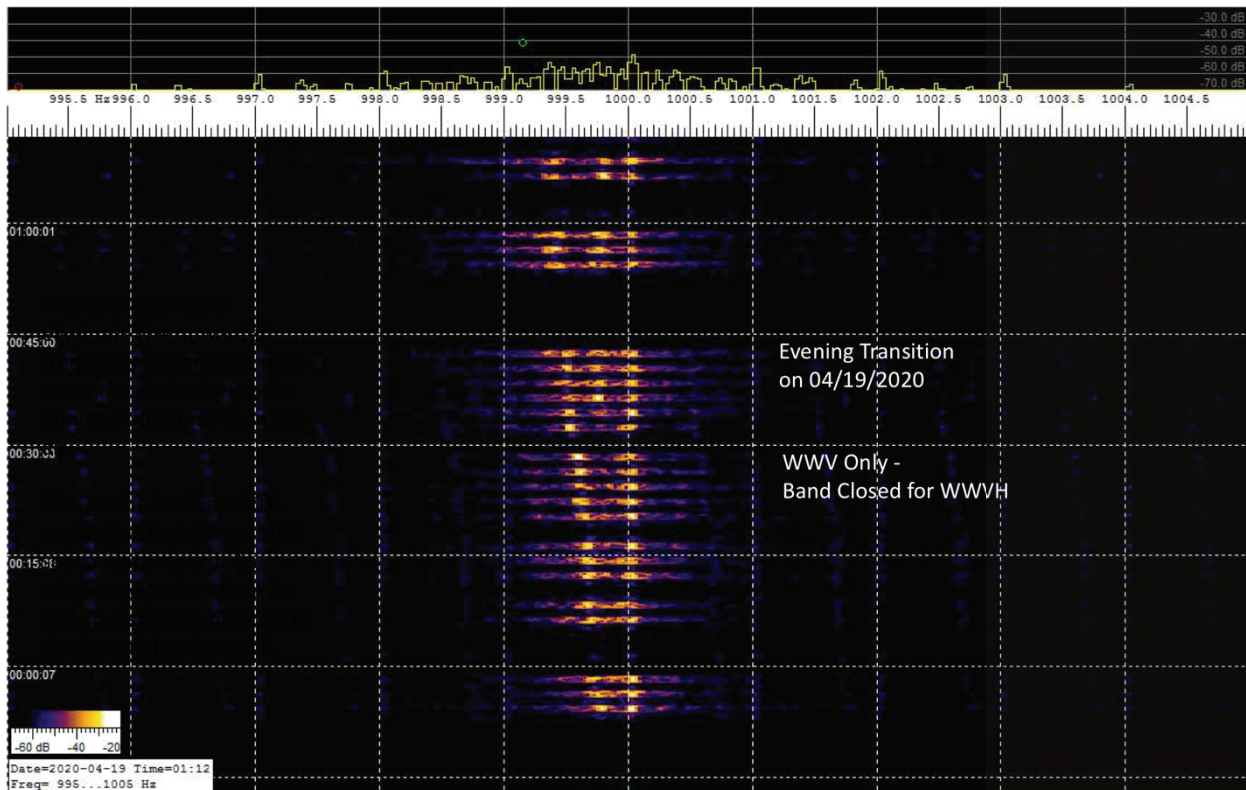


Figure 15 Spectrogram Using a Single Tone but with Alternate Minutes Missing

3. Fully Deinterlaced Processing Placing 100% Continuous Data in Separate Data Records

This method provides full data separation with complete records for both stations but requires two receivers with the ability to change frequency every minute so that one receiver continuously monitors WWV and the other WWVH. Alternatively, full deinterlace can be accomplished if the center frequencies of the data recorders are programmable. A subset of this method is to use only one programmable receiver or data recorder to yield 100% data on only one station. Figure 16 shows data acquired with this method with WWV-only shown on the left side of the figure and WWVH-only on the right. The data traces show all of the transmitted data; the missing minutes are those intentionally not transmitted by WWV. The short gaps between sequential traces are when the tones are muted for the voice announcements prior to the top of each minute. The scale factors were kept the same for both spectrograms so relative amplitude information would be preserved in trace brightness. Mode splitting is clearly present in the WWV trace. The WWVH trace shows a mostly single diffuse mode with a different amplitude profile. Without deinterlacing these spectra would have been superimposed. The mode splitting on WWV would still be evident but blurred by the presence of the WWVH spectra. The spectral information on WWVH would have been unrecognizable.

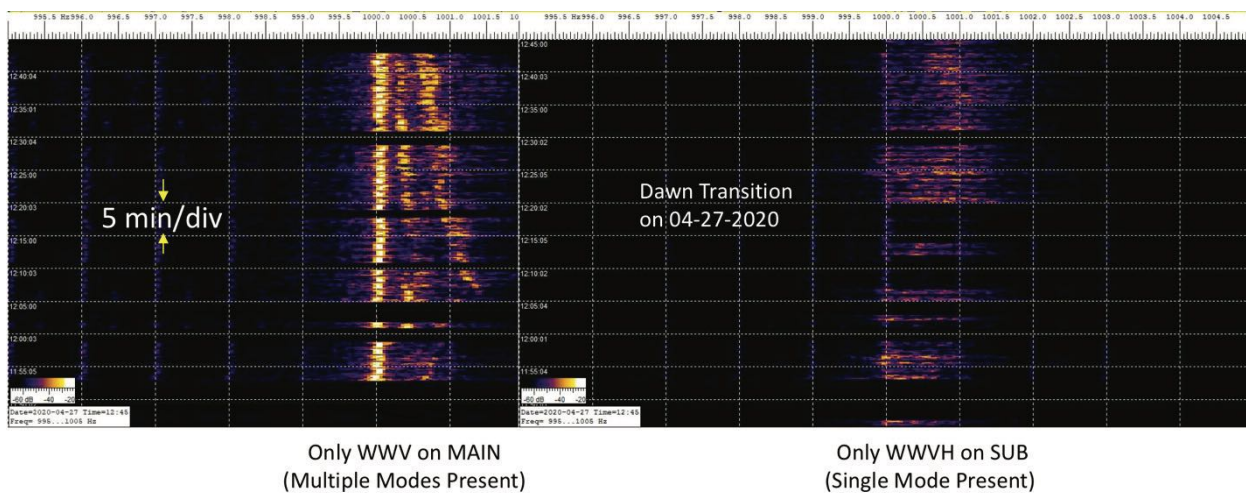


Figure 16 Spectrograms Showing Full Separation with Complete Data Records

Figure 17 shows the setup used to acquire the data for the 5 MHz WWV/WWVH data. Both receivers in an Icom IC-7610 were used and set for USB mode. The Main receiver was set to 4999.5 MHz to translate the 500 Hz tone to 1000 Hz. The Sub receiver was set to 4999.6 MHz to translate the 600 Hz tone to 1000 Hz. Two instances of Spectrum Lab were run on separate laptop computers with Main audio ported to one and SUB audio to the other. Both programs were set up for a 1000 Hz center frequency with a 10 Hz span. The receivers were toggled manually using the CHANGE button prior to the top of each minute throughout the entire course of the experiment.

The left side of the figure shows an automated method to accomplish receiver toggling if an Odd/Even marker can be obtained from an external WWV receiver. This signal could be used, for example, with a microcontroller to toggle the CHANGE command via the CI-V interface.

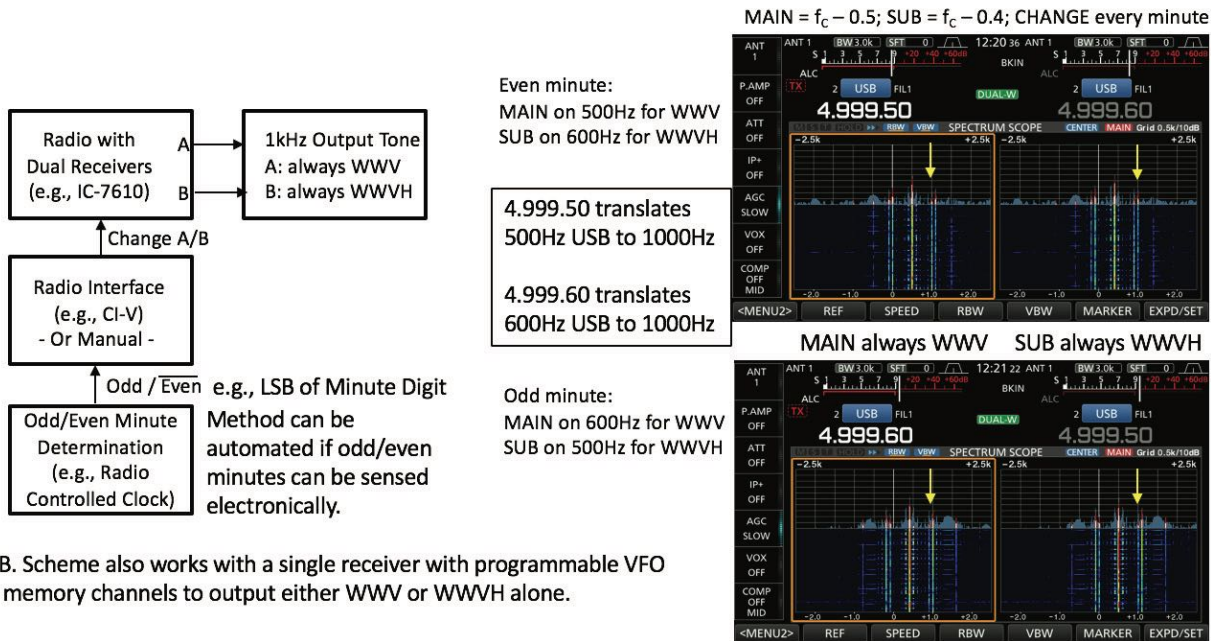


Figure 17 Setup for Full WWV/WWVH Deinterlace Using an Icom IC-7610 Dual Receiver

The right side of the figure shows annotated screen shots of the IC-7610 in the two receive states. The top photo shows MAIN setup for 500 Hz and SUB for 600 Hz. The bottom photo shows reversal after pressing the CHANGE button. The waterfall displays the carriers and both the 500 and 600 Hz tones. Using USB mode and only the high side tones preserves frequency direction sense.

V. Conclusion and Acknowledgements

Ionospheric skywave propagation is a marvelously complex phenomenon with many interesting aspects. The amok times during the morning and evening transitions where apparent frequency and propagation modes run wild is especially fascinating. The underlying physics possess many subtleties which are mysterious to the author in particular and perhaps much of the rest of the amateur radio community in general. But they are still fun to think about and to do so is as engaging as it is addictive. Using different experimental tools can help in deciphering the multiple physical processes involved. The purpose of this paper was to give examples of some of the tools developed for the HamSCI community for the study of ionospheric behavior.

The author would like to thank Nathaniel Frissell W2NAF, Carl Luetzelschwab K9LA, David Kazdan AD8Y, Bill Liles NQ6Z, and Bill Engelke AB4EJ for their valuable insights.

QMesh: A Synchronized, Flooded Mesh Network Protocol for Voice

Dan Fay, Ph.D. (KG5VBY)

Abstract

Wireless mesh networking protocols, such as APRS and Gotenna Mesh, allow for reliable communication between devices by relaying data between nodes until the data reaches its destination. While these mesh networks successfully communicate telemetry and small data bursts, they cannot provide the continuous streaming bandwidth needed to carry real-time voice communications. Presented here is QMesh, which leverages the FM capture effect in the LoRa waveform to allow for reliable synchronized, flooded mesh networking that can support low data rate digital voice communications. An overview of the theory, protocol design, experimental hardware, initial test results, and plans for QMesh will be discussed.

Overview of LoRa

LoRa, which is a concatenation of “Long Range”, is Semtech’s proprietary Chirp Spread Spectrum (CSS) modulation. It is targeted as a low data rate, low-power, low-cost wireless communications waveform that is used for low-power sensing and other Internet-of-Things (IoT) uses. LoRa possesses two advantages over other, more commonly used waveforms such as Frequency Shift Keying (FSK) and On-Off Keying (OOK). The first advantage is that by spreading its energy across a large bandwidth (up to 500KHz), it can better resist interference from narrowband interference. Second, LoRa enjoys substantially better receive sensitivity than most other narrowband waveforms for a given bitrate.

Semtech provides several LoRa products: the SX127X series, the SX126X series, and the SX1280. The SX127X series is the original LoRa radio line designed for sub-GHz communications. The SX126X series is a successor to the SX127X series that enjoys lower receiver power consumption and can cover all frequencies from 150MHz to 960MHz. The SX1280 is a LoRa product designed to work on the 2.4GHz ISM band. It enjoys higher LoRa communication rates than the sub-GHz modems. There are also several products by other companies that combine a LoRa radio with a microcontroller. These include the Microchip SAM R34/R35 and the STM32WL series. It is of note that the STM32WL is likely the first LoRa modem die that is not fabricated by Semtech, as the SX1262 modem on it is on the same silicon die as the STM32L0 MCU (the SAMR34/35 package together an MCU die and an SX1276 die).

The LoRa Waveform

LoRa is a form of Chirp Spread Spectrum (CSS). CSS modulation encodes information within chirps, typically with the goal of improved system gain and narrowband interference rejection. The purported benefits of the LoRa waveform are several. First, it enjoys a higher receive sensitivity for a given bitrate than do common FSK and OOK waveforms. Second, the spread spectrum characteristics of the waveform improve the resistance of the LoRa waveform to narrowband interference. These characteristics make it well-suited for low-power, battery-operated wireless sensing devices that operate on the unlicensed Industrial, Scientific, and Medical (ISM) bands like the 868MHz and 915MHz bands.

Unfortunately, Semtech does not provide detailed information about the LoRa waveform, so the information presented here comes from a combination of Semtech-provided documentation, reverse-engineering projects, and Semtech patents. At a basic level, LoRa is a chirped m -ary FSK, where a large set of FSK tones are multiplied by a chirp. This operation encodes the information as a “break” in the chirp. A LoRa chirp is broken up into multiple, short-duration tones, known in the Semtech documentation as “chips”. There range from 32 up to 4096 chips per chirp, depending on the Spreading Factor (SF) setting. The number of chips per chirp is equal to 2^{SF} .

There are three major LoRa settings: bandwidth (BW), spreading factor (SF), and coding rate (CR). On the sub-GHz LoRa products, bandwidths range from 500KHz all the way down to 7.8KHz, although the most commonly-used bandwidths are 500KHz, 250KHz, and 125KHz (NOTE: the 2.4GHz LoRa waveform uses wider bandwidths than the sub-GHz waveform). Narrower bandwidths trade higher receive sensitivity for lower bitrates. Spreading factors range from 5 to 12, with lower spreading factors trading a higher datarate for lower receive sensitivity. Finally, the coding rate specifies the Hamming code used to provide error detection/correction. Officially, the coding rate can be 4/5 (CR=1), 4/6 (CR=2), 4/7 (CR=3), and 4/8 (CR=4); however, there does exist an undocumented feature in at least some of the LoRa modems that allows for completely disabling the built-in error correction by setting CR=0.

LoRa does not tolerate much frequency drift within a given packet transmission, due to the close frequency spacing between chips. There is a Low Datarate setting that trades two bits per symbol in exchange for a sixteen-fold improvement in frequency drift tolerance. Regardless, while it is not necessary for higher datarates, a TCXO is necessary for lower datarates (less than BW=62.5KHz/SF=12), and long transmission times (>~3 seconds). A TCXO is also recommended for transmit power over 20dBm due to heating of the oscillator by the power amplifier (PA), unless the board is carefully designed to thermally isolate the oscillator from the modem chip. Semtech’s LoRa modems, however, are highly tolerant of absolute frequency error, and can successfully receive LoRa packets whose center frequencies are off by up to +/-10-25%.

LoRa Implementation

LoRa packets have a preamble to enable the receiver to detect and synchronize the receiver, as well as a “sync word” which appears to be a reverse chirp that may also synchronize the receiver. Different sync words can be specified; the LoRa receiver will ignore other sync words that it is not set to. Typically, two different sync words are used: 0x12 for private networks and 0x34 for public networks. While other sync words can be set, they should be thoroughly tested, as some sync words are not reliably received. Note that it is possible to set sync words that are incompatible between the SX127X series and the SX126X series.

Another feature provided by Semtech’s LoRa implementations is the ability to reverse the I and Q samples in the receiver. This has the effect of turning upchirps into downchirps, and vice versa. The practical use for this is that forward and reverse chirps are orthogonal and largely non-interfering with each other, which facilitates full-duplex communications. LoRaWAN, for example, uses different IQ

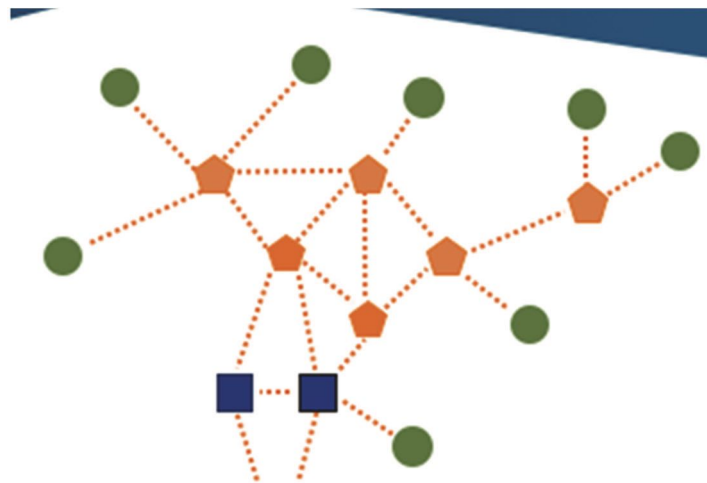
polarities on the uplink and downlink to make it possible to simultaneously transmit and receive, although real-world LoRaWAN gateways do not appear to perform full-duplex communications.

Semtech provides two different packet modes: “implicit header” and “explicit header”. Implicit header mode is a standard fixed-length packet. Explicit header mode provides a quasi-link layer with a header that describes the length of the data payload, the coding rate of the payload, is protected by a header CRC. LoRa chipsets can also automatically compute the CRC of a packet.

Semtech implements the LoRa waveform on several product lines. The original one is the SX127X series, which is the original LoRa chipset for sub-GHz frequencies. There is also the newer SX126X series, which is another sub-GHz chipset with some improved features such as lower receiver power consumption, higher transmit power (22dBm on the SX1262 vs. 20dBm on the SX1276) on some models, as well as support for SF=5. Finally, the SX1280 is a LoRa chipset for 2.4GHz that uses a wider bandwidth for higher datarates.

Overview of Wireless Mesh Networking

Mesh networking is a non-hierarchical network topology where nodes work cooperatively to move data packets through the network. Typically, mesh networks can handle nodes entering and leaving the network while still maintaining a functioning network. Wireless mesh networks are a common use case for mesh networking, particularly for Mobile Ad-hoc NETWORKS, or MANETs. MANETs are highly ad-hoc networks where the network topology is expected to dynamically change: nodes can enter or leave the network on a whim, nodes can change location, etc. Real-world examples of MANETs include e.g. groups of hikers sharing telemetry data, vehicle-to-vehicle communications, and drone swarms.



Source: <https://blog.particle.io/2018/04/28/how-to-build-a-wireless-mesh-network/>

Figure 1: Diagram of a mesh network.

Mesh network protocols fit into two broad categories: flooded and routed protocols. Flooded protocols are where every node repeats every packet that it receives. Flooded protocols enjoy the benefits of

simplicity (no need to generate and maintain routes) and self-healing at the expense of efficiency (lots of unnecessary retransmissions of packets). Routed protocols, where nodes only repeat packets if they are along a defined route, are more efficient, but more challenging to build and maintain working routes as network conditions change.

Another issue with mesh network protocols, particularly ones that use a shared medium such as RF communications, is known as the *broadcast storm problem*. When multiple nodes transmit at the same time, their packets collide. If the collisions are completely destructive, the wireless communication will fail, requiring a retransmit. Multiple retransmits can in turn destructively collide, causing a situation where endless collisions heavily degrade the wireless channel. Mitigations for this problem include using Time Division Multiple Access (TDMA) slotting (such as “Slotted ALOHA”) and Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA).

An alternative technique that is explored in this paper is what is known as Synchronized Flooding. In synchronized flooding, all nodes repeat a received packet at the *exact same time* (or at approximately the same time). While this guarantees that all the packets will collide, different physical-layer techniques can be used to reduce the likelihood of destructive collisions. Since all retransmitted packets contain the same data, a node only needs to receive one of the retransmitted packets to receive all of the data. An additional benefit of synchronized flooding is that it provides isochronous bandwidth for streaming applications such as real-time voice communications.

Examples of physical-layer techniques to increase the likelihood of successful packet receipt include:

1. **Communicating simultaneously on multiple channels.** A specific case of this is Frequency-Hopping Spread Spectrum-Multiple Access (FHSS-MA). FHSS-MA uses multiple, orthogonal hopping patterns across different nodes to reduce the probability of collisions.
2. **Code Division Multiple Access (CDMA).** Use different pseudo-noise (PN) codes with different nodes while transmitting with Direct Sequence Spread Spectrum (DSSS).
3. **Capture effect.** Many receivers will receive a stronger signal and ignore a weaker signal. This effect is well-known with constant-envelope modulations, where it is called the “FM capture effect”.

The capture effect describes when a receiver, in the presence of two overlapping signals, receives exclusively the stronger of the two signals (i.e., “captures” it). A well-known version of this effect is known as the FM capture effect, as constant-envelope receivers generally experience it. This effect can be seen with FM radio in a marginal reception area. When there are two stations in the area, either one or the other station is received. This contrasts with amplitude modulation receivers (e.g., AM or SSB), where overlapping signals add together. The capture effect is important for any Industrial-Scientific-Medical (ISM) band device, as these devices need to be able to operate in an uncontrolled RF environment where other like devices may be communicating as well. As a constant-envelope modulation, LoRa receivers exhibit the capture effect, and as described in subsequent sections, the LoRa modulation exhibits properties that can be leveraged to increase the likelihood of successful capture.

Other examples of synchronized flooding exist in the literature. One key example is Glossy¹, which demonstrated that successful capture was possible for the DSSS used in the physical layer of 802.15.4 transceivers. One caveat, however, was that the nodes must stay synchronized within one chip period (0.5 μ s in the case of the 250Kbps DSSS-based data rate used). Given that radio waves can travel roughly 150m in that time period, it is easy for wide-ranging networks to lose synchronization. While the situation improves at a lower symbol rate (say 4800baud), a 4x chip rate still has a chip period of roughly 52 μ s, or the time period it takes radio waves to travel roughly 15km.

Later work^{2 3} showed that LoRa transmissions enjoy similar capture benefits to Glossy for the LoRa waveform, provided that colliding LoRa symbols do not completely overlap with each other. In these papers, separation leading to successful capture was provided by a combination of small frequency errors in the modems' oscillators as well as an induced, random timing offset of up to half of a symbol period. A major benefit for LoRa for this application is the comparatively low symbol rate relative to the chip rate of DSSS. For a bitrate comparable to 4800bps, LoRa uses a symbol rate of 1ms, or roughly 19x the period of the DSSS-chipped 4800bps example mentioned previously. In a separate paper, the authors showed that non-spread spectrum, FSK-based synchronized flooding suffers from significantly lower capture rates, although these rates are improved when an interleaved FEC algorithm is used⁴.

Barrage Relay Networks⁵ are another synchronized, flooded network protocol. It appears to be a higher performance waveform/protocol than either LoRa or 802.15.4. It also appears to split up payloads into multiple FEC-coded blocks. Each block is phase-dithered to reduce the likelihood that it constructively interferes with other simultaneous transmissions. Barrage Relay Networks also appear to have more sophisticated receiver processing, with some sort of combining algorithm used to leverage the multiple, colliding signals.

Overview of the QMesh Protocol

The QMesh protocol is a synchronized, flooded mesh protocol that leverages the characteristics of the LoRa waveform to enable successful capture of one of the colliding packets. QMesh adds forward error correction to the LoRa PHY to improve the likelihood of successfully capturing colliding packets. The goal of this network protocol is to support voice communications over this mesh network via `codecs2`-encoded voice. By enabling every QMesh-supporting device to serve as a repeater, better coverage can be obtained for amateur voice in ad-hoc situations such as special event communications and emergencies. Moreover, the single-frequency, store-and-forward architecture of the QMesh protocol,

¹ Ferrari, Federico & Zimmerling, Marco & Thiele, L. & Saukh, Olga. (2011). Efficient network flooding and time synchronization with Glossy. Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN'11. 73 - 84.

² Hou, Y., Liu, Z., & Sun, D. (2020). A novel MAC protocol exploiting concurrent transmissions for massive LoRa connectivity. *Journal of Communications and Networks*, 22, 108-117.

³ Liao, C., Zhu, G., Kuwabara, D., Suzuki, M., & Morikawa, H. (2017). Multi-Hop LoRa Networks Enabled by Concurrent Transmission. *IEEE Access*, 5, 21430-21446.

⁴ Chun-Hao Liao, Makoto Suzuki, and Hiroyuki Morikawa. 2016. Toward Robust Concurrent Transmission for sub-GHz Non-DSSS Communication: Poster Abstract. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys '16)*. Association for Computing Machinery, New York, NY, USA, 354–355. DOI:<https://doi.org/10.1145/2994551.2996703>

⁵ Halford, T.R., & Chugg, K.M. (2010). Barrage Relay Networks. *2010 Information Theory and Applications Workshop (ITA)*, 1-8.

combined with the coding gain provided from high-performance forward error correction allows for compact, low-power (i.e. solar-powered), low-cost repeaters.

LoRa has two potential ways to “spread out” transmissions in order to increase the likelihood of successful capture. The first technique involves spreading out transmit energy at the chip level. Given that LoRa chirps are not continuous frequency sweeps, but rather a series of discrete chirps, slightly shifting the frequencies of the chips should reduce their interference between them. Likewise, by shifting the chirps in frequency and/or time, it should be similarly possible to reduce the amount of interference between colliding packets and increase the likelihood of successful capture. Figures 2 and 3 show how frequency and time shift reduce overlap between transmissions and increase the likelihood of successful capture.

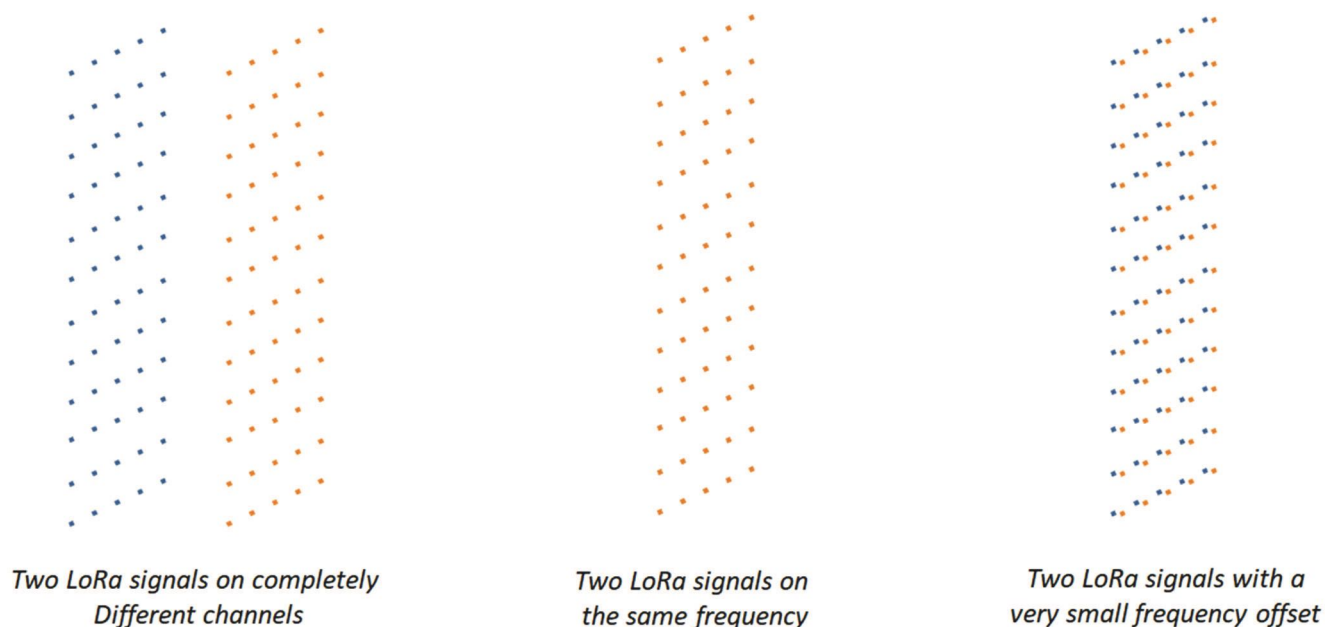


Figure 2: Overview of fine-grained frequency offset.

Another way to increase the likelihood of successful capture is to separate the chirps themselves, which is accomplished by randomly spreading out the chirps by larger amounts in frequency and/or time.

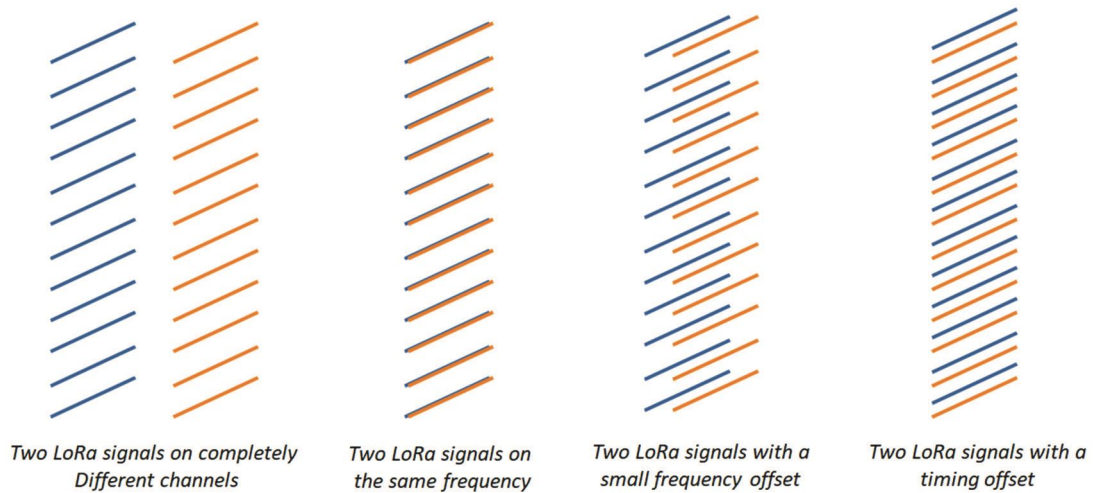


Figure 3: Coarse-grained frequency offset and timing offset.

QMesh's link-layer network protocol is a TDMA protocol that is based on equal-length timeslots, as shown in Figure 4. Within every third timeslot, a node can originate a transmission. Adjacent nodes that receive this transmission wait one timeslot, and then retransmit the packet. This procedure continues down the line until there are no more connected nodes that have not received the packet.

Such a setup provides several benefits. First, having a gap timeslot provides time for the system to receive a packet, decode it (some FEC algorithms can require significant time to decode), and prepare for retransmission. Second, this protocol eliminates interference from adjacent downstream transmissions. Finally, this protocol allows nodes a "second chance" to receive a missed transmission by receiving the transmission of a downstream retransmission.

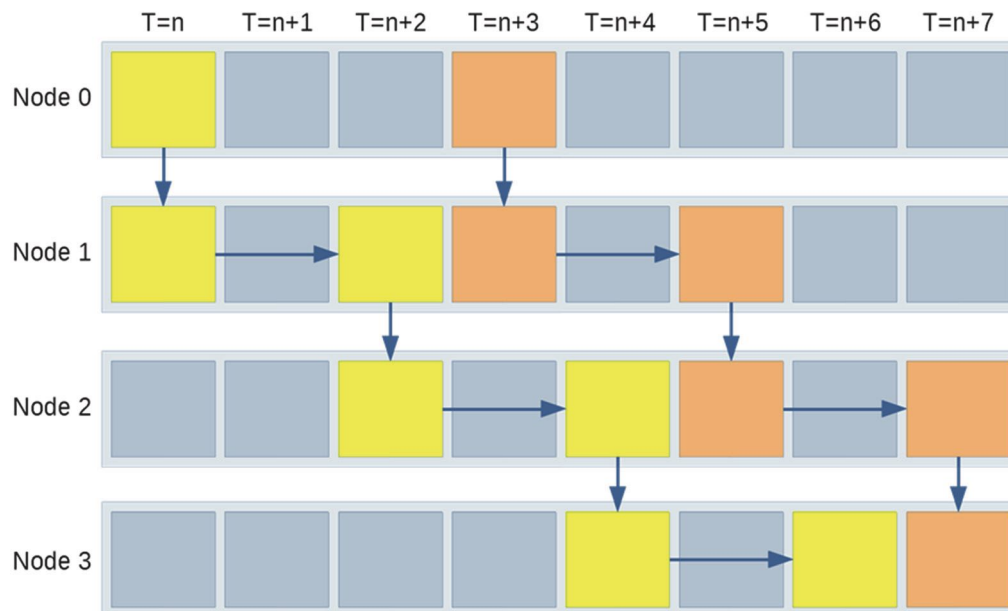


Figure 4: Overview of QMesh's TDMA protocol.

Forward Error Correction

QMesh eschews LoRa's Hamming Code-based FEC in favor of its own, more-sophisticated FEC by using an undocumented feature in the SX126X that allows for completely disabling the built-in FEC by setting the CR=0. This capability was discovered in the documentation for the STM32WL⁶, which combines an STM32 microcontroller and an SX1262 onto a single silicon die. Using a more sophisticated FEC algorithm allows for improving the range through coding gain and also increasing the likelihood of successful packet capture. Coding gain is a metric that describes, typically in dB, the improvement in receive sensitivity due to the ability to correct bit errors in received data.

QMesh currently implements the Reed-Solomon-Viterbi (RSV) forward error correction algorithm. RSV concatenates a convolutional code inner code with a Reed-Solomon outer code. RSV FEC has been used in e.g. the Voyager spacecraft and is currently used with the telemetry frames on the AO-40 amateur satellite⁷. QMesh's implementation uses a (32,24) Reed-Solomon outer code with a ½ rate convolutional inner code with constraint length of 7. A bit-level "rectangular" interleaver, conceptually like the AO-40's FEC interleaver, spreads error bursts throughout the data packet.

RSV was chosen over more modern FEC algorithms such as LDPC, Turbo, and Polar codes for several reasons. First, the RSV algorithm has an easy-to-use, microcontroller implementation-friendly library implementation in `libcorrect`⁸. `libcorrect` is a simplified version of Phil Karn's `libfec`⁹ implementation that removes many of the GNU/Linux (and x86)-specific aspects of the library. Likewise, there was not an easy-to-use, fast version of Turbo codes or Polar codes; moreover, practical implementations of polar codes appear to be encumbered by patents. LDPC codes, while having numerous, reasonable-to-use implementations, require the generation of a parity check matrix specific to the block size and number of parity bits.

While the AO-40's RSV FEC can offer up to 10-11dB of coding gain, it is expected that the amount of coding gain is significantly lower in QMesh due to using hard decoding (the LoRa radio does not provide soft decoding information in any sort of straightforward way) and due to the shorter message length. Soft decoding, where the receiver conveys its confidence in the received symbol's correctness to the FEC decoder, typically improves the coding gain by 1-2dB. Another issue reducing the realized coding gain is the short message size. Short messages can considerably reduce the coding gain by having fewer observations from which to correct messages. A very rough estimate of the RSV FEC in QMesh is that it provides 5-7dB of coding gain versus the built-in Hamming codes. While the author estimates a 5-7dB coding gain for the RSV implementation, this may underestimate the coding gain in certain situations, given that Haystack Technologies claims a 13dB performance gain using their FEC, which appears to use soft decoding and Polar codes¹⁰.

⁶ RM0461, "STM32WLEx advanced Arm®-based 32-bit MCUs with sub-GHz radio solution". Available at https://www.st.com/resource/en/reference_manual/dm00530369-stm32wlx-advanced-armbased-32bit-mcus-with-subghz-radio-solution-stmicroelectronics.pdf on July 11, 2020.

⁷ James Miller G3RUH, "Oscar-40 FEC Telemetry", Available at <https://www.amsat.org/amsat/articles/g3ruh/125.html>, accessed July 11, 2020.

⁸ Brian Armstrong, "libcorrect". Available at <https://github.com/quiet/libcorrect>, accessed July 11, 2020.

⁹ Phil Karn, "Forward Error Correcting Codes". Available <http://www.ka9q.net/code/fec/>, accessed July 11, 2020.

¹⁰ "Say Hello to XR2", Available <https://www.haystacktechnologies.com/xr-error-correction/>, accessed July 11, 2020.

QMesh Hardware and Software Implementation

A key enabler of QMesh is the massive improvement in the compute power and RAM/storage in commonly available, low-cost, low-power microcontrollers over the last decade. These microcontrollers allow for fine-grained, low-level, real-time control while running Real-Time Operating Systems (RTOSes) and supporting high-level C++ constructs like STL containers and smart pointers. These software constructs greatly improve programming productivity.

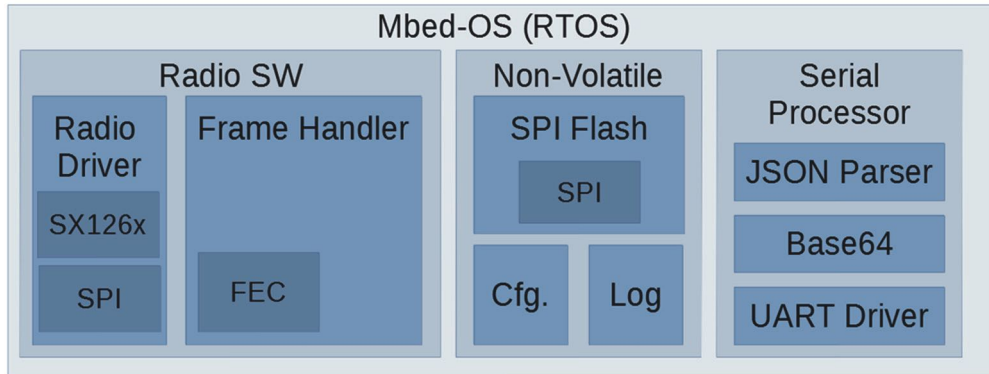


Figure 5: QMesh Software Architecture.

QMesh's current proof-of-concept hardware uses the STM32 NUCLEO-144 boards. ST designs the NUCLEO series of development boards to provide a "simple" development environment for their STM32 products. Non-MCU components are typically limited to a few LEDs and for some boards an Ethernet interface. The NUCLEO line comprises different form factors ranging from small boards that are similar to the Adafruit Feather form factor, all the way up to the larger NUCLEO-144 series, which provides a large number of I/O pins and uses the 144-pin STM32 MCUs. These MCU boards are relatively affordable, with the high-end STM32H743ZI2 costing roughly USD \$30 on Digi-Key as of the writing of this paper.

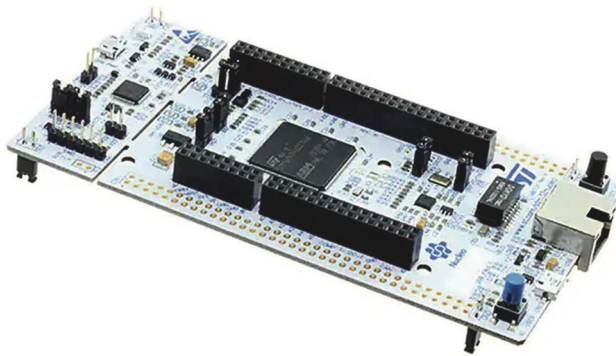


Figure 6: An STM32 NUCLEO-144 board.

The STM32H7 is the high-end line for STM32 microcontrollers. All H7s contain at least an ARM Cortex-M7, whose microarchitecture is like a Pentium MMX, and runs at 400-480MHz. Some variants also contain a secondary Cortex-M4. The STM32H743ZI contains an ARM Cortex-M7 running at 480MHz, 1MB of SRAM, and 2MB of flash.

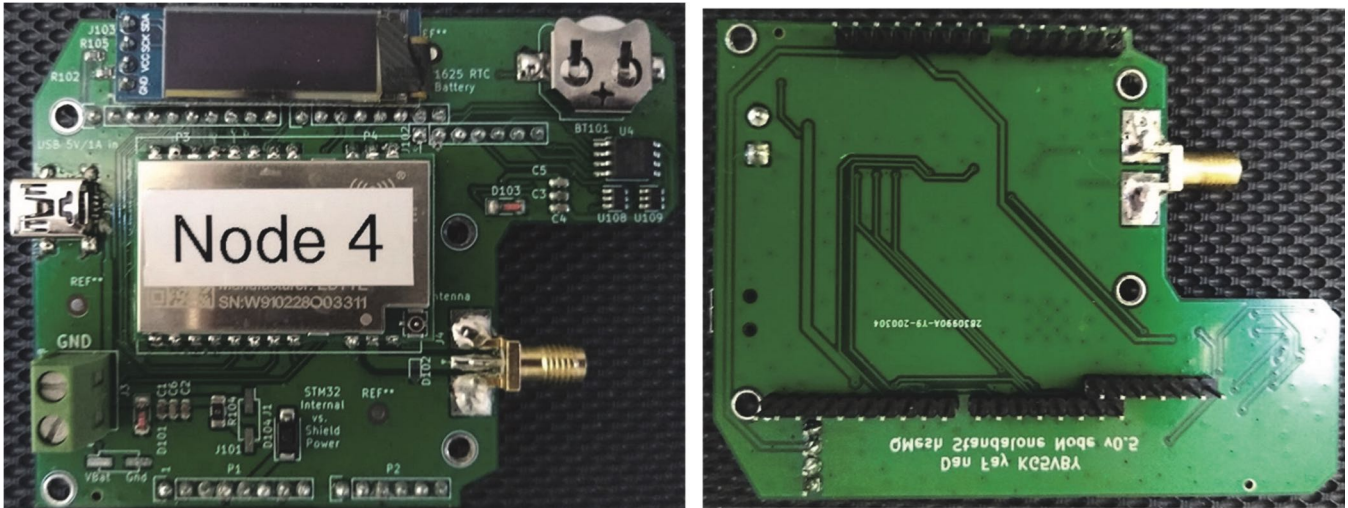


Figure 7: Closeup of the QMesh radio board.

The radio board is a custom design designed to utilize the “Zio” connector on the STM32 NUCLEO-144 boards. It is mostly compatible with the Arduino Shield form factor, save for some additional pins used to connect to the MCU’s QSPI bus. It is designed to power both itself as well as the STM32 board via a 5V source supplied via either screw terminals or a mini-USB-B connector. The combined power consumption for both the radio shield as well as the STM32 board stays below 1A, which allows it to be powered by common USB power sources.

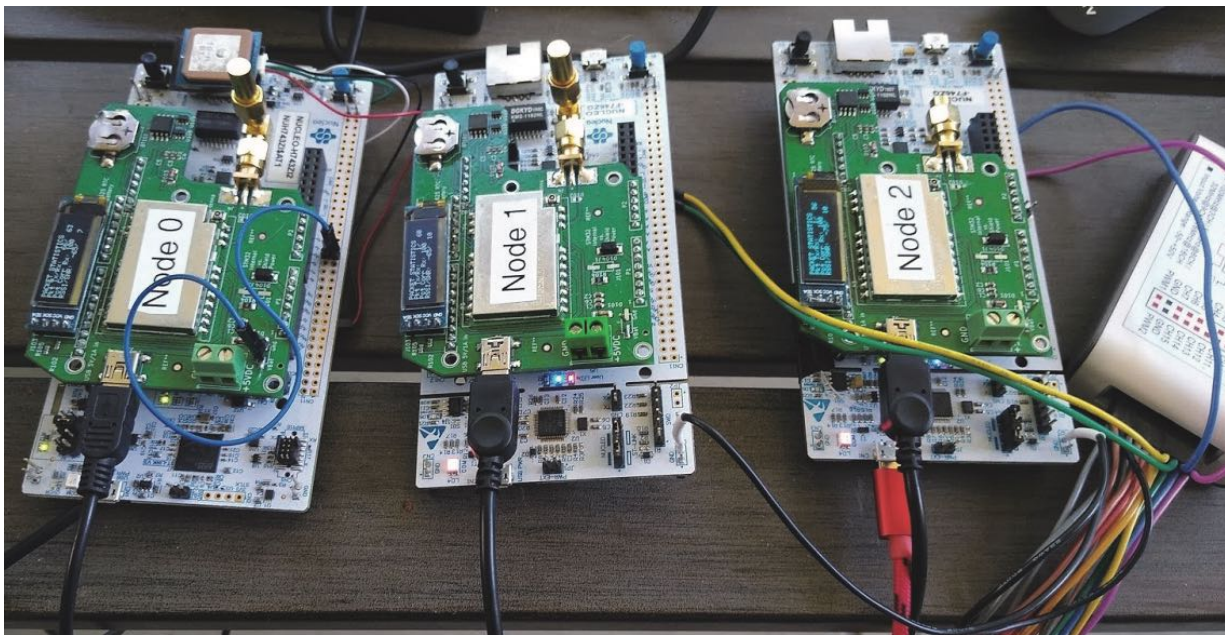


Figure 8: Nucleo-144 with radio shield.

