RXAMADRM - A LINUX PROGRAM FOR DIGITAL SSTV

Ties Bos - PA0MBO

April 12th, 2012

Abstract

A linux program was developed for the reception of digital SSTV picture transmissions conforming to the HAM DRM standard as developed by HB9TLK. It is modeled after the MATLAB program DIORAMA for the reception of commercial DRM broadcast stations.

Keywords: Digital SSTV - Hamradio - DRM - Linux

INTRODUCTION

The program RXAMADRM can be used to receive (picture) files in the HAMDREAM standard as developed by HB9TLK [1]. In contrast to the programs based on HB9TLK's code like WINDRM [2], EASYPAL [3] and DIGTRX [4], RXAMADRM runs under Linux and its source code is available. It is published under the GNU General Public License (GPL), so you can freely modify and improve it.

In the HAMDREAM standard published by HB9TLK [5] some elements are missing, especially the θ_{1024}-values for the E-mode. These values were obtained by analyzing the signals generated by DIGTRX. The code was not built from scratch, but is based on the program DIORAMA that was written by Andreas Dittrich and Torsten Schorr [6] from the university of Kaiserslautern. DIORAMA is written for MATLAB, a high level interpreter for complex mathematical operations with very nice graphical and GUI-tools.

Initially DIORAMA was adapted to the amateur standard. However for amateur use MATLAB is rather expensive and to circumvent the use of MATLAB, the MATLAB-specific code was translated to standard C.

An important addition in RXAMADRM compared to DIORAMA is its "frontend". DIORAMA uses a 12 kHz IF signal for input, which necessitates the use of special hardware. In ham practice it is easier to use the received audio signal directly from the receiver. RXAMADRM transforms the acquired audio signal from the receiver in software to a 12 kHz IF which can be processed by the original filter/demodulator code of DIORAMA that has shown its worth.

The choice of DIORAMA as a starting point instead of DREAM (on which all existing Ham DRM SSTV programs are founded) was based on the paper "*Digital Radio Mondial (DRM) Receiver using MATLAB*" by T. Schorr [7] et.al. Moreover DIORAMA turned out to give a somewhat better performance in the reception of the BBC station on 1296 kHz when compared to DREAM under marginal conditions.

Because RXAMADRM is receive-only, its user interface could be kept simple. All pertinent data of the decoding process are shown during receive, the most important being the Power Spectral Density (PSD) of the received signal, TIME SYNC, FRAME SYNC, FAC CRC and MSC CRC. The PSD, the SNR of the received signal and the DC-offset of the received signal allow for easy tuning.

