Introduction in SDR and GNUradio project 101

Belgian SDR Meetup

Kristoff Bonne (ON1ARF)
4/9/2020

Agenda

- Introduction to SDR
  - Question time 1
- Project RTTY decode:
  - the hardware
  - the RTTY-signaal
  - Intro GNU Radio + DWD demodulator in GNU Radio
  - Question time 2

Agenda (2)

- Project RTTY decoder:
  - Baudot Decoder
  - Everything together
- GNU Radio .. summary
- Question time 3

Practical Issues

This is a video-conference, so please follow these rules

- Use a headset
- Best browsers:
  - chrome / chromium / …
  - Firefox
- Mute mic if you have nothing to say
- Switch off your camera if you have nothing to show

Practical Issues

This is a video-conference, so please follow these rules

- Feel free to give feelback!
  - There are three “question time” moments
  - You can also use the “chat”
  - Feel free to use your own language to ask questions
    - I will reply in English

Practical Issues

The video-presentation will be recorded and can be placed online.

Note, that, if you speak, your video (if on) and your name / call will be visible!

Goal

- Practicle goal:
  - Receive, demodulate and decode RTTY signal
  - DWD: always active, very strong

- Secundairy goals:
  - Starting to learn GNUradio
  - Starting to learn decoding
    - RTTY: simpel, easy to demodulate (FSK), slow

Legal

What we want to achieve

$telnet pan2.local 5301

Connected to pan2.local.

9579 10018 47998 /3116 10051 21050 40069 52015 700//
22200 10910 70048=

NNNN
ZCZC 993
FQEW57 EDZW 030600

SEEWETTERBERICHT FUER WESTEUROPAEISCHE GEWAESSER HERAUSGEGEBEN
VOM SEEWETTERDIENST HAMBURG 03.04.20, 00 UTC:

WETTERLAGE:
STURMTIEF 966 BO ….

SDR: … a very Confusing Name

SDR: Software Defined Radio

Although SDR is based on software, it is not the main element of SDR

A better name should be .. MDR
(more on that later)

Explaining SDR

SDR

My (personal) definition:

The use of DSP techniques and hardware peripherals to simulate a radio receiver and/or transmitter inside a computing-device

So .. what is DSP?

Explaining SDR / DSP

SDR: Software Defined Radio is based on DSP.

DSP: ”Digital Signal Processing”

So … what is a signal?

Explaining SDR / DSP / Signal

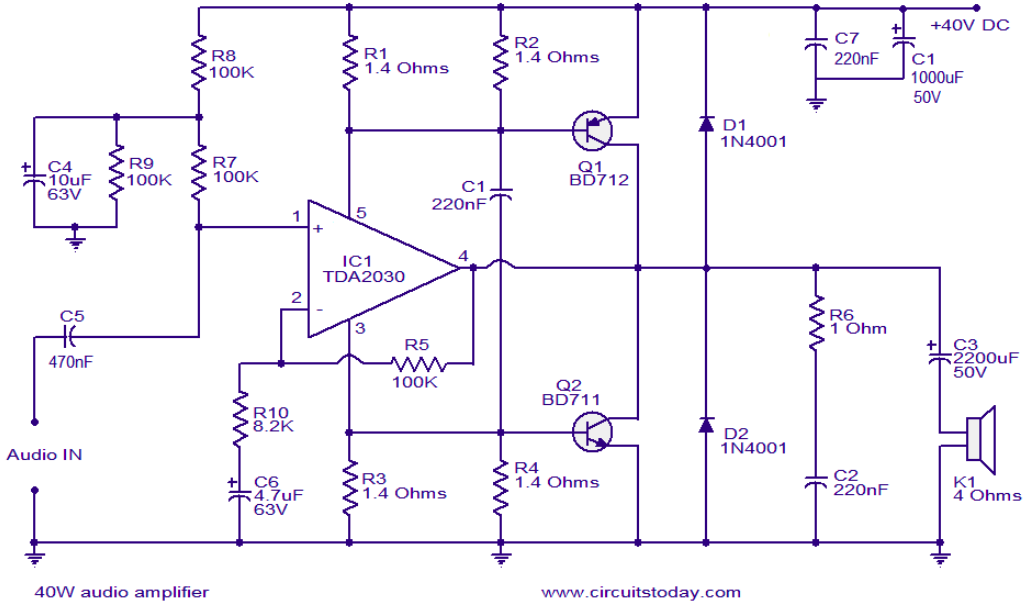
Signal: a medium carrying information

- Medium: electricity, light, Electromagnetic waves (i.e. radiowaves)
- Information: .. audio, video, text, data, ...

Signal Processing

Analog:
e.g. using analog electronics:

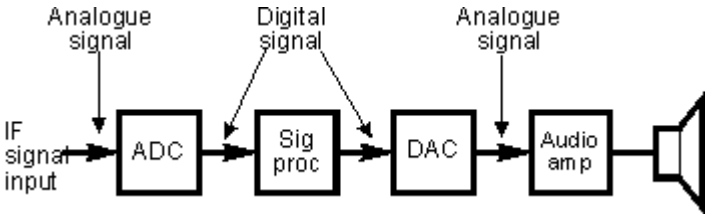
- E.g. Circuits using electronic components (active, passive) or physical characteristics of the components (speaker, PCB trace, ...)



https://www.circuitstoday.com/40w-audio-amplifier

Signal Processing

Digital signal processing:
Processing signal using “digital” components



Note: most signals are in essence analog by nature

- ADC: Analog Digital Convertor
- DAC: Digital Analog Convertor

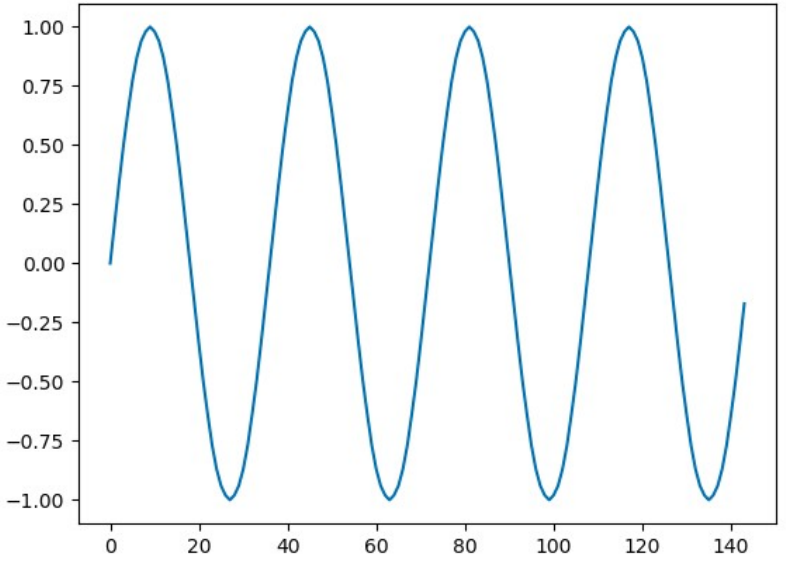
http://blog.arviups.com/explain-details-digital-signal-processing/

Signal Processing

- Analog signal: (sine wave)

  a voltage on the input port of an amplifier circuit, as it varies over time

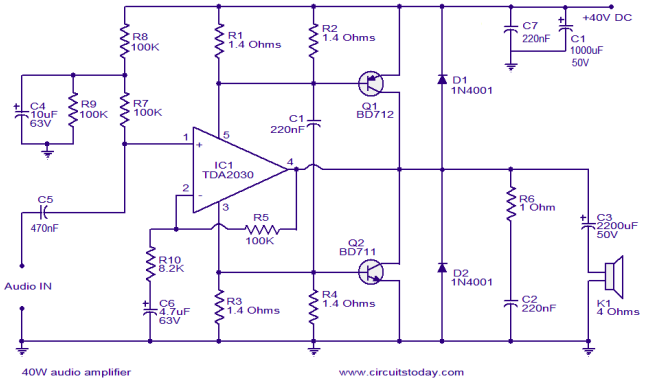
- Digital Signal:
  A list of values, that represent an analog signal (voltage, current, ...), either in time or Frequency domain



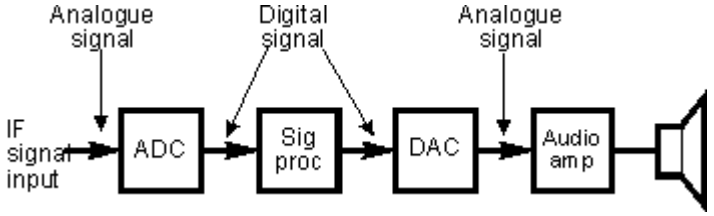
[0.00, 0.17, 0.34, 0.50, 0.64, 0.77, 0.87, 0.94, 0.98, 1.00, 0.98, 0.94, 0.87, 0.77, 0.64, 0.50, 0.34, 0.17, 0.00, -0.17, -0.34, -0.50, -0.64, -0.77, -0.87, -0.94, -0.98, -1.00, -0.98, -0.94, -0.87, -0.77, -0.64, -0.50, -0.34, -0.17]

But what does this mean?

How do you go from



To



But how does this work?

Some misconceptions:

- Well, it’s a computer, and the computer does this, no?

  But how does the computer know how to do this?

But how does this work?

Some misconceptions:

- Well, it’s software

  OK, but software is only as good as the person who wrote this.
  So, how does this software work?

But how does this work?

Some misconceptions:

- I guess it is .. magic ?

  Not really ...

But how does this work?

Some misconceptions:

- I guess is it .. magic ?

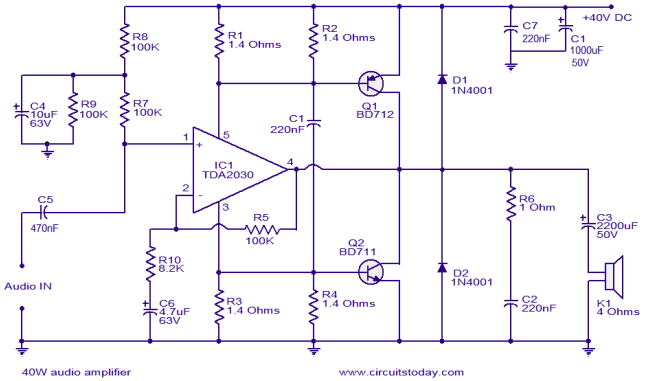
  Not really … it is MATH!

Mathematical functions

DSP program-code is based on mathematical functions (“algorithms”)

- Functional equivalent to the analog circuit.
- Examples:

Amplifier:
$y(t) = A * x(t)$



Every value of the output-stream at time “t”, is the value of the input stream at time “t” multiplied by “A” (amplification value)

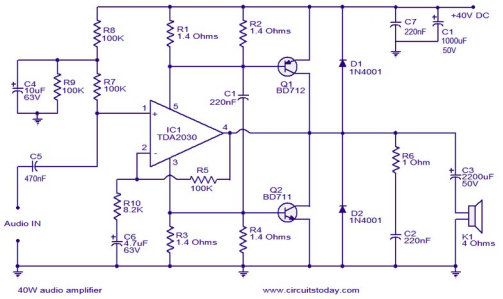
Mathematical functions

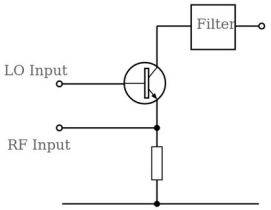
- Amplifier:
  $y(t) = A * x(t)$

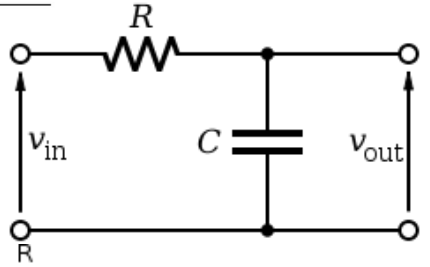
- Frequency Mixer:
  $y(t) = x_1(t) * x_2(t)$

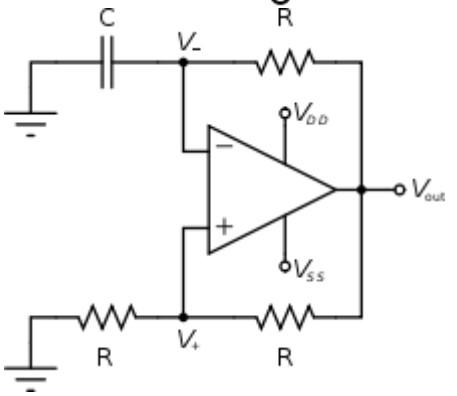
- Low-pass Filter:
  $y(t) = ½ (A_1 * x(t) + A_2 * x(t-1))$

- Oscillator
  $y(t) = \sin(t*(2*pi)/F)$





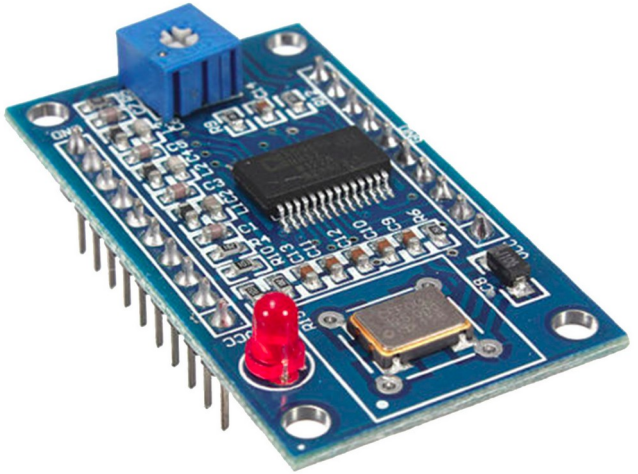




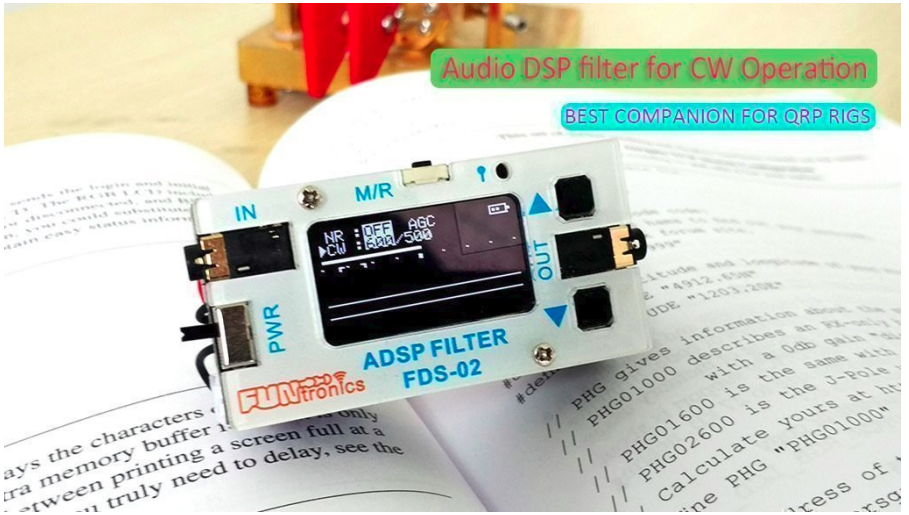
https://www.electronics-notes.com/articles/radio/rf-mixer/bipolar-transistor-mixer.php
https://en.wikipedia.org/wiki/Low-pass_filter
https://en.wikipedia.org/wiki/Electronic_oscillator

