OFFZONE 2025

Hack The Air

Как разговаривать с воздухом не привлекая внимания санитаров

Сергей Андреев

Assume Birch Team

@rc_manarag

Whoami

Радист, Сетевик, ИБ-шник

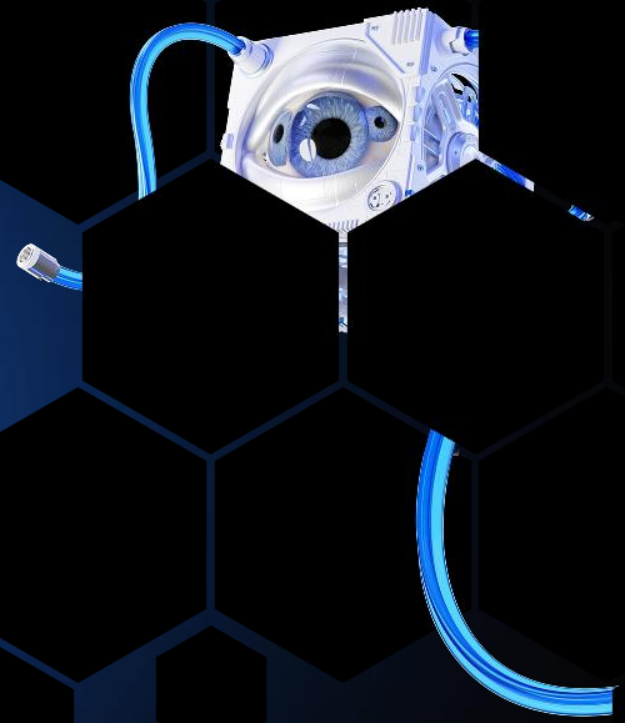
- ВВУЗ: SIGINT, COMINT, ELINT
- Admin → Эксп. Сетевик
- Sec. Assessment (ICS)

Assume Birch

- Attack
- Defense
- Research
- Party

Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A



Дисклеймер #1

Служба радиоконтроля не дремлет, поэтому не стоит забывать: слушать можно, передавать нельзя

Дисклеймер #2

Это не курс лекций, а тайм-слот всего 45 мин

Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A

Мотивация
Духовная

Развитие интереса

Мир беспроводной связи – это не только Wi-Fi, Bluetooth, LTE/GSM, Zigbee – он более разнообразный и интересный

Дать общее представление

О том, с чего начать и как прийти к цели

Покрыть непокрытое

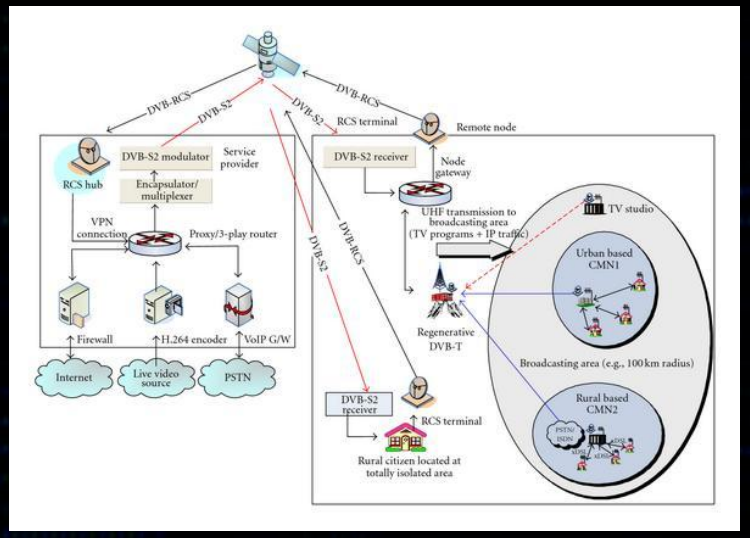
Много беспроводных устройств с малоизвестными протоколами проходит под радарами сообщества