At the heart of QMesh’s radio board is an EByte E22 module. While Chengdu EByte produces several different LoRa modules, these modules (the E22 400M30) have several useful features:

1. **1W transmit power.** This output level is sufficient to serve as an input into “DMR amplifiers”, allowing for higher transmit power if required or desired.

2. **External LNA.** This component can improve receive sensitivity by ~2dB.
3. **TCXO.** A TCXO improves frequency stability and accuracy which facilitates the frequency wobbling feature implemented by QMesh. It also enables the ultra-low-datarate, narrow-bandwidth, high-sensitivity LoRa settings.
4. **SPI interface.** SPI interface enables the precise control of the LoRa modem that QMesh needs to implement its TDMA-based protocol.
5. **33cm and 70cm versions.** Two variants of the module support the 33cm and 70cm amateur bands.

In addition to the LoRa module, the radio board contains a 128Mbit Winbond QSPI NOR flash chip that is used for storing configuration and logging information and a 128x32 monochrome OLED display that can display live information such as packets received, SNR, RSSI, etc. The OLED display uses an SSD1306 controller chip and communicates with the STM32 microcontroller over I2C. Electrostatic discharge (ESD) and overvoltage protection has been added to the board as well through Transient Voltage Suppression (TVS) diodes. The board was designed in KiCAD and manufactured at JLCPCB. JLCPCB is a large board house in Shenzhen, China that can inexpensively manufacture the PCB itself as well as populate many of the board's components.

Approximate Bill of Materials for a QMesh “Standalone” (repeater) Node

Item	Purpose	Vendor	Cost (USD)
NUCLEO-144 STM32H743ZI2	MCU Board	Digi-Key	\$27.00
EByte E22 400M20	1W LoRa Module	AliExpress	\$10.50
Solar charger board	Charge Li-Ion battery, supply node	AliExpress	\$8.99
Solar panel 3W	Power	AliExpress	\$5.44
26650 Li-Ion Cell	Battery storage	AliExpress	\$4.49
26650 Holder	Battery holder	AliExpress	\$0.94
Retevis RHD-771	2m/70cm whip antenna	AliExpress	\$4.71
SMA Connector	SMA port for RF shield	AliExpress	\$0.21
SMA Cable	Connect RF board to antenna connector	AliExpress	\$1.42
SMA Bulkhead Connector	Provide weatherproof antenna connector through case	AliExpress	\$1.00
OLED Display	Live display of information	AliExpress	\$1.43

PCB Mfg. and assembly of RF board	Partially populated, assembled RF shield board	JLCPCB	\$5.82
Apache 1800 Case	Weatherproof Case	Harbor Freight	\$12.99
Total			\$84.94

Experimental Setup

The experimental setup presented here is an attempt to simulate a worst-case scenario for QMesh communications: where identical nodes transmit at under the exact same conditions: same location, same polarization, etc. The setup used here involves placing three 2m/70cm whip antennas a quarter wavelength (at 433MHz) from each other, as shown in Figure 6. This setup allows for testing collisions between two and three identical nodes. The worst-case scenario for QMesh is multiple transmitters transmitting at nearly the same power. To test this situation, a setup with three boards transmitting on three 2m/70cm whip antennas spaced one-quarter wavelength apart was used. Successful communications using this setup should adequately demonstrate the ability of QMesh to successfully capture packets amid large amounts of collisions.



Figure 9: Setup used to test capture success.

Testing multipath effects is highly challenging, as fading can quickly change by even small changes in location. For this early-phase testing, the packet reception was tested by the author walking around his backyard with a node and measuring the packet receive rate. Walking around the backyard should help

provide a variety of multipath fading situations to test the ability of QMesh to withstand interference from two and three nodes simultaneously retransmitting.

Experimental Results

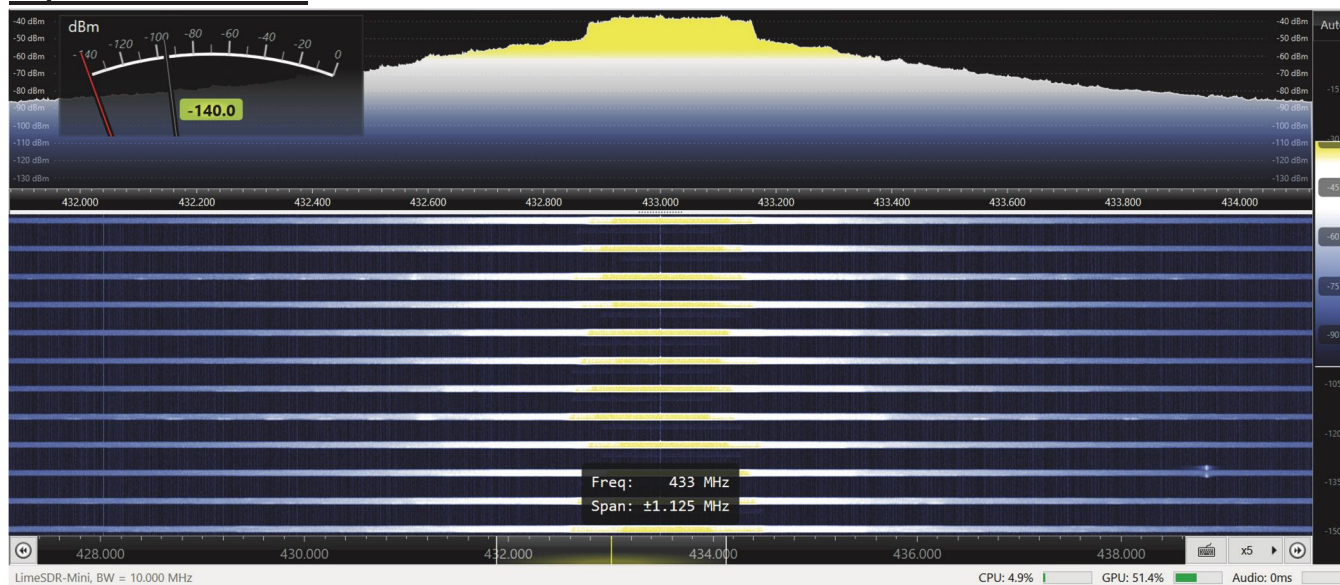


Figure 10: Waterfall of three QMesh nodes retransmitting a signal.

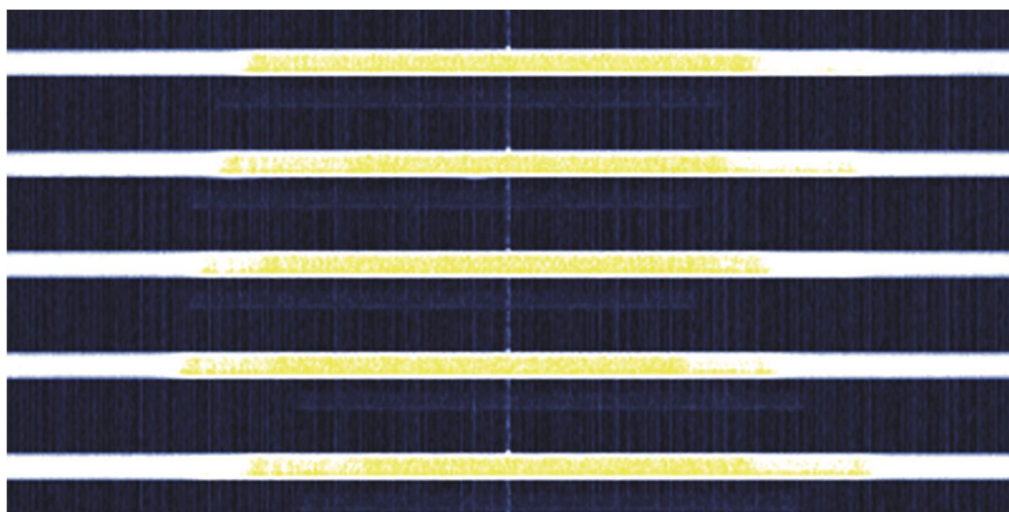


Figure 11: Several transmits enlarged. Visible are the varying transmit power levels as well as the (very faint) originating transmission.

The above figures show the waterfall for three nodes retransmitting. Three things can be observed here. First, the frequency wobbling is visible as a left-right shift in frequency between the chirp packets. Second, the variable transmit power can be seen as well, with the overlapping transmits creating lighter shades of yellow at the edges of the overlapping chirps. Finally, the original transmit is visible as a very faint LoRa packet.

In a single transmitting node configuration, FEC appears to “steepen” the packet error rate curve. Essentially, it causes the LoRa modem to either receive the packet correctly, or not receive anything at all -- there are considerably fewer cases where a received packet is incorrect. It also does appear to

improve the receive floor by about 2-4dB. It is likely that the relatively modest improvement in the receive floor is due to the SX126X's receiver being tuned to only receive packets that are likely to be error-free. Future work will look at using different detection parameters in the SX126X's Channel Activity Detect (CAD) feature¹¹ as a way of lowering the detection floor.

Packet reception appears to work well. When leveraging the combination of FEC, frequency wobbling, time offsets, and variable transmit power, a QMesh node can receive packets at a 99% PRR when two or three nodes are simultaneously retransmitting. Removing the variable transmit power does not appear to have a significant negative effect on the Packet Receive Rate (PRR), with a 99% PRR for both the two and three retransmitter situations. The final test looks at the PRR with no FEC. For this scenario, three nodes leads to a 90% PRR, two nodes a 93% PRR, and one node a PRR of 100%. The non-FEC-using tests strongly suggest that forward error correction is essential and highly beneficial for synchronized flooding scenarios. Without FEC, the PRR drops considerably from 100% to 93% for two repeating nodes and 90% for three repeating nodes.

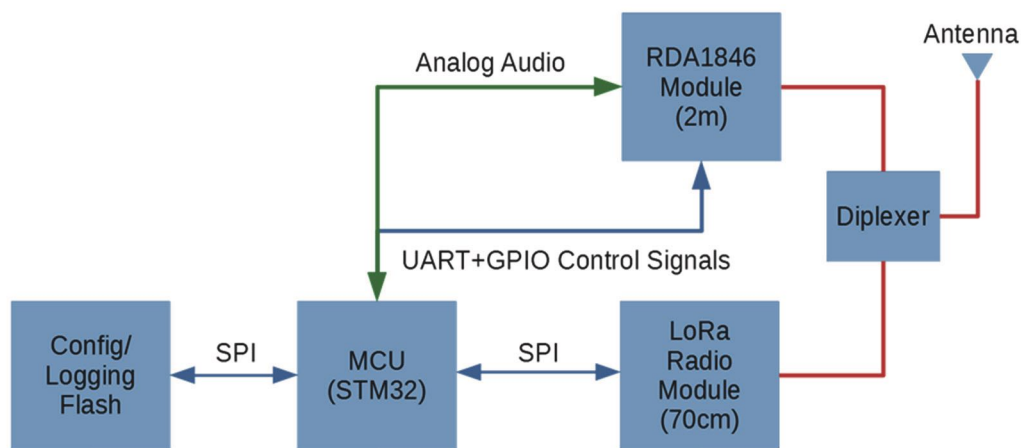


Figure 12: Proposed architecture of QMesh Analog Repeater

Future Work

A medium-term use of QMesh is to develop a series of small repeaters that bridge narrowband analog FM communications with QMesh's digital LoRa mesh. The goal of this micro-repeater is to make QMesh as accessible to radio amateurs as possible by allowing for the use of existing FM radios. FM radios would communicate with QMesh repeaters in the same manner as they would with a standard FM repeater; the QMesh repeater would receive the FM voice, encode it as `codecs2`-encoded voice, send it out digitally over QMesh, and the other repeaters would in turn decode the `codecs2` voice and transmit it out as FM. This capability can also be used to bridge the AFSK-over-FM waveform typically

¹¹ "Application Note: SX126x CAD Performance Evaluation", <https://semtech.my.salesforce.com/sfc/p/#E000000JelG/a/2R000000Q1Ec/b29BFJzTZY420v0tqI3kVC6nzHRQizlFWGjDfcC1Dyc>, accessed August 14, 2020.

used by APRS with the QMesh protocol. This setup is physically compact and low-power, enabling it to serve as a small, solar-powered self-linking repeater.

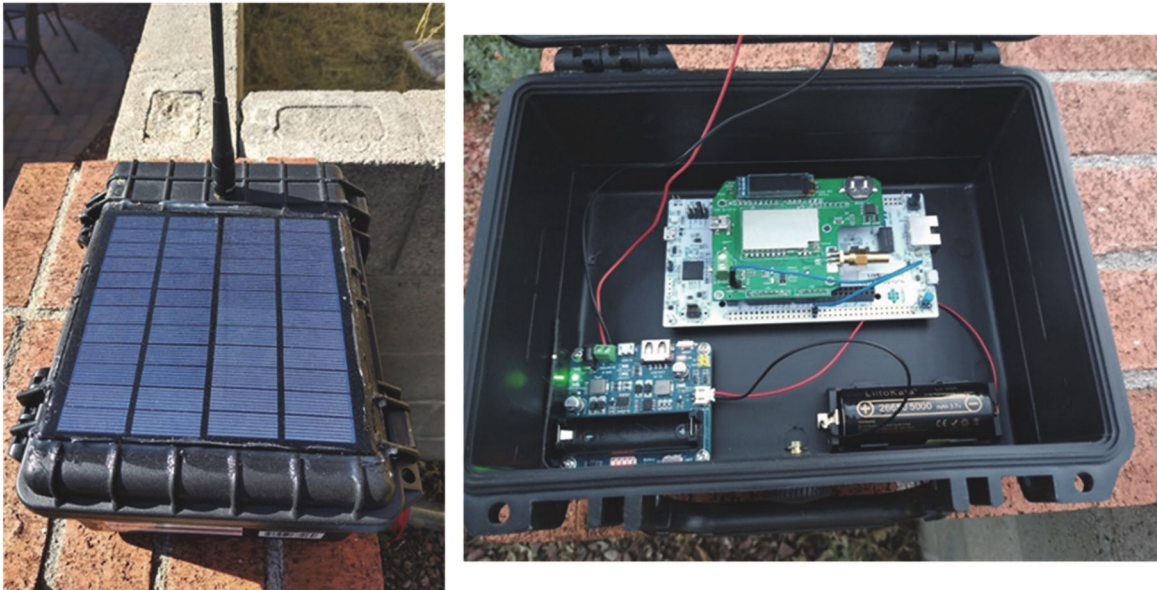


Figure 13: Mockup of a solar-powered QMesh standalone repeater.

Enabling this feature are the existence of compact, low-cost, low-power RDA1846S modules that are essentially a “Baofeng-on-a-module”. The RDA1846/RDA1846S is an SDR-based narrowband FM chipset that contains nearly everything needed to implement a narrowband FM radio. The Dorji DRA818(U/V) is an example of this module, where Dorji combines the RDA1846S with a microcontroller to provide a simple, UART-based interface to command and control the RDA1846S chip. An alternate version with full-duplex FM communication capabilities would use two of the DRA818V modules, with either both on the same band (requiring a duplexer and/or separated antennas), or on 2m/70cm in a cross-band mode. For these setups, QMesh would operate on 33cm instead of 70cm.

One application being explored is to develop trackers for the America’s Cup gas balloon race that occurs every year at Albuquerque’s Balloon Fiesta. These trackers would collect position information via GPS and ultimately forward it to an APRS digipeater or iGate over 2m AFSK. Since there is limited APRS coverage in the interior regions of the United States, QMesh would serve to link the various competing balloons together so that one within range of an APRS digipeater or iGate would forward traffic from other balloons that are not. Using one of the low-datarate LoRa settings, the receive sensitivity should be high enough that some limited beyond-line-of-sight QMesh communications should be possible.

Other future work will examine other improvements to the QMesh protocol, such as channel access (i.e., CSMA-CA vs. ALOHA, etc.). Some physical-layer improvements could be explored, such as developing soft decoding of the LoRa modem, more advanced FEC algorithms such as Low-Density Parity Check (LDPC), and polar codes. Another physical-layer improvement to explore is receive-antenna diversity. It should be straightforward to implement a two-way selection diversity scheme with two QMesh standalone nodes.

Other Related Work

There exist some other projects performing similar goals to the QMesh project. The M17 project¹² intends to replace various digital narrowband standards such as DMR, DStar, and C4FM with a fully open protocol (they all use the proprietary AMBE vocoder), including using the open source `codec2` vocoder. FreeDV¹³ is a project that intends to supplant analog SSB voice with a digital voice protocol that combines `codec2`-vocoded digital voice with a custom OFDM waveform. A key goal is to develop a voice mode that can function at a lower SNR than SSB.

Multiple mesh network projects exist that serve to share APRS-like telemetry information between nodes. The most-successful commercial project is Gotenna Mesh¹⁴. This device works on the 915MHz/868MHz ISM bands to link together small dongles. These dongles interface with a smartphone via Bluetooth to provide location information as well as text messaging/chat. Meshtastic¹⁵ provides similar functionality to Gotenna Mesh but is open source. It is designed to work with various common, open-source LoRa boards. Disaster Radio¹⁶ is similar in concept, although it specifically targets stationary, solar-powered nodes spread throughout an area.

QMesh also draws from ideas used by New Packet Radio (NPR)¹⁷. NPR carries medium-speed IP traffic over 70cm. While not a mesh network, NPR inspired several ideas within QMesh. The first one is to use common, 1W ISM band RF modules for RF communications, instead of attempting to develop a custom waveform or RF board design. Doing so simplifies the design, lowers its cost, and lowers its power consumption as well. Second, QMesh shares the notion that so-called “DMR amplifiers” can be used to provide higher power outputs from these devices if necessary (in practice, a 1W input is sufficient to drive these amplifiers to an output of 20-25W). Finally, NPR demonstrated that STM32 microcontrollers can provide the precise timing needed for TDMA protocols.

Commercial projects which inspired QMesh include Virtual Extension,¹⁸ which uses synchronized flooding for IoT devices, as well as the Bluetooth Mesh standard¹⁹, which uses flooding. Both protocols showed that flooded mesh networks are suitable for mesh network communications. Finally, Haystack²⁰ demonstrated that adding a Forward Error Correction algorithm more sophisticated than the built-in Hamming FEC can provide big performance improvements for LoRa.

Summary and Conclusion

¹² “M17 Project”, <https://m17project.org/>, accessed July 29, 2020.

¹³ “FREEDV: OPEN SOURCE AMATEUR DIGITAL VOICE”, available <https://freedv.org/>, accessed August 15, 2020.

¹⁴ “GoTenna Mesh”, <https://gotennamesh.com/products/mesh>, accessed July 30, 2020.

¹⁵ “What is Meshtastic?”, <https://www.meshtastic.org/>, accessed July 30, 2020.

¹⁶ “Disaster.radio: “, <https://disaster.radio/>, accessed July 30, 2020.

¹⁷ “New Packet Radio”, <https://hackaday.io/project/164092-npr-new-packet-radio>, accessed July 31, 2020.

¹⁸ “VEMesh IoT Products”, <https://virtual-extension.com/products/iot-products/>, accessed July 31, 2020.

¹⁹ “Introducing Bluetooth Mesh Flooding”, <https://www.bluetooth.com/blog/introducing-bluetooth-mesh-networking/>, accessed July 31, 2020.

²⁰ “How to Quadruple the Range of LoRa”, <https://www.haystacktechnologies.com/2019/10/07/xrmode2/>, accessed July 31, 2020.

This paper provided an overview of QMesh, which is a synchronized flooded wireless mesh networking protocol leveraging unique characteristics to allow for successfully receiving colliding packets. It examined several features of QMesh, including the use of sophisticated Forward Error Correction, randomized frequency offsets, and randomized timing offsets. The results presented here demonstrate that a LoRa-based synchronized flooding protocol can achieve high packet receive rates if Forward Error Correction is used.

Finally, this paper discussed the long-term goals of QMesh, which is to create a voice-capable mesh network. It showed a proposed hybrid analog FM/QMesh architecture that is compatible with existing narrowband FM radios, including the AFSK-over-FM waveform commonly used by APRS. This hybrid setup should make QMesh more accessible and useful immediately without requiring people to buy new radios. It also allows for bridging between analog FM repeater systems and/or APRS digipeaters/iGates.

For more information about QMesh, please see the project's Github repository at <https://github.com/faydr/QMesh>. The author can be contacted at daniel.fay@gmail.com and periodically posts about Amateur Radio and QMesh on the Twitter account [@faydrus](https://twitter.com/faydrus).

A Comparison of Affordable, Self-Assembled Software-Defined Radio Receivers Using Quadrature Sampling Down-Conversion

Caleb Froelich
Dr. Rob Frohne, KL7NA
Konrad McClure
Joshua Silver
Jordyn Watkins, KN6FFS

Walla Walla University

ABSTRACT

The software-defined radio (SDR) is a low-cost radio with the potential to draw attention from both experienced and inexperienced operators. As part of a class project, we designed and built quadrature sampling down-conversion SDR receivers. With the receivers we built and the resulting knowledge from that experience, we wanted to contribute to the impact SDR receivers have on growing the amateur radio community. This paper describes the quadrature sampling detector (QSD) in an easy to understand manner, analyzes the advantages and disadvantages of four different SDR receiver designs, and presents a design guide to building an SDR receiver with relevant equations and considerations.

INTRODUCTION

As the future of the world is uncertain amidst the COVID-19 pandemic, the future of amateur radio is equally uncertain. Noting the majority of ham radio operators are in their 60s and 70s and the growth of the number of new amateur radio licenses has held constant at 1% for the past few years, it appears that the pertinent questions are “How do we attract younger operators?” and “How will this younger generation change the course of amateur radio?” Ham radio has already been changed significantly by the introduction of low-cost radios. Handheld VHF/UHF transceivers like those made by Baofeng are affordable and easy to use. And now software-defined radios (SDRs) are bringing that same budget price tag to HF.ⁱⁱ

Low-cost is not the only reason SDRs have become more popular among the amateur radio community. More recently, Guido Ten Dolle’s μ SDX open source transceiver has generated increasing interest in quadrature sampling down-conversion SDRs in the homebrew QRP community. Guido, PE1NNZ, was able to modify the QCX, QRP transceiver for SSB operation with an efficient class-E amplifier, using only an ATMEGA328 and Arduino code to run the QSD SDR. This groundbreaking work in this type of SDR has inspired various renditions of Guido’s radio, fostering a lively groups.io group that can be followed at <https://groups.io/g/ucx>.

For our Electronics II class at Walla Walla University, we designed, built, and tested SDR receivers with the primary goal of furthering our education in electronic engineering concepts while remaining within a \$25 budget. Although we had the privilege of many hours of dedicated class time for researching and designing an SDR receiver, we recognize not everyone has this same

opportunity. We resolved to compose an easy to understand explanation of quadrature sampling detectors (QSDs), (the heart of the type of SDR we all designed), provide a comparison of our SDR receivers with an emphasis on aspects in the designs that enhance or diminish performance, as well as provide a design guide to quadrature sampling down-conversion software-defined receivers with design tools, resources, and tips. This information can help flatten the learning curve in addition to supplementing hobbyist interests in this type of SDR, such as Guido Ten Dolle's μ SDX transceiver.

FUNDAMENTAL INFORMATION

The fundamental ideas pertinent to QSD SDR radios, especially the four presented designs, are the operation of mixers and the use of quadrature signals. This section will cover each in an accessible level of detail.

The goal of the mixer is to sample the input from the antenna for information. To start with a simple analogy, have you ever seen a video where a plane propeller starts up? If so, you have likely noticed it appears to spin in one direction, slow down until stationary, and then spin in the reverse direction. This phenomenon is useful to explain mixers. Imagine we have a propeller that is spinning, and a camera is set up to take snapshots of the propeller as it spins.

As a visual example, **Figure 1** shows a plot that traces only the vertical height of the tip of one blade of the propeller over time as it spins clockwise at a rate of 10 rotations/second. The line traces the true position, while the small images of the propeller blade along it shows what would be seen if an image was taken at that moment. In this case, the camera is taking pictures at a rate 8 times faster than the blade rotation rate.

To continue the example, if pictures are taken at the same time the propeller makes one full rotation, as shown in **Figure 2**, it will appear stationary as if the propeller is not spinning at all.

If pictures are taken at a rate slightly slower than the rotation rate of the propeller, like in **Figure 3**, the images will show the propeller slowly turning in the same direction. Conversely, if pictures were taken faster than the rotation rate of the propeller, like in **Figure 4**, the images would show

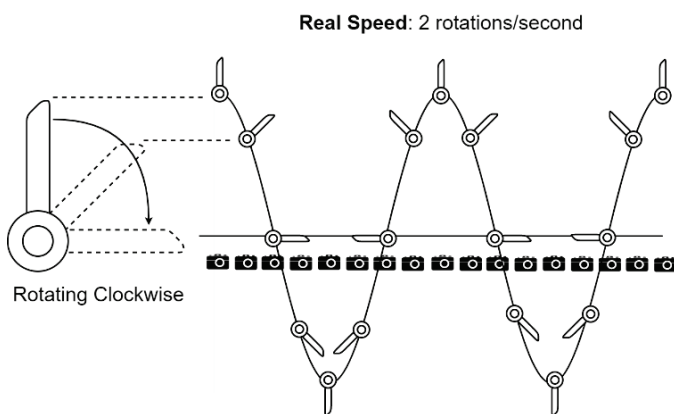


Figure 1. Vertical position of the tip of one propeller blade tracked via pictures taken at 8x the rate of rotation.

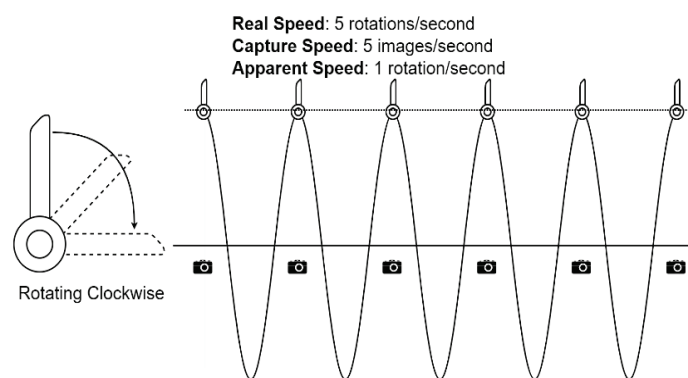


Figure 2. If the images are taken at the same rate as the propeller makes a rotation, the propeller would appear stationary.

the propeller slowly turning in the opposite direction. Interestingly, the trace of the height of the propeller tip is identical despite the propeller apparently turning in reverse.

By taking snapshots you have effectively changed the apparent frequency of rotation to be the difference between the frequency of rotation of the propeller and the frequency of snapshots taken.

As can be seen in the examples above, the frequency of the apparent rotation is equal to $(\text{real speed} - \text{capture speed})$. If the capture speed is faster than the real speed, the difference is negative, indicating the propeller will appear to be turning in the opposite direction. All three cases occur when a video is taken of a propeller, though it is the framerate of the video remaining constant while the propeller increases in speed.

The frequency of rotation of the propeller is like the frequency of the radio signal (“RF signal” with frequency “ f_{RF} ”) you wish to tune to. The frequency of the snapshots is the frequency you can control. This frequency is generated by a clock, known as the local oscillator (“LO signal” with frequency “ f_{LO} ”). This turns our apparent rotation speed equation from $(\text{real speed} - \text{capture speed})$ into $(f_{RF} - f_{LO})$. The local oscillator is used to time the switches that take the “snapshots” used to slow down the frequency of the signal you wish to listen to down to a frequency slow enough for your sound card and computer to process. The classic example of an ideal mixer at first appears to be different than the propeller blade analogy, will output both a difference frequency ($f_{RF} - f_{LO}$) as well as a sum frequency ($f_{RF} + f_{LO}$). In the example to the right in **Figure 5**, a 10 MHz RF signal is mixed with a 9.99 MHz LO signal to produce two frequencies: $(f_{RF} - f_{LO}) = 10\text{kHz}$ and $(f_{RF} + f_{LO}) = 19.99\text{ MHz}$. The slow 10kHz signal is the desired

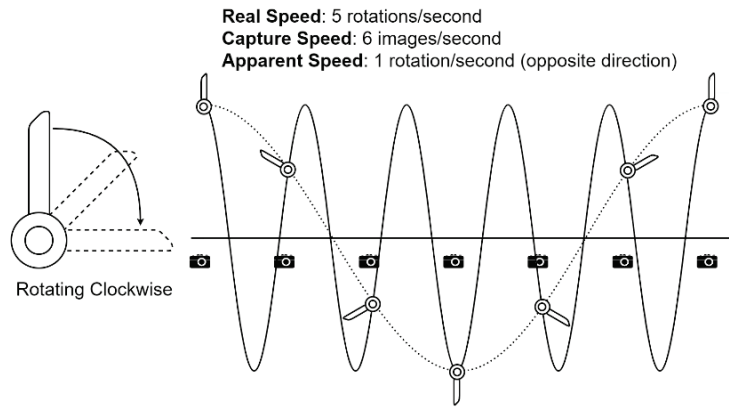


Figure 3. If the images are taken at a faster rate than the propeller spins, the images display a rotation that is slower than, but in the opposite direction of, the propeller's spin.

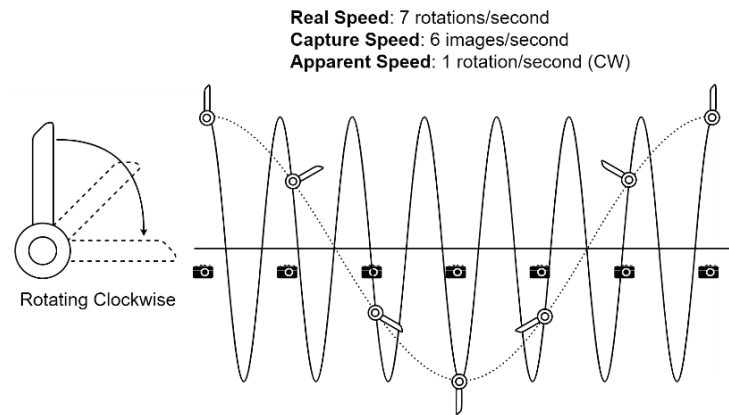


Figure 4 - If images are taken at a slower rate than the propeller spins, the images display a rotation that is slower than, but in the same direction of, the propeller's spin.

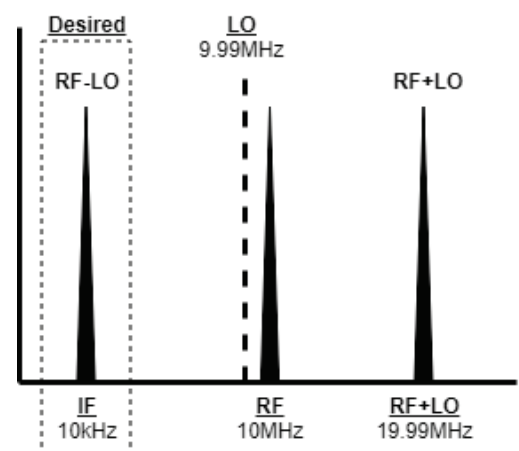


Figure 5. To down-convert a received signal to a slower frequency, the Radio Frequency (RF) and Local Oscillator (LO) frequencies are mixed, resulting in a new Intermittent Frequency (IF)

frequency since the sound card can process it. The 19.99 MHz signal is easily filtered out with a low-pass filter due to its distance from the desired 10kHz frequency. The mixer circuitry used in all the following designs expand on the classic example somewhat by including sampling capacitors on the output, which keep the sampled voltages like the film stores the snapshots of the camera. These capacitors create a low-pass filter that will smooth out the instantaneous points captured into a continuous curve as illustrated in the figures of this section. In the same way, by the nature of the human brain, we assume that between each snapshot is a smooth transition between the two positions, which results in that same continuous “apparent” trace that the low-pass filter creates.

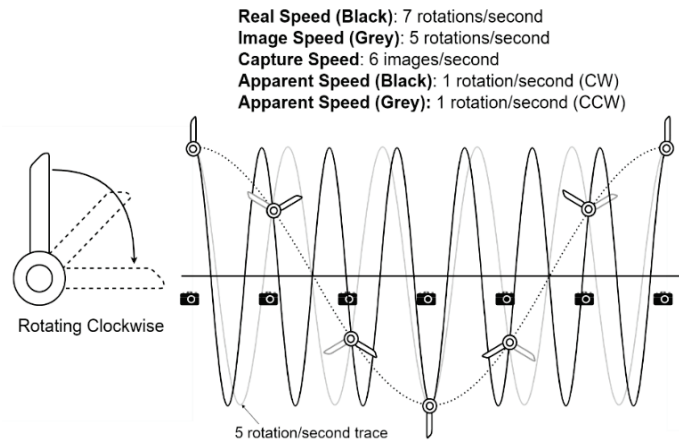


Figure 6. The real (desired) and image frequencies create the same plot on the graph when tracking the vertical position. We have no way to tell the difference with just the vertical position of the blade tip.

A key detail of the slower and faster rate examples is that the apparent rates of the propeller both have the same 1 rotation/second rate and produce the exact same trace on the graph. The only difference between them is the direction of rotation. The two are superimposed together in **Figure 6** with the faster rate in black and the slower rate in grey. Unfortunately, there is a key difference between our analogy of taking pictures of rotating propeller and the reality of radio mixers. We are able to identify the direction of apparent rotation from these graphs through the little images of the propeller along the trace of the vertical position. If we only had a plot of the apparent vertical position, however, it would be impossible to tell which direction it was rotating. The two rates, 5 rot/sec and 7 rot/sec are referred to as “images” of each other.

In terms of radio signals and the previous radio example, our desired RF signal would be the 10MHz signal would be down-converted by the 9.99MHz LO frequency to a positive (“CW”) 10kHz. However, as shown in **Figure 7**, if there was a second frequency broadcasted at 9.98MHz, that is the same distance from LO as our desired RF (this frequency would be referred to as the Image), the Image would also get down-converted to 10kHz IF frequency, except the Image is negative (“CCW”). As a result, we would hear both broadcasts from the speakers.

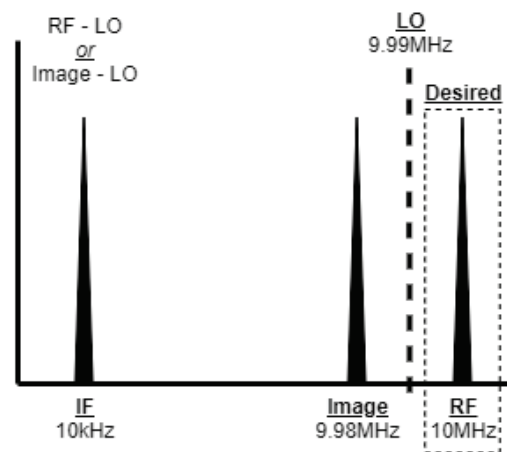


Figure 7. Both the RF and Image frequencies will be down-converted to 10kHz. A method is needed to tell the difference between the positive and negative

The solution in the propeller example is to track both the horizontal and vertical positions of the tip simultaneously, which previously was done artificially by including the small propeller images along the trace. If instead two axes are created to track them simultaneously, the direction of rotation can be determined. **Figure 8** and **Figure 9** display this on the following page.

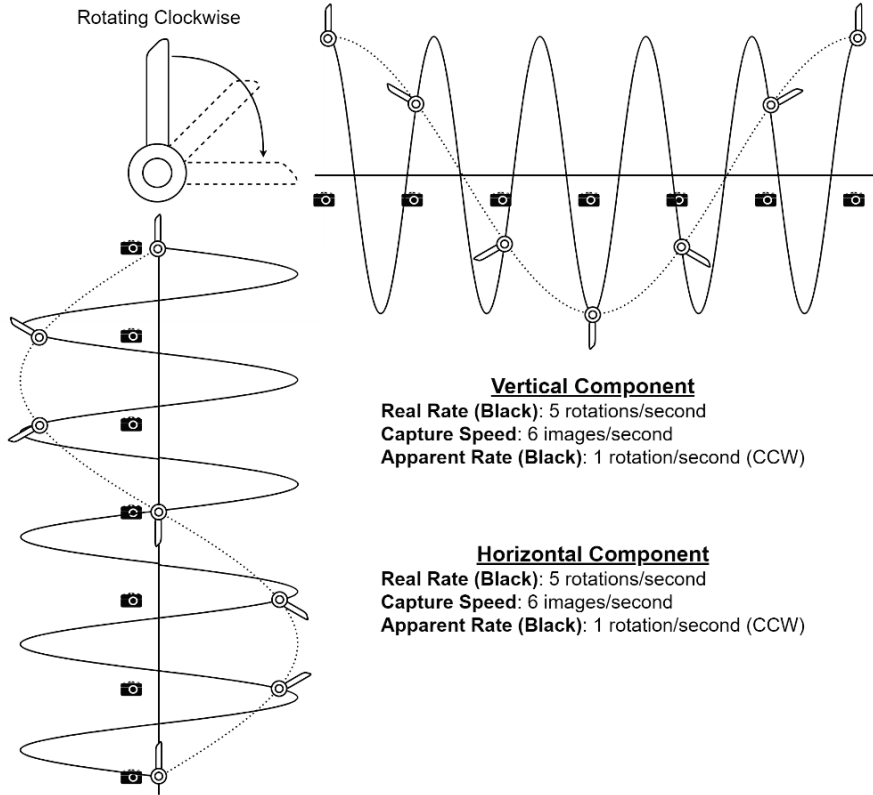


Figure 8. With a rotation rate (RF) slower than the capture rate (LO), the resulting apparent frequency ($IF=RF-LO$) is negative, resulting in a counter-clockwise (CCW) rotation. This is the undesired Image signal

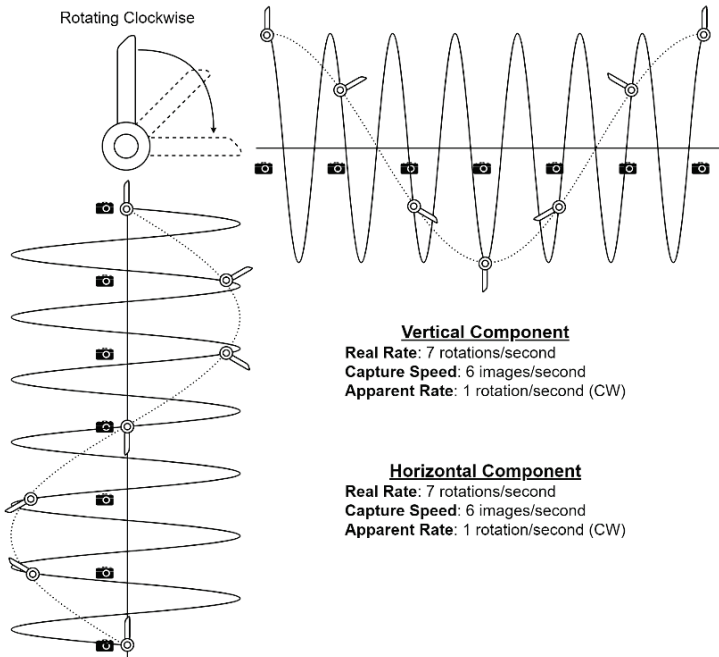


Figure 9. With a rotation rate (RF) faster than the capture rate (LO), the resulting apparent frequency ($IF=RF-LO$) is positive, resulting in a clockwise (CW) rotation. This is the desired RF signal.

With this, we introduce what are known as Quadrature signals. In their most simple explanation, Quadrature signals are signals that are 90° (quarter of a cycle) out of phase with each other.

Examining **Figures 8** and **9**, we can see that **Figure 8** produces a CCW output while **Figure 9** produces a CW output. Looking closer, we see that while the vertical positions trace the same path as previously observed, the horizontal positions are reflected across the axis. Superimposing these two axes from the vertical and horizontal positions, as shown in **Figure 10**, we can easily see how the “clockwise-ness” is determined. On a positive frequency, the propeller reaches the “upmost position” a quarter of a cycle before the “rightmost position.” Conversely, on a negative frequency, the propeller reaches the “rightmost position” a quarter of a cycle before the “upmost position.” Since the vertical trace remains constant no matter the sign of the frequency, it is referred to as the In-Phase (I) signal. It is used as a reference point to compare the horizontal trace to, known as the Quadrature (Q) signal. Another illustration of this is seen in **Figure 11**, where each scenario has been superimposed.

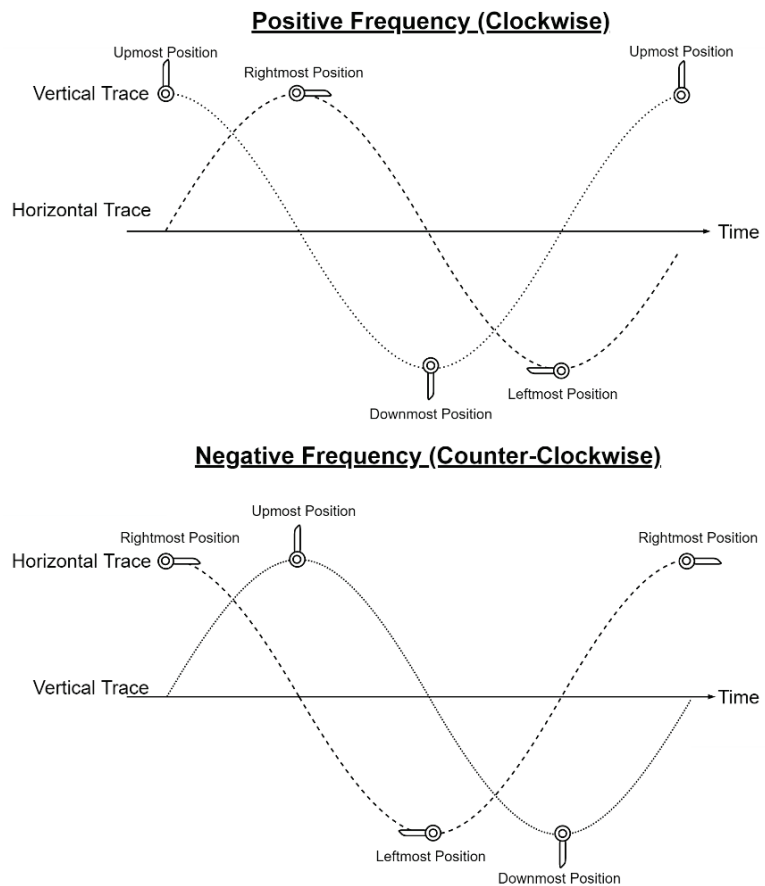


Figure 10. As time goes on, a positive (CW) frequency will reach the upmost, rightmost, downmost, and leftmost positions in that order. A negative frequency will cycle through the rightmost, upmost, leftmost, and downmost positions in that order. This allows us to identify whether the quadrature signals represent a positive or negative frequency.

Back to our example in **Figure 7**, the Image signal at negative 10kHz can now be ignored by the software processing the digital signal from the soundcard if it is provided with *both* the I and Q components of the signal. Often, this can be done with two mixers: one that produces the I signal, and another that produces the Q signal. This kind of configuration can be seen in Design 3 using two voltage-controlled switches as mixers. Another method, as can be seen in Designs 1, 2, and 4, is to use

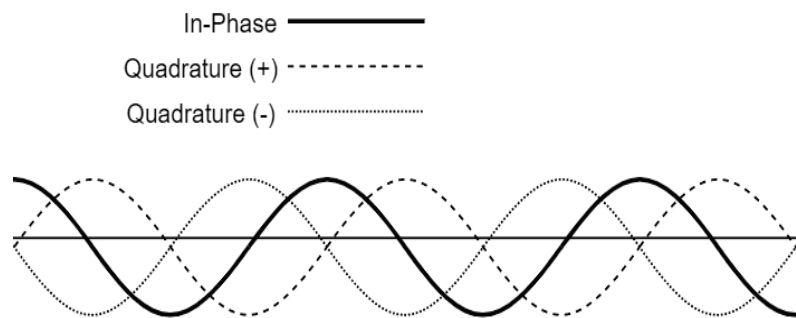


Figure 11. The In-Phase, Negative Quadrature, and Positive Quadrature signals superimposed to show the phase difference between them. Both quadrature signals are measured relative to the constant In-Phase signal.

a Quadrature Sampling Detector such as a Taylor Detector, which converts the input into the I and Q signals directly.

There is one more issue to consider. Returning to the RF equals LO example in **Figure 2** where the propeller appears stationary, consider if the propeller is rotating at twice the rate of the snapshots. It will also appear to be stopped, just as if it had been rotating at the same rate. After some thought, you'd come to realize this is true for any integer multiple (2x, 3x, 4x, ...) of the frequency, as shown in **Figure 12**. Evidently, there are an infinite number of rotation rates that can cause any apparent slow rotation rate you might guess by just looking at the snap shots. The same trace will match any frequency of $f = f_{real} + n \times f_{capture}$ where n is any integer, as can be seen in **Figure 13**. Since any value of n will result in the same down-converted trace on both the vertical and horizontal axis.