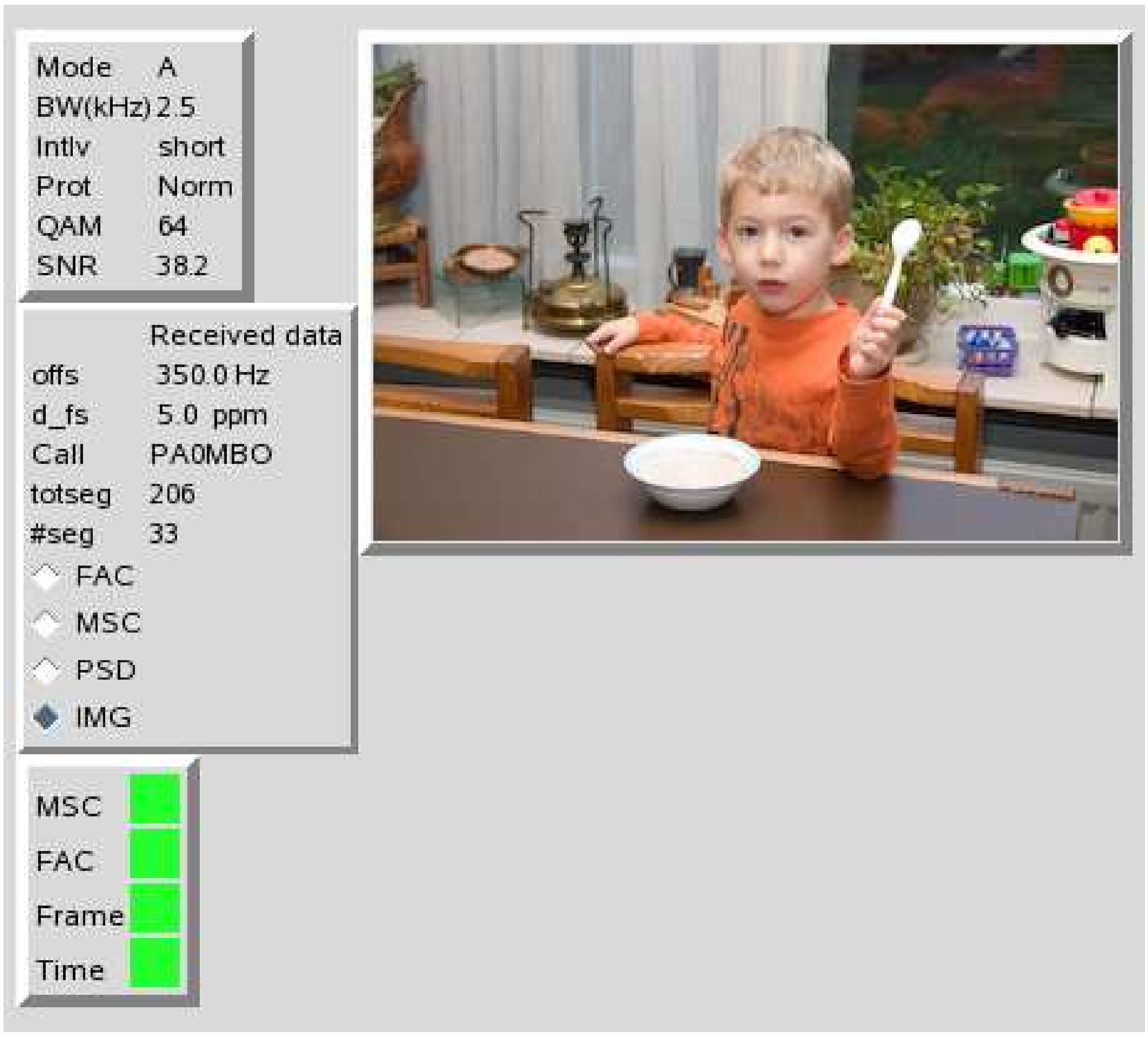
The screen of RXAMADRM shows either the last picture received, the FAC- or MSC-constellations or the PSD. The choice is made via radiobuttons.

Figure 1 shows the screen with a received picture on display, whereas figure 2 shows the MSC-constellations.

The graphical user interface (GUI) was developed with Tk/Tcl as a frontend for the console oriented C-program that takes care of the decoding of the received signal to an image that is stored on disk. The latter program can run standalone and then just fills the picture directory on disk with image files that happen to be received correctly.

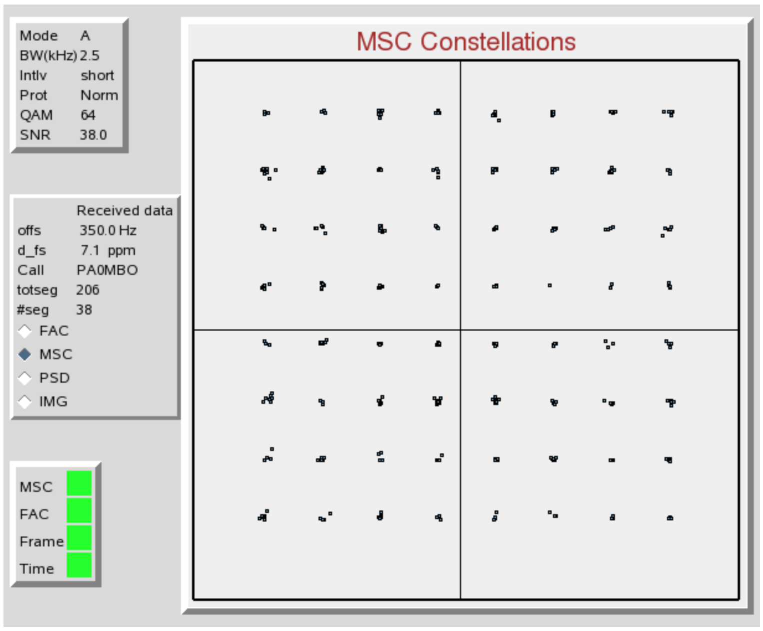
Tk/Tcl is also used for displaying the image files that are received. This limits the images that can be displayed by RXAMADRM itself to image formats that are supported by Tk/Tcl, i.e. bitmap- and gif-files. If the package Img, that is freely available, is installed into Tk/Tcl much more formats can be accommodated.

