Rick Muething, KN6KB
6143 Anchor Ln., Rockledge, FL 32955; **rmuething@cfl.rr.com**

Tom Lafleur, KA6IQA
Box 3029, Rancho Santa Fe, CA 92067; **tom@lafleur.us**

Tom Whiteside, N5TW
228 Wind Ridge Cv., Georgetown, TX 78628; **tomw@ecpi.com**

IONOS Simulator: An Open Source Ionospheric Simulator

A low cost stand-alone open source ionospheric simulator for ARQ protocols.

This article covers the detailed design and implementation of a single chip microprocessor Ionospheric Simulator based on the popular Watterson model. This article also includes several of the results using the simulator to compare popular HF and VHF/UHF digital ARQ modes.

The project was begun in December 2019 as both a learning tool and proof of concept for a fully open-sourced project to implement a high-quality easy-to-use simulator based on the Watterson model. To make this successful we had to develop hardware and software that provided not only the required DSP manipulation of the audio but also provided a simple-to-learn user interface. A big factor in the successful development was the use of the Teensy Audio library — originally targeted primarily for music applications. We validated the simulator through extensive testing of several HF and VHF digital protocols and compared the simulated results with other Waterson model simulators. This article reviews the approaches taken for the system design, software, hardware, and application of the simulator and presents some of the more important simulation results. Since this is a fully open source project, we have provided all the design details, documentation, and source code on a GitHub site.

The Need for an Ionospheric Simulator

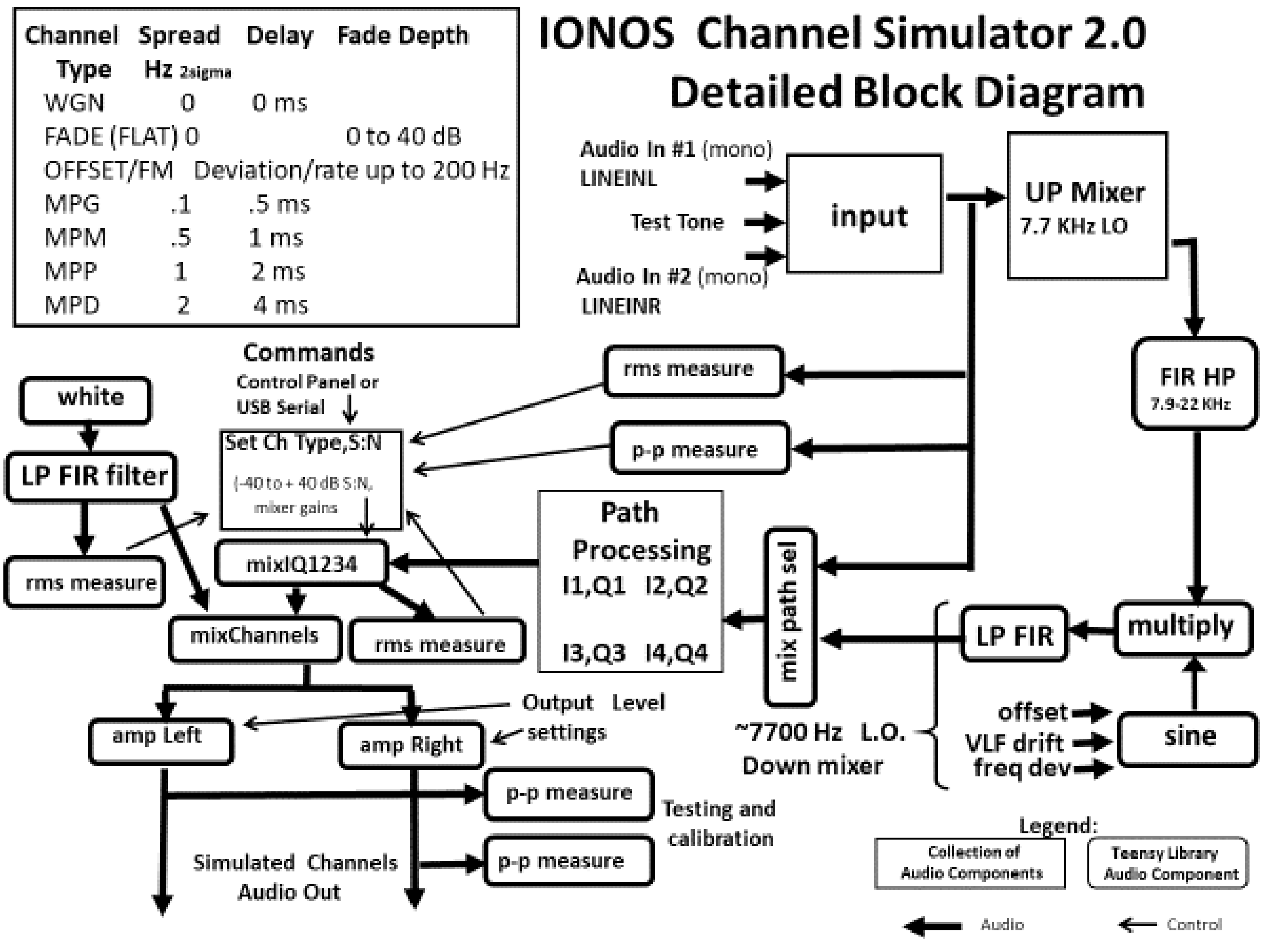
As hams we understand that radio propagation is heavily influenced by the earth's ionosphere. This ionized region of the upper atmosphere allows single or multiple hop communications through continuously-changing paths. In 1970 Watterson, Juroshek, and Bensema presented a landmark paper **[1]** in the IEEE Transaction on Communication Technology. They described a mathematical process (model) that could fairly accurately model and simulate radio propagation over narrow band HF (< 15 kHz) channels. This and follow-on papers by others, and the CCIR/ITU **[2]** defined a set of standardized "representative test channels" that would allow computer software or hardware simulators to statistically model HF radio propagation. This approach has been successful and simulators (hardware and software) built using the Watterson model have proved to be invaluable in the development of high-performance HF and VHF/UHF protocols for both military and ham radio use.

System Design

We wanted to leverage our experience with earlier projects with the higher performance Teensy microprocessors and the use of an earlier basic simulator described in a 1999 DCC paper **[3]** by Johan B. Forrer, KC7WW.

