Interview with the GNU Radio project's Johnathan Corgan

By **Forrest Cook**
June 8, 2009

The GNU Radio project is one of the more unusual open-source projects to come about in recent years. From the project's web site: "GNU Radio is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readily-available, low-cost external RF hardware and commodity processors. It is widely used in hobbyist, academic and commercial environments to support wireless communications research as well as to implement real-world radio systems."

We conducted an email interview with GNU Radio developer Johnathan Corgan.

Greetings, Could you start off by telling us how long the GNU Radio project has been in existence and how is it funded?

> The GNU Radio project began in 2001 and initial development was funded by John Gilmore and implemented by Eric Blossom. Later, additional work was completed as part of a United States National Science Foundation grant. The aim of the project was to make software radio technology more accessible with lower cost hardware and off-the-shelf computers.
>
> As the project has gained in popularity and user base, most development is contributed by a community of free software developers, though some companies and organizations have paid to have certain features developed for their specific use. All of our project code is assigned copyright to the Free Software Foundation and is released under the terms of the GPL, version 3.

What are your responsibilities with the GNU Radio project?

> I joined the project in 2005. My first effort was to combine the various bits and pieces of GNU Radio that had been spread over several source code trees and repositories. This led to the creation of the gnuradio.org website and release 3.0 in 2006. At this point, there was a single, unified tree and build system for all of GNU Radio, and a central repository, issue tracking system, and Wiki based on Subversion and Trac.
>
> Today I act as release manager, site administrator, and coordinate integration of various developers' code into our development trunk and stable branch releases. Apart from the project itself, I work full-time as a software radio consultant, providing custom GNU Radio application development services and technical training for corporate, academic, and government clients.

How many primary team members does the project have, and what are their functions?

> Eric Blossom continues to lead the overall project direction and scope, and serves as our own "benevolent dictator." He focuses primarily on the core GNU Radio infrastructure itself, and his emphasis on code quality and maintainability has allowed us to sustain heavy growth in the code base in the last several years. Eric also maintains our relationship with the Free Software Foundation.
>
> Matt Ettus, of Ettus Research LLC, manufactures the most popular RF hardware for use with GNU Radio, the Universal Software Radio Peripheral (USRP). While this is distinct from GNU Radio proper, and both projects work with other hardware or software, the combination of GNU Radio and the USRP has become the de-facto platform for open software radio research. Matt has also played a key role in developing

> many of our DSP blocks.
>
> Josh Blum is the developer and maintainer of the GNU Radio Companion graphical editor package. This application has allowed GNU Radio to be more easily used by engineers whose skills lie more in the RF/DSP area than in the programming area.
>
> We have a regular part-time contributor base of about 15 developers. These vary from maintainers of specific architecture ports to development of specific DSP algorithms and documentation and build system updates.

What are some of the more interesting projects that have been accomplished using GNU Radio?

> GNU Radio has been used to develop a very diverse variety of applications such as digital communication links, low-power radar, satellite ground stations for command uplink and telemetry reception, ionospheric research, weather monitoring, direction finding/radiolocation, acoustic signal processing, spectrum sensing, radio astronomy, seismic analysis, marine ship tracking, amateur radio, and many more.
>
> Kestrel Signal Processing has developed OpenBTS, a USRP-based GSM base station implementation with VOIP backhaul, successfully demonstrated at Burning Man 2008. (LWN.net looked at OpenBTS in September, 2008.)
>
> A number of researchers have used GNU Radio as a means to evaluate the security of radio-based systems, such as the recent demonstration of the vulnerability of the MTBA's T card fare system, or the ability to reprogram certain models of implantable cardiac defibrillators. (See this article for more information.)
>
> George Nychis, of Carnegie Mellon University, maintains the "Comprehensive GNU Radio Archive Network" (CGRAN), with a small but growing number of 3rd party applications and blocks for GNU Radio.

Could you give us a brief explanation of the software used on the Linux side of a GNU Radio development system?

> GNU Radio signal processing blocks are implemented in C++, with top-level applications written in either C++ or Python, and the environment is fully based on the GNU toolchain and autotools cross-platform build system.
>
> It is designed as a library to be incorporated into a larger application that might be using other libraries or IDEs, and in general follows typical free software best practices. It has some convenience code for working with the wxPython and QT windowing systems, but these are not required.
>
> While most of its signal processing is implemented natively, including taking advantage of SIMD instruction sets of x86 processors, it does rely on the FFTW Fourier-transform library, the GNU Scientific Library, and the Numeric Python library for some of its functions.