In the remainder of this paper the signal processing program and the Tk/Tcl GUI are described, instructions for their installation and use are given and some results of its use are presented. The frontend signal processing that allows the use of the audio output of your receiver directly, is described in some detail, as this is new and not modelled after the DIORAMA code. Finally some suggestions are given for (future) possible developments/changes.



```
Mode      A
BW(kHz)   2.5
Intlv     short
Prot      Norm
QAM       64
SNR       38.2

          Received data
offs      350.0 Hz
d_fs      5.0  ppm
Call      PA0MBO
totseg    206
#seg      33
◇ FAC
◇ MSC
◇ PSD
◆ IMG

MSC    🟩
FAC    🟩
Frame  🟩
Time   🟩
```

Figure 1: GUI in picture mode



```
Mode      A
BW(kHz)   2.5
Intlv     short
Prot      Norm
QAM       64
SNR       38.0

          Received data
offs      350.0 Hz
d_fs      7.1  ppm
Call      PA0MBO
totseg    206
#seg      38
```

FAC [✓ MSC] PSD IMG

MSC	🟩
FAC	🟩
Frame	🟩
Time	🟩

Figure 2: GUI showing MSC-constellation

OVERVIEW OF RXAMADRM

The signal processing tasks that RXAMADRM has to perform are organized in a number of modules, each in a separate soubroutine or procedure.
RXAMADRM follows the organization of DIORAMA:

1. acquisition of 400 msec worth of sound data
2. demodulation/equalization
3. channel decoding
4. source decoding

These signal processing tasks are gathered in one standalone program called **drmtst**. Its main routine has an infinite loop in which the subroutines that perform these 4 tasks are executed in sequence repeatedly. The code of RXAMADRM follows the DIORAMA code with one exception and this concerns the handling of incompletely received files in the source decoding part. This will be described later. Those parts of RXAMADRM that are straightforward translations of DIORAMA's MATLAB-code into C-code will not be described here; for this the reader can consult [7]. In the following only the Linux and hamradio specific details of RXAMADRM are given.

Besides the signal processing tasks, RXAMADRM offers a (barebones) GUI that allows the user to view the progress and the quality of the reception of the SSTV images. The code for this GUI is written in Tk/Tcl and is strictly separated from the signal processing code. This code is in the file **rxamadrm.tcl**.

PROGRAM DETAILS

Real-time aspects

RXAMADRM or rather **drmtst** is a real-time program, it calculates its result (the image file that was sent) **during** the reception of the signal. As the audio signal comes from the receiver continously, but has to be analyzed in chunks, some way has to be found to read the sound data into the computer and analyze it simultaneously.

In most Linux systems the recording of sound input via a soundcard is handled by the ALSA soundsystem. This system sees to it that the sound signal which is presented to the microphone or line input connector of the soundcard is sampled by an Analog to Digital Converter (ADC) and stored in an internal buffer. Application programs like RXAMADRM have to see to it that the content of this internal buffer is processed before it overflows and part of the incoming signal is lost. Unfortunately the size of the internal buffer of ALSA is much smaller than corresponds to the duration of the sound that we need to be able to process it. The information contained in the sound signal in DRM is organized in socalled frames with a duration of 400 ms. We need at least this amount of recorded sound to be able to decode some data from it. If the recording is not synchronized with the frame we need even more.

