VGASIG
FM radio transmitter using a VGA graphics card

Bartek Kania <mrbk@gnarf.org>

April 19, 2009

Contents

1 What is it?

A collection of programs to generate FM radio signals in software and then transmit them using a VGA graphics card.

Basically a software radio experiment without having to spend lots of money on radio hardware.

2 Warning!

These programs and this document are for educational purposes only!
The author takes no responsibility whatsoever for any problems caused by use of these programs.

These programs implement a low power radio transmitter. All countries have regulations regarding use of radio transmitters. It may very well be illegal to use these programs where you live. Make sure to check the laws in your country before running any of these programs.

Although modern computers and monitors should have no problem dealing with the signals generated or the very unusual X configurations used there might still be a chance that something breaks. You use these programs at your own risk.

This is not for everyone. You need a good understanding of C programming, the X window system and a basic understanding of electronics for these programs to be of any use to you.

3 Requirements

- A fast modern computer running Linux or BSD or somesuch. Preferably a dualcore or quadcore.
- A graphics card that has an analog VGA port. The card needs to be very configurable. Not all cards work, and some cards may require patched drivers. See section 4 for more info.
- The `sox` program.[1]
- The SDL media library.[2]
- An antenna.[3]

4 The graphics card

The VGA card is the main component. The card is responsible for converting the digital representation into an analog signal.

VGA cards have high speed DACs[4] that are used to generate the video signal sent to the monitor. These DACs basically convert the image into an analog signal line by line that is fed into the monitor. This allows us to create a radio signal, display it as an image and have the card convert it to a radio signal for us.

[1] sox is used to preprocess the audio before it gets to the modulator.
[2] SDL is used to output the signal to the X server.
[3] A straight wire about 0.7m in length will do fine.
[4] Digital to Analog Converters.

This all sounds good in theory, however reality isn't quite as cooperative as one would wish and there are some obstacles to overcome.

4.1 The sync-signal problem

Unfortunately the signal that a monitor needs not only contains the actual image data but also contains things like horizontal and vertical sync pulses and H/V blanking+sync intervals during which the DACs are idle. These things cause trouble for us that want to abuse a VGA card for radio transmission.

It is possible to minimize the distortions caused by these signals by making the sync intervals as small as possible. Some cards may even let you turn them off completely. Unfortunately I haven't found any card that lets you do this yet.

To minimize the sync intervals you will need to configure custom modelines for your X server. This also has the side effect that you will not be able have a monitor connected to see what you are doing.[5]

4.2 The dot-clock problem

The next problem to deal with is the dot clock frequency of the VGA card.

Some chips let you set this to any frequency you want, and in those cases there is no problem! Just set it to 73MHz and use the first harmonic to get to the FM band. Or if your computer is fast enough, try setting it to 220MHz and set the programs to directly synthesize the correct FM carrier.

Some chips can only set their pixel clock in certain increments. For these chips you will need to find a dot clock between 73MHz and 87MHz that the card accepts and then use the first harmonic to transmit on the FM band.

4.3 Using harmonics

DACs generate a lot of harmonics.[6] We can use this property to get our signal up to the FM band if we are unable to directly generate a FM carrier. The signal will be less strong but it will still be receivable within a few meters of your computer.

In general, if f_d is our dot-clock frequency and f_c is the carrier frequency then the harmonics will appear at $nf_d + f_c$ for all integer values of n.

So, if you have a dot-clock frequency of 73MHz and a carrier frequency of 35MHz then the first harmonic will end up at 73 + 35 = 108MHz which is right at the top of the FM band.

Just remember that the dot-clock frequency meeds to be at least twice that of the carrier due to the sampling theorem.

4.4 Getting the outputs enabled without a connected monitor

All VGA cards I tested have the annoying habit of testing if the monitor is connected before they decide to send a signal out the VGA port. This is a big problem when using the VGA port for radio since the

[5] This is not 100% true... If you have a card with multiple outputs it might let you reconfigure only one output and keep the other connected to your monitor. My laptop with a Intel GM965 chip lets me do this.

[6] If you don't know what harmonics are or how much harmonics DACs generate then I recommend that you stop here, and go ask google a question or two.

card won't generate a signal.

The Intel card in my laptop lets me activate the VGA port using the `xrandr` command, and will hapily send a signal out of it as long as my laptop monitor is active. Fortunately it also lets me set the VGA port parameters independently from the LCD parameters so this works out fine.

Another Intel card I had in my workstation wasn't so kind. The driver even refused to let X start unless I had a monitor connected. In the end I was forced to patch the driver so it always thinks the monitor is connected.

4.5 Usable VGA cards

From my own tests and from what I have read the following cards should do the job:

Intel GM965 for laptops This card works great! You just need to configure the VGA port correctly using `xrandr` and have your laptop monitor active.