The approach taken was to use the high performance Teensy 4.0 microprocessor **[4]**. All programming and debugging were done using the simple and free Arduino Integrated Development Environment (IDE). We made good use of the Teensy Audio Library **[5]** that processes 16-bit audio (from a stereo sound card chip) at 44.1 kHz sample rate through various audio functions. Many of these library components were initially targeted to musical applications but we required only basic DSP functions such as multiplication, mixing, FFT, FIR filters, sine wave and random generators and rms and p-p measurement tools. A total of 35 Teensy Audio components were used connected by 38 virtual connections as shown in the detailed block diagram in **Figure 1**. This model is detailed and formalized in the C++ source code for the simulator.

We wanted the simulator to be self-contained and OS independent with its own basic user interface. This meant it would have to include at least basic parameter input and display capability. We also included an alternate serial USB interface to allow automation of longer more complex testing. The simulator contains a number of filters such as basic low-pass, band-pass and high-pass along with a few more complex filters for the generation of the imaginary Q components (Hilbert Filter) and the specialized very low frequency Gaussian filters needed for the Watterson model. All filters were designed with free and available on-line tools **[6, 7]**. Another requirement was the ability to use a single simulator to easily model both paths of an ARQ half-duplex connection without any external mixers or switching combiners. This was solved using the stereo input and output capability of Teensy Audio codex assigning one mono channel to each side of



Channel Type	Spread Hz 2sigma	Delay	Fade Depth
WGN	0	0 ms	
FADE (FLAT)	0		0 to 40 dB
OFFSET/FM	Deviation/rate up to 200 Hz		
MPG	.1	.5 ms	
MPM	.5	1 ms	
MPP	1	2 ms	
MPD	2	4 ms	

QX2109-Muething01

Figure 1 — Detailed block diagram of the IONOS simulator.

the connection. Two simulators would be required to model a full duplex connection. Finally, as shown in **Figure 1**, we included a second path through the Upmixer, FIR HP and Downmixer to allow simulating tuning offset and FM drift important in testing some protocols.

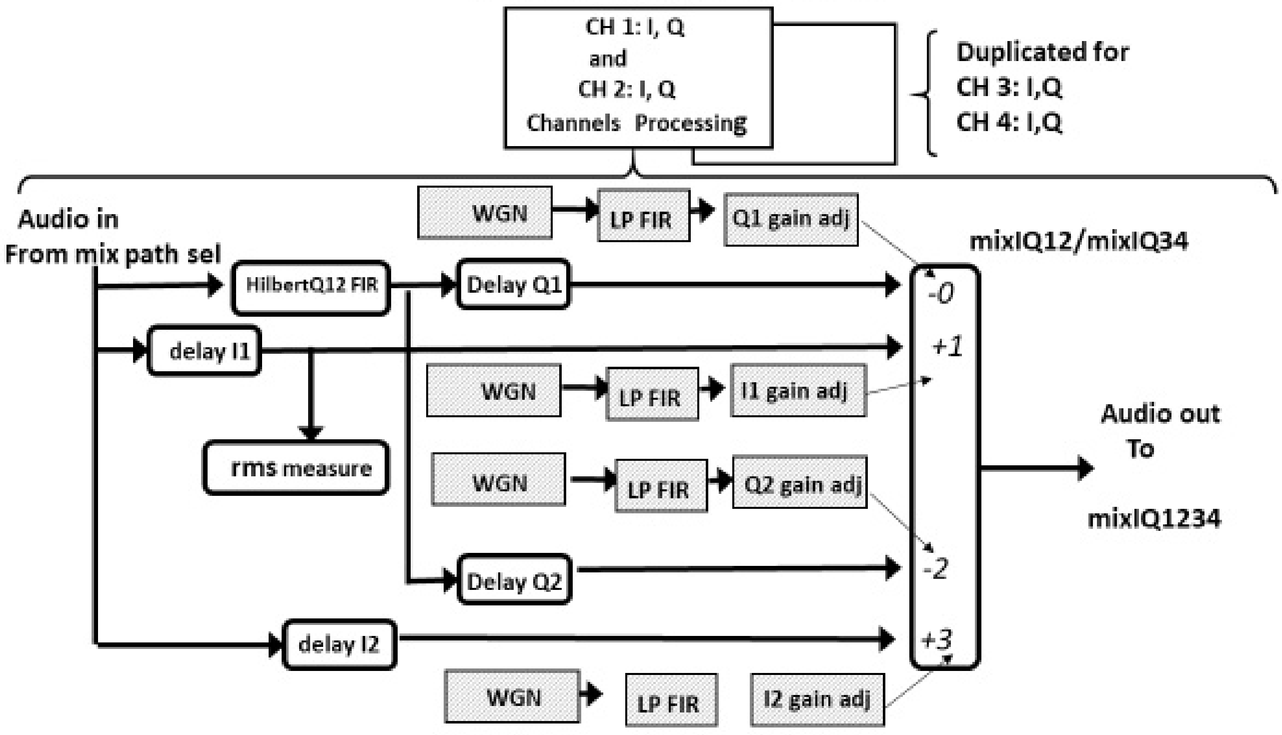
In the upper left corner of **Figure 1** is a summary of the primary operating modes and channels of the simulator. The CCIR/ITU standardized the Doppler spread and delay values of representative channels MPG-MPP along with specific filter requirements to better allow comparison using different simulators.

The implementation of the Waterson model is expanded in the Path Processing Detail shown in **Figure 2**. This implements the modeling and simulation for two or four complex (I and Q) paths. In the Waterson model the variation of these I and Q paths are independent and randomized so this requires a total of four or eight individual paths to implement the two or four IQ pairs.

The basic audio (real component) of the modem output drives two delay elements delay I1 and delay I2 along with a Hilbert Filter, which shifts the real audio 90° to generate the Q (imaginary) component. The Hilbert filter drives two additional delay elements Q1, Q2 that delay the Q (imaginary) components. The I and Q delays are adjusted to offset the delay of the Hilbert filter and introduce the constant path delay defined by the channel to be modeled (e.g., 2 ms delay defined in the MPP channel shown in **Figure 1**). The CCIR/ITU guidelines also put specific requirements on the bandwidth, shape and roll-off of the low-pass filters that are used to generate the low frequency complex modulation of the audio I and Q paths. The final result output is generated by combining — with proper polarity — all the real components (total of 4 as shown in **Figure 2**). This audio, which is now modulated by a low frequency randomized complex vector, is what causes the frequency selective slow fading and peaking that we recognize on HF propagation. The imaginary components are not summed but discarded because only real audio is required by the modem. Additional details on how this important step is done and the details of the various filters used are included in the slide presentation on the GitHub page.