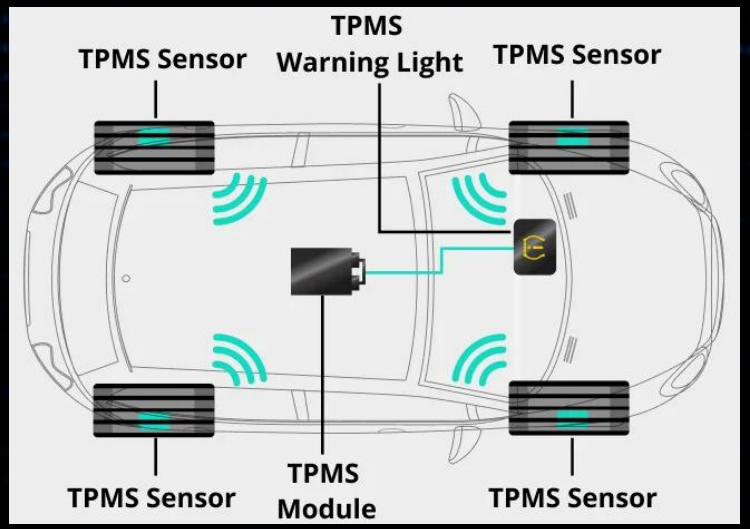
Сделать мир безопаснее

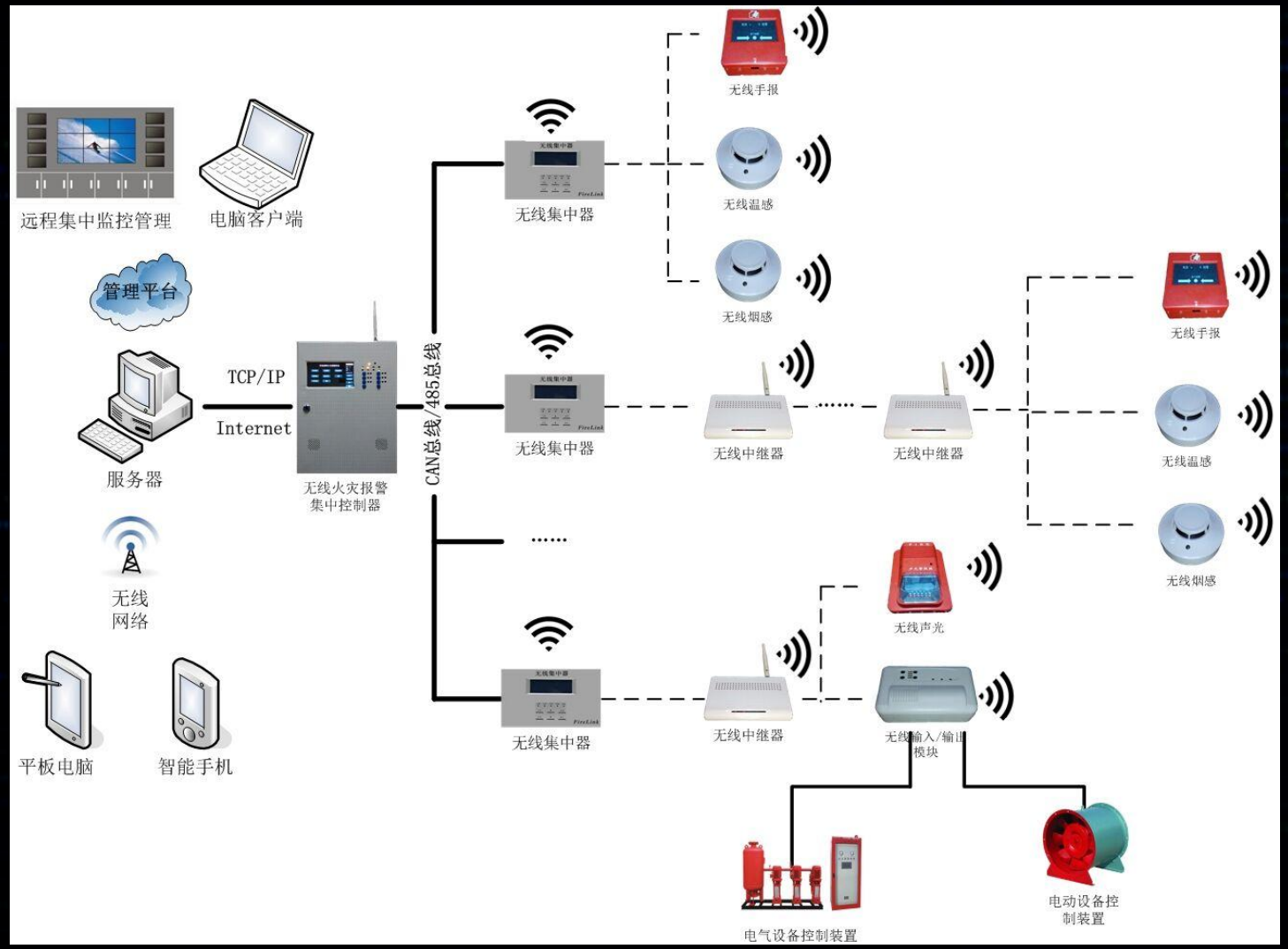
Как результат предыдущих шагов и долгосрочная цель

Мотивация

Материальная







Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A

Как общаться с «пациентом»

Необходимые примитивы

Получать данные (PDU) из RF

Возможность анализа коммуникаций выше уровней реализованных в RF, доступ к чувствительным данным

Отправлять данные (PDU) в RF

Возможность воздействовать на исследуемое устройство и эксплуатировать уязвимости в обработчиках PDU

Управлять стеком

Настройка адресации, шифрования и других параметров канального и сетевого уровней стека

Читать и обрабатывать поля всех уровней стека

Возможность анализа протокольной логики уровней, реализованных в RF

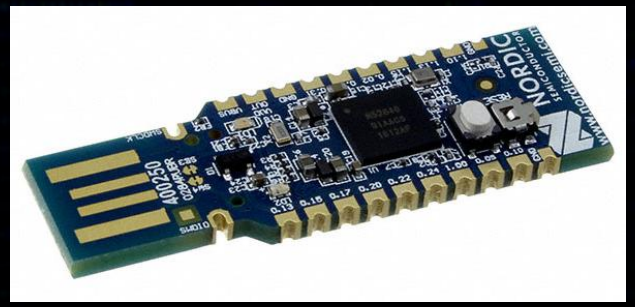
Генерировать кадры произвольного формата

Возможность реализации атак на уровни протокола, реализованные в RF

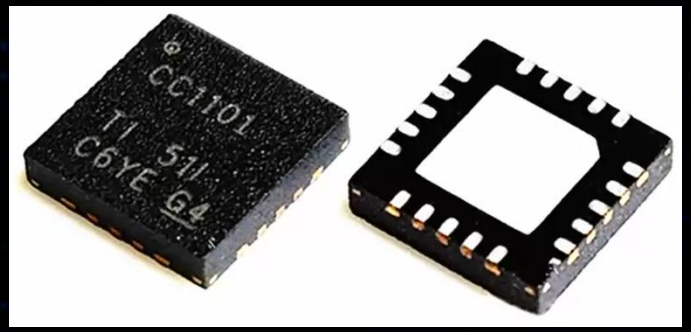
Как общаться с «пациентом»