In RXAMADRM this problem is tackled by providing an extra layer of buffering between ALSA and the processing of the recorded sound. This bufffering should take place at the moment that ALSA's internal buffer has not yet overflowed and during the time that the processing of the last completely received frame can still be in full swing. The ALSA sound system provides a way to **signal** that its internal buffer needs attention. Via the operating system Linux this signal can be used to trigger an **asynchronous** handler routine. In RXAMADRM this asynchronous signal handler fills a (larger) buffer in RXAMADRM itself by interrupting the work it was doing. As soon as the transfer of sound samples from ALSA's internal buffer to RXAMADRM is completed, RXAMADRM resumes the work it was doing.

Synchronization between ALSA and RXAMADRM now is simple: RXAMADRM only has to check whether there is enough recorded sound in its own buffer. If not, it should wait until there is. On the other hand, if RXAMADRM's own buffer overflows before it is emptied, the processing of the former data has taken too long. If the latter situation arises stopping other task running on the computer may help; if not the only thing to do is to look for a faster CPU.

Audio frontend

The DIORAMA (and DREAM) software is designed to operate on an 12 kHz IF. To be able to use this software you need to offer the signal of interest at this 12 kHz IF to the soundcard. Generally this is realised by a small piece of hardware that mixes the 455 kHz IF from the receiver with an oscillator signal at 467 kHz. To be able to use the audio output of a receiver with the demodulator/filter software as designed in DIORAMA the audio of the receiver should be shifted to this 12 kHz IF after it has been sampled by the soundcard. The ARRL Handbook [8] shows how this can be done by using a Hilbert transformer in a half complex mixer.

The FIR-filters for the Hilbert transformer in RXAMADRM were designed with the use of MATLAB. The design was started with a 83 tap low pass filter with a passband frequency of 3000 Hz and a stopband frequency of 3300 Hz. This lowpass filter was shifted to a bandpass filter with a passband from 200 to 3200 Hz by multiplying with a complex exponent at 1700 Hz. The resulting in-phase and quadrature filtercoefficients are incorporated in the program in the routine **monorec.c** in the variables *B_Inphase*[83] and *B_Quad*[83] respectively.

Graphical User Interface

As stated above, the signal processing in RXAMADRM is strictly separated from the GUI. All signal processing work is done in the standalone program **drmtst** that operates autonomously. This program creates a separate file on disk for each file that it receives correctly, either a picture or any other file. These files are placed in the subdirectory **./pics**. Furthermore **drmtst** determines the DRM-mode, bandwidth, the type of QAM modulation, the interleaving, protection mode, tuning offset and SNR of the signal and prints these data to standard output.

The GUI is implemented in Tk/Tcl in the script **rxamadrm.tcl**. This script starts **drmtst** and pipes its standard output into this script. The data produced in **drmtst** is now intercepted by the script and put into variables. These variables in their turn can then be used to construct a nice screen in which the progress of the reception of the images or files can be visualized. If you don't like the text output of **drmtst** continously scrolling on your terminal screen you can start the program with the script **startdrm** that redirects this output to the bitsink /dev/null.

If you use **drmtst** on its own you can redirect its standard output to a diskfile and later see all its data with the aid of some editor. If

