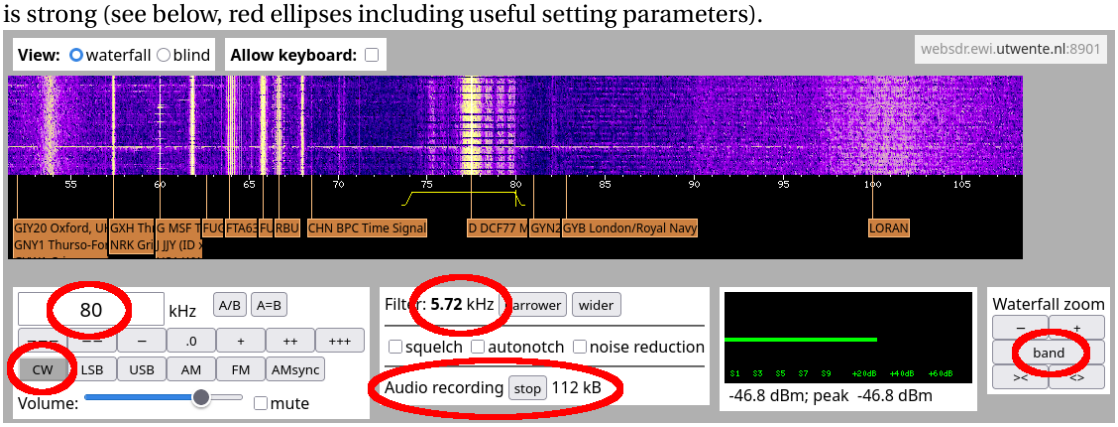
Digital Communication

J.-M Friedt, April 3, 2025

DCF77 is a German, Very Low Frequency (VLF) station broadcasting timing signals for synchronizing clocks accross Europe. The carrier frequency of DCF77 is 77.5 kHz. Although this frequency range is withing the sampling rate of high-quality sound cards sampling at 192 kHz, we shall benefit from signals collected remotely and streamed over the Internet, named WebSDR. One of the best known websdr is `http://websdr.ewi.utwente.nl:8901/` at the Dutch university of Twente in Enschede. Although the receiver location is 285 km from the emitter location in Mainflingen, the received signal is strong (see below, red ellipses including useful setting parameters).



Connect to `http://websdr.ewi.utwente.nl:8901/` (**not** accessible through an Eduroam connection [1]) and set the carrier frequency to 80 kHz, CW (Continuous Wave) mode, 5 kHz filter width (the waterfall chart can be adjusted by setting "band") and Start recording an audio signal ("Audio recording"). After collecting a few seconds worth of data, "stop" the recording and save a local copy of the WAV file.

1. The "`file`" unix command allows for getting information on the content of a file whose name is given as argument: what is the sampling rate and data format of the collected information? are they real or complex data?

We wish to analyze the structure of the signal and the various modulation schemes for transfering time over VLF. This processing can be completed on your favorite signal processing framework, whether GNU Octave, Numpy or GNU Radio or any other, whichever you are most comfortable with and will allow you to tackle most challenges. Under GNU Octave, the characteristics of an audio record saved as a WAV file is obtained with `audioinfo()` and reading the content of the file is achieved with `audioread()`. GNU Radio provides the `Wav File Source`. Python's `wave` module can be used as described at `https://docs.python.org/3/library/wave.html`.

2. The useful signal is off centered and must be brought to baseband for processing. What frequency offset do you expect, or can observe on a Fast Fourier Tranform of the recording?

3. Bring the signal to baseband, centered on 0-Hz. What additional spectral component are we left with and how can we get rid of this unwanted signal? What would have happened if we had sampled the signal at 6 ksamples/s instead of the observed sampling rate?

The initial modulation scheme implemented in 1959 was amplitude modulation to transmit the timing information.

[1] in case of failure to download a live dataset, a pre-recorded dataset is provided at `http://jmfriedt.free.fr/websdr_recording_start_2025-03-15T12_55_25Z_80.0kHz.wav`

4. display the amplitude of the signal brought to baseband: what is the time interval of the markers identified as a drop in the signal amplitude?

5. How accurately did you have to tune the centering of the carrier close to 0-Hz to achieve amplitude demodulation?

The amplitude modulation was complemented in 1988 by Hetzel[2] with a phase modulation carrying a 2^9-chip long pseudo-random sequence, with a small phase swing of $\pm 10°$ to minimize the impact on existing AM-demodulation receivers.