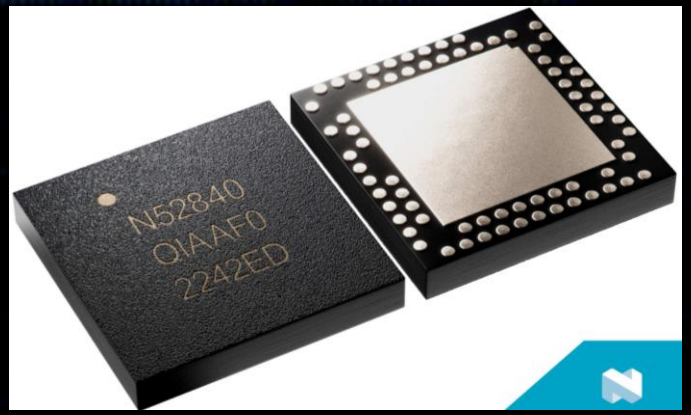
Оборудование

Module



Chip

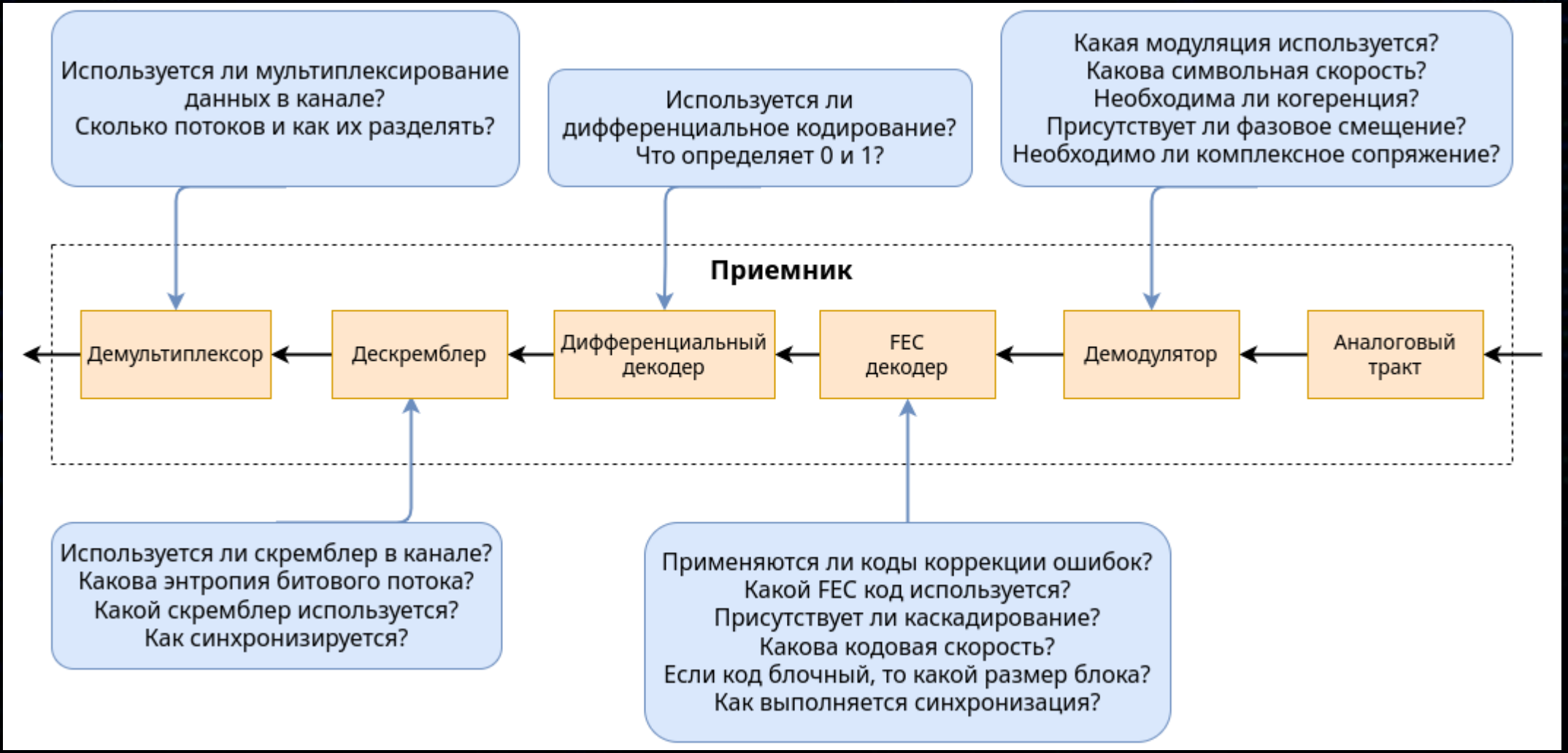




SDR

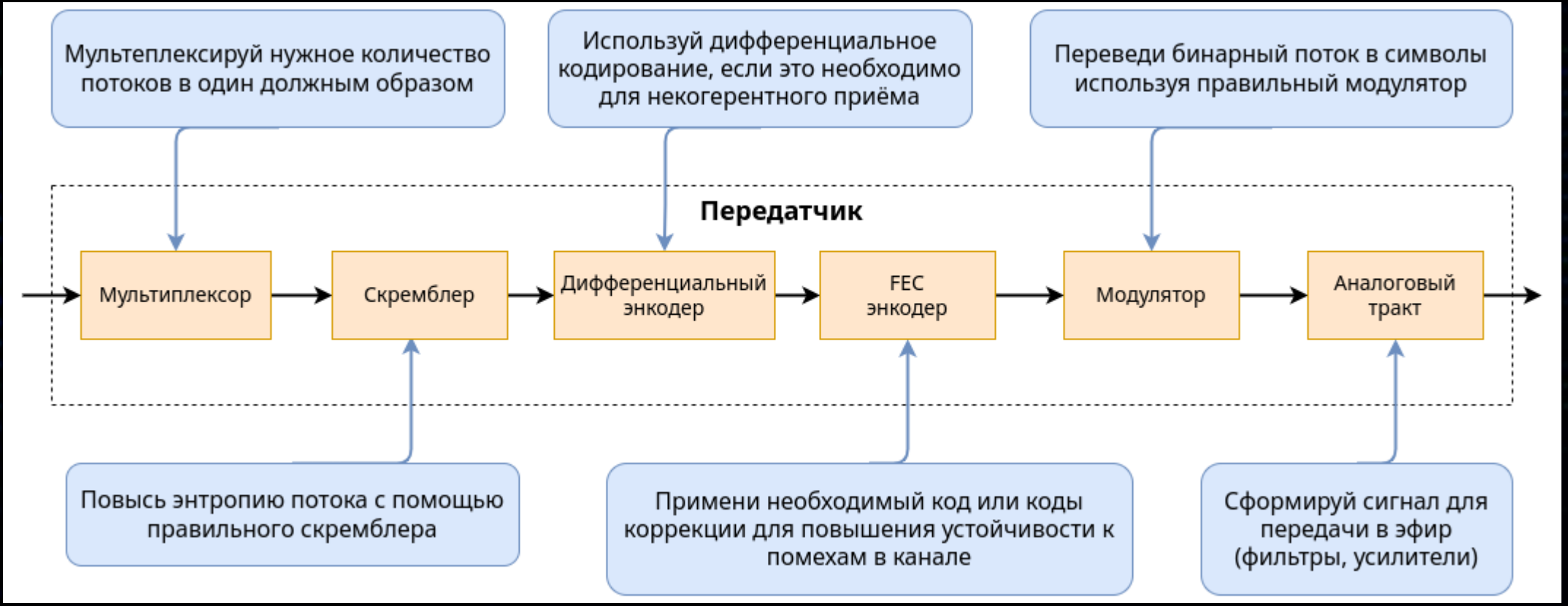
Как общаться с «пациентом»

Приёмный тракт - Что нужно знать для приёма?



Как общаться с «пациентом»

Передающий тракт – Обратный порядок всех преобразований



https://static.squarespace.com/static/543ae9afe4b0c3b808d72acd/543aee20e4b09162d086341f/543aee20e4b09162d086342e/1353278929018/Seeber-BlindSignalAnalysis.pdf