Do any GNU Radio projects use Linux systems to provide the user interface?

> Yes. As a software radio toolkit, it is up to the developer to implement the graphical user interface of their choice, though again, there is high-level support for wxPython and QT.
>
> The GNU Radio Companion allows creation of graphical Python applications using GNU Radio, without the need to explicitly write GUI code.

Could you explain what's new in GNU Radio 3.2, which was announced on May 24?

> GNU Radio 3.2 is the start of a new "stable branch" release series. We make all of our intermediate development work available via Subversion, so in a sense, release 3.2 is not "new", as many of its features have been available for quite some time. We periodically freeze the API into a stable release that developers can code to, and this release branch gets bug fixes and backports of selected functionality in a way that maintains source code compatibility with users' projects.
>
> The release 3.2 API has undergone many internal changes since 3.1. The core

> infrastructure has been made multi-threaded, to better scale with multi-core CPU architectures. The use of Python has become optional; it is now possible to write entire GNU Radio applications in C++. Some internal refactoring of code has sped up the build process and reduced memory requirements. A variety of new DSP blocks have been contributed.
>
> This release supports the recently announced second generation USRP2 hardware from Ettus Research. Full support for native C++ applications for the USRP(1) has been implemented.
>
> The GNU Radio Companion, originally a stand-alone 3rd party application, has been integrated into GNU Radio and has its first official release in 3.2.
>
> Release 3.2 brings formal support to the IBM Cell processor, with cross-build support for Linux on the PS3 platform and support for using the SPE floating point engines.

What are the plans for the next release and beyond in GNU Radio?

> For most of its lifetime, GNU Radio has focused on the lowest layer of radio functionality, what would be called the "PHY layer" in networking terms. This is the domain of continuously streaming, unbounded, sample-by-sample RF or DSP processing, and the GNU Radio system architecture reflects this.
>
> There is strong interest in using GNU Radio for higher layers of radio system functions, where data tends to be dealt with in packetized form. MAC-layer protocols which govern access to the communication channel and TDMA systems which require very tight timing and sequencing of transmit and receive waveforms are examples of this. Furthermore, radio systems often require processing of high-level metadata about signals flowing through the system.
>
> GNU Radio release 3.3 will augment the existing streaming data flow model with a message passing architecture. This will allow developers to write signal processing blocks that operate in either of these domains, to easily cross between them, and to annotate streams or packetized data with metadata that will propagate with them through the signal chain.
>
> The VITA Standards Organization has developed the Digital IF transport standard (VITA 49). This is a standard means of moving digitized RF data over a network or other transport between components of an RF system. Release 3.3 will include an implementation of this standard, with the USRP2 as the first of several hardware platforms to be supported.
>
> As the capabilities of GNU Radio grow, it is important to remain accessible to the newcomer to software radio technologies and to those without formal software development training. The GNU Radio Companion will be expanding the types and complexity of radio applications it can design.

Device driver writers for programmable WiFi cards often have to tiptoe around the frequency regulatory agencies due to the cards' ability to transmit on arbitrary frequencies. Has the GNU Radio project encountered any issues with a software definable transmitter and do you have any plans for dealing with such issues if they should arise?

> GNU Radio itself, as software, has none of these issues, of course.
>
> As the USRP hardware is sold separately, as test equipment, it becomes the responsibility of the end user to comply with any applicable regulatory agency rules regarding the emission (or reception, in some cases) of RF.

Is there anything else you would like to share with our readers about the project?

> While GNU Radio is focused on real-time signal processing applications, it is also possible to experiment with simulated or stored waveforms. Instead of using hardware, one can write GNU Radio transmitter applications that generate waveforms on-the-fly, but record them to files. Receiver systems can be developed using this synthetic data as a source of RF samples instead of actual hardware. Furthermore, these waveforms may be created outside of GNU Radio using applications such as GNU Octave or Matlab.

> Various channel models and channel impairments can be simulated using GNU Radio blocks. We frequently use this method ourselves to implement modulator and demodulator blocks under "carefully controlled" conditions before testing over-the-air. This is an excellent way to learn about RF signal processing without making any actual hardware investment.

Thank you for your time.

> Thank you as well, and for the excellent source of Linux related news over the years.

(Log in to post comments)