DSP applications

AD9851 Sinewave generator



DSP Filter

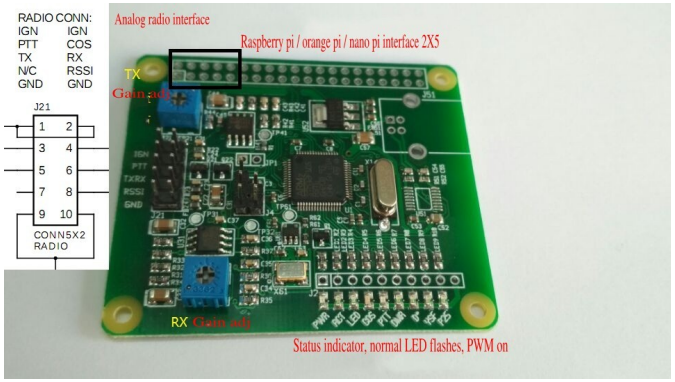


https://www.tinytronics.nl/shop/nl/diversen/overige/ad9851-dds-signaal-generator-module-0-70mhz
https://elekitsorparts.com/product/external-audio-dsp-filter-module-for-ssb-cw-ft8x7-dsp-filter

DSP applications

Baycom modem
(packet radio)

TCM3105 tx: SDR



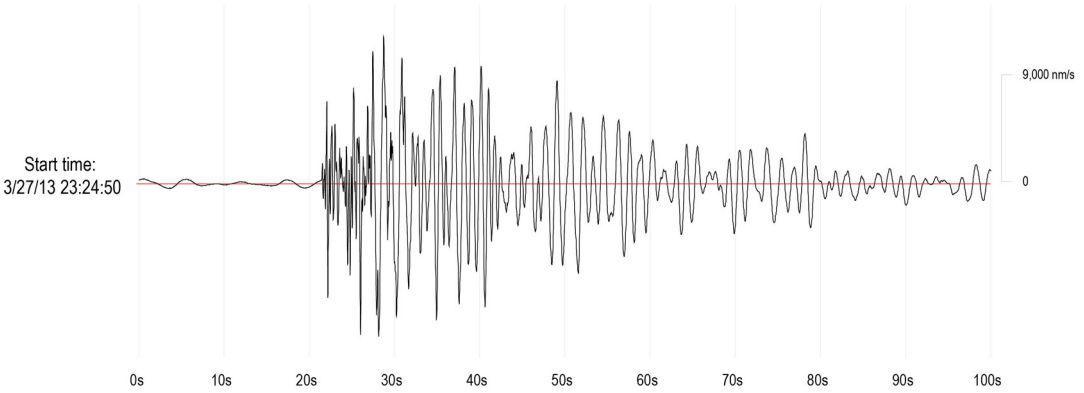
MMDVM

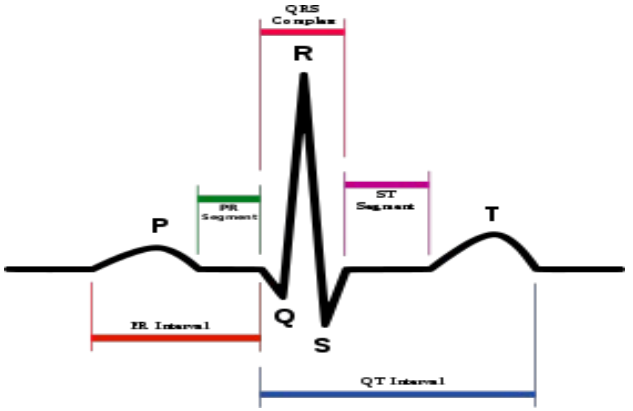
```
{
  ::memset(m_rrcState,       0x00U,   70U * sizeof(q15_t));
  ::memset(m_gaussianState,  0x00U,   40U * sizeof(q15_t));
  ::memset(m_boxcarState,    0x00U,   30U * sizeof(q15_t));
  ::memset(m_nxdnState,      0x00U,  110U * sizeof(q15_t));
  ::memset(m_nxdnISincState, 0x00U,   60U * sizeof(q15_t));
  ::memset(m_dcState,        0x00U,    4U * sizeof(q31_t));

  m_dcFilter.numStages = DC_FILTER_STAGES;
  m_dcFilter.pState    = m_dcState;
  m_dcFilter.pCoeffs   = DC_FILTER;
  m_dcFilter.postShift = 0;

  m_rrcFilter.numTaps = RRC_0_2_FILTER_LEN;
  m_rrcFilter.pState  = m_rrcState;
  m_rrcFilter.pCoeffs = RRC_0_2_FILTER;

  m_gaussianFilter.numTaps = GAUSSIAN_0_5_FILTER_LEN;
  m_gaussianFilter.pState  = m_gaussianState;
  m_gaussianFilter.pCoeffs = GAUSSIAN_0_5_FILTER;

  m_boxcarFilter.numTaps = BOXCAR_FILTER_LEN;
  m_boxcarFilter.pState  = m_boxcarState;
  m_boxcarFilter.pCoeffs = BOXCAR_FILTER;

  m_nxdnFilter.numTaps = NXDN_0_2_FILTER_LEN;
```

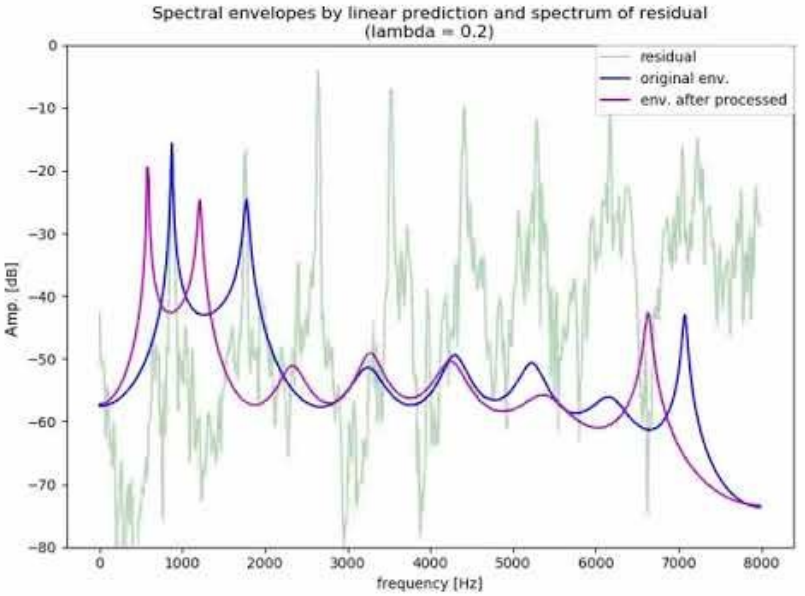
DSP code in MMDVM:
https://github.com/g4klx/MMDVM/blob/master/IO.cpp

http://sm0vpo.altervista.org/use/baycom.htm
https://www.aliexpress.com/item/32322805517.html
https://www.ebay.com/itm/MMDVM-Multi-Mode-Digital-Voice-Modem-for-Raspberry-Pi-Arduino-Orange-Pi-/112744287264

DSP applications







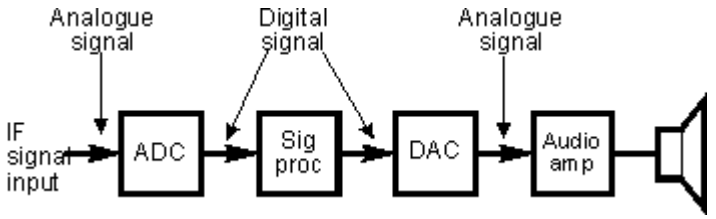
https://www.youtube.com/watch?v=Vjg0fkXzVgM
http://ds.iris.edu/aed2/c/alaska/earthquakes/eq_5.html
https://en.wikipedia.org/wiki/Electrocardiography

SDR: definition (2nd try)

Use of Mathematical functions

to process digital radio signals,

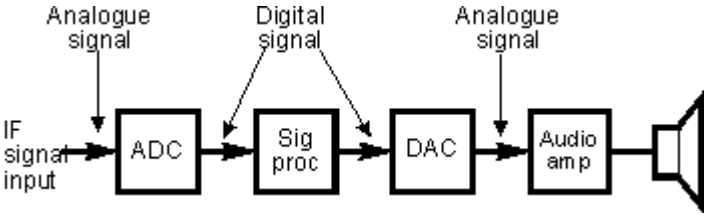
as implemented in software on a digital device (mpu, cpu, gpu) or digital logic (fpga, asic)

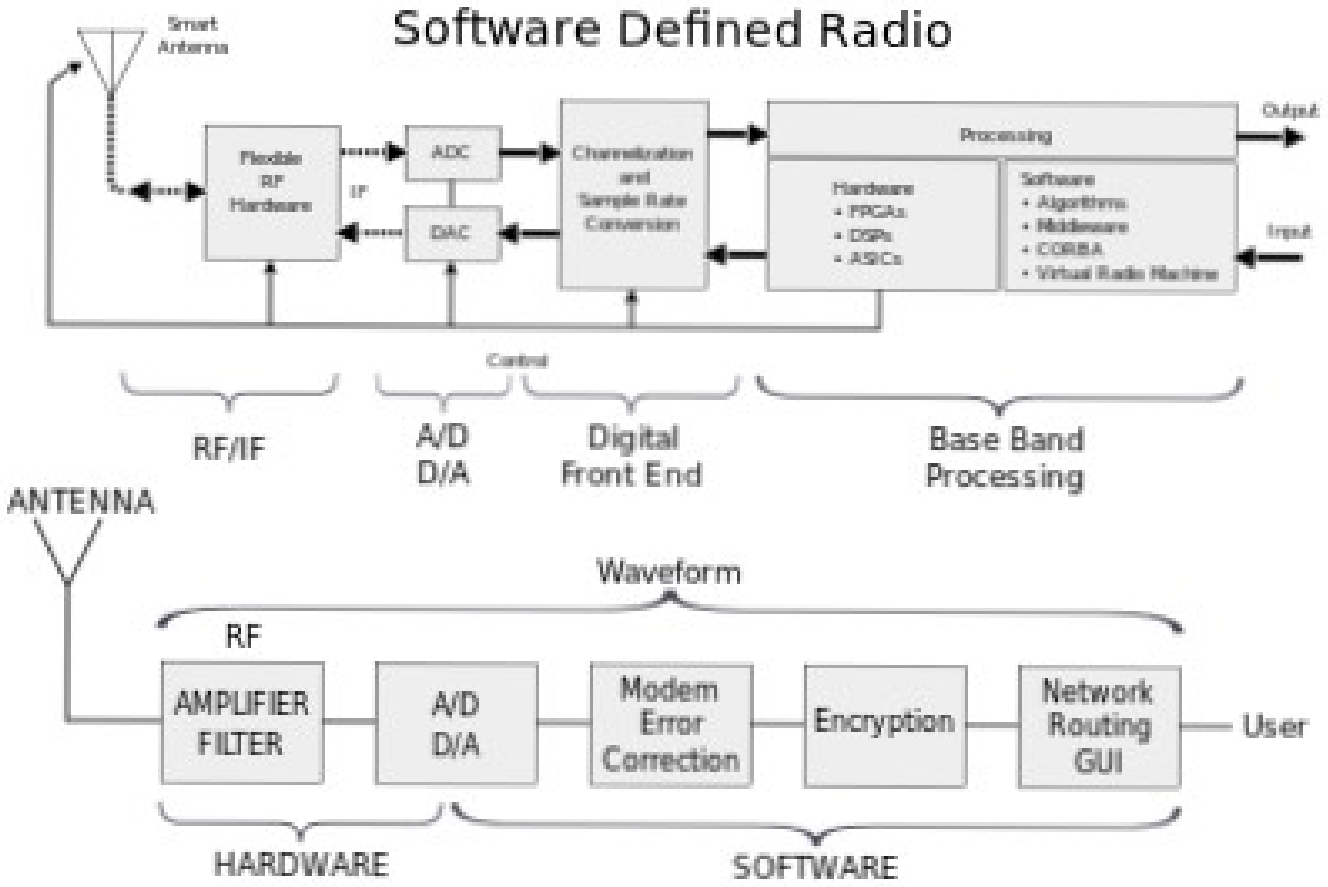


Low-pass Filter:

$y(t) = ½ (A_1 * x(t) + A_2 * x(t-1))$

SDR: Overview



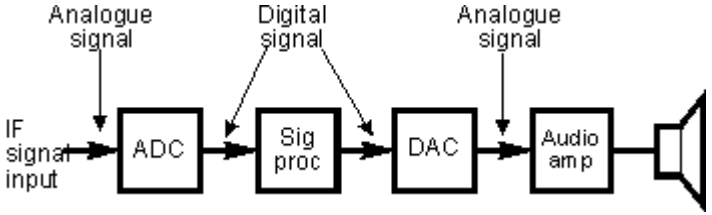


https://en.wikipedia.org/wiki/Software-defined_radio

SDR: features (1)