Как общаться с «пациентом»

Методы сбора параметров

Документация_

- Документация на устройство и маркировка
- Публичные документы по сертификации (FCC ID и др.)
- Документация на чип

Технические методы_

- Перехват инициализации логическим анализатором
- Чтение конфигурационных регистров
- Реверс прошивки SoC[1]
- Технический анализ

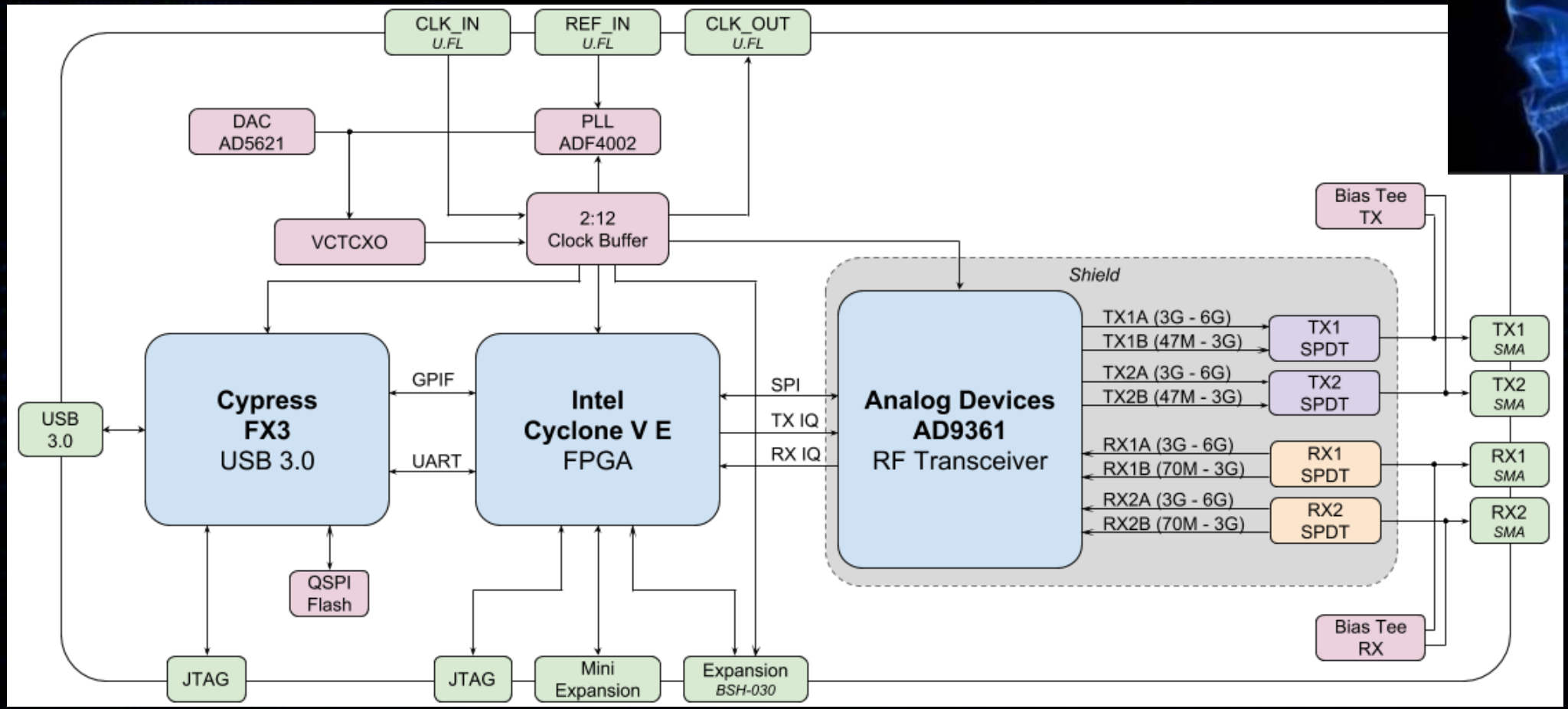
1: https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html

Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A

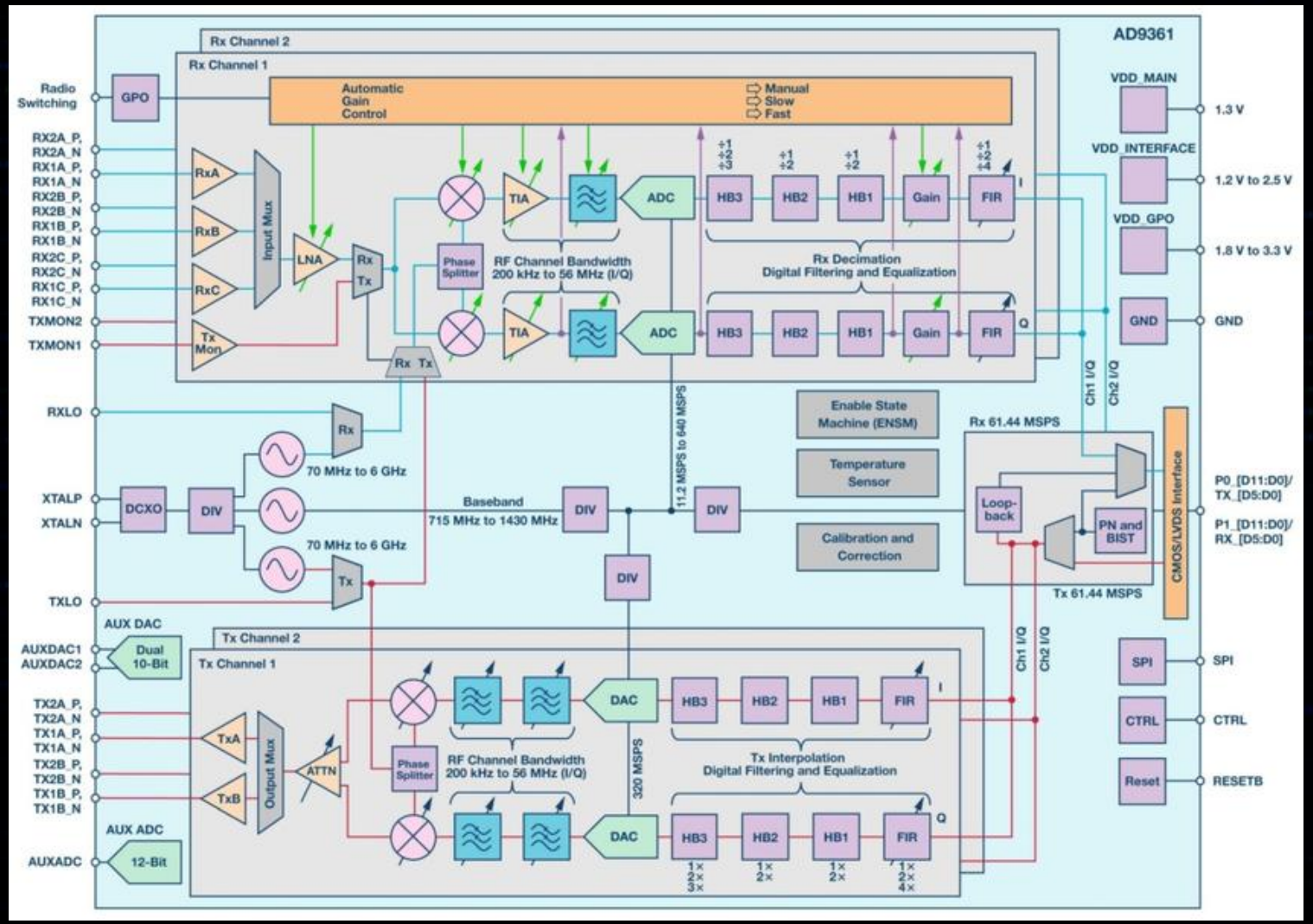
Теория: SDR, DSP, Signals

SDR – На примере bladeRF 2.0



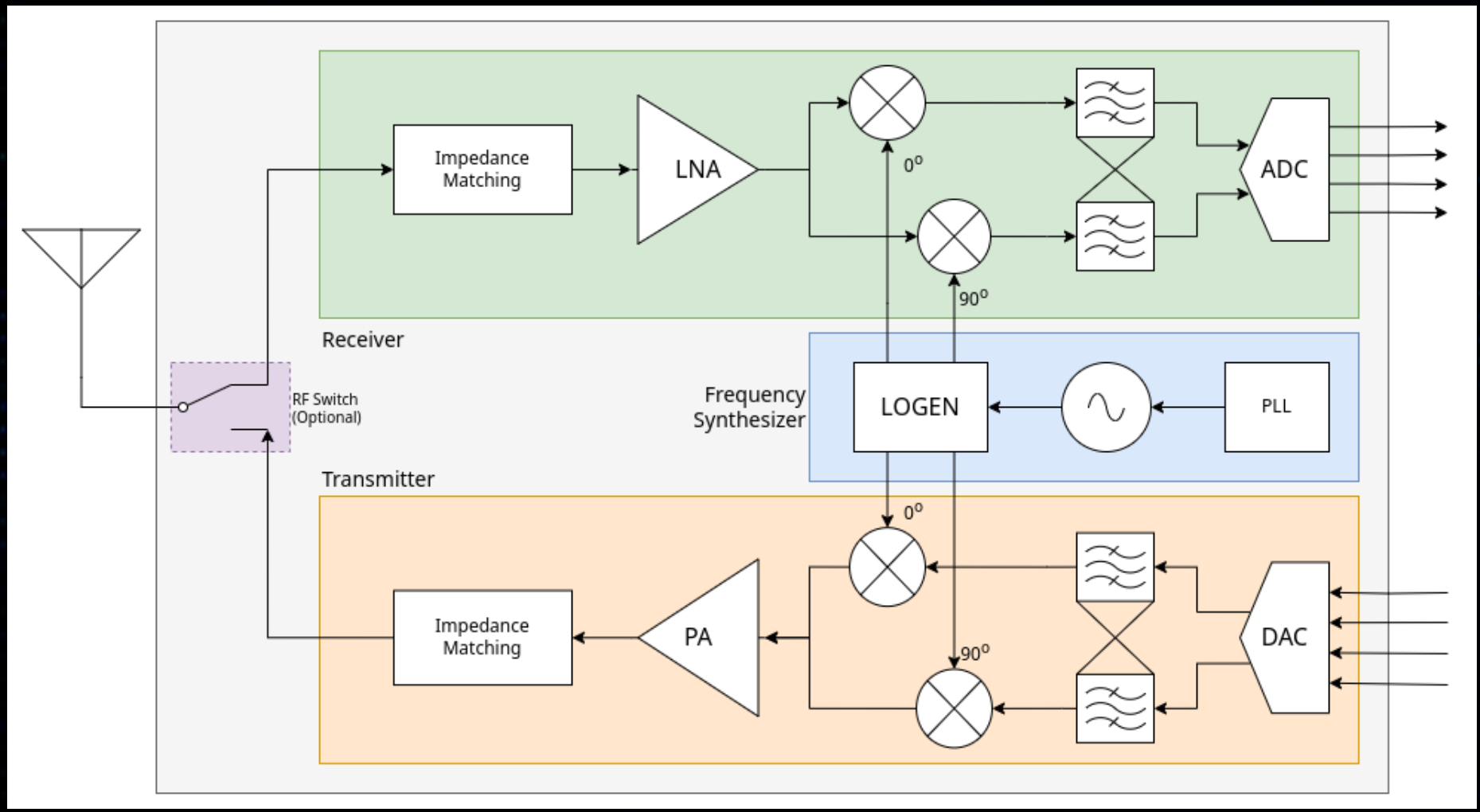
Теория: SDR, DSP, Signals

SDR – RF Transceiver AD9361



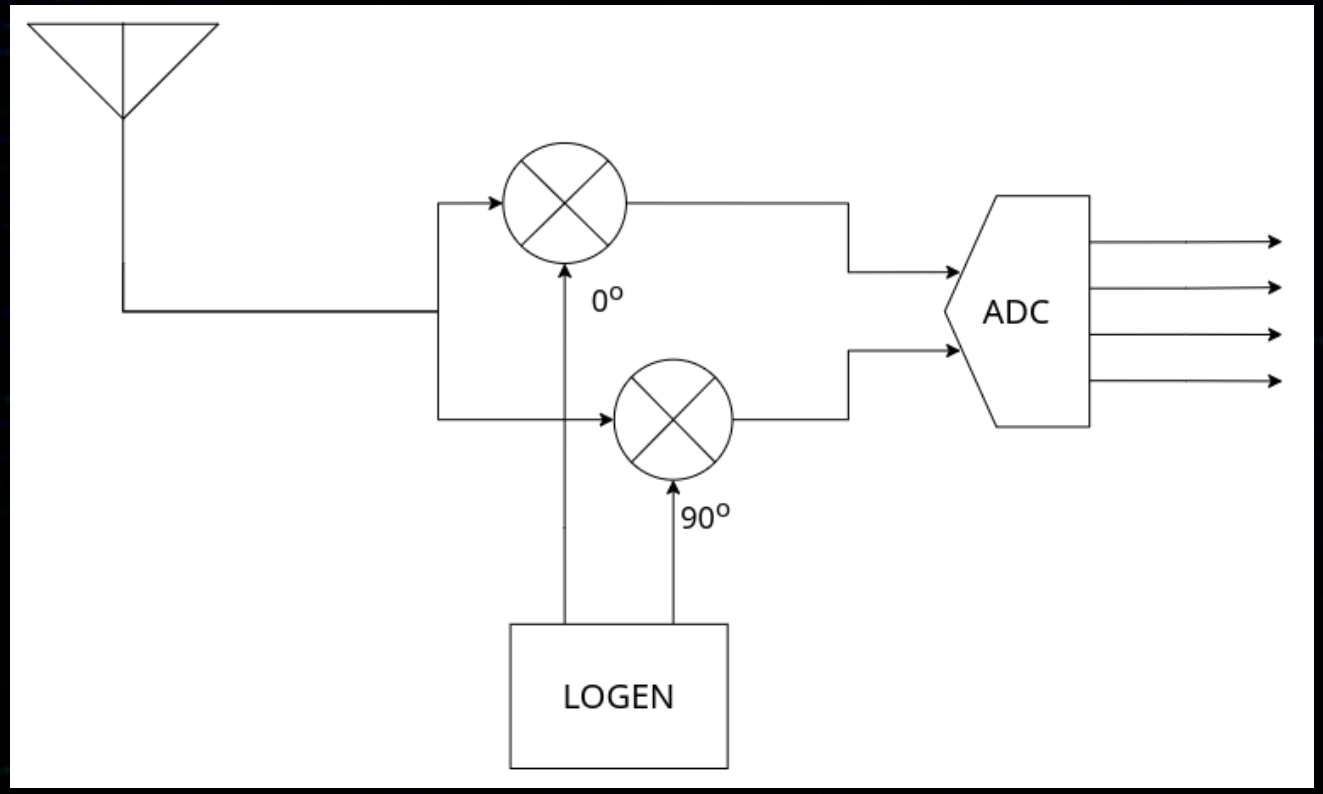
Теория: SDR, DSP, Signals

SDR – Упрощенная схема AD9361



Теория: SDR, DSP, Signals

SDR – Упрощенная схема упрощенной схемы AD9361



IQ Signals

$$s(t) = I(t)\cos(2\pi f t) + Q(t)\sin(2\pi f t) \quad \rightarrow \quad s(t) = I(t) + jQ(t)$$