Finally, to accommodate high performance ARQ modes, we paid close attention to audio path alignment and processing delays to keep the overall audio

Path Processing Detail



1. Normal audio processing is done at 44.1KHz sample rate, 16 bit audio.
2. C++ Functions in crosshatched boxes are updated at 64 times the selected path Doppler spread frequency
3. Delay tap values I1,Q1,I2,Q2 are constant for each modeled Path type.
4. mixIQ12 gain signs extract only real parts to send to mixIQ1234 mixer. (see following slide)

QX2109-Muething02

Figure 2 — Path processing detail for two and four I and Q paths.

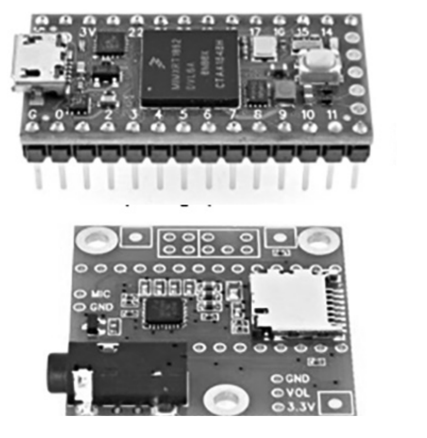
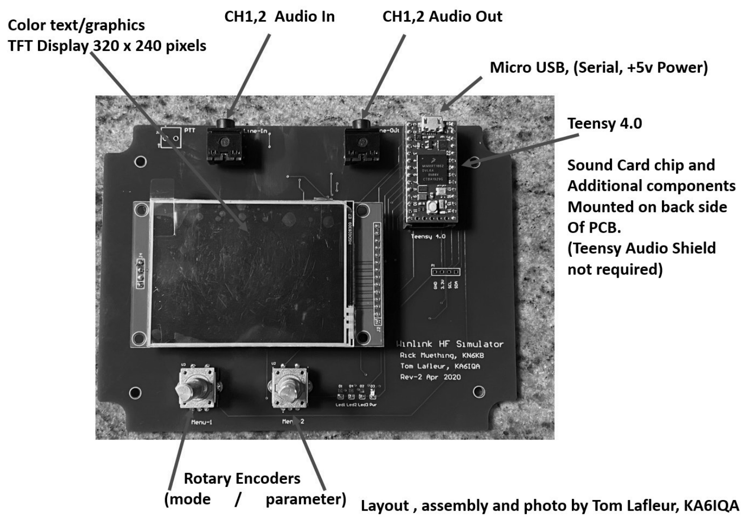


Figure 3 — Teensy 4.0 processor and audio shield.

Figure 4 — Photo of Rev 2 prototype PCB.

Photo of Rev 2 prototype PCB



Layout , assembly and photo by Tom Lafleur, KA6IQA

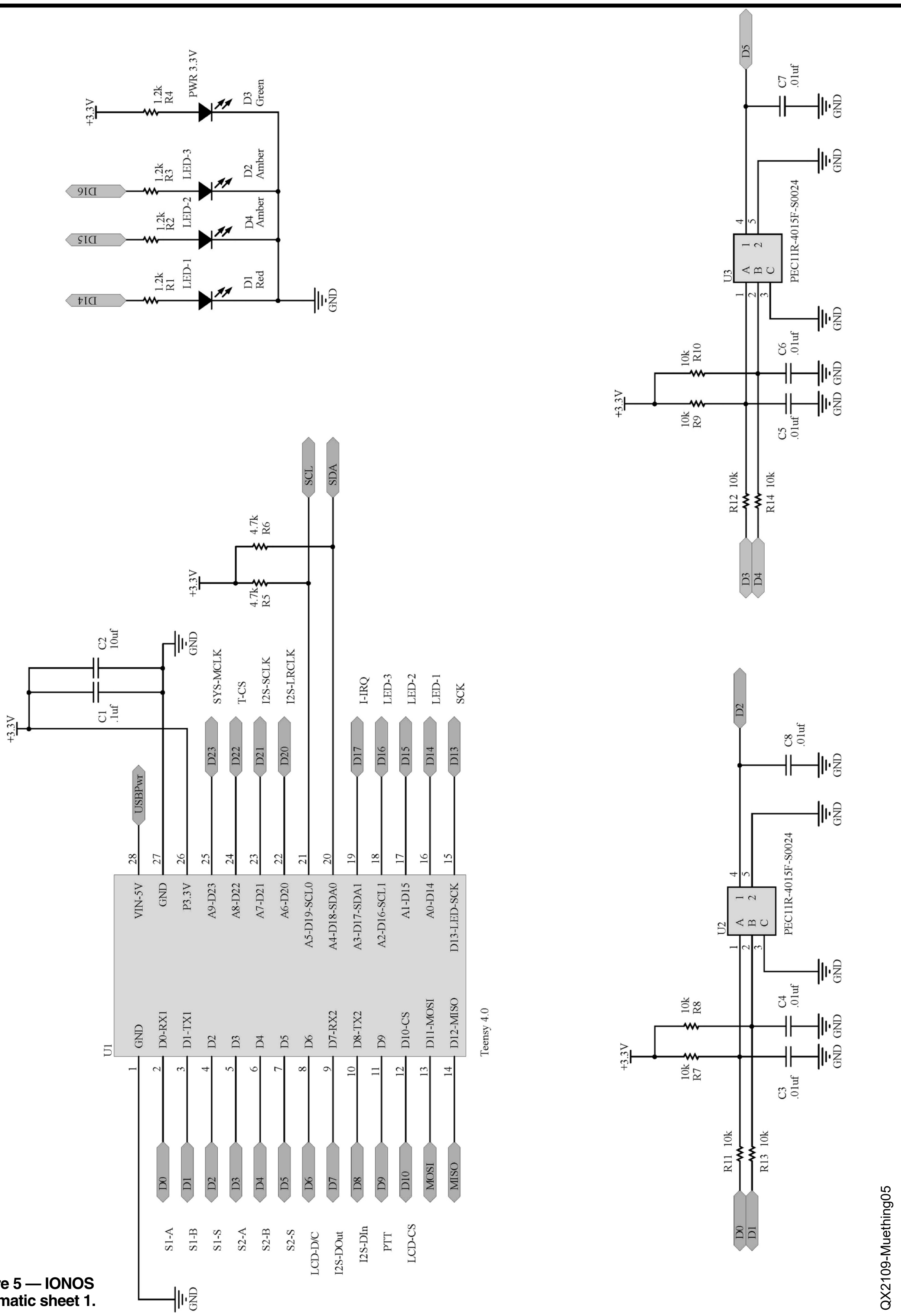


Figure 5 — IONOS schematic sheet 1.