6. display the phase of the baseband signal. What striking characteristics do you observe that might prevent identifying the pseudo-random sequence?

7. How do you identify the coarse frequency correction which might allow for correcting this issue? What is the frequency resolution of the proposed method?

8. In case the phase correction from the previous method was insufficient, what fine frequency tuning strategy can we use to cancel any leftover frequency offset?

9. Why is fine frequency correction needed for phase demodulation?

10. How often does the pseudo-random sequence pattern repeat within the message? What mathematical tool did you use to reach this conclusion? How sensitive is this mathematical estimator to frequency offset?

11. The 512-chip long pseudo-random sequence is provided at `http://jmfriedt.free.fr/dcf77_lfsr.txt` in text (ASCII) format and `http://jmfriedt.free.fr/dcf77_lfsr.bin` in binary (1-byte/chip) format. We are told in Hetzel's paper that the chip duration is 120 carrier periods. What is the chiprate?

12. How does it compare with the sampling rate? How shall we process the samples read from the `dcf77_lfsr` file so we can compare them with the data collected in the WAV file? How do you find the parameters for this processing, and what function do you use (in whatever processing framework you selected)?

13. Find the copies of the pseudo-random sequence in the recorded signal: how often does it repeat? What mathematical tool did you use to achieve this result?

14. How accurately did you need to correct the frequency to achieve this result? Justify this conclusion.

15. Why do you think Hetzel added the phase modulation to the amplitude modulation? What benefit can you identify from the signals you processed?

16. Why did we off-center the receiver frequency when recording the DCF77 signal we analyzed? How is this kind of receiver architecture named?

[2] P. Hetzel, *Time dissemination via the LF transmitter DCF77 using a pseudo-random phase-shift keying of the carrier*, Proc. 2nd EFTF (1988) at `https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/pdf/5_1988_Hetzel_-_Proc_EFTF_88.pdf`

Answers

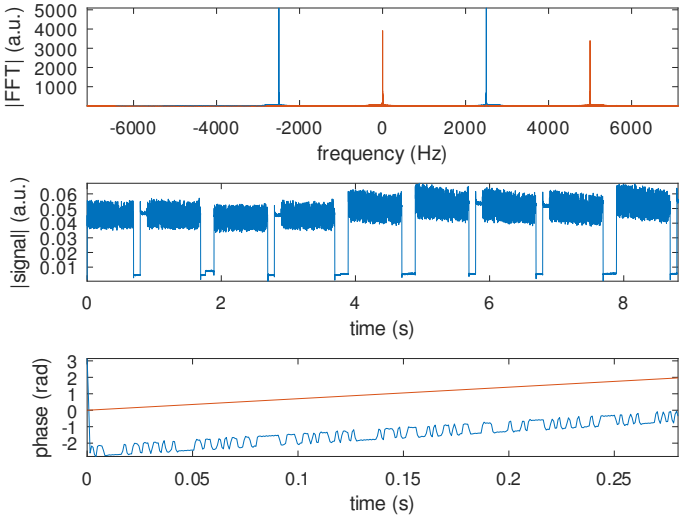
1. `file websdr.wav` tells `WAVE audio, Microsoft PCM, 16 bit, mono 14238 Hz` so the recording only includes real (no imaginary part) samples recorded at a sampling frequency of 14238 Hz.
2. We centered the local oscillator receiver and 80 kHz and DCF77 is broadcasting on 77.5 kHz, so we expect a 2.5 kHz offset, which is indeed observed on the FFT when correctly setting the X-axis from $-f_s/2$ to $+f_s/2$.
3. GNU Radio's `Frequency Xlating FIR Filter` brings the useful signal to baseband either by multiplying with a +2.5 or −2.5 kHz local oscallator since both signals are present in the spectrum of the real signal. Had we sampled at 6 kHz, the rejected image would have been reintroduced within the spectrum by aliasing and would have needed a narrower filter for removal.
4. the magnitude of the complex signal is representative of the AM modulation. The amplitude drop occurs at the beginning of each second.
5. AM demodulation only requires a coarse frequency offset compensation since AM is an incoherent demodulation scheme.
6. Despite correcting for the 2.5 kHz offset, a small frequency offset remains observed as linear phase drift.
7. The coarse frequency offset is identified with an FFT, or by fitting the phase as a function of time up to the first wrapping by 2π. The resolution is the bin-width of the FFT (f_s/N with N the number of samples in the FFT) or the inverse of the duration to the first phase wrapping.
8. Linear phase fitting.
9. The phase drift prevents the correlation with the pseudo random sequence.
10. The autocorrelation searches for the repeated pattern and is insensitive to the frequency offset since it occurs in both the reference and the delayed signal.
11. The chip rate is 77500/120 = 645.83 Hz
12. Since we decimated by 10 after frequency transposition and low pass filtering, the sampling rate has become 14238/10 = 1423.8 Hz, which is more than twice the chip rate. We must resample the pseudo random code so it contains as many samples per symbols than the signal: `rats(1423.8/(77500/120),6)` indicates 97/44 so we interpolate by 97 and decimate by 44 in the `Rational Resampler Block` of GNU Radio or `resample` of GNU Octave.
13. The cross-correlation peaks repeat every second, even though the result is not striking with GNU Radio.
14. The frequency offset must be corrected better than a fraction of the inverse of the duration of the code.
15. The bandwidth of the phase modulation is much broader than the AM signal bandwidth which only changes state once every second with a slow rise and fall time. Hence, the timing resolution is greatly improved with the addition of the phase modulation by Hetzel.
16. Had we centered the local oscillator on the signal carrier, we would have missed half of the information and not been able to recover a phase. The use of the intermediate frequency to create the complex signal close to baseband is called a superheterodyne architecture.