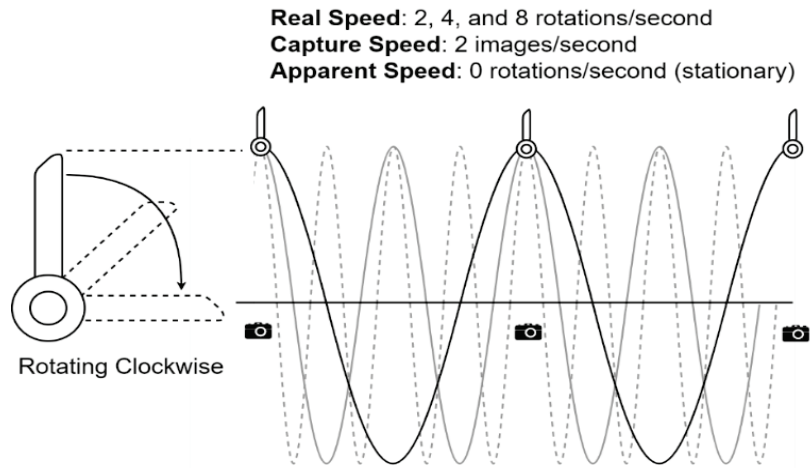


Figure 12 - The propeller appears stationary at $f_{real} + 0 \times f_{capture} = 2 \frac{rot}{sec}$, $f_{real} + 1 \times f_{capture} = 4 \frac{rot}{sec}$, and $f_{real} + 2 \times f_{capture} = 8 \frac{rot}{sec}$

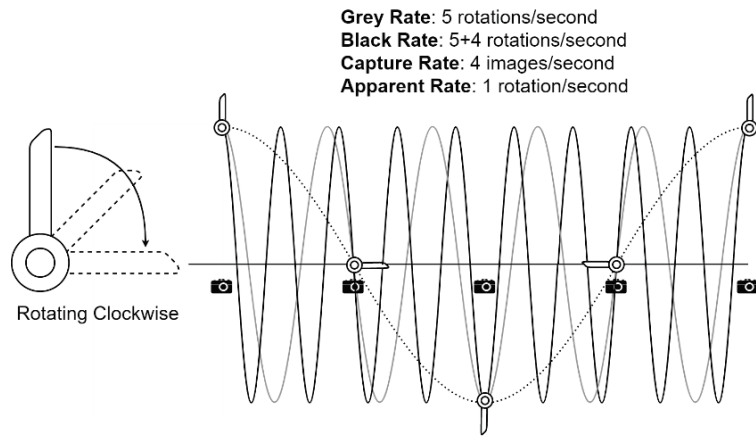


Figure 13 - The apparent rate of the propeller can match $f_{real} + 0 \times f_{capture} = 5 \frac{rot}{sec}$ or $f_{real} + 1 \times f_{capture} = 9 \frac{rot}{sec}$. This will be true for any value of n .

We only want our system to respond to rotation rates very close to the local oscillator frequency (the 5 rot/sec), not to every harmonic of the local oscillator (the 9 rot/sec, etc.). We want to translate the frequencies very near the local oscillator (around 4 rot/sec) down to near DC and keep only those, not the others (near 8 rot/sec, etc.). If there are no

responses to these other frequencies, we can use a low pass filter to smoothly connect the sampling points and keep the desired band of frequencies near the local oscillator, now translated down to DC, and be done. These undesired responses, however, are a problem.

Similar undesired parasitic responses also occur in a radio receiver using instantaneous sampling like in the camera analogy (for both I and Q downconverters). The effect is undesired, because you want to listen to one and only one radio station at once. You want to remove all signals except the desired one near the local oscillator frequency. A key to solving this problem in QSD receivers is realizing the frequency conversion or sampling process is actually weighting (multiplying) the

signal by a time varying periodic function. The frequency of this local oscillator weighting function is just the number of Hertz we wish to shift the signal down. Because the weighting function is periodic, it is made up of a fundamental, and usually has harmonics. In the snapshot method, the weighting function is a series of very high short pulses, one at each shutter time when the photos are taken, and zero in between, where the shutter is closed on the camera. Each harmonic component in this weighting function is every bit as big as its fundamental frequency component. The undesired responses come from the harmonics shifting undesired signals down too, right on top of our desired signal. The ideal weighting function would have no harmonics at all, just a single sinusoid at the local oscillator frequency.

Unfortunately, multiplying by a time varying sinusoid in analog electronics is difficult, so it is common to use switches controlled by the local oscillator clock to create a periodic stepped weighting function instead. A simple example of this kind of weighting function is a square wave. The weighting function is selected to have less harmonic content than the snapshot method, thus reducing the undesired responses. It should be noted that sometimes we cannot create a weighting function with a high enough frequency, and in those cases, a harmonic (instead of the fundamental) can be used to down-convert the desired signal to audio frequencies. To really ensure the undesired RF signals do not get through, they are filtered out with bandpass filters before they reach the down conversion stage. If they are not present, we don't have to worry about them interfering with the desired signal, even if using the "snapshot" weighting function.

A single pole single throw (SPST) switch connected to the RF signal multiplies it by one when closed, and zero when open. A center tapped transformer creates, on the secondary, both plus the RF signal and minus the RF signal. A single pole double throw (SPDT) switch controlled by the local oscillator clock accomplishes the multiplication by the square wave weighting function by selecting alternately one or the other. This method is used in Design 3, as shown in **Figure 14**. (Designs 1 through 4 are described in the next major section.) Sometimes amplifiers are used with the switches instead of transformers to set the levels of the segments of the stepped weighting functions. Designs 1, 2 and 4 use this approach.

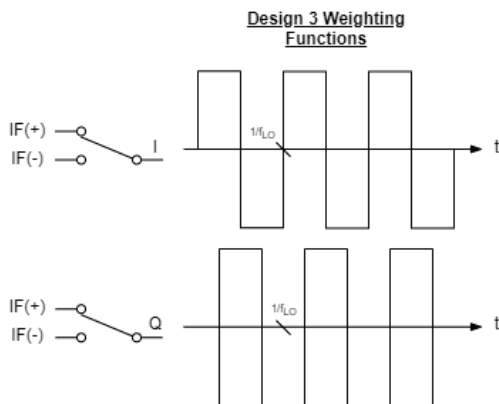


Figure 14 - Weighting function used by Design 3.

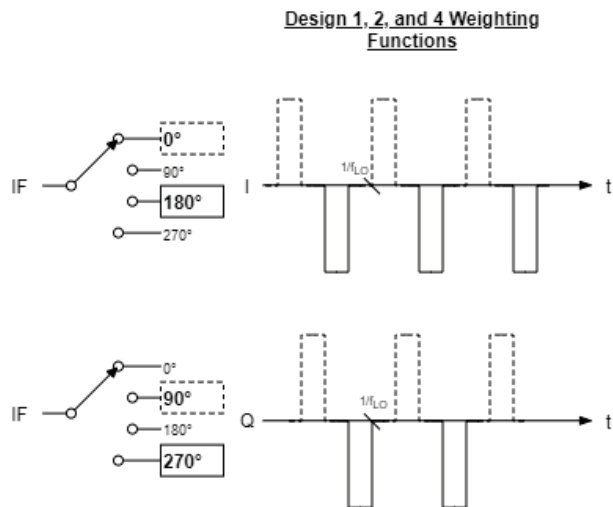


Figure 15 - Weighting function used by Designs 1 and 2. Design 4 is nearly identical, but the positive pulses are slightly taller than the negative ones.

In the figure above, the weighting functions for the simple mixer and the Tayloe down converter are shown. It should be noted that all these waveforms have no even harmonics, and the odd harmonics all fall off as $1/m$, where m is the harmonic number, which means the harmonics are dropped by using them, but not as far as you probably want. To really eliminate the undesired responses, a bandpass filter at the input which only allows the desired band of frequencies in, filtering out the harmonics, is the solution. To ensure only the output connects the smoothest function through the samples and the desired band of low frequencies which was shifted down remain in the result, we filter it with low pass filters, consisting of at least the sampling capacitor, and often further audio filters in subsequent stages.

Designs 1 and 2 use the Tayloe mixer weights shown in **Figure 15**, and Design 3 uses the square wave weighting, while Design 4 uses the Tayloe mixer with a circuit where the 0° and 90° signals are weighted slightly more than the 180° and 270° signals because of the amplifier gains in that circuit. Thus, upon close examination, that circuit would have very slightly higher positive going pulses, but the effect is too small to be visible on this scale.

DESIGN ANALYSIS AND COMPARISONS

We have four SDR receivers from which we will evaluate and compare the designs and draw conclusions. One was designed by Caleb Froelich and Konrad McClure (Design 1), a second by Joshua Silver and Jordyn Watkins (Design 2), and the third and fourth radios were designed by Rob Frohne (Design 3, Design 4).

We explore the differences between our radios with a focus on the aspects in the design that affect performance. We evaluate each of the circuits in the receiver designs, specifically focusing on the following: bandpass filter, local oscillator, mixer, amplifiers, and low-pass filter.

BANDPASS FILTER (BPF)

Bandpass filters are widely used in radio receivers as they allow the designer to change the frequency agility of the radio receiver. Design 1 utilizes a fixed 3rd order Bessel LC bandpass filter. By using a series-first topology, a coupling capacitor can be saved. While a design with a fixed bandpass filter is easier to make, maintain, and fit on a small board, it doesn't offer the flexibility and performance that switchable bandpass filters allow radio enthusiasts.

Designs 2 and 3 both utilize switchable bandpass filters, controlled via two digital output pins on an Arduino Nano v3. With careful planning, the switchable bandpass filters better eliminate spurious signals as discussed above and reduce the overall noise. This increase in performance comes at the price of design complexity. With the introduction of multiple BPFs, the need for a switching circuit (a 4:1 multiplexer is a perfect solution) and software to control the filters are needed. Additional bandpass filters also add more passive components to the board, taking up precious PCB space.

For a designer who has a fixed frequency band with which he would like to communicate, a single bandpass filter is more than adequate. The pure minimalist could even neglect using a bandpass filter. Excluding the BPF exchanges the inclusion of a few spurious signals for a receiver that is

light, compact, cheap, and easy to assemble and debug. Design 4 takes on a composite approach, placing pin headers for an optional bandpass filter to be installed post-facto.

LOCAL OSCILLATOR

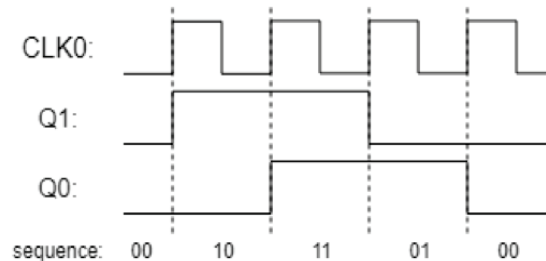
The clock generator and local oscillator (LO) configuration is homogeneous for the first three designs. A Si5351a clock oscillator is used to generate the LO signal and dual flip-flops connect as a divide-by-four Johnson counter to produce the I and Q local oscillator signals at one quarter the frequency of the driving clock. This approach offers a large bandwidth, as the LO can vary from the lower limit of the Si5351a of (2.5 kHz)/4 to the maximum frequency of (200 MHz)/4.

Following the design of the μ SDX transceiver by Guido Ten Dolle, Design 4 capitalizes on the capabilities of the Si5351a to generate the I and Q signals without the need of flip-flops. Programmable via the I²C protocol, the Si5351a allows you to lock the phase of the two clock outputs, to generate two signals at the same frequency and in quadrature. The Si5351a does, unfortunately, place a limitation on the range of frequencies that can be generated. Utilizing the Arduino Etherkit Si5351 library, written by Jason Milldrum, NT7S, the minimum quadrature frequency that can be generated should theoretically be $600 \text{ MHz}/128 = 4.6875 \text{ MHz}$, because the phase register is 7 bits and the minimum PLL frequency is 600 MHz. However, through direct I²C programming, Guido works the minimum quadrature frequency down to 3.2 MHz, fully encompassing the popular 80-meter band (3.5 MHz to 4.0 MHz).ⁱⁱⁱ

If the minimum frequency of the quadrature method is not low enough for the desired application, there is an alternative solution. By utilizing two clocks, a sequential counting sequence (00, 01, 10, 11) can be obtained by setting the low order bit to change at two times the desired local oscillator frequency. The higher bit is then set to be half the frequency of the low bit using a divider built into the Si5351a that keeps the bits synchronized. A timing diagram comparing the counting method to the quadrature signals generated via a Johnson counter is shown in **Figure 16**. Interestingly, generating quadrature signals directly from the Si5351a increases image rejection. The quadrature method utilized in Design 4 provides 40 dB of image rejection, and Guido gets 45 dB. Experimentation and testing showed that the Johnson counter LO in Design 2 has 30-35 dB of image rejection, while the well-known SoftRock RXTX we tested only has 25-30 dB depending on the part of the passband you are in. Bottom line: use the quadrature method for frequencies above 3.2 MHz and the counting method if frequencies below 3.2 MHz are desired. Both

Quadrature LO utilizing Johnson Counter*

- * used in Designs 1, 2, and 3.
- CLK0 is 4x the desired LO freq.
- 2 memory elements required.
- Si5351 can generate this signals without flip-flops.



LO utilizing counting method*

- * no memory elements necessary.
- CLK0 is 2x the desired LO freq.

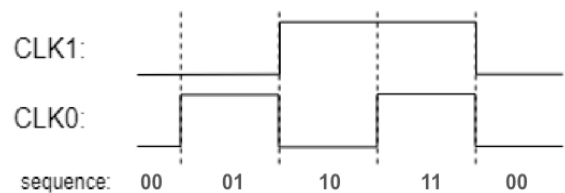


Figure 16. Square wave visual representation of how the counting sequence differs between the two types of local oscillator configurations.

approaches will result in more software than utilizing a Johnson counter, but they cheaper solution and save board space.

MIXER

An important consideration in the design of SDR receivers is selection of a mixer topology that combines the signal of interest and the local oscillator output. When evaluating mixers, it is imperative that one compare the conversion losses, or the ratio of the RF input power to the IF SSB output power, expressed in positive dB^{iv}. The lower the conversion losses, the more efficient a mixer is at converting energy from the RF to the IF signal. Noise is also a concern and should be considered when designing a mixer.

While there are numerous types of mixers, we utilized two common designs. Design 3 utilizes a doubly balanced sampling mixer, essentially a switch with a capacitor feeding into a summing amplifier. If the amplifier selected has a high impedance input, the capacitor will hold the input voltage when the switch is open. This allows the duty cycle of the switch to be lowered, which results in a lower conversion loss. Design 3 does not have this advantage as the amplifier configuration utilized has a low impedance input, as we'll elaborate on in the next section. The mixer in Design 3 is doubly balanced due to the center tapped transformer that negates the RF input signal for half the period.

An extension of the sampling mixer design well known among radio amateurs is the Tayloe detector. Despite its striking similarities with the sampling mixer, Dan Tayloe emphatically states that his detector is not a mixer since it only produces the difference frequency. Perhaps that means Design 3 uses a sampling detector too. All but one of the designs presented in this paper utilized the Tayloe detector due to its compact, simple design and low conversion loss (less than 1 dB). In a comparison of switch-based frequency converters by Michiel Soer, it was concluded that the “a double balanced Tayloe mixer with 25% duty cycle provides the best balance between noise figure and conversion loss”^v. The low-cost, high-performance Tayloe detector is one of the better options for SDR receiver designs.

AMPLIFIERS

While the first two designs (Designs 1 and 2) share many fundamental similarities: (they both utilize bandpass filters and a Tayloe (I/Q) mixer to convert the radio spectrum to a low IF which is then demodulated via software) the key difference between the two lies in the amplification of the IF signals. Design 1 utilizes instrumentation amplifiers while Design 2 utilizes a basic differential amplifier. The biggest difference between these two amplifiers is the input impedance. The instrumentation amplifiers provide a very high input impedance which makes the amplifier more like a voltage amplifier, while the differential amplifier more closely resembles a trans-resistance amplifier due to its lowered input impedance. The input impedance is a byproduct of the type of amplifier topology. As seen in **Figure 17**, the instrumentation amplifier used in Design 1 internally utilizes a non-inverting topology which completely isolates the input signal from the output. On the other hand, the inverting topology of the differential amplifiers allows for current to sneak through the feedback loop. This difference in amplifier topology, and subsequently the input impedance of the amplifier introduces a multitude of intriguing effects that influence other aspects of SDR receiver design.

For one, the difference in input impedance affects the way that the mixer sampling capacitor functions. With the high input impedance of the non-inverting topology, the sampling capacitor follows the RF signal when the switch is closed and holds the voltage level when the switch is open, since there is no path for the capacitor to discharge. However, the inverting amplifier with its lower input impedance doesn't charge the sampling capacitor as much.

A voltage amplifier with high input impedance also uncouples the amplifier stage from other sections of the design, discretizing the SDR receiver into a system of easily understood blocks. The many interdependencies experienced with a coupled amplifier design (like Design 2) are eliminated, allowing for easier design. Debugging is also easier in a voltage amplifier like Design 1. The tools that we have at our disposal (oscilloscope, multimeter, etc.) measure voltage much easier than current. Since each designer shared the same primary goal of furthering their

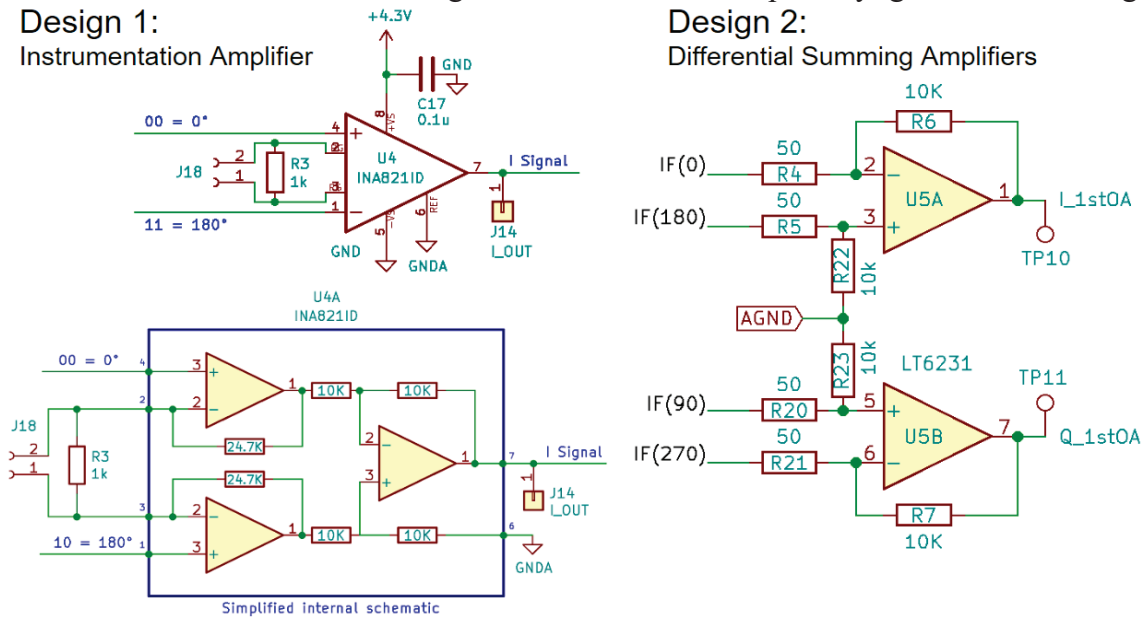


Figure 17 - Comparison between amplifiers used in Designs 1 and 2.

understanding of electronic engineering concepts, the simplicity and understanding that the instrumentation amplifier provides cannot be understated.

Additionally, instrumentation amplifiers remove the ambiguity of antenna impedance from the amplifier gain equation. Design 1 utilized an INA821ID instrumentation amplifier. Since a single resistor sets the gain of this amplifier, it would be trivial to provide two or more gain settings through a relay or switch. This would allow for further optimization and customization of the radio receiver.

The primary advantage of the differential summing op amp is that it can possibly provide a lower noise figure especially at low gain settings, but for HF the atmospheric noise is large enough that if you pick low noise parts, your design should be fine. The voltage noise and noise figure of instrumentation amps increase with decreasing gain^{vi}. Because of this, instrumentation amps are less attractive for low gain situations below about 20 dB, though the differential summing amplifiers have similar problems. The price and availability can make differential summing op amps a more appealing option for a Ham radio amateur on a low budget. The single-circuit

INA821IDs utilized in Design 1 were 6% more expensive than the dual-circuit LT6231s utilized in Design 2.

Despite their dissimilarities, both the differential summing amplifier (utilized in Design 2) and the instrumentation amplifier (utilized in Design 1) take the difference between the IF signals (v_{0° and v_{180° , v_{90° and v_{270°). In the case where a specific linear combination of the signals is desired (i.e. $v_{0^\circ} - v_{180^\circ}$) and the other combinations are undesirable, then a differential amplifier or instrumentation amplifier is optimal. Differential amps magnify the difference signal and reject the common mode signals through clever balancing of the circuit. But in our case, either v_{0° or v_{180° or any other linear combination of these signals can be used. Dan Tayloe's quadrature detector design exploits this fact by eliminating all the input and bias resistors and maintaining just the feedback resistors of the amplifier. This results in the non-inverting path gain not equaling the inverting path gain, and the amplifier does not sum differentially. A side effect of straying from the classic differential amplifier is a larger common mode gain, however in this case the common mode gain is one, just perfect, so the amplifier in Design 4 doesn't need to be DC biased like the amplifiers in designs 1 and 2. This produces a more simplistic design with less parts, reduced power loss and a decreased input noise level.

Design 4 follows Tayloe's approach for amplification yet maintains the input resistors before both inverting outputs so that the gain of the amplifiers is not solely dependent on antenna/system resistance. Resistors before the non-inverting capacitors (R18 and R19) are included in some designs to make the system more symmetrical. However, they have no effect on the operation of the amplifier, other than adding a little noise, so the present plan is to replace them with wire. The two capacitors on each feedback path allow for slightly different cutoff frequencies and eliminate the need of a separate low-pass filter. Two filters are necessary in Design 4, with the bandwidth set at about 6 kHz, as all the digital signal processing (DSP) can optionally be done on the board itself and without the horsepower of a PC using Guido's great code. This allows the sampling to be conducted at a slower pace so that the processor has more time to filter the signals, but no spectral display is possible using only the MCU.

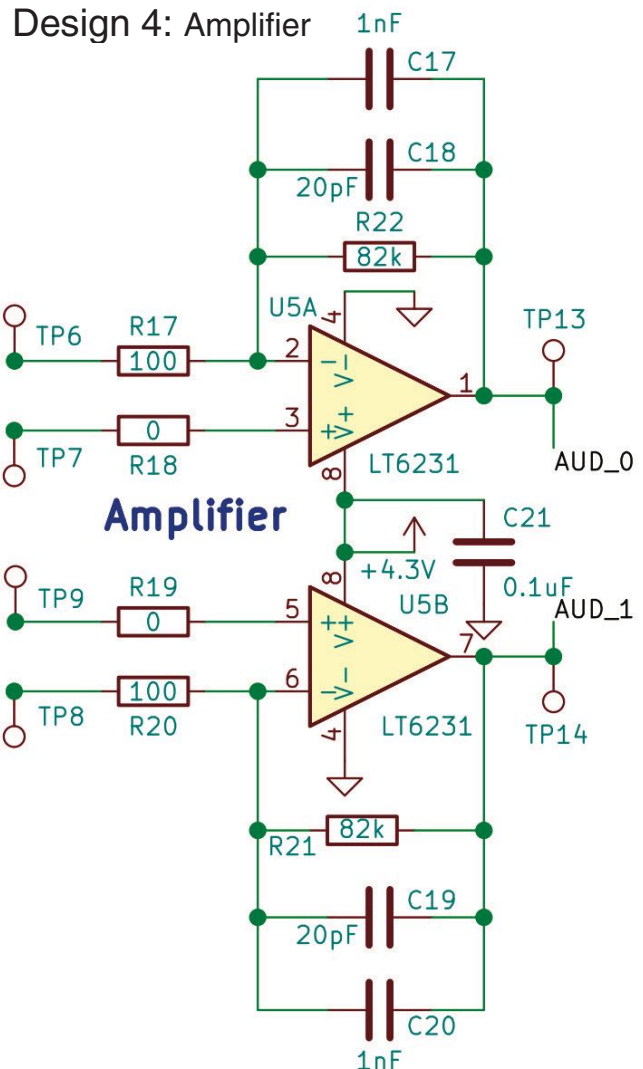


Figure 18. Amplifier used in Design 4 which follows Guido's uSDX design.

LOW PASS FILTER

Separate low-pass filters are utilized in the first two designs. Design 2 uses the low-pass filter to distribute the overall gain of the receiver across two different amplifier stages. The differential summing amplifiers have a theoretical gain of 46 dB (200 V/V) and the low-pass filters contribute another 20 dB (10 V/V) of gain. By placing more gain in the preamplifier stage before the signals get too big, the amplifiers in the low-pass filter don't have to be as robust. Design 2 could have avoided the use of a low-pass filter altogether by simply placing a capacitor in parallel with the feedback resistors of the summing amplifiers, however, the 2nd order Butterworth filter boasts an extra pole of attenuation over Designs 3 and 4. Design 1 must use a secondary low-pass filter since the negative feedback loop occurs internally inside the INA821's. Some instrumentation amplifiers can be purchased with access to the feedback loop; however, these amplifiers are far too expensive for a low-budget radio project. Design 1 utilizes an LPF with no gain since the gain is already controlled by instrumentation amps. Additionally, the noise figure for the INA821s gets better with increasing gain, thus, minimizing the gain on the low pass filter helps improve overall system noise.

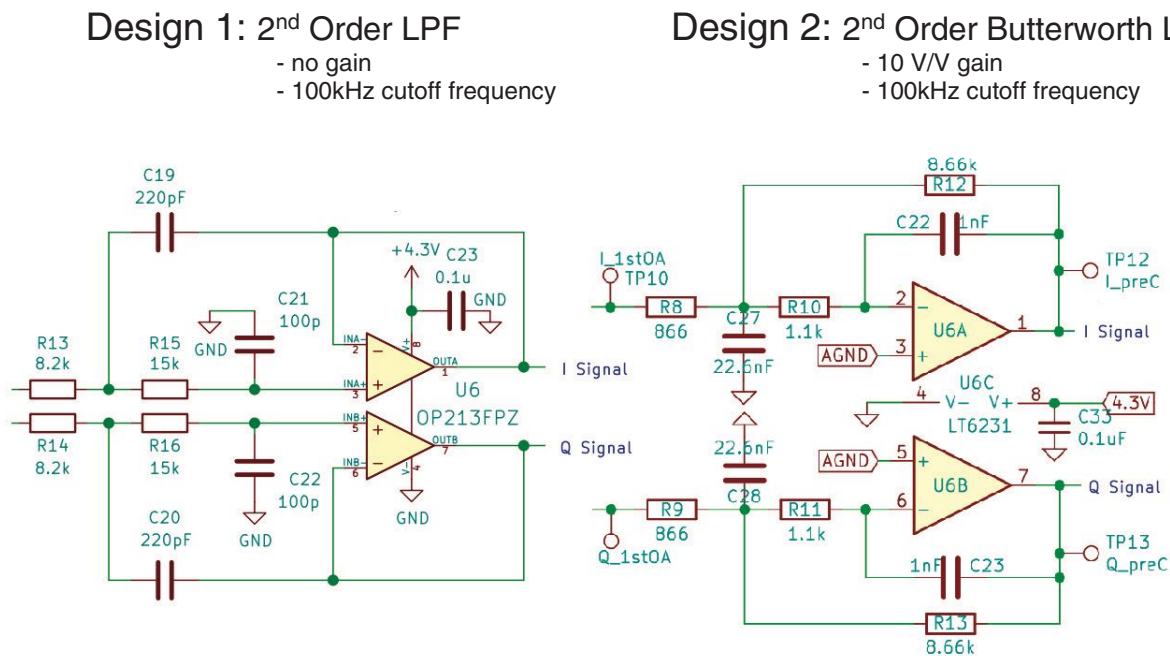


Figure 19 - Comparison of 2nd order filters used in Designs 1 and 2.

OTHER DESIGN NOTES

Ground loop noise is minimized in Design 2 using a transformer to isolate the radio antenna from the receiver. Experimentation and testing found that the noise level was reduced by nearly 20dB (from -90dB down to -110dB) thanks to the inclusion of the transformer. The transformer is directly responsible for removing ground loops.

Grounding problems can arise when there are multiple different paths to ground. A ground loop is formed when there are two ground connections between a component and the ground plane. This loop acts as a single-turn inductor and can infuse substantial noise into the audio signal, corrupting the signal. Since SDR radios utilize extremely sensitive audio cards, it is imperative to take

precautions to avoid the devastating effects of ground loops (see the design guide for more details)

vii

Maintaining a clean power source is also important towards minimizing noise. When powered via a USB, the SDR can experience a lot of noise due to the rapid switching that happens in computers. The solution utilized in all the designs is a common emitter (CE) amplifier that regulates and smooths the output voltage at the emitter. In the first three designs, an RC low pass filter with a cutoff frequency of 16 Hz is used. Design 4 utilizes an LC low pass filter with a cutoff frequency of 90 Hz. Even though this is a second order filter, it did not provide appreciable improvement in the noise levels, whereas the RC network in Design 3 did. It is not recommended to copy this circuit as it was not carefully designed. The only downside to the CE amplifier is it comes with an approximate 0.7V drop across the transistor. Depending on the IC's selected in the design, this 0.7V loss could be significant. Including a bypass jumper allows flexibility to test operation on 4.3V or 5V to determine which is preferred.

	MDS (μ V)	Image Rejection(dB)	Power Consumption (Watts)	Cost (\$)	Size (cm x cm)
Design 1	0.5	30	—	\$30	10.0 x 10.0
Design 2	0.5	30	—	\$25	10.0 x 10.0
Design 3	0.1	15	$5V \cdot 0.027A$	\$25	5 x 10
Design 4	0.5	33-40	$5V \cdot 0.035A$	\$25	5 x 10
SoftRock RXTX	0.2	25-30	$12V \cdot 0.028A$	\$75	6.35 x 12.7

Table 1. Quantitative Design Comparison performance summary.

DESIGN GUIDE

After spending many hours researching quadrature sampling down-conversion receivers, we decided that creating a condensed design guide to creating a QSD SDR receiver would be beneficial to the amateur radio community and draw interest from a less experienced audience that may have otherwise become lost in the process. The QSD receiver is best for simplicity, power consumption, cost and size. However, if better image rejection and the ability to tune many receivers with the same SDR are desired however, a digital down conversion (DDC) receiver is more appropriate.

The first thing to consider when designing this type of SDR receiver is, “What is your goal for building one?” Is your aim to experiment with electronics? Have a working radio for cheap? Take the radio backpacking? Is there a specific range of frequencies you are hoping to listen to? Do you want it to be as small and lightweight as possible? Or are you shooting for building the highest quality radio possible? These are all questions you should ask yourself beforehand, as the answers can greatly affect your design choices.

In this design guide, we try to break down the design process into a step by step manual. At each step, we provide tables with several design choice options and explanations of when using each option may be appropriate based on your goal with your radio. We also provide relevant equations, links to online design tools, and general board construction and testing tips we found helpful when designing our own radios. We specifically address the documentation and design preparation,

hardware design, circuit simulation, board layout and build, and software with regards to software-defined radio design and construction.

DOCUMENTATION AND DESIGN PREPARATION

The documentation and design preparation are foundational for any project. Keeping careful documentation is important because it can not only help remind yourself of the design and test procedures conducted, but it can also be useful to others who are looking at your project for reference. Documentation can take many forms, from paper and pencil to a GitHub repository. Keeping careful record of corrections needed, mistakes made, concepts learned and more can be of great benefit in the long run. We found Kicad’s^{viii} schematic capture tool, EESchema, was a real help in documentation and design.

Design Planning: When designing electronics, it is helpful to create a block diagram as shown in **Figure 20**. Each block can then be designed separately. In addition to designing the project in segments, try to anticipate problems and incorporate possible solutions in your design. In concordance with anticipating issues, make trouble shooting the issues easier by adding many test points and pins for bypassing entire sections. Being able to identify the problem can sometimes be even more challenging than resolving it.

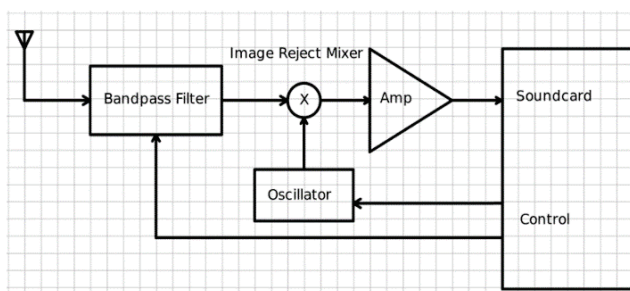


Figure 20. Example Block Diagram

Circuit Simulation: It is crucial to confirm that the design will work using a simulation software before building a prototype. For our receivers, LTspice was used to simulate each component. This will not ensure that the final design will work. However, this will prevent many mistakes that could have been made. It is recommended to start simulating with “ideal components” as this will speed up runtime and makes simulating easier. Once a design is complete and you are ready to pick parts then you can simulate real-world components to find parts that will work for your application. The circuit simulation is discussed in depth in a later section.

Component Selection: There are multiple options when selecting components to use on your circuit board. One of the more important things to keep in mind is to select components you are comfortable soldering. Many individual components (resistors, capacitors, ICs) come in different packages. A summary of the two primary types of packages described in **Table 2**.

TYPE	DESCRIPTION
SMD/SMT	Surface Mount Device/Technology is smaller and is generally soldered onto the surface of the board using solder paste. An oven reflows (melts) the paste which then cools and hardens onto the solder pad. If you do not have access to an industrial reflow oven, a \$20 toaster oven can work just fine. These components tend to behave more ideally than their THT counterparts because they have no leads. Leads have unwanted parasitic inductance and capacitance. Another advantage is this package type can be cheaper than THT. Furthermore, if you

	have the equipment and steady hands, it is easier and faster to assemble boards using this package type. However, we recommend not using anything smaller than the 0805 package size, especially if you are inexperienced placing components by hand with tweezers.
THT/THD	Through-Hole Technology/Device components, an older technology, are generally larger and the pins of the device stick through the circuit board which can then be secured using a standard soldering iron. This is by far the simpler method for those with little to no experience. It is also cheaper to begin with as you do not need to purchase any additional equipment to attach these components. If you anticipate needing to switch out component values, designing plug-in sockets for THT components can serve as a potential solution.

Table 2. Comparison of component package types.

It is also important to choose components that are not obsolete and are easier to find. The simplest way to do this is to go to your favorite electronics part's website (Mouser or DigiKey work great) and sort the parts you are looking for by availability. Manufacturers will stock the parts most used in industry. When in doubt, check the datasheets for the components to confirm the specs. It's also important to choose these parts with high availability as the lead-time to order parts not in stock can take at least 6-8 weeks.

HARDWARE DESIGN

General Good Hardware Design Practices: Use a bypass capacitor at every IC and transistor. A bypass capacitor is a capacitor from the DC power pin to ground. It needs to be as close as possible to the IC or transistor it is bypassing. Often 0.1uF ceramic capacitors are used. The purpose is to ensure that the DC power supplies look like short circuits for AC currents. If they are not used the parasitic impedance of conductors connecting them to the actual DC source cause problems the designers were not anticipating, because they designed as if DC sources were true AC shorts.

Bandpass Filter Selection: As discussed previously in the design comparisons, there are several options when considering using bandpass filters. The type of bandpass filter selected is dependent on the type of radio you desire to build. The first question to consider when selecting a bandpass filter is the preferred filter complexity. **Table 3** summarizes reasons for selecting to omit the use of a bandpass filter, use a single fixed filter, frequency specific filter, or opt for a switchable bandpass filter.

FILTER COMPLEXITY	PURPOSE
No Bandpass Filter	Desired if the user prioritizes simplicity over a clearer signal, or perhaps wants to add an external filter later.
Single Fixed Bandpass Filter	Ideal for receivers where the user knows in advance a specific range of frequencies they wish to listen to.
Switchable Bandpass Filter	Preferable for users who wish to eliminate as many spurious signals as possible while maintaining the ability to listen to a large range of frequencies. One octave (the lower cutoff frequency is half the upper) is a good design choice when using multiple filters.

Frequency Specific Filter	If certain signals are known to be problematic, filtering out only the problematic signals such as a local AM station may be adequate.
---------------------------	----------------------------------------------------------------------------------------------------------------------------------------

Table 3. Different bandpass filter complexity selections and suggested application for selection each type.

After determining the complexity of the filter, the next step is to decide the *type* of filter. LC filters are the best choice for most HF radio applications, so a list of different types of LC filter configurations are listed in **Table 4** along with their advantages and disadvantages. It should assist in determining which type of filter is appropriate for your application.

The final few things to consider when designing the bandpass filters are selecting the frequency range, input and output impedance, the order, and choosing a series-first or shunt-first configuration. Once these characteristics are determined, the next step is to calculate the inductor and capacitor values. These values can be easily calculated through an online calculator such as that at RF Tools (<https://rf-tools.com/lc-filter/>).

The center frequency and bandwidth are selected based on your interest. Note that the wider the bandwidth, the lower the quality. Likewise, a narrow bandwidth increases quality. It is also important to match impedances. When the input impedance does not match the antenna and the output impedance does not match the load, the filter will not have the desired response. When matching impedances, the input impedance does not necessarily have to equal the output impedance. Rather, the filter tools give us the freedom to choose different input and output impedances. This is advantageous when designing the filter because we can have the input impedance equal to that of the antenna, then choose the output impedance to match that of the remaining circuitry, specifically the impedance of the amplifiers. For example, if we were using an instrumentation amplifier which has a theoretical input impedance of infinity, it would be appropriate for us to choose an output impedance of 10M Ω and input impedance of 50 Ω for our bandpass filter. The order of the filter determines the effectiveness of the filter. The higher the order, the steeper the roll-off, and therefore the more effective the filter. However, as the order is increased, the magnitude by which the effectiveness is increased decreases, because nominal filter component values are never exact. Typically, we find a sweet spot around the 3rd order. Lastly, one can choose between series-first (T) or shunt-first (Pi) configurations. If you are using 50 Ω and 10 M Ω as in the previous example, shunt is preferable over series because the series filter will require component values that are very hard to make or procure.

FILTER TYPE	ADVANTAGE	DISADVANTAGE
Butterworth	Very flat response within passband with essentially no ripple. Low level of overshoot. Linear. Tolerant of component variation.	Reaches ultimate roll-off rate more slowly thus the performance around the cut-off frequency is poor.
Chebyshev	Provides faster transition from passband to stop-band. Steep roll-off provides significant attenuation of unwanted out of band spurious signals and limiting harmonics.	This fast transition comes at the cost of in-band ripple. Less tolerant of component variation.

Elliptic	Produces the fastest transition from passband to stop band of any type of filter.	Presence of gain ripple in both passband and stop-band.
Bessel	Very flat group or linear phase delay. Therefore, it is ideal for applications where the wave shape of signals within the passband must be preserved. No overshoot. Linear.	Slower transition from passband to stopband than other filters in this table, of the same order (slow cut-off).

Table 4. Comparison between the different types of bandpass filter topologies and their advantages and disadvantages ^{ix}.

With the design of the bandpass filters complete, the last task is to gather components. You should strive to get inductor and capacitor values as close to the calculated ones as possible or simulate the filter in LTspice or similar circuit simulator to visualize the effect of variations in component values. Some components, such as bypass capacitors, are not very sensitive to the value, but filter components are. For the capacitors, surface mount capacitors work slightly better, because of their low parasitics, but really either works perfectly fine for most HF filters. The inductors can either be purchased pre-wound as a through hole or surface mount part (caution: make sure that it is rated for the appropriate frequencies), or you can make them yourself using powdered iron toroids and wire. Some inductors are lossy. These are less desirable, and if used, you need to simulate the losses, and make sure you are okay with the effects. There are several online winding calculators, two that we've found to work well are:

- <http://www.66pacific.com/calculators/toroid-coil-winding-calculator.aspx>
- <https://toroids.info>

Hand-wound transformers provide a high-quality solution by minimizing losses. However, they are not the cheapest option, and take time, effort, and testing to ensure proper operation. Because winding inductors by hand tends to be time consuming and tedious, generally we try to select bandpass filters with central frequencies and bandwidths that enable us to limit the number of inductors needed in a circuit and keep their values low.

An important note when doing the board layout for the bandpass filters is to recognize the influence the components may have on each other. Generally, iron powder or ferrite toroid cores will keep their magnetic field relatively close, allowing the designer to place the components close to each other without any issues. However, if the designer chooses to use an air core, then mutual inductance may become an issue. It's also good practice to add a surplus of test points, with at least one before the filter and one after for each band. For further ease of testing, we recommend connecting the bandpass filter to the rest of the circuit using jumpers. This way, the user can easily bypass the bandpass filters for troubleshooting purposes.

Ground-Loop Avoidance: Simple solutions can often be used to solve the ground loop problem. Design 1 places a jumper wire from a component to the ground plane that helps break ground loops. While this is simpler and cheaper than winding a transformer, it's not as universal. Jumper wires can be finicky and require readjusting when the receiver is moved to a new QTH. We recommend using a 1:1 transformer between ground and the antenna. This solution is reliable and rugged in mitigating ground loops. If hand-winding a transformer using wire and a toroid core, the

following equation can be used to calculate the inductance of the transformer given an impedance of 50 Ω:

$$wL_m \gg R_s \quad \Rightarrow \quad 2\pi f L_m = 50\Omega(20) \quad \Rightarrow \quad L_m = \frac{50\Omega(20)}{2\pi f}$$

With the inductance calculated, you can now use an online calculator like the ones listed previously to determine the number of windings. Another method used to help avoid ground loops is to have a single ground reference plane on the circuit board and use flood fill with ground on both sides of the printed circuit board. Some radios may not need any ground loop mitigation. For example, when using a battery connected very close to the PCB, with a self contained SDR like Guido's or Design 4. In general, if there is more than one connection to ground, then watch out for ground loops.

Voltage Smoothing: If you use one of these circuits to remove noise coming from the computer's USB port, use the one from Design 3, rather than Design 4. The latest one was not carefully designed and did not improve the noise, while the voltage smoother from Design 3 did in a significant way.

Local Oscillator: There are various options for local oscillators, but all four designs from this paper used the same chip, the Si5351A. As mentioned previously, three different configurations were used and are summarized in **Table 5**.

TYPE	PARTS REQUIRED	COUNTING ORDER	FREQUENCY RANGE
Johnson Counter Circuit	2 (Dual D Flip-Flops, Si5351a)	00, 01, 11, 10	2.5/4 kHz to 200/4 MHz
Counting Method	1 (Si5351a)	00, 01, 10, 11	< 3.2 MHz
Quadrature Method	1 (Si5351a)	00, 01, 10, 11	> 3.2 MHz

Table 5. A comparison of different local oscillator configurations and important characteristics.

When constructing the local oscillator of the SDR receiver, be aware of the voltages of different devices. For instance, the Si5351a is inherently a 3.3V device whereas an Arduino Nano is a 5V device. We need to be careful when connecting different voltage devices. We can connect devices with different logic levels by the use of a pull-up resistor to the lower voltage device. In the case of this example, we would have pull-up resistors to 3.3V as recommended in the Si5351a datasheet. However, in our experimentation, we found this method inconsistent. A safer approach would be to implement the use of a level shifter such as that used by Etherkit. The schematic by Etherkit is shown in **Figure 21^x** where the pull up resistor and MOSFETS that are part of the level shifter are connected to the SCL and SDA pins of the Si5351a clock generator. More information about level shifters can be found at:

- https://etherkit.github.io/si5351abb_landing_page
- <https://www.nxp.com/docs/en/application-note/AN10441.pdf>
- <https://www.digikey.com/product-detail/en/diodes-incorporated/2N7002DW-7-F/2N7002DW-FDICT-ND/750003>

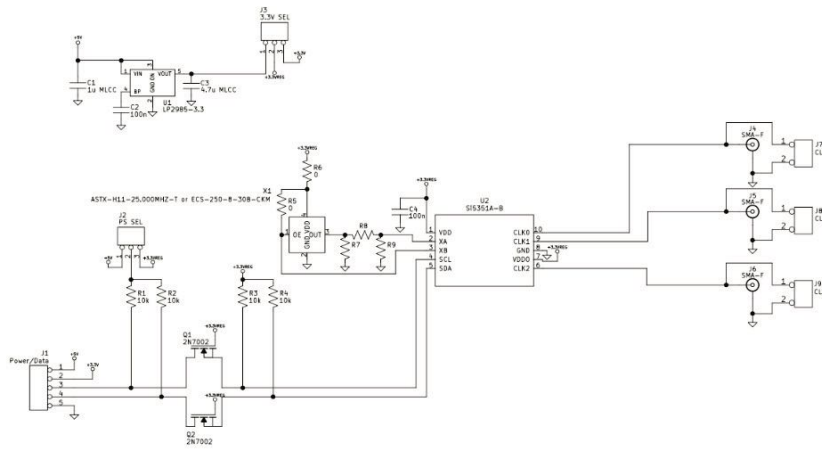


Figure 21 - Etherkit Schematic of the Si5351a with level shifter. ^x

Mixer Selection: Based on the discussion on mixers described in an earlier section, we recommend you use a Tayloe Mixer. An SN74CBT3253 or a similar IC can be used as the 4:1 multiplexer. The schematic for the Tayloe Mixer will typically take the form shown in **Figure 22^{xi}**. The sampling capacitors, as part of the Tayloe Mixer configuration, can be selected based on the following equation:

$$C = \frac{1}{n \times \pi \times f_{ReceiverBandwidth} \times R}$$

R is the antenna impedance plus any other impedances in series before the sampling capacitor, n is the number of capacitors being charged, and $f_{ReceiverBandwidth}$ is selected by the designer. The sampling capacitors essentially become a simple RC low-pass filter when analyzed with the impedance, R .