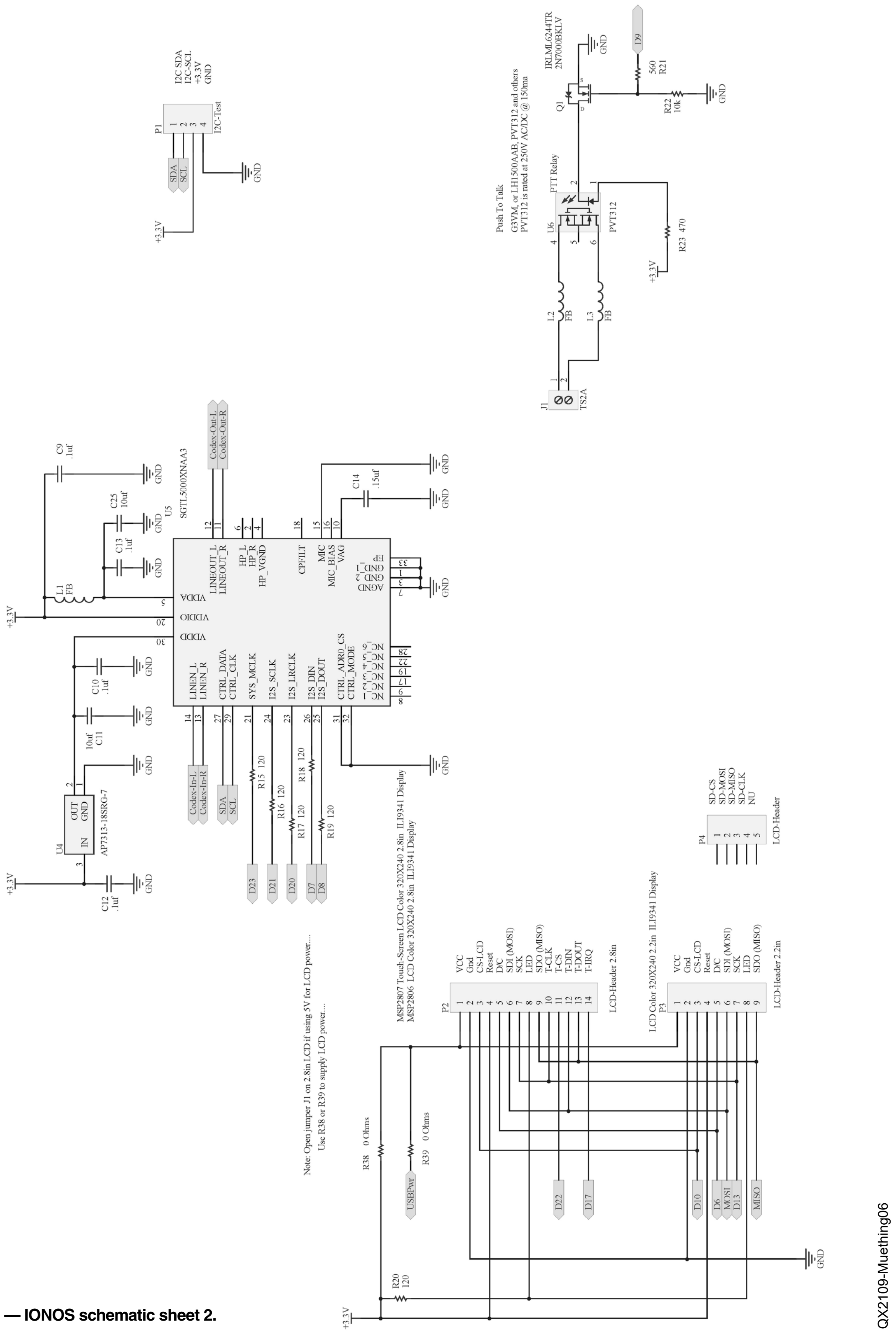

U5
SGTL5000XNAA3
U4
AP7313-18SRG-7
P1
I2C-Test
I2C SDA
I2C-SCL
+3.3V
GND
Push To Talk
G3VM, or LH1500AAB, PVT312 and others
PVT312 is rated at 250V AC/DC @ 150ma
PTT Relay
U6
PVT312
Q1
IRLML6244TR
2N7000BKLV
J1
TS2A
MSP2807 Touch-Screen LCD Color 320X240 2.8in ILI9341 Display
MSP2806 LCD Color 320X240 2.8in ILI9341 Display
LCD-Header 2.8in
LCD Color 320X240 2.2in ILI9341 Display
LCD-Header 2.2in
LCD-Header
Note: Open jumper J1 on 2.8in LCD if using 5V for LCD power....
Use R38 or R39 to supply LCD power....
R38 0 Ohms
R39 0 Ohms
USBPwr
QX2109-Muething06

Figure 6 — IONOS schematic sheet 2.

transit time down to a maximum of less than 10 ms.

Hardware Design

The first proof of concept for this project was testing with a standalone Teensy 4.0 processor with the PJRC Teensy audio codex (shield) board rev D. (see **Figure 3**).

When we were satisfied with the concept, a rev 1 PCB was designed and fabricated that included an LCD display, 2 rotary encoders, the audio codex, some I2C devices, and a standalone power supply. This was used for the early development of the software. We then migrated to a version 2 board, for final development, optimizing a few items that were not needed to simplify the design.

The heart of the simulator is a Teensy 4.0 processor module from PJRC **[4]** that features an ARM Cortex-M7 processor with a floating-point unit, 1024 K RAM and 2048 K Flash EEPROM memory, running at 600 MHz, with a NXP iMXRT1062 processor chip. It's very fast and was selected to provide the horsepower needed to run the simulator code.

The board has two Bourns 12 mm incremental rotary encoders for user interface control, an NXP SGTL5000 16-bit stereo audio codec, support for a 2.2 in or 2.8 in (56 mm or 71 mm) TFT LCD display and op-amps for input buffers to isolate the codec. The op-amps are TI OPA2322, rail-to-rail I/O with ESD clamping diodes on the inputs, and very low noise.