http://news.cqham.ru/articles/detail.phtml?id=519

Теория: SDR, DSP, Signals

SDR – Формат данных bladeRF 2.0

```
enum bladerf_format {
        BLADERF_FORMAT_SC16_Q11,
        BLADERF_FORMAT_SC16_Q11_META,
        BLADERF_FORMAT_PACKET_META,
        BLADERF_FORMAT_SC8_Q7,
        BLADERF_FORMAT_SC8_Q7_META
}
```

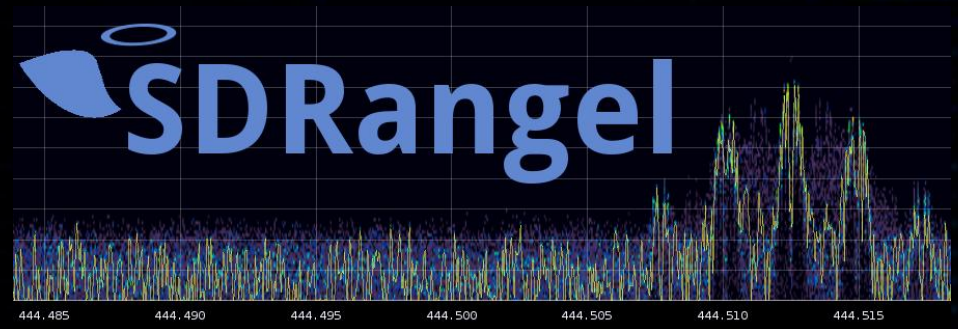
BLADERF_FORMAT_SC16_Q11

https://www.nuand.com/libbladeRF-doc/v2.5.0/group___s_t_r_e_a_m_i_n_g___f_o_r_m_a_t.html