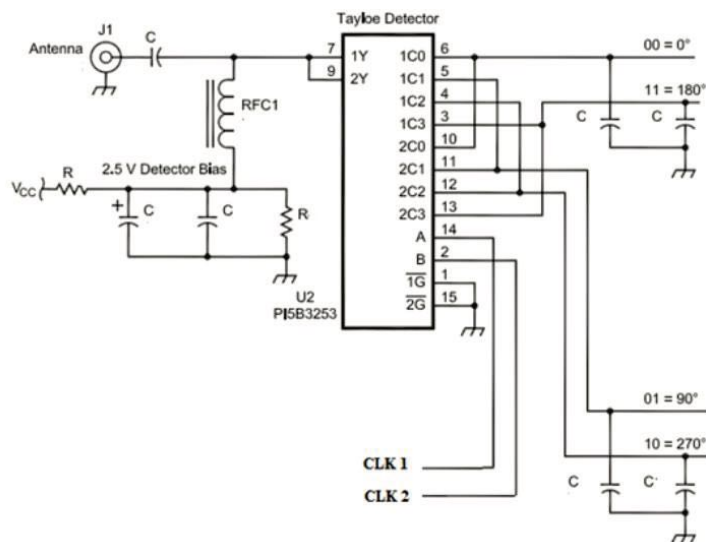


Figure 22 - Tayloe Mixer Schematic with CLK 1 and CLK 2 as select line inputs to 4:1 multiplexer and sampling capacitors, C , on the 00, 11, 01, and 10 outputs of the multiplexer. The RF signal comes directly from the antenna to 1Y and 2Y. ^{xi}

Amplifiers and Low Pass Filter Selection: In order to design the amplifiers and LPFs properly a few things must be considered. Op-Amps can be expensive, so the model chosen has to balance cost and efficiency. The best way to do that is to compare the datasheets of the amplifiers to the specifications required. From the designs shown in this paper, a common IC that is used in Design 2 is the LT6321, chosen for its low noise performance and built-in model for LTspice simulations. The INA821 instrumentation amps used in Design 1 have the benefit of low noise, effectively infinite impedance on their inputs, and gain set by a single resistor, but are quite costly. The instrumentation amplifiers appear to be the simplest to analyze since they act purely as a voltage amplifier whereas the differential amplifiers act as trans-resistance amplifiers. The amount of gain on the amplifiers implemented post-mixer should strive to keep the signal to noise ratio within reason for the given application while also taking into account the sensitivity of the ADC of the soundcard to minimum voltages. Choosing an amplifier that has a pre-made or built-in spice model can be very useful for quality simulation. If there are no pre-made models for one you've selected, you can use the universal op amp model, and fill in the appropriate specifications for the one selected. The NE5532 and SA5532 are op amps with not quite the noise performance of the LT6321, but a much lower price. They can be had in the same package as the LT6321, so could be substituted for a less expensive receiver, especially for the lower bands using the same bands with the same PCB.

For low-pass filters, we utilized the online Filter Design Tool from Texas Instruments: <https://webench.ti.com/filter-design-tool/filter-type>. This saves time as you can get a complete filter design in minutes if you know the parameters needed. More in-depth discussion on the amplifiers and low-pass filters are presented earlier in this paper in the Design Comparison section.

CIRCUIT SIMULATION

Simulating a circuit before building the hardware, as mentioned briefly before, can save the designer many hours of troubleshooting. Often, simulation illuminates potential issues that weren't easily visible before. When an issue is identified in simulation, it is much easier to change the simulation component than it is to solder and de-solder a component from a physical board. In this section, we show several circuit examples with advice on how to simulate them properly. Images are included here, but the LTspice .asc files are available for download at http://fweb.wallawalla.edu/~frohiro/ClassHandouts/?dir=Electronics/LTspice_simulations_of_QS_D_SDR_Receivers.

LTspice Introduction: We primarily used LTspice for our circuit simulations. LTspice is a “high performance SPICE simulation software, schematic capture and waveform viewer with enhancements and models for easing the simulation of analog circuits.” It can be downloaded for free at <https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html> and runs on Linux, OSX and Windows. LTspice also has many readily available help resources including help forums and instructional YouTube videos.

Circuit Simulation Examples: The portions of the SDR receiver circuit that are essential to simulate beforehand are shown with some examples here. Figures are paired with descriptions of what each circuit simulates and how to use appropriate *run* commands for each.

The bandpass filter hardware design can easily be simulated in LTspice using an AC analysis, though the design tools also give simulated results. **Figure 24** demonstrates the topology of a Bessel 3rd order bandpass filter generated through the RF tools website. When simulating the bandpass filter, it is important to include the input and output impedances. Here, the output impedance was assumed to be a 50Ω load as shown by R12. The input 50Ω impedance was included intrinsically in the voltage source V6. To implement an ac sweep, first select a small signal amplitude. For this example, the amplitude was set as 0.5mV. We selected a decade sweep

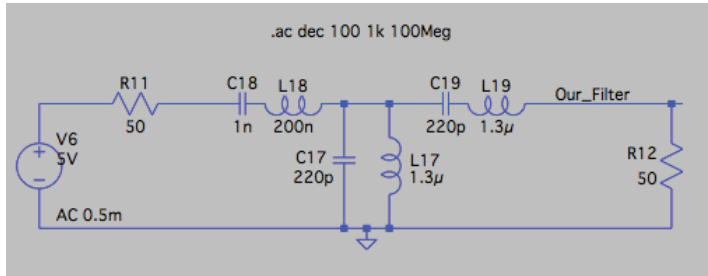


Figure24. LTspice Bandpass Filter Simulation

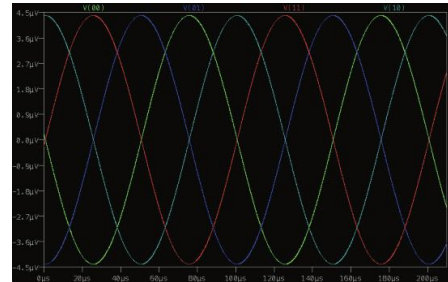


Figure 23. Bandpass Filter Simulation Results

with 100 points per decade spanning a frequency range from 1kHz to 100MHz. The results of running this simulation are shown in **Figure 23**.

Simulating the mixer in LTspice can potentially save hours of time in the long run. For this example, we will show you the simulation schematic for a Tayloe Mixer. Since LTspice is an analog circuit simulation tool, the 4:1 multiplexer used in the circuit must be illustrated by voltage-controlled switches. Recall that the Tayloe Mixer uses a local oscillator to drive the select pins of the FST5351 or similar multiplexer to “mix” the local oscillator frequency with the RF from the antenna. We use voltage-controlled switches and specific signals for the voltage sources to simulate this “mixing” phenomenon. There are two primary ways of specifying the signals in the voltage sources for the voltage-controlled switches.

The first way is by using the PULSE voltage source style. **Figure 26** shows the LTspice edit menu for inputting the settings for the pulse waveform. The pulse waveform is used to generate a square wave at 4 times the switching period to turn the voltage-controlled switch on when the wave is above a specified voltage. We use 4 separate switches controlled by 4 separate pulse waveforms.

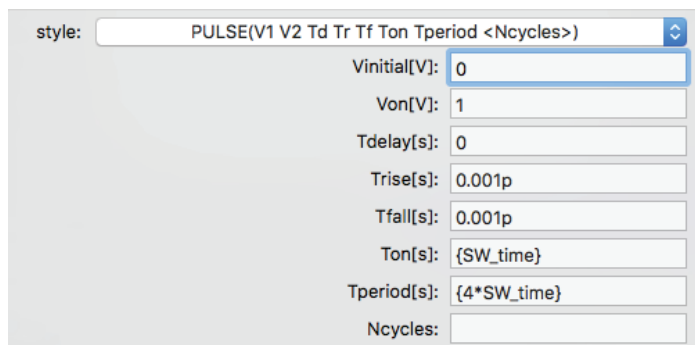


Figure 26. Settings for the PULSE wave 0-degree voltage control

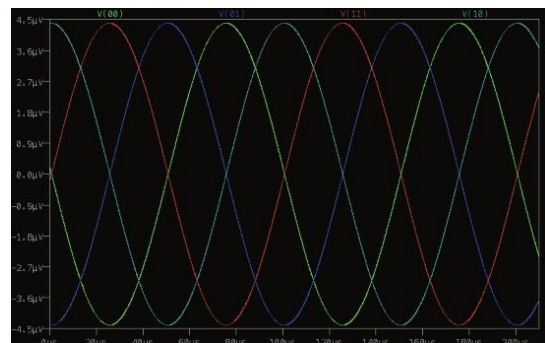


Figure 23. Output of the Tayloe Mixer Simulation with no load attached.

The only difference between each pulse wave is the delay time ($T_{delay}[s]$). This delay is essential in generating 4 signals that are out each 90 degrees out of phase with each other.

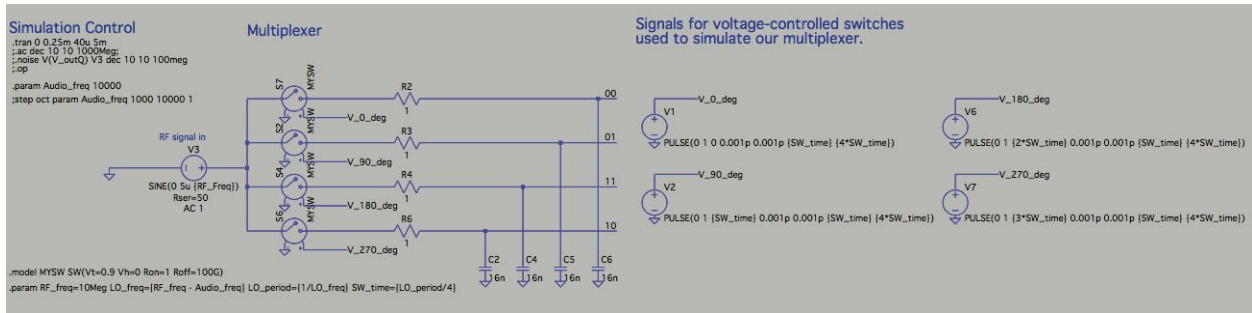


Figure 27. Taylor mixer LTSpice simulation utilizing voltage controlled switches with a PULSE wave control.

The second way is by using four voltage sources as sine waves that are phase shifted 90 degrees from each other. We phase shift them by editing the sine function, as shown in **Figure 29**, such that Φ is the degree (0, 90, 180, or 270), the frequency is 10kHz less than the RF signal phase, and the amplitude is at least 50V. Another important thing to note is that in order for this circuit to work as a multiplexer, we must change the switch model. The model must be adjusted such that Vt is the appropriate value, as shown in **Figure 28**. The rest of the circuit topology is the same as the previous illustration.

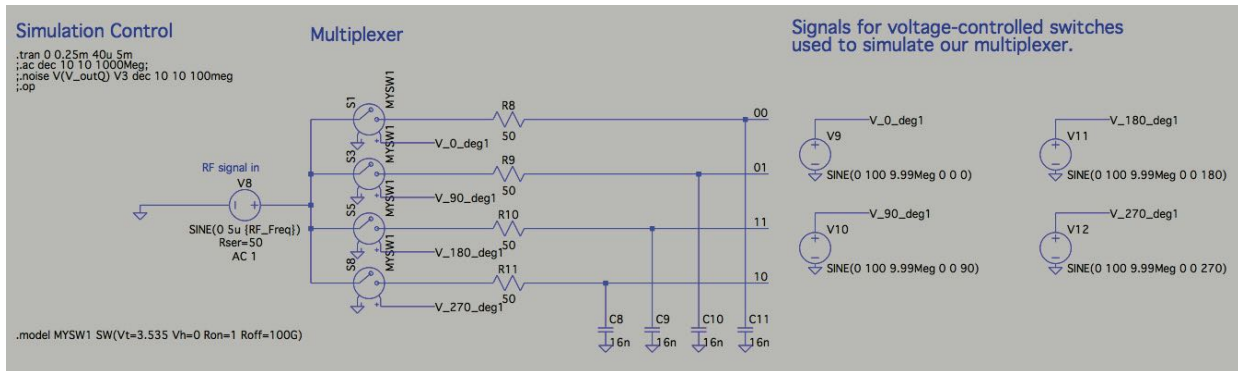


Figure 28. Taylor Mixer LTSpice simulation utilizing voltage controlled switches with SINE wave control

It is important to note that in order to have a simulation output that looks like **Figure 25**, we must include the sampling capacitors. The sampling capacitors in this example are C2, C4, C5, and C6. We must also test this mixer without the amplifiers attached. When the amplifiers are attached to the outputs of the mixer, we see a loaded effect as a result of the characteristics of the op-amps, which may not be the result expected. This emphasizes the importance of testing the receiver in

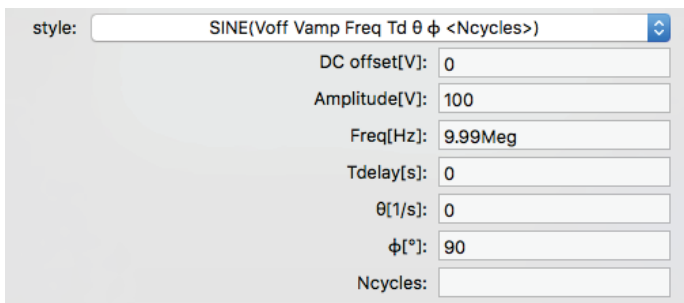


Figure 29. Settings for 90-degree SINE wave voltage output.

compartmentalized sections rather than as a whole. In this way, we can verify each component individually making troubleshooting much easier and faster.

The first method of simulating the mixer is more intuitive and tends to operate more cleanly than the second. The second method offers an easier way of changing the local oscillator frequency. However, with some variables like the local oscillator frequency, period, and switch time coded as *.param* variables, it becomes just as easy to change the local oscillator values with the first method.

The last section of the circuit that is beneficial to simulate beforehand are the amplifiers and low pass filters with the final I and Q signals that will be entering the audio jack. It's important to remember that the voltage output from the Tayloe Mixer will not be the same as discussed previously due to the characteristics of the filters and amplifier chosen for the design. There are multiple amplifier topologies that can work for the radio. As mentioned in the design comparison section of this paper, the amplifiers have different characteristics that can be advantageous or disadvantageous in depending on the application. **Figures 30** and **31** illustrate two different amplifier circuits and how they can be simulated in LTspice as a starting point for your design. These represent only the I signal. A duplicate of this circuit can be used for the Q signal.

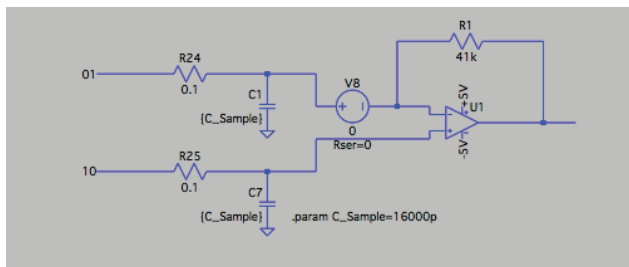


Figure 30. Instrumentation amplifier LTspice simulation topology

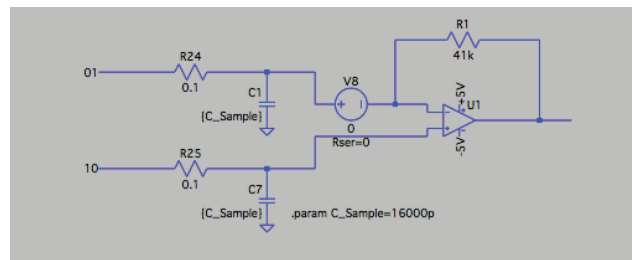


Figure 31. Softrock amplifier LTspice simulation topology

BOARD LAYOUT AND BUILD

The layout of a PCB board can potentially have extreme effects on the performance of the circuit. A checklist of good design practices to keep in mind when designing the board follows:

- Include a prototype area
- Use sockets for components, such as capacitors or resistors, that will likely need adjustment.
- Include add-in areas with jumpers that will allow you to try multiple options on a single board
- Add test points everywhere! This makes testing and troubleshooting a lot easier.
- Strive to make the I and Q signal paths as similar as possible.
- Match component values and paths as closely as possible, as this can impact image rejection.
- Think about the types of connectors you will use on your board: audio jack, SMA, Arduino connector, etc.

- Contemplate the mechanical aspect of the design: how will it fit in a case? Is the display readable? Are the controls or test points difficult to reach?
- Anticipate which traces will be carrying large amounts of current and adjust for this with the trace sizes. There are multiple online tools to calculate trace impedance (and therefore size) for different current values, but we found the KiCad trace impedance calculator to be convenient.
- KiCad provides a bug check in the schematic capture and the board layout tools. Use them! The PCBnew bug check will look for unconnected traces and other violations. Ask a qualified peer to review your schematic especially, and your PCB. Utilize the board manufacturer's bug checker and their design rules.
- Label the PCB using the silkscreen with important information such as what it is and where the relevant documentation is located.
- Use the 3D viewer on PCBnew to ensure silk screen text is readable and placed correctly.
- Use a flood fill ground plane on both the front and back of the board to avoid ground loops, and to help bypass capacitors do their job.
- Try to keep the analog and digital components separate, because digital components cause analog noise.
- Refrain from having the input and output signals close to each other on the board, as it can lead to oscillation.

Board Bring-Up Plan: Before ordering a printed circuit board, or assembling the electronics, it is helpful to create a board bring-up plan. This is a guide to how one plans to assemble and test the different parts of the radio. By going through each sub-circuit or block as described early, one can troubleshoot sub-circuits and ensure that they work properly before moving on. Before applying power, check each sub-circuit and verify that there are no shorts between power and ground. It is not uncommon for parts to be damaged by electrostatic discharge, or accidents. This is one of several reasons to purchase extra components, especially the smaller package components as one lost or damaged component can set you back a week waiting for new parts.

As part of the board bring-up plan, include debugging strategies. Some debugging tips to keep in mind are:

- Verify jumpers are in the correct places.
- Carefully inspect the board with a magnifier for soldering problems.
- Use an ohmmeter to verify connections
- Check for oscillator signals, as we found sometimes you think they are there, but something happened in your last tests.
- Look for saturated op amps. Saturated op amps have their output voltages close to the power supply rails. They are not amplifying your signal if they are saturated.
- If more noise appears to be present than you like, try checking for ground loops.
 - The ground loop may be caused by the computer or soundcard attached to your board.
 - Experiment with running the receiver on batteries instead of through a laptop USB port

- Try disconnecting your laptop from ground by running it on batteries instead of the charger connected to the wall.
- Play around with the ground loop jumpers you included to prevent ground loops in anticipated problematic areas (if applicable).
- Add transformers to isolate the antenna input and the audio output.

A good example of a board bring-up plan can be found here:

- https://github.com/froeca/Software-Defined-Radio/blob/master/Milestone_Submissions/Board%20Construction%20and%20Testing%20Plan.pdf

Ensure that the designer has an adequate test bench to assemble and troubleshoot the device. Ideally one should work in a well-lit space at a large desk with an anti-static mat. For the SDR radio we required a multimeter, signal generator, and an oscilloscope for a basic test bench. For an all-in-one solution, many students used a Digilent Analog Discovery 2, a device that plugs into a USB port and can perform the testing requirements for a project like this. The only drawback to this handy piece of equipment, is the limited frequency coverage it affords. This can be somewhat addressed by using harmonics of the signal generator for frequencies above 10 MHz. In addition, be sure to have a good soldering set up. A good kit may include the following.

- Soldering Iron (Wedge tip if possible)
- Solder
- Solder-Wick
- Flux
- Helping Hands
- Magnifying glasses
- Small Fume-Extractor Fan
- Wire Strippers

Remember, always turn off the soldering iron after use to prevent oxidizing the tip. For more soldering tips check here <https://www.jameco.com/Jameco/workshop/techtip/soldering-tips.html>

SOFTWARE

The software we used was Quisk, a Python program running on the PC, and Arduino code running on the SDR board. The details of installing and setting up the software will not be discussed directly in the paper. However, the following resources cover the software set up and bugs more in detail:

- <http://james.ahlstrom.name/quisk/docs.html>
- <https://groups.io/g/n2adr-sdr/topics>
- https://github.com/frohro/IQ_SDR/tree/master/Quisk/Arduino
- https://github.com/KonradMcClure/SDR_Receiver
- https://github.com/greenjacketgirl/SDR_Receiver/wiki/9.-Software
- <https://github.com/threeme3/QCX-SSB>

A great idea when testing the software is to write simple test programs. For instance, try writing an Arduino program that directly controls the Si5351a rather than one that communicates with Quisk to do so. An example of a simple test program is found in **Figure 32**.

When selecting the soundcard, be aware of the effect it will have on your radio. A sound card with a poor signal to noise ratio can degrade the quality of your receiver. Fortunately, nowadays there are inexpensive USB sounds cards with 24-bit ADCs and at least 90 dB of signal to noise ratio. Another characteristic to check on the sound cards is whether it is dual channel. It must have both a right and a left channel for there to be any image rejection.

```
#include "si5351.h"
#include "Wire.h"

Si5351 si5351;

void setup()
{
  bool i2c_found;
  i2c_found = si5351.init(SI5351_CRYSTAL_LOAD_8PF, 0, 0);
  si5351.set_freq(4*90000000ULL, SI5351_CLK0); // set local oscillator frequency
  Serial.begin(115200);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  digitalWrite(3, HIGH); // D3 (s0) select line for bandpass filter selection
  digitalWrite(4, LOW); // D4 (s1) select line for bandpass filter selection
}

void loop()
{
  si5351.update_status();
  Serial.print("SYS_INIT: ");
  Serial.print(si5351.dev_status.SYS_INIT);
  Serial.print(" LOL_A: ");
  Serial.print(si5351.dev_status.LOL_A);
  Serial.print(" LOL_B: ");
  Serial.print(si5351.dev_status.LOL_B);
  Serial.print(" LOS: ");
  Serial.print(si5351.dev_status.LOS);
  Serial.print(" REVID: ");
  Serial.println(si5351.dev_status.REVID);
}
```

Figure 32. Arduino simple test program code to experiment with the local oscillator and bandpass filters. The local oscillator is set for 9MHz and multiplexer select lines to 01.

CONCLUSIONS

In this paper, we have discussed the theory behind quadrature signals, drew comparisons between four different self-assembled software-defined radio receivers, and addressed design tips for those interested in building their own SDR receiver. The interest Guido Ten Dolle's μ SDX drew from the community illustrates the relevance of studying and presenting the knowledge we have on similar types of SDR receivers.

We broke down the theory of quadrature signals and mixers into an analogy of plane propeller rotations with visuals to illustrate the concepts. We hope this explanation will help extend understanding about this topic to a larger audience.

Through our design comparison analysis, we discovered that depending on the desired application of your receiver, different types of local oscillators may be selected. In addition, we noted the Tayloe Detector is the best selection for a mixer due to its low conversion loss and simplicity in design. The amplifiers can be difficult to analyze, even in a simulation program, due to some configurations acting as current or transresistance amplifiers rather than solely a voltage amplifier. The instrumentation amplifier presented itself as the simplest amplifier to analyze since it has infinite input impedance and acts as a voltage amplifier.

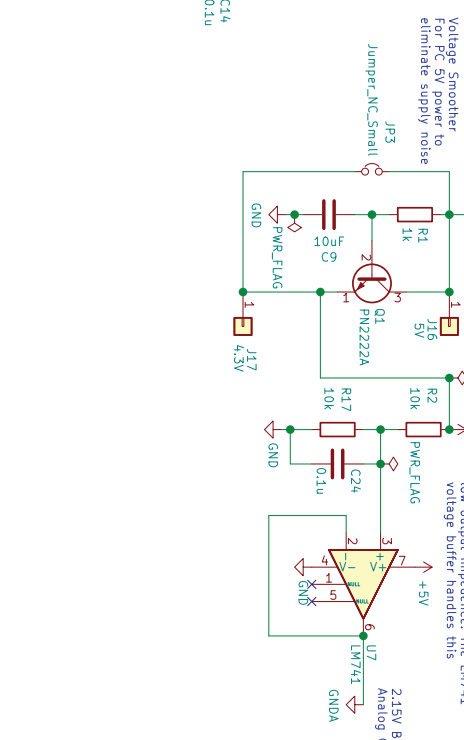
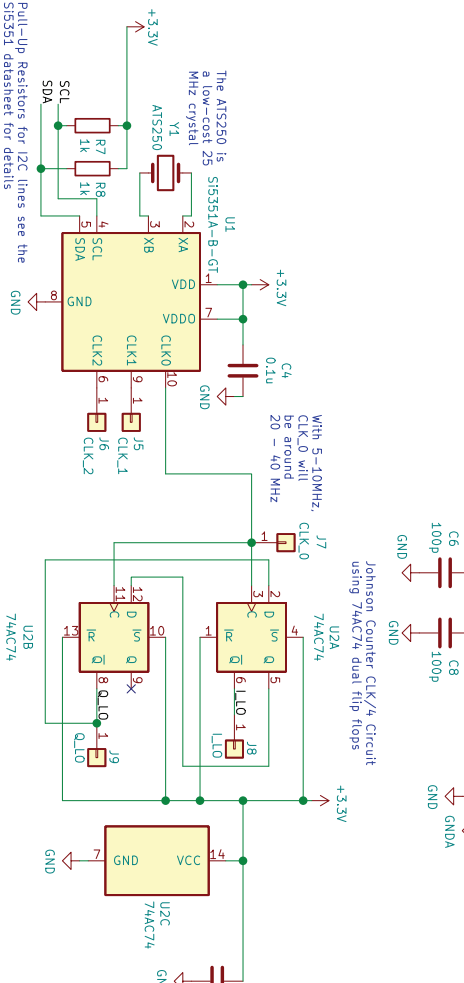
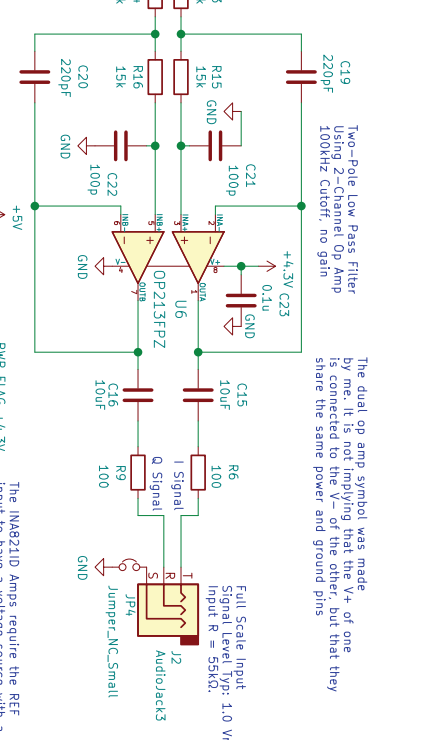
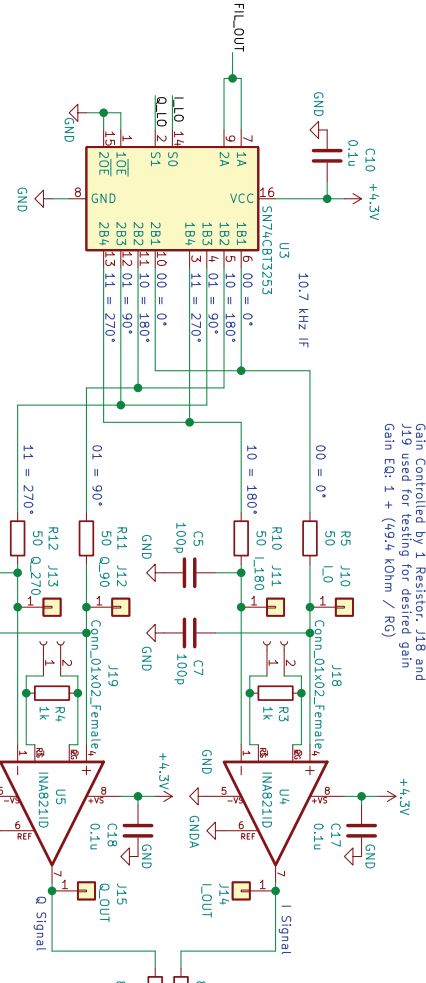
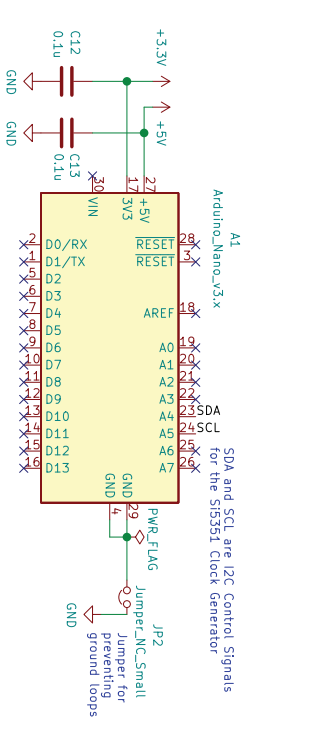
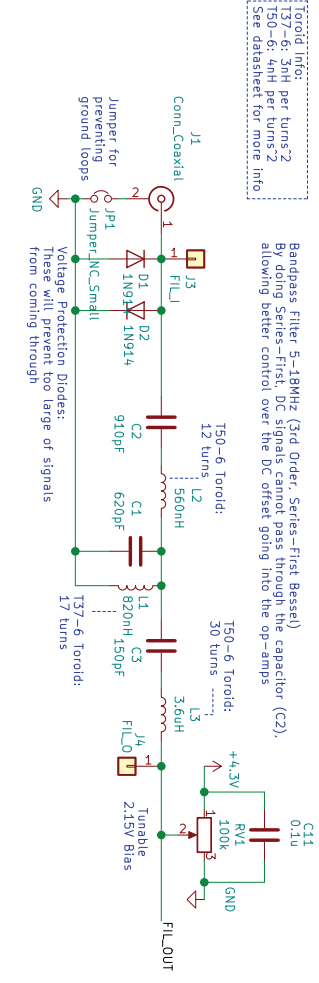
The design guide will be helpful to those unsure of where to start in the design process of an SDR receiver. We included a few issues that we encountered and possible solutions to them so that others can avoid the problems we had.

The details and instructions compiled in this paper are catered towards making the study and design of SDR receivers more accessible to a wide audience. With the information and time barrier lowered, we hope ham radio operators of a wider range of ages and expertise can discover the joys of designing and building a self-tailored software-defined radio with a thin wallet.

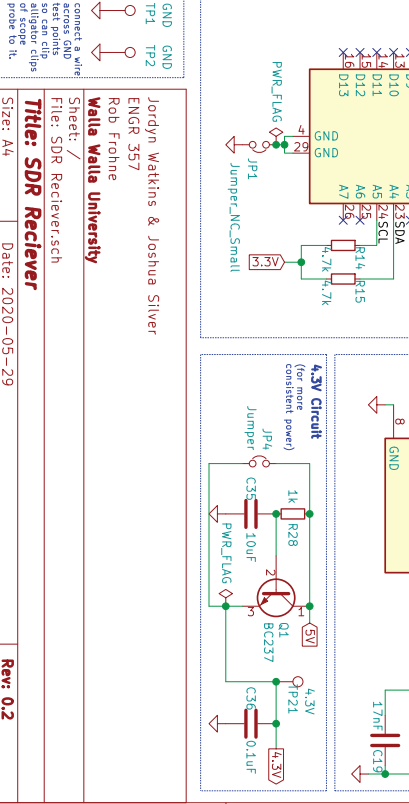
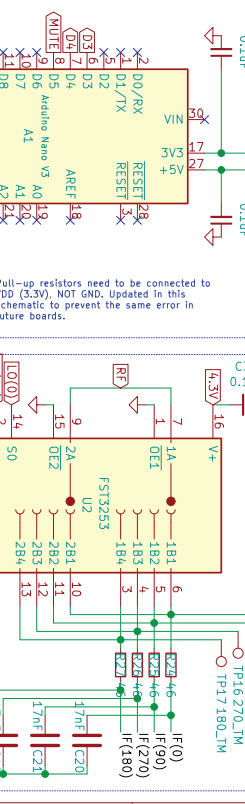
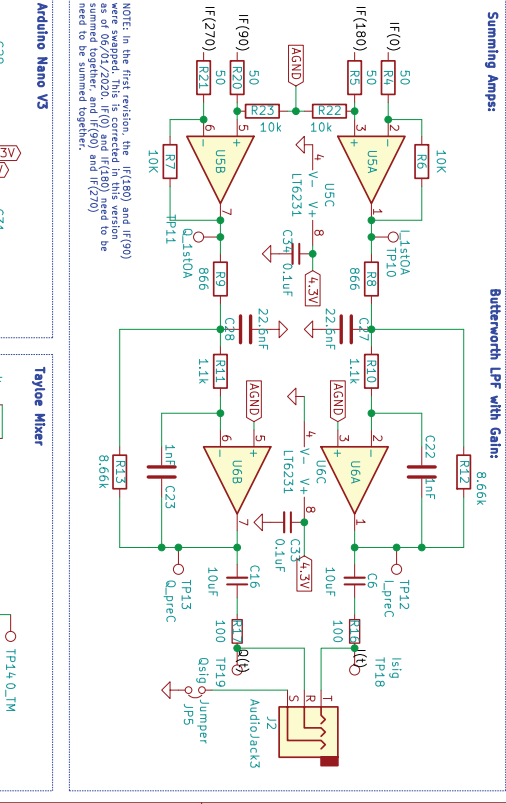
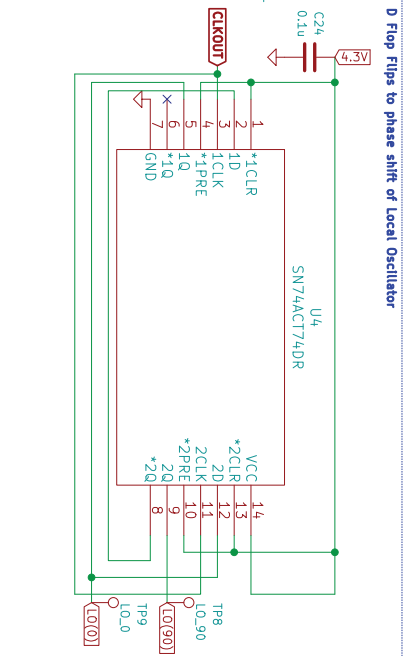
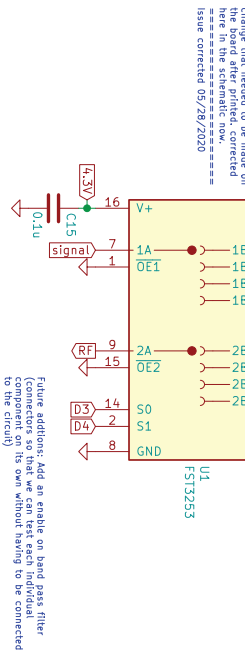
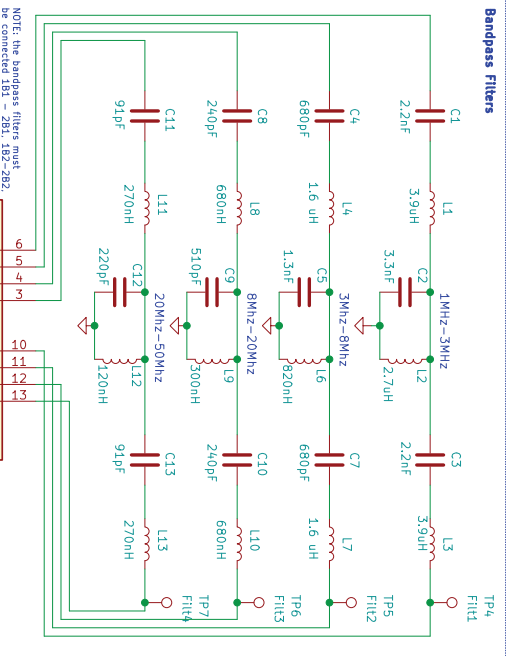
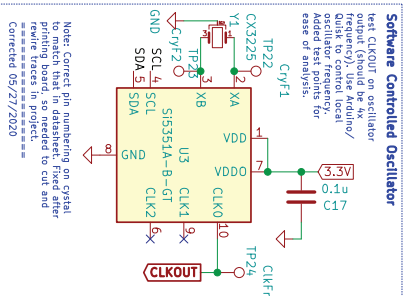
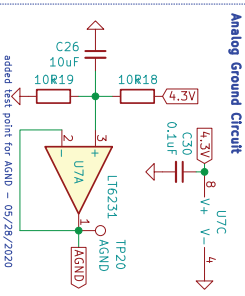
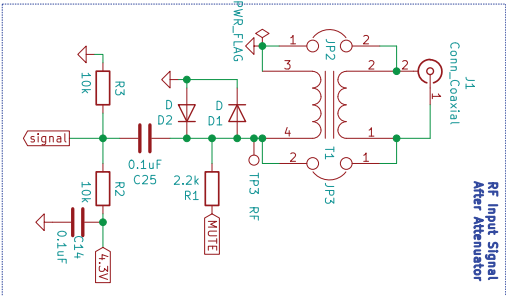
-
- ⁱ “US Amateur Radio Population Grows Slightly in 2018.” *ARRL: The National Association for Amateur Radio*, 14 Feb. 2019, www.arrl.org/news/us-amateur-radio-population-grows-slightly-in-2018.
- ⁱⁱ Pepitone, Julianne. “The Uncertain Future of Ham Radio.” *IEEE Spectrum: Technology, Engineering, and Science News*, 10 July 2020, spectrum.ieee.org/telecom/wireless/the-uncertain-future-of-ham-radio.
- ⁱⁱⁱ “Si5351a CLK0 and CLK1 Signals.” *Groups.io*, 12 Aug. 2020, <https://groups.io/g/ucx/topic/76162126#>
- ^{iv} Kebede, Zenaneh Ashebir. “Low frequency Quadrature detector design, simulation and implementation for use in underground communication.” 2014. *Graduate Theses, Dissertations, and Problem Reports*. 219. <https://researchrepository.wvu.edu/etd/219>.
- ^v Soer, Michiel. “Analysis and comparison of switch-based frequency converters.” *University of Twente: Faculty of Electrical Engineering, Mathematics & Computer Science*, Sept. 2007, https://essay.utwente.nl/58276/1/scriptie_Soer.pdf.
- ^{vi} Youngberg, Gerald. “A Software Defined Radio for the Masses, Part 4.” *QEX*, Mar./Apr. 2003, <http://www.arrl.org/files/file/Technology/tis/info/pdf/030304qex020.pdf>.
- ^{vii} “More on Ground Loops and Audio Settings - g4zfqradio.” *Google Sites*, Aug. 2011, sites.google.com/site/g4zfqradio/more-on-ground-loops-and-audio-settings.
- ^{viii} Kicad is an EE schematic design tool, <https://kicad-pcb.org/>
- ^{ix} Notes, Electronics. “RF & Microwave Filters: the Basics.” *Electronics Notes*, www.electronics-notes.com/articles/radio/rf-filters/understanding-rf-filters-basics-tutorial.php.
- ^x “Si5351A Breakout Board.” *Si5351A Breakout Board | Etherkit Documentation*, etherkit.github.io/si5351abb_landing_page.
- ^{xi} Soer, Michiel. “Analysis and Comparison of Switch-Based Frequency Converters.” *University of Twente, Faculty of Electrical Engineering, Mathematics & Computer Science*, 2007.

Design 1

Toroid info:
 137-6: 3rd turn's 2
 150-6: 4th turn's 2
 See datasheet for more info



Design 2

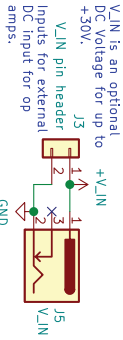
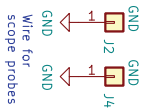


Jordyn Watkins & Joshua Silver
ENGR 357
Rob Frohne
Walla Walla University

Title: SDR Receiver
Date: 2020-05-29
Rev: 0.2
Id: 1/1

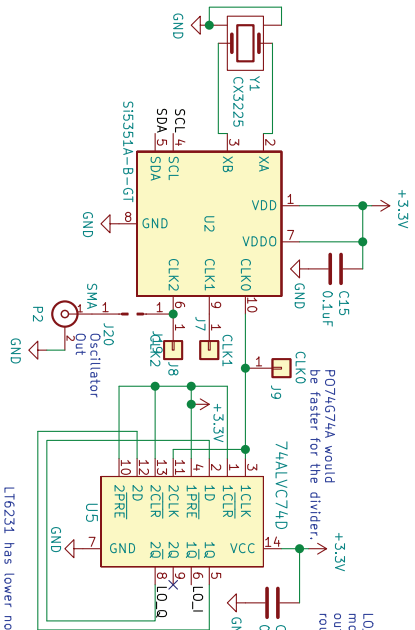
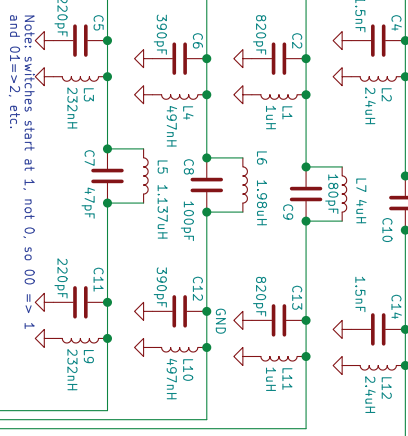
Design 3

-param freq=10000000
 .param if:freq=(freq + 1000)



Inductors need to go to 1.5VDC in BPF instead of GND.

BPF Responses:
 1.5MHz-2.5MHz
 4MHz-12.8MHz
 8MHz-16MHz
 16MHz-30MHz

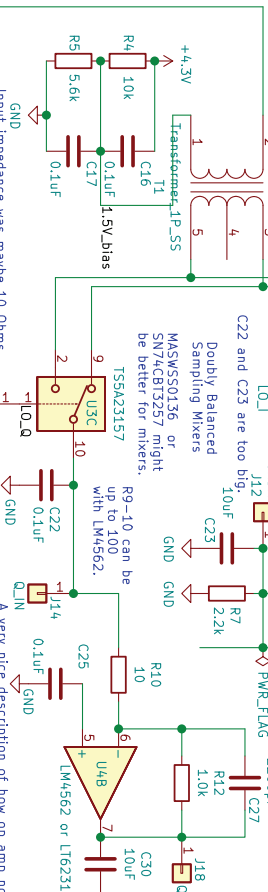


LT6231 has lower noise figure (4.7 dB) but costs about 7 times as much as the SAs5532. LME49860 is intermediate option. LM4562 could be even better (10.6 dB $2.7nV/\sqrt{Hz}$).

Making T1 a step up transformer makes the S/N and NF better, but op amp input. With 10 Ohm Z_{in} sampling capacitor, need to be adjusted.

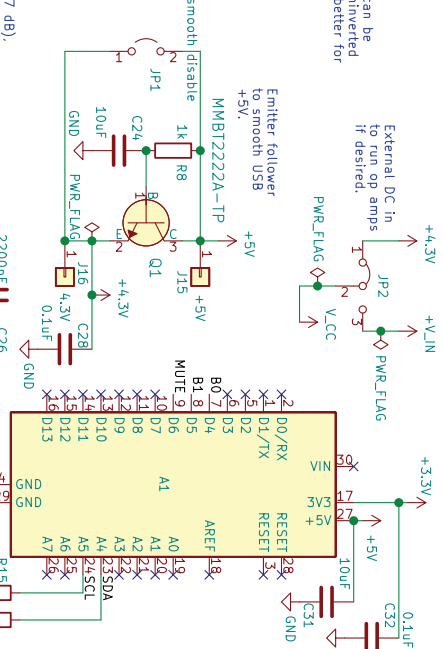
PO3814A might be better for Tayloe mixer.

Input impedance was maybe 10 Ohms. Make R9-R12 five times larger, 50 Ohm Z_{in} and 5k. C26-C27 should be 1/5 the size.



R9-10 can be up to 100 with LM4562.

A very nice description of how op amp noise effects the noise figure is given in this data sheet: <https://www.ti.com/lit/ds/symlink/lmh6529.pdf> in sections 7.3.4 and 7.3.5 (pages 24-7). Finding an optimum R9 (input resistance for the inverting amplifier) seems to imply that the optimum gets better as it goes to zero, assuming R_s is fixed. The excess noise is approximately $(e_n)^2 \cdot 2 \cdot (1 - \frac{R_s}{R_9})^2 \cdot (R_s + R_9)^2 \cdot 4kTR_9$. The best you can do is to multiply by R_s+R₉, so for R_s = 10 and R₉ = 50, e_n is more important in all the cases I've seen so far. A nice calculator is at: <http://dicks-website.eu/noisecalculator/index.html>



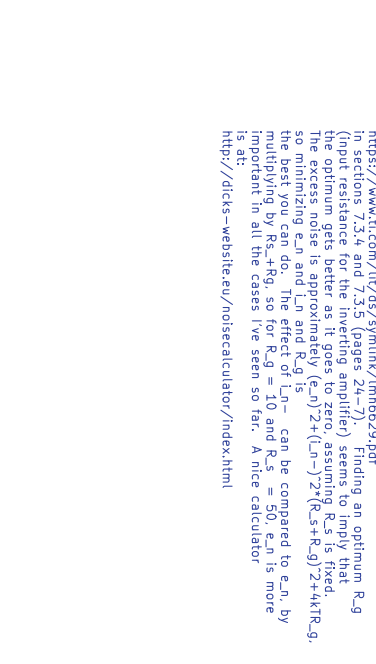
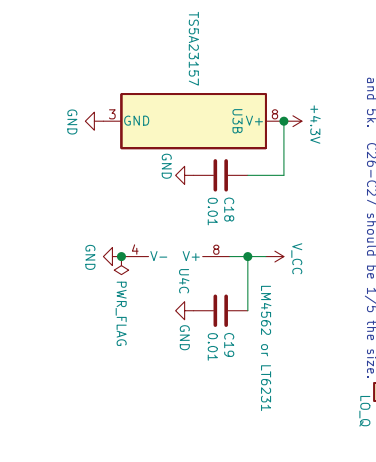
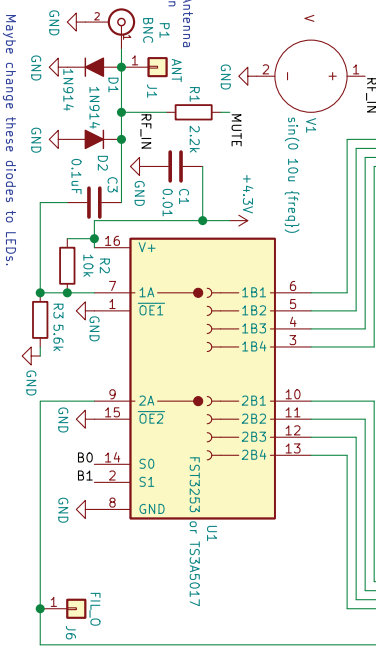
LO1 and LO2 can be moved to the uninverted outputs if it is better for routing.

