Niskopoziomowa analiza sygnałów radiowych

Artur „Lacky” Łącki
lackylab.pl
alacki93@gmail.com

- Zawodowo inżynier systemów wbudowanych i analityk podatności.
- Nie zajmuję się zawodowo radiokomunikacją. Jest to wyłącznie hobby.
- Uwagi do moich obserwacji/wniosków mile widziane.
- Pytania najlepiej zadawać w trakcie prezentacji.
- Część rzeczy jest omówione ogólnie ze względu na ograniczenia czasowe.

Czemu radio?

- Bo można.
- Radiokomunikacja to ciekawy kawał wiedzy technicznej.
- Można dorabiać „interfejsy” do taniej elektroniki radiowej.
- Bo SDR jest względnie tani.
- Bo można.

Pogoda za oknem

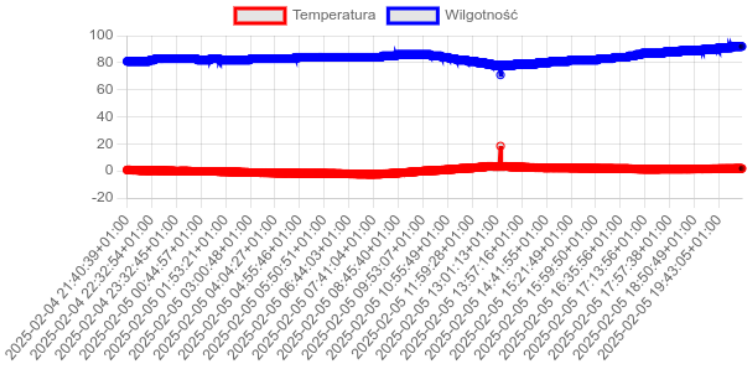
Czas ostatniego pomiaru: 2025-02-05 20:40:05+01:00

Temperatura: 1.8°C

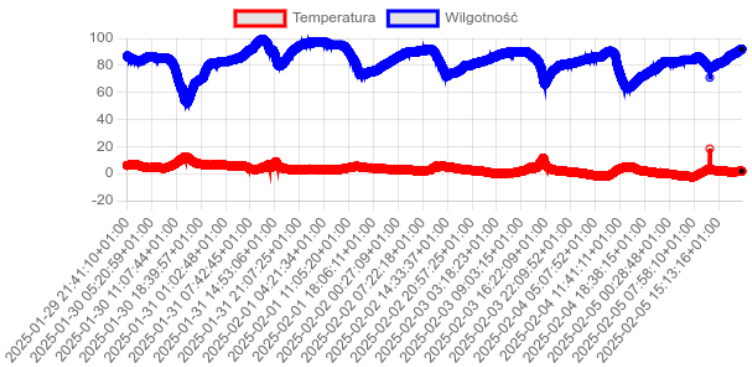
Wilgotność: 92%

Bateria: OK

Ostatnia doba



Ostatni tydzień



ESP8266 + tani (<10zł) moduł radiowy na 433,92MHz

Cel

- Stworzenie programowego odbiornika do pilota od taniego alarmu.
- Nauka GNU Radio.
- Pilot sterował alarmem w starym Seicento (niech mu korozja lekką będzie [*]).
- Transmisja radiowa na bazie układu scalonego PT2262.


ARM
DISARM
Cheetah


PT2262
PTC
0139Z

PT2262

DESCRIPTION

PT2262 is a remote control encoder paired with PT2272 utilizing CMOS Technology. It encodes data and address pins into a serial coded waveform suitable for RF or IR modulation. PT2262 has a maximum of 12 bits of tri-state address pins providing up to 531,441 (or 312) address codes; thereby, drastically reducing any code collision and unauthorized code scanning possibilities.

FEATURES

- CMOS Technology
- Low Power Consumption
- Very High Noise Immunity
- Up to 12 Tri-State Code Address Pins
- Up to 6 Data Pins
- Wide Range of Operating Voltage: Vcc = 4 ~ 15V
- Single Resistor Oscillator
- Latch or Momentary Output Type
- Available in DIP and SOP

APPLICATIONS

- Car Security System
- Garage Door Controller
- Remote Control Fan
- Home Security/Automation System
- Remote Control Toys
- Remote Control for Industrial Use

↑ Brak kodów dynamicznych

PT2262 - transmisja

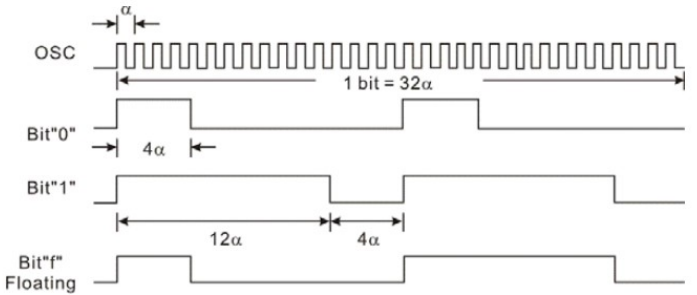
Trójstanowa logika →

CODE BITS

A Code Bit is the basic component of the encoded waveform, and can be classified as either an AD (Address/Data) Bit or a SYNC (Synchronous) Bit.

ADDRESS/DATA (AD) BIT WAVEFORM

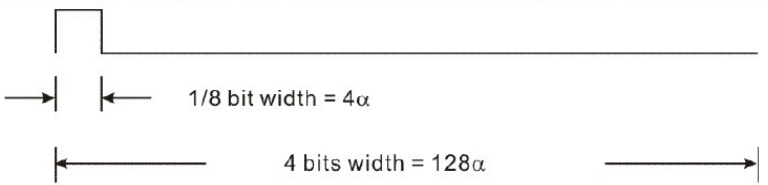
An AD Bit can be designated as Bit “0”, “1” or “f” if it is in low, high or floating state respectively. One bit waveform consists of 2 pulse cycles. Each pulse cycle has 16 oscillating time periods. For further details, please refer to the diagram below:



where: α = Oscillating Clock Period

SYNCHRONOUS (SYNC.) BIT WAVEFORM

The Synchronous Bit Waveform is 4 bits long with 1/8 bit width pulse. Please refer to the diagram below:



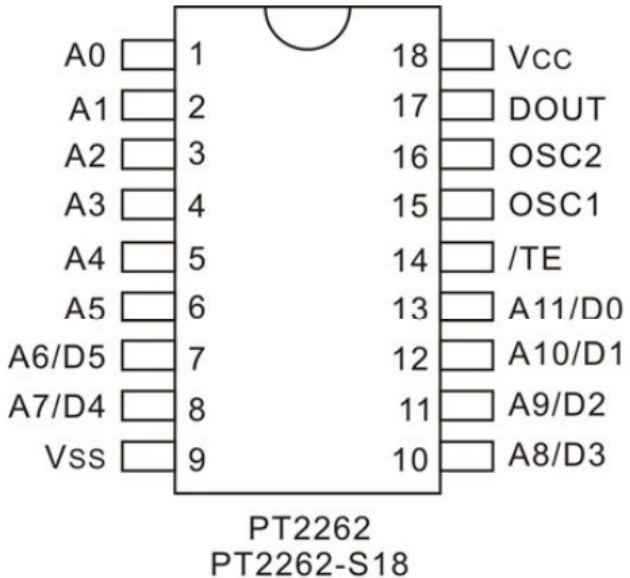
Note: 1 bit = 32α

PT2262 - transmisja

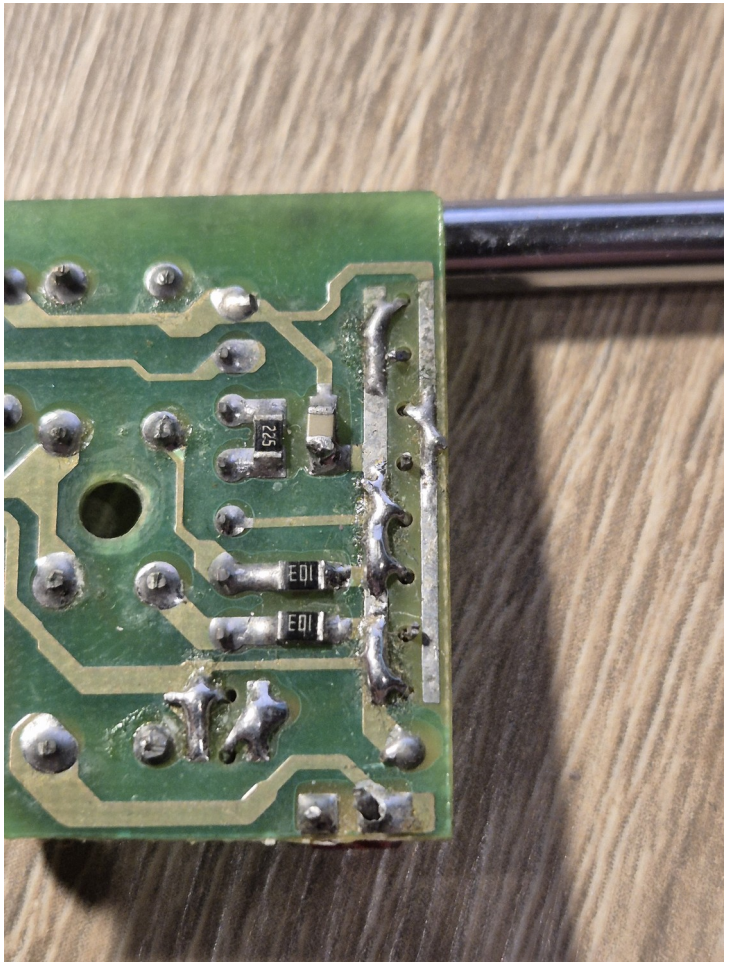
A0	A1	A2	A3	A4	A5	A6/D5	A7/D4	A8/D3	A9/D2	A10/D1	A11/D0	SYNC. Bit

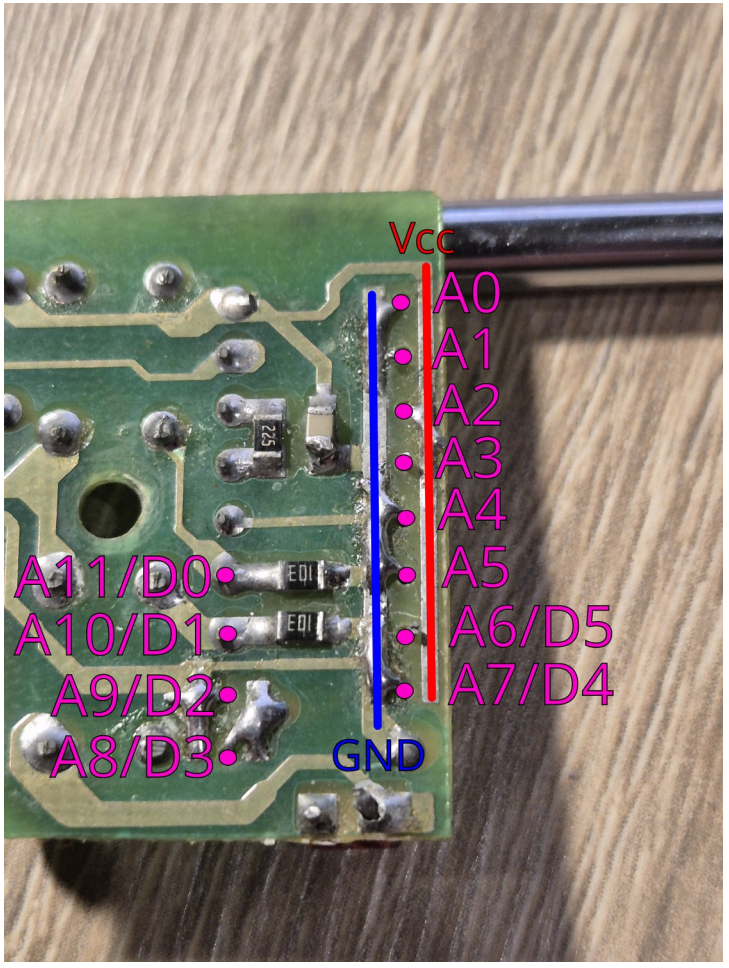
One Complete Code Word

0 Data:	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	Sync Bit
1 Data:	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	D0	Sync Bit
2 Data	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	D1	D0	Sync.Bit
3 Data	A0	A1	A2	A3	A4	A5	A6	A7	A8	D2	D1	D0	Sync.Bit
4 Data	A0	A1	A2	A3	A4	A5	A6	A7	D3	D2	D1	D0	Sync.Bit
5 Data	A0	A1	A2	A3	A4	A5	A6	D4	D3	D2	D1	D0	Sync.Bit
6 Data	A0	A1	A2	A3	A4	A5	D5	D4	D3	D2	D1	D0	Sync.Bit



