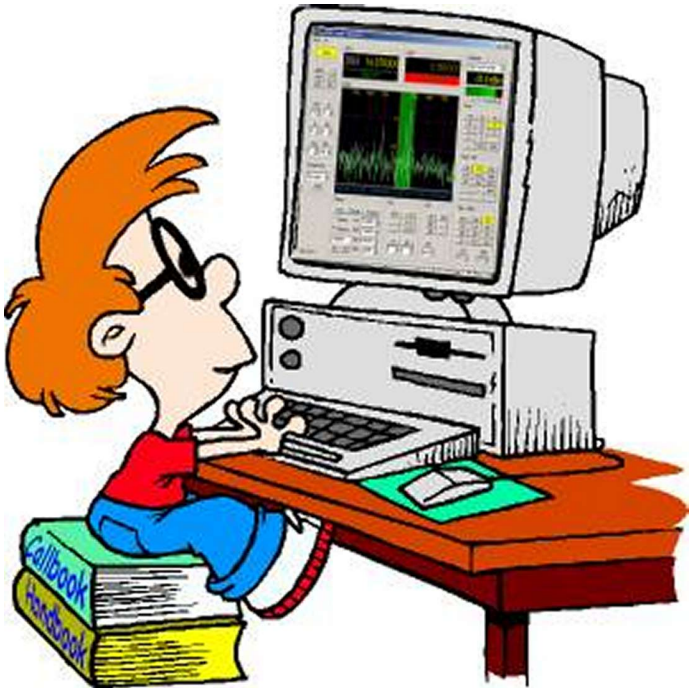
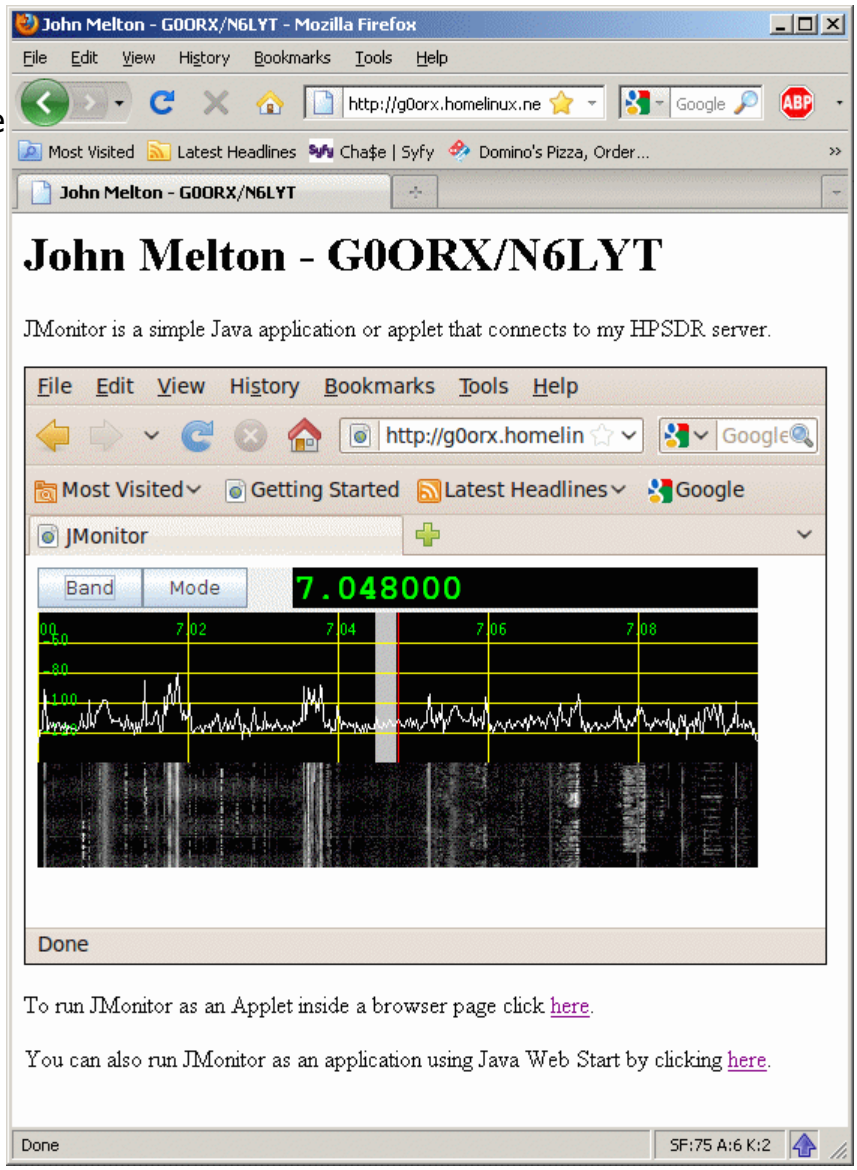