Emitter follower to smooth USB to +5V.

External DC in to run op amps if desired.

smooth disable

smooth disable



Maybe change these diodes to LEDs.

Forward Error Correction and Pictures from Mars
David Garner, N6WY
N6WY@arrl.net

Abstract.

This paper illustrates how a sender can encode binary data words so that the data words can be sent over a weak noisy communication channel and a receiver can detect the correct data words. No abstract mathematical equations are used; just two numerical examples. The first example is a simple example. The second example is the encoding method used on Mariner 9 spacecraft to transmit pictures of the Martian surface as it orbited Mars. The Hadamard linear block code is used for both examples.

Introduction.

While doing research to reverse engineer the HF digital mode called Olivia, I developed two numerical examples (no strange mathematical symbols and abstract formulas.) that can explain to Radio Amateurs about linear Forward Error Correction (FEC) techniques for sending binary symbols over a radio link. Mathematicians and Communications Engineers may look at my examples with some disdain but my examples do illustrates how FEC works.

This paper covers the following:

- A short introduction of the various methods of Forward Error Correction for communications links.
- Background information about the Hadamard linear block code.
- A simple example of a “Covert Number Station” broadcasting random numbers.
- A non mathematical method for constructing a Hadamard matrix used for encoding and decoding Forward Error Correction Code Words.
- Background information and numerical examples for the code used on the Mariner 9 spacecraft. Mariner 9 mission was to orbit Mars and transmit pictures back to Earth. Forward Error Correction was used to send pictures of the Martian surface back to earth on a weak and noisy communication link.
- Bit Error Rate determination using Forward Error Correction.

Introduction to Forward Error Correction.

According to the BitcoinWiki, Forward Error Correction (FEC) is a technique used for controlling errors in data transmission over unreliable or noisy communication

channels. The central idea is that the sender encodes the message in a redundant way. The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message. FEC gives the receiver the ability to correct errors without needing a reverse channel to request retransmission of data, but at the cost of a fixed, higher forward channel bandwidth. FEC is therefore applied in situations where retransmissions are costly or impossible, such as one-way communication links and when transmitting to multiple receivers in multicast. The maximum number of errors that can be corrected is determined by the design of the FEC code, so different forward error correcting codes are suitable for different conditions [1].

A simplistic example of FEC is to transmit each data bit 3 times, which is known as a (3,1) repetition code. This allows an error in any one of the three samples to be corrected by "majority vote". Though simple to implement and widely used, this triple modular redundancy is a relatively inefficient FEC.

The two main categories of FEC codes are block codes and convolutional codes.

There are several linear block codes such as Hamming codes, Reed-Solomon codes, Hadamard codes, Expander codes, Golay codes, and Reed-Muller codes.

In convolutional codes, the message comprises of data streams of arbitrary length and a sequence of output bits are generated by the sliding application of Boolean functions to the data stream. The convolutional codes can operate on a continuous string of data, whereas block codes operated on words. Convolutional codes also have memory—the behavior of the code depends on previous data [2].

Hadamard Code Background.

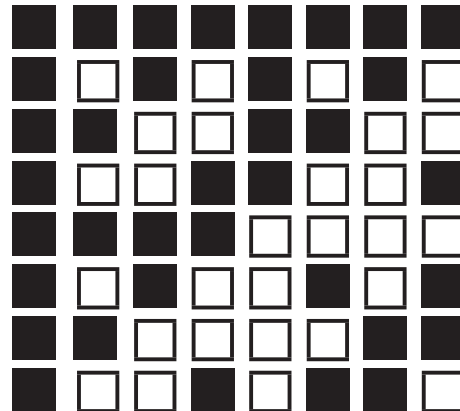
In this paper, I am limiting my discussion and examples to the Hadamard linear block code. The Hadamard code is also known under the names Walsh code, Walsh family, and Walsh–Hadamard code in recognition of the American mathematician Joseph Leonard Walsh [3]. Hadamard matrices are simple matrix structures and are used to generate the Hadamard code.

A Hadamard matrix is square, have entries +1 or -1 and have orthogonal row vectors and orthogonal column vectors. Figure 1 shows an 8 X 8 Hadamard matrix. Figure 2 is also an 8X8 Hadamard Matrix. The point I am trying to make is that you do not use the mathematical format for FEC.

Figure 1. 8X8 Hadamard Matrix

$$\begin{bmatrix}
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
 \hline
 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
 \end{bmatrix}$$

Figure 2. Hadamard Matrix using Black and White Colors



Hadamard matrices have been actively studied by mathematicians for about a century and half and new uses are still being discovered. At the end of the referenced web page [3] is a list of practical applications for Hadamard matrices including an amateur radio protocol.

The Hadamard code is also used for code division multiple access (CDMA). In CDMA, the Hadamard code is referred to as Walsh Code, and is used to define individual communication channels. Each user will use a different Code Word, to modulate their signal. Because Walsh Code Words are mathematically orthogonal, a Walsh-encoded signal appears as random noise to a CDMA capable mobile terminal (cell phone), unless that terminal uses the same Code Word as the one used to encode the incoming signal [3].

Simple Example.

My simple example is not a very practical example because it corrects only one error. This example is of a covert number station that needs to broadcast what appears to be random numbers. This number station broadcasts only integers in the range of 0 to 7 using binary numbers. For this simple example, the broadcast station located in Cuba uses two different drum sounds in some music that starts the next segment of a radio program. A high drum sound is a one and a low sound is a drum zero.

Each Data Word contains three bits that is one number in the range of 0 through 7. To send eight different Data Words using the Hadamard Code, you need 8 different Code Words. An 8 X 8 Hadamard Matrix has 8 rows and each row is a Code Word. At this point in my simple example, I am not going to show how to make a Hadamard Matrix. Later, I will show a non mathematical way of creating a Hadamard Matrix of the size required for the Mariner 9 mission.

Since computers use 1's and 0's, I am going to change the -1's to 0's for the simple example Hadamard matrix as shown in Table 1. To the right of the matrix are the Data Words and to the left are the Data Values. Instead of sending 011 for the number 3, after Hadamard encoding the number 3 becomes 10011001.

Table 1. Encode Table		
Data Value	Hadamard matrix	Data Word
0	11111111	000
1	10101010	001
2	11001100	010
3	10011001	011
4	11110000	100
5	10100101	101
6	11000011	110
7	10010110	111

Table 2. Decode Table					
Data Value	Hadamard Matrix	RCW0 score	RCW1 score	RCW02 score	RCW3 score
0	11111111	4	3	2	3
1	10101010	4	5	4	3
2	11001100	4	5	4	5
3	10011001	8	7	6	5
4	11110000	4	3	4	3
5	10100101	4	5	6	7
6	11000011	4	5	6	5
7	10010110	4	3	4	5

As a side note, I did not need to assign the top row 0 and the bottom row 7. Any assignment would work as long as the receive stations know the assignment.

Using the Table 1 Encode Code Table, the covert number station is going to broadcast the Data Value 3, for four consecutive times. The Data Word that is

being encoded four times is 011 and the Transmitted Code Word that is being sent four times is 10011001.

The receive site has some interference and the agent in the field receives the four Code Words with some errors in the Code Words. With the help of Table 2, the field agent can decode the message. (Figure 3 defines the column headings used in Table 2.) The field agent needs four digits to identify the mission that are listed in the field agent mission book — the mission number book is not shown in this paper because it is Top Secret. The mission for this broadcast is number 3333.

Figure 3. Column Headings Definitions for Table 2

Send Data Value = 3 and Data Word = 011
Transmitted Data Code Word = 10011001
RCW_x = Received Code Word with x errors
RCW₀ = 10011001 (0 error.)
RCW₁ = 10001001 (1 error. red indicates received bit error)
RCW₂ = 10000001 (2 errors. red indicates received bit error)
RCW₃ = 10000101 (3 errors. red indicates received bit error)

The receive decode Hadamard matrix has the same values as the transmit encode Hadamard matrix. A simple scoring method identifies the Transmitted Code Word. The Received Code Word is given one point for each time it has the same bit value in the same position as a row in the Hadamard Matrix. The row with the highest score identifies the data value.

The first Received Code Word (RCW₀) has zero bit errors and has a perfect score for the Code Word that corresponds to the Data Value of 3. When a Received Code Word has a perfect match in the decode table all the other Received Code Words have a score that is equal to the distant value that a row in the Hadamard matrix has to all the other rows in the Hadamard matrix. The distant value is simply the number of bits that one row has that are different than any other row in the Hadamard Matrix. In standard coding theory notation for block codes, my simple example Hadamard code is a [8,3,4] where the 8 is block length (the number of bits in a Hadamard row), the 3 is the message length (the number of bits in the Data Word) and the 4 is the distance. The Mariner 9 code that is discussed later has a large block length compared to the message length that allows many bit errors in the Received Code Word.

The second Received Code Word (RCW1) has one bit error. My simple example using a Hadamard code [8,3,4] can only correct one bit error. The highest score in the decode table is for the Received Code Word for the Data Value of 3. Based on the score in the decode table, the field agent knows that the Code Word has one error and the Data Value is 3.

The third Received Code Word (RCW2) has two bit errors. Based on the score for RCW2 in the decode table, the field agent knows that RCW2 has two bit errors but does not know the Data Value. The field agent, who is in Oregon, checks the mission book and see that all missions that have the first three numbers of 335 and 336 are in Chicago. Using deduction, the field agent knows that the first three numbers in his mission are 333.

The fourth Received Code Word (RCW3) has three bit errors. The field agent does not know the RCW3 has three bit errors and using the scoring in the decode table the agent concludes the RCW3 has one bit error and he prepares to complete mission number 3335 instead of mission number 3333. FEC does have limitations.

How to construct a Hadamard Matrix.

A Hadamard matrix H of order n is an $n \times n$ matrix. My simple example used a Hadamard matrix of order 8 and is an 8×8 matrix. A Hadamard matrix can exist only if n is a multiple of 4. A 2×2 Hadamard matrix does exist and is a start point for building larger Hadamard matrices.

Figure 4 shows Hadamard matrices of order 2, 4, 8, and 16. The subscript on the H indicates the order of the Hadamard matrix. In the larger matrices, construction lines divide a matrix in 4 quadrants.

A Hadamard order 4 matrix can be made from the Hadamard order 2 matrix. To construct the Hadamard matrix of order 4, make a new Hadamard matrix of order 2 by changing the sign of every entry in the Hadamard order 2 matrix. Place that new Hadamard order 2 matrix in the lower right quadrant of the order 4 matrix. Copy the original order 2 matrix in the other 3 quadrants of the order 4 matrix. Similarly, a Hadamard order 8 matrix can be made from the Hadamard order 4 matrix and a Hadamard order 16 matrix can be made from the Hadamard order 8 matrix. A large Hadamard matrix can be made using this technique. There are easier ways to make Hadamard matrices using mathematics.

Figure 4. Hadamard Matrices Construction

$$\begin{aligned}
 \mathbf{H}_2 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\
 \mathbf{H}_4 &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \\
 \mathbf{H}_8 &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \\
 \mathbf{H}_{16} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{bmatrix} \\
 \mathbf{H}_{16} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

Figure 2 is from reference [4]

Mariner 9 Background.

In 1971, the Mars Mariner 9 Spacecraft was the first man-made object to orbit a different planet and send back pictures of Mars that was 84 million miles from earth using a 20-watt transmitter. The pictures were black and white. Each Martian photograph required 4.5 million bits. The 4.5 million bits were split into 6 bit Data Words. Using 6 bits per Data Word, 64 different Data Words were created [5],[8].

Because of the weak signal from Mariner 9, the maximum useful data length of a Data Word with forward error correction was about 30 bits. The 5-repeat code was a possibility, having the advantage that it is very easy to implement, but it is only a capability of correcting 2 errors in an encoded word. To send a 6 bit Data Word with 5-repeat code becomes a 30 bits per Transmitted Code Word. (For sending a six bit Data Word like 010101, the data stream with 5-repeat code becomes 000001111100000111110000011111.) The Hadamard code was chosen for Mariner 9 because encoding a 6 bit Data Word into a 32 bit Transmitted Code Word gives a capability of correcting 7 errors at the received location [6].

A Hadamard order 5 matrix has 32 Code Words and has 32 bits per Code Word. The 32 bits per Code Word meets the Mariner 9 requirement but a total of 64 Code Words are required. A Hadamard order 6 matrix has 64 Code Words but the Mariner 9 communication link cannot support sending 64 bit Code Words. The solution is to use an augmented Hadamard code.

An augmented Hadamard code for Mariner 9 uses two different Hadamard order 5 matrices stacked together such that it becomes one matrix with 64 rows and 32 columns. The second matrix has 0's where the first matrix has 1's and the second matrix has 1's where the first matrix has 0's. All the rows in the augmented Hadamard code matrix of order n are different and will have the same distant value as the standard Hadamard code matrix of order n. In standard coding theory notation for block codes, the Mariner 9 augmented Hadamard code is [32,6,16] and the standard Hadamard code matrix of order 5 is [32,5,16]. Table 3 contains the augmented Hadamard code matrix that I am going to show FEC capability for Mariner 9. Table 3 also has the scores for Received Code Words. The Mariner 9 Code Words discussion is later in this paper.

Now before I proceed to show the decode capability of the [32,6,16], I need to comment about the design of Mariner 9. Processors and memory was not that great during the 1960s. The central processor for Mariner had a memory of 512 words [5]. Mariner 9 could not store 64, 32-bit Code Words. Mariner 9 did use a hardware design that was more economical in terms of speed, space, and weight. The hardware designed calculated a Transmitted Code Word from a Data Word rather than read it out of a stored array [7]. Amateur Radio HF modes that use Hadamard Code Words are not read out of a stored array but use a software program that calculates a Transmitted Code Word from each Data Word.

The earth receiving station for Mariner 9 required a fast decoding algorithm. The decision to use the FEC Hadamard code was based primarily on the decoding algorithm. A NASA engineer with the last name of Green designed what was called "The Green Machine". "The Green Machine" used a Fast Fourier Transform to calculate the Data Words from the Received Code Words [7].

Using Table Look-Up for Mariner 9 Code Words.

The earlier simple example had a small Code Word look-up table because the simple example had only 8 Data Words. The Mariner 9 has 64 Data Words and uses an augmented Hadamard code that requires two 32 X 32 Hadamard matrices. I copied the Mariner 9 Data Word to Code Word assignment from reference [8]. I added the score columns for two different Received Code Words for the table shown in Table 3. I had to use two pages for Table 3. Table 3a contains the Received Code Word assignments for the Data Words with value of 1 for the most significant bit. Table 3b contains the Received Code Word assignments for the Data Words with value of 0 for the most significant bit.

Figure 5 has the Column Headings Definitions for Table 3. Red bit values in the Received Code Words are bit errors. The values in the table that are blue indicates the Data Word with the most significant bit value of 1 and green indicates the Data Word with the most significant bit value of 0.

The Table 3 shows that the highest score indicates the correct Data Word for a Received Code Word that has less than 8 bit errors. If a Received Code Word has 8 or more bit errors, the correct Data Word cannot be determined. Every entry in a column does not need to be completed for determining the correct Received Code Word. A score of greater than 24 determines the correct Received Code Word. For example, if a cell in table 3 has a score of 25 then no cell in that column will have a score greater than or equal to 25.

Figure 5. Column Headings Definitions for Table 3

Send Data Words alternates between 100011 and 000011

Transmitted Code Words alternates between:

10101010010101011010101001010101
00000000111111110000000011111111

First Received Code Word is 1E0 and last is 0E8

1

1E0 = 0 errors for Received Code Word that has 1 as most significant bit

0

0E0 = 0 errors for Received Code Word that has 0 as most significant bit

The number above E indicates the value for the most significant bit and the number following E indicates the number of errors in the Received Code Word.

Received Code Words. Red bit values are bit errors.

1E0 = 10101010010101011010101001010101
0E0 = 00000000111111110000000011111111
1E1 = 10101010010101011010101001010100
0E2 = 00000000111111110000000011111100
1E3 = 10101010010101011010101001010010
0E4 = 00000000111111110000000011110000
1E5 = 10101010010101011010101001001010
0E6 = 00000000111111110000000011000000
1E7 = 10101010010101011010101000101010
0E8 = 00000000111111110000000000000000

Note: The position of any bit error does not affect the score

Table 99a. Score for Received Code Words

Data Word	Code Word	1	0	1	0	1	0	1	0	1	0
		E0	E0	E1	E2	E3	E4	E5	E6	E7	E8
111111	11111111111111111111111111111111	16	16	15	14	15	12	15	10	15	8
100000	10101010101010101010101010101010	16	16	17	16	19	16	21	16	23	16
110000	11001100110011001100110011001100	16	16	17	18	15	16	17	18	15	16
101111	10011001100110011001100110011001	16	16	15	16	15	16	15	16	15	16
111000	11110000111100001111000011110000	16	16	17	18	17	20	15	18	15	16
100111	10100101101001011010010110100101	16	16	15	16	13	16	13	16	15	16
110111	11000011110000111100001111000011	16	16	15	14	17	16	17	18	15	16
101000	10010110100101101001011010010110	16	16	17	16	17	16	15	16	15	16
111100	11111111000000001111111100000000	16	0	17	2	17	4	17	6	17	8
100011	10101010010101011010101001010101	32	16	31	16	29	16	27	16	25	16
110011	11001100001100111100110000110011	16	16	15	14	17	16	15	14	17	16
101100	10011001011001101001100101100110	16	16	17	16	17	16	17	16	17	16
111011	11110000000011111111000000001111	16	16	15	14	15	12	17	14	17	16
100100	10100101010110101010010101011010	16	16	17	16	19	16	19	16	17	16
110100	11000011001111001100001100111100	16	16	17	18	15	16	15	14	17	16
101011	10010110011010011001011001101001	16	16	15	16	15	16	17	16	17	16
111110	11111111111111110000000000000000	16	16	17	18	17	20	17	22	17	24
100001	10101010101010100101010101010101	16	16	15	16	13	16	11	16	9	16
110001	11001100110011000011001100110011	16	16	15	14	17	16	15	14	17	16
101110	10011001100110010110011001100110	16	16	17	16	17	16	17	16	17	16
111001	11110000111100000000111100001111	16	16	15	14	15	12	17	14	17	16
100110	10100101101001010101101001011010	16	16	17	16	19	16	19	16	17	16
110110	11000011110000110011110000111100	16	16	17	18	15	16	15	14	17	16
101001	10010110100101100110100101101001	16	16	15	16	15	16	17	16	17	16
111101	11111111000000000000000011111111	16	16	15	14	15	12	15	10	15	8
100010	10101010010101010101010110101010	16	16	17	16	19	16	21	16	23	16
110010	11001100001100110011001111001100	16	16	17	18	15	16	17	18	15	16
101101	10011001011001100110011010011001	16	16	15	16	15	16	15	16	15	16
111010	11110000000011110000111111110000	16	16	17	18	17	20	15	18	15	16
100101	10100101010110100101101010100101	16	16	15	16	13	16	13	16	15	16
110101	11000011001111000011110011000011	16	16	15	14	17	16	17	18	15	16
101010	10010110011010010110100110010110	16	16	17	16	17	16	15	16	15	16

Table 99b. Score for Received Code Words

Data Word	Code Word	1 E0	0 E0	1 E1	0 E2	1 E3	0 E4	1 E5	0 E6	1 E7	0 E8
010101	01101001100101101001011001101001	16	16	15	16	15	16	17	16	17	16
001010	00111100110000111100001100111100	16	16	17	18	15	16	15	14	17	16
011010	01011010101001011010010101011010	16	16	17	16	19	16	19	16	17	16
000101	00001111111100001111000000001111	16	16	15	14	15	12	17	14	17	16
010010	01100110100110011001100101100110	16	16	17	16	17	16	17	16	17	16
001101	00110011110011001100110000110011	16	16	15	14	17	16	15	14	17	16
011101	01010101101010101010101001010101	16	16	15	16	13	16	11	16	9	16
000010	000000001111111111111100000000	16	16	17	18	17	20	17	22	17	24
010110	01101001011010011001011010010110	16	16	17	16	17	16	15	16	15	16
001001	00111100001111001100001111000011	16	16	15	14	17	16	17	18	15	16
011001	01011010010110101010010110100101	16	16	15	16	13	16	13	16	15	16
000110	00001111000011111111000011110000	16	16	17	18	17	20	15	18	15	16
010001	01100110011001101001100110011001	16	16	15	16	15	16	15	16	15	16
001110	00110011001100111100110011001100	16	16	17	18	15	16	17	18	15	16
011110	01010101010101011010101010101010	16	16	17	16	19	16	21	16	23	16
000001	00000000000000001111111111111111	16	16	15	14	15	12	15	10	15	8
010100	01101001100101100110100110010110	16	16	17	16	17	16	15	16	15	16
001011	00111100110000110011110011000011	16	16	15	14	17	16	17	18	15	16
011011	01011010101001010101101010100101	16	16	15	16	13	16	13	16	15	16
000100	00001111111100000000111111110000	16	16	17	18	17	20	15	18	15	16
010011	01100110100110010110011010011001	16	16	15	16	15	16	15	16	15	16
001100	00110011110011000011001111001100	16	16	17	18	15	16	17	18	15	16
011110	01010101101010100101010110101010	0	16	1	16	3	16	5	16	7	16
000011	00000000111111110000000011111111	16	32	15	30	15	28	15	26	15	24
010111	01101001011010010110100101101001	16	16	15	16	15	16	17	16	17	16
001000	00111100001111000011110000111100	16	16	17	18	15	16	15	14	17	16
011000	01011010010110100101101001011010	16	16	17	16	19	16	19	16	17	16
000111	00001111000011110000111100001111	16	16	15	14	15	12	17	14	17	16
010000	01100110011001100110011001100110	16	16	17	16	17	16	17	16	17	16
001111	00110011001100110011001100110011	16	16	15	14	17	16	15	14	17	16
011111	01010101010101010101010101010101	16	16	15	16	13	16	11	16	9	16
000000	00000000000000000000000000000000	16	16	17	18	17	20	17	22	17	24

Bit Error Rate (BER).

The average Bit Error Rate (BER) can be calculated for FEC linear block codes by using the average number of bit correction required for each Received Code Word. For the table look-up method, the number of errors for the Received Code Word is the difference between the perfect score and the actual score. For the words in Table 3 that have the high score, there are 10, 32 bit Received Code Words and the average BER is:

$$\text{Average BER} = (0+0+1+2+3+4+5+6+7+8)/(10*32) = 36/320 = .11 = 11\%$$

I used the word with 8 errors because the Mariner 9 FEC can detect 8 bit errors but it can correct only 7 bit errors.

Summary.

The table look-up method is generally not used for Hadamard FEC encoding and decoding Data Words. Encoding and decoding can be quickly done using math and a small computer program. The Mariner 9 augmented Hadamard code is [32,6,16], where the 32 is block length (the number of bits in a Hadamard row), the 6 is the message length (the number of bits in the Data Word) and the 16 is the distance. For FEC linear block codes, the maximum number of errors that can be corrected is determined by half the distance minus 1. The maximum number of errors that can be detected is half the distance.

References

1. BitcoinWiki, 'Forward error correction', https://en.bitcoinwiki.org/wiki/Forward_error_correction, 2017.
2. BitcoinWiki, 'Convolutional code', https://en.bitcoinwiki.org/wiki/Convolutional_code#Popular_convolutional_codes, 2017.
3. Wikipedia, 'Hadamard code', https://en.wikipedia.org/wiki/Hadamard_code, 2020
4. math.uic.edu, 'Hadamard Matrices and Hadamard Codes', http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/Hadamard_codes.pdf, date unknown
5. MIT, '6.02 Fall 2012 Lecture 6', https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-02-introduction-to-eecs-ii-digital-communication-systems-fall-2012/lecture-slides/MIT6_02F12_lec06.pdf, 2012
6. Malek, 'Coding Theory', https://nanopdf.com/download/coding-theory-hadamard-codes-california-state-university-east_pdf, 2015
7. UC Denver, 'Combinatorics in Space', <http://www-math.ucdenver.edu/~wcherowi/courses/m7409/mariner9talk.pdf>, date unknown
8. Anderson, University of Glasgow, 'ERROR – CORRECTING CODES' https://www.gla.ac.uk/media/Media_293296_smx.pdf, 2013

Aids to the Presentation and Analysis of WSPR Spots: TimescaleDB database and Grafana

Gwyn Griffiths, G3ZIL and Rob Robinett, AI6VN

Abstract – The `wsprnet.org` database provides a very good service for collecting, and making available, some 2.4 million spots from about 2500 reporters on a typical day. Its web page query tool, and those of third parties that scrape data from `wsprnet.org`, fulfill the needs of very many users. However, for users seeking to glean additional information from their own WSPR spots, or from those of a wider community, the tools provided by a relational database and a data visualization package become necessary. This paper outlines the rationale behind the `WsprDaemon` time series database, our initial experience with Influx as the database, and the reasons for moving to TimescaleDB. The system's architecture is described, highlighting resilient data gathering with user and server caches, an ability to handle delayed spot reporting, and a close coupling to Grafana as the visualization package. Three examples of Grafana Dashboards illustrate this approach and the utility of the results in providing users with a richer set of graphics to help them understand what WSPR spots tell them about propagation and their own installations and noise environment.

1. INTRODUCTION

Since the introduction of the Weak Signal Propagation Reporter (WSPR) protocol in 2008 reception reports have been uploaded to **`wsprnet.org`**. Originally written by Bruce Walker, W1BW, and now maintained by a small team of volunteers, `wsprnet.org` provides a simple, web-based interface of pull-down options to query its SQL database. There is also the 'old database' interface at **`wsprnet.org/olddb`** that supports web-page queries and 'scraping' using `curl`, a command line tool for data

transfer to or from web pages (**`curl.haxx.se`**). In addition, there is an invitation-only Application Programming Interface (API) by the `wsprnet` administrator Gary McMeekin, W1GJM that returns JavaScript Object Notation (JSON) data [1].

Writing in August 2010 Taylor and Walker reported that the WSPR database comprised 32 million spots, with 300-500 stations reporting 50,000 to 100,000 spots daily [2]. It is a tribute to the early work, the on-going support and investments in hardware that `wsprnet.org` provides an effective central reporting database for some 2500 reporting stations handling 2.4 million spots a day. Harder to quantify is the load on the database from its web-page users and from several third-party scraper and API applications.

Third-party applications come in several forms. There are near-real time [3], daily [4] and monthly [5] ranked lists, a map with simple pull-down options [6], suites of tables, charts and maps as a website [7], an app for mobile devices [8], to a comprehensive graphing tool drawing on its own copy of the full `wsprnet.org` database [9].

Given `wsprnet.org` itself and the plethora of third-party applications, why did we see the need for a separate database and graphing tool? There were several drivers, in part cascading as the initial project proved useful:

1. Author Robinett's concept of a robust and reliable reporting tool for WSPR spots from the multi-channel KiwiSDR (**`www.kiwisdr.com`**) led to him writing a Linux application, `WsprDaemon` (**`wsprdaemon.org`** and [10]), now with over 40 users.
2. Recognizing that the KiwiSDR was capable of estimating noise level at its input author Griffiths contributed an investigation of noise estimation at the same time and on the same frequencies as receiving WSPR spots [11]. As that data could not be reported to `wsprnet.org` a separate database was needed. Tommy Nourse, KI6NKO, suggested using Influx, a database specifically designed for time series data, with Grafana as the graphical display.
3. Following encouragement from several members of the HamSci community (**`hamsci.org`**) we added spots from `WsprDaemon` users to an Influx

Gwyn Griffiths is an independent amateur and citizen scientist based in Southampton, U.K. (corresponding author e-mail: gwyn@autonomousanalytics.com).

Rob Robinett based in Berkeley California, is CEO of TV equipment manufacturer Mystic Video and he's founded a series of Silicon Valley startups. He recently "rediscovered" amateur radio - after an absence of more than 40 years - and he applies his software expertise to developing systems to measure short wave radio transmission conditions.

(www.influxdata.com) database, and as its usefulness became apparent we added spots for all reporters, obtained from wspnnet.org.

- Over time, an ambition grew to be able to offload some of the third party data scraping from wspnnet.org by providing a secondary source of reported spots.

The remainder of this paper is organized as follows: section 2 introduces time series databases, summarizes our experience of Influx, and the reasons for our move to TimescaleDB (www.timescale.com); section 3 describes our TimescaleDB implementation, its capability and current user interfaces; section 4 describes how Grafana links to TimescaleDB and the features available; section 5 provides examples of insights into propagation and station performance that can be obtained from different graphical representations of the basic and derived data, section 6 shares some thoughts for future development of WsprDaemon, the database and graphical visualization.

2. TIME SERIES DATABASES: INFLUX AND TIMESCALEDB

Time series databases have emerged as a class of tools that deliver performance improvements over traditional databases by recognizing several characteristics of many time series. These include many insert operations, often in batches, and often timely, that is, few inserts are delayed, updates to existing inserts are rare, and queries often specify a time interval [11]. Time series databases 'chunk' the data in time, keeping the most recent data in memory. This facilitates fast response times for simple queries on recent data but requires paging between memory and disk for data in older chunks, slowing the response. In addition to Influx and TimescaleDB others include Clickhouse, OpenTSDB, Riak TS, and Gorilla. As database novices we were steered toward Influx.

2.1 Influx time series database

Influx is a purpose-built time series database with its own query language, similar to SQL. Using its excellent documentation [12] our database was set up in hours. The downside was that it took us time and accumulated experience to become aware of the limitations of Influx's approach for the characteristics of the WSPR dataset.

Briefly, Influx creates a 'measurement' (similar to an SQL table) with indexed tags (character fields such as Callsigns and Locators, but also Band, as only tags can be queried) and non-indexed fields (numeric fields such as SNR or Drift). Data are not stored as simple rows as they would appear in a spreadsheet or in many databases but are associated with the tag sets.

Cardinality is a term for the number of unique

combinations of measurements, tags and fields in a database. As cardinality grows, Influx's response time to queries increases, as does its CPU and memory requirements. It helps that most callsigns are always associated with the same locator. But even so, after four days, with 5.8 million spots, the cardinality had reached over 256,000. Influx's documentation considered this 'moderate' suggesting hardware with 4-6 cores and 8-32GB memory. As we were running on a Digital Ocean Droplet with 2 cores and 4GB memory the performance had become unacceptable; the time taken to return the results of a single-user simple query to list 10,000 spots increased from a barely acceptable 15s to 28s.

While we were prepared to move to higher-performance hardware, the continued, albeit reducing, rise in cardinality, with no sign of reaching a stable value, Figure 1, led us to look for an alternative database. Potential users of Influx should carefully consider the likely cardinality and associated hardware requirements for their own particular requirements.

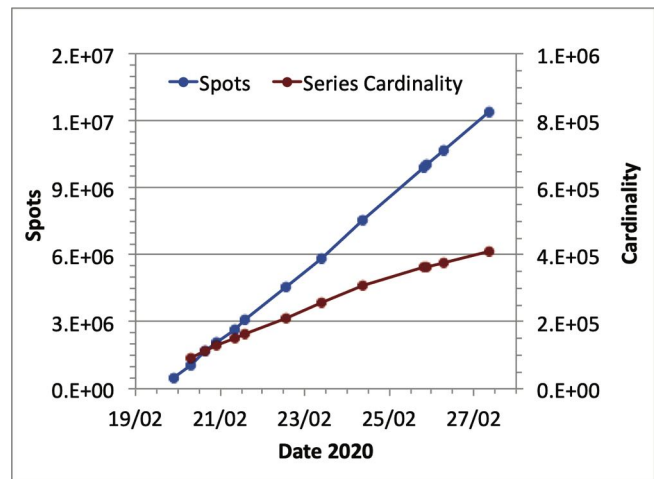


Figure 1. An almost linear rise in the number of spots in our Influx database gave rise to a cardinality that was still rising after 7 days. The cardinality : spots slope of 0.036 after 1 day had reduced to 0.018 after 7 days, but its continued rise would give unacceptable query times on the available hardware.

2.2 Why a TimescaleDB time series database

Further reading, our experience with Influx giving us a better insight into what was needed, and a readable, fair-minded comparison of Influx and TimescaleDB [12] led us to TimescaleDB. TimescaleDB is an extension to the very well established open-source relational database PostgreSQL (www.postgresql.org) that optimizes its performance with time series. It B-tree approach to

indexing data promised a better ability to handle the high cardinality inherent in WSPR data.

Importantly, as a relational database, TimescaleDB enables join operations both within individual tables (self-join) and between tables. This feature was not available in v1.7 Influx that we were using. Join operations allow queries such as: calculate the difference in SNR for the same sender at the same time in the same band for two different reporters.

3. IMPLEMENTING A TIMESCALEDB DATABASE

3.1 Data architecture overview

The data architecture of our August 2020 implementation is shown in Figure 2. In brief:

1. WsprDaemon client software on the reporting station's computer caches spot data, to avoid gaps during outages. Records comprising the normal WSPR fields are forwarded using HTTP Put to wsprnet.org and an extended set by FTP to logs.wsprdaemon.org, including estimates of local noise if enabled.
2. WsprDaemon server logs.wsprdaemon.org has two spots input paths: direct from WsprDaemon

users, and from all users from wsprnet.org via its API every two minutes. Preprocessing adds fields including numeric latitude, longitude and azimuth at the receiver as useful variables to plot when visualizing data [Section 5].

3. Spots from WsprDaemon users with additional fields are inserted into table wsprdaemon_spots and their noise data goes into table wsprdaemon_noise in the TimescaleDB database. Spots arriving via the wsprnet.org API go to table spots. Users connect to this database to query data. At current loads the server is well able to handle data insertion and user queries.
4. Data is also obtained from third parties via a scraper or an API. Currently we obtain the three-hourly geomagnetic disturbance index Kp from the US Space Weather Prediction Center.
5. As this data arrives at the server it is cached before being mirrored to a backup server logs1.wsprdaemon.org and inserted into a replica TimebaseDB database.

3.2 Hardware outline

High availability, sufficient memory to hold at least one

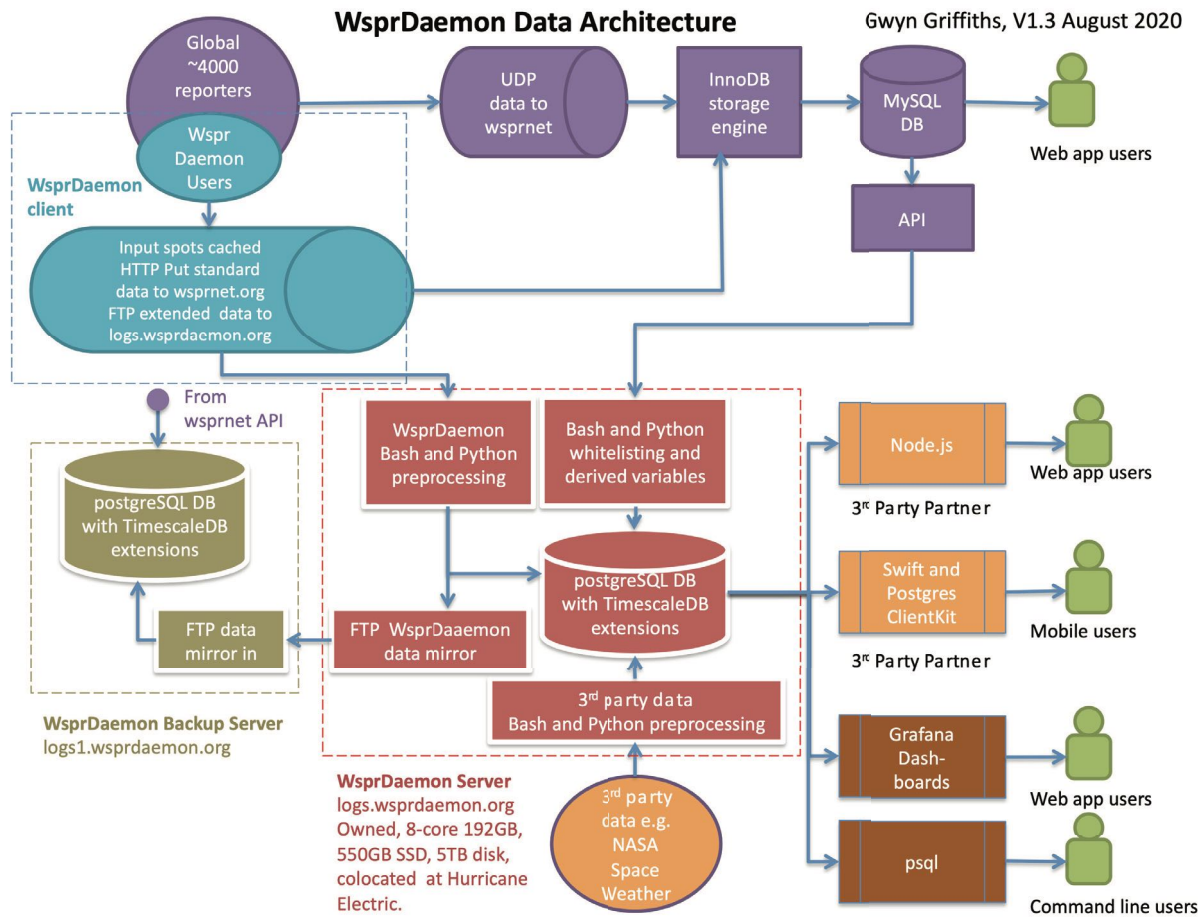


Figure 2. Diagram showing the data architecture of WsprDaemon, its data input pathways, its resilient design in having a server with a TimesacleDB database mirror data to a separate backup.

month of data in memory, appropriate processing power and good Internet connection are key requirements for the server logs.wsprdaemon.org. Currently these requirements are met with an owned Dell 8-core 192GB memory, 550GB SSD and 5TB disk machine. To ensure resilience against power outages and to provide a Gb Internet connection the server is collocated at a Hurricane Electric data center. A backup machine provides a hot-standby facility.

3.3 TimescaleDB installation and database set-up

Open Source, Community and paid-for Enterprise versions of TimebaseDB are available for a range of operating systems, including Windows, MacOS and several Linux distributions [14]. Installation was straightforward, edits to the configuration file were necessary to allow, among others, remote connections, access by password, and to set time zone to UTC. Port 5432 needs to be open on the router/firewall(s).

Initial set-up of a database is well described in the TimescaleDB documentation, although new SQL users will find the tutorial and examples pages at www.postgresqtutorial.com very helpful. The TimescaleDB extension need only be loaded once. An interactive program, *psql*, enables effective communication with the database for administrative tasks and to download the results of queries as csv files.

For our initial database we migrated WSPR data from Influx using a purpose-coded tool, Outflux [15]. If one is migrating to TimescaleDB from another PostgreSQL database use the *psql* command `pg_dump`.

On creation, database tables will simply be

PostgreSQL tables. The *create_hypertable* TimescaleDB extension converts a PostgreSQL table to a hypertable - these are interlinked 'chunk' tables, where a chunk covers a user-set time period. The choice of duration for a chunk needs to consider the incoming average data rate (say in MB/day), the size of the associated index, and the free memory available across all of the active hypertables and databases on the machine. It is advisable that all active chunks fit into 25% of memory.

Early experience showed the spots table from wsprnet.org, with derived variables, grew between 370 and 460MB/day from data, and 58 to 72MB/day from the index. For logs.wsprdaemon.org with 192GB memory we have set a conservative chunk size of 30 days rather than the 83 days implied by the 25% guide. This is to allow us to ensure that there is adequate CPU performance for queries spanning 30 days as our user base grows.

3.4 Query response times

There may be little value in providing examples of query response times without a full, detailed analysis of the load on the server. Nevertheless, given the minimal information in Figure 3, queries via third party app wsprd.vk7jj.com on all bands from OE9GHV for up to 50,000 spots over 24 hours returned 33,908 spot lines in 2.3 seconds, and for up to 500,000 spots over 7 days returned 275,784 spot lines in 13.6 seconds. The average time for 33 queries for all spots over the last hour for a variety of stations was 1.4 seconds. [Note: when preparing this figure logs.wsprdaemon.org was labeled WDI].

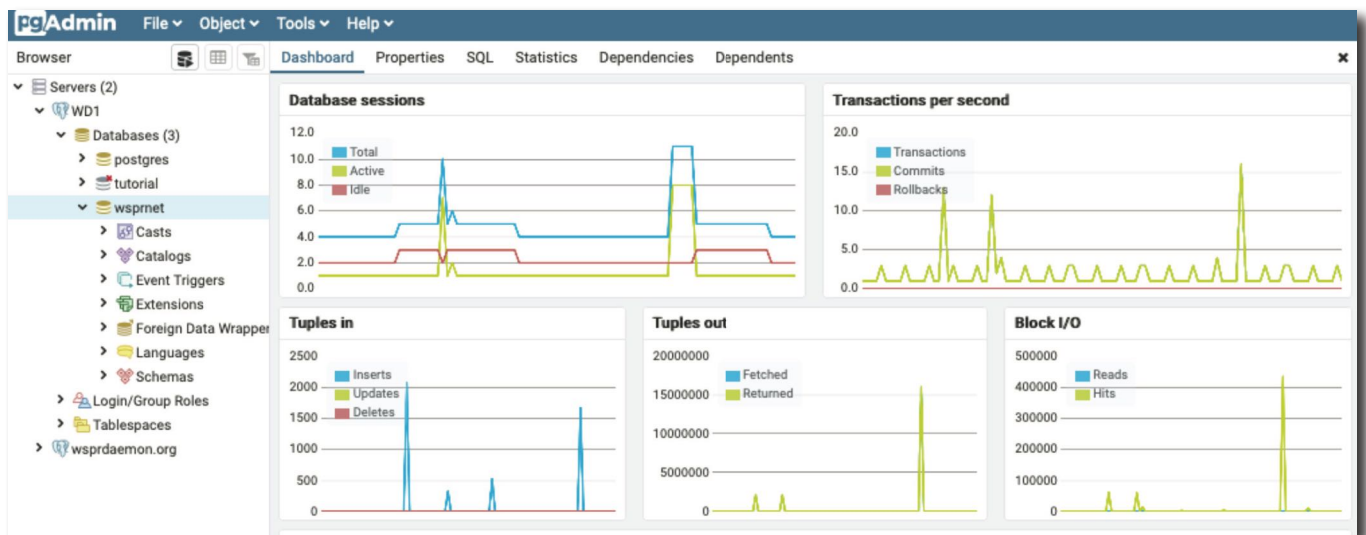


Figure 3. Screen shot of a *pg_admin* window showing statistics for the logs.wsprdaemon.org server. There are 1 to 8 active sessions; the *Tuples in* graph shows the Inserts from wsprnet.org; the *Tuples out* graph shows traffic from user queries. The two peaks at left were for queries via third party app wsprd.vk7jj.com for up to 50,000 spots over 24 hours received by OE9GHV on all bands, the queries returned 33,908 spot lines in 2.3 seconds, the right hand peak, with over 15 million Tuples out was for a query for all spots (275,784) from OE9GHV for 7 days.

4. GRAFANA VISUALISATION WITH TIMESCALEDB

Grafana (grafana.com) is a multi-platform open source data visualization package with extensive support for connections to numerous databases including PostgreSQL and TimescaleDB. In its terminology the user creates a 'Dashboard' comprising one or more 'panels', a panel may be a time series graph, a simple gauge, a digital readout, a map, or a host of other data representations. In this paper we focus on using Grafana time series graphs with WSPR data from our TimescaleDB database.

We have set up a few read-only example Dashboards at logs.wsprdaemon.org:3000 with access information available at wsprdaemon.org/grafana.html. However, users may want to install Grafana on their own machine to create custom Dashboards. Using Share → Export options the json code for our Dashboards can be saved and imported into a user's local Grafana instance as starting templates for their own versions.

4.1 Setting up Dashboards and panels

In summary, the steps to creating a data visualization site after installing Grafana are:

- Add a data source: there are currently pull-down options for over 20 sources, including database types, cloud repositories, and enterprise plug-ins. A single Dashboard may draw on data from one or more of these sources. Adding a data source

requires connection details: host IP address, userid, password, SSL mode, any connection time limits etc.