Programowanie pilota

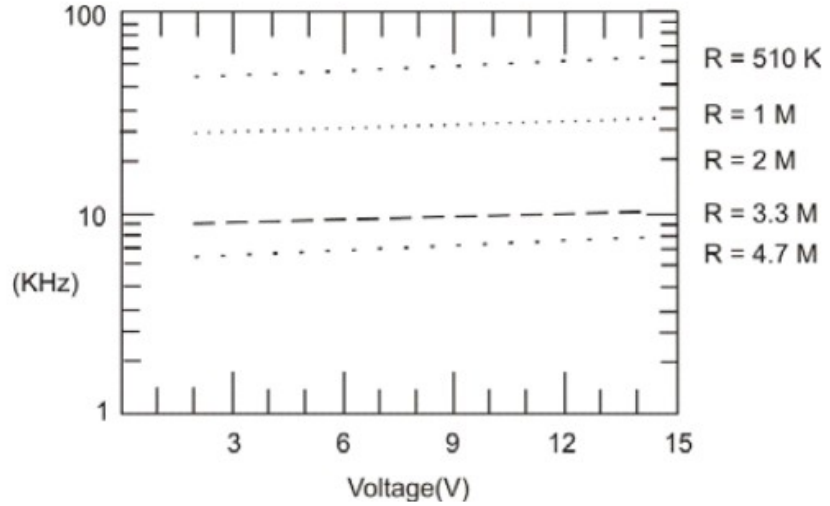




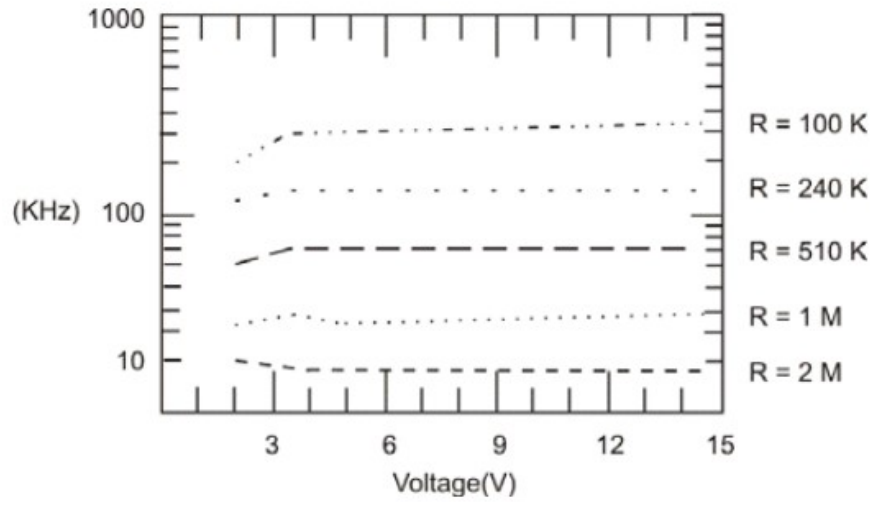
Kod pilota:
0212002010xx

Programowanie pilota

Encoder OSC Frequency



Decoder OSC Frequency



Suggested oscillator resistor values are shown below.

PT2262	PT2272
4.7 M	820 K*
3.3 M	680 K*
1.2 M	200 K**

Note:
* -- Operates when PT2272's Vcc=5V to 15V
** -- Operates when PT2272's Vcc=3V to 15V

Nagranie transmisji

W większości wypadków domowe urządzenia radiowe korzystają z pasm ISM (Industrial, Scientific, Medical), które są nielicencjonowane.

Typowe częstotliwości:

- 315MHz
- 433,92MHz
- 868MHz
- 2,4GHz

Nagranie transmisji

Universal Radio Hacker (URH) - https://github.com/jopohl/urh

Słowo o zapisie

- *.complex16s – surowy zapis danych z SDR.
- W praktyce binarny zapis liczb zespolonych: I (część rzeczywista) i Q (część urojona).
- Więcej o próbkowaniu IQ https://pysdr.org/content/sampling.html
- Upewnijcie się jak wg twórców danej aplikacji wygląda dwubajtowa próbka IQ.

URH:

3.1 Importing a signal

Apart from recording, a signal can be added to Interpretation tab via File → Open. File ending determines how URH handles the signal. URH understands these file endings:

- `.complex` files with complex64 samples (32 Bit float for I and Q, respectively). This is the default signal file format and will also be used in case the file has no ending at all.
- `.complex16u` using two unsigned 8 Bit integers for I and Q
- `.complex16s` using two signed 8 Bit integers for I and Q
- `.wav` files can be imported, but must not be compressed, i.e., they should be PCM.
- `.csv` e.g. from USB oscilloscope can be imported using a CSV wizard available with File → Import → IQ samples from csv.

