Software Defined Radio in Max/MSP

Tutorial 1 – System Design, IQ Signals, Mixing

November 17th 2011

This set of four tutorials explores the process of building a software defined radio receiver in Max/MSP.

The patches work with any baseband IQ signal source, including hardware devices and IQ data files. Sample IQ data files are provided with the tutorials – No additional hardware is needed.

You will need either the runtime or the full version of Max/MSP from http://cycling74.com. The runtime version is free, but it doesn't allow editing. At this time Cycling74 provides a 30 day 'demo' period for the full version of Max.

The tutorials assume a basic understanding of software defined radio, receiver design, and experience with Max/MSP.

Tutorial 4 shows an example of device control for the Soft66LC2 and FUNCube. Drivers are provided for both devices in MacOS and Windows.

To access the tutorial patches - download and unzip the file from the tutorial link at http://zerokidz.com/radio. You can also get the tutorial files by downloading the entire Maxradio project from the same site.

Contents of tutorial folder:

Documents and patches:

maxradiotutorial1.pdf - Tutorial 1

minimal.maxpat – the smallest radio
crytstalSet.maxpat – modern crystal set
basicSDR1.maxpat – basic software defined radio

maxradiotutorial2.pdf - Tutorial 2

basicSDR2.maxpat – basic software defined radio plus bandscope, AGC, and AM detector

maxradiotutorial3.pdf - Tutorial 3

basicSDR3.maxpat – basic software defined radio plus bandscope, AGC, and AM, FM, and audio filters

maxradiotutorial4.pdf - Tutorial 4

basicSDR3.maxpat – basic software defined radio plus bandscope, AGC, and AM, FM, audio filters, and device control for Soft66lc2 & FUNCube.

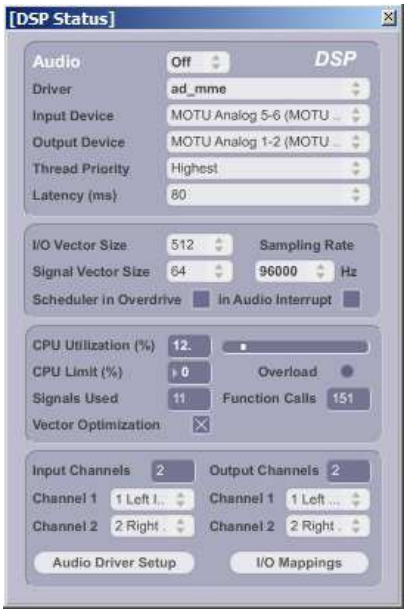
The other files in the tutorial folder include a README file, sample IQ data (.wav files), subpatches used by the tutorials, and library files required for the Soft66lc2

If you are planning to use the Soft66lc2 its worth checking out the README file.

Hardware setup

Each of the patches include a green object called [audioSettings]

If you are using the runtime version of Max, double-clicking this object allows you to configure your audio hardware. It also works in the full version of Max and is the same as selecting: Options | Audio Status (DSP Status in Max5) from the Max menu. When you double-click [audioSettings] the following setup screen appears:



Select the drivers, input, and output for your hardware.

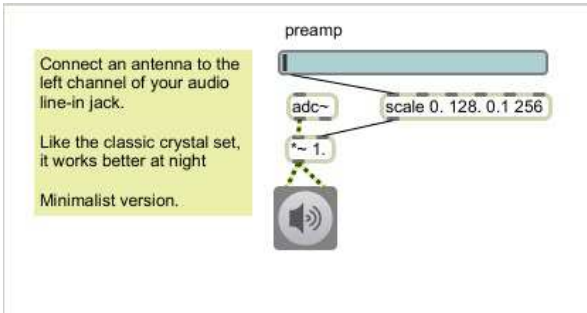
- If possible set your audio sample rate to 96 KHz or higher. It improves bandwidth and audio quality.
- If audio is breaking up try increasing the Signal Vector Size to 512.
- For other troubleshooting tips, check the README file.

When you close this window, your audio settings will be saved automatically.

The smallest radio

Signals are everywhere. In the 1965 TV series “Gilligans Island” an accidental blow to the head caused Gilligan's tooth fillings to act as a radio receiver.

