QEX ARTICLE **Rev 8, 10/30/2020**

(Note this revision is formatted as requested by QST/QEX with text and graphic separated with title Figure references)

Title**: IONOS Simulator: A low cost stand-alone open Source Ionospheric Simulator**

Authors: **Rick Muething, KN6KB Tom Lafleur, KA6IQA Tom Whiteside, N5TW**

Introduction: This is a companion article to an overview of the IONOS Simulator project in QST. This covers considerably more detail in the 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.

This project was begun in Dec 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 [5] (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.

What is 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 a landmark paper [1] by Watterson, Juroshek, and Bensema was presented 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 by Johan B. Forrer, KC7WW [3].

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] which processes 16-bit audio (from a stereo

sound card chip) @ 44.1 KHz sample rate through various audio functions. Many of these library "components" were initially targeted to musical applications but we only required 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 fig 1. This model is detailed and formalized in the C++ source code for the simulator.

Reference Fig 1. Detailed block diagram of the IONOS 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 the connection. Two simulators would be required to model a full duplex connection. Finaly as shown in Fig 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 Fig 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 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 4 individual paths to implement the two IQ pairs. A future software revision of the IONOS Simulator may be expanded to include a total of up to four I Q paths.

Reference Fig 2. Path Processing Detail for two I & Q paths

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 degrees to generate the Q (imaginary) component. The Hilbert filter drives two additional delay elements Q1, Q2 which 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 Fig 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 inf Fig 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 (also 4) are not summed but discarded because only real audio is required to modulate the transmitter. 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 transit time down to a maximum of 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 fig 3).

Reference Fig 3. Teensy 4.0 Processor and Audio Shield

When we were satisfied with the concept, a rev 1 PCB was designed and fabricated which included an LCD display, 2 rotary encoders, the audio codex and 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.

Reference Fig 4 Photo of Rev 2 prototype PCB.

The heart of the simulator is a Teensy 4.0 processor module from PJRC [4], it 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 select to provide the horsepower needed to run the simulator code.

The board has two Bourns 12mm 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 or 71mm) 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 debounced with two 10k resistor and a .01uf capacitor on the encoder pins and a single .01uf capacitor on the switch pins. Second level debouncing is provided in software with the Encoder2 library package.

Th SGTL5000 require both 3.3v and 1.8v power for proper operation, the 3.3v 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 provide the 1.8v from the 3.3v source.

The LCD is a common 2.8 in (or 2.2 in) low cost 240x320 TFT device available from a number of suppliers in the USA and China 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 3.3v or 5v from the Teensy USB power pin by selecting one of two 0-ohm resistors on the board. Also note that you will need to cut a jumper on the LCD

display if you are using 5v, see LCD display data sheet.

The input op-amps are connected as a buffer in the basic configuration, with a zero-ohm resister connecting the – input to the output. It can be change to provide a +12dB gain by changing the zero-ohm resistor to 4.2k and adding the 10k and 10uf capacitor to the -input. (or any other gain desired). The software and parameter encoder provide an input gain adjust range of 1 to 200 in 1,2,5 Log steps.

The input and outputs are via standard stereo 1/8 in (3.5mm) audio jacks. The input is terminated into to a 100k resistor, to a 10uf capacitor to provide DC isolation, to the +input of the op-amp, two 100K resistor provide a ½ of the 3.3 volt's as bias on the op-amp input to give symmetrical operation. The outputs from the codec are via a 10uf capacitor into a 120-ohm resistor to a 1/8 in (3.5mm) stereo audio jack.

All design schematics, Altium Gerber files, and mechanical files are located on our GitHub page as well as details on how to setup the IDE development environment. Fig 8 is a picture of the completed packaged simulator showing screen and USB and audio jack placement.

Reference Fig 5. IONOS Simulator Schematic Sheet 1

Reference Fig 6. IONOS Simulator Schematic Sheet 2

Reference Fig 7. IONOS Simulator Schematic Sheet 3

Reference Fig 8. Photo of first prototype production run with enclosure.