- Build a Dashboard: Starting with a new blank panel we add one or more queries if, as with TimescaleDB, the source is a database. After selecting the data source from its known list Grafana provides a comprehensive skeleton SQL Query Builder that the user modifies by deleting unwanted parts, using pull-down options to add specifics, e.g. to select SNR or distance as variables to plot, whether to use aggregate functions, e.g. to count the number of spots received in a time interval, or to form an average. For simple queries the Query Builder is sufficient; more complex queries can be written directly in SQL.
- Choose and customize visualization: Simple graphs of time series are excellent for many variables, but other Grafana options are very useful. Heat maps show the value of a derived quantity, e.g. a count of instances, as a color (effectively a z-axis) within a time period (a time bucket) and over a range of the y-axis variable (bins). Examples of heat maps include the number of spots in 20-minute time buckets and 1000km range bins, or in 15° bins of azimuth at the receiver.

5. INSIGHTS INTO PROPAGATION AND STATION



Figure 4. Grafana Dashboard that enables a user to compare distance statistics in 10 minute intervals (lower quartile, median, upper quartile) for two receivers and two bands. As in this example, a comparison may be between two different receivers on the same band, but can also be used for the same receiver on two different bands. Pull down options at top right set the time span and the user can save or export the plot, or save the data in csv or json formats.

PERFORMANCE

In this section we show three graphical examples to illustrate the value of bringing together a time series database and Grafana visualization. A common feature of most of these Grafana Dashboards is that by using template variables the user can easily select, using pull-down options, the stations, bands etc. to plot.

5.1 Insights into propagation - simple time series plots

Figure 4 shows a Dashboard that enables us to look at simple statistics with time of the distances between receiver and transmitters for two receivers and for two bands. The statistics here are the lower and upper quartiles and the median. The two receivers may be the same, but the bands may differ, or, as in this example, we can compare two different receiver stations on the same band.

In this example on 40m, AI6VN/KH6 is on Maui,

Hawaii, while KK6PR is in central Oregon. For AI6VN/KH6 the lower quartile represents the distance to the west coast of N. America. Propagation over that path is almost continuous, except for gaps between 2000 and 0000UTC. About 0200UTC the band opens to the US South- then Mid-West (upper quartile and mean very close), followed very soon by a step in the upper quartile from propagation extending to the Eastern Seaboard. About 1300UTC the path to the Eastern Seaboard closes. The hours until about 1800UTC (spanning sunrise) show more day-to-day variability: on 29 July propagation was open to Australia (11,000km), but not on 30 or 31 July, the path to South Africa (19,000km) opened each day, albeit very briefly on the 31st.

For KK6PR the overall pattern is similar, but the band starts to open to the Eastern Seaboard 2-3 hours



Figure 5. An example Grafana Dashboard that combines simple time series graphics from three sources with heatmap representation of data. The data spans four days for 40m from KA7OEI-1, a KivisDR at the Northern Utah SDR site. The top panel shows the number of spots received in 20-minute intervals in green, with three-hourly estimates of the planetary geomagnetic disturbance index Kp in yellow. The second panel is a heatmap of the number of spots in 20-minute intervals within 1000km distance bins out to 20,000km. The third panel shows azimuth at the receiver in 15° and 20 minute bins. The bottom panel shows the c2_FFT and RMS noise estimates with a distinct diurnal pattern.

earlier, at just before 0000UTC, and it is open for longer. The increased upper quartile distances spanning sunrise were from path openings to Japan, New Zealand and Australia.

While undoubtedly useful, these simple graphs require the user to consult listings of spots received to fully interpret the paths involved. Presenting the data as heat maps of distance and azimuth at the receiver helps (admittedly with ambiguity over short and long path).

5.2 Displaying other data and heat maps alongside WSPR spots

Figure 5 is an example Grafana Dashboard combining simple time series graphics from three sources with heat maps. The data spans four days for 40m from KA7OEL-1, a KiwiSDR at the Northern Utah SDR site [16]. The top panel shows the number of spots received in 20-minute intervals in green, with three-hourly estimates of the planetary geomagnetic disturbance index Kp in

yellow. While there is day-to-day variation, there is a daily broad minimum spanning local noon. The heat map in the second panel shows a distinct daily periodicity in the number of stations received at a distance of 2500-4000km – from the Eastern Seaboard – an interpretation helped by the azimuth heat map in the third panel. However, reference to a list of received spots is still needed to check that the first evening DX is from South Africa, then New Zealand, followed by Australia and Europe. But on some days, southern European and North African stations appear during early evening. The bottom panel shows the c2_FFT and RMS noise estimates with a distinct diurnal pattern suggesting that, above the troughs, the noise was being propagated in rather than from a local source.

5.3 Visualizing derived data - SNR difference

As postgresQL, the foundation of TimescaleDB, is a



Figure 6. As postgresQL is a relational database that allows self-joins a query in Grafana can return data derived from one or more receivers. This Dashboard shows the SNR difference between G3ZIL and G4HZX on 40m, for spots from the same sender at the same time, as a time series scatter plot in the top panel, as median and lower and upper quartiles in the second panel and as a heatmap in the third. Heat maps of spot distance and azimuth at G3ZIL help interpret the SNR difference.

relational database it is easy to use join queries, across different tables and databases but also within a table (a self-join). Grafana's query builder does not have sufficient options to construct a self-join but it is straightforward to write the necessary PostgreSQL queries. Figure 6 shows a Dashboard designed to derive and plot the SNR difference between any two selected receivers where they spotted the same sender at the same time on the same band. The SNR difference is shown as a scatterplot, as median, lower and upper quartile in 20-minute intervals and as a heat map. Heat maps of the spot distance and azimuth at the first receiver are added to help interpretation.

In this example G3ZIL and G4HZX are 110km apart, G3ZIL generally has higher local noise than G4HZX, and the antenna is a vertical at G4HZX and a pair of low inverted V dipoles at G3ZIL. The statistics and heat map panels are the most useful for seeing the SNR differences. Clearly, a snapshot over a few minutes or even a few hours would not convey the complexity of the changes in SNR with time.

While there are undoubtedly short term variations there are clear diurnal (daily) variations as well as longer-term changes. Comparing the SNR difference and distance heat maps show higher SNR at G4HZX during hours of darkness with 40m open to the west, to North America, as seen in the azimuth heat map. The vertical out-performs the low dipoles for signals with low arrival angles. Conversely, spanning local noon, the low dipoles give a higher SNR than the vertical for signals from 0–1000km arriving at higher angles.

6. CONCLUSION

Implementing a database of our own has reinforced our opinion that wsprnet.org does a very good job at data collection and fulfilling simple web-based queries. With wsprnet.org as the primary data collector we have shown how a time series database and a visualization package can provide a richer set of data queries and graphical output.

Along the way we learnt about the strengths and weaknesses of time series databases. While Influx's documentation was excellent and a working system was easy to implement it took some time for its drawbacks in dealing with the very high cardinality of WSPR data to become obvious.

TimescaleDB's design better suits WSPR data, although on a machine with limited memory query response times into 10s of seconds occur for data not in the current time chunk. Our affordable solution has been to move to a server with 192GB of memory, sufficient to hold 30 days of WSPR data in memory, and with sufficient SSD and disk storage to provide (albeit slower,

but still online) access to older data when required.

Grafana comes with built-in connection paths to TimescaleDB and other data sources. Its own visualization options for time series graphs and heat maps are an excellent starting point for examining WSPR spot data, as illustrated in this paper. Plug-ins from a growing community of third-party sources provide an even wider range of options such as forms of Hovmöller diagrams where the y-axis is hour of the day and the x axis time, and a range of maps.

Finally, there is clear potential from adding non-WSPR data. We have only scratched the surface in this regard, sourcing geomagnetic disturbance measurements from the US Space Weather Prediction Center. Through the means outlined in this paper we look forward to users making far more of the terrific resource that is the global WSPR community and its data.

Acknowledgment

We are grateful to Rick Whal (KK6PR), Clint Turner (KA7OEI) and Nigel Squibb (G4HZX) for permission to refer to their data, to constructive discussion and comment from WsprDaemon users at their weekly teleconference and to Glenn Elmore for his support, advice and indefatigable testing.

REFERENCES

- [1] github.com/garymcm/wsprnet_api and derivatives such as github.com/dl2sba/WsprNet
- [2] Taylor, J. and Walker, B., 2010. WSPRring around the world. *QST*, 94(11), 30-32.
- [3] www.jimlill.com:8088/today_int.html
- [4] wspr.pe1itr.com/
- [5] mardie4.100webspaces.net/wspr/
- [6] wspr.aprsinfo.com/
- [7] wspr.vk7jj.com/
- [8] apps.apple.com/us/app/wspr-watch/id532487317
- [9] wspr.fggs.de
- [10] Robinett, R., 2019. WsprDaemon: A low cost, high performance, all band WSPR decoding system. *ARRL / TAPR Digital Communications Conference*, Detroit, 2019. www.youtube.com/watch?v=nHVN8oUUtlE
- [11] Struckov, A., Yufa, S., Visheratin, A.A. and Nasonov, D., 2019. Evaluation of modern tools and techniques for storing time-series data. *Procedia Computer Science*, 156, pp.19-28.
- [12] docs.influxdata.com/influxdb/v1.8/introduction/get-started/
- [13] blog.timescale.com/blog/what-is-high-cardinality-how-do-time-series-databases-influxdb-timescaledb-compare
- [14] docs.timescale.com/latest/getting-started/installation
- [15] github.com/timescale/outflux
- [16] www.sdrutah.org/

Packet Compressed Sensing Imaging (PCSI): Robust Image Transmission over Noisy Channels

Scott Howard, Grant Barthelmes, Cara Ravasio,
Lisa Huang, Benjamin Poag, & Varun Mannam

Department of Electrical Engineering, University of Notre Dame

Abstract

Packet Compressed Sensing Imaging (PCSI) is digital unconnected image transmission method resilient to packet loss. The goal is to develop a robust image transmission method that is computationally trivial to transmit (e.g., compatible with low-power 8-bit microcontrollers) and well suited for weak signal environments where packets are likely to be lost. In other image transmission techniques, noise and packet loss leads to parts of the image being distorted or missing. In PCSI, every packet contains random pixel information from the entire image, and each additional packet received (in any order) simply enhances image quality. Satisfactory SSTV resolution (320x240 pixel) images can be received in \approx 1-2 minutes when transmitted at 1200 baud AFSK, which is on par with analog SSTV transmission time. Image transmission and reception can occur simultaneously on a computer, and multiple images can be received from multiple stations simultaneously - allowing for the creation of “image nets.” This paper presents a simple computer application for Windows, Mac, and Linux that implements PCSI transmission and reception on any KISS compatible hardware or software modem on any band and digital mode.

Contents

1	Introduction	2
1.1	Radio Image Transmission Modes Comparison	2
1.2	PCSI's Predecessors	4

This work is licensed under a Creative Commons “Attribution 4.0 International” license.



Presented at ARRL/TAPR DCC 2020

2	The Magic in the Math	5
2.1	Compressed Sensing Imaging	5
2.2	Chroma Compression and Color Depth	6
3	Implementation: The PDP Specification	7
3.1	What is the PDP?	7
3.2	PDP Specification Scope	7
3.2.1	AX.25 Framing	8
3.2.2	SSDV-style Framing	8
3.2.3	Framing Comparison	9
3.3	PDP Specification Details	10
3.3.1	Packet Payload Preparation	10
3.3.2	Pseudo-random Number Generation for Picking Pixels . .	12
3.3.3	PCSI Payload base91 Encoding	12
3.4	Reconstructing PCSI Images	13
4	Future Work	13

1 Introduction

Packet compressed sensing imaging (PCSI) is a solution to the technical challenge of transmitting a complete image to multiple receivers over a channel where each receiving station may miss different parts of the transmission due to channel noise. PCSI is implemented in a way such that a low-power micro-controller (e.g., Arduino) is capable of transmitting the image from challenging and remote environments (e.g., high altitude balloon).

PCSI achieves these capabilities by using a technique known as ‘compressed sensing imaging,’ a computational method that allows one to reconstruct a complete image when only given a random selection of pixels from that image. Therefore, even after receiving only a single packet of random pixels, the receiver can begin to reconstruct the complete original image. Each additional packet received further increasing image quality. PCSI is, therefore, robust against packet loss.

The initial version of an open-source software tool¹ that implements sending and receiving PCSI in Python for Windows, macOS, and Linux is illustrated in Figure 1. The image experienced $\approx 50\%$ packet loss, yet the image was fully reconstructed (bottom frame). In other techniques, 1/2 of the image would be completely missing.

1.1 Radio Image Transmission Modes Comparison

When transmitting an image, the operator is presented with many possible methods and must select an image transfer method that best suits the application. The following is a quick summary of popular techniques.

¹<https://maqifrnswa.github.io/PCSI/>

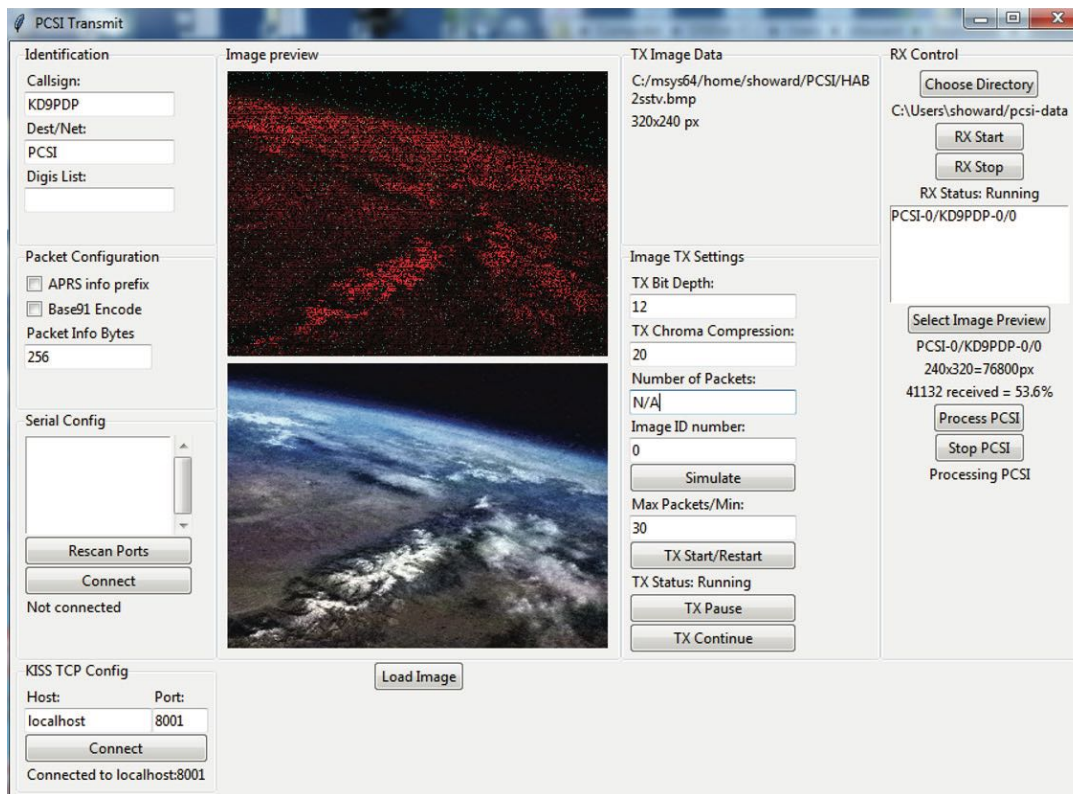


Figure 1: Demonstration of PCSI using pcsiGUI, an open-source tool written in Python for Windows, macOS, and Linux computers. Only 53.6% of the packets were received (indicated by the red pixels in the top frame), yet the entire image can be reconstructed (bottom image). The software tool connects to any KISS compatible TNC or software modem and is capable of simultaneous (duplex) receiving/transmitting, and capable of receiving multiple images from multiple stations simultaneously.

Analog transmission can either encode an image's color values using frequency modulation (as in SSTV) or amplitude modulation (as in radiofax). In all cases, each point is transmitted sequentially; therefore, channel noise and signal loss leads to color distortion and pixel loss.

Digital techniques divide an image into individual pixels and transmit the quantized (i.e., represented in binary) values of each pixel's color channels as groups of data in packets. Each packet contains additional information that allows the receiver to check if the received packet was distorted during transmission (e.g., a "checksum" as used in AX.25), and more recently, can correct for some errors using forward error correcting (FEC) codes (as used in FX.25). In both cases, unrecoverable errors lead to complete loss of pixel information for parts of the image.

Digital techniques also allow for the use of compression to send image data efficiently in fewer bits. A great example of this is slow scan digital video² (SSDV). SSDV was developed by Philip Heron with the UK's high altitude ballooning community to transmit high-quality images. The technique takes images, converts them to a specific type of JPEG file, and transmits sections of the JPEG file in packets using FEC. This method transmits high quality images successfully; however, JPEG encoding and FEC generation is prohibitively complex for low-memory microcontrollers, and signal loss leads to missing parts of the image.

Hence, there is a need to develop a computationally simple image transfer method that is robust to signal loss and channel noise.

1.2 PCSI's Predecessors

In developing PCSI, we found that the idea of transmitting limited information over a noise channel while "filling in the blanks" at the receiving end were present in at least two other technologies.

APRS Vision System: At the 1997 DCC, Bob Bruninga (WB4APR) proposed the "APRS Vision System."³ That approach was an attempt to relay imaging information over the APRS network. The idea was to transmit APRS packets that contain increasingly higher amounts of spatial information content. The first packet contained an extremely low quality image, and each subsequent packet doubled the resolution. When the receiver feels the image is "clear enough" (i.e., the receiver can "fill in the gaps" in the image), the receiver can tell the sender to stop. This way the receiver can possibly control or react to the image before the whole image became "clear." However, this system required that every previous packet had been received perfectly for subsequent packets to be used, and the receiver needed to communicate with transmitter to indicate missed packets and to stop. Any missed packet caused the whole system to fail from that point forward.

Hellschreiber: This mode, developed in the mid 20th century, is a fasci-

²<https://ukhas.org.uk/guides:ssdv>

³<https://www.tapr.org/pdf/DCC1997-APRSvision-WB4APR.pdf>

nating approach that enabled robust transmission of text over noisy channels.⁴ Each character of text to be transmitted is converted in to a 7x7 pixel image of that character. Each character is then transmitted using extremely narrow bandwidth transmission (e.g., on-off CW keying). The receiver takes the received data and reconstructs the image. The operator then “reads” the resulting image text. Fundamentally, the operator’s trained character recognition neural network (i.e., their brain) does pattern recognition to “fill the the gaps” in the image that was caused by channel noise and signal loss. The entire message can therefore be received even if some pixels were not received or if noise distorted some pixels.

2 The Magic in the Math

While it may seem “too good to be true” to be able to fully reconstruct an entire image using only a few randomly selected pixel values, PCSI is made possible via the magic in the math. PCSI accomplishes this using two techniques: compressed sensing imaging and chroma compression.

2.1 Compressed Sensing Imaging

The mathematical concept of compressed sensing imaging exploits the fact that data can often be represented in “sparse” domains. By “sparse,” we mean that “most of the values are zero.” For example, compare a photograph of a blooming garden during the day to a photograph of the night sky. The image of the garden will have lots of red, green, and blue color, and therefore, many non-zero values for the R, G, and B channels in the image. The night sky is mostly empty, with a few stars, planets, and the moon. The night sky image, therefore, has many pixels that have zero value in the R, G, or B channels. We would say the garden image is not sparse while the night sky is sparse.

The magic happens when you convert the image from the spatial domain (i.e., a photograph) to another “domain.” For example, you can perform a mathematical operation on an image called the discrete cosine transform (DCT). The resulting DCT “image” ends up containing all the same information from the original image, just arranged in a different way. You can then do an inverse discrete cosine transform (IDCT) on the “DCT image,” and you will restore the original image accurately. Why would we want to take the DCT of an image? We do this because practically all images are “sparse” after taking the DCT.⁵ In other words, most of the values of the DCT of an image are zero! Both the garden and night sky look sparse after taking the DCT!

How can we use the fact that most images are sparse after taking the DCT? This is the magic of PCSI. You ask the computer to find the DCT of an image such that:

⁴<https://www.nonstopsystems.com/radio/hellschreiber-function-operation.htm>

⁵Technically, the image is sparse in the DCT domain because cameras typically massively “oversampling” an image.

1. The values for the DCT it finds has a lot of zeros in it (i.e., is sparse).
2. When you reconstruct the image using the values for the DCT the computer found, the resulting real image closely matches whatever values of the pixels that were received

Mathematically, you are asking the computer to do something similar to “basis pursuit denoising” to find the simplest image (i.e., the fewest non-zero DCT values) that also matches the pixels that you have received so far. You can do that by using a computer program to find the values of the DCT of the final image (\mathbf{X}) such that it gives you the smallest value of the following expression:

$$\sum_n |\text{IDCT}(\mathbf{X})_n - b_n|^2 + C \sum |\mathbf{X}| \quad (1)$$

where b_n is the value of the n th pixel that was received, and C is a scaling factor (typically in the range of 3-5). The first term is the sum of the squared error between the values of the pixels in the reconstructed image and the value of the pixels that were actually received. Ideally, this will be zero. The second term is the L1 norm, which is adding up all the values of \mathbf{X} , and minimizing that term is a good method for finding a sparse \mathbf{X} . Once you find \mathbf{X} , you can take the IDCT of \mathbf{X} to find the reconstructed image.

While the current PCSI reference implementation does use the above basis pursuit technique, it is just one of many ways to reconstruct an image from a collection of random pixels. PCSI actually does not require any particular reconstruction algorithm as there may be variations in the method that yield superior results.

2.2 Chroma Compression and Color Depth

PCSI image transmission speed is increased by reducing the bit depth of an image and by utilizing chroma compression. These techniques are described below.

- **Bit Depth:** Reducing an image from 24 bit color (8-bit in each of R, B, G) to 12 bit color (or any other color depth) is trivial, may be acceptable for many applications, and therefore is an option in PCSI.
- **Chroma compression:** The human eye has 20-times high density of rods (greyscale photoreceptors) than cones (color photoreceptors), and therefore detects greyscale with better spatial resolution than color. It is therefore not necessary to transmit the same resolution for both luma (brightness) and chroma (color information). JPEG exploits this by representing an image in the YCbCr color space and sub-sampling the chroma channels (Cb and Cr) relative to the luma channel (Y). PCSI uses the same general concept and achieves chroma compression by sending a combination of full-color (YCbCr) and greyscale-only (Y) pixels in each packet. This step leads to the receiver receiving more Y channel pixels than Cb

and Cr channel pixels. Each channel, separately, undergoes the compressed sensing basis pursuit to reconstruct the original channel, and the channels are then converted back to RGB. The resulting image appears to have much higher quality for the same number of packets.

3 Implementation: The PDP Specification

PCSI requires packet payloads such that each individual payload contains all the information necessary to reconstruct a single image. To achieve that, the pseudo-random datagram payload (PDP) specification has been developed. Version 1.0.0 is described below.

3.1 What is the PDP?

PDP is a specification for the payloads of data packets such that each packet contains all the information needed to reconstruct a single image. An image is then transmitted as the collection of datagrams (i.e., packets in a connectionless network). Unlike other packetized image transmission formats, the pixels contained in a packet are selected in a pseudo-random, yet deterministic, way. This allows images to be restored using compressed sensing techniques regardless of packet loss.

3.2 PDP Specification Scope

The PDP spec merely defines the packet payload for the transmission of a single image. It can be used in any packet protocol or digital mode. Framing is independent of the specification. This allows for the separation of a “session” (consisting of a sending station sending one or more images) from the minimal content required for a single image. The “session” information is in the framing; the image information is in the payload. The payload is designed to be similar to SSDV.

For example, a PDP can be placed as the payload in:

- AX.25 amateur radio packets. Transmitted using any mode (e.g., AFSK, PSK, etc.) Therefore it is compatible with APRS, TNCs, digipeaters, software modems (direwolf, fldigi, soundmodem, etc.). Example implementation is in Section 3.2.1.
- SSDV-style framing done in a KISS TNC compatible way. Example implementation is in Section 3.2.2.
- UDP or TCP (although the benefits of PCSI provide more benefit to multicast UDP packets than connected TCP packets).

Name	Size (bits)	Description
Flag	8	HDLC flag '0x7E'
Dest. Address	56	Callsign of intended receiver OR alias of an image net, encoded following AX.25 spec. 'PCSI' recommended for general use.
Source Address	56	Sender's callsign encoded following AX.25 spec. 'PCSI' recommended for general use.
Digi Addresses	$d \times 56$	d optional digipeater addresses, encoded following AX.25 spec.
Control	16	'0x03F0' indicating UI frame with no response requested, and no layer 3 implemented
PDP	$N \times 8$	PDP data , $N \leq 256$
FCS	16	CRC-CCITT
Flag	8	HDLC flag '0x7E'

Table 1: Example AX.25 framing that could be used for PCSI.

Offset (bytes)	Name	Size (bytes)	Description
0	Flag	1	HDLC flag '0x7E'
1	Packet Identifier	1	ASCII 'v' = '0x76'
2	Callsign	4	Base-40 encoded callsign following SSDV encoding convention
6	PDP	$N \leq 256$	PDP data
N+6	Checksum	2	CRC-CCITT
N+8	Flag	1	HDLC flag '0x7E'

Table 2: Example SSTV-style framing that could be used for PCSI.

3.2.1 AX.25 Framing

While not part of the PDP spec, an example of using AX.25 UI framing⁶ of a PDP is given in Table 1. This is easily compatible with existing TNCs. This framing adds at least 20 bytes of overhead.

3.2.2 SSDV-style Framing

While not part of the PDP spec, a simple session framing of PDP can be done in a way that is compatible with existing KISS hardware and software TNCs. An example is seen in Table 2. This example framing is designed to be easy to use with any KISS TNC. One would simply send the concatenated "Packet Identifier + Callsign + PDP" and let the TNC add the flags and do the checksum. This framing adds at least 9 bytes of overhead.

⁶<https://www.tapr.org/pdf/AX25.2.2.pdf>

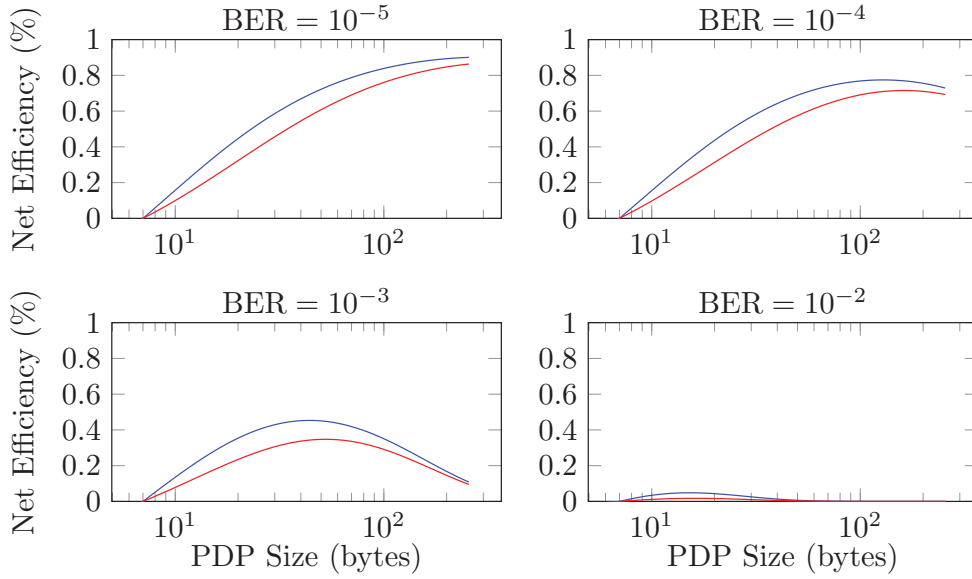


Figure 2: Comparison of net bit efficiency versus PDP payload size for both SSDV-style and AX.25 framing.

3.2.3 Framing Comparison

Both AX.25 framing and SSDV-style framing can be used. AX.25 is more powerful as it can leverage existing packet radio infrastructure at the cost of larger overhead. However, if channel bit error rate (BER) is high (as is common in longer-distance HF modes), smaller packets are more likely to be successfully received. The lower overhead SSDV-style framing may be superior in this case. This trade-off is explored in Figure 2. The net efficiency (percent of each transmitted bit that will successfully transmit pixel-level image information to the receiver) is calculated as the product of the probability that the entire packet will be received properly and the percentage of bits in a packet that correspond to pixel information.

$$\text{Net Efficiency AX.25} = \frac{x-7}{x+20} \times (1-\text{BER})^{8x+160} \quad (2)$$

$$\text{Net Efficiency SSDV-Style} = \frac{x-7}{x+9} \times (1-\text{BER})^{8x+72} \quad (3)$$

$$(4)$$

where x is the total PDP length in bytes and the term $x-7$ comes from the fact that the PDP has a 7 byte header as described in 3.3.

Results from Figure 2 give guidelines for ideal PDP length. First, find the approximate BER by estimating packet loss for an AX.25 packet with a 256 byte payload using the equation:

$$\text{BER} = 1 - (1 - L/100)^{1/2192} \quad (5)$$

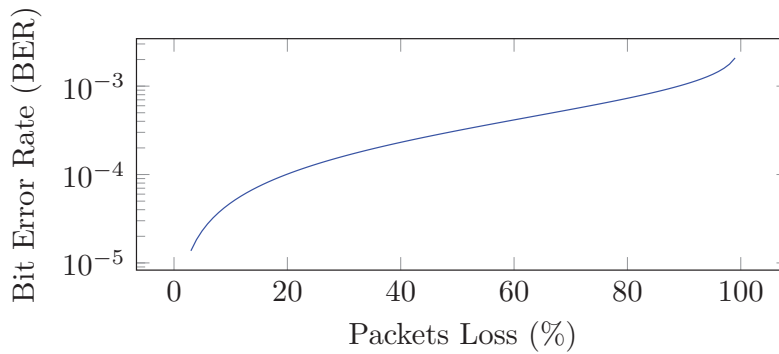


Figure 3: BER vs packet loss for AX.25 frames with 256 byte payloads.

where L is packet loss in percent. If the packet loss percentage is known (or can be estimated), BER can be found using Equation 5, which is depicted by Figure 3.

Now based on the estimated BER, you can choose the appropriate framing style and PDP size (referring to Fig. 2):

- For low BER $\leq 10^{-5}$ Environments: Framing style does not matter that much, and payloads should be the full 256 bytes long.
- For BER $\approx 10^{-4}$ Environments: Framing style does not matter that much, and payloads should be 130 bytes long.
- For BER $\approx 10^{-3}$ Environments: SSDV-Style framing increases efficiency (and speed) by $\approx 25\%$ compared to AX.25. Payloads should be 40-50 bytes long.
- For BER $\approx 10^{-2}$ Environments: AX.25 is practically unusable; SSDV framing will barely be usable. Payloads should be 10-11 bytes long.

3.3 PDP Specification Details

Since each packet contains information of the whole frame, each packet **MUST** be the same size of every packet in an image (same number of pixels per packet). Total packet size is determined by the framing protocol used. For example, AX.25 packet payloads are 256 bytes by default. The payload contains the following data transmitted in order as described by Table 3.

3.3.1 Packet Payload Preparation

Given a bit mapped image to transfer, follow the following procedures

1. Using a pseudo-random number generator (see Section 3.3.2), generate the sequence of pixels to be transmitted.

Offset (bits)	Name	Size (bits)	Description
0	Image ID	8	Identifies unique images within a PCSI session. (uint8)
8	Rows	8	Number of lines in the image divided by 16. (4096 lines max, uint8)
16	Columns	8	number of columns in the image divided by 16. (4096 columns max, uint8)
24	Packet ID	16	used as the starting point of the pseudorandom pixel list. (uint16)
32	Number of YCbCr Pixels	8	Number of full color pixels transmitted in this packet. (uint8)
40	Color depth	8	Color depth encoded as (color depth/3 -1). e.g., 24bit color = 7. This only uses 3 bits, so there are 5 bits available for future use. (uint8)
48	YCbCr Pixel Data	(Number of YCbCr Pixels) * (Color bit depth)	Full color (YCbCr) pixels listed first as a binary stream. For example, if color is transmitted as 12-bit color, each pixel is 12-bits long with the first 4 corresponding to the Y channel, the next four corresponding to the Cb channel, and the final 4 corresponding to the Cr channel.
48 + (YCbCr Pixels) * (Color bit depth)	Y-only Pixel Data	N	Black and white (Y only) pixels follow in a binary stream of Y values encoded as a uint with the same bit depth as a single channel of the YCbCr pixels.
	Zero padding	Z	Zero padding for byte alignment as needed.

Table 3: PDP Specification Version 1.0.0

2. Given the number of bits available in the payload (e.g., AX.25 UI frames have 256 bytes minus 7 bytes of image info equals 1992 bits total), the desired chroma compression level, and the desired color bit depth to transmit, determine the list of pixels to transmit that will be full color and solely back and white.
 - (a) All packets consist of the same number of pixels (e.g., every packet for an image has exactly 25 YCbCr pixels and 75 Y only pixels for a total of 100 pixels. You can choose whatever numbers you want, as long as they are the same for every packet of the image).
3. Prepare the packet payload
 - (a) Convert full color pixels to YCbCr per ITU-T T.871⁷ and black and white only pixels to Y as per the same spec.
 - (b) If color bit depth is being reduced, approximate the value to be transmitted using rounding. For example, the 8 bit number 200 will be represented as the 4 bit number $\text{round}(200/255*15)=12$.

3.3.2 Pseudo-random Number Generation for Picking Pixels

Compressed sensing imaging requires that the measurements are uncorrelated in the sparse domain that is used to reconstruct the image. Taking random samples ensures this condition, however, both the transmitter and receiver need to know which pixel values correspond to which pixels in the image. To do this, PCSI uses a Linear Congruential Generator⁸ as a deterministic pseudo-random number generator using GCC's default coefficients (modulus = 2^{31} , $a = 1103515245$, $c = 12345$, starting with a seed = 1). The pseudo-random number generator is then used with the modern Fisher Yates shuffle algorithm⁹ to generate a random list of the pixels to be sent. See reference code for details. This approach will allow all receivers and the transmitter to generate identical lists of order that pixels will be transmitted. Since every packet has the same number of pixels, the packet ID will give the receiver the starting pixel number from which the list of pixels received in the packet can be extracted.

Pixels are indexed column-first as seen in C, not row first as is typically done in Python. You therefore have to transpose a matrix before selecting and assigning pixels if you are working in Python.

3.3.3 PCSI Payload base91 Encoding

If you are transmitting over channels that only allow printable ASCII text, the entire PDP can be converted to base91 as described below. This is a combination of APRS base91 and basE91.¹⁰ Compared to basE91, the method used in PCSI is simpler and deterministic at the cost of slightly more overhead.

⁷https://en.wikipedia.org/wiki/YCbCr#JPEG_conversion

⁸https://en.wikipedia.org/wiki/Linear_congruential_generator

⁹https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle

¹⁰<http://base91.sourceforge.net/>

While there are 13 bits or more to convert, read in 13 bits Convert those 13 bits to two ASCII bytes using $\lfloor \text{bits}/91 \rfloor + 33$ for first and $[\text{bits}\%91+33]$ for the second byte. Next, if there are fewer than thirteen and more than seven bits available (the end of the stream), read in and zero pad (to the right, i.e., least significant bits) the remaining bits so that there are 13 bits total. Convert those 13 bits to two ASCII bytes using $\lfloor \text{bits}/91 \rfloor + 33$ for first and $[\text{bits}\%91+33]$ for the second byte If there are 6 or few bits remaining: Read in and zero pad (to the right, i.e., least significant bits) the remaining bits so that there are 6 bits total. Convert those 6 bits to one ASCII byte using $\text{bits}+33$.

3.4 Reconstructing PCSI Images

There is no specification or standard on how to reconstruct the images. Users can experiment with different methods and find what is appropriate. The reference implementation follows these steps:¹¹

1. Decode all the pixel values and pixel numbers from as many packets as have been successfully received.
2. For each color channel (Y, Cb, Cr), use OLW-QN for basis pursuit¹² to find the discrete cosine transform (DCT) coefficients that best fit the received data and minimizes the L1 norm. This is the key to compressed sensing!
3. After finding the DCT coefficients, use the inverse DCT to generate the color channels for the image.
4. Convert from YCbCr to RGB, and save the image.

4 Future Work

PCSI is available for use any band and KISS compatible TNC or software modem. Now that it has been demonstrated, some additional features can be explored:

- **Low-power micro-controller transmission client:** PCSI transmission is computationally simple enough to be performed on low-memory micro-controllers. High-altitude balloons would benefit from integrating PCSI transmission with Arduino radios such as the HamShield.¹³
- **PCSI aggregation server:** Since different stations can receive different packets, and increasing the packets increases image quality, a centralized server can be used to aggregate packets to improve image quality. This system would be similar to the Automatic Picture Relay Network (APRN¹⁴)

¹¹based on <http://www.pyrunner.com/weblog/2016/05/26/compressed-sensing-python/>

¹²https://en.wikipedia.org/wiki/Limited-memory_BFGS#OWL-QN

¹³<https://inductivetwig.com/>

¹⁴<http://www.aprs.org/aprn.html>

and to what is done with SSDV¹⁵ to receive images from high-altitude balloons that move out of range of the original receiving system.

- **Integrate PCSI with APRS:** While transmitting entire PCSI images over APRS channels would severely strain the network, APRS could be leveraged to announce ongoing transmission or upcoming “ImageNets” on non-APRS frequencies. APRS frequency objects can be transmitted following the conventions of the Automatic Frequency Reporting System¹⁶ (AFRS) and APRS Local Frequency Info Initiative.¹⁷

¹⁵<http://ssdv.habhub.org/>

¹⁶<http://www.aprs.org/afrs.html>

¹⁷<http://www.aprs.org/localinfo.html>

List of Terms

Notation	Meaning
AFSK	Audio Frequency Shift Keying
APRS	Automatic Packet Reporting System
ASCII	American Standard Code for Information Interchange
AX.25	Amateur X.25
BER	Bit Error Rate
CRC	Cyclic Redundancy Check
CW	Continuous Wave
DCC	Digital Communications Conference
DCT	Discrete Cosine Transform
FCS	Frame Check Sequence
FEC	Forward Error Correction
FX.25	Extension to AX.25 with FEC
GCC	GNU Compiler Collection
GUI	Graphical User Interface
HDLC	High-Level Data Link Control
HF	High Frequency
IDCT	Inverse Discrete Cosine Transform
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
KISS	Keep It Simple, [Silly]
PCSI	Packet Compressed Sensing Imaging
PDP	Pseudo-random Datagram Payload (PDP)
OLW-QN	Orthant-Wise Limited-memory Quasi-Newton
RGB	Red, Green, Blue
SSDV	Slow Scan Digital Video
SSTV	Slow Scan Television
TCP	Transmission Control Protocol
TNC	Terminal Node Controller
UDP	User Datagram Protocol
UI	User Interface
YCB _C R	Luma, blue-difference and red-difference chroma components

Digital signal processing: I2S in ESP32

Anthony LE CREN, KF4GOH
f4goh@orange.fr

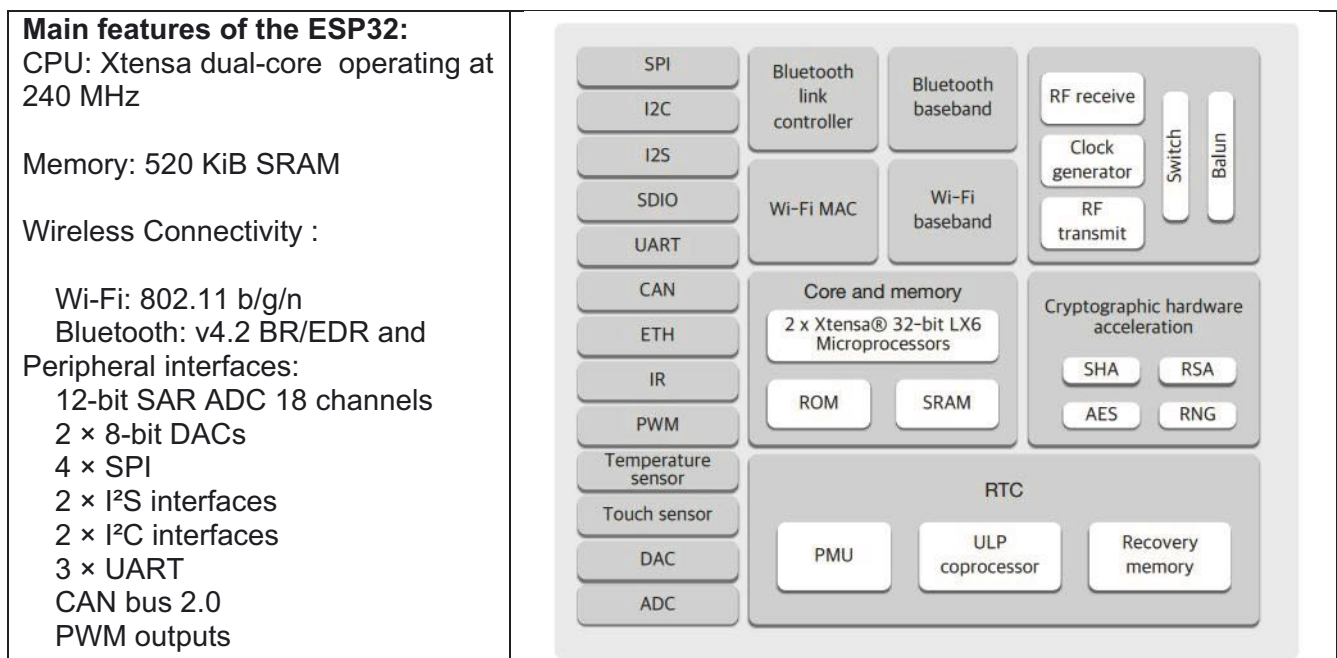
Abstract

How to decode and encode RTTY with an Espressif ESP32 and I2S (Inter-IC Sound)

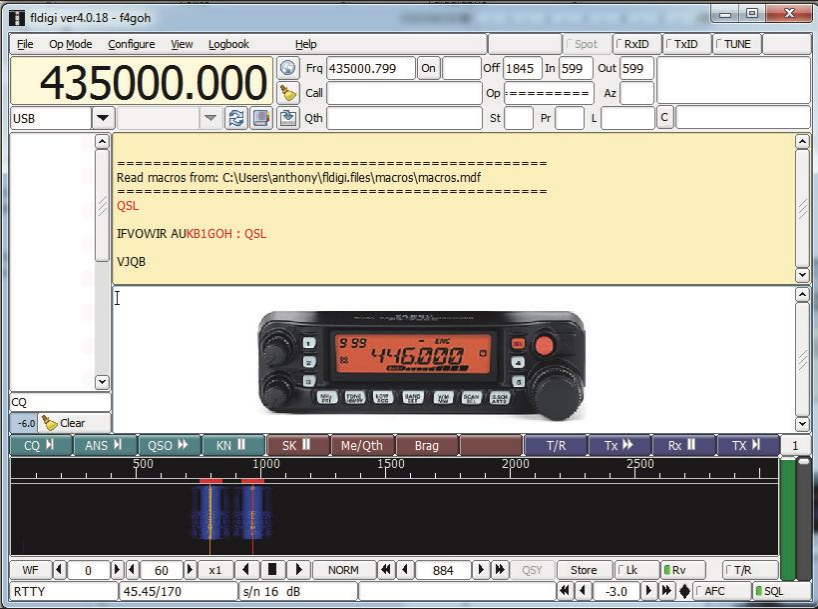

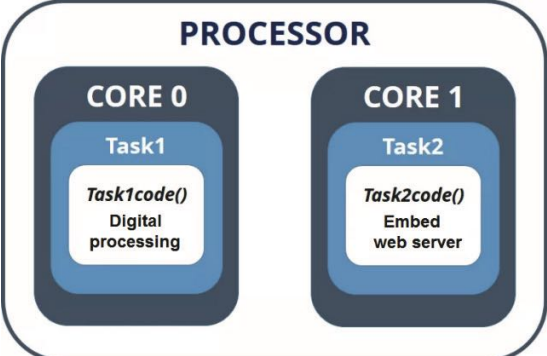
1 Introduction

For several years I have been using an Arduino UNO (Atmega328p) to realize all kinds of applications around the radio. Most of the time, it's very easy to generate an FSK signal using the internal PWM of the processor added with an external low-pass filter. (see Standalone HAM modulation generator: TAPR N°37). On the other hand, it is much more complicated to decode an FSK-type audio signal with the same processor. Indeed, you have to apply signal processing algorithms and you realize that it is difficult to use an FFT algorithm in an Atmega328p while having real-time decoding. Robert Marshall (KI4MCW) explains how to decode an FSK signal with an Arduino Uno [1]. I was able to successfully test the algorithm written by Dennis Seguire in the realization of my APRS repeater with a DRA818 [2,3].

The manufacturer Espressif is known for the ESP8266, a processor that embeds WIFI connectivity. This integrated circuit has managed to federate a very large community of "makers", particularly around IOT connected objects. More powerful, the ESP32 integrates more memory and bluetooth connectivity.



