LINUX CORNER

Rolling Your Own with Digital Amateur Radio

Gary L. Robinson
Linux Journal Issue #189, January 2010

Fldigi to FLDigiROL: an Amateur Radio operator's digital journey to open source.

Amateur Radio operators are generally free-thinking individualists who don't mind getting their hands dirty to get something done right. Many of us do not think twice about buying a brand-new radio for hundreds or even thousands of dollars and popping the lid on it to see if we can modify it to make it better. You do not have to look hard to find myriad articles on how to modify different pieces of Amateur Radio equipment. So, it is not surprising that we might feel the same way about the software we use.

Open-source software and Amateur Radio are a natural fit. Few operators ever would buy a piece of radio gear if it came with a license that said they could *not* modify it, and it's natural to see why a lot of us navigate toward open source in general and Linux in particular. My personal computing journey started with DOS in 1990, OS/2 in 1993, Windows in 1998 and Linux since 2000. In the true Amateur Radio tradition, I taught myself how to write batch files in DOS, then started tinkering with Pascal. From there, it went to C and eventually, C++. Then, after learning how to use those languages, I took college-level classes to relearn them the right way. It's almost an Amateur Radio tradition to do things backward sometimes and without the manual first.

I started looking at Linux after IBM killed off its OS/2 operating system in the late 1990s. It took more than a few years until I felt comfortable with Linux, but in the past four years, it has been my primary operating system. For the past two years, it has become my *only* operating system on my three desktop machines and my new Netbook.

Open source on Linux already had supplied me with most of the software I needed, with one exception—a suitable Amateur Radio program for digital soundcard modes. Several programs were available, but compared to a few of the Windows offerings, they were feature-poor, and their user interfaces were not as friendly.

I left Windows on my Amateur Radio computer in a dual-boot configuration with Linux until a few years ago when I discovered an excellent digital program called Fldigi (Figure 1) written by Dave Freese, W1HKJ; Stelios Bounanos, M0GLD; and Leigh Klotz, WA5ZNU. It had all the digital modes I was interested in, and it worked very well. Prior to this, I used the Amateur Radio Deluxe DM780 program on my Windows XP partition for most digital contacts. DM780 also is a very fine program and is written by Simon Brown, HB9DRV. He uses open-source code (some of which came from Fldigi) in the digital decoding DLL files for his program, but part of the program is proprietary and, like many programs, I feel DM780 is starting to suffer from feature overload, and in its quest to do everything, it is beginning to get a bit bloated. So, I was excited to find Fldigi for Linux and started using it immediately.