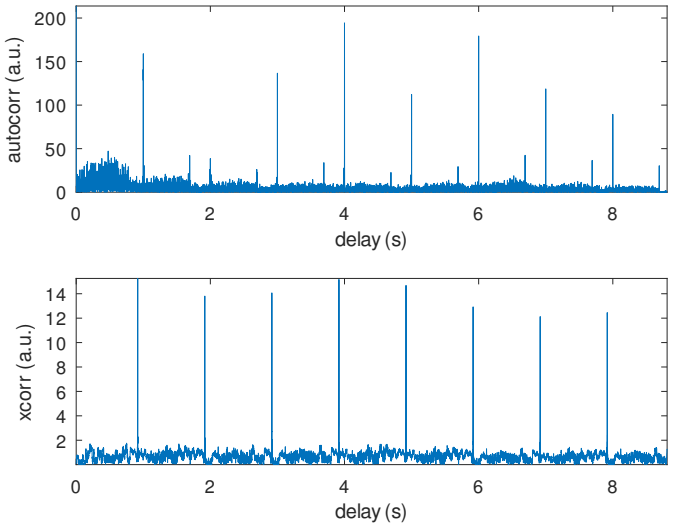
In GNU Octave, the processing script might be

```
close all

pkg load signal
d=dir('*wav');
d(1).name
a=audioinfo(d(1).name)
fs=a.SampleRate;
x=audioread(d(1).name);
freq=linspace(-fs/2,fs/2,length(x));  % graduer l'axe des frequences
subplot(311)
plot(freq,abs(fftshift(fft(x))))
t=[0:length(x)-1]'/fs;
nco=exp(j*2*pi*t*2500);               % transposition de la frequence nominale
y=x.*nco;                             % Xlating ...
hold on
plot(freq,abs(fftshift(fft(y))))
axis tight;xlabel('frequency␣(Hz)');ylabel('|FFT|␣(a.u.)');

subplot(312)
b=firls(32,[0 1000 1200 fs/2]*2/fs,[1 1 0 0]);
y=filter(b,1,y);                      % ... FIR filter and decimate
y=y(1:10:end);                        % decimate
t=t(1:10:end);
fs=fs/10
plot(t,abs(y))
axis tight;xlabel('time␣(s)');ylabel('|signal|␣(a.u.)');

subplot(313)
plot(t(1:400),angle(y)(1:400))     % recherche du residu (ou fft) : resolution
[a,b]=polyfit(t(1:400),angle(y)(1:400),1);          % de l'erreur en frequence
hold on
plot(t(1:400),a(1)*t(1:400));
axis tight;xlabel('time␣(s)');ylabel('phase␣(rad)');

figure
subplot(211)
y=y.*exp(-j*t*a(1));
[a,b]=polyfit(t,angle(y),1);
y=y.*exp(-j*t*a(1));
z=angle(y);z=z-mean(z);
plot(t,abs(xcorr(z,z)(length(t):end)))  % autocorrelation
axis tight;m=max(abs(xcorr(z,z)(length(t)+200:end)));ylim([0 m*1.1]);
xlabel('delay␣(s)');ylabel('autocorr␣(a.u.)');
fc=77500/120;
% resample code: "Each chip spans 120 cycles of the carrier"
% @ https://en.wikipedia.org/wiki/DCF77 = 645.83 Hz
% rats(1423.8/(77500/120),6)=97/44

subplot(212)
load lfsr.dat
lfsr=2*lfsr-1;
lfsr=resample(lfsr,97,44);
```