Setting up an online Java Jmonitor server using the EXPERIMENTAL code from John Melton GØORX/N6LYT

*This is **NOT** a commercial endeavor. John created jmonitor as an one EXPERIMENT along his path toward multiple receiver channels from a single Mercury board. John welcomes the input of other Java programmers who may wish to share their expertise with the OpenHPSDR group.*

John created jmonitor and other innovative code as an EXPERIMENT toward his goal of having multiple OpenHPSDR Mercury receiver channels. He did not imagine others' would want to duplicate his setup. Consequently, the documentation and GUI features are the minimum he needed for his testing and do not fulfill the needs of an experienced Mercury operator. John created the program "ghpsdr" that gives a full GUI interface for the Mercury receiver. ghpsdr is included in his svn distribution.



O.K. With that understanding, if you wish to try and setup your Mercury + Ozy + Atlas as an online Internet resource, here are the lessons that I learned and passed along to Chris 4X1RF when he setup his system in Haifa Israel.

First you need to have a subversion utility on your Linux box that will download John's ghpsdr tree. I like RapidSVN, however you may wish to use KDEsvn or some other similar tool.

The location of the "bookmark" svn is here:

svn://64.245.179.219/svn/repos_sdr_hpsdr/trunk/N6LYT

When you have downloaded the N6LYT tree, you will see three main entries(a) ghpsdr; (b) ghpsdr3; and (c) ozy. You can browse the sub-directories to see the interesting code that John has created.

In the folder "ghpsdr" you will find instructions for building ghpsdr the full GUI for

the Mercury receiver. You should follow these instructions to be sure that you have proper USB connectivity to the Ozy board and can load the FX2 and Mercury FPGA's with the proper code. The README is repeated here for your convenience:

========== Melton ghpsdr README ==========

```
ghpsdr - Gtk+ Application for running HPSDR on Linux

ghpsdr is a an application written specifically for HPSDR. It uses a modified
version of DttSP that is ported from the Windows version. DttSP is built as a
static library that is linked with the GUI code.

The following has been verified with a clean install of Ubuntu 9.04.

Building ghpsdr
---------------

You will need to install gtk+, fftw and libusb1.0

    sudo apt-get install libgtk2.0-dev
    sudo apt-get install libfftw3-dev
    sudo apt-get install libusb-1.0-0-dev

If you are going to build DttSP.a you will also need to install libgsl

    sudo apt-get install libgsl0-dev

From the ghpsdr install directory run the command

    make

If all is OK you will end up with the ghpsdr binary.

Included with the the ghpsdr source is a built version of libDttSP.a for 32 bit
Linux.  If you are building on 64 bit linux then copy over the libDttSP.a.64 to
libDttSP.a.  See below for instructions on how to build libDttSP.a.

Running ghpsdr
--------------

To be able to run as a non root user you will need to add a udev script to set the
permissions on the USB Ozy device.

In /etc/udev/rules.d create a file called 90-ozy.rules with the following line in
it:

SUBSYSTEMS=="usb",ATTRS{idVendor}=="fffe",ATTRS{idProduct}=="0007",SYMLINK+="ozy",M
ODE="666"

When Ozy is powered on and plugged in udev will recognize it and create /dev/ozy
with permissions to allow you to access it as a non root user.

You will also need to download initozy and the related files to load Ozy and the
FPGA code.
```

```
From the ghpsdr install directory run the command:

./ghpsdr


Building libDttSP.a
-------------------

Included in this directory is DttSP.tar.gz which when unpacked contains the modifed
source of DttSP ported from the Windows dll.

If you wish to build libDttSP.a then from within the ghpsdr directory perform the
following commands:

    tar xvzf DttSP.tar.gz
    cd DttSP
    make
    cp libDttSP.a ..
================================================
```

I will assume you were successful and have a working ghpsdr GUI that is quite nice and fun to use to tune your Mercury receiver. You should have worked out your audio problems in Linux using either your ALSA mixer or the Pulse-Audio interface.

If you had difficulty, please ask for help on help on the OpenHPSDR email list:

HPSDR Discussion List
To post msg: hpsdr@openhpsdr.org
Subscription help:
http://lists.openhpsdr.org/listinfo.cgi/hpsdr-openhpsdr.org
HPSDR web page: http://openhpsdr.org
Archives: http://lists.openhpsdr.org/pipermail/hpsdr-openhpsdr.org/

O.K. now you are ready to setup up the two servers and the utility "jmonitor" that are needed to run your Mercury receiver on the Internet. Navigate from ghspdr into the ghspdr3 directory. You will see that there are branches (java) and trunk (server) sections.