Intel 82G33/G31 integrated chip This card works great after the X driver has been patched to think that the monitor is always connected.

ATI Radeon 9200SE Haven't tested it myself, but I hear that it will do the job.

4.6 Configuring your VGA port

The first thing you need to do is decide which resolution you are going to use. If you are using the on-board VGA card in an laptop, or a VGA card that has two ports and a monitor connected to one of the ports then you should use the same resolution as you are using on your monitor. If you are using a separate VGA card for radio, then you can choose any resolution you like, but you should probably choose the highest resolution possible.

Next, you need to select a dot-clock frequency.

Now you are ready to configure the VGA port. You should try to configure it to use the minimal possible sync intervals. Below are two examples of how this can be done.

Intel GM965 laptop chipset

On my laptop I use 1152x864 as my screen resolution. I've selected 73MHz as my dot-clock for this example. Configuring the VGA port is done using the commands below:

```
xrandr --newmode RadioDAC 73 1152 1152 1152 1153 864 864 864 865
xrandr --addmode VGA RadioDAC
xrandr --output VGA --mode RadioDAC
```

The first command sets up a new modeline[7], the first parameter is the dot-clock, the second is the width of the display, third is the start of the horizontal sync, fourth is the end of the sync and the fifth parameter is the total width including sync pulse. The next four parameters are the same thing but for the vertical dimension.

[7] If you don't know how X11 modelines work then you need to read some manuals first

The HTOTAL value in the above example is 1153, which is the lowest that I could set on my card and still get a signal out of it. This causes us to loose about a pixel to the horizontal sync, which isn't that bad.

The VTOTAL value is also just one more than the height of the display. Unfortunately this causes us to loose an entire scanline and will certainly cause some signal distortion. I have not found any way around this.

Intel 82G33/G31 integrated pci-e chip

Since I have no monitor connected to this card I use the maximum resolution of 2048x2048. I use a 73MHz dotclock here aswell.

The commands for this configuration are the same as the previous, with the only difference being the larger resolution.

```
xrandr --newmode RadioDAC 73 2048 2048 2048 2049 2048 2048 2048 2049
xrandr --addmode VGA RadioDAC
xrandr --output VGA --mode RadioDAC
```

5 How the modulators work

The FM broadcasts, as the name implies, use frequency modulation. If you don't understand how frequency modulation works then you might want to check out [5] before continuing.

FM radio transmitters modulate a carrier in the frequency range of 87.5–108MHz with a baseband audio signal that has a 15kHz bandwidth. The frequency shift is 75kHz, so the transmitted signal frequency varies between 75kHz below the carrier frequency to 75kHz above the carrier frequency.

For stereo FM transmission the modulation itself remains the same, but the baseband signal is no longer a simple audio signal. It is instead a composite of the 15kHz mono audio (Left + Right channel), a 19kHz pilot signal and a 15kHz difference signal (Left - Right channel) modulated using DSB-SC on a 38kHz subcarrier. This composite signal is then frequency modulated as for the mono case. The result is that the signal is fully compatible with old mono receivers.

For more information on FM broadcasts see [6]

5.1 The mono modulator

The mono FM modulator is a straight-forward implementation of basic fm modulation. An overview of the structure is in figure 1.

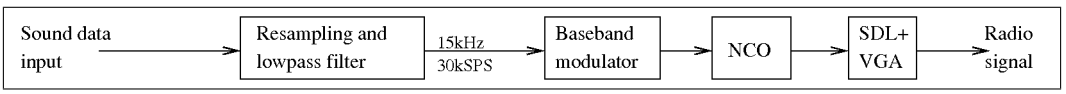

Figure 1: Mono modulator structure

Resampler and lowpass filter

FM broadcasts use a 15kHz audo bandwith so we first need to low-pass filter the audio to 15kHz and resample it to 30kSPS[8] which is the lowest possible sample rate needed to represent 15kHz.

This processing is done using the `sox` program in a separate thread so we don't have to do it ourselves.

Baseband modulator

The 30kSPS signal is then processed by the baseband modulator, whose sole function is to convert each sample to the frequency used to represent it in the radio signal.

The modulator first converts the sample from signed 16bit integers to a real valued sample in the range [-1,1] and then calculates the frequency used to represent this sample in the radio signal according to the formula below:

$$F_t = m_t * f_\Delta + f_c \tag{1}$$

Where F_t is the radio frequency, m_t is the sample value, f_Δ is 75kHz[9] and f_c is the carrier frequency.

The reason this is done at this point is because of the lower sampling rate of the baseband signal, this leaves more processor power over when the real radio signal is generated.

NCO

This is just a Numerically Controlled Oscillator[10] that generates a radio signal of a given frequency. The previous sentence is just a fancy way to describe a sine wave generator.