Uses “SDR Peripheral” (receiver / transmitter)

- RTL-SDR, hackRF, PlutoSDR



It’s like a soundcard, only for radio-signals (M. Ossman)

https://greatscottgadgets.com/sdr/5/
https://www.maes-electr.be/nl/radioamateur-sdr/rtl-sdr/rtl-sdr-v3-usb-dongle/2176/

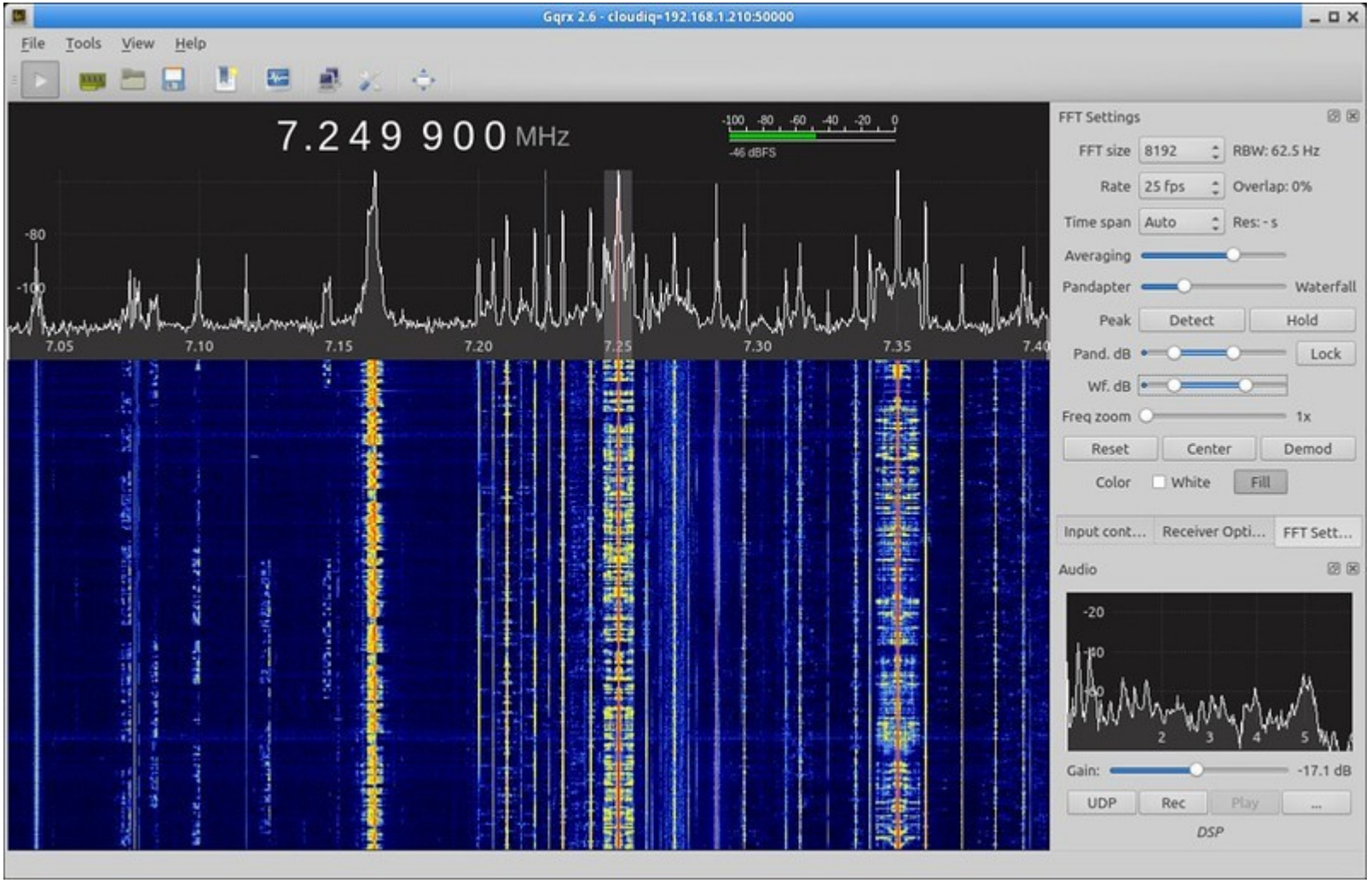
SDR: features (2)

- Uses I/Q signals

For every sample, two values are samples, 90 degrees apart.
Adds phase-information to the received signals

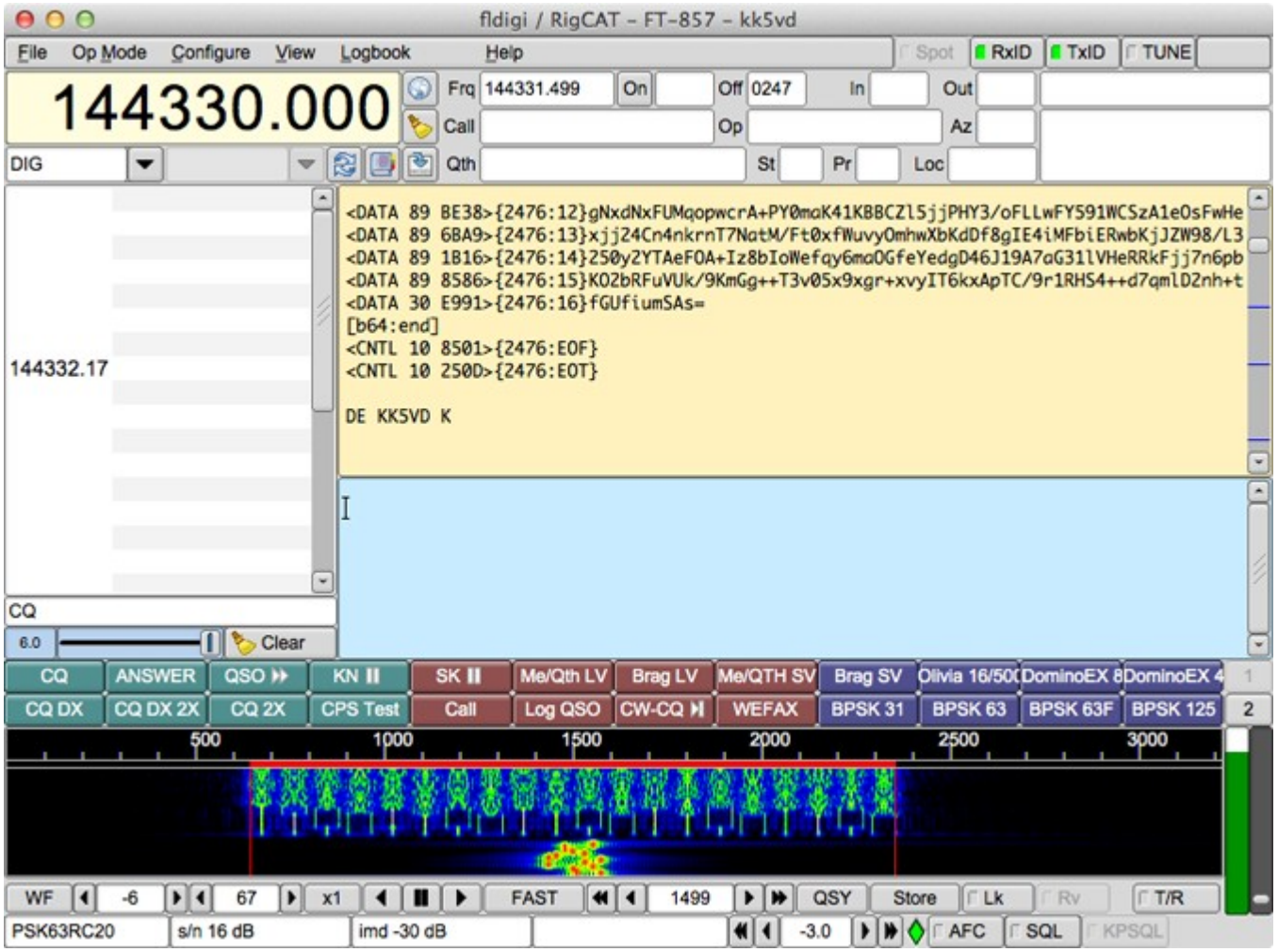
https://www.youtube.com/watch?v=JuuKF1RFvBM

SDR applications



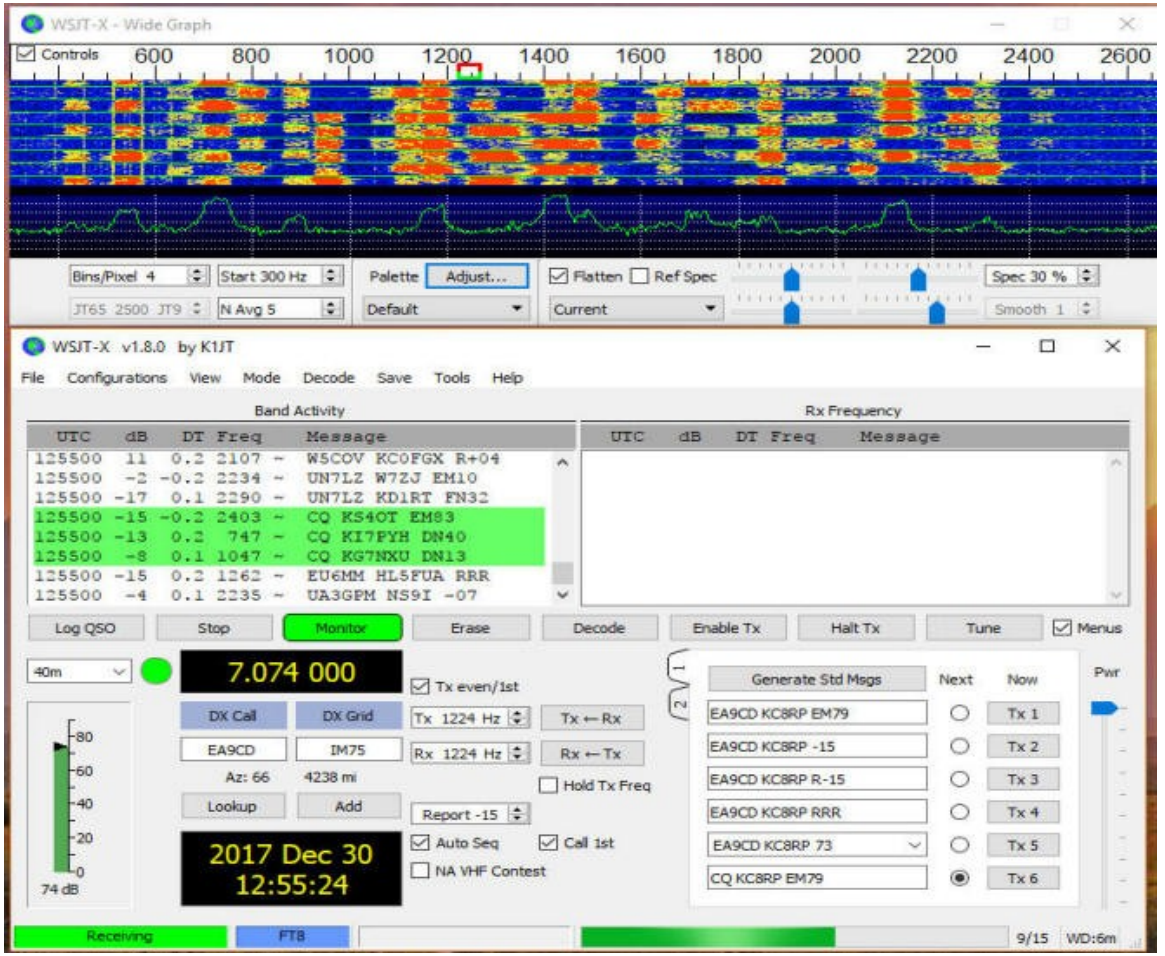
https://gqrx.dk/

SDR applications



https://sourceforge.net/projects/fldigi/

DSP/SDR applications

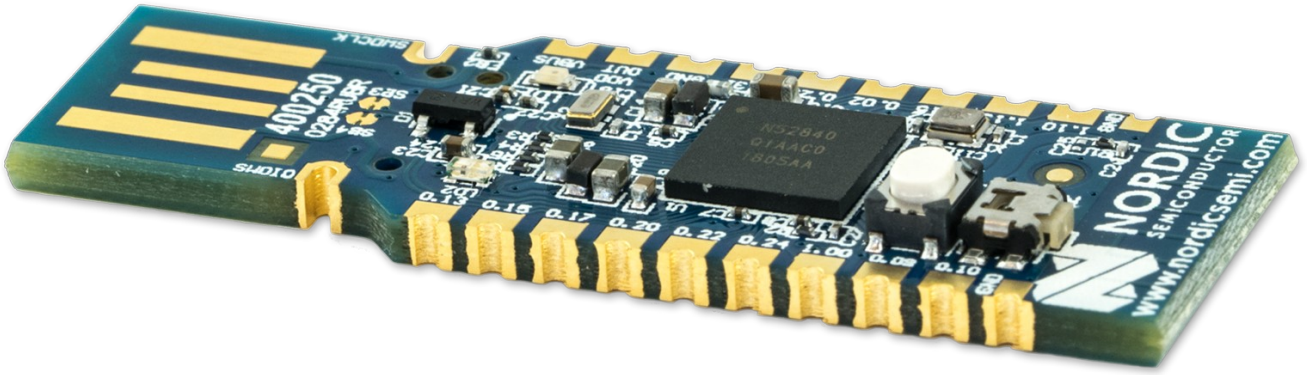


https://www.pestingers.net/pages-images/ham-stuff/digital-modes/digital-software/wsjt-x/wsjt-x.htm