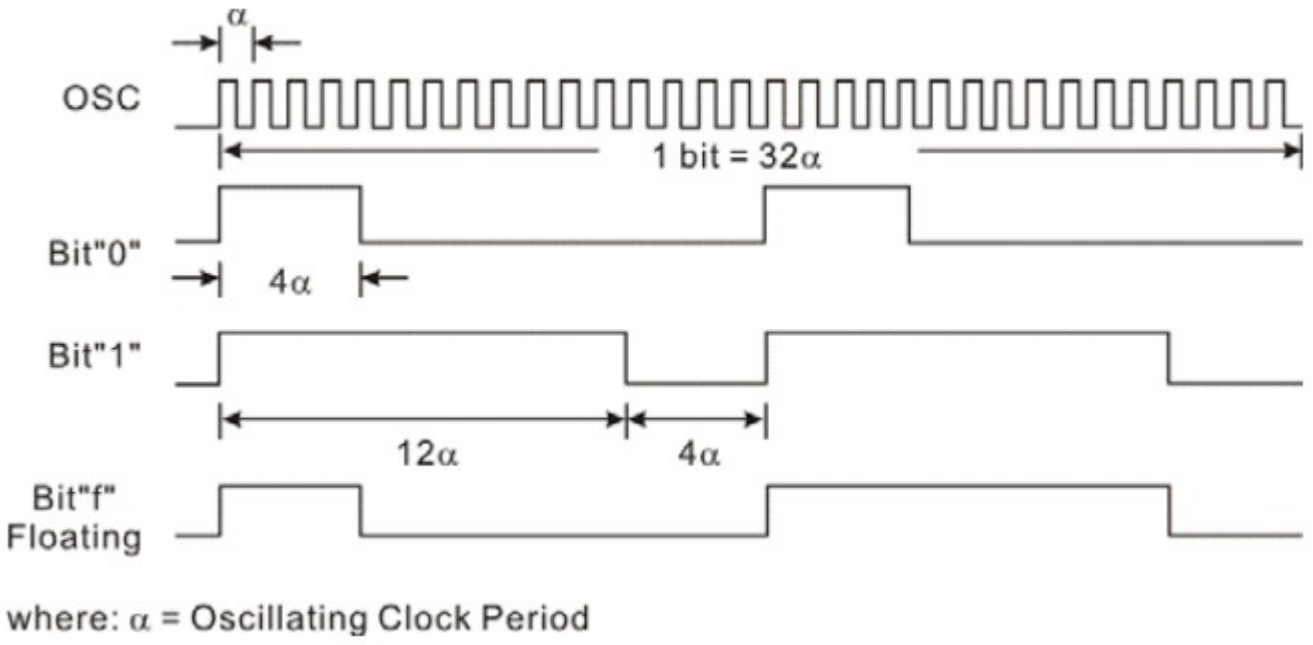
GNU Radio:

Complex samples stored as interleaved 16-bit integers:

```
[ I sample 0: 16 bit int ][ Q sample 0: 16 bit int ][ I sample 1: 16 bit int ][ Q sample 1: 16 bit int ][ I samp
```

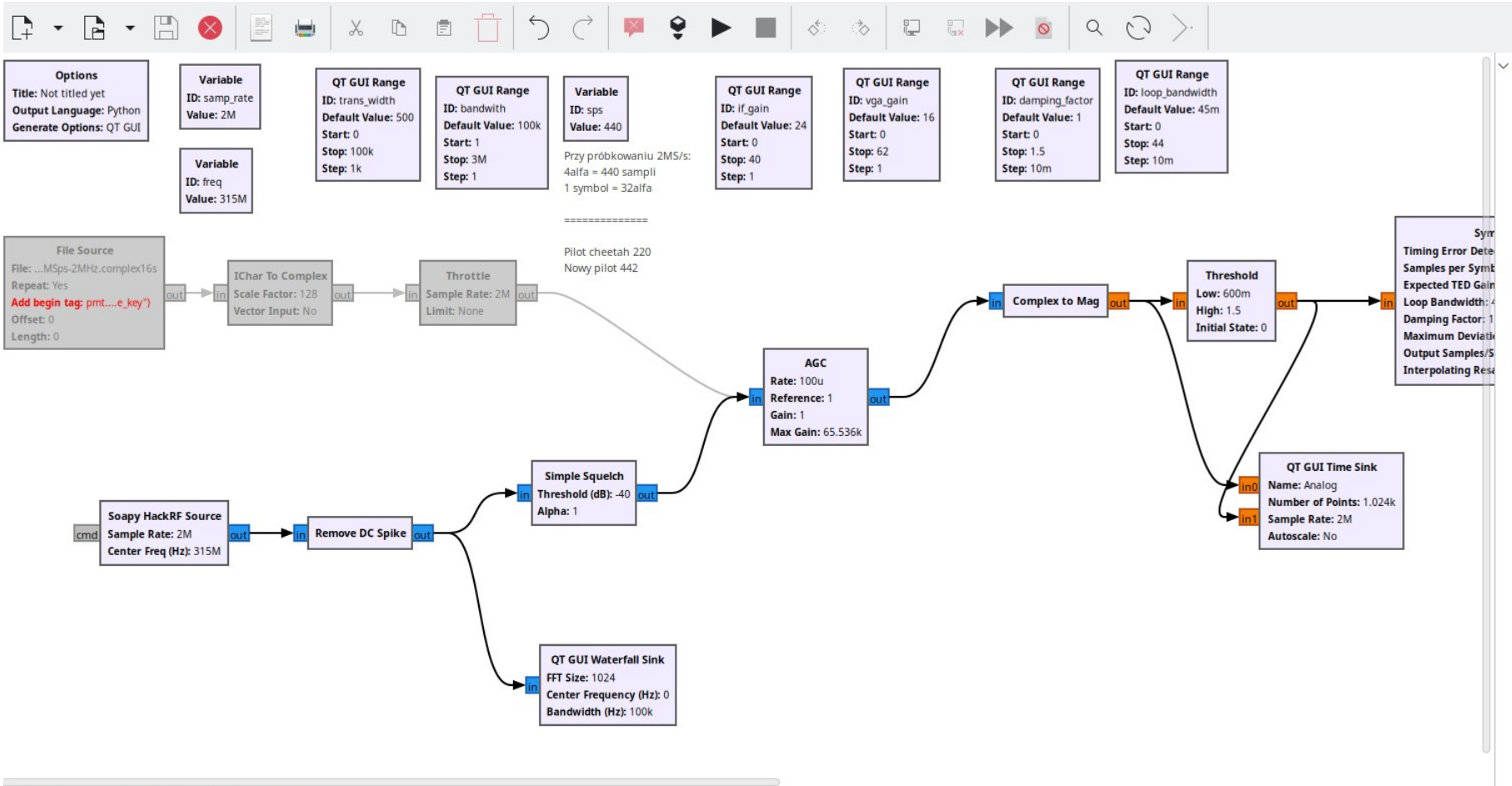
Dekodowanie

- Bit 0 – 10001000
- Bit 1 – 11101110
- Bit f (2) – 10001110



GNU Radio

- Program do DSP (nie tylko SDR).
- Nie bierze jeńców ;_;
- Nierówny poziom dokumentacji.
- Nawet jeżeli jest dokumentacja, to może zakładać, że znacie teorię z DSP i nie będzie jej tłumaczyć.