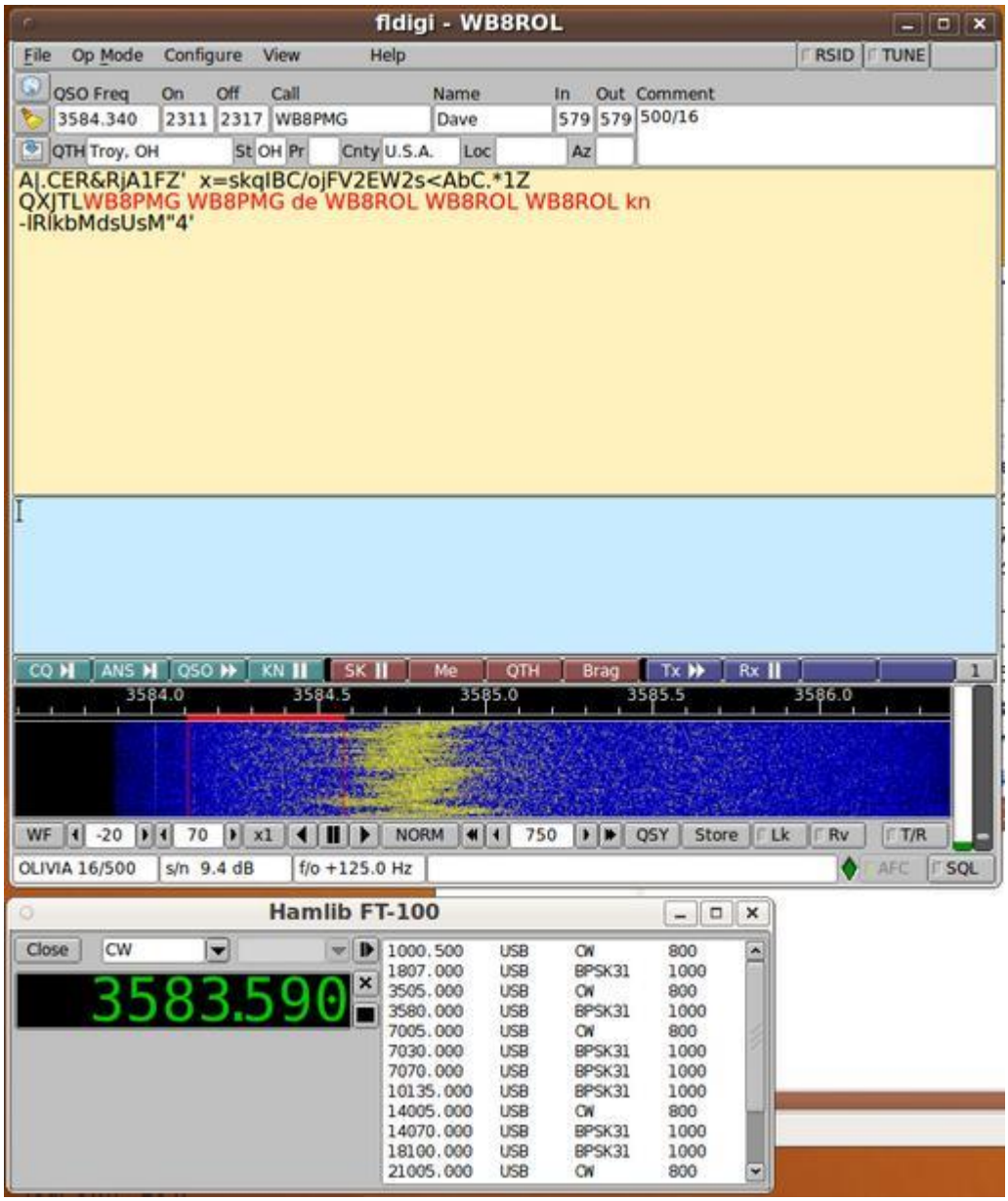

Figure 1. Original Fldigi v 3.10 with Frequency Control Dialog

I used it for several months and was fairly satisfied with it. However, I found some things I did not like and missed a few features from DM780 that Fldigi did not have. (Did I mention that we Amateur Radio operators are a picky bunch?) Fldigi was 95% on the road to where I wanted to be, but that last 5% itched my hide a little bit.

I finally decided to download the open-source code for Fldigi and see what it would take to get that last 5% that would make me happier. I simply could have contacted the authors of Fldigi and sent them ideas and suggestions on how I felt Fldigi could be improved, but everyone's ideas differ as to how software should work and what to expect from it. What might make me happy may not be important to others, or it might be a low priority on a long list. Open source makes it possible for you to have it your way, if you don't mind learning a little and working hard. So, I assembled a list of about 25 (mostly minor) things I thought needed to be changed or added to the Fldigi program.

The documentation for the Fldigi source code was easy to understand, and I rapidly set up a build environment to work on it, including dependencies listed in the Fldigi documentation along with additional code required for development. The user interface part of the program is based on the Fast Light Toolkit (FLTK). FLTK is a lightweight set of libraries that supplies all of the modern graphical controls and window elements most modern computer users would expect to see. I downloaded the documentation for the FLTK and familiarized myself with it as well.

I initially decided to import the whole Fldigi project into a KDevelop project—an easy-to-use IDE that I was already familiar with, which can be used for almost any type of project, not just KDE programs. I could just as easily have used any of a dozen other such environments or a simple text editor and a standalone debugger program. I elected to call my revised version of Fldigi FLDigiROL (Figure 2), adding the suffix of my Amateur Radio call sign (WB8ROL) to the end of the original name.