The encoders are first level de-bounced with two 10 kΩ resistor and a 0.01 μF capacitor on the encoder pins and a single 0.01 Ω capacitor on the switch pins. Second level de-bouncing is provided in software with the Encoder2 library package.

The SGTL5000 require both 3.3 V and 1.8 V power for proper operation. The 3.3 V is provide by the P3.3V output from the Teensy module low drop out regulator and filtered for clean power to the codec, an AP7313-18 LDO provides the 1.8 V from the 3.3 V source.

The LCD is a common 2.8 in (or 2.2 in) low cost 240x320 TFT device available from a number of suppliers under part number MSP2806 or MSP2807 with touch screen for the 2.8 in devices or MSP2202 for the 2.2 in. Connection to the Teensy is via an SPI interface. Both boards use an ILI9341 driver chip with readily available driver support in the Arduino environment. The back-light LED drive can be supplied via 3.3 V or 5 V from the Teensy USB power pin by selecting one of two 0 Ω resistors on the

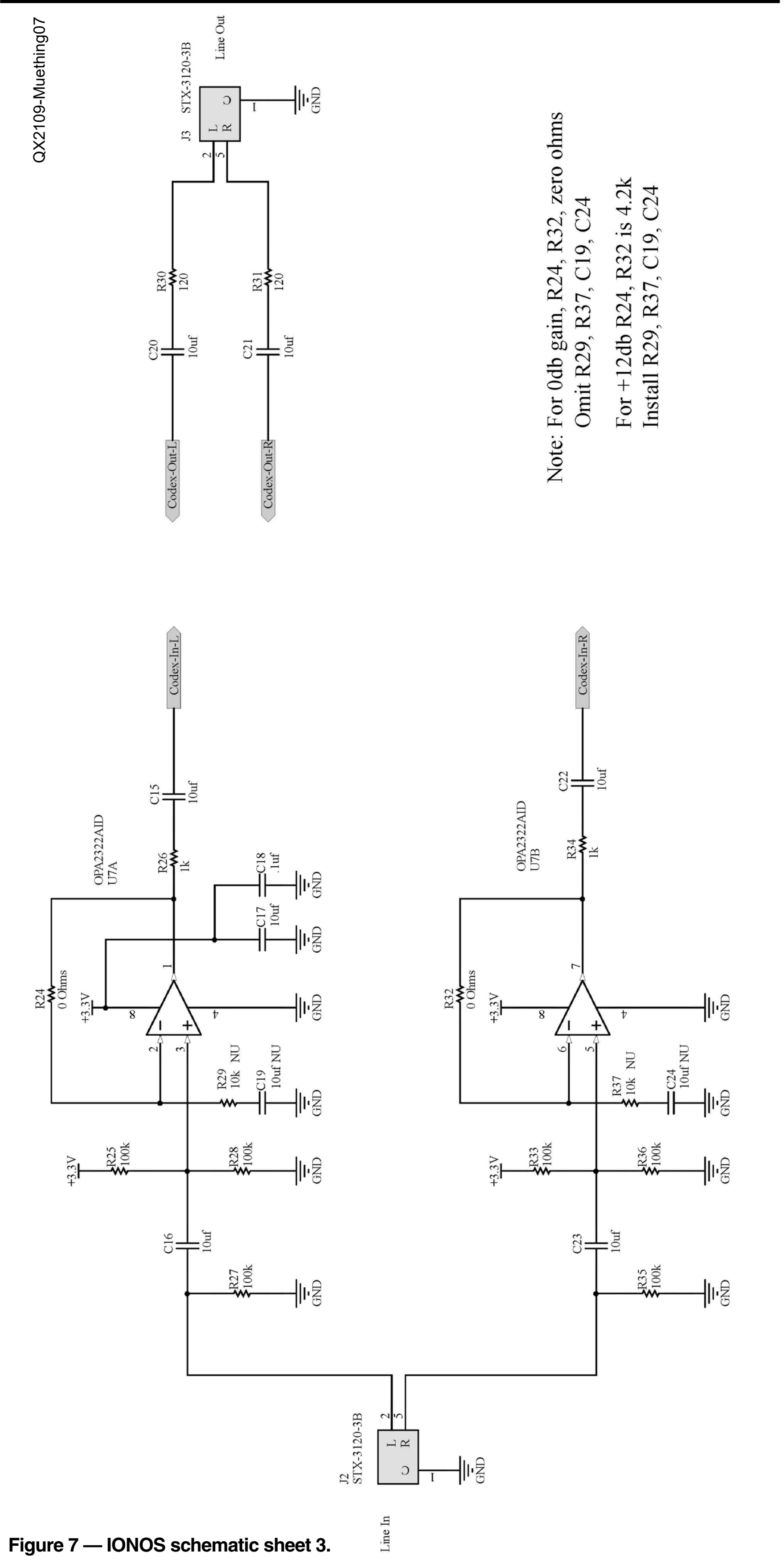
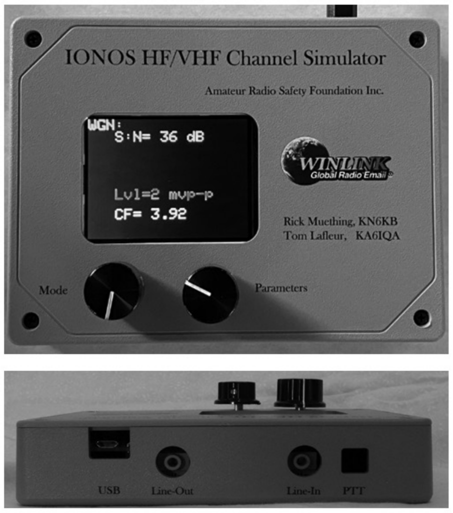


Figure 7 — IONOS schematic sheet 3.

Photo of First prototype production Run



Size: 6.1" x 4.7" x 1.4" (155 mm x 120 mm x 36 mm) (excluding knobs)

Figure 8 — First prototype production run with enclosure.