<< Welcome to GNU Radio Companion 3.10.9.2 >>

lock paths:
/usr/share/gnuradio/grc/blocks

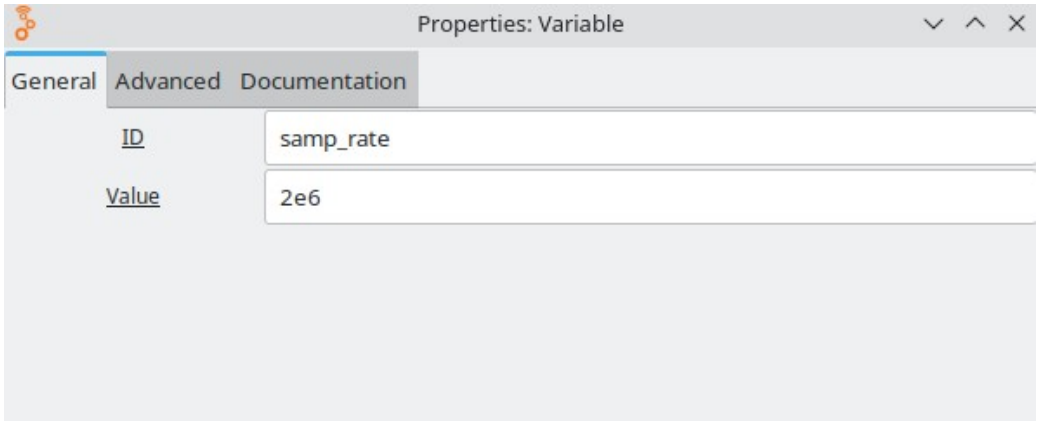
ID	Value
Imports	
Variables	

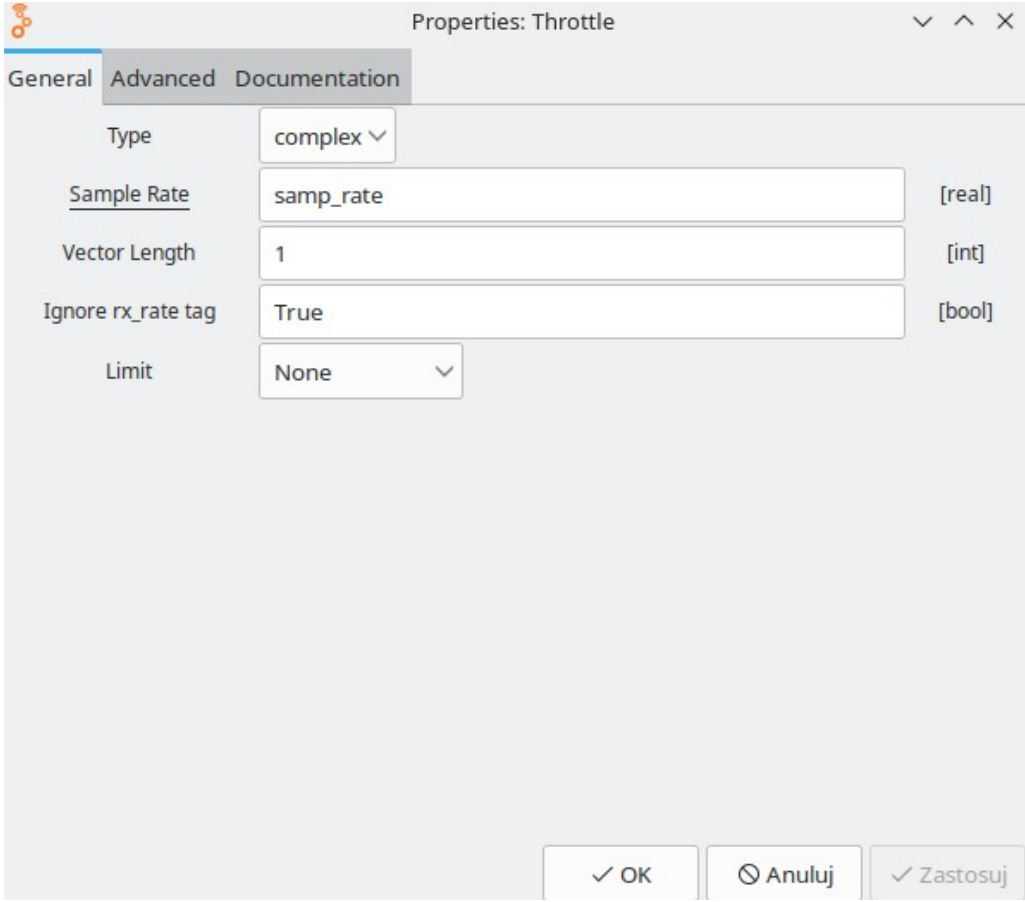
- Core
  - Audio
  - Boolean Operators
  - Byte Operators
  - Channel Models
  - Channelizers
  - Coding
  - Control Port
  - Debug Tools
  - Deprecated
  - Digital Television
  - Equalizers
  - Error Coding
  - File Operators
  - Filters
  - Fourier Analysis
  - GUI Widgets
  - Impairment Models
  - Industrial I/O
  - Instrumentation
  - IQ Correction
  - Level Controllers
  - Math Operators
  - Measurement Tools
  - Message Tools
  - Misc
  - Modulators
  - Networking Tools
  - OFDM
  - Packet Operators
  - PDU Tools
  - Peak Detectors
  - Resamplers