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 on Windows, MacOS and Linux). Approximately 2200 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 (<3 Hz) LP 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 Fig 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.

Reference Fig 9. Gaussian 9 pole LP FIR Filter

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. Fig 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.

Reference Fig 10. Modem connections to the IONOS Simulator for half duplex ARQ modeling.

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 made that were usually between 5-15 minutes per run at various S:N 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. We also simulated VARA FM and AX.25/FX.25 packet for a VHF channel.

The first example shows results for wideband modes PACTOR 3 and 4, VARA HF, ARDOP 2000 and WINMOR 1600 with 3KHz White Gaussian Noise:

Reference Fig 11 Performance verses S_N for 5 ARQ Protocols White Gaussian Noise

PACTOR 4 clearly beat the other modes over the realistic range of Signal to Noise likely to be encountered. VARA 2300 did very well edging out PACTOR 3 for S:N better than 17dB. 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 arcane 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:

Reference Fig 12. Performance verses S_N for 5 ARQ Protocols Multipath Good MPG

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:

Reference Fig 13. Performance verses S_N for 5 ARQ Protocols Multipath Poor MPP

Again, the same relationship between the modes persisted in this pessimistic HF channel.

As a final example, Figure 14 shows the results for VARA FM versus AX.25 and FX.25 packet:

Reference Fig 14. Performance versus S:N for VARA FM and AX.25/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 S:N 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 performance based on simulation results using the Teensy IONOS Simulator – Take Three at: https://github.com/ARSFI/HFSimulator

What was Learned

There were of course minor PCB and mechanical issues, software bugs and, in addition, 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 S:N 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

While the GitHub site contains all the information and documentation necessary to duplicate the simulator, the ARSFI will provide a limited number of completed and tested simulators at a cost of $250 plus shipping to radio clubs or individuals on a first come first served basis. The ARSFI will also supply a limited number of PCBs (require surface mounting components) for a price of $35 with free shipping in

the USA. Ordering information for complete unit and boards is available at https://github.com/ARSFI/HFSimulator .

References:

[1] Watterson, C.C., J.R. Juroshek, & 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] A Low-Cost HF Channel Simulator for Testing and Evaluating HF Digital Systems. Johan B. Forrer, KC7WW ARRL/TAPR DCC 1999

[4] https://www.pjrc.com/teensy/

[5] https://www.pjrc.com/teensy/td_libs_Audio.html

[6] TFilter http://t-filter.engineerjs.com/

[7] Iowa Hills Software http://www.iowahills.com/

Figure References: (See .png and .pdf files)

1) **Fig 1. Detailed block diagram of the IONOS Simulator**
2) **Fig 2. Path Processing Detail for two I & Q paths**
3) **Fig 3. Teensy 4.0 Processor and Audio Shield**
4) **Fig 4. Photo of Rev 2 prototype PCB**
5) **Fig 5. IONOS Schematic sheet 1**
6) **Fig 6. IONOS Schematic sheet 2**
7) **Fig 7. IONOS Schematic sheet 3**
8) **Fig 8. photo of first prototype production run with enclosure.**
9) **Fig 9. Gaussian 9 pole LP FIR Filter**
10) **Fig 10. Modem connections to the IONOS Simulator for half duplex ARQ modeling**
11) **Fig 11. Performance verses S_N for 5 ARQ Protocols White Gaussian Noise**
12) **Fig 12. Performance verses S_N for 5 ARQ Protocols Multipath Good MPG**
13) **Fig 13. Performance verses S_N for 5 ARQ Protocols Multipath Poor MPP**
14) **Fig 14. Performance for VARA FM and AX.25_FX.25 Packet**

Contact Information:

Rick Muething, KN6KB rmuething@cfl.rr.com

Tom Lafleur, KA6IQA lafleur@lafleur.us

Tom Whiteside, N5TW tomw@ecpi.com