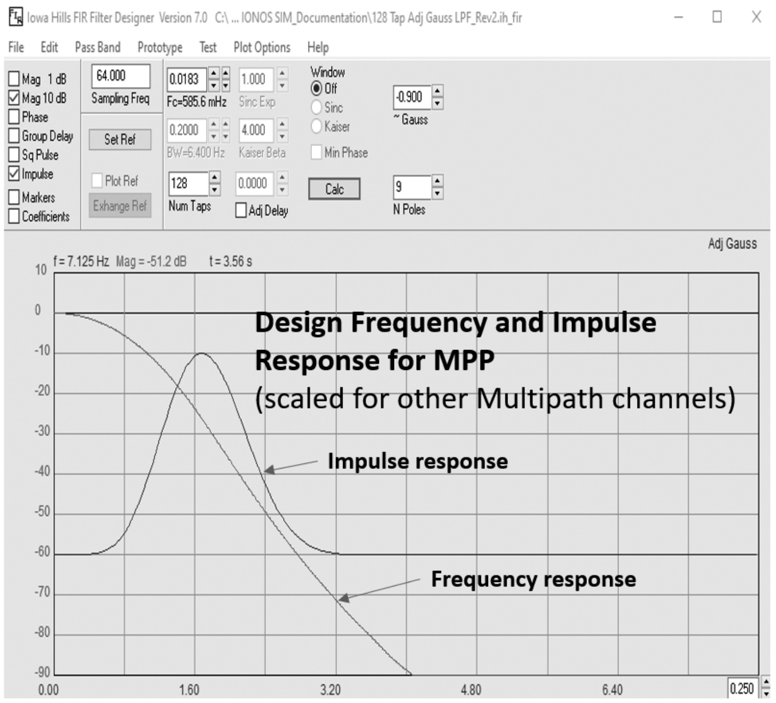
~Gaussian 9 Pole LP FIR Filter:

(used to narrow band filter Gaussian Noise for multipath processing)

Sample rate: 64 x Doppler Spread (2x CCIR recommended minimum)

Follows CCIR guidelines for Bandwidth and roll off.

Iowa Hills FIR Filter Designer Version 7.0



Iowa Hills FIR Filter Designer Version 7.0 C:\ ... IONOS SIM_Documentation\128 Tap Adj Gauss LPF_Rev2.ih_fir

☐ Mag 1 dB
☑ Mag 10 dB
☐ Phase
☐ Group Delay
☐ Sq Pulse
☑ Impulse
☐ Markers
☐ Coefficients

Sampling Freq: 64.000
0.0183 Fc=585.6 mHz
1.000 Sinc Exp
0.2000 BW=6.400 Hz
4.000 Kaiser Beta
Num Taps: 128
0.0000 ☐ Adj Delay
Window: ● Off ○ Sinc ○ Kaiser
☐ Min Phase
~ Gauss: -0.900
N Poles: 9

f = 7.125 Hz Mag = -51.2 dB t = 3.56 s

Design Frequency and Impulse Response for MPP
(scaled for other Multipath channels)

High resolution Spectrum Analyzer measured Response for MPP
(scaled for other Multipaths)

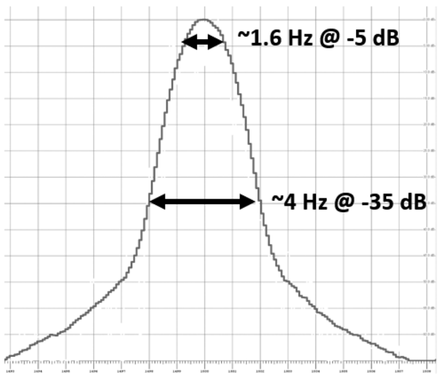


Figure 9 — Gaussian 9 pole LP FIR Filter.

board. Also note that you must cut a jumper on the LCD display if you are using 5 V, see the LCD display data sheet.

The input op-amps are connected as a buffer in the basic configuration, with a 0 Ω resistor connecting the minus input to the output. It can change to provide a +12 dB gain by changing the 0 Ω resistor to 4.2 kΩ and adding the 10 kΩ and 10 µF capacitor to the minus input. The software and parameter encoders provide an input gain adjustment range of 1 to 200 in 1, 2, 5 steps.

The input and outputs are via standard stereo 1/8 in (3.5 mm) audio jacks. The input is terminated into to a 100 kΩ resistor, with a 10 µF capacitor to provide dc isolation, to the + input of the op-amp. Two 100 kΩ resistor provide a half of the 3.3 V as bias on the op-amp input to give symmetrical operation. The outputs from the codec are via a 10 µF capacitor into a 120 Ω resistor to a 1/8 in (3.5 mm) stereo audio jack.

All design schematics, Altium Gerber files, and mechanical files are located on our GitHub page as are details on how to setup the IDE development environment. **Figures 5, 6,** and **7** show the IONOS Simulator schematic sheet. **Figure 8** is a picture of the completed packaged simulator showing screen and USB and audio jack placement.

Simulator Software

The simulator and user interface are completely controlled by the Teensy 4.0 microprocessor and most audio processing is handled by the Teensy Audio Library acting on the 16 bit 44.1 kHz sample rate stereo audio. The software was developed on the simple and free Arduino platform (available for Windows, MacOS and Linux). Approximately 2,200 lines of C++ code implement the Simulator, user interface, serial interface and interfaces to the encoders and TFT display. All the normal audio processing including high- and low-pass FIR filtering is done using the Teensy Audio components and library. The Very Low Frequency (VLF), <3 Hz, low-pass filters, which must meet CCIR/ITU specific bandwidth and response curves, are implemented in *separate* 32-bit-precision floating-point FIR implementations that process at a sample rate of 64 times the selected channel Doppler spread (2 sigma values shown in **Figure 1**). Originally IIR filters were used for these VLF filters but during verification, stability concerns and precision requirements dictated better performance and stability was needed and these filters were changed to modest 128 tap FIR filters, see **Figure 9**.