SDR applications

https://www.pinterest.com/pin/659495939160969257/

SDR applications



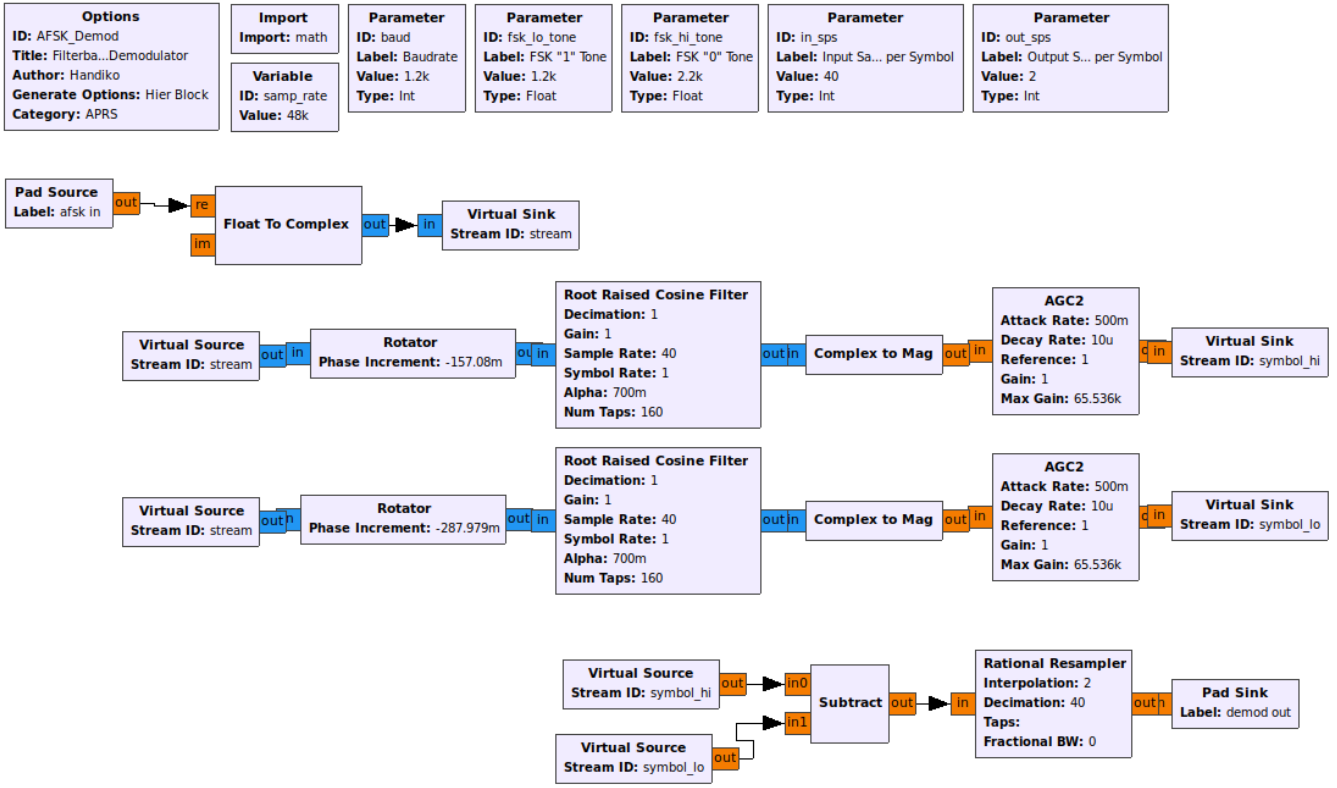
https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52840-Dongle

DSP/SDR applications

https://www.amazon.com/BaoFeng-UV-5R-Dual-Band-136-174-400-480/

DSP/SDR applications

GNU Radio



https://handiko.github.io/gr-APRS/

Question time (1)

Questions?

SDR project: RTTY receiver project

Goal:

Receive Deutsche Wetter Dienst RTTY (147.3 Khz): 50 baud RTTY

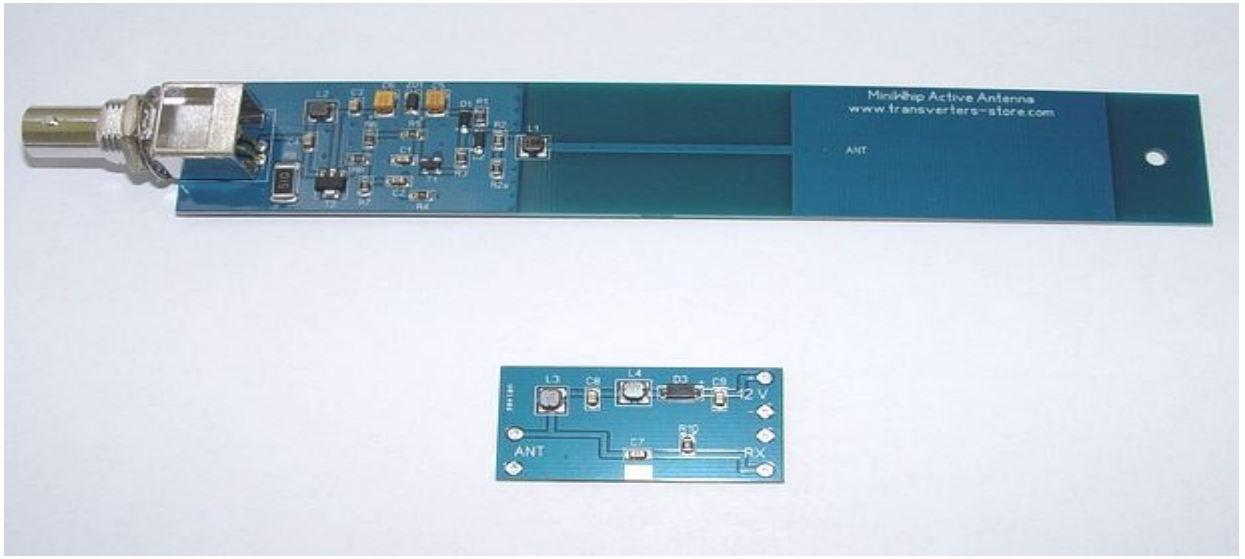
Learn GNU Radio

Hardware

The antenna

Hardware

Antenna: active whip-antenna:
http://transverters-store.com/whippcb.htm



Hardware

- RTLSDRv3
  in “direct mode”
- Bias-T (12 V) for
  Antenna
- Netbook (debian)
  - Power Supply
    (relative ‘clean’)

Software: RTL_TCP (1)

RTL_TCP: Part of the “RTL-SDR” software under linux / MacOS:

Allows you to “listen to” a rtl-sdr dongle at a distance.
(e.g. RPi or other single-board computer + RTL-SDR dongle at the attic / mast / garage …)

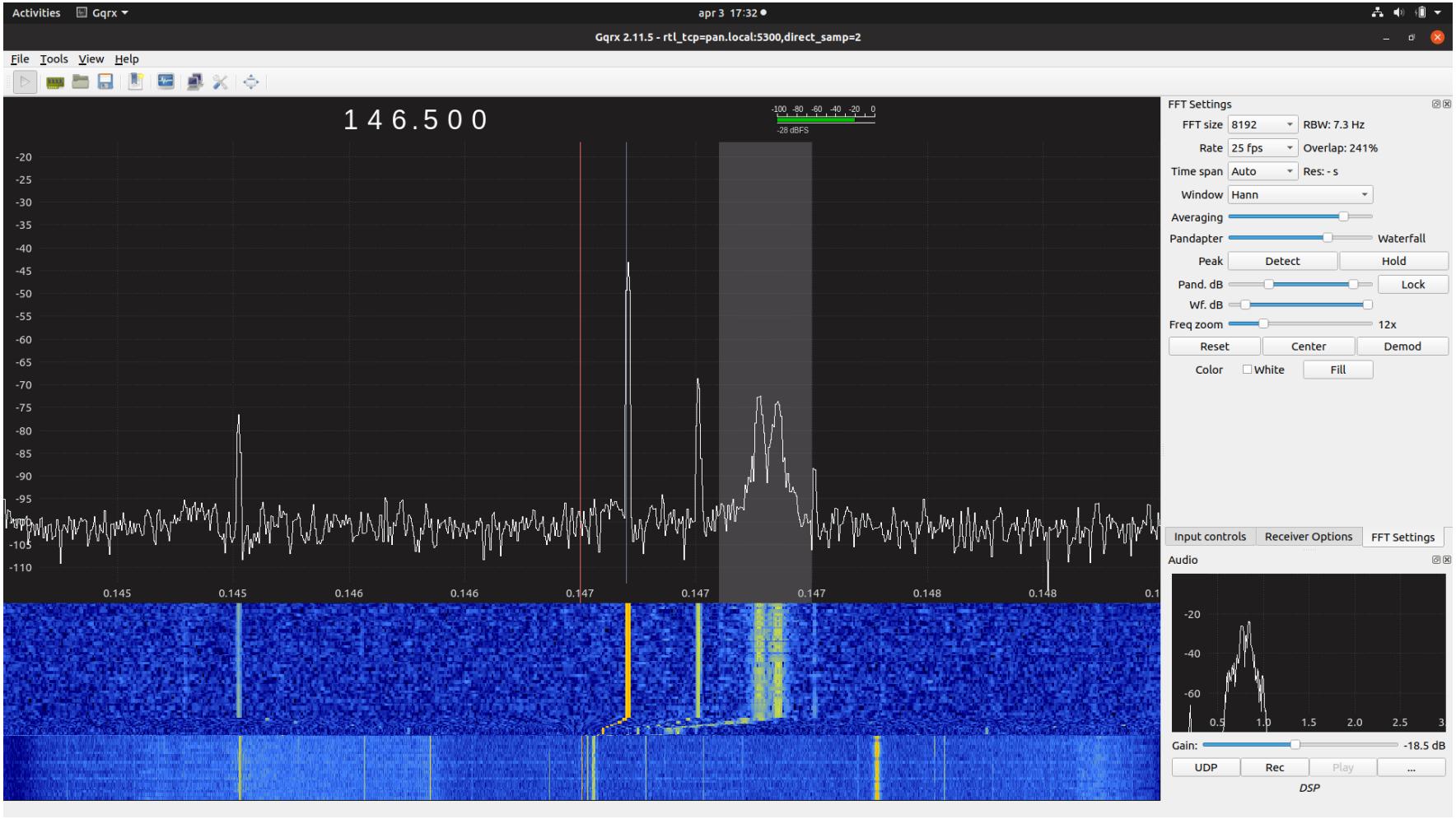
- active whip antenne
- QO-100 receiver

Software: RTL_TCP (2)

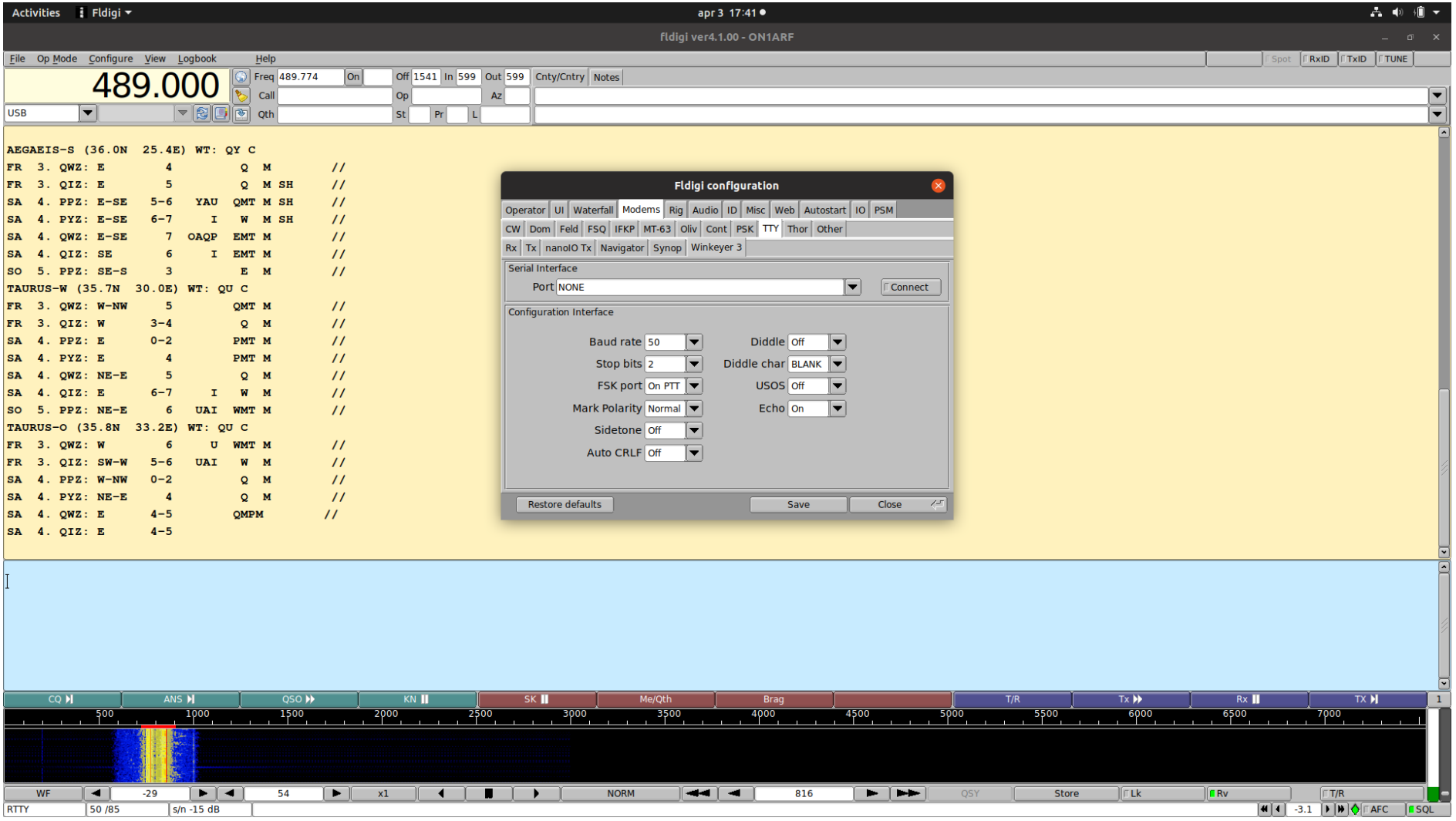
RTL_TCP: “RTL-SDR” software

- Note: you need a special version for the RTL-SDRv3 dongle in “direct mode”
- rtl_tcp -p 5300 -a 0.0.0.0 -s 960000 -D 1