Zmienne

Variable
ID: samp_rate
Value: 2M

Variable
ID: freq
Value: 315M

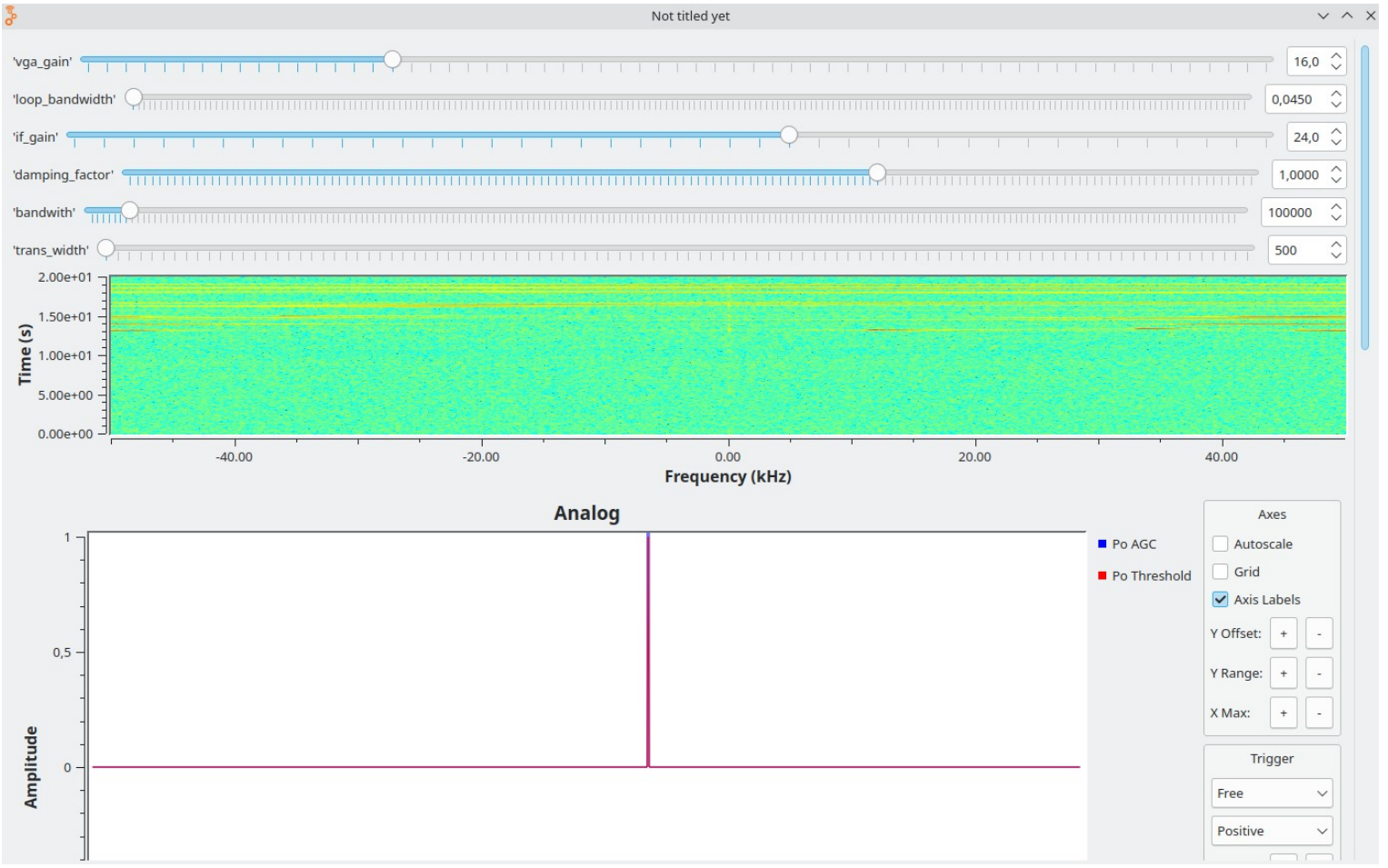




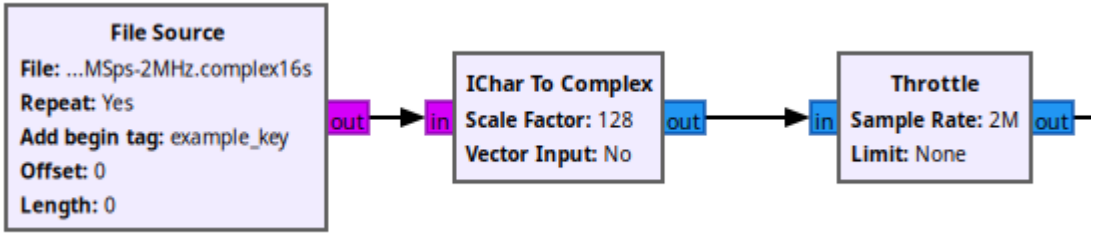
Wizualizacja

QT GUI Range
ID: if_gain
Default Value: 24
Start: 0
Stop: 40
Step: 1

QT GUI Range
ID: vga_gain
Default Value: 16
Start: 0
Stop: 62
Step: 1

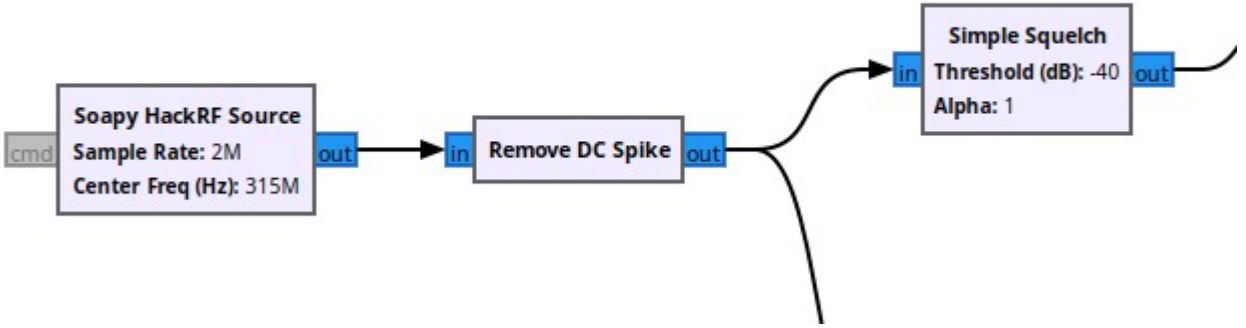


Źródło sygnału - plik

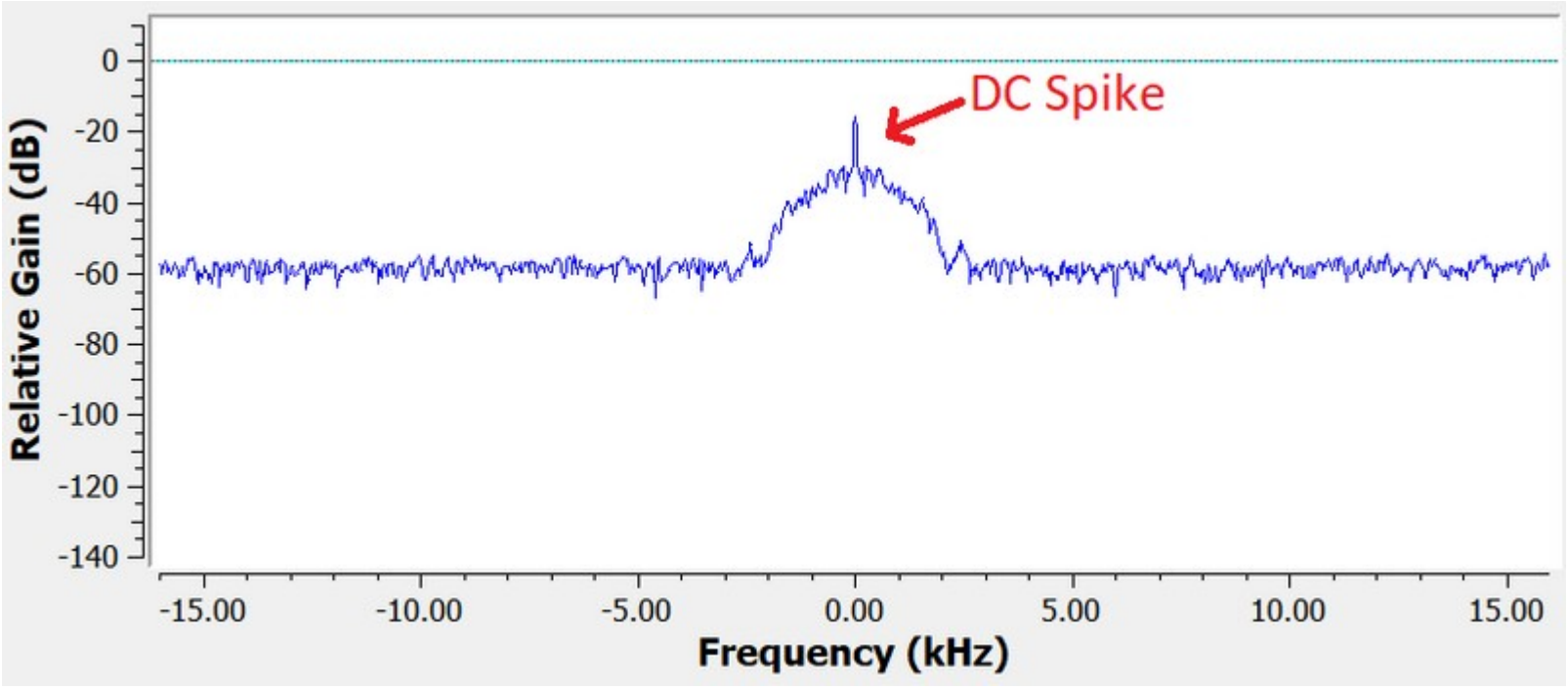


- **File source** – Ścieżka do pliku, informacje o strukturze, opcjonalne tagowanie itd.
- **IChar To Complex** – Konwersja próbek IQ na potok danych zmiennoprzecinkowych.
- **Throttle** – Ograniczenie częstotliwości próbkowania.

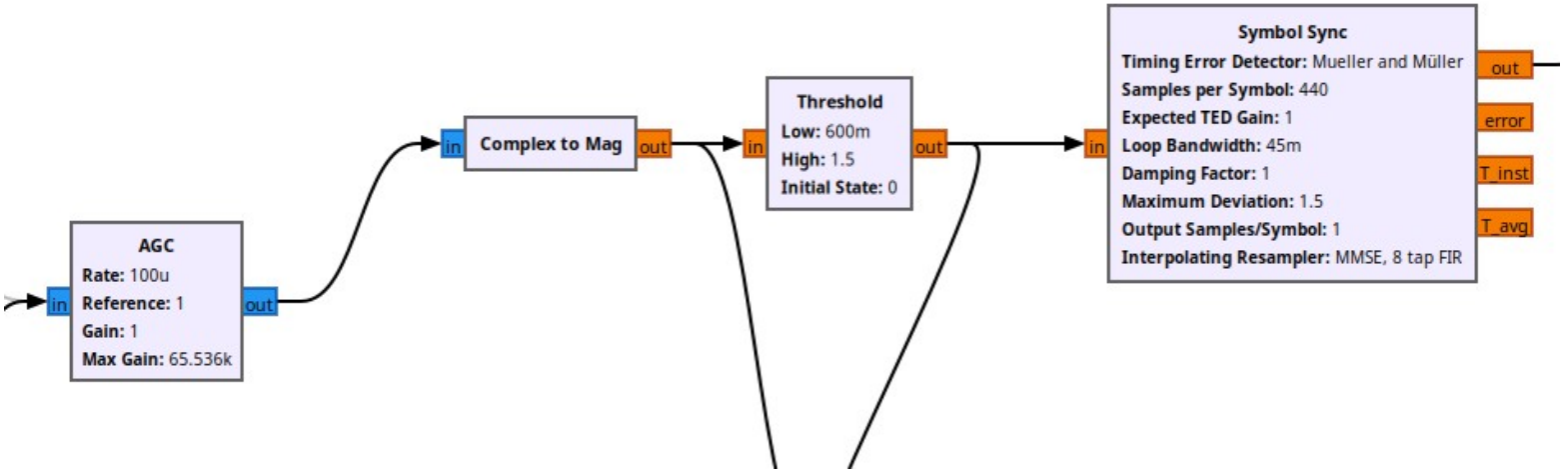
Źródło sygnału - SDR