In these particular programs the NCO also implements a very simple interpolation filter and upsampler so it can accept a 30kHz baseband signal to modulate a carrier at a significantly higher sample rate.

This oscillator operates at the dot-clock frequency, and must be very fast for the CPU to be able to cope.

The output of this block are the samples of the radio signal, which in this particular case are the pixel lightness values in the "image".

SDL+VGA

In this last block the signal is simply rendered as an image into a SDL surface and "displayed" using the VGA card. The samples (or pixel values) have already been generated by the NCO.

5.2 The stereo modulator

The stereo modulator is very similar in structure to the mono modulator. The main difference is the addition of a stereo encoder and pilot signal generator. Figure 2 shows the structure of this modulator.

Resampler and lowpass filter

As in the mono modulator the signal needs to be filtered to 15kHz and resampled. In the stereo modulator this is done for both the right and left channels.

[8] SPS = Samples Per Second.
[9] The broadcast FM frequency shift
[10] Also known as DDS which stands for Direct Digital Synthesizer

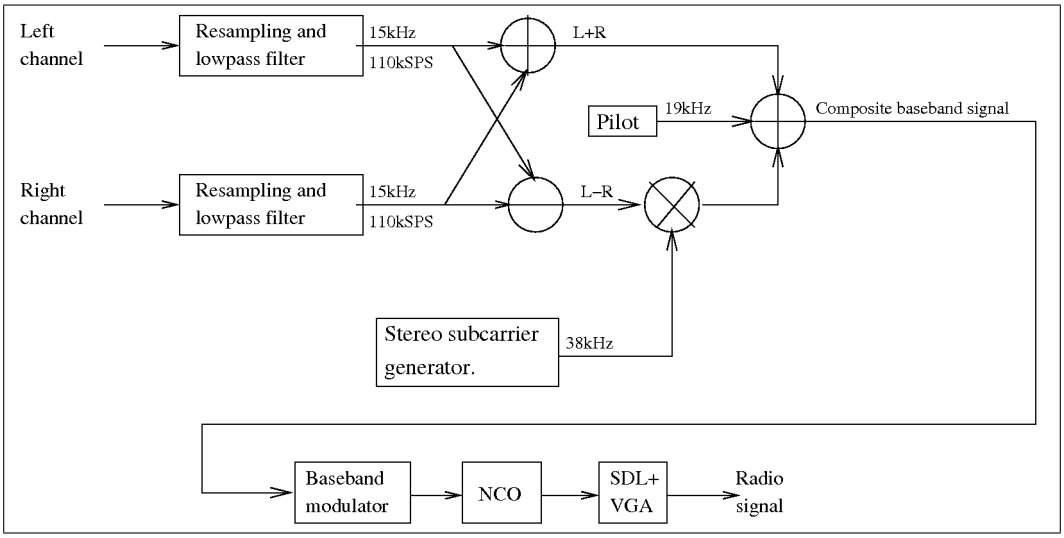

Figure 2: Stereo modulator structure

For stereo modulation we need a higher baseband sampling rate to accomodate the stereo pilot and subcarrier, so the signal is resampled to 110kHz instead of 30kHz.

Like in the mono modulator `sox` is used for this block.

Sum and Difference signals

For stereo modulation to be compatible with mono receivers the modulator actually creates a mono signal by adding the left and right channels. This is the L+R signal in the figure.

The modulator also creates a difference signal that the receiver will use to reconstruct the left and right channel. This is the L-R signal in the figure.

Pilot generator

A 19kHz pilot signal is used to indicate to the receiver that this radio signal contains stereo information. The 19kHz signal is also phase-locked to the 38kHz stereo subcarrier to aid the receiver in correctly demodulating the difference signal.

Stereo subcarrier generator

This is a 38kHz NCO/DDS that is phase locked to the 19kHz pilot signal. All this “phase-locked” stuff might sound complicated, but it basically just means that both oscillators were started at the same time and run in parallell.

Stereo subcarrier modulation

The 38kHz carrier is modulated using DSB-SC[11] with the difference signal. DSB-SC modulation is quite simple and can be described with the following formula:

$$s_t = m_t * c_t \tag{2}$$

Where s_t is the output sample of the modulator, m_t is a sample of the *L-R* difference signal and c_t is a sample of the 38kHz subcarrier.

Signal combination

To create the composite baseband signal the three signals described above are simply added together. The pilot signal is multiplied by 0.1 since the pilot signal level should only be 10% of the signal power.

The resulting sample is then scaled by a constant so that it is within [-1,1] before being sent to the baseband modulator.

Baseband modulator

The base band modulator does exactly the same as it did in the mono modulator. The only difference is that the baseband signal has a higher sample rate.

NCO+SDL+VGA

These are identical to the ones in the mono modulator.