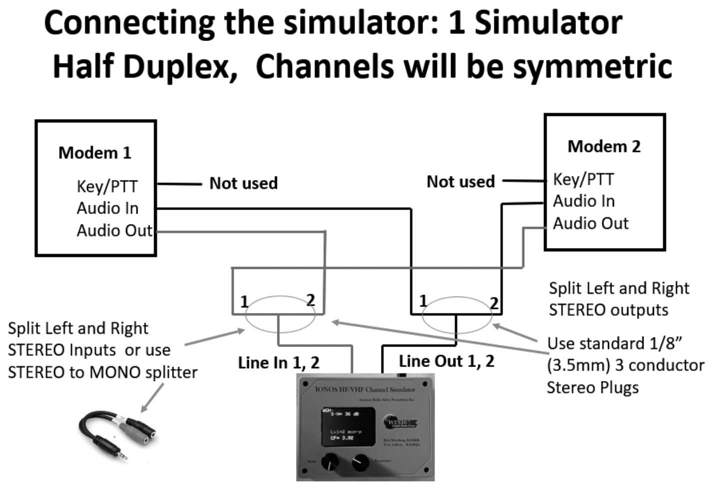
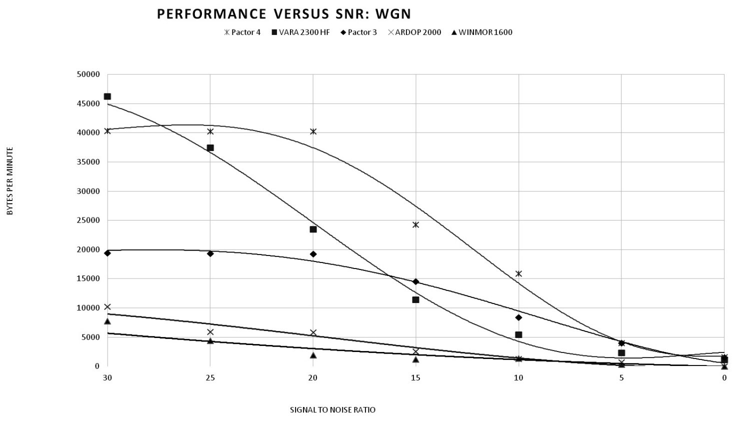


Figure 10 — Modem connections to the IONOS Simulator for half duplex ARQ modeling.



QX2109-Muething11

Figure 11 — Performance verses SNR for 5 ARQ protocols in Additive White Gaussian Noise (AWGN).

Using the Simulator

The simulator is used in the audio path(s) connecting one "station" to another. No radios are used or required. The Simulator creates audio distortion in a precise and "statistically-standardized" way that simulates what would happen due to the RF distortion when the radio wave is passed through the ionosphere on one or more paths at the selected signal to noise ratio (SNR) between –40 and +40 dB. **Figure 10** shows typical connections for a half-duplex modeling using one simulator. The modems could be hardware, firmware, or software DSP (sound card) implementations and are the same as used to connect to the radio's audio inputs and outputs.

Example Simulations

As part of the testing, debugging, and verification of the simulator, we encouraged some interested beta testers to run tests of real-world amateur protocols and modems using the standardized CCIR/ITU channels. This was compared when possible by runs on other software or hardware simulators to compare and confirm results. Runs were typically between 5 and 15 minutes per run at various SNR values for each channel type. Because of the randomization used in the path model and the complexity of adaptive protocols there is always some variation of measured net throughput between runs.

Simulator Results for Various Winlink Digital Modes

As part of testing the simulator, we decided to systematically evaluate the various Winlink digital modes figuring the results would be useful to users deciding which modes were best under what circumstances. For HF ARQ protocols PACTOR 2, 3 and 4 were evaluated as were sound card modes WINMOR, ARDOP and VARA HF 2300. We also simulated VARA FM and AX.25/FX.25 packet for a VHF channel.

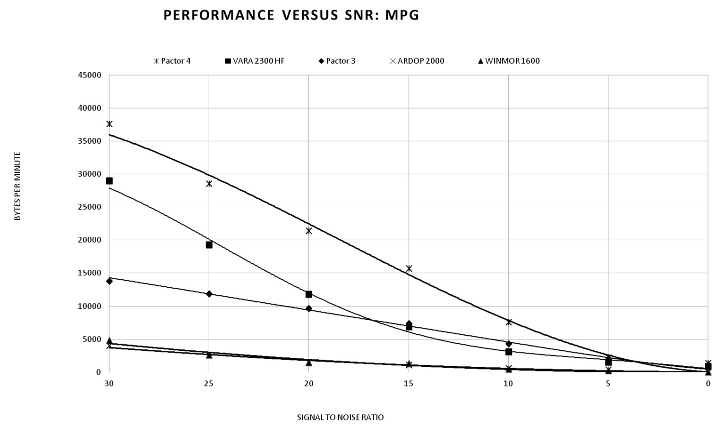
The first example, **Figure 11**, shows results for wideband modes PACTOR 3 and 4, VARA HF, ARDOP 2000 and WINMOR 1600 with 3 kHz Additive White Gaussian Noise (AWGN).

PACTOR 4 clearly beat the other modes over the realistic SNR range likely to be encountered. VARA 2300 did very well edging out PACTOR 3 for SNR better than 17 dB. The older ARDOP and WINMOR modes trailed badly versus the other modes. PACTOR 4 (1.9 kHz) uses less bandwidth than VARA (2.3 kHz) or PACTOR 3 (2.4 kHz). PACTOR 4's higher symbol rate with automatic path equalization delivers over 70% throughput improvement in typical multipath channels. PACTOR 4 cannot currently be used in the United States due to an FCC symbol rate restriction. Imagine what new sound card modes operating without this restriction might achieve.

Figure 12 shows how these same modes stacked up with multipath "good" (MPG) conditions simulated. Results are similar with PACTOR 4 having dominance across the entire range of conditions. As a final HF example, **Figure 13** shows how performance looked for multipath "poor" (MPP) conditions. Again, the same relationship between the modes persisted in this pessimistic HF channel.

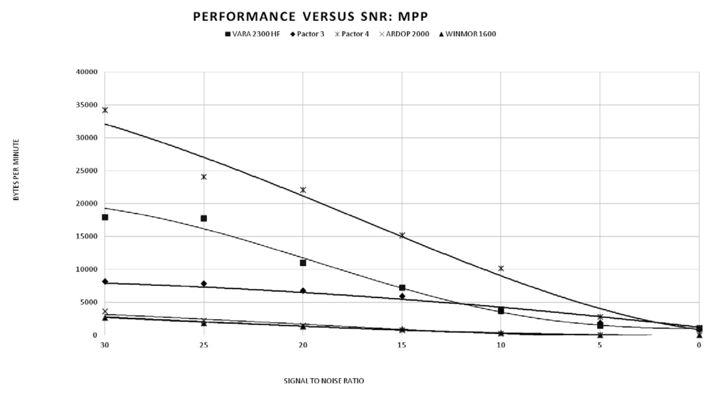
Finally, **Figure 14** shows the results for VARA FM versus AX.25 and FX.25 packet. VARA FM is an exciting new digital mode with much better performance than traditional packet. It is so much faster that we had to use a log scale to depict the difference. FX.25 is a superset of the AX.25 with additional overhead for including forward error correction information. FX.25 did work at SNR levels that AX.25 would not achieve but VARA FM was never slower than FX.25 even under very poor conditions.