The Signal (1): gqrx



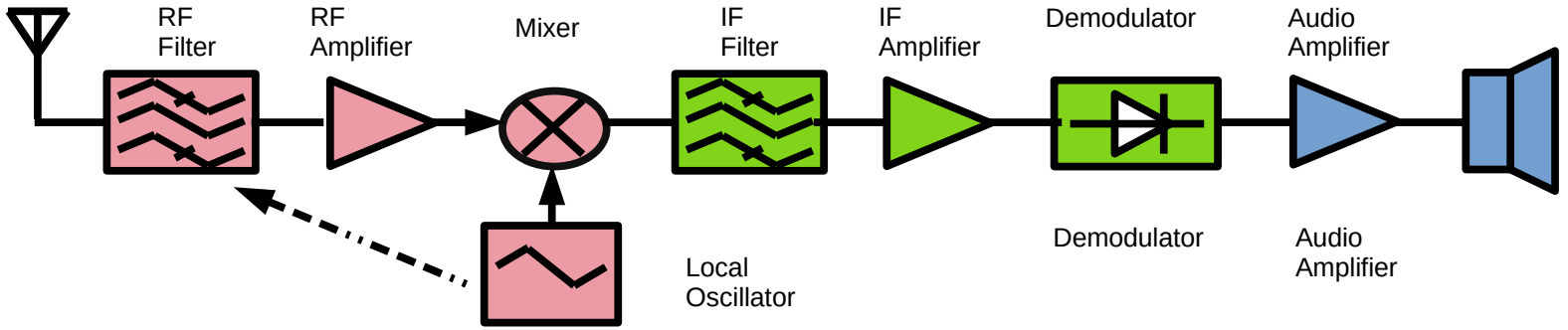
The Signal (2): fldigi



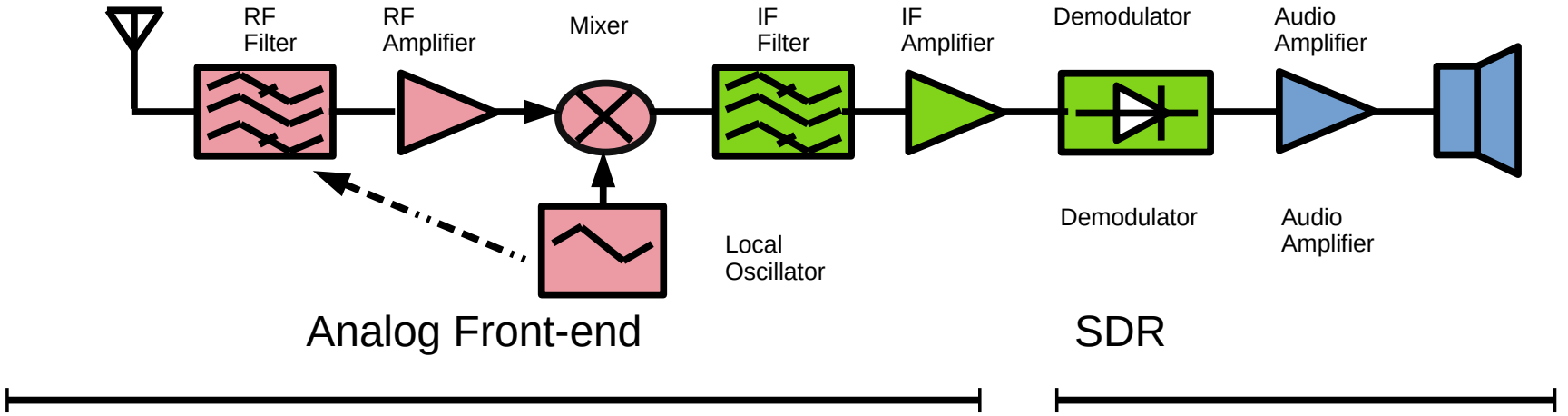
GNU Radio: design your own SDR application

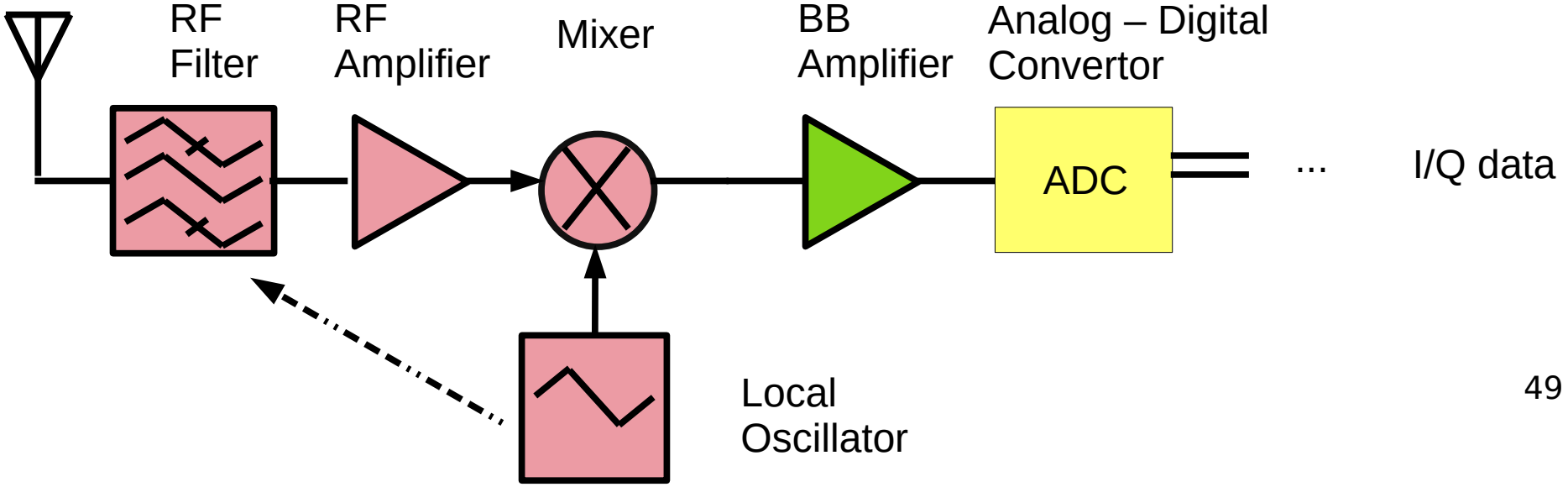
- SDR Framework
- Create SDR “flowgraphs”
- Visual: “GNURadio Companion” (GRC)
- Blocks (but: what do they exactly do?, Which one? Parameters?)
- Block: implements a SDR or DSP “function”
- Note: you do are creating an radio-device, so radio-knowledge IS REQUIRED

How to make a receiver? (analog)

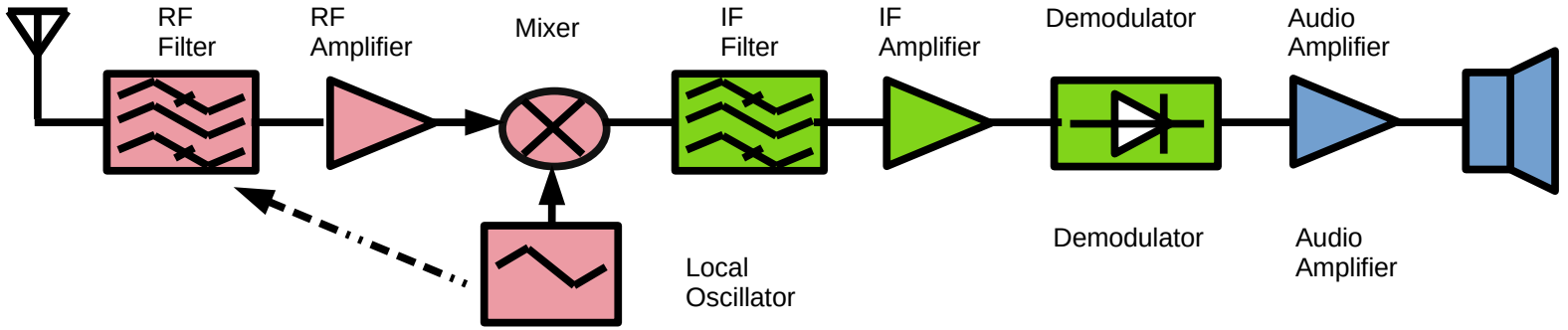


How to make a receiver? (SDR)





But this is how you must think!

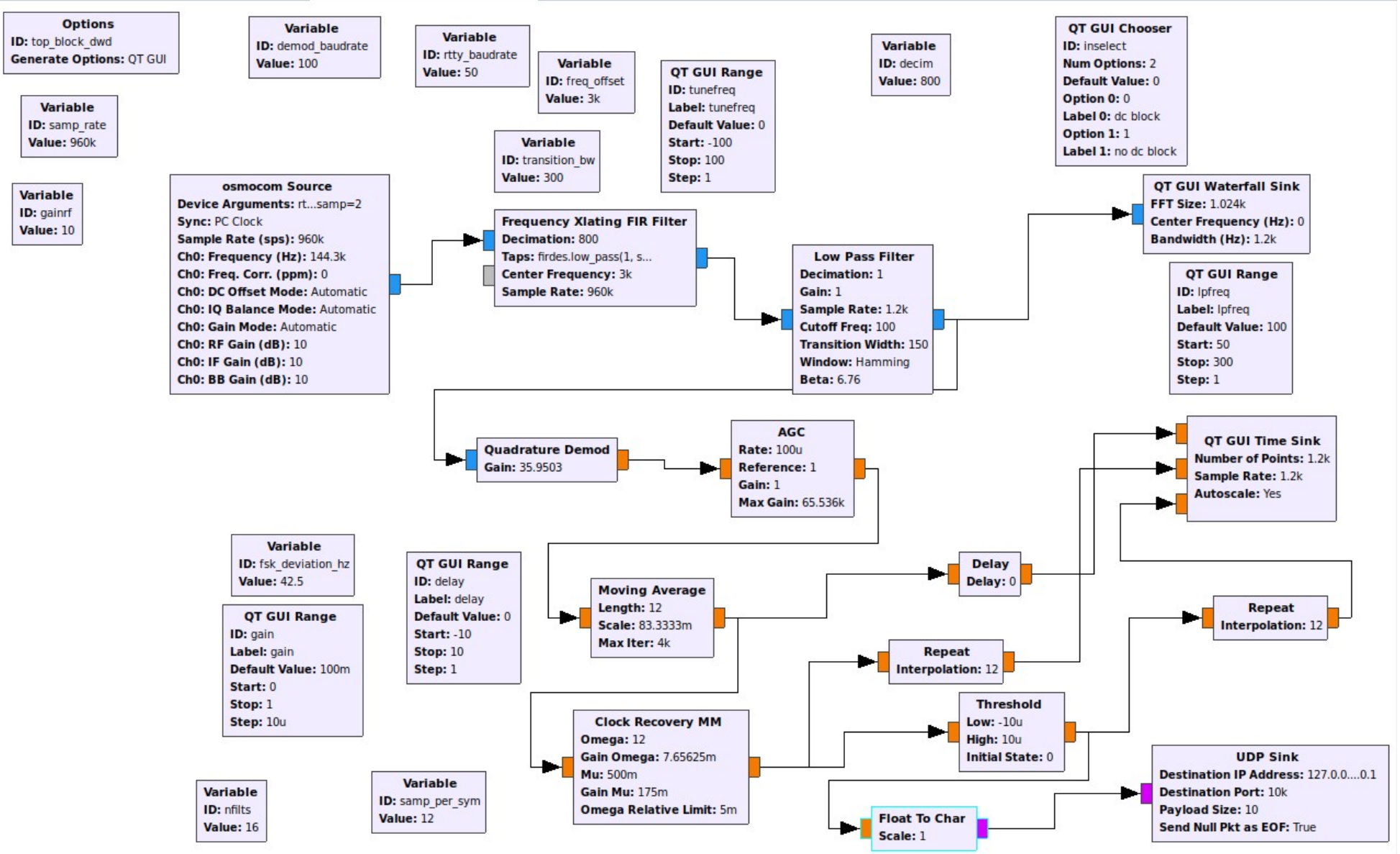


How to make a receiver? (SDR)