- **Soapy HackRF Source** – Wykorzystaj API Soapy do komunikacji z SDR https://github.com/pothosware/SoapySDR
- **Remove DC Spike** – Usuwa zakłócenia mające charakter napięcia stałego (DC)
- **Simple Squelch** – Wyciszanie szumów i zakłóceń.

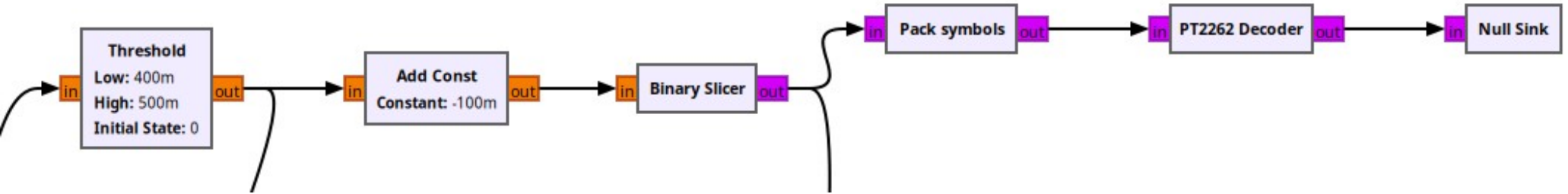

DC Spike
Relative Gain (dB)
0
-20
-40
-60
-80
-100
-120
-140
-15.00
-10.00
-5.00
0.00
5.00
10.00
15.00
Frequency (kHz)

Odzyskiwanie symboli



- **AGC –** Automatyczna kontrola wzmocnienia.
- **Complex to Mag** – Obliczanie modułu liczby zespolonej. W praktyce: $\sqrt{I^2+Q^2}$
- **Threshold** – Ustawianie progu sygnału.
- **Symbol Sync** – Synchronizacja symboli. Pozwala określić początek i koniec danego symbolu z uwzględnieniem przesunięć czasowych i zmianie długości. Słowo klucz: TED (Timing Error Detector).