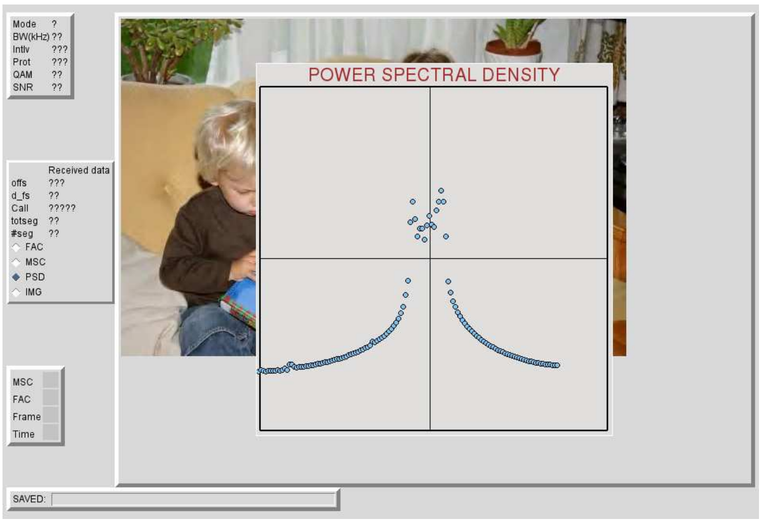
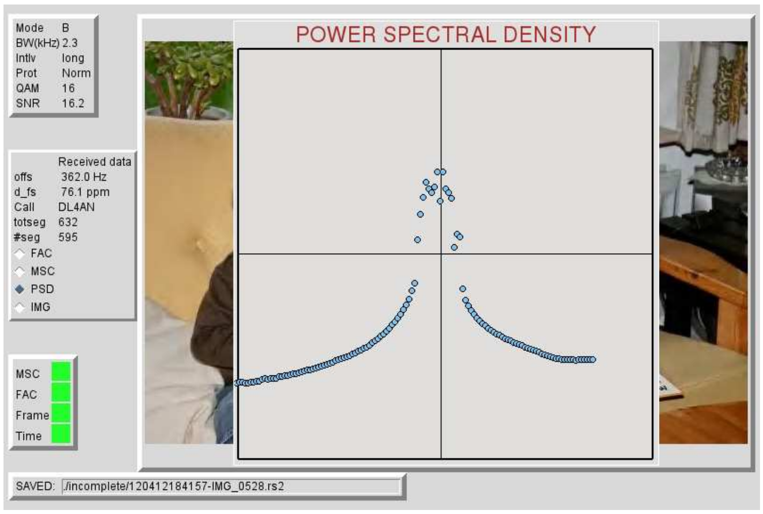

```
Mode     ?
BW(kHz) ??
Intlv    ???
Prot     ???
QAM      ??
SNR      ??

         Received data
offs     ???
d_fs     ??
Call     ?????
totseg   ??
#seg     ??
◇ FAC
◇ MSC
◆ PSD
◇ IMG

MSC
FAC
Frame
Time

SAVED:
```

Figure 3: PSD - no signal - volume OK

you tune your receiver to one of the standard digital SSTV frequencies and start **drmtst** in standalone mode and leave it on, the subdirectory **./pics** fills up with the received files which you can then inspect with some external viewing program or even publish on your website automatically. From version v0_2 **rxamadrm** also stores files when segments are missing due to qrm or qsb. If the number of missing segments is not too big and the file was reedsolomon coded the original file can be recovered by one of the programs **rs1decode, rs2decode, rs3decode or rs4decode**. These programs and their source code are in the root directory of the **rxamadrm** distribution. The files with missing segments are stored in the subdirectory **./incomplete**.

Generally it will be more fun to follow the progress of the reception of a picture, see the actual signal to noise ratio (SNR), have some aid in tuning, view the received image as soon as it is available, etc. In RXAMADRM these user interface functions are provided by **rxamadrm.tcl**. This script interprets the text messages printed by **drmtst** and displays the pertinent data. Some data like the MSC and FAC constellation are presented in the form of a graph. These graphs are shown one at a time. You can choose which one is displayed by activating one of the radio buttons.

Accurate tuning can be accomplished by watching the PSD graph, the DC-tuning offset (should be around 350 Hz) and the SNR (should be maximized). The PSD graph is also useful for setting the volume of the soundcard input. With receiver noise only, adjust the volume to give the top of the peak somewhat above the horizontal crossbar (figure 3). When tuning a drm signal adjust for 350 Hz offset and



```
Mode      B
BW(kHz)   2.3
Intlv     long
Prot      Norm
QAM       16
SNR       16.2

          Received data
offs      362.0 Hz
d_fs      76.1 ppm
Call      DL4AN
totseg    632
#seg      595

MSC    🟩
FAC    🟩
Frame  🟩
Time   🟩

SAVED: ./incomplete/1204121841557-IMG_0528.rs2
```

Figure 4: PSD - signal tuned OK

a sharp PSD peak. When the drm signal is OK the PSD peak will be sharp pointed with no more than 3 - 4 dots left and right of the vertical center line (figure 4). If side humps appear at the bottom of the PSD peak, the volume should be lowered (figure 5).

Rxamadrm.tcl uses some features that are not present in the standard Tk/Tcl distribution. You will also need the packages **Expect** and **Img**. These can be downloaded from the internet. **Expect** is used in the decoding of the printed output of **drmtst** while **Img** allows the direct viewing of other image format files besides the standard ones in Tk/Tcl (only gif and bitmap).