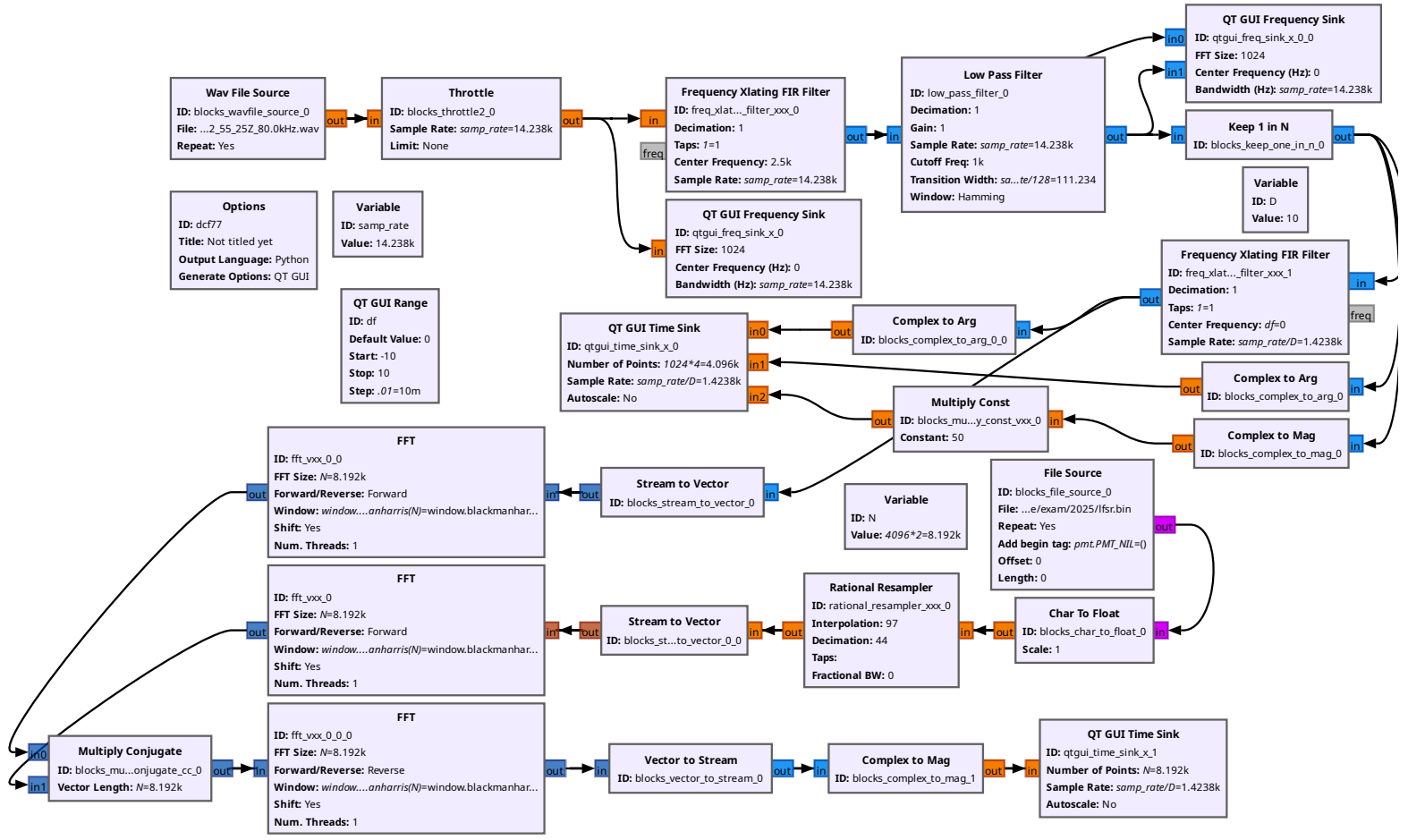
```
53 plot(t,abs(xcorr(lfsr,y))(1:length(y)))
54 axis tight;xlabel('delay␣(s)');ylabel('xcorr␣(a.u.)');
```