First install the Sun Microsystems NetBeans 6.8.1 IDE compiler and the JDK/JRE that are needed for a complete java server environment. You can get the code here:

http://netbeans.org/features/index.html

the JDK and JRE may be loaded separately with Synaptic package manager or downloaded the convenient NetBeans + JDK bundle from here:

http://java.sun.com/javase/downloads/widget/jdk_netbeans.jsp

When you have NetBeans loaded and ready, open it from your Applications ---> Programming menu. On the upper left you will see a FILE ---> open project option. Use that option and navigate to the N6LYT/ghpsdr3/bundle/java/jmonitor folder where you will automatically find the jmonitor project. Open the project and look for the symbol at the top of the NetBeans window that looks like a Hammer and Broom. That is the "CLEAN BUILD" icon. Do a clean build and you should succeed with no errors. The code even suggests where you can go to execute the newly compiled jmonitor code.

If you like to use the command line, open a terminal window in the N6LYT/ghpsdr3/branches/java/jmonitor/dist directory. Here is the command used to execute the jmonitor application and point it to one of the existing online servers: **java -jar "jmonitor.jar" --server 24.192.100.58** for example.

Using Synaptic package manager, make sure you have Apache2 installed and ready to operate.

In the folder **N6LYT/ghpsdr3/branches/java** you will see a "**www**" subdirectory. sudo cp -r the contents of that folder into /var/www/

Now you should be able to connect your jmonitor to your own local server. Use the following command **java -jar "jmonitor.jar" --server 127.0.0.1** (which is obviously localhost).

I assume you were successful. Lets go on to build the server components. Navigate to **N6LYT/ghpsdr3/trunk/src** where you will see there are 9 sub-directories. You need to navigate into each sub-directory and run a the "make clean" and "make" command. Be careful to note any errors and ignore the warnings. The receiver should load properly with the Dttsp fftw3 library that is needed by the dspserver. The output of the compiles appears in the sub-directory "**bin**" just one level above the source.

NOTE: Chris notes an important additional step that I forgot; one needs to edit the jmonitorapplet.jnlp and jmonitor.jnlp files which have John's codebase url and needs to be changed to YOUR URL so that you serve the jnlp code.

If you were successful, you can go to the bin sub-directory and in separate command / terminal windows, start each of the online servers. The commands are:

./server --receivers 1 --samplerate 96000

(note the word receivers is plural for future experimentation)

and

./dspserver --receiver 0

(note the word receiver is singluar in this instance)

as shown in John's README. John separated the DSP and USB interfaces for full flexibility in the future. The servers can run on separate computers if you like. the ./server much be physically connected to the USB port of the Ozy board to load the .hex and .rbf files.

If you had difficulty, please ask for help on help on the OpenHPSDR email list:

HPSDR Discussion List
To post msg: hpsdr@openhpsdr.org
Subscription help:
http://lists.openhpsdr.org/listinfo.cgi/hpsdr-openhpsdr.org
HPSDR web page: http://openhpsdr.org
Archives: http://lists.openhpsdr.org/pipermail/hpsdr-openhpsdr.org/

Go to your router and as Admin setup up your outside IP address to NAT to your Mercury connected computer. This is often seen as a "virtual server" on an inexpensive router.

If you have done everything correctly, you can now start jmonitor with your outside IP address:

java -jar "jmonitor.jar" --server {*your outside IP address*}

You may wish to go to the /var/www sub-directory and edit the your-callsign.html, jmonitor.html and index.html entries to suit your own location and preferences. When a user encounters the main jmonitor page he is given two choices: (a) to run the java applet from his browser, or (b) to run the Java Windows Start application that downloads the jar file. I prefer the second method. I get much better results with Firefox on both my Linux and WindozeXP machines.

When you have everything operating to your satisfaction, please publish a note to the OpenHPSDR list to announce your receiver is online.

Here are the three receivers that are currently online when there is no threat of lightning from local thunderstorms:

Chris in Haifa Israel 4X1RF that his Mercury is now online:
http://4x1rf.no-ip.org

John Melton is in the UK:
http://g0orx.homelinux.net/jmonitor.html

Ken N9VV in Chicago, IL USA:
http://24.192.100.58/jmonitor.html

best regards, Ken N9VV and Chris 4X1RF

APPENDIX A Customization

I like to use NetBeans to make source changes. You can expand the tree display on the left hand panel to see all the source modules.

ControlPanel.java to find the default frequencies for the bands.

SpectrumPanel.java around line 210 for the color of the spectrum line color

```
 // plot the data
graphics.setColor(Color.GREEN);
graphics.drawPolyline(X,Y,X.length);
```

Main.java for the default frequency, mode and filter when the application starts:

```
Client client=new Client(server,receiver,audio);
client.start();
client.setFrequency(780000);
client.setMode(6);
client.setFilter(-4000,4000);
client.setGain(30);
```

Alphabetical Index