Installation

To install **rxamadrm** untar and unzip the **rxamadrmv0_4.tgz** archive in your home directory. This process will create a subdirectory called **rxamadrmv0_4** where the sources as well as the executables can be found. To be able to run the program the alsa soundsystem should be installed. In most linux distributions this will be provided. Furthermore the following dynamic link libraries will be needed:

- libasound
- libz
- libfftw (version 2.0.5)

For the graphical user interface Tk/Tcl is needed and thus should be installed on your linux system. The **rxamadrm.tcl** script calls the

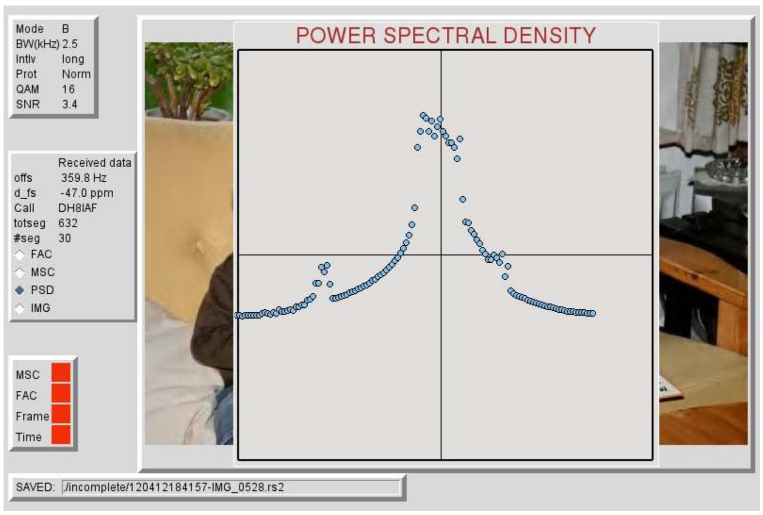

```
Mode     B
BW(kHz) 2.5
Intlv    long
Prot     Norm
QAM      16
SNR      3.4

         Received data
offs     359.8 Hz
d_fs     -47.0 ppm
Call     DH8IAF
totseg   632
#seg     30
◇ FAC
◇ MSC
◆ PSD
◇ IMG

MSC   🟥
FAC   🟥
Frame 🟥
Time  🟥

SAVED: ./incomplete/120412184157-IMG_0528.rs2
```

Figure 5: PSD - signal too loud

wish-interpreter of Tk/Tcl. The name and place of this interpreter depends on the version of Tk/Tcl. Try to find it (by the command **which wish**) and change the first line of the script **rxamadrm.tcl** if **/usr/local/bin/wish8.5** is not OK.

To be able to show jp2-pictures in the GUI, rxamadrm needs the program **jasper**. Most linux distributions have it, if yours does not have it, install it.

If you want to compile rxamadrm yourself you will also need the development packages for these libraries. A Makefile for this compilation is included in the source directory.

For OpenSuse use libdfftw and change the Makefile accordingly. Moreover in the files channeldecode.c, demodulate.c, drm.h, getofdm.c, getofdmsync.c the sourcecode should be changed to include the file **dfftw.h** and not the file **fftw.h**.

Before starting the program check the contents of the file **rxamadrm.ini**, look for the line:

```
devin=0
```

and change the number to the number of the sounddevice that you are going to connect your radio.