Теория: SDR, DSP, Signals

DSP

GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM



yate BTS

GQRX-SCANNER

python™

C++

Теория: SDR, DSP, Signals

Signals – Общие параметры – Понятие модуляции

Параметры сигнала_

$$s(t) = A\cos(\omega t + \varphi)$$

Где:
A – амплитуда
$\omega = 2\pi f$ – частота в радианах
φ – начальная фаза

Модуляция_

$s(t) = A(t)\cos(\omega t + \varphi)$ - амплитудная

$s(t) = A\cos(\omega(t) + \varphi)$ - частотная

$s(t) = A\cos(\omega t + \varphi(t))$ - фазовая

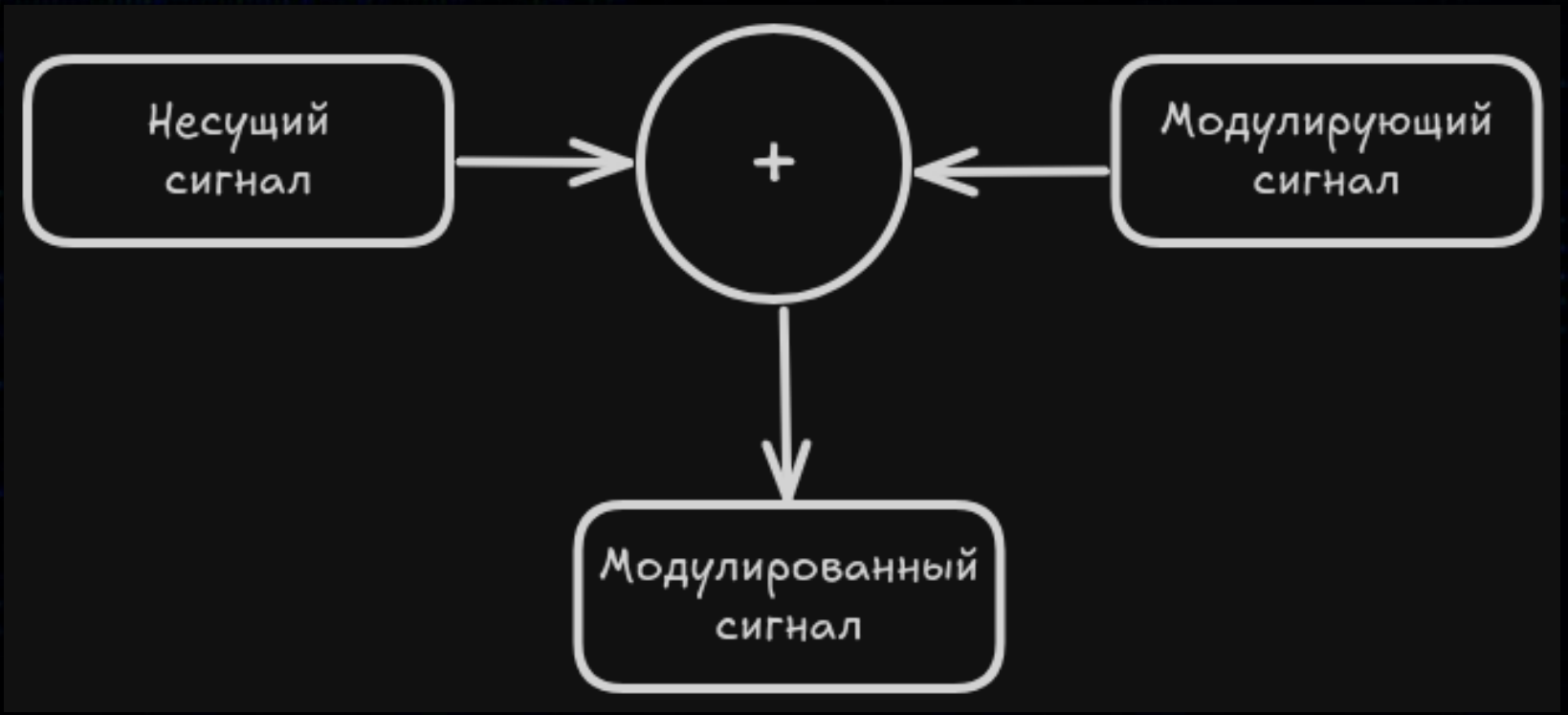
Обобщённая запись_

$$s(t) = A(t)\cos(\omega(t) + \varphi(t) + \varphi_0)$$

Где $A(t)$, $\omega(t)$ и $\varphi(t)$ - задающие функции амплитуды, частоты и фазы соответственно.

Теория: SDR, DSP, Signals

Signals – Классификация модуляции – Понятие модуляции

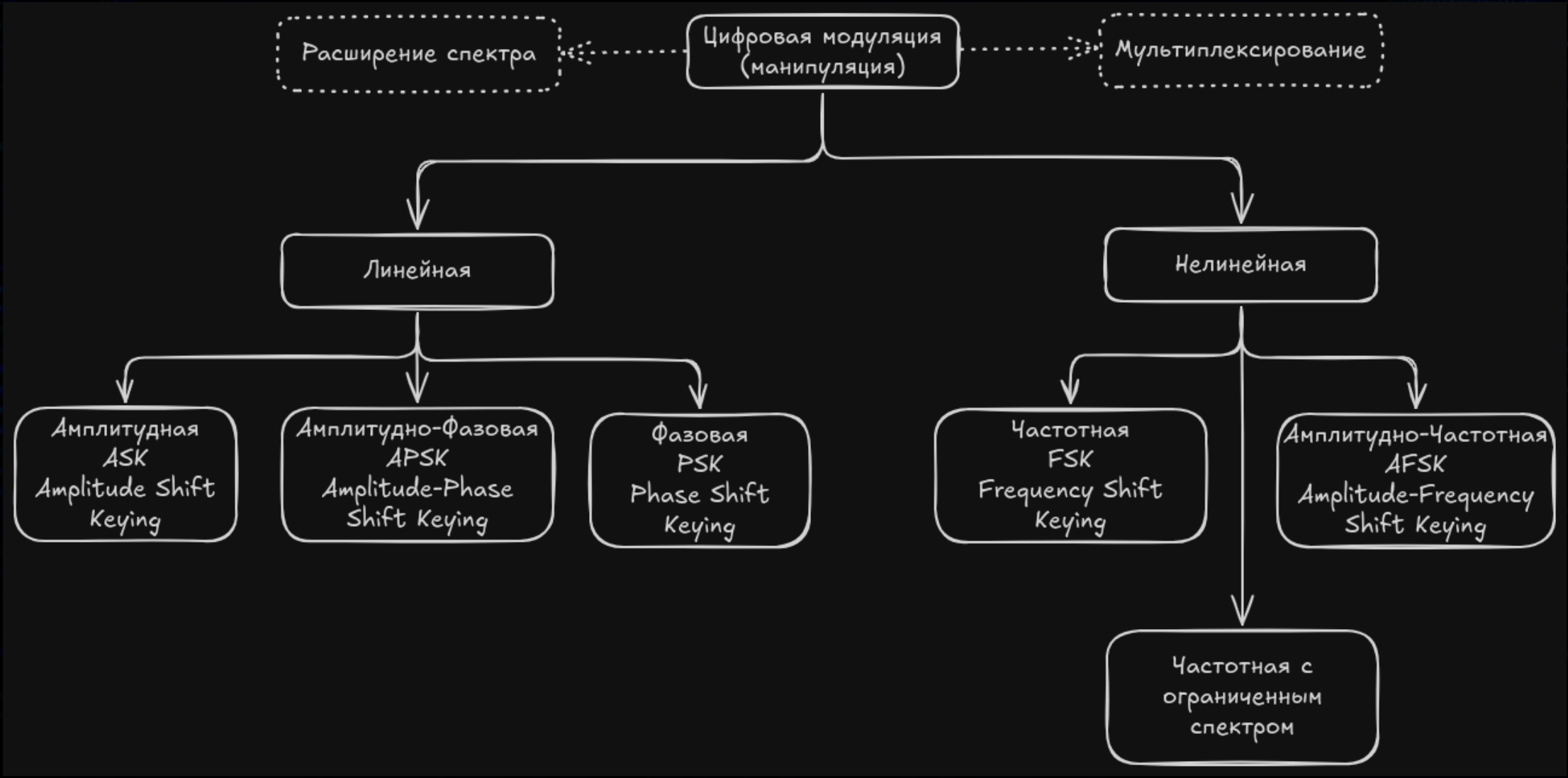


Теория: SDR, DSP, Signals

Signals – Классификация модуляции – Несущий сигнал