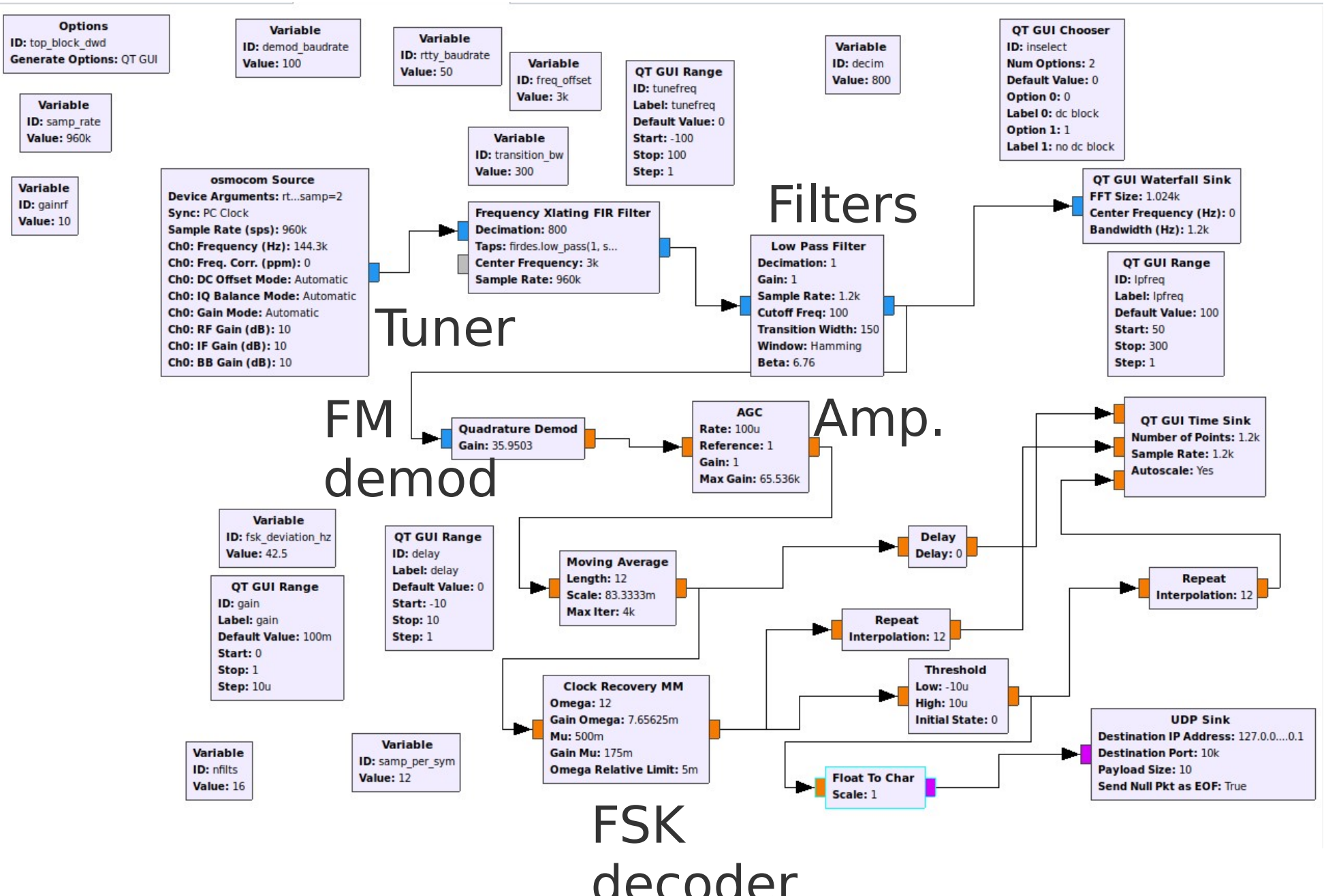
What do we need?

- Receiver (RTL-SDR dongle + rtl_tcp)
- Tuner (down-convertor IF / AF)
- Filters + amplifiers
- Demodulator
  - Audio out (speaker)
  - Decoder (RTTY)

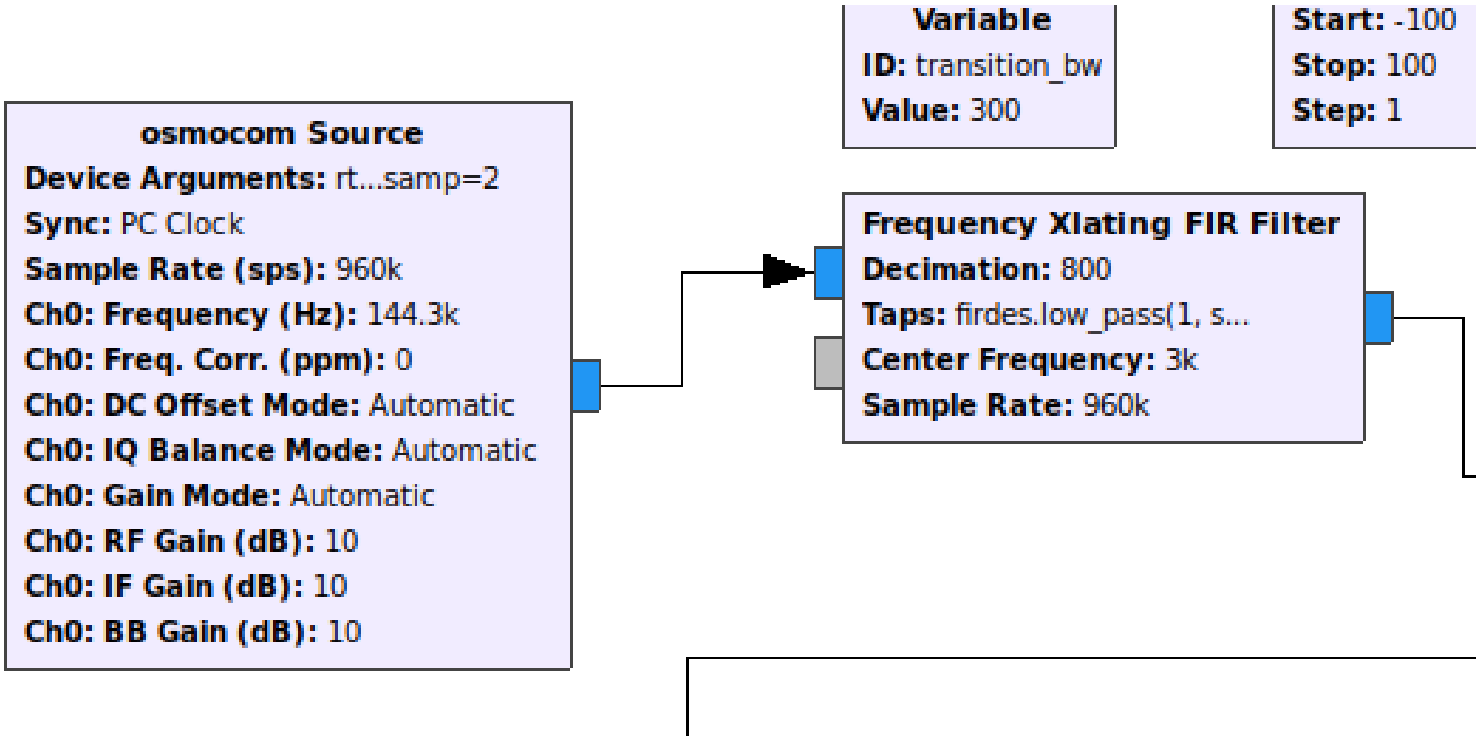
GRC


Options
ID: top_block_dwd
Generate Options: QT GUI
Variable
ID: demod_baudrate
Value: 100
Variable
ID: rtty_baudrate
Value: 50
Variable
ID: freq_offset
Value: 3k
QT GUI Range
ID: tunefreq
Label: tunefreq
Default Value: 0
Start: -100
Stop: 100
Step: 1
Variable
ID: decim
Value: 800
QT GUI Chooser
ID: inselect
Num Options: 2
Default Value: 0
Option 0: 0
Label 0: dc block
Option 1: 1
Label 1: no dc block
Variable
ID: samp_rate
Value: 960k
Variable
ID: transition_bw
Value: 300
Variable
ID: gainrf
Value: 10
osmocom Source
Device Arguments: rt...samp=2
Sync: PC Clock
Sample Rate (sps): 960k
Ch0: Frequency (Hz): 144.3k
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Automatic
Ch0: IQ Balance Mode: Automatic
Ch0: Gain Mode: Automatic
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 10
Ch0: BB Gain (dB): 10
Frequency Xlating FIR Filter
Decimation: 800
Taps: firdes.low_pass(1, s...
Center Frequency: 3k
Sample Rate: 960k
Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 1.2k
Cutoff Freq: 100
Transition Width: 150
Window: Hamming
Beta: 6.76
QT GUI Waterfall Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.2k
QT GUI Range
ID: lpfreq
Label: lpfreq
Default Value: 100
Start: 50
Stop: 300
Step: 1
Quadrature Demod
Gain: 35.9503
AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 65.536k
QT GUI Time Sink
Number of Points: 1.2k
Sample Rate: 1.2k
Autoscale: Yes
Variable
ID: fsk_deviation_hz
Value: 42.5
QT GUI Range
ID: delay
Label: delay
Default Value: 0
Start: -10
Stop: 10
Step: 1
Moving Average
Length: 12
Scale: 83.3333m
Max Iter: 4k
Delay
Delay: 0
Repeat
Interpolation: 12
QT GUI Range
ID: gain
Label: gain
Default Value: 100m
Start: 0
Stop: 1
Step: 10u
Repeat
Interpolation: 12
Threshold
Low: -10u
High: 10u
Initial State: 0
Clock Recovery MM
Omega: 12
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m
Variable
ID: nfilts
Value: 16
Variable
ID: samp_per_sym
Value: 12
Float To Char
Scale: 1
UDP Sink
Destination IP Address: 127.0.0....0.1
Destination Port: 10k
Payload Size: 10
Send Null Pkt as EOF: True

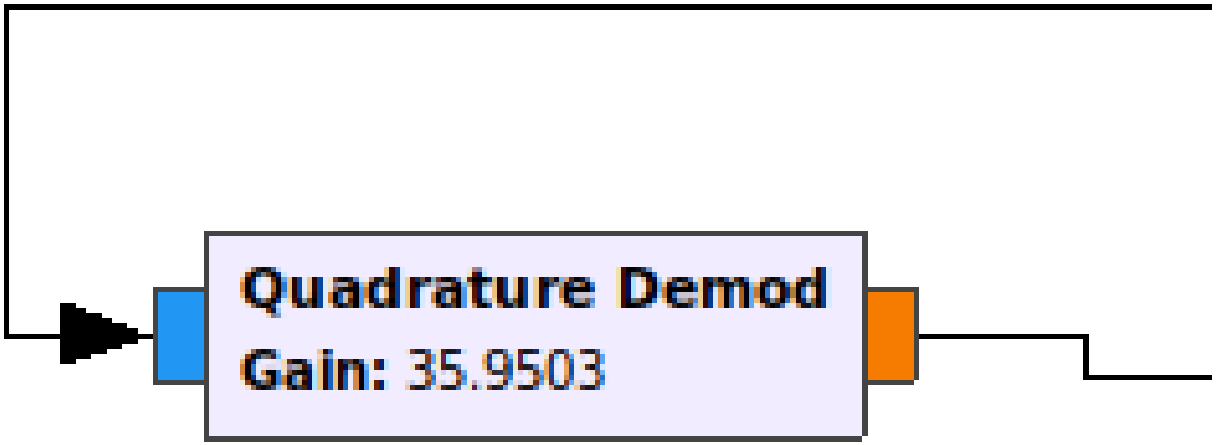
GRC


Options
ID: top_block_dwd
Generate Options: QT GUI
Variable
ID: demod_baudrate
Value: 100
Variable
ID: rtty_baudrate
Value: 50
Variable
ID: freq_offset
Value: 3k
QT GUI Range
ID: tunefreq
Label: tunefreq
Default Value: 0
Start: -100
Stop: 100
Step: 1
Variable
ID: decim
Value: 800
QT GUI Chooser
ID: inselect
Num Options: 2
Default Value: 0
Option 0: 0
Label 0: dc block
Option 1: 1
Label 1: no dc block
Variable
ID: samp_rate
Value: 960k
Variable
ID: transition_bw
Value: 300
Variable
ID: gainrf
Value: 10
osmocom Source
Device Arguments: rt...samp=2
Sync: PC Clock
Sample Rate (sps): 960k
Ch0: Frequency (Hz): 144.3k
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Automatic
Ch0: IQ Balance Mode: Automatic
Ch0: Gain Mode: Automatic
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 10
Ch0: BB Gain (dB): 10
Frequency Xlating FIR Filter
Decimation: 800
Taps: firdes.low_pass(1, s...
Center Frequency: 3k
Sample Rate: 960k
Filters
Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 1.2k
Cutoff Freq: 100
Transition Width: 150
Window: Hamming
Beta: 6.76
QT GUI Waterfall Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.2k
QT GUI Range
ID: lpfreq
Label: lpfreq
Default Value: 100
Start: 50
Stop: 300
Step: 1
Tuner
FM demod
Quadrature Demod
Gain: 35.9503
AGC
Rate: 100u
Reference: 1
Gain: 1
Max Gain: 65.536k
Amp.
QT GUI Time Sink
Number of Points: 1.2k
Sample Rate: 1.2k
Autoscale: Yes
Variable
ID: fsk_deviation_hz
Value: 42.5
QT GUI Range
ID: gain
Label: gain
Default Value: 100m
Start: 0
Stop: 1
Step: 10u
QT GUI Range
ID: delay
Label: delay
Default Value: 0
Start: -10
Stop: 10
Step: 1
Moving Average
Length: 12
Scale: 83.3333m
Max Iter: 4k
Delay
Delay: 0
Repeat
Interpolation: 12
Repeat
Interpolation: 12
Threshold
Low: -10u
High: 10u
Initial State: 0
Clock Recovery MM
Omega: 12
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5m
Variable
ID: nfilts
Value: 16
Variable
ID: samp_per_sym
Value: 12
Float To Char
Scale: 1
UDP Sink
Destination IP Address: 127.0.0....0.1
Destination Port: 10k
Payload Size: 10
Send Null Pkt as EOF: True
FSK decoder

Tuner + reduce bandwidth



FM demodulator



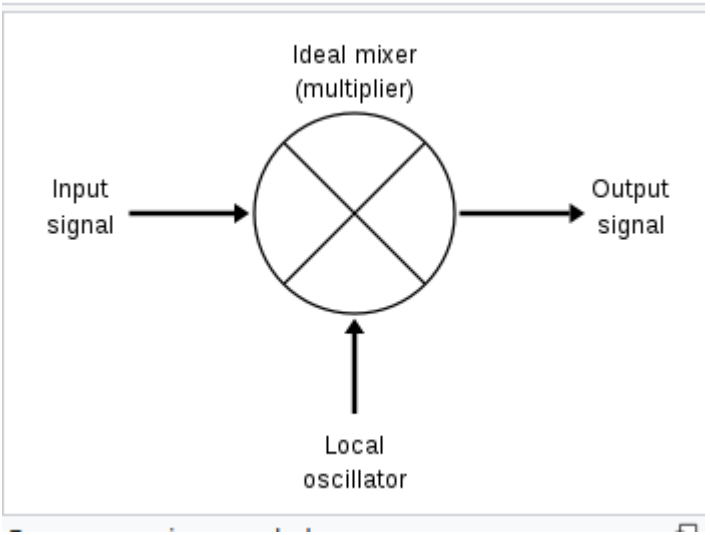
FM demodulator + filter

Let’s experiment !!!