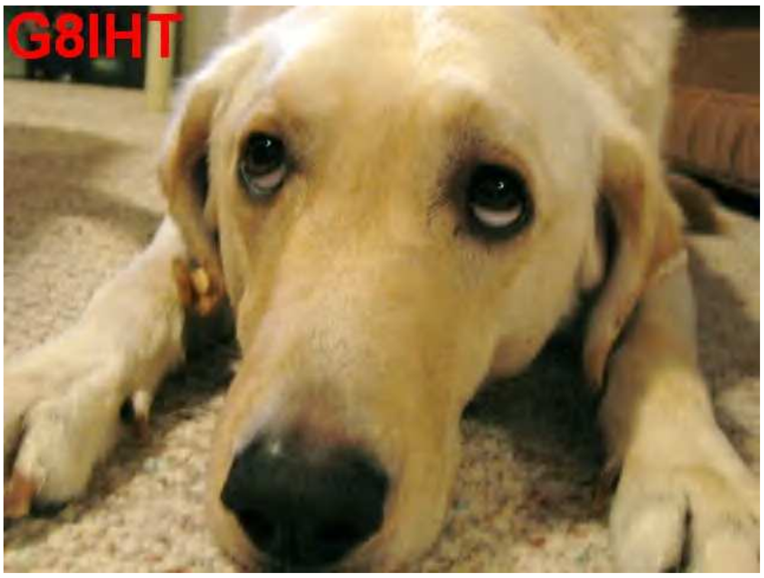


Figure 6: Picture sent by G8ITH

RESULTS

Computer to computer testing

During the development of RXAMADRM the audio signal generated by DIGTRX running on a second computer was used. The audio line output of the soundcard in the computer running DIGTRX was connected to the microphone input of the Linux system running RXAMADRM. All DRM-modes A,B and E were checked with both settings of interleave (long or short), with both settings of protection (Normal or Low) and all possible QAM-modes. SNR was over 38 dB in this set up. Transmitted files were jpg's. Under all circumstances the reception was without missing segments. A screenshot of the reception of a 50 kB file in mode A with QAM-64 is given in figure 1.

Testing on the 80 meter band

Figure 6 is the first picture I received using RXAMADRM on the band at 10 dB SNR. It was a Reed Solomon coded file 090301202608-Clip.rs2 of 14025 bytes from the station G8ITH. RXAMADRM cannot view this kind of files directly, I had to decode and view it with external programs.

Figures 7 and 8 show some more received pictures. I have no data on SNR because they were received in the standalone mode of **drmtst**. I just found them in the **./pics** directory after I left the program on

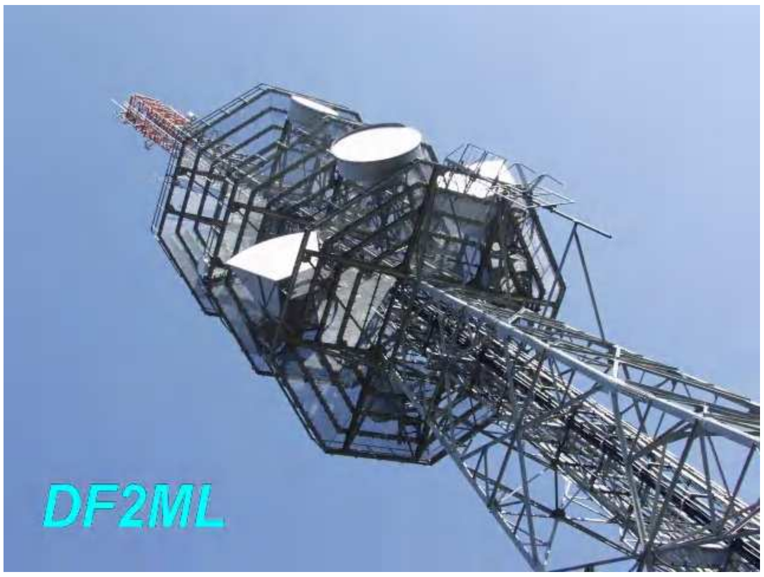


Figure 7: Picture sent by DF2L on 80 m on March 4th 2009

for some time.

DISCUSSION

The main reason to develop RXAMADRM was to provide the Linux community of hams with a good starting point to develop their own drm-based digital SSTV software. Linux offers a large variety of GUI-builders of which Tk/Tcl is just one. With toolkits like Gtk+ it is possible to build very sophisticated screens with attractive readouts, display of images and controls. Especially when these toolkits are combined with scripting languages they make it easy to automate many tasks like logging, web publishing, generating e-QSL's etc. RXAMADRM offers all required real-time data-acquisition and processing to convert the lf audio output of a receiver tuned to a ham DRM SSTV station to a diskfile of the transmitted image that is an exact copy of the one the sending station used.