The complete results can be found in Tom Whiteside's (N5TW) paper "A comparison of Winlink digital mode



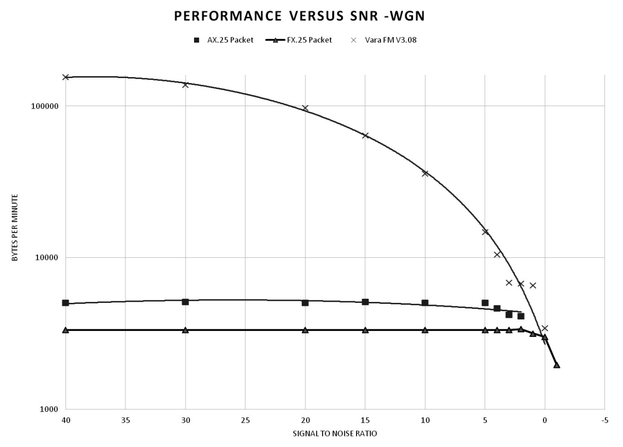
QX2109-Muething12

Figure 12 — Performance verses SNR for 5 ARQ protocols in Multipath Good (MPG).



QX2109-Muething13

Figure 13 — Performance verses SNR for 5 ARQ protocols in Multipath Poor (MPP).



QX2109-Muething14

Figure 14 — Performance for VARA FM and AX.25_FX.25 Packet.

performance based on simulation results using the Teensy IONOS Simulator – Take Three **[8]**.

What we Learned

There were of course minor PCB and mechanical issues, software bugs and also some confusion of how to correctly implement and calibrate the Watterson model. Sections of the code that processed the VLF IIR filters for the path modeling had to be redone as FIR filters with higher precision. We also had to verify calibration against other simulators and compare results using the standard CCIR channels at the same SNR levels. Not expected but beneficial results were modifications to the modem client program (Winlink Express) that were needed to keep the Pactor4 modem buffered to achieve full speed and some firmware modifications to the Pactor4 modem. The sum of these changes contributed to approximately a net 15% improvement in Pactor 4 net throughput for several simulation scenarios. Those kinds of detailed and repeated simulations would have been impossible with conventional over-the-air or basic WGN simulations.

Contributors

We would like to thank all those participating in the project, especially Hans-Peter Helfert, DL6MAA, and Phil Sherrod, W4PHS. Other beta testers included Phil Karn, KA9Q, David Rowe, VK5DGR, and Gordon Gibby, KX4Z.

Availability

The GitHub site **[8]** contains all the information and documentation necessary to duplicate the simulator. The ARSFI will provide a limited number of completed and tested simulators and surface mount PC boards to radio clubs or individuals on a first come first served basis. Ordering information for complete units and boards is available at **https://winlink.org/content/ionos_simulator**.

Rick Muething, KN6KB holds an Advanced Class license, and was first licensed in 1962. He graduated the University of Cincinnati BSEE Honors, 1969 and Northeastern University MSEE High Honors 1972. Raytheon and Honeywell Aerospace computer design. Senior engineer (Military Computer Design) at Applied Technology/ Itek Sunnyvale, CA. Co-founded Paratronics Inc. (Logic Analyser Instruments) as VP Engineering in 1976. Co-founded startup Asix (ASIC chip tester) and went public as Credence in 1992. Part time Graduate school faculty at Northwestern Polytechnic Univ. Freemont, CA. 1992-1995. Consultant to Teradyne (IC Chip Testers) 1996- 2002. Board member and Sect/Treas. of Amateur Radio Safety Foundation. 1997- present Member of Winlink development team doing software and protocol development.

Tom Lafleur, KA6IQA, holds a General Class license, and was first licensed in 1975. Taught computer technology at the college level. Co-Founder GNAT computers an early maker of Intel 8008, 8080 and Z80 systems. Member of technical staff US Naval Electronics Laboratory (NEL) in the Radio Propagation Group's, "La Posta Astro-Geophysical Observatory" 18M radio Telescope facility. VP Engineering and information technology M/A-COM-Linkabit. Director of R&D and VP information technology Digital Research (CP/M). VP Engineering and information technology QUALCOMM and a member of its engineering management team. Co-Founder and CTO Rhythms Net-Connection, an early DSL company. Co-founder, CTO and board member of DriveCam now Lytx, a driving safety company. I am now mostly retired but continue as a board members for a number of companies and continue doing system engineering consulting. I am a member of the Winlink Development Team and on its board of directors.

Tom Whiteside, N5TW, was first licensed as a Technician-plus in November 1995 and upgraded to Amateur Extra Class in April 1996. He received a Bachelor of Science in Electrical Engineering from the University of Texas at Austin in May 1975. He worked as an engineer for the Motorola Government Electronics Division and the Motorola Microprocessor groups from 1975 to 1982. He worked as an IBM development engineer / manager from 1982 to 1993 in microprocessor and systems architecture and was the first co-director of the Somerset PowerPC alliance with Motorola and Apple. In 1993, he moved to California as President of MIPS Technology, Inc where he was a Silicon Graphics Vice President. In 1995, he returned to IBM as a Vice President responsible for Austin Microprocessor development including the PowerPC alliance. He retired from IBM in 1995 as part of a long-term goal to retire on his 45th birthday.

References

[1] C. C. Watterson, J.R. Juroshek, and W.D. Bensema, "1970 Experimental confirmation of an HF channel model," IEEE Transaction of Communication. Technology. Vol COM-18. pp 792-803 Dec 1970.
[2] CCIR/ITU Rec. 520-2 1 9E2c: "Influence of the ionosphere Recommendation 520-2 Use Of High Frequency Ionospheric Channel Simulators," (1978-1982-1992).
[3] J. B. Forrer, KC7WWA, "Low-Cost HF Channel Simulator for Testing and Evaluating HF Digital Systems," ARRL/TAPR DCC 1999.
[4] **https://www.pjrc.com/teensy/**.
[5] **https://www.pjrc.com/teensy/td_libs_Audio.html**.
[6] TFilter: **t-filter.engineerjs.com/**.
[7] Iowa Hills Software: **www.iowahills.com/**.
[8] **https://github.com/ARSFI/HFSimulator**