2 Brief presentation of the project

<p>Step 1</p> <p>A PC with Fldigi software is connected to a Yaesu ft7900 transceiver.</p> <p>The user kb1goh sends a message in RTTY for example: qsl.</p>	 <p>The screenshot shows the Fldigi software interface. At the top, the frequency is set to 435000.000. Below the frequency, there is a text area displaying the received RTTY message: "Read macros from: C:\Users\anthony\fldigi.files\macros\macros.mdf", "QSL", "JFVOWIR AUKB1GOH : QSL", and "VJQB". A Yaesu FT-7900 transceiver is shown in the center of the interface. The bottom part of the interface shows a spectrum display and various control buttons.</p>
<p>Step 2</p> <p>The RTTY frame is received by the DRA818 module and decoded by the ESP32.</p> <p>The esp32 is configured as a WiFi access point and web server.</p> <p>The user F4GOH consults the RTTY messages on his phone using a WEB page.</p> <p>There is no specific application to install and no Internet connexion.</p> <p>communication is bi-directional.</p>	 <p>The image shows a photograph of the ESP32 Access point hardware on the left. The board is green and has various components, including a DRA818 module and an antenna. The text "KB1GOH" and "kb1goh:qsl" are visible on the board. On the right, there is a screenshot of a mobile web browser displaying the received RTTY messages: "f4goh:hello" and "kb1goh:qsl". The browser address bar shows "192.168.4.1/msg.htm". Below the messages is a "Send a message:" input field and a keyboard.</p> <p>ESP32 Access point</p>
<p>We notice that the processor has 2 cores. Core 0 handles digital processing and thus RTTY decoding while core 1 handles the main loop and the web server.</p>	 <p>The diagram illustrates the processor architecture. It is titled "PROCESSOR" and shows two cores: "CORE 0" and "CORE 1". Core 0 is associated with "Task1" and "Task1code()", which handles "Digital processing". Core 1 is associated with "Task2" and "Task2code()", which handles the "Embed web server".</p>

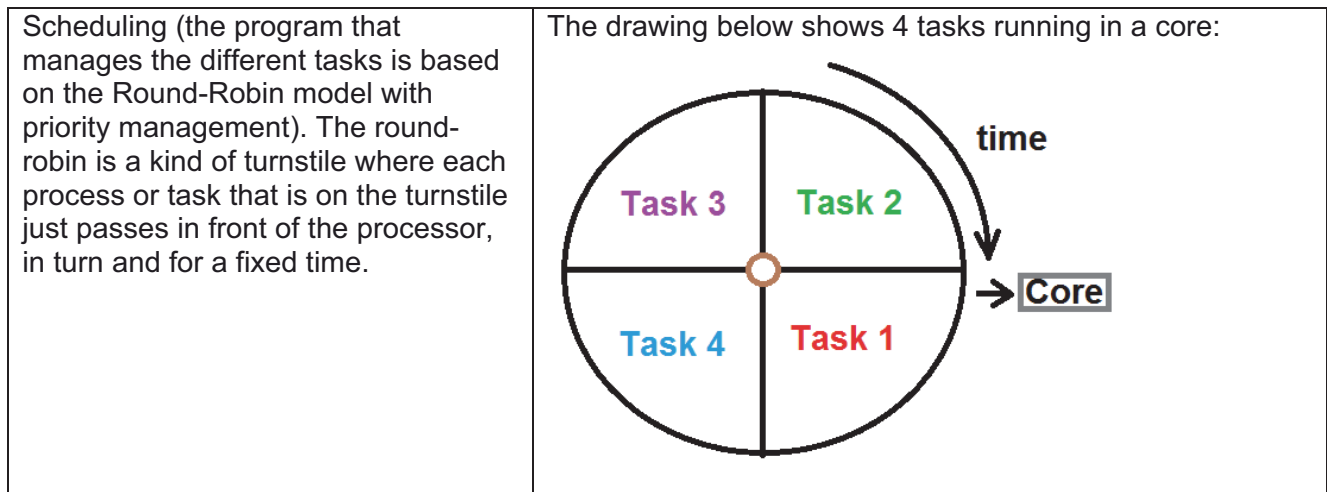
3 FreeRTOS

FreeRTOS is a portable, open source real-time operating system (RTOS) for microcontrollers. Created in 2003 by Richard Barry, it is today one of the most widely used RTOS in the real-time operating system market, including the ESP32

The advantage of using freeRTOS [4] is to be able to manage several tasks in parallel. The number of tasks executed simultaneously and their priority are limited only by ESP32.

To assign specific parts of the code to a specific kernel, you need to create tasks. When you create a task, you can choose in which kernel it will run, as well as its priority. Priority values start at 0, where 0 is the lowest priority. The processor will run the tasks with the highest priority first.

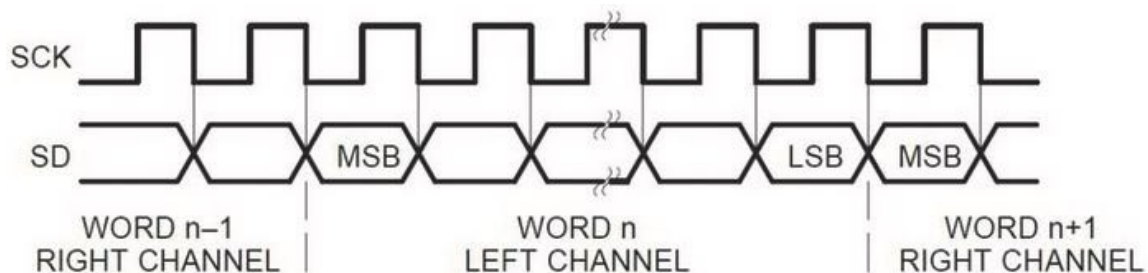
Tasks are pieces of code that execute something. For example, it can be flashing a LED, making analog/digital acquisitions, measuring sensor readings, publishing these readings on a server, and so on.



4 Inter-IC Sound

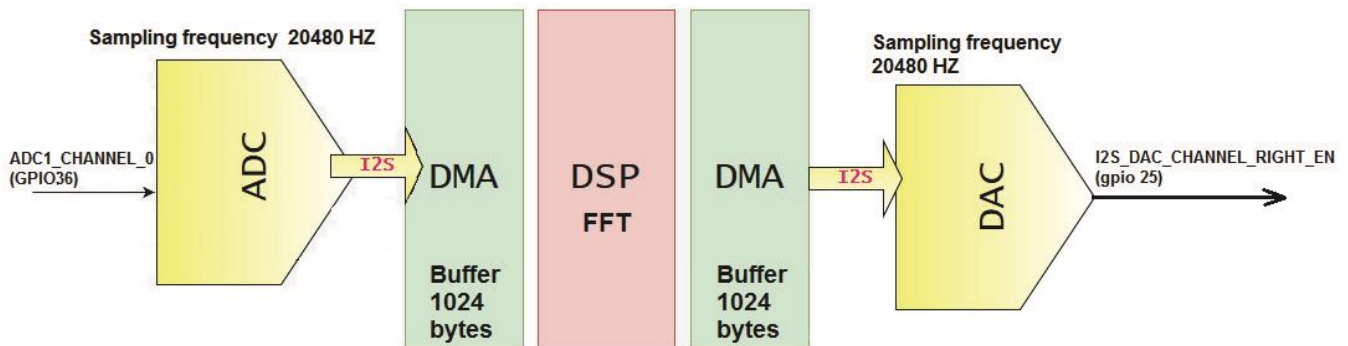
I²S (Inter-IC Sound), pronounced eye-squared-ess, is an electrical serial bus interface standard used for connecting digital audio devices together. It is used to communicate PCM audio data between integrated circuits ADC/DAC in an electronic device. The I²S bus separates clock and serial data signals, resulting in simpler receivers than those required for asynchronous communications systems that need to recover the clock from the data stream.

The chronogram below shows the clock and the data signals. In the case of stereo acquisition, the right and left channels are alternated.

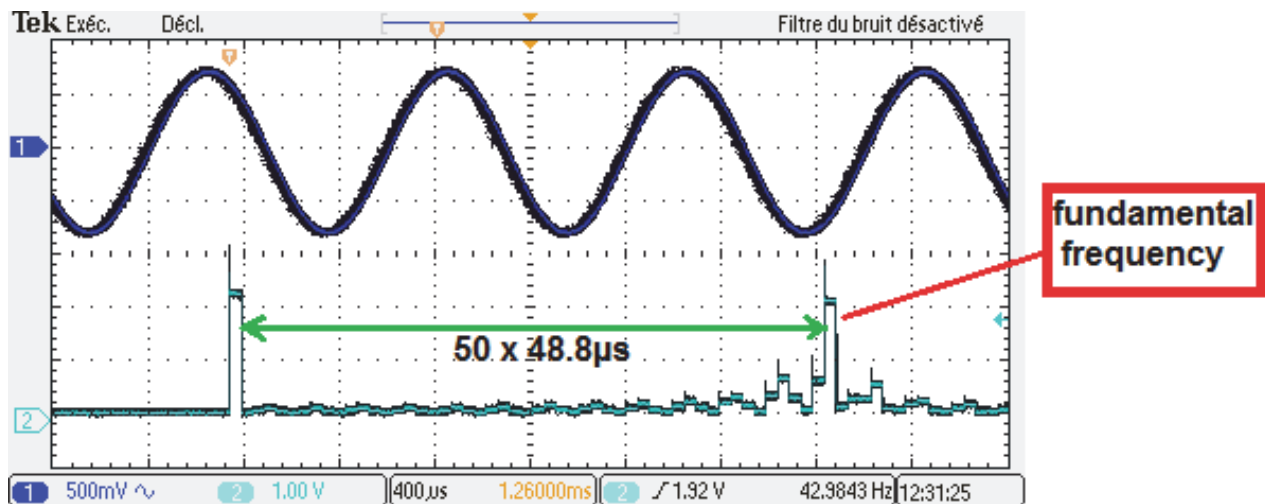


In an ESP32, the I2S bus uses DMA (Direct Memory Access) transfer. DMA is a process in which data flowing to and from a device is transferred directly by a suitable controller to the main memory of the machine, without any intervention by the microprocessor except to initiate and complete the transfer.

This allows the processor to have more time to perform signal processing calculations. This is a huge advantage over the Arduino UNO.



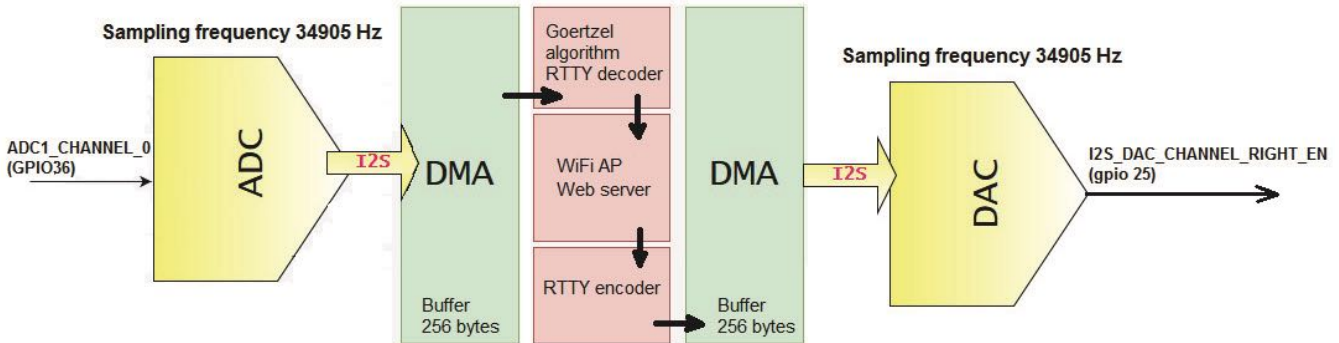
In order to test I2S, I injected with a function generator a 1000Hz sinusoidal signal on the input of the ADC0 (GPIO36: CH1). The processor then calculated the FFT of the signal. The result of the magnitude is then sent to the digital to analog converter DAC(CH2). Each step of 48.8µs (1/20480) corresponds to a 20Hz step (20480/1024) in the spectrum.



The time between the DC component and the fundamental is 50 x 48.8µs, so this corresponds to a frequency of 20 x 50 = 1000Hz.

5 RTTY decoding

An RTTY signal is composed of two frequencies (MARK and SPACE). There is no need to perform a full FFT. For this I used the Goertzel algorithm [5], which allows to detect the presence of a frequency in a sequence of samples. This is an efficient method to evaluate a particular term of the discrete Fourier transform; it requires only one multiplication and two additions per sample. The Goertzel algorithm is obviously used twice, one for the MARK frequency and the other for the SHIFT frequency.



The choice of sampling rate and buffer size is related to the RTTY baud rate (45.45 Bauds). I used the chunk method to detect a logical one or zero level corresponding to the RTTY transmission. If 3 consecutive chunks have the same frequency, I deduce the corresponding logical level of the RTTY bit. I can then calculate the sampling frequency:

Sampling frequency = RTTY speed x Number of chunks x buffer size.

$$34905 \text{ Hz} = 45.45 \times 3 \times 256$$

At each change of state frequency, I count the number of consecutive previous chunks. The byte is composed of the start bit (always 0) followed by the five data bits and two stop bits (always 1).

0,0,1,->3	1,1,0,->3
1,1,1,1,1,1,1,1,1,0,->12 = 11+1 (previous)	0,0,1,->3
0,0,1,->3 = 2+1	1,1,1,1,1,0,->6
1,1,1,1,1,0,->6 = 5+1	Byte->D4 (1101 0100 char decoded 4)
Byte->DE (1101 1110 char decoded k)	0,0,1,->3
0,0,1,->3	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,->21
1,1,0,->3	Byte->FE (char toggle to letters)
0,0,1,->3	0,0,0,0,1,->6
1,1,1,1,1,0,->6	1,1,0,->3
0,0,1,->3	0,0,1,->3
1,1,1,1,1,0,->6	1,1,1,1,1,1,1,1,1,1,0,->12
Byte->DA (1101 1010 char decoded f)	Byte->F4 (1111 0100 char decoded g)
0,1,shift!	0,0,0,0,0,0,0,0,0,0,1,->12
->3	1,1,1,1,1,1,1,1,1,1,0,->12
1,1,1,1,1,0,->6	Byte->F0 (char decoded o)
0,0,1,->3	0,0,0,0,0,0,0,1,->9
1,1,1,1,1,1,1,1,1,1,0,->12	1,1,0,->3
Byte->F6 (1111 0110 toggle to figures)	0,0,1,->3
0,0,0,0,1,->6	1,1,1,1,1,1,1,0,->9
1,1,0,->3	Byte->E8 (1110 1000 char decoded h)
0,0,1,->3	

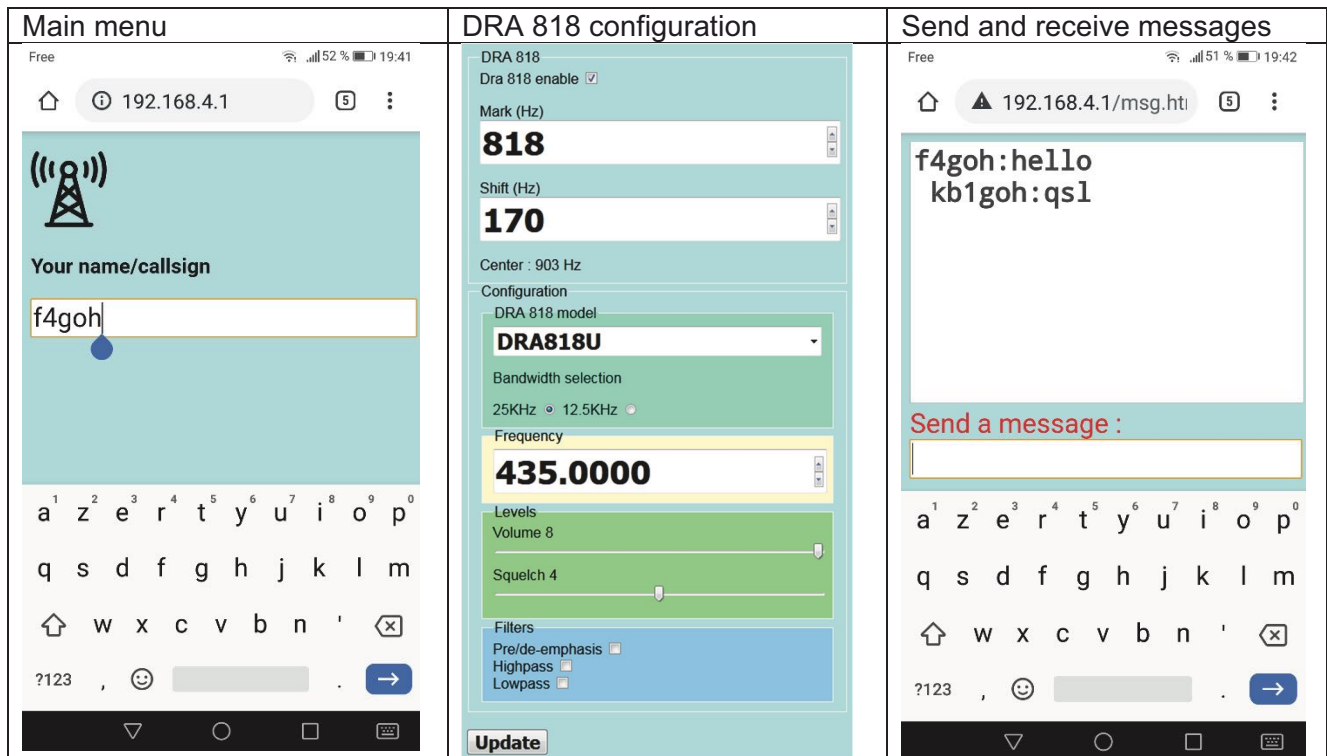
I preferred to use 2 stop bits rather than the traditional 1.5 stop bits. This to facilitate decoding during my first attempts. Nevertheless, I must detect the space character in order to synchronize the decoder and not to display random characters.

When sending an RTTY frame, I always start by sending a space character first. This is obviously useless when using 1.5 stop bits.

6 The web server

The WEB server is composed of four HTML and JavaScript pages. A home page allowing you to choose your callsign and an option to configure the DRA818 and the MARK / SHIFT frequency.

The HTML page that manages the sending and receiving of messages makes AJAX requests to the server to update the reception area. This reception area is refreshed every two seconds. Sent and received messages are also displayed on the mini OLED screen.



7 Conclusion

Through these experimentations around the esp32 and the I2S bus, my main goal is achieved. It is possible to decode an FSK audio signal while managing a WEB page in the same processor. This reduces the hardware, but the time spent to develop the software increases. It would be interesting to switch to APRS decoding. I will end by pointing out a very interesting site, that of HA2NON which gives examples of decoding under PC using the JAVA language [6].

[1] <https://sites.google.com/site/ki4mcw/Home/arduino-tnc>

[2] <https://hamprojects.wordpress.com/2015/07/01/afsk-dra818-aprs-tracker/>

[3] <https://github.com/f4qoh/TNC>

[4] <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html>

[5] https://en.wikipedia.org/wiki/Goertzel_algorithm

[6] <http://dp.nonoo.hu/projects/ham-dsp-tutorial/>

Continued lessons from the RF-Seismograph

Matching HF noise and Propagation to the US Geological Survey's earthquake catalog.

Goups.io user group: <https://groups.io/g/MDSRadio>

MDSR website: <https://www.qsl.net/rf-seismograph/>

Alex Schwarz VE7DXW walexschwarz@telus.net

Earthquake vs. Propagation

A discovery that belongs to all Amateur radio operators.

Everybody gets frustrated when you turn on a shortwave radio and propagation is poor. And right now propagation cannot get any worse, we are at the bottom of the solar cycle and even though there are some promising signs, the new solar max is at least 5 years away. Even propagation is down and the solar flux is around 70, there is still a lot of propagation especially below 20 m, but it is sporadic. What we need is software that records these openings and can display them on the web, freely accessible to anyone; this was the idea of the RF-Seismograph. With the RF-Seismograph we have instant access to propagation records and now we have more than 4 years of data!

When the sun is quiet, information on propagation becomes pseudo science, unless you start to look at a different cause of propagation, such as earthquakes. There are always at least small earthquakes. With the usual 5 or more significant quakes daily, the changes in propagation can be much better modeled and understood. This idea came via a Scientific American article "Earthquakes in the Sky" and it described the piezo electric effect of Bruce's (N7RR) article in his QST article. So the idea was, to look for changes in the recorded data during time of earthquakes. The best record of worldwide earthquakes can be found at the USGS website. At first we looked at events bigger than M6.0, and sure enough from the 160 quakes that were looked at, 70% did have an effect. That was a year ago and we posted our findings and as you can imagine, there are a lot of critics. The difference to previous attempts was, that with HF (short wave radio), we were able to measure the S/N ratio of random stations at a global scale. If there is a change in the ionosphere, we would be able to pick up the modified propagation from almost any quake, anywhere in the world! Thanks to all operators that keep their shortwave radios on and in TX mode, so that our test station was able to pick it up. This is why the discovery of the propagation vs. earthquakes should belong to all active Ham operators! It was a global problem and everyone worked together to solve it.

Setting up and developing the RF-Seismograph has been a challenge

The RF-Seismograph has been developed with interference in mind and the RF-Seismograph separates noise from propagation. This is done by measuring S/N noise level and not just noise. When the RF-Seismograph receives a signal, the S/N ratio is displayed as a vertical line below the noise level. Regular noise is displayed as spikes that go above the noise floor. When there is no signal, the RF-

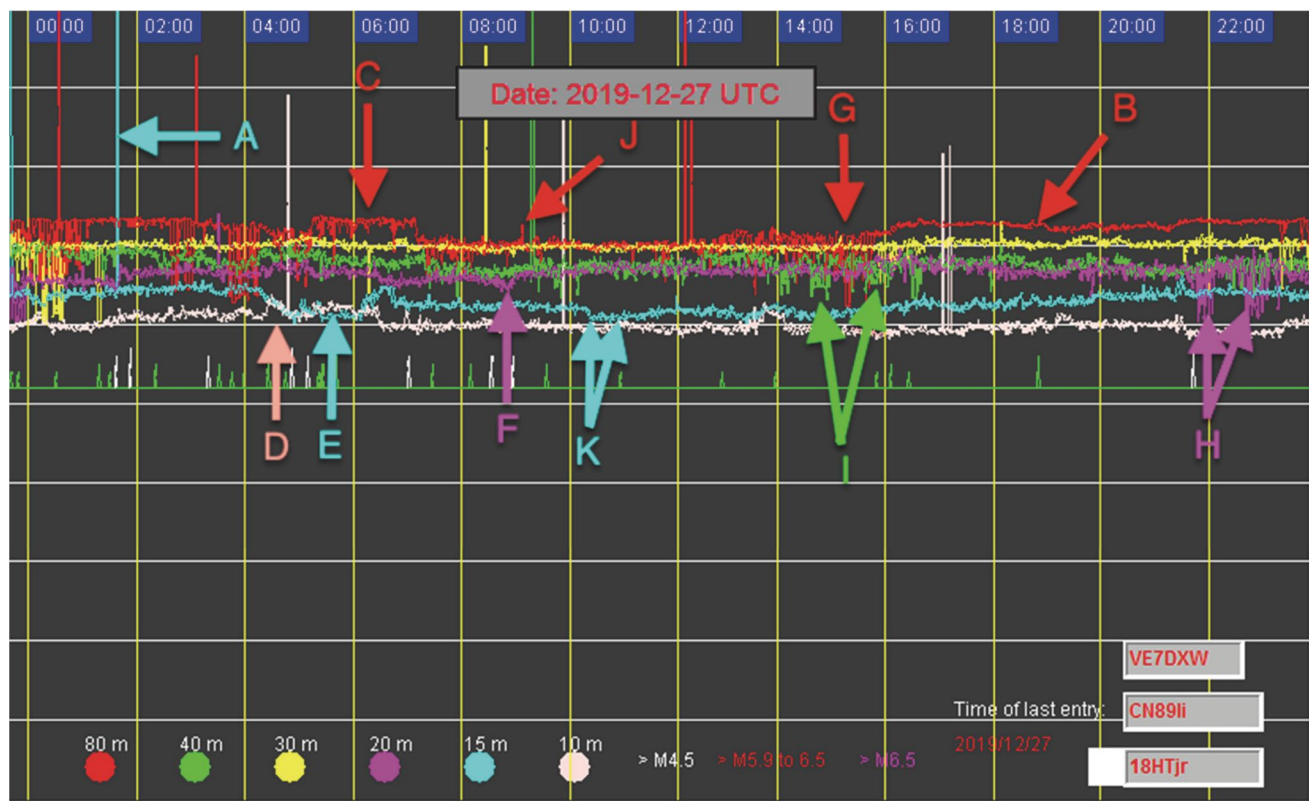
Seismograph displays a thin line that moves up and down as the noise level changes. The RF-Seismograph operates in North Vancouver (CN89) on a 24by7 basis and has been in operation since August 2016.

Discussion of a day of RF-Seismograph measurements

Recognizing Earth Quake Indicators:

This is a “dictionary” of showing received signals on an RF-seismograph and what each bump, attenuation, spike, etc. is called and what is causing them. (**Note:** for viewing in B&W – the normal noise level distribution is highest for 80 m and then goes down as the bands go up – for better viewing in color use the web version). The daily graphs are also available in our group at:

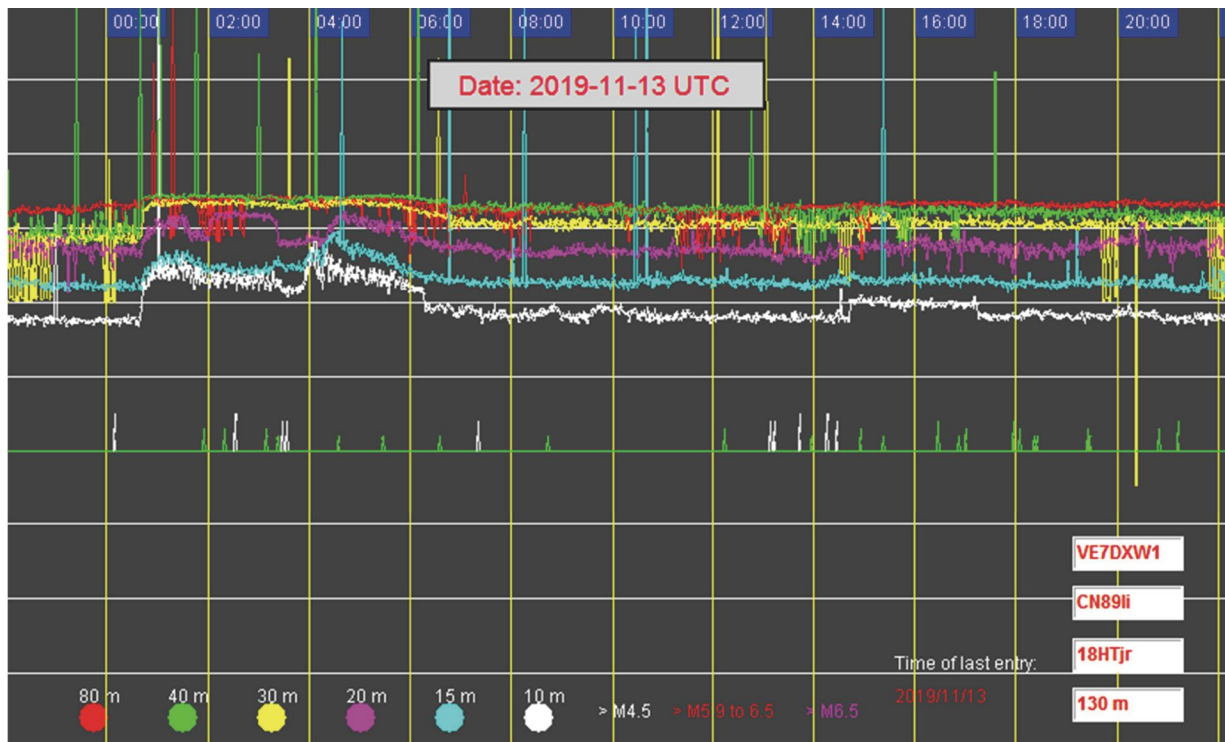
<https://groups.io/g/MDSRadio>



- A spike** – caused by braking magnetic field lines (seen on all bands)
- B increased long term noise (mostly on 80 m)** – ongoing earth quake activity
- C noise dome** – can be caused by a single or multiple quake (shows ongoing quake activity)
- D bump (mostly on 15 m and 10m)** – caused by either fast flowing solar wind or if none present by the interaction zone between quakes (check the NOAA website for fast lowing solar wind)
- E dip (mostly on 15 m and 10m)** – caused by quakes
- F small Dip** – caused by quakes

- G propagation** — shown as vertical lines (when propagating is very strong the line goes below the noise level)
- H sporadic propagation** – caused by scattered ionosphere clouds
- I sporadic propagation** – transition waves with fast noise transitions (mostly seen on 40 m and 30 m)
- J small spike** – human caused interference
- K Noise floor reduction** – show a transition between different earthquake intervals

Graph of RF-Seismograph for Nov. 13. 2019 in UTC time (-7 h for PDT – 17:00 PDT = 00:00 UTC :



When the RF-Seismograph picks up earthquakes, there are two distinct ways it is displayed and it is dependent on the size of the quake. The graph above uses the USGS database to query the size, location and time of the quakes (green and white spikes) and then matches the times of the quakes with the noise record. All the measured data and the daily summaries are archived at our MDSRadio.io group. All group members receive a summary at the end of the day. For a better display of the graph, please join the <https://groups.io/g/MDSRadio> and check the messages and the file section.

Quakes below M4.9 usually do not have a precursor or a large effect on the ionosphere and the S/N of the received signal does not change, but they do create spikes and noise level changes. Good examples are the 10 m spikes that can be seen at 07:15 UTC, 09:00 UTC, 12:10 UTC, 13:10 UTC, 14:45 UTC, 18:00 UTC and 21:00 UTC. Seeing this many time correlations makes it very difficult to assign the measured spikes to random events. We still would like establish more monitoring stations to

collaborate these findings, but the initial measurements look very promising; especially the fact that we see so many of these events and not just on Nov. 17, 2019, but every day!

Quakes above M4.9 do have a distinct effect on propagation as measured by the RF-Seismograph

Note: A synopsis of the given band conditions and earthquake activity as stated below is given with each daily report that is available from the group and mailed to all members after 24:00 UTC every day.

The quake that occurred at 00:10 UTC bends the ionosphere, to allow 30 m propagation at 00:00 UTC for about 15 min. Then the precursor of the quake starts to increase the noise level on all bands starting at 01:00 UTC. This also creates sporadic conditions on 80 m. The spikes go below the noise floor of the signal, so we know that this is not noise, but a real signal. As we can see on the 80 m graph the reception is sporadic starting at 01:00 UTC lasting until 03:00 UTC, with a distinct gap just before the green quake indicator at 02:00 UTC. This day (Nov. 17) was especially interesting, because one can see two quakes interfering with each other! The quake that released at 2:30 UTC started the effect on the ionosphere at 01:00 UTC and it lasts to 06:15 UTC. The two additional quakes that release at 3:00 UTC and 03:05 UTC actually create a noise drop on 20 m, 15 m and 10 m! The quake at 07:15 UTC can be seen because it attenuates 80 m. At 06:00 UTC the 80 m band can be seen trying to open up, but then stops. This is due to the effect of this quake also and 0.5 h after 80 m returns. At 12:15 UTC a minor quake can be seen attenuating 80 m again, which had opened at 11:00 UTC. At 13:00 UTC, two quakes in short succession create another short opening on 80 m. The quake at 13:45 UTC created a short opening on 40 m as well. The double quake that occurred at 14:45 UTC creates 30 m propagation for 5 min only. The 10 m noise increase from 15:00 UTC to 17:15 UTC was created by the clusters that started to release at 14:45 UTC until 17:00 UTC. The 30 m band on this day did come up after 19:45 UTC, but was disrupted by two quakes and only came back just before 23:45 UTC.

Because it is autumn, the daylight and nighttime propagation changes are minimized. The RF-Seismograph is located on the west coast of Canada in Vancouver and the time zone is PST, which is - 8 h from UTC. The date change occurs at 5:00 PM during daylight saving time and 4:00 PM in the winter, so the early hours of UTC reflect the nighttime of the location CN89.

What's next?

The RF-Seismograph team is hard at work to develop a real time Propagation vs. Earthquakes monitor that will run on a PC, a RbPi and a java enabled phone and Mac computer. This will allow everyone interested to monitor RF-conditions on real-time basis and to report their own findings in the IO group. In the mean time we are putting out a daily report that allows us to familiarize us with the way the ionosphere behaves and how quakes change propagation. To share these findings we put out a daily report that is accessible through the group.

Long term goal is still to get other operators to set up their own monitoring station, by using Raspberry Pi4 and the LIF2016 units that are still available through our website. If you are planning to offer your

location and equipment for monitoring, please contact me. We also have a solution for areas that are in lightning danger zones, with our special grounded balun and lightning arrestor antenna system.

Why Use a Vertical Antenna

The antenna system (model: 18HTjr) has been specifically chosen as the best antenna and cost-effective omni directional solution for this application. It is 12 m (36 ft.) tall and has resonators for 40 m, 20 m, 15 m and 10 m. With the height of the antenna the takeoff angle is very shallow, which makes it more resilient to local noise. The 80 m part uses NVIS shooting the beam straight up. Below is the image of the antenna on top of a 30 ft townhouse; ground is provided by a 4” copper water pipe, which is also grounded at the basement.

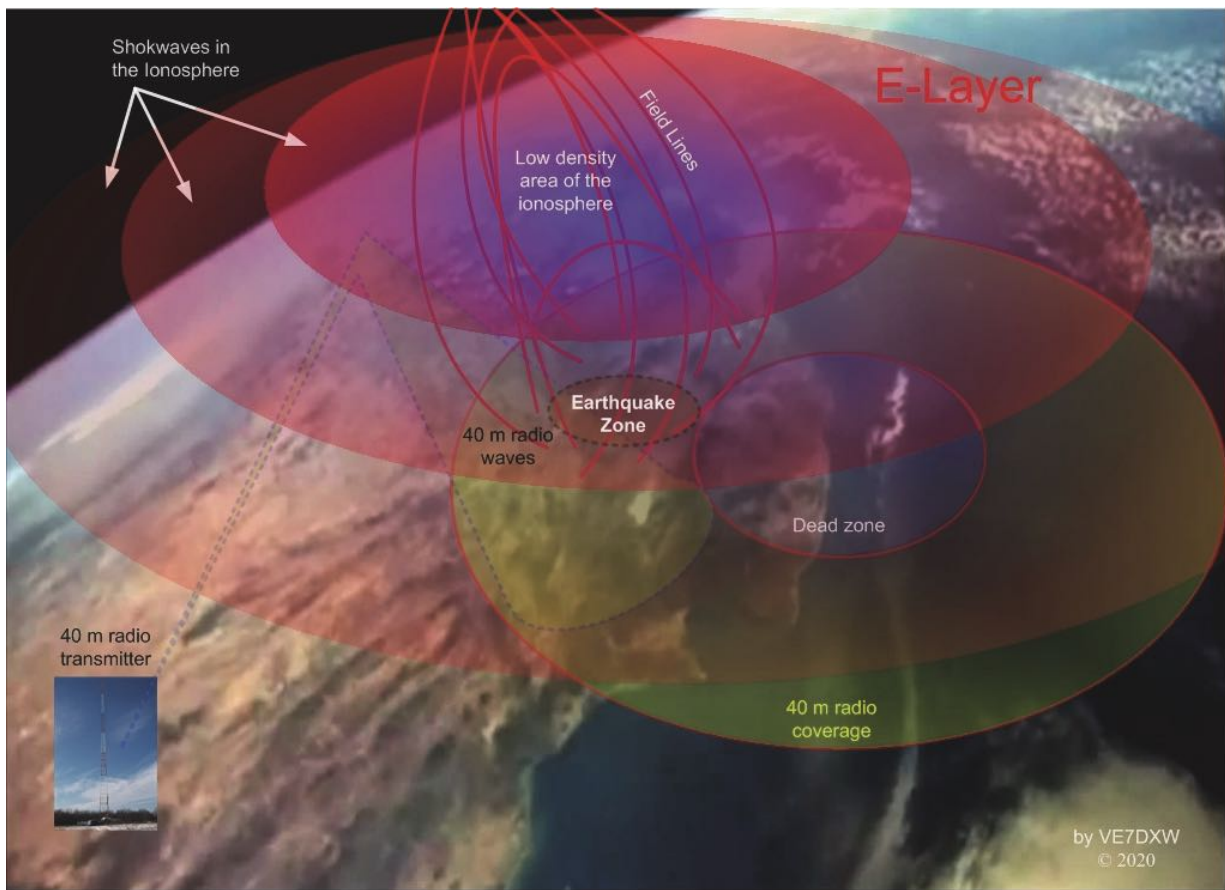


Conclusion

Now we have a good understanding, why propagation is so sporadic during solar minimum; it is caused by the changes of the magnetic fields created by earthquakes that affect the ionosphere. With digital modes and their superior low signal characteristics, amateur operators can keep using their shortwave radios even in the lowest solar flux conditions. By using the RF-Seismograph to catch these signals and detecting how they propagate, is helping, to solve one of the big mysteries in Earth quake science. It seems like it is more beneficial to monitor the USGS earthquake reports for propagation, than to watch solar flux indicators – or just build your own RF-Seismograph.

If we update our repertoire and accept quakes do be more than a movement of soil, the implications will have a very positive and long lasting effect on the world and not just for shortwave radio.

An illustration on how the Ionosphere changes in response to an Earthquakes



References

Scientific American Oct. 2018: “Earthquakes in the Sky”

[http://www.ep.sci.hokudai.ac.jp/~heki/pdf/Scientific American Vance2018.pdf](http://www.ep.sci.hokudai.ac.jp/~heki/pdf/Scientific_American_Vance2018.pdf)

Earthquakes Canada:

<http://www.earthquakescanada.ca>

U.S. Geological Survey

<https://www.usgs.gov/>

Access to Study for 2017, 2018 (2019 is part of 2018)

[http://www3.telus.net/public/bc237/MDSR/Matches-RF-Seismograph and Seismic data for 2017.pdf](http://www3.telus.net/public/bc237/MDSR/Matches-RF-Seismograph_and_Seismic_data_for_2017.pdf)

[http://www3.telus.net/public/bc237/MDSR/Earthquakes visible with RF-Seismograph 2018.pdf](http://www3.telus.net/public/bc237/MDSR/Earthquakes_visible_with_RF-Seismograph_2018.pdf)

Download and Install RF-Seismograph for Linux and Raspberry Pi

<https://groups.io/g/MDSRadio/wiki/home>

Download MDSR software for PC from:

<http://users.skynet.be/myspace/mdsr/>

Many thanks to Joe Joncas (WA7MHB) for the input and the definition of the earthquake signatures.

How to Kill Packet-Radio & APRS? Come to Serbia! (Part 3)

Miroslav "Misko" Skoric, YT7MPB

IEEE Austria Section; NIAR India

email: skoric@ieee.org

packet: YT7MPB@YT7MPB.#NS.SRB.EU

Abstract

In this paper, I continue analyzing continual country's ham radio leaderships' wrongdoings that significantly lead not only to the stagnation in development of VHF/UHF/HF ham radio data infrastructure but also to possible extinction of anything else but telegraphy contests and 'fox-hunting'.

1. Introduction

As you see, this is the third installment of this 'regulatory' series. This year I almost decided not to write, but eventually I changed my mind the 'last minute'. At first, I would like to remind you that my first paper for DCC conferences was published in the proceedings of the "22nd Annual ARRL and TAPR Digital Communications Conference" (Skoric, 2003). Several years after that, I announced: "I ceased any connection with governing people in Serbian ham organizations for good." (Skoric, 2018). But the last installment showed that I was wrong: "Almost twenty years later I came back to Amateur Radio Union of Vojvodina (Savez radio-amatera Vojvodine, abbr. SRV) to serve as the Union's secretary" (Skoric, 2019).

You are right if you ask me now: Why do you waste your time with people who do not deserve your attention & assistance, and why do you bother submitting another paper about? Well, my answer could be this: If I do not do that, and instead fly into apathy, who else would be a witness to all wrongdoing in Serbian ham radio!

As I informed you last time, Polish radio amateur Armand, SP3QFE, asked me to assist

him in locating any local ham in the area of Sremska Mitrovica city (Serbia) who was skilled in ham satellite operations. At that time (October 2018) I was not involved in any 'official' ham activity in the country. So the real question was how SP3QFE reached me at all – having in mind that for ham topics he could have contacted SRV or SRS (*Savez radio-amatera Srbije*, Amateur Radio Union of Serbia). But SP3QFE found me thankfully to my recent (2016, 2017) visits to his country of Poland for participating in some technical (non-ham) conferences. And because I promoted ham 'data' modes in those scientific events, my visits were recognized among amateur radio circles in SP3QFE's homeland.

So that is why I am sure that writing research papers, and presenting such papers in technical events is crucial for our hobby. Without my activities in Poland, it is questionable whether SP3QFE would get in touch with me or not.

In any case, thankfully to that link, a group of interested hams (at that time not-so-proficient in satellite operations) was found in the local ham club in my city of Novi Sad (loc. JN95WF). The preparations for an ARISS contact with scientists at the International Space Station (ISS) started on time, and the program was performed

successfully at the end of February 2020. Although I was glad that everything went well with the ISS sked, and that both pupils and teachers in Sremska Mitrovica were happy, I felt slightly uneasy because I did not participate in the ‘working group’ who performed the session.

In fact, I had offered to the ham club’s president my best services when it comes to writing a research paper on that ARISS contact (means coauthoring an article together with the school’s teachers, which could be then published in some domestic or international publication). Unfortunately, my offer was ignored without any explanation, and I could only say that it was a result of common misunderstanding among Serbian hams on what technical & research papers are all about.

You bet, when our club’s ARISS team returned from Sremska Mitrovica, I listened to their comments on that experience. The main feeling was that the school staff has done the most relevant parts of the session, while the hams “just assisted” in providing radios, antennas, and related logistics. Except local newspapers and electronic broadcasters being invited to the show (as well as local politicians), there was no ‘proceedings’ of the ham event. Now it is several months after that radio contact with ISS, but there is no sign that anyone from the school was interested in keeping closer contacts with our ham club. In my view, a technical paper written together with the teachers would significantly contribute to the visibility of our club in local and wider educational community (not only in Sremska Mitrovica but also elsewhere in Serbia), and it would remain as a document of ham-school collaboration. Not to mention that such a document would also serve as an instructional venue for similar joint actions in years to come.

2. The ham union’s agenda

2.1 Year 2019

I probably haven’t mentioned in the previous installment that, as the Union’s secretary, I actively participated in preparing documents for annual meetings – SRV ‘assemblies’. So, for the 2019 Assembly I asked the Union’s board to include few things to the yearly plan of activities. For example, I urged that the status of former no-Morse license holders (‘E’-class) in Serbia should be harmonized with pan-European CEPT recommendations, by means that ‘E’-class holders (recently renamed to ‘2nd class’) shall be allowed to travel internationally in the status of ‘CEPT NOVICE’ category. I based that idea on the fact that many no-Morse hams wanted to use their FM walkie-talkie portables in their summer holidays & vacations in neighboring countries. Why not to allow those guys & gals to be more happy during their family vacations?

As I said in the previous installment, I took the opportunity to amend the official proposal for the Union’s yearly plan, so I added activities related to improving the rules – more precisely the necessity to accept CEPT recommendations in its entirety. As discussed in (Skoric, 2018), Serbia has accepted only some parts of CEPT recommendations that actually favored telegraphers. At the time of writing this (August 2020), nothing changed in domestic rules.

Ok, my amendments were included to the plan, and I expected the Union’s board to act accordingly in negotiations with our authorities.

I was wrong. Nothing was done.

In the same time, domestic *APRS* network remained underdeveloped: Only the nearest digipeater was in working condition – probably thankfully to the fact that our radio club was responsible for it. Month after month, few other digipeaters, located southern from Novi Sad, disappeared from scene one by one. Nobody from SRS seemed interested in maintaining the national *APRS* infrastructure. It is a shame for Serbia not to have an *APRS* backbone! See Figure 1 for a recent view to Serbia (Србија).