Frequency mixer? (1)

Mixer (analog):
input 1 = f1
input 2 = f2

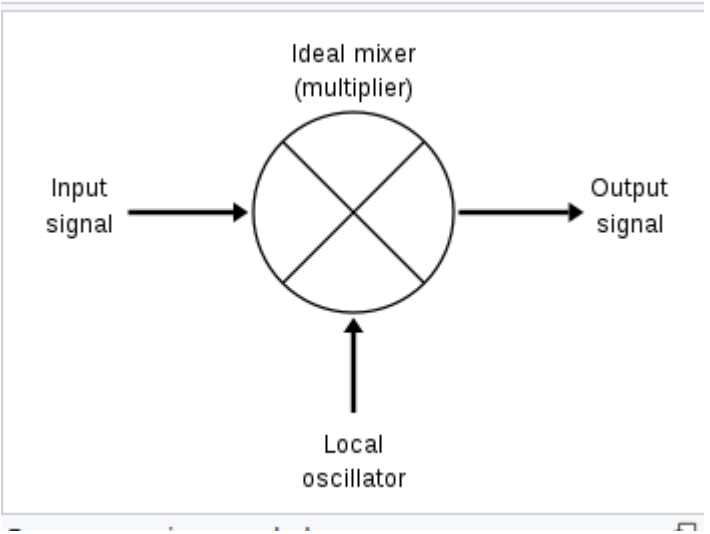
output: ?



https://en.wikipedia.org/wiki/Frequency_mixer

Frequency Mixer? (2)

Mixer (analog):
input 1 = f1
input 2 = f2



output:

- f1 + f2
- f1 – f2

So: 1000Hz * 1001 Hz → 1 Hz + 2001 Hz

Frequency Mixer? (3)

Mixer (Digital):
input 1 = f1
input 2 = f2

Frequency Mixer:

$y(t) = x_1(t) * x_2(t)$

output IF USING I/Q SIGNALS:

- f1 + f2

So: 1000Hz * 1001 Hz → 2001 Hz

Negative frequencies? (SDR) (1)

I/Q signal have a ‘direction’ (does ‘I’ preceed ‘Q’ or does ‘Q’ preceed ‘I’?)

- Q Preceeds I: + frequency
- I Preceeds Q: - frequency

So:

1000Hz * 1001 Hz → 2001 Hz

1001 Hz * -1000 Hz → 1 Hz

https://www.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-ac-analysis/v/ee-negative-frequency

https://www.pe0sat.vgnet.nl/sdr/iq-data-explained/

Negative frequencies? (SDR) (2)

1000Hz * 1001 Hz → 2001 Hz
1001 Hz * -1000 Hz → 1 Hz

- So:

1001 Hz * -1000 Hz → 1 Hz
999 Hz * -1000 Hz → -1 Hz

→ Baseband signal

FSK demodulation step 1: hardware tuner

- DWD: 147.3 KHz +- 37.5 Hz
- Hardware Tuner: 144.3 Khz
  - 2 tones on output of “tuner”:
    3000 Hz +- 37.5 Hz
  - 3000 Hz + 37.5 Hz = 3037.5 Hz
    3000 Hz – 37.5 Hz = 2962.5 Hz

FSK demodulation step 2: Freq.Xlating filter

3000 Hz + 37.5 Hz = 3037.5 Hz
3000 Hz – 37.5 Hz = 2962.5 Hz

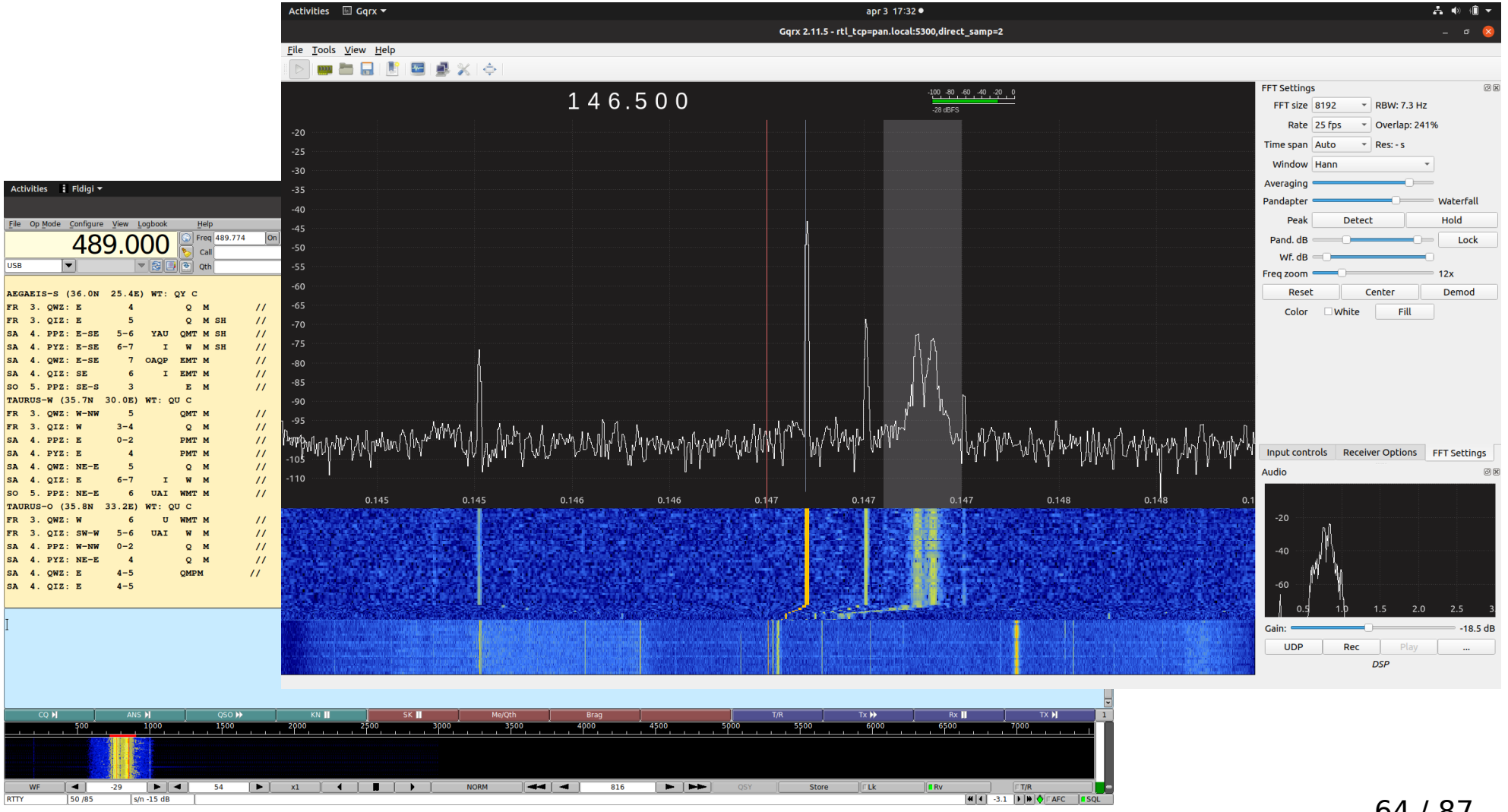
“freq. Xlating FIR filter”:

- Bandpassing filter
- Tuner: -3 Khz

  +37.5 Hz

  - 37.5 Hz
- Decimation: Reduce bandwith
  - 960 Khz bw → 1200 Hz

FSK demodulation
step 2: Freq.Xlating filter



FSK demodulatie: Step 3: FM demod

- Mixer (“freq. Xlating FIR filter): 3 Khz

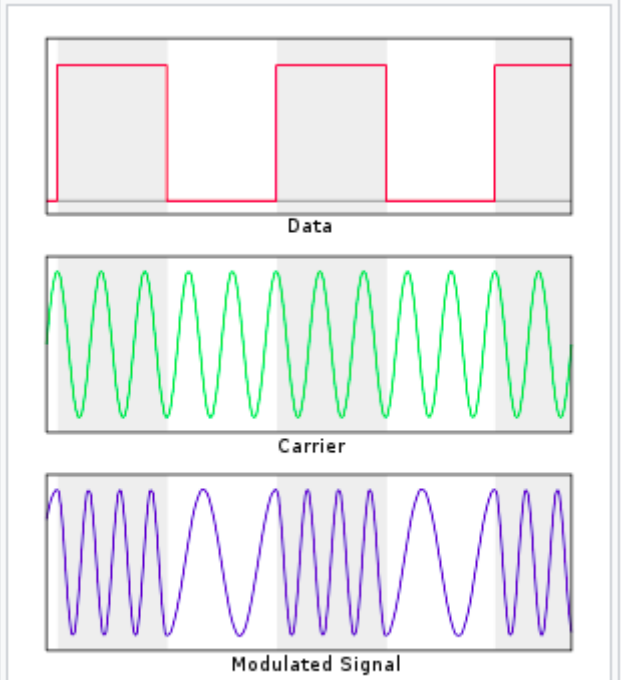
  +37.5 Hz

  - 37.5 Hz

- FM demod (“quadrature demod”)

  +37.5 Hz → positive output

  -37.5 Hz → negative output



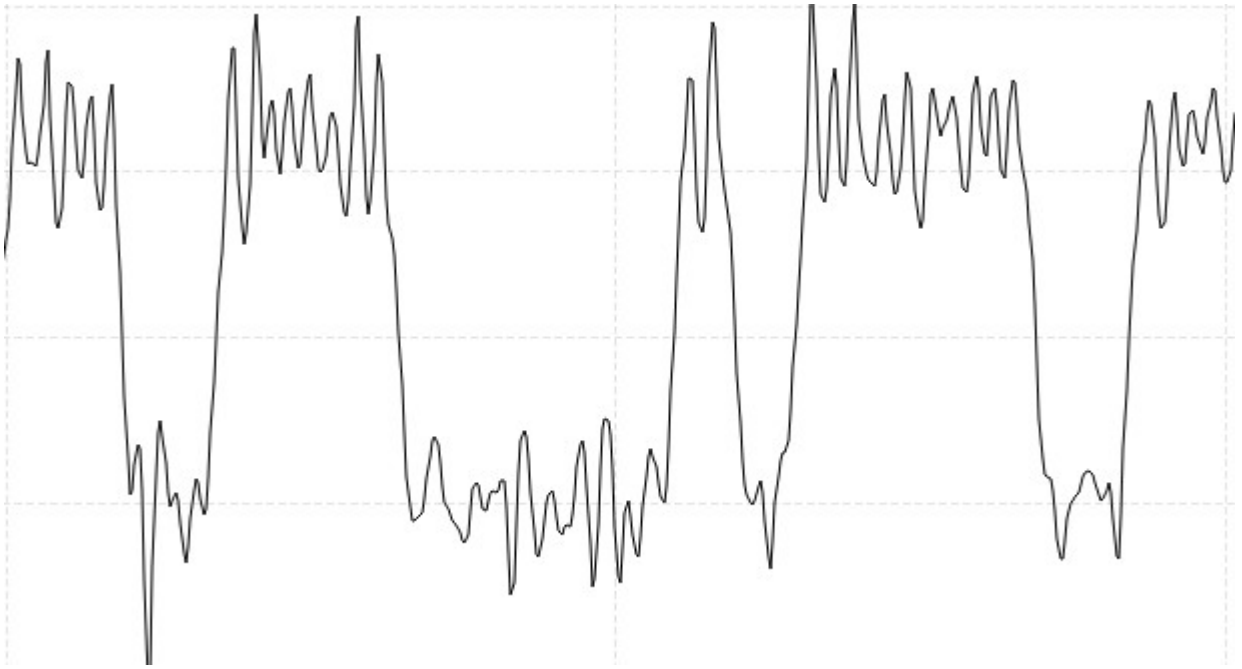
An example of binary FSK

https://en.wikipedia.org/wiki/Frequency-shift_keying

Clock recovery (1)

Sampling-rate (I/Q signal from receiver)
: 1200 samples / sec
Bitrate RTTY = 100 bits / sec.
(why?)

- 1 bit = 12 samples



Clock recovery (2)

Sampling-rate (I/Q signal from receiver)
: 1200 samples / sec
Bitrate RTTY = 100 bits / sec.
(why?)