Końcowe przetwarzanie



- **Add Const** – Dodaj (odejmij) stałą do sygnału.
- **Binary Slicer** - Zamiana sygnału analogowego na pojedyncze bity. Wartości wejściowe **poniżej** zera są konwertowane na bity o wartości 0.
- **Pack symbols** – Skrypt Pythona pakujący pojedyncze bity do symboli.
- **PT2262 Decoder –** Skrypt Pythona. Parsowanie symboli.

Własne skrypty

Podstawowe zasady:

- Wasz blok (skrypt) działa cały czas podobnie jak inne bloki.
- *__init__* jest wykonywany tylko raz, przy starcie przetwarzania.
- *work* jest wykonywany zawsze gdy pojawią się nowe dane.
- Używajcie NumPy gdzie tylko możecie (zwłaszcza zamiast pętli).
- GNU Radio daje wam gotowe bufory na dane wejściowe (input_items) i dane wyjściowe (output_items).
- Rozmiary tych buforów mogą być różne przy każdym uruchomieniu *work*.
- Funkcja *print* może skutecznie zawiesić cały graf.

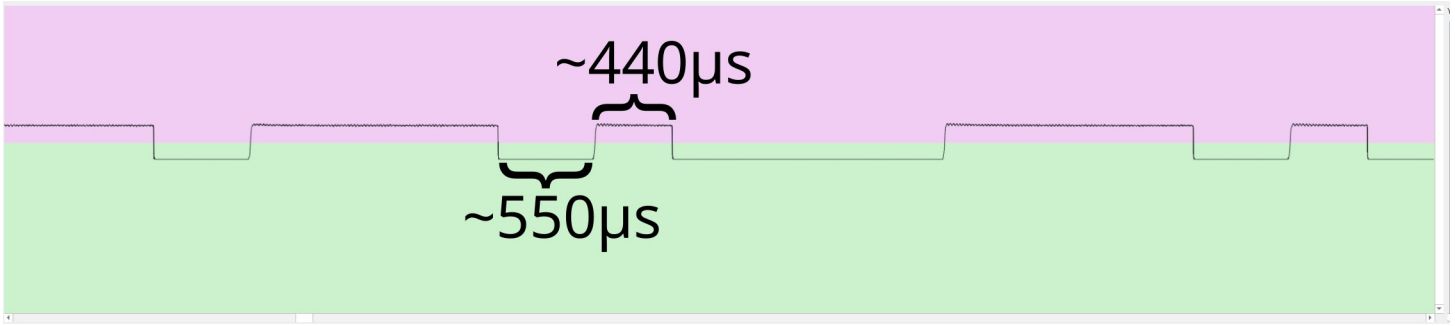
```
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, example_param=1.0):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Embedded Python Block',   # will show up in GRC
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).
        self.example_param = example_param

    def work(self, input_items, output_items):
        """example: multiply with constant"""
        output_items[0][:] = input_items[0] * self.example_param
        return len(output_items[0])

```

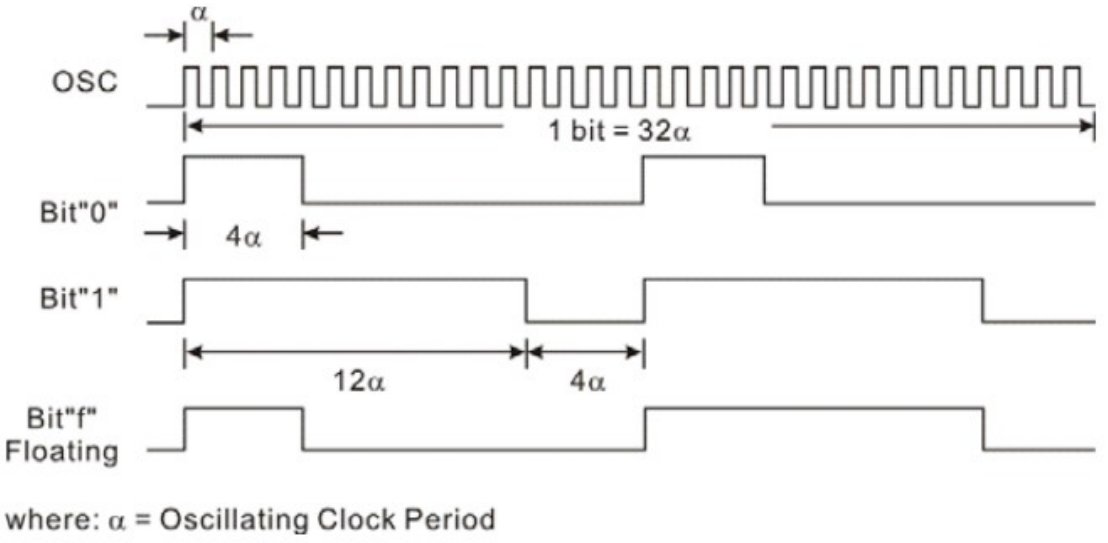
Demo

Inny pilot

- Pilot kompatybilny z PT2262…
- ...ale w środku zamiast PT2262 siedzi SC2262.
- Wg dokumentacji układ jest w pełni kompatybilny…
- ...chyba, że jednak nie.

Inny pilot


~440µs
~550µs


α
OSC
1 bit = 32α
Bit"0"
4α
Bit"1"
12α
4α
Bit"f"
Floating
where: α = Oscillating Clock Period

Co dalej

- Dalsze uzupełnianie braków wiedzy teoretycznej.
- Poprawa kompatybilności z dwoma pilotami.
- Stworzenie aplikacji niezależnej od GNU Radio.
- Być może LuaRadio https://luaradio.io/

Pytania?