6 The programs

6.1 sig1

This program was the first attempt at generating an FM signal. The only thing it does is generate a .pgm image of an FM mono signal modulated with a tone of a given frequency.

No parameters are accepted by this program. To change the parameters of the generated signal the `#define`s at the beginning of sig1.c must be changed and the program recompiled.

If you want to understand how these programs work then this is the obvious place to start.

6.2 monofm

The `monofm` program implements the mono modulator described in section 5.1.

The program accepts the following parameters:

-r Dot-Clock frequency.
-c Carrier frequency.
-w Width of display.

[11] Double SideBand Suppressed Carrier

-W Optional HTOTAL value for VGA port. Defaults to width+1.
-h Height of display.
-H Optional VTOTAL value for VGA port. Defaults to height+1.
-f Filename of audio file. Defaults to "tst.mp3"

An example invocation:

```
./monofm -r 73000000 -c 35000000 -w2048 -h2048 -f tst.mp3
```

6.3 stereofm

`stereofm` implements the stereo modulator described in section 5.2.

It accepts the same parameters as `monofm`.

An example invocation:

```
./stereofm -r 73000000 -c 35000000 -w2048 -h2048 -f tst.mp3
```

7 The sourcecode

This section will give you a short walk-through of the source code for the `monofm` and `stereofm` programs. Use the modulator description in section 5 as a functional reference.

The programs are written to be fast so they can be used in real time. Because of this the source is not always readable and is not really a good example of how to write well structured C programs. Please keep this in mind when checking out the source.

dds1.h and dds2.h

These two files contain a fast approximate DDS signal generator based on a lookup table with sine function values.

The dds in `dds1.h` uses a linear interpolation to approximate sample values that fall in between entries in the lookup table giving it slightly better precision compared to `dds2.h` that just returns the value closest to the phase of the signal. However the generator in `dds2.h` is faster.

fmdds.h

`fmdds.h` extends the generator in `dds2.h` by adding a slope factor that skews the frequency a bit after every sample. This is useful since the generator is fed with a baseband signal at a significantly lower sample rate. The slope acts as a very efficient and quite good low-pass interpolation filter.

fmmain.h

The main() function of the mono and stereo modulators is contained here. It only parses command line options and then starts the modulator.

monofm.c

The mono baseband modulator is contained in the function `baseband_mod`. It runs in it's own thread so as to take advantage of multicore CPUs.

The baseband modulator opens a pipe to `sox` and receives samples that it converts to a frequency and slope. These parameters are stored in two ring buffers `freqbuf` and `slopebuf` that are read by the NCO when generating the radio signal.

stereofm.c

This file contains the pilot signal generator, stereo encoder and baseband modulator all packed together in the `baseband_mod` function.

First a sample from both the left and right channels is read from the pipe to `sox`. Then the sum and difference signals are calculated and a composite sample is created. This sample is then converted into a frequency and slope which are saved in the ring buffers.

fm.h

This file contains the function `modulate_sample` which is the one that actually does the baseband modulation and stuffing of samples into ring buffers.

It also contains the function `fmsignal` which reads the ring buffers and creates the actual radio signal. The function loops over the pixels and lines of a VGA frame and calculates the sample value (pixel lightness) for each sample in the radio signal. It reads the frequency and slope information from the ring buffers when it needs it during signal generation.

When the frame is finished it waits the apropriate amount of time and then instructs the VGA card to display it. After that it starts generating the next frame.

This function runs in its own thread.

8 FAQ

Q: Why?!?

A: Why not?

Q: What distortions can I expect due to the sync intervals?

A: The horizontal sync intervals are so short, and so high in frequency that you shouldn't hear them. They will probably cause some HF distortion in the radio wave though.

The vertical sync intervals are a bigger problem. They will cause static or humming

with the same frequency as the framerate. Unfortunately this is usually very audible.

Q: Can you disable the sync intervals entirely?

A: I don't know. I haven't managed it yet. If you succeed please email me how you did it and what VGA card you used.

Q: Why are there no mutexes to avoid problems with the ring buffers?

A: Because they are not needed.

The thread writing data to the ring buffers is much faster than the thread that reads the data. So it only needs to fill the buffer up when the signal generator has consumed data, and then when the buffer is full it just waits a while for more data to be consumed.

References

[1] Lyons, Richard G. *"Understanding Digital Signal Processing"*

[2] Couch, Leon W. *"Digital and Analog Communication systems"*

[3] *Analog and Digital TV (DVB-T) Signal Generation*: http://bellard.org/dvbt/

[4] *Tempest for Eliza*: http://www.erikyyy.de/tempest/

[5] Wikipedia: *"Frequency Modulation"* http://en.wikipedia.org/wiki/Frequency_modulation

[6] Wikipedia: *"FM Broadcasting"* http://en.wikipedia.org/wiki/FM_broadcasting