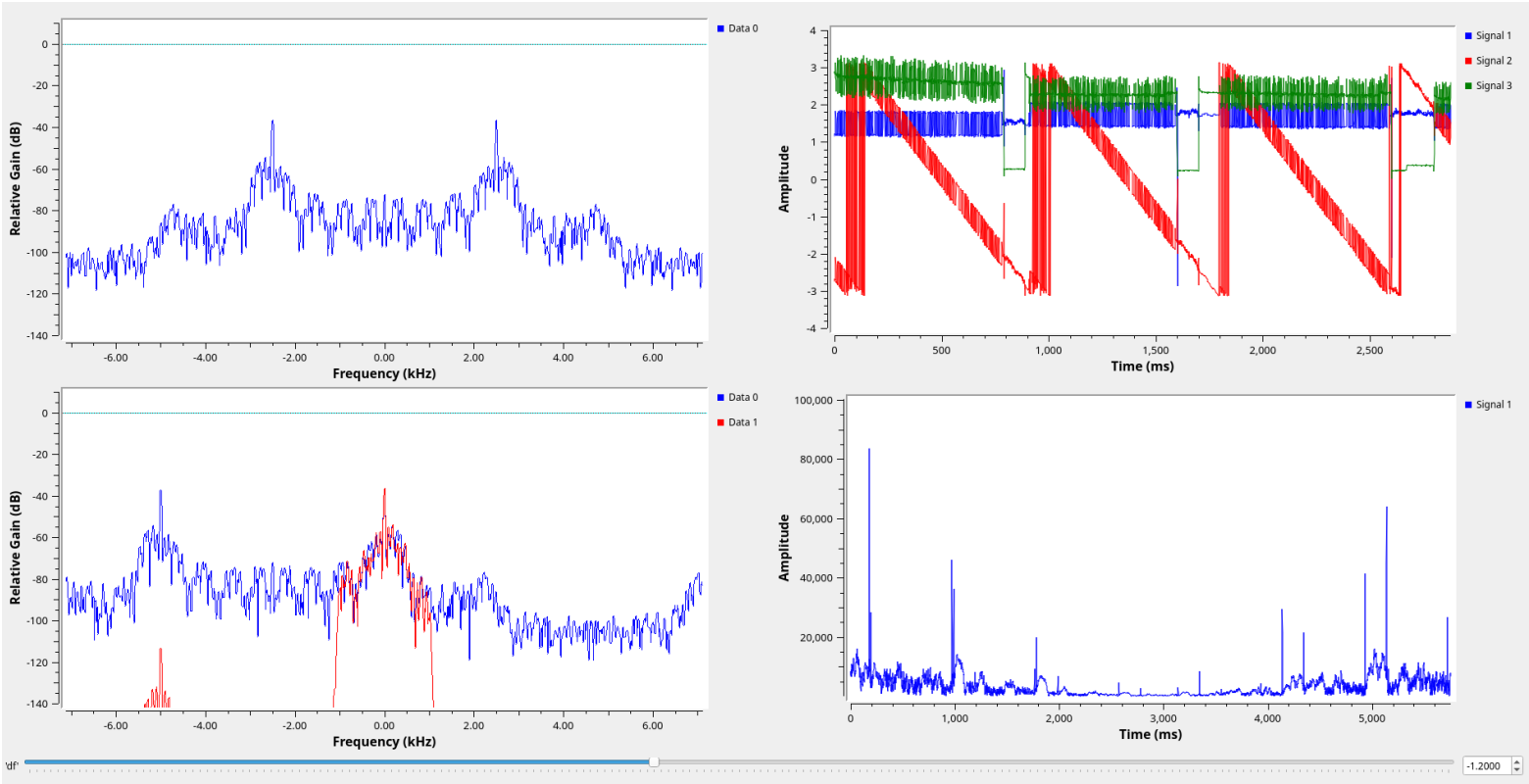
leading to the following figures:





With GNU Radio:




Data 0
Relative Gain (dB)
0
-20
-40
-60
-80
-100
-120
-140
-6.00
-4.00
-2.00
0.00
2.00
4.00
6.00
Frequency (kHz)
Signal 1
Signal 2
Signal 3
Amplitude
4
3
2
1
0
-1
-2
-3
-4
0
500
1,000
1,500
2,000
2,500
Time (ms)
Data 0
Data 1
Relative Gain (dB)
Frequency (kHz)
Signal 1
Amplitude
100,000
80,000
60,000
40,000
20,000
0
1,000
2,000
3,000
4,000
5,000
Time (ms)
'df'
-1.2000