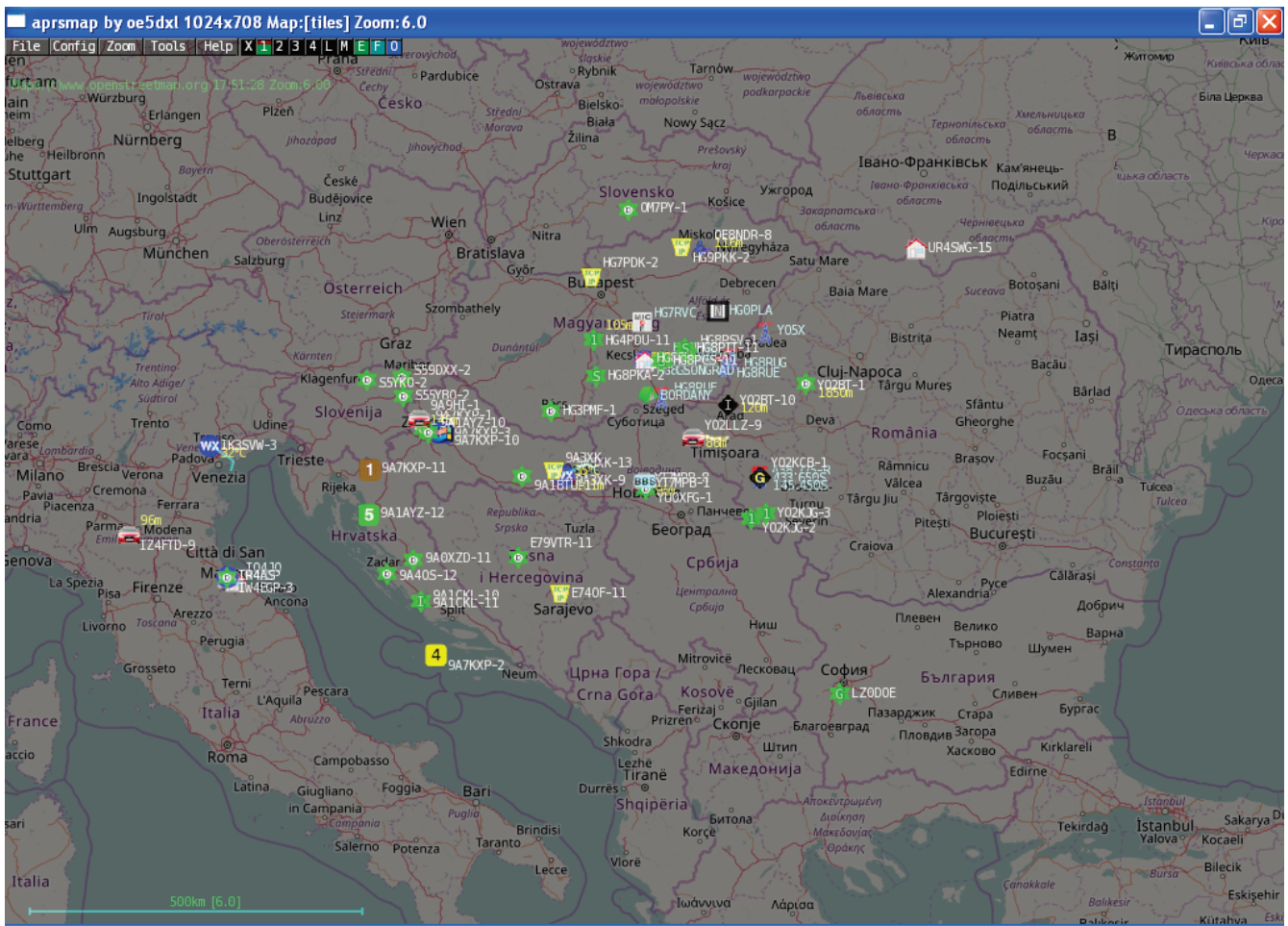


Figure 1. Non-existing APRS network in most part of Serbia/Србија (screen-shot taken on July 29, 2020).

2.2 Year 2020

Year 2019 was busy with other activities. I actively participated in an IEEE-related technical conference in Goa, India, where I promoted ham ‘data’ modes among Indian students & teachers. Thankfully to good relationships in between Government of India and NIAR (National Institute of Amateur Radio) from Hyderabad, India performed in the past a nice pilot-project in implementing HF modes (preferably *factor*) within its national-wide rescue infrastructure.

Although that pilot-project was almost forgotten some years after its ending, I based my talks on its positive outcomes. In that direction, I continued practicing *factor*, AX.25 packet-radio, and other ‘data’ modes in the local ham club.

Incidentally, my amendments on CEPT recommendations became ‘silently’ removed from the new annual plan. When I objected that to the Union’s board, one of the vice-presidents, Stanisa YU7AC, responded me in a personal mail, which I bring here in its entirety (Figure 2).

From Stanisa Zaklan <szaklan@gmail.com> ☆
Subject **E klasa**
To Me <skoric@uns.ac.rs> ☆

Da ne bi smo "davili" ostale u vezi te "E" klase, evo nekog mog mišljenja.

Prvo, mislim da sam ti već ovo napomenuo . Predlažem da, ako možeš, ubuduće neke tvoje predloge i mišljenja izlažeš što sažetije. O problemu te E klase , napisao si čitav "roman".

Ja, a mislim i većina onih koji dobio ovakav teks, uopšte ga ne čitaju. Mislim da je tvoja kompletna rasprava oko E klase, imajući u vidu tvoju elokventnost, sve si to mogao sažeti u dvadesetak rečenica. Mislim da nam je u svim problemima koje imamo, položaj E klase potpuno nevažan.

Pošto sam bio u vreme kada je ona utvrđena, bila je namenjena za one koji hoće da rade samo FM i samo na VHF/UHF opsezima.

Pošto nisam razumeo poentu tvog stava oko E klase, da li ti hoćeš da se njima dodeli prva klasa?

Da li ćeš , uskoro, do SRV u vezi onih papira?
73
yu7ac

Figure 2. YU7AC diminishes my efforts in fight for liberalizing ham radio regulations.

Here is the partial translation in English:

“In order not to ‘kill’ others regarding that ‘E’ class, here is my opinion. I think that I have already talked to you about. My suggestion to you is to write more concisely any of your ideas and opinions. Regarding that ‘E’ class you submitted a whole ‘essay’. I myself, and suppose the majority of other recipients are not going to read it at all. In my opinion, all your discussion related to ‘E’ class, having in mind your eloquence, you might have submitted in twenty sentences only. I think that, considering other problems we have, the status of ‘E’ class is totally irrelevant. Because I was there in times when it [‘E’ class] was incorporated, it was intended for those who wanted to work FM only and restrictively to VHF/UHF bands. Because I haven’t understood your point on ‘E’ class [now categorized as the 2nd class], do you

maybe advocate that they just shall be granted the 1st class?” (Zaklan, 2020)

I did not respond to such offensive words. It was clear to me that both SRV and SRS were main (if not the exclusive) sources of the problems.

2.3 No real political will yet

Interestingly, both the SRV’s president (Keki Laszlo YT7DQ) and two other vice-presidents (Bariček Laslo YU7CM, Popov Aleksandar YT7WE) never voiced any objection to my proposal for improving the rules. It remained unclear why YU7AC became so furious about my ‘open dialogue’ initiative. Maybe because it appeared that he actually lied to me in another occasion when he tried to ‘assure’ me that the strongest opponent to my amendments was his ‘good friend’ Sinisa, YU1RA, the one of those in SRS leadership who had created bad rules back in 2011, (Skoric, 2019).

After having realized that SRV's governing board is a 'bad company' too, I tried to initiate an 'open dialogue' with our membership. I did that by using *YU0V* mailing list (that list is in Serbian language only). I started that in an unrelated discussion on possible introducing new ham bands (new in Serbia). It was obvious that self-proclaimed ham 'elites' in SRV & SRS only wanted new bands to be spared for the 1st class ticket holders. In opposite, they kept silent on any possible move to improve the status of the 2nd class (former no-code 'E' category).

The strongest proponents for new advantages were those who opposed my amendments. I couldn't resist not to react to all selfish and egocentric opinions I received in the mailing list.

So I posted a series of comments to the leaders' essays. As expected, the 'chorus' of them tried (again) to diminish my efforts for liberalizing national regulations. Among the loudest ones was Zoran YU1EW, the acting president of SRS.

I am not going to copy correspondence from the mailing list. What I can conclude for now is that

I was fully satisfied with that opportunity to inform our members about wrongdoers and their misbehaves. And to show no fear, too ...

On the practical side, I did my best to continue exploring 'data' modes in the local club (RC "Novi Sad", YU7BPQ). By using my personal & club's equipment, I perform radio email exchange with a handful of European stations. My test system is based on a laptop running a Linux version of 'BPQ' Node/BBS/RMS software, written by John Wiseman G8BPQ. See Figure 3 for actual setup. At the time of writing this, my BBS exchanges its message content by connecting forwarding partners via HF radio on 20m and 40m bands, and then using pactor 1-4 modulations. Until a month ago, the secondary BBS port was linked to a VHF radio station, by using a sound-card interface for 1200 baud packet-radio (the 'red box' on the left side in Fig. 3). Eventually the club decided to devote the VHF radio to *Echolink* operations, so my VHF port rendered inactive. My intention is to explore all possibilities for HF data exchange, having in mind that club's antennas are in not-so-perfect condition, and need significant repair.

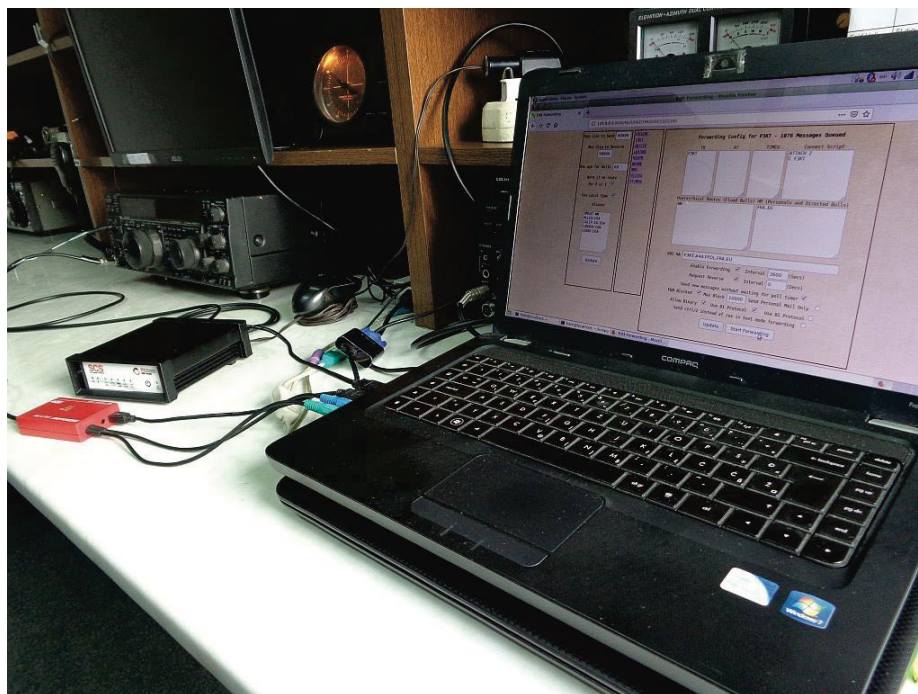


Figure 3. Pactor 1-4 equipment in RC "Novi Sad", Serbia.

2.4 Sporadic statements ...

For the above-mentioned HF ‘data’ activity, I still do not get visible support from most club members. To make things worse, the actual *Coronavirus* pandemic discouraged many (if not almost all) members to visit the radio shack. So most of them are uninformed about this effort. I can only hope that health situation will improve in the near future, and give me a chance to make a good presentation on this infrastructural topic.

As we all know, *Covid-19* caused many public events to be postponed or canceled. One of them was the ham gathering in Friedrichshafen, Germany. After the event’s cancellation was officially announced in April 2020, YU7AC posted in the mailing list: “I am so sad now because I planned to win a lottery there [in

Friedrichshafen]”. So far on ‘important’ things that ham ‘politicians’ in Serbia are focused on. On the other side, they never wanted to create a positive environment where each & every ham radio segment would feel valued and motivated enough to give their best contribution.

Amazingly, the ham ‘politicians’ in Serbia have never written a textbook for ham beginners. For several times I asked them: Why? Do you have not enough knowledge? Or there wouldn’t be enough profitable outcome from such an effort?

Most responses are based on a false premise that “there is already enough literature for beginners” - but none of them can show a new title written by a domestic ham expert. The last book I have personally seen and read, was made by Djordje Stojanovic YU1KH in 1995, see Figure 4.

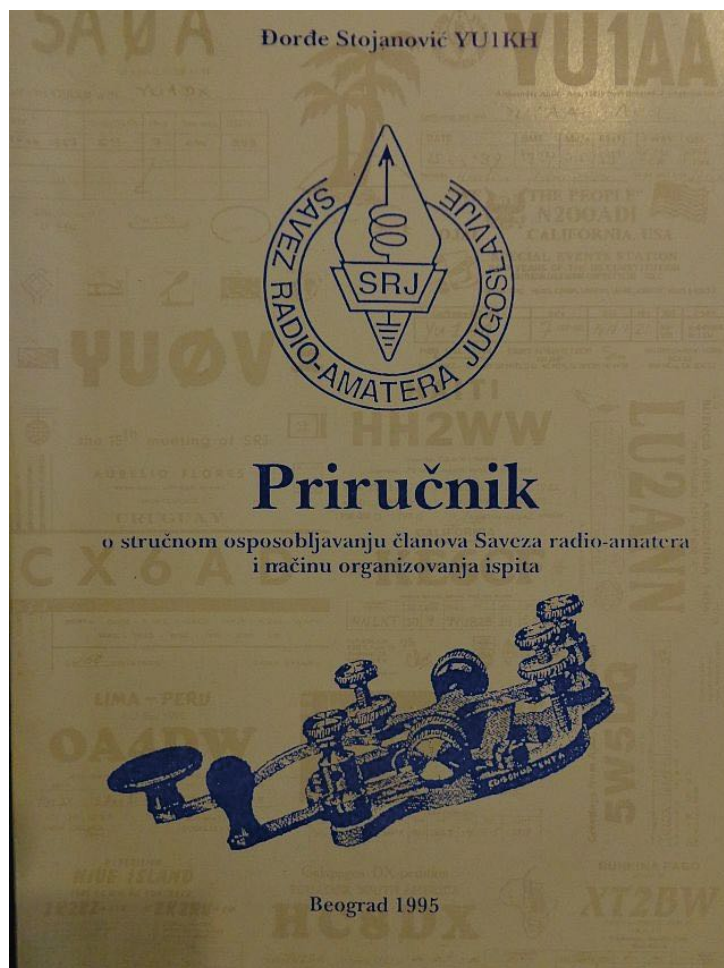


Figure 4. Handbook for educating SRS members and organizing ham examinations.

Interestingly, when I used facts from YU1KH's publication in my open dialogue with the members, the acting president of SRS, YU1EW, diminished the validity of that book by claiming that the publication was "too old". He did not comment the notorious fact that the book was accepted (at the time of publication) as the educational reading i.e. a textbook for SRS members and candidates for ham examination. That was clearly written in the book's preamble.

Newest findings related to wrongdoing in SRS, I have recently sent by email to Prof. Dr. Irini Reljin, deputy minister in Serbian ministry for telecommunications, tourism, and trade (TTT). She is in charge for sector of electronic communications and postal traffic. No responses from her, or someone else in TTT ministry yet. In my email I made it clear that SRS officials had **deceived** the Ministry and the regulatory agency RATEL, by **misleading** governmental bodies into thinking that SRS had acted honestly and on behalf all categories of Serbian hams, and on behalf the nation's technology level as well.

At the time of writing this I still hope that my suggestions sent to Ms. Reljin will be of use.

3. More issues

I mentioned in the previous installment that few individuals in SRS, SRV, and ham clubs have strong intention for keeping *status quo*, in order to protect their selfish personal interests (read: their own or their family businesses). Among those I listed, is for example YU7AC, one of the acting vice-presidents of SRV. His private company, named "ELMED", works in the area of telecommunications. I have recently learned that "ELMED" was a supplier of DMR radio equipment for Government of Vojvodina province (the northern part of Serbia, YU7/YT7 prefix). In parallel of this lucrative business, YU7AC has campaigned for building a local DMR ham repeater in Novi Sad city – the capital of Vojvodina province. His campaign alone would not be observed in a negative context, had

it been performed by someone who was just an enthusiastic radio amateur and DMR promoter. That was not the case because, besides businesses with governmental entities, "ELMED" had a not so frequent permission for importing ham gear. That means, by campaigning for a DMR repeater, YU7AC probably planned to import a quantity of cheap DMR hand-held radios of Chinese production. Eventually he imported a few items. However the number of DMR network participants is still very low. (Please note that building a new DMR repeater was financed by the SRV's membership money, combined with a donation from the local RC "Novi Sad". It is unknown whether YU7AC invested a single dollar from his own pocket.)

Once again, I ask you for advice on how you deal with such cases in the USA or elsewhere.

3.1 Strange budget handling

The legality of the Union's budget is also under a question-mark. For several years now SRV office is located in a municipality-owned property that the Union pays for rent. (Note that during its 70-year tradition, SRV haven't managed to purchase any real estate for its office.) According to the municipality rules & regulations, it is not allowed for organizations that pay for rent to make a profitable business by further renting the space to third parties. But SRV does just like that, so a half of the previously borrowed space is now re-rented to a restaurant around the corner!

So instead of using the (paid) space for installing something of a broader amateur radio interest, such as a technical display room, or a kind of a 'ham museum' or like, the Union uses just one room (of circa ten square meters) for the administrative tasks.

Furthermore, there is no visible interest in using paid space for establishing a packet-radio network node, or a multi-facet BBS with VHF/UHF/HF forwarding features, and/or some satellite opportunities. None of that!

On the other side, a big (if not the biggest) chunk of the Union's income is spent on a variety of contest diplomas and plaques – even though a minority of SRV members participate in contests organized by the Union. Not to mention spending a lot of (membership) money for printing thousands of new QSL cards under special call signs, even though the Union itself does not have (and use) **any** workable radio station under its roof. And even that would not be a big deal, if all those awards had not gone so often to foreign ham winners who personally do not contribute to our organization's budget.

I objected that situation for several times, but it seemed that no one cared. Furthermore, I advocated opening a bank account for foreign membership subscriptions, intended for hams residing in other countries, and in such case awards would be sent them at no cost. In opposite, international postal parcels to non-members shall be charged accordingly. My proposal was rejected with an explanation that "ARRL sends diplomas internationally at no cost". (Interestingly, I did not hear a single word on high standards of living in the USA, foreign membership as a standard category in ARRL, a number of good quality publications made under the League's umbrella, and so on.)

3.2 Dangerous historical revisionism

Even though I am not a contest-inclined ham, I recently witnessed something I found shameful. Let me give you some introduction: Every October there is a traditional ham contest titled "Vojvodjanski oktobar" ("October in Vojvodina"). It has been organized by SRV. In my opinion, the title was properly chosen (long time ago) because of the contest's timing and its organizer.

And it was no problem with that name until recently: Last August-September (2019), rumors started telling that the name was about to change because of a "reminiscence to communism". (For those who are not familiar with the last days of the WW2 in this part of Europe, October 1944

has been remembered for the liberation of Vojvodina province and the other eastern parts of former Yugoslavia. So many Yugoslavian (Serbian, Croatian, Montenegrin, Bosnian, etc.) 'partisan' soldiers and volunteers, alongside with Soviet military, had unnumbered casualties during the liberation operations. I have lost two family members: A younger brother of my late father, and also a brother of my late mother. We have never lost them from our memories though.

But recently, almost 75 years after those heroic times, "Vojvodjanski oktobar" ("October in Vojvodina") somehow got a 'bad taste' in the mouth of the SRV's leadership. When I tried to discover what the problem was with the old name, YU7AC, the vice-president 'explained' me that the word October gave them a "reminiscence of the communist times". I was shocked! What a shameful attitude in YU7AC who himself worked as a fully-paid employee (in the role of the administrative secretary of SRV) in those "communist times"! And not only that: Without being a loyal Communist Party member, he would have never been an employee in such a high position – in the times when all hams were carefully registered, well controlled, and treated as a paramilitary communication squadron of the national security interest.

I have personally seen in SRV office an old photography where YU7AC posed besides a political parole "I posle Tita, Tito" ("After Tito dies, long live Tito"). But, amazingly, just a few decades later, YU7AC eagerly wants to "wash his personal biography". He wants to forget everything he was doing during the "communist times", and you bet – to become a 'fresh nationalist', fully loyal to Mr. Vucich, the acting autocratic president of Serbia, and his rigid political regime. Why? Because of business.

Nevertheless, when I asked for other contest name-change proponents, some members suggested that the SRV's president (Keki Laszlo YT7DQ) and one of the vice-presidents (Bariček Laslo YU7CM) also supported the name change. I really cannot say whether that is truth or not, but the fact is that both persons are of Hungarian

national minority (having double citizenship & passports of both Hungary and Serbia). And having in mind that Hungary was a part of Nazi axis during the WW2 and, as such, lost the war, it would not be far from truth that many Hungarians would like to ‘re-paint’ that part of the region’s history.

In any case, “Vojvodjanski oktobar” (“October in Vojvodina”) was quickly renamed to “Memorijal Mihajla Pupina” (“Memorial of Mikhail Pupin”), a researcher-inventor of Serbian origin who did good deeds in technology areas – but only after his moving to the United States of America.

Not a problem for me: But then I proposed to establish at least a new ham contest or diploma devoted to the USA, because neither Mr. Pupin nor Mr. Nikola Tesla would have ever been in a position to make great things for humanity – without being welcomed by the USA.

My proposal was ignored.

4. Conclusion

Having said this story, I remain unsure what could and should be done in Serbian amateur radio, in order to change (improve) the things. Maybe strong ignorance, or a visible boycott to the contests organized by this country would provoke some reconsideration among ham radio practitioners in Serbia. Like it or not, what is for sure and has been proven as a workable way against wrongdoers in Western Balkans, is *pressure*. Pressure of all kinds. As soon as local politicians (incl. ham radio ones) realize that their stupidity is not going to be tolerated (anymore) by (western) democracies, there will be more chances for a better society and for modern computer-based communication systems to be (re-)established and then well-maintained.

5. References

Skoric, M. (2019). How to Kill Packet-Radio & APRS? Come to Serbia! (Part 2) In *Proceedings of 38th ARRL and TAPR Digital Communications Conference* (ISBN 978-1-62595-112-0, pp. 75-86). Newington, CT: American Radio Relay League.

Skoric, M. (2018). How to Kill Packet-Radio & APRS? Come to Serbia! In *Proceedings of 37th ARRL and TAPR Digital Communications Conference* (ISBN 978-1-62595-101-4, pp. 79-89). Newington, CT: American Radio Relay League.

Skoric, M. (2016). Adaptation of Winlink 2000 Emergency Amateur Radio Email Network to a VHF Packet Radio Infrastructure. In El Oualkadi, A., & Zbitou, J. (Eds.), *Handbook of Research on Advanced Trends in Microwave and Communication Engineering* (pp. 498–528). Hershey, PA USA: IGI Global.

Skoric, M. (2014). Software in amateur packet radio communications and networking. In Matin, M. (Ed.), *Handbook of Research on Progressive Trends in Wireless Communications and Networking* (pp. 122–188). Hershey, PA USA: IGI Global.

Skoric, M. (2013). Security in amateur packet radio networks. In Khan, S., & Pathan S. (Eds.), *Wireless Networks and Security: Issues, Challenges and Research Trends* (pp. 1–47). Berlin - Heidelberg, Germany: Springer.

Skoric, M. (2012). Simulation in amateur packet radio networks. In Al-Bahadili, H. (Ed.), *Simulation in Computer Network Design and Modeling: Use and Analysis* (pp. 216–256). Hershey, PA: IGI Global.

Skoric, M. (2009). Amateur radio in education. In Song, H., & Kidd, T. (Eds.), *Handbook of Research on Human Performance and Instructional Technology* (pp. 223–245). Hershey, PA: IGI Global.

Skoric, M. (2003). Legal rules and regulations in the amateur radio computer networks. In *Proceedings of 22nd ARRL and TAPR Digital Communications Conference* (ISBN 0-87259-908-6, pp. 215-221). Newington, CT: American Radio Relay League.

Zaklan, S. (2020). *E klasa* Personal communication.

Build a Mobile Mesh Tower Fleet

Erik Westgard, NY9D ny9d@arri.net

8/10/2020 1.4 (revised)

Hams in Minnesota have just purchased eight 30' class mobile tower/generator trailers. These are intended for our various mesh networks such as the TWINSLAN Medical Command Network built for the Medtronic Twin Cities Marathon, AREDN and other applications and services as required.

The public service agencies we support have been asking for more real time data and information from us. This includes charts, graphs, dashboards and live video. “Better data, better decisions” is a regular message from an area Red Cross leader we know well. This is driving the use of more mesh technology.

More towers were purchased than expected, so we are developing a proposed set of standards and a new operational model- based on the bass boat armada of the Cajun Navy – we are not pretending to be from Homeland Security.

Hardware- Surplus Construction Light Towers

In January of 2020 we got a call to provide a medical command center to the Loppet Winter Festival- a popular urban ski event. This had several aid stations and a medical center in the back of a ski chalet. The building was in a valley- and I was worried about repeater coverage. No tower trailers were available- ours was frozen in. A request for a government trailer was rejected.

An online search for “tower trailer” became “light tower” – a forlorn 30 foot unit with a broken generator and missing lights was purchased from a corn field for \$700. Sources for these include Craigslist, eBay, Facebook Marketplace and www.ironplanet.com.

Light Towers are common in construction sites and rental fleets. Along with the rotatable 24-30 foot tower you get outriggers, four lights and a 6KW (~12hp) two to three cylinder diesel 115/240V generator. In running condition, they rent for around \$50 per day so command a sale price of \$500-\$4000 used. Dead engines or burned out generators are expensive to repair and sale prices for scrap /retired units can be very low- \$100-\$700. You should view the price as for the tower, and the broken engines are normally in rough shape- fuel injection and rebuilds are costly.

If you plan to use the diesel, running ones can be a better investment, given falling auction prices and the potential repair parts cost. An injection pump is around \$200-\$300, and injectors are \$100 or so. Running diesels at less than full power causes “wet stacking” which can be bad for them. There are lots of videos on diesel diagnosis and repair for those so inclined. If the diesels are hopeless small dual fuel portable generators can be bolted in and are common and cheap.

Ours have usually needed new tires, trailer jacks replaced, the light system transformers removed and new deep cycle 12V batteries. I have installed low cost solar charge controllers and panels. One thing we learned- some towers have electric/hydraulic hoists, which may require the engine to be running.

We added a “shore power” 115V inlet for one. You can often fit one or even two towers in a residential garage. Be aware there are newer ones with non-tilting vertical towers- these can be 8+ feet tall retracted

Hazards include lead paint and used diesel oil which are toxic and fraying lift cables. Two of ours have bent axles. Insurance was cheap- \$35/year- and unlike a motor vehicle could potentially be assigned to a club. Assume the trailer lights are broken- bring a magnetic set. Get a bill of sale signed and dated for licensing- title documents are rare. Once in while these have a “pintle” hitch- common on dump trucks.

Trailers come in several sizes- the bigger, usually older ones can perhaps handle a triband antenna- probably with guy ropes. Weight is around 2000-3000 pounds so a large tow vehicle is not required.

Normally one solid antenna mount can be developed at the top (most come with a cross bar for the lights) and with a piece of aluminum square tube you can add a cross bar for additional smaller antennas.

Software /Networking

We tested the early ham radio mesh around 2010 – some consumer grade access points did poorly outdoors and 2.4 GHz had a high noise floor in a crowded finish line space. So in 2011 we adopted Ubiquiti, OpenWRT, OLSR and Part 15 802.11A frequencies. Our addressing system was based on our D-Star DD network which was a /24 per node. And we got an agency complaint about in fighting in one of the ham radio mesh groups. So we made our own standards based setup.

The need for FCC licensed control operators on a mesh network puzzles my leadership who are all medical providers when we talk about area wide incidents. On the other hand we get endless grief from hams for having a “closed/private” mesh on Part 15, but the idea of a 10ish megabit, backup emergency use network also being an ISP and test bed for hundreds of people frightens us, or one that cannot easily support crypto. We have a 9.1 mile link between Minneapolis and St Paul, and a few omni nodes at each end. So there are two mesh networks. Ours, which supports volunteer medical use cases, and AREDN mesh.

We have each mesh node of ours with static addresses- so video cameras, IP PBXs and database servers can be set up in advance. DHCP is also nice when deploying workstations or IP phones.

We engage with our served agencies using this model:

1. Understand served agency needs
2. Embed our volunteers and technology
3. Execute flawlessly
4. Continuous improvement.

We are trying to support more mature and sophisticated volunteer applications and services. Our events have never used go-kits (bringing backpacks to major sporting events is now discouraged nationally) or formal message traffic.

I have been working with K8JK on a maturity model for the delivery of volunteer emergency services via Amateur Radio. These models are common in Information Technology, and can be applied to us. The idea is you move up on the list over time following best practice.

1. Integrated volunteer emergency management
2. Family reunification /volunteer medical coordination
3. Interoperable communications /backup EOC
4. Amateur Radio traffic handling /shadows
5. Backup communications

Our key deliverable is usually based on updates we made to a software package called trivnetdb. This is a Linux / PostgreSQL port of the old ARES-Data.

At our race events we are handed a flat file (.csv) with the first name, last name, gender and race bib number of participants. We can then query these and record and report status changes, primarily location. This application supports a web interface so works fine over 90kbps D-Star DD links, mesh of all flavors and the Internet. We did experiment with limited AX.25 support.

We have a chat function and any database status updates scroll across that screen which can be used in a management or net control center. Data entry is at Net Control from course deployed hams. Reports arrive by voice radio and updates are then entered there. At the Hospital Tent front desk or in our “Find Your Runner” family information tent, numerous laptops or tablets are used over secure WiFi.

In a disaster, the database allows people to be added- we would use a telephone number or even an email address- those are not particularly protected /private and everyone has one or the other. The database scales- and we have a fleet of 32 old laptops in sets of eight. We toyed with a mobile app- but a web page is free and easy and works on anything. Our Medical team adopted a mobile application with cloud dashboard called RaceSafe. (<https://www.iracesafe.com/>). In any missing persons situation, the common requirement to pre-register seems a barrier.

The idea is not to be prescriptive here- whatever you find useful can be supported. Standards are needed for mesh backbones and IP addressing and security otherwise you have a mess. There is a lot of work underway on compute packages, IP PBXs, video servers, etc. DNS can also be used. We reject the often cited example case “use the Internet to back up the Internet.” Just waiting around for commercial cellular networks to fail seems uninteresting.

Standard load-out

The load out on the first trailer includes several services. We would hope any trailer would at a minimum support an unmodified radio to support the local mesh network(s) and be a relay node- even if no apps or cameras are provided. AREDN is air gapped (or carefully firewalled) from our production mesh:

Main antenna mount- rotator and beam or tri band omni (2m/440/1.2) as required + a pulley for dipoles
5 GHz 802.11 mesh Ubiquiti radio for our Part 15 network (happy on 12V with a \$4 12V POE adapter)
5 GHz 802.11 radio for AREDN mesh (locally channel 180 /10 meg, etc.)

Ubiquiti NanoSwitch™ on the tower (12v POE- waterproof)

2Ghz 802.11 access point (also on 12v POE)

PTZ IP camera (12v supply- 48V POE) Cameras should have a simple local web server /IP not require an encrypted cloud app to be best on mesh IMHO. Hennepin County Sheriff volunteers came up with/tested this idea. Video comes from mobile trailers and is fed via mesh to the command truck which curates the video and feeds it to the Internet for mobile devices and EOCs to use.

2m/440 MHz small dual band antenna

Dual band radio

1200-3500 W dual fuel propane generator (these are \$269 and up)

Two Group 27 deep cycle batteries

30A solar charge controller + battery protector (\$10)

100W outboard solar /30 W onboard solar

115V Shore power inlet

LTE router if requested

Portable mesh video cameras

Medical command /missing persons database server <http://www.kb8zqz.org/trivnetdb/>

IP PBX (Pi based)

Proposed “Rules”

With a fleet of trailers we can cover very large events, or more than one event at a time. The trailers performed well at Field Day with beams and rotators. We tested mesh video with portable cameras at the Klondike Dog Derby- on a 15,000 acre frozen lake.

1. Trailers get cute tactical names/colors (ask your wife). Or it was suggested to have your kids design a paint scheme for them.
2. Trailers are interoperable /tested with our mesh network(s). This includes our most important area government command trucks. These trucks can act as video and data aggregation points for large events.
3. The ownership should be widely distributed to prevent any one person from taking them over.
4. Ideally we have a three hour response time state wide. My two are plugged in on warm standby. Four hours to find which box in the basement has the antennas and which bin has the feedlines is not good.
5. Trailers should travel in convoy for the best visual effect. I can see these in parades, county fairs, etc.
6. Citizen Science is the idea here- creative technology is encouraged- this is a technology/service/hobby project.
7. If more than one trailer is needed at once - ask- "Barn Raising" is the model

8. These are intended to be cheap and abundant - more like cattle vs a prized race horse. If you have \$10,000 in it or it is powering your house it can't be out (or loaned out) helping others.
9. We found epic reflective callsign decals <https://www.ebay.com/itm/123054221928> - the idea is we are proud to be volunteer Ham Radio operators and are all about rescuing/helping people.

Thanks to N0NAS and KD8GBL for support and advice.

Erik Westgard, NY9D is the Volunteer Medical Communications Coordinator for the Medtronic Twin Cities Marathon, Red White and Boom Half Marathon and Loppet Winter festival.



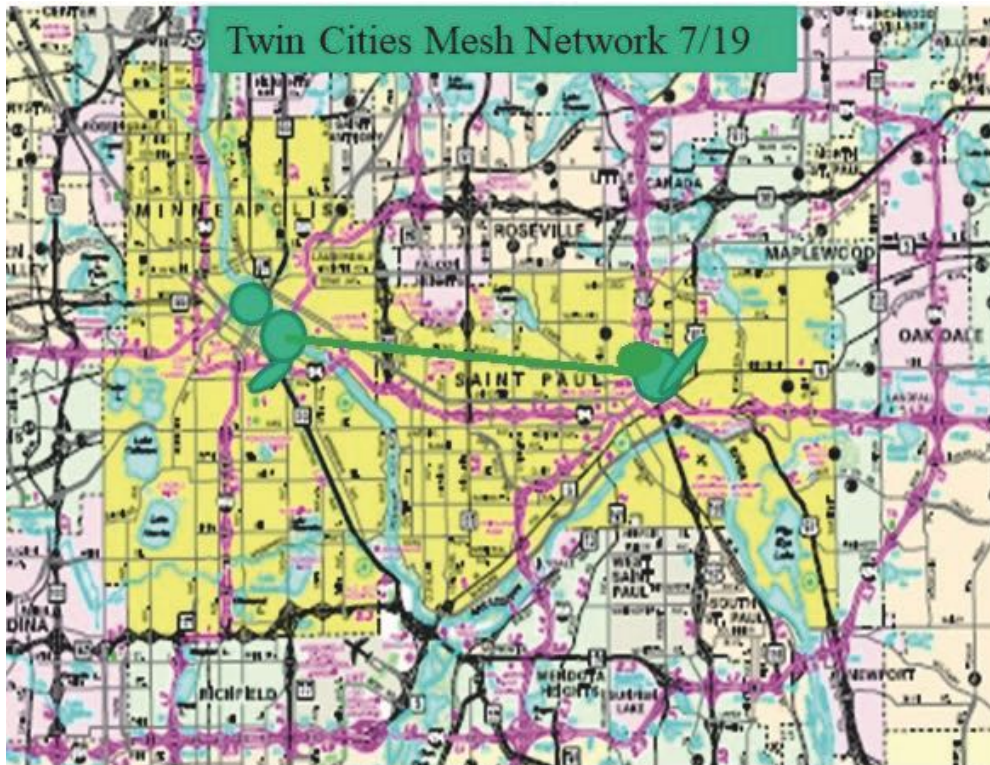
Our first trailer, an Over-Lowe (code name - it used to be orange and is now black) with a six meter beam, several dipoles and mesh at Field Day 2020



An Allmand Brothers ML4- larger and wider with a tri-band beam, dipoles and electric hoist.



Trailer #1 (L) and #8 (R) “Jolly Green Generator” being setup at Hams in the Park here for mesh interop testing.



Our (Part 15) TWINSLAN Medical Command Network

Our Open Source Tracking Software: Trivnetdb

The screenshot shows a web browser window with the URL `/trivnet/query.php`. The page title is "TRIVNETDB - Amateur Radio Information Network". The navigation menu includes "Home", "Search", "Multi-Edit", "Batch Edit", "Messaging", and "Admin".

The main content area contains two identical search forms. The top form is highlighted with a green arrow and text: "Missing /dropped out/ill runner web runner lookup and update".

Each search form includes the following fields and options:

- Last Name:
- First Name:
- Bib Number:
- Gender:
- Race Entered:
- Age:
- Middle Initial:
- Search options: Match all (AND search) Match any (OR search)
- or by status message:
- Buttons:

Below the search forms, it displays "0 rows".

Text annotations on the right side of the forms:

- Runner location only- non HIPAA
- Can be used for family reunification, vaccine stations

The browser window shows a tab with the number "10" and a "FoxyProxy Disabled" notification.

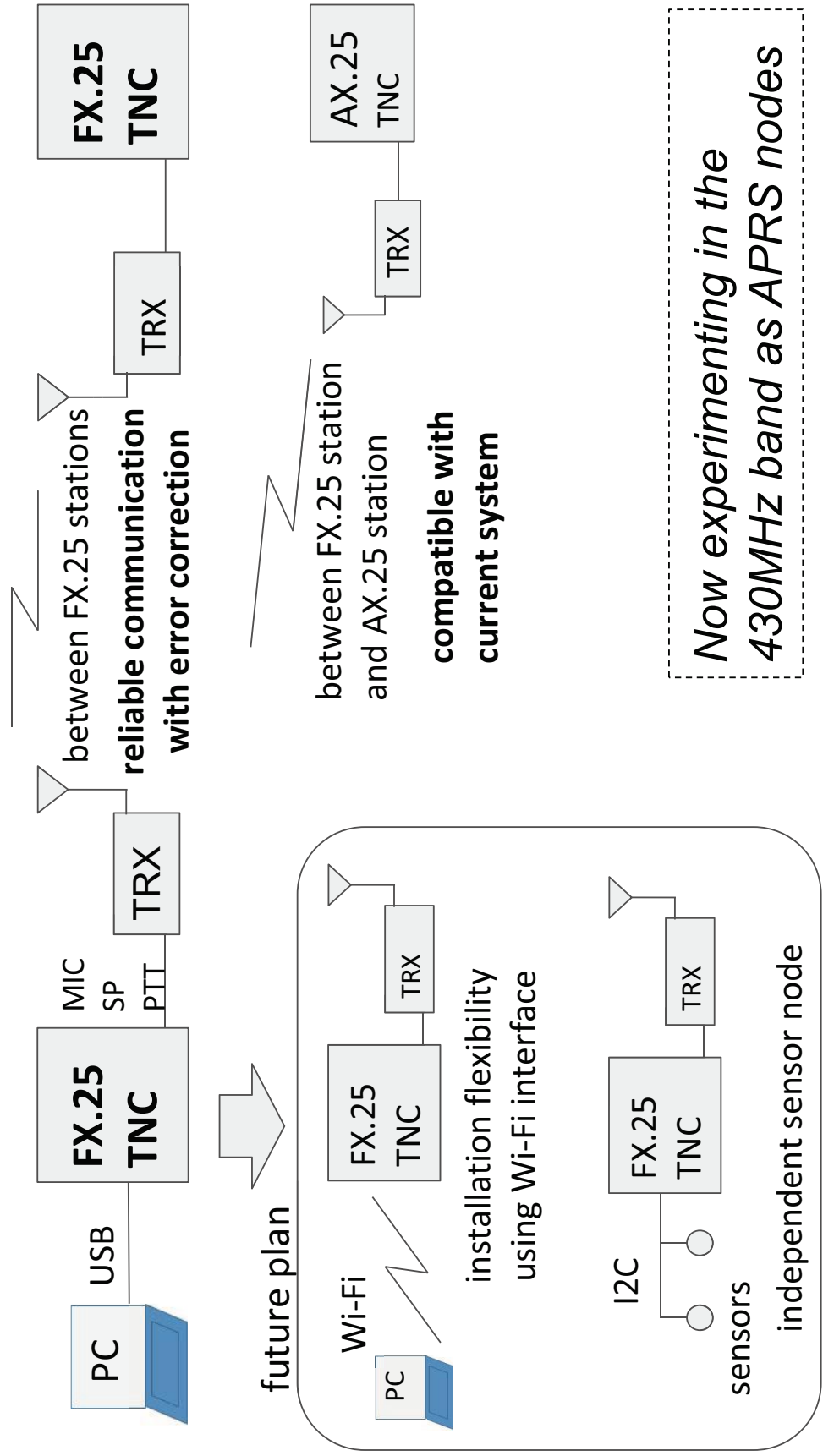
Above - Trivnetdb- our data entry/query screen

Current status report of FX.25 KISS TNC development

Kazuhisa Yokota, JN1DFF
Masaaki Yonezawa, JE1WAZ

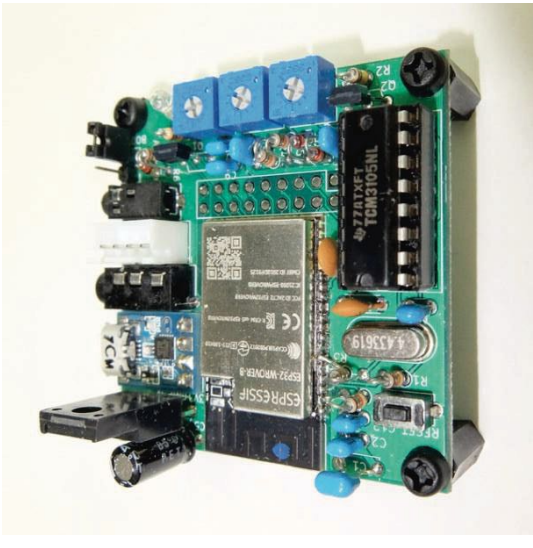
Packet Radio Users' Group
Aug.12, 2020

Use of FX.25 KISS TNC and future plan

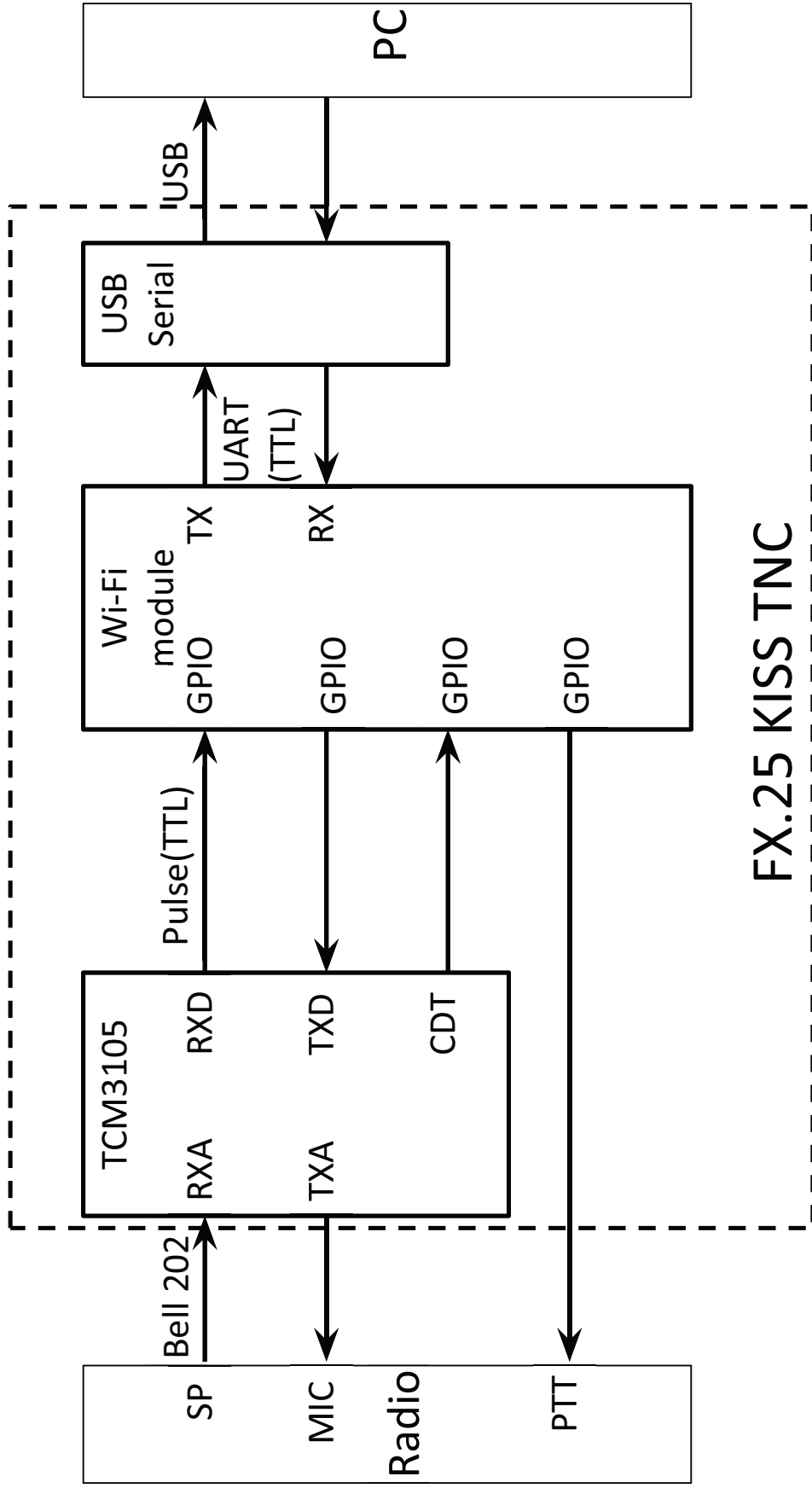


Features of FX.25 KISS TNC

- Using TI TCM3105 Bell 202 modem chip
 - Implements software modem on next version
- Using ESP-WROOM-32 Wi-Fi module
 - Dual core 32bit RISC CPU, clock 80-240MHz
 - RAM 520kB, flash ROM 4MB
- Host interface is USB serial and TCP/IP on Wi-Fi
- KISS mode only
- supports full FX.25 draft spec.
 - http://www.stensat.org/docs/FX-25_01_06.pdf
- can receive AX.25 packet, too



FX.25 KISS TNC hardware



Features of TNC software

- Implemented by C language
- Running on FreeRTOS
- Each functions implemented as tasks of OS
- Using queues inter task communication
- Using interrupt to read RXD signal of modem
- Using infra red I/F to send the data to modem
- Implements software modem on next version
 - The TNC software is available on GitHub
 - <https://github.com/amedes/fx25-kiss-tnc>

NOTES

ISBN: 978-1-62595-125-0



90000

9 781625 951250