- 1 bit = 12 samples

Clock recovery (3)

- Clock Recovery MM block
- Based on a PLL
- Requires “peaks” to well synchronize
- “dark magic”

Samples to bits

Comparator:

- >0 → “1”
- <0 → “0”

→ output uit GNURadio → UDP pakket

Outputting the bits (to baudot decoder)

Output from GNURadio to external python script

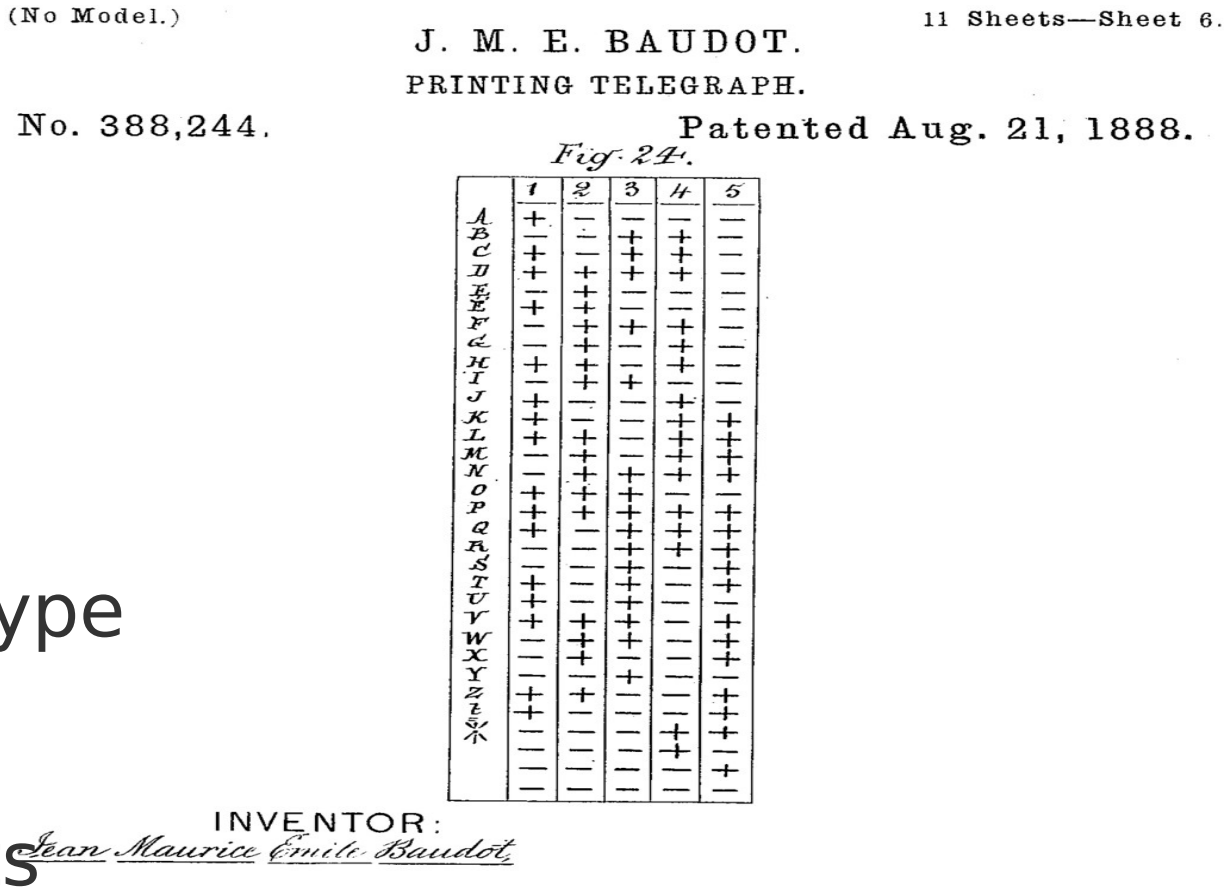
→ UDP pakket

This can probably also work with GNU Radio “messages”

Bits (dump)

01 01 01 01 01 01 01 00 00 01 01 00 00 00 01 01
01 01 01 01 01 01 00 00 01 01 00 00 00 01 01 01
01 00 00 01 01 01 01 01 01 00 00 00 01 01 01 01
00 00 01 01 01 01 01 01 00 00 00 01 01 00 00 00
00 00 00 00 00 00 00 00 00 00 01 01 00 00 00 00
01 01 01 01 00 00 00 00 00 01 01 00 00 01 01 01
01 01 01 01 01 00 00 00 01 01 00 00 01 01 00 00
01 01 01 01 00 00 00 01 01 01 01 01 01 01 01 01
01 00 00 00 00 00 01 01 01 01 00 00 01 01 01 01
00 00 00 00 00 01 01 00 00 00 00 01 01 00 00 00
00 00 00 00 01 01 01 01 01 01 00 00 00 00 00 00
00 00 00 01 01 00 00 00 00 00 00 00 00 00 00 00
00 00 01 01 00 00 00 00 00 00 00 00 01 01 00 00
00 01 01 01 01 01 01 01 01 00 00 00 00 00 00 00
01 01 01 01 00 00 01 01 00 00 01 01 00 00 00 01
01 00 00 01 01 00 00 01 01 01 01 00 00 00 01 01
00 00 00 00 01 01 00 00 00 00 00 00 00 01 01 00
00 01 01 00 00 00 00 00 00 00 00 00 01 01 00 00
00 00 00 00 00 00 00 00 00 00 00 01 01 00 00 01
01 00 00 01 01 01 01 00 00 00 01 01 01 01 01 01

Baudot Decoder



(No Model.) 11 Sheets—Sheet 6.

J. M. E. BAUDOT.

PRINTING TELEGRAPH.

No. 388,244. Patented Aug. 21, 1888.

Fig. 24.

	1	2	3	4	5
A	+	−	−	−	−
B	−	−	+	+	−
C	+	+	−	−	−
D	+	−	−	+	−
E	+	−	−	−	−
F	+	−	+	+	−
G	−	+	−	+	−
H	−	−	+	−	−
I	+	+	−	−	−
J	+	−	+	+	−
K	+	−	+	+	+
L	−	+	−	−	+
M	−	−	+	+	+
N	−	−	+	−	+
O	−	−	−	−	+
P	−	+	−	+	+
Q	+	+	−	+	+
R	−	+	−	−	+
S	+	−	+	−	−
T	−	−	−	+	−
U	+	+	−	−	−
V	−	+	+	+	−
W	+	−	−	−	+
X	+	+	+	−	+
Y	+	+	−	+	−
Z	+	−	+	−	+
↕	−	−	−	+	−

INVENTOR:

Jean Maurice Émile Baudot

- “telex” alphabet
- 5 bits

RTTY: RADIO teletype

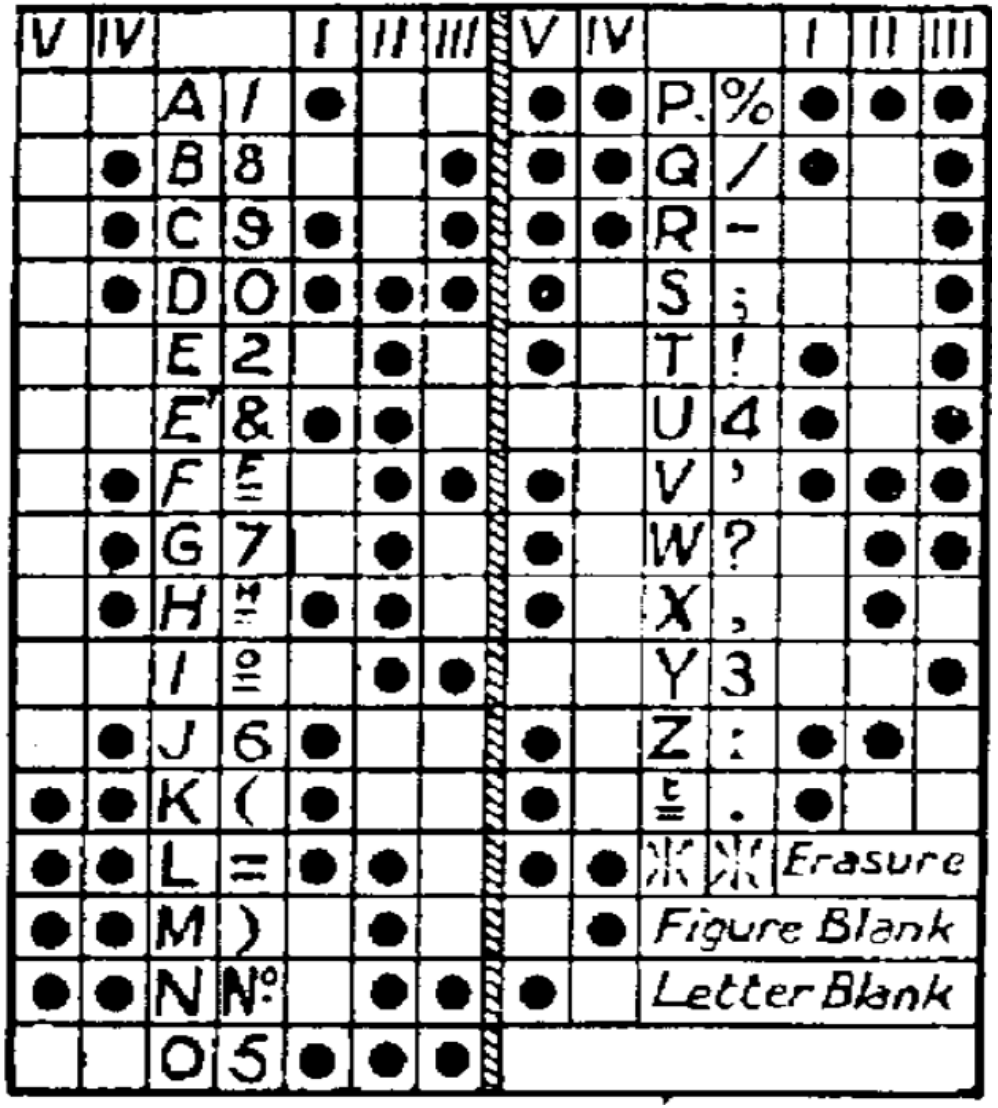
- 1 or 2 startbits
- 1, 1.5 or 2 stopbits

DWD: 50 baud, 1 startbit, 1.5 stopbits

→ demodulate FSK at 100 Baud

https://en.wikipedia.org/wiki/Baudot_code

Baudot Decoder



V	IV			I	II	III	V	IV			I	II	III
		A	/	●			●	●	P.	%	●	●	●
	●	B	8			●	●	●	Q	/	●		●
	●	C	9	●		●	●	●	R	-			●
	●	D	O	●	●	●	●		S	;			●
		E	2		●		●		T	!	●		●
		E	&	●	●				U	4	●		●
	●	F	≡		●	●	●		V	'	●	●	●
	●	G	7		●		●		W	?		●	●
	●	H	≠	●	●		●		X	,		●	
		I	≈		●	●			Y	3			●
	●	J	6	●			●		Z	:	●	●	
●	●	K	(	●			●		≤	.	●		
●	●	L	=	●	●		●	●	✳	✳	Erasure		
●	●	M	)		●			●	Figure Blank				
●	●	N	№		●	●	●		Letter Blank				
		O	5	●	●	●							

https://www.rtty.com/England/fiveunits.htm

Baudot Decoder (3)

Decode at double bitrate

So we are looking for ?

- 2 startbits (“1”)
- 5 * 2 databits (which should be equal)
- 3 stopbits (“0”)

S1	S2	D11	D12	D21	D22	D31	D32	D41	D42	D51	D52	E1	E2	E3
1	1	d1	d1	d2	d2	d3	d3	d4	d4	d5	d5	0	0	0

Baudot Decoder (3)

Take in 29 bits and search for the “baudot” pattern

Compare received bits with pattern, give ”score”

```
1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 1 1 0 0 0 0 0 1 1
1 1 a a b b c c d d e e 0 0 0                                   2 + 2 + 2
  1 1 a a b b c c d d e e 0 0 0                                 1 + 4 + 2
    1 1 a a b b c c d d e e 0 0 0                               ...
      1 1 a a b b c c d d e e 0 0 0                             ...
        1 1 a a b b c c d d e e 0 0 0                           ...
          1 1 a a b b c c d d e e 0 0 0                         ...
            1 1 a a b b c c d d e e 0 0 0                       ...
              1 1 a a b b c c d d e e 0 0 0                     2 + 5 + 3
                1 1 a a b b c c d d e e 0 0 0                   ...
(…)
                                    1 1 a a b b c c d d e e 0 0 0  ...
```

Once found, shift out bit-pattern with the highest score,
and fill up the buffer (up to 29 bits)

Baudot Decoder (4)

Look for character in the table, or switch to other table

Print character

Putting it all together

- Run GNU Radio flow on receiver PC
  - Demodulated bits → UDP packets (IP multicast)
- Additional python script:
  - tcp server (i.e. wait for incoming connections)
  - Per tcp session: baudot decoder

Baudot decoder enhancements (1)

- FSK demodulator at 200 bits/sec (instead of 100 bits/sec.)
  - 100 bits/sec: For every “baudot” bit → 2 bits received
    - Ex. 3 bits received (010):

```
a1 a2 : b1 b2 : c1 c2
0  0    1  1    0  0
```

  - 200 bits/sec: For every “baudot” bit → 4 bits received
    - Ex. 3 bits received (10):

```
a1 a2 a3 a4 : b1 b2 b3 b4 : c1 c2 c3 c4
0  0  0  0    1  1  1  1    0  0  0  0
```

Baudot decoder enhancements (1b)

- Ex. 3 bits received:

```
a1 a2 a3 a4 : b1 b2 b3 b4 : c1 c2 c3 c4
0  0  0  0    1  1  1  1    0  0  0  0
```

- Ignore bits “1” and “4” (adjacent to previous / next bits). Only look at bits 2 and 3.
- Should help against “inter-symbol-interference

  Q: but is that the real problem?
- Q: will the clock recovery block be able to handle this?

Baudot decoder enhancements (2)

- DWD transmissions contain a lot of text, a lot of very simular text

- “SEEWET?…”
  Next char? “U” (1010**0**) or “T” (1010**1**)
- Machine learning:
  - Make lists, what character is the most common after “WET”?
  - Based on texts that are known to be correct (100 % “baudot pattern” match)

Question time (2)

Questions?

GNU Radio … summary (1)

Use:

- Practical applications:
  - Less known protocols:
    - Satellite telemetry decoder
  - Simulations
    - Radio Channel simulations
- Experimenting / learning receivers/decoders – encoders/transmitters
- Educational tool

GNU Radio … summary (2)

Use:

- Advanced telecom techniques
  - e.g. spacial diversity reception
- Cybersecurity

GNU Radio … summary (3)

- Already a lot of blocks ready to use
  - But what block does what?
  - What do all these parameters do?
- Relative easy to create your own blocks
- But you must understand signal-processing techniques

GNU Radio … Getting started (1)

- Main development environment is linux
  - apt / yum / …
- Mac: brew, MacPorts
- Windows:
  - No idea (no personal experience)
  - Run GNU Radio in Linux VM
    - Pentoo linux: GNU Radio pre-installed
  - Use single-board computer (raspberry-pi, …)

GNU Radio … Getting started (2)

- Tutorials
  - Articles on GNU Radio website
  - Videos
- SDR knowledge
  - See resources list
- “sdrbelgium” mailinglist
- Workshop ???

Question time (final)

Questions?