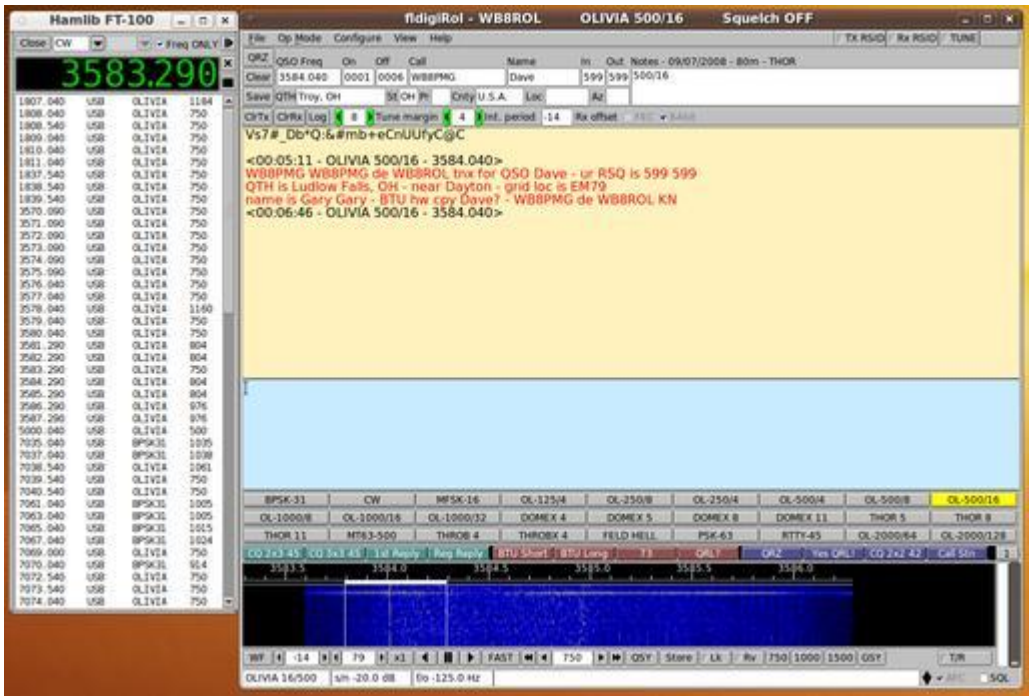

Figure 2. FLDigiROL v 3.10.r31 with Its Frequency Control Dialog, Button Bar and Other New Controls

Many things I wanted to change or add to Fldigi involved the program's graphical interface. The interface is almost ideal for general operation, but parts of it bothered me. I felt I spent too much time accessing menus, submenus and tabbed dialog boxes to change modes and mode parameters as I operated my digital station. Menus and dialog boxes really help organize things and reduce clutter, but I thought that if I had to access certain items continually, perhaps they should be right on the main window. My philosophy always has been that software should do the work and not make me work any harder than I have to.

Amateur Radio digital operation can be challenging, because there are a number of different modes. Fldigi has ten different digital modes, but many of them can be configured in multiple ways, which can be considered submodes, all of which can be decoded only if you can identify the mode and its parameters before the station stops transmitting. For example, my favorite digital mode, Olivia, can be configured 40 different ways. Even an experienced digital operator often will find it difficult to identify these modes accurately and in a timely manner.

Several automatic features in Fldigi help operators determine a digital signal's mode when it is received, but they are not all very effective, especially under adverse band or weak signal conditions. Quite often, you just have to guess and change modes and configurations, as rapidly as possible, and hope you eventually guess the right one before the station quits transmitting. If you have to go through multiple dialog tabs and menu items to accomplish this, it will slow you down and greatly decrease your chance of success.