Open the patch: **minimal.maxpat**

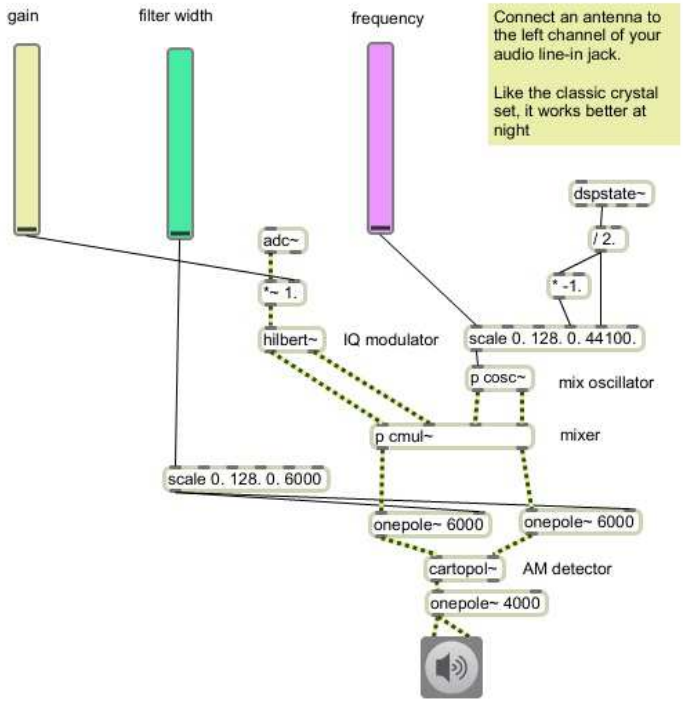


This patch passes audio from input to output with some amplification. If you hook a microphone to the input, you’ll have a PA system. If you connect a dipole antenna to the input you’ll have a radio. I have listened to Radio Havana with this patch. There’s no control over frequency or selectivity.

If you work with audio and live in an urban area you’ve probably had the experience of unwanted radio signals getting into your audio gear. The wiring becomes an inadvertent antenna. That’s really all that’s going on here.

If you have an antenna, connect it to your computer’s audio input jack. Select line-in as the audio source. It works best with shielded transmission (coaxial cable). Also try it on a winter night under a full moon when signal propagation is optimum for low frequencies.

You may also want to try the patch: **crystalSet.maxpat**, which simulates a classic crystal set radio along the same lines as the previous patch.

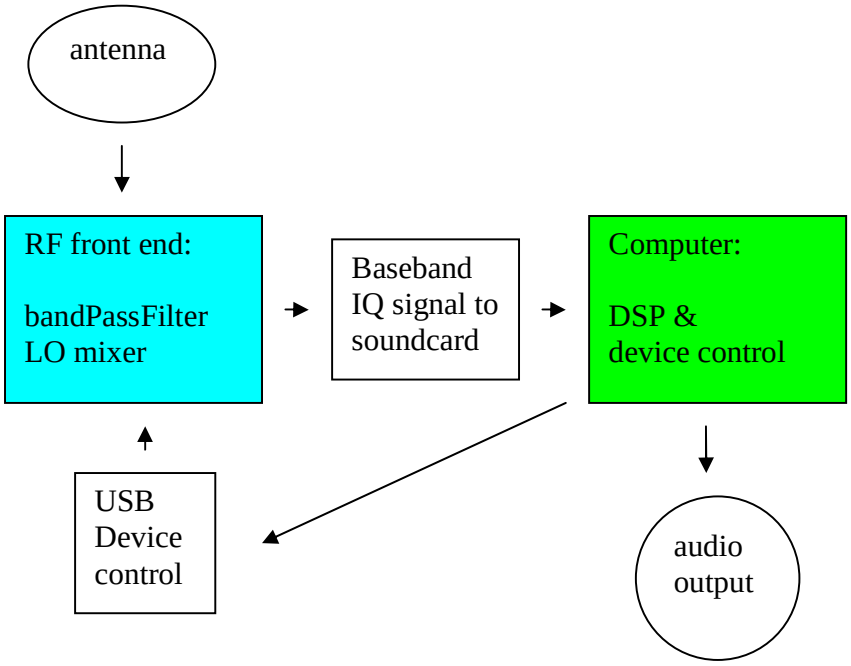


Note: please close any open patches before moving on to the next section.

System Design

In a software defined radio system (SDR), components traditionally built with hardware, are implemented in software using digital signal processing (DSP). Familiar products like cell phones and wireless routers are examples of SDR's. The project described here is a software defined radio receiver with three physical components:

- Antenna
- RF front-end
- computer



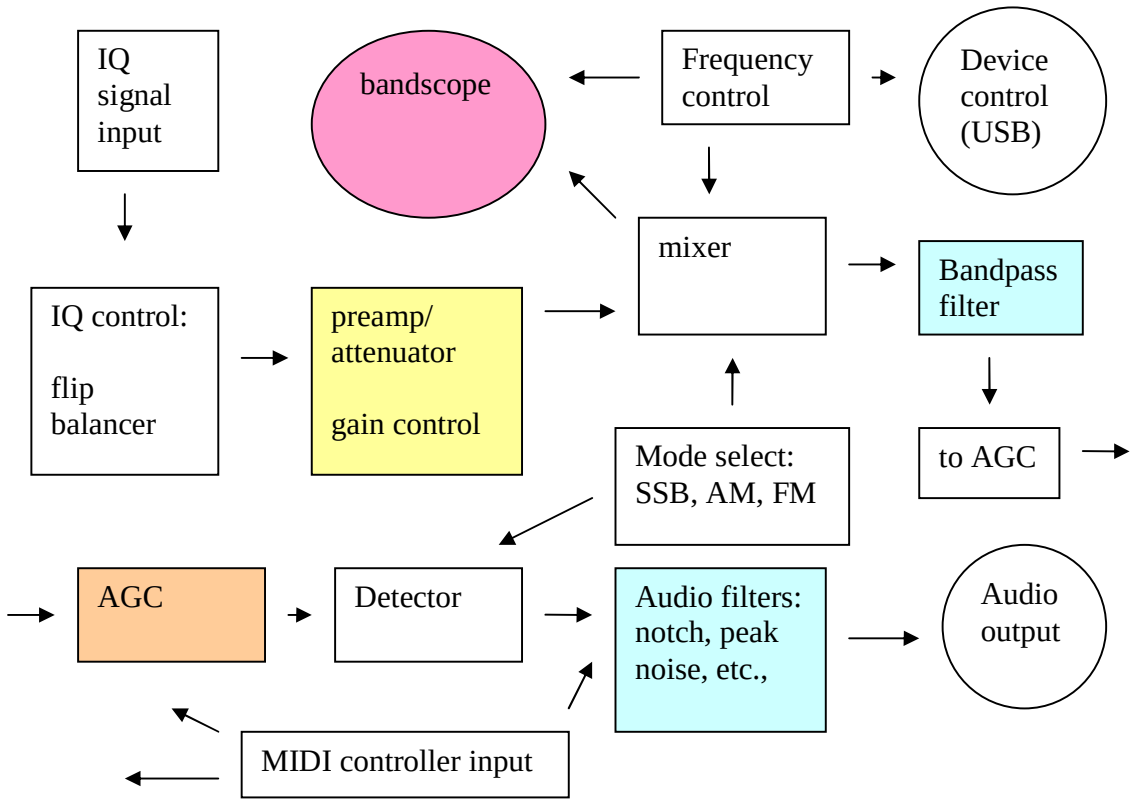
In a direct conversion SDR, the RF front end sends input signals through a bandpass filter to a mixer which shifts the frequency down into the audio spectrum. The mixer uses two local oscillator (LO) signals that are 90 degrees out of phase to produce “I” and “Q” signals (in-phase and quadrature). IQ signals are also referred to as complex or analytic signals. Using IQ signals simplifies frequency shifting and demodulation.

In these tutorials we’ll use baseband IQ data files pre-recorded from SDR devices.

The computer reads baseband IQ signals, via the sound card, and converts them to digital data (ADC). The rest of the processing happens in software. This includes mixing, filtering and demodulation. The final result is converted to an analog audio by the sound card (DAC).

Typically an SDR program also sets device frequency and selects appropriate hardware filters via a serial (USB) interface.