The **rxamadrm.tcl** script shows an example of how this functionality can be "dressed" with a graphical user interface that can be customized to one's personal liking. It also allows to supplement the system with convenience functions like logging, archiving of pictures, etc. As of version 0_4 **rxamadrm.tcl** incorporates the decoding of Reed-Solomon coded pictures (-rs1, -rs2, -rs3 and -rs4). Moreover jp2-picture files are converted to -jpg files to show them in the GUI.

Figure 8: Picture sent by DF2ML on 80 m on March 5th 2009

Update June 29th 2011

With the installation of a new version of Linux (SuSE 11.2) on a new multicore processor computer some slight changes were necessary to compile and successfully run **rxamadrm**. First of all it was necessary to use the new **libdfftw** libraries. This necessitated the change of the line #include <fftw.h> into #include <dfftw.h> in all files were this line is present. Also the Makefile was updated to have the -ldfftw instead of the -lfftw compile option.

A major change has been introduced in the realtime data-acquisition. It now uses a **mutex semaphore** to prevent corruption of the sound buffer pointers, a measure that turned out to be necessary in the new multiprocessor environment.

The name of the new version is **rxamdrmv0_2**.

Update November 26th 2011

With the now popular use of .rs2 files with some redundancy in the data sent, it is possible to construct a good picture even when some segments are missed during reception. Rxamadrmv0_2 and v0_1 only saved a received picture when it was perfectly received. To be able to exploit the benefits of the Reed/Solomon encoding, files containing the incomplete rs-coded data are needed. This new version **rxamadrmv0_3exp** provides these files with incomplete data in the subdirectory **./incomplete** of the main directory with the executables. It has only be tested until now on a 64 bits Ubuntu 11.04 natty system.

To reconstruct the data from the incomplete files you need the external program **rsdecoder** that was available with **windrm**. Currently work is in progress to provide this Reed-Solomon decoding on the fly in **rxamadrm**.

Update January 2nd, 2012

In the v0_3 release reedsolomon decoding (only rs2-format) has been added. Its executable is called **rs2decode** and it is located in the root directory **rxamadrmv0_3** when the archive is extracted. Its source code is in the same directory and by using **make** the excutable will be built alongside the executable of the main program **drmtst**. If **drmtst** is used standalone received files will fill up the subdirectories **./incomplete** or **./pics**. The **.rs2** files in these subdirectories can be converted to their original counterparts manually by running **rs2decode** from its own directory, i.e.:

```
./rs2decode ./pics/mypicture.rs2
```

This can also be tried with **.rs2**-files found in the **./incomplete** subdirectory. Success will then depend on whether enough good segments will have been received to allow all errors to be corrected.

Together with the incorporation of the reedsolomon decoding **jp2**-decoding has now been added to **rxamadrm.tcl**. It uses an exteral picture format conversion program called **jasper**, which should be installed on your linux system. If it is not available **rxamadrm.tcl** will crash.

Update March 15th 2012

Thanks to very useful comments from David KI6ZHD a number of bugs could be removed from and some useful additions could be added to **rxamadrm**. Version 0_4 has automatic Reed-Solomon decoding for -rs1, -rs2, -rs3 and -rs4 coded pictures. It converts -jp2 pictures to the jpg-format before showing them in the GUI and the documentation was updated and corrected. If your have more than one soundcard in your system, the one that you are going to connect your radio to can be set in the file **rxamadrm.ini** which is located in the main directory of rxamadrm.

References

[1] Cesco HB9TLK, http://www.qslnet.de/member/hb9tlk/.

[2] http://n1su.com/windrm.

[3] VK2CZU, http://www.users.on.net/ trevorb/.

[4] http://www.tima.com/ djones/hamdrm.htm.

[5] Cesco HB9TLK, http://www.qslnet.de/member/hb9tlk/drm_h.html.

[6] T. Schorr A.Dittrich, http://nt.eit.uni-kl.de/forschung/diorama.

[7] T.Schorr A.Dittrich W. Sauer-Greff R.Urbansky, http://www.fh-kl.de/ drm/dokumente/sonstige/ieee_sp2005_diorama.pdf.

[8] R. Dean Straw, editor. *The ARRL Handbook For Radio Communications.* ARRL, 2006.