I decided to add three programmable button bars to the main screen of FLDigiROL. I made them similar to the macro button bar that Fldigi already had, so the existing code was helpful as a model. Each button could be set to a specific digital mode or submode, giving me a total of 27 buttons that could be visible all at the same time. This would hardly be enough buttons for all the possible modes available, but it easily could cover most of the commonly used ones. This would let me switch from one specific mode with specific parameters to another one with a single click of the mouse and greatly speed up the process (Figure 3).

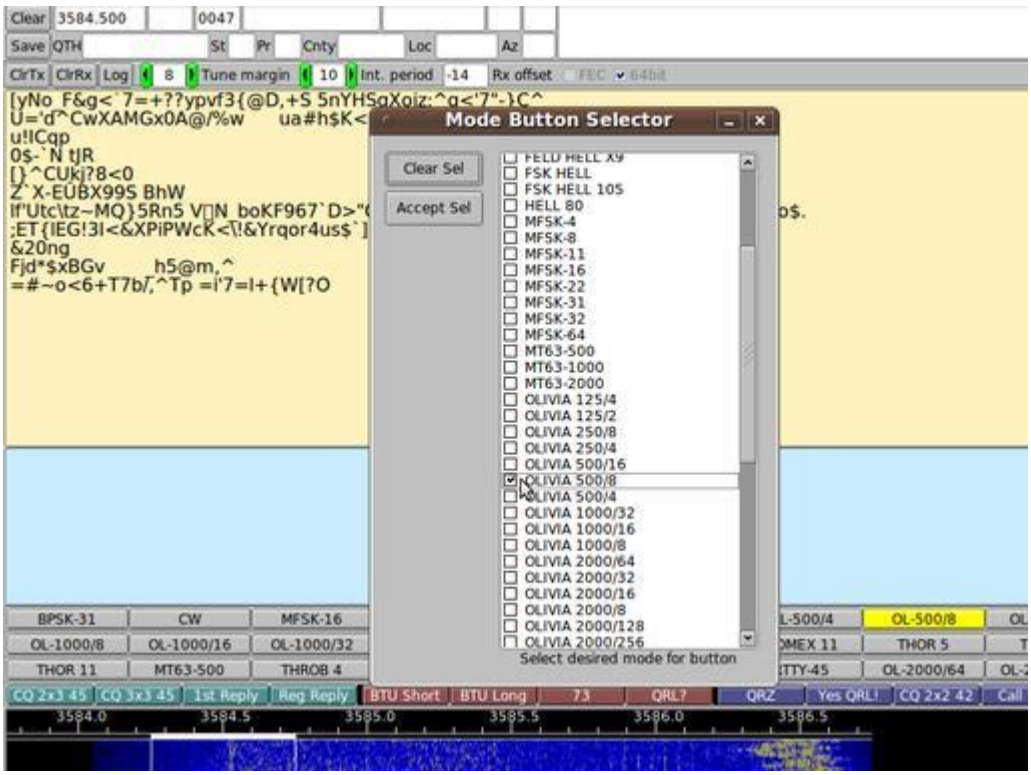

Figure 3. The Programmable Button Bar Dialog for FLDigiROL

I added an additional panel just below the contact (QSO) entry fields with several controls to clear the transmit (TX) quickly and receive (RX) text fields, access the log book, set important Olivia-MT63-DominoEX mode parameters on the fly, and a few other important things that demanded to be on the main screen.

I also added some more user feedback code in the form of both pop-up dialog boxes and information written to the title bar and status bar. I felt that Fldigi did not always let me know what was going on when I clicked some buttons and menu items. The QSO SAVE button, for example, did not give any audible or visual indication that anything occurred when you saved a contact. Often, I would forget if I had saved a contact and would have to check the log to make sure the information was there. I also added code so that if I tried to save the same contact more than once, the program would know it and notify me.