Software Signal Path

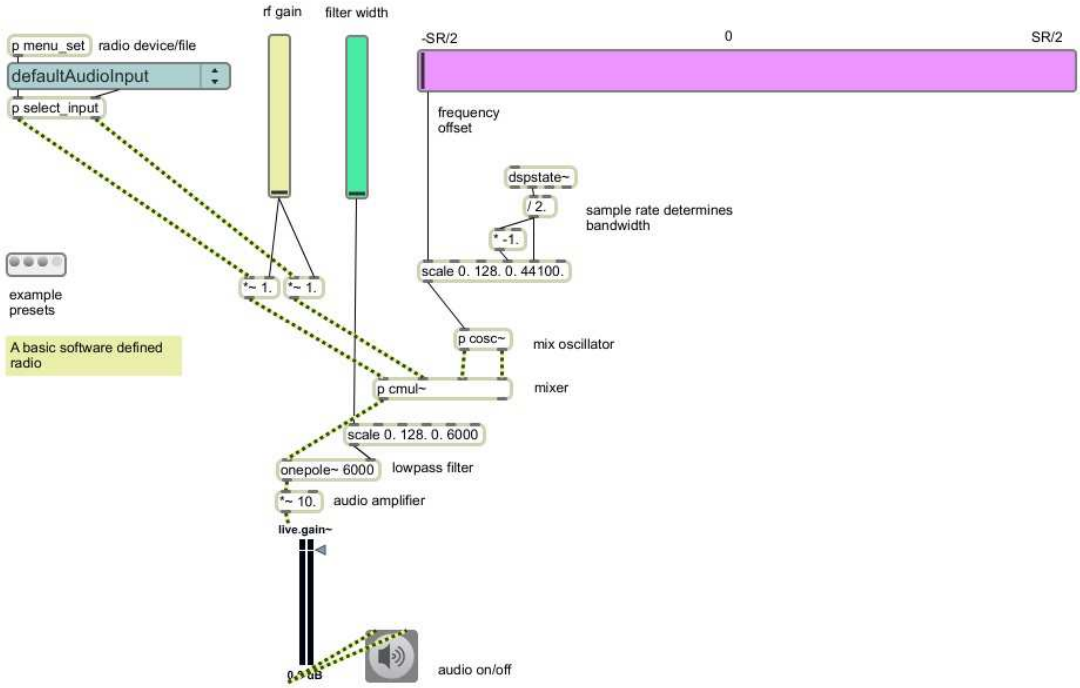


A Basic Radio

Sample IQ data files have been provided for this tutorial. Looping these files simulates signal input from a hardware device.

Note: If you have a hardware device capable of generating baseband IQ signals - and the software to control its frequency - it should work with the tutorial patches. Just connect it to the computer's audio input.

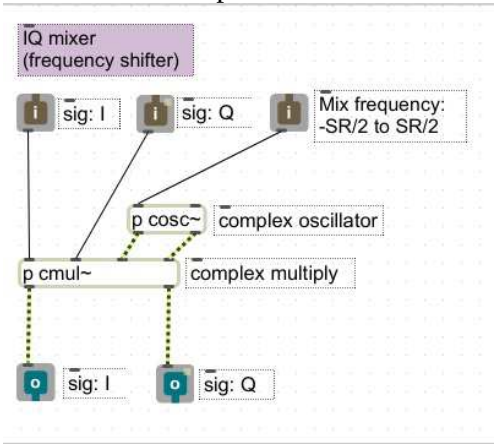
Open the patch: **basicSDR1.maxpat**



Start the audio by clicking the speaker icon [ezdac~] at the bottom of the patch. Then select one of the example preset buttons. (They're the miniature dots inside a gray rectangle). The example data (10meter96.wav) is from a 28 MHz amateur radio SSB contest. Try adjusting the frequency slider and filter width slider.

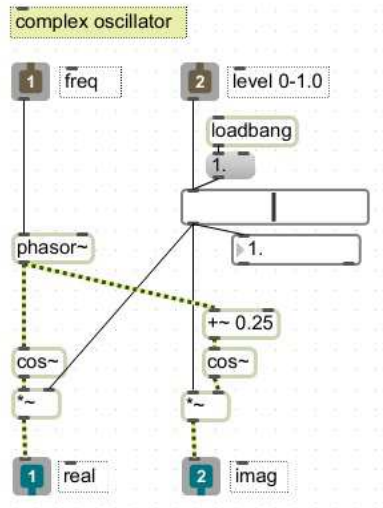
Let's follow the signal path starting in the upper left corner. The [umenu] selects an input source, in this case a looped IQ data file. Next the RF gain slider amplifies or attenuates incoming signals to a reasonable level.

The heart of this patch is the mixer.



The mixer shifts the center frequency of a baseband signal by complex multiplying it with output of a mix oscillator. The mix oscillator is a variable complex oscillator

[cosc~], controlled by the frequency slider. It produces a frequency ranging from –SR/2 to SR/2 (where SR is the current sample rate).



The mixer output frequency is the sum of the mix oscillator and the baseband signal. So essentially, the mixer acts like a tuning dial (within the passband). Sample rate determines passband width. With IQ signals sample rate is passband width. At this point, if you're new to the world of radio, I hope you have questions about negative frequencies and the apparent disregard for Nyquist.

A lowpass filter [onepole~] is next in the signal path. An SSB signal has a bandwidth less than 3 KHz. The filter reduces overall bandwidth down from the width of the entire passband (SR) to the actual width of the signal. This helps eliminate interference from adjacent channels.

The next stage is the detector (demodulator) but as you can see its not there. An interesting feature of IQ signals is that SSB is already demodulated. The left outlet of the [cmul~] object is the upper sideband. All that remains is to amplify signal level and output the audio.

What's Next?

In the next tutorial we'll add a preamp, a band-scope, AGC, and an AM detector. Please send comments and questions to radio@zerokidz.com