Another thing I really missed from the DM780 program was the auto log lookup feature, which let me know if I had talked to another Amateur Radio station before I entered the call into the CALL entry field. So, I spent more than a few hours writing and perfecting a log-book lookup routine for FLDigiROL that did all of that and also would display the date and frequency (band) information for the previous contact in the label control above the NOTE text field. Later, I added code so that I could simply put my mouse cursor over the call sign in the RX text control and see the same information before I even entered the call sign into the CALL entry field.

I made several cosmetic changes to the FLDigi internal contact log book and also altered the way it saved the contacts to disk. Fldigi did not save log entries directly to disk when I clicked the QSO SAVE button. I discovered that fact when Fldigi crashed on me once after finishing three consecutive contacts and finding out the hard way that it saved the contact information to disk only when the program was closed in a normal fashion. I changed the code so the log book would save the information immediately to the hard disk file and also made the UPDATE button in the log book dialog do the same after I edited an existing entry.

I initially spent a couple months adding features, mods and playing with the code to get it closer to where I wanted it to be. Eventually, during the past year, I added a dozen more small enhancements and had a lot of fun in the process. It was an experience that made me respect the original authors even more than I already did. Because I also documented all my code additions and changes, I was able to download newer versions of Fldigi as they were released and apply my mods in just a matter of days. Some of the things I added to FLDigiROL have since been added to the original Fldigi program. I suspect that most of those items were going to be added anyway and were just waiting in the queue for a long time, but I would like to think my efforts also may have resulted directly or indirectly in a few of them being adopted.

I installed Cygwin, a Windows-based cross-compiler environment, in a Windows virtual machine and eventually managed to make a Windows version of FLDigiROL for a few operators who requested it. However, since the release of version 3.13 of Fldigi, the authors have started supporting the use of MinGW, another cross-platform compiler environment that can be hosted easily in Linux. I still use my virtual machine to test the compiled Windows version of FLDigiROL, but I now use MinGW on my main distro and compile it in Linux. MinGW seems to work much better than Cygwin did for me, and it was a lot easier to set up.

My experience with the Fldigi open-source code has inspired me to try another project. My next goal is to take the Fldigi code and make yet another version—a lite version of Fldigi for new or inexperienced digital operators. I plan to take out most of the configuration settings and simplify the program so that new operators can get it up and running within minutes and not have to do much more than enter call sign, name and home location (QTH). They still would need to select a soundcard and set up their interface COM or USB port(s), but they would not have to worry about a lot of advanced mode or configuration settings. I want to make it simple enough that it will not overwhelm new operators, so they easily can try some of the more advanced modes like Olivia, MT63, DominoEX and THOR. I plan to hard-code a lot of arbitrary default setting choices for them. When they get to the point where they start to feel more at ease with some of the advanced modes and want to have more control, they can move on to the regular release of Fldigi. I want Fldigi-Lite to be a little more like the older versions of the classic DigiPan program, except that it will have more digital modes than just PSK (phase shift keying—another mode).

The power of open-source software has made a big difference in my Amateur Radio and computing life. It's given me a measure of control over the software I use, and it has allowed me to do a lot more than just make suggestions or complain about commercial programs. It's given me the opportunity to make software personal, to learn a lot at the same time and to contribute not just ideas but actual code to the user community. I hope to see more of my fellow Amateur Radio developers (and those who would like to learn how to program) support the Open Source movement and continue to help Linux lead the way. It's part of what open source is all about—sharing and improving software, allowing choice and engendering freedom in general.

Resources

Fldigi: www.w1hkj.com/Fldigi.html

Ham Radio Deluxe: ham-radio-deluxe.com

FLTK: fltk.org

KDevelop: www.kdevelop.org

FLDigiROL Source Code: home.roadrunner.com/~rolswana

Cygwin: cygwin.com

MinGW—Minimalist GNU for Windows: mingw.org

DigiPan Download Page: digipan.net

Gary L. Robinson has been a ham operator for more than 46 years and a programmer since 1993. He is semi-retired and lives with his wife and a small herd of cats in the tiny village of Ludlow Falls, Ohio. Please feel free to send him comments at grobin1949@gmail.com, or catch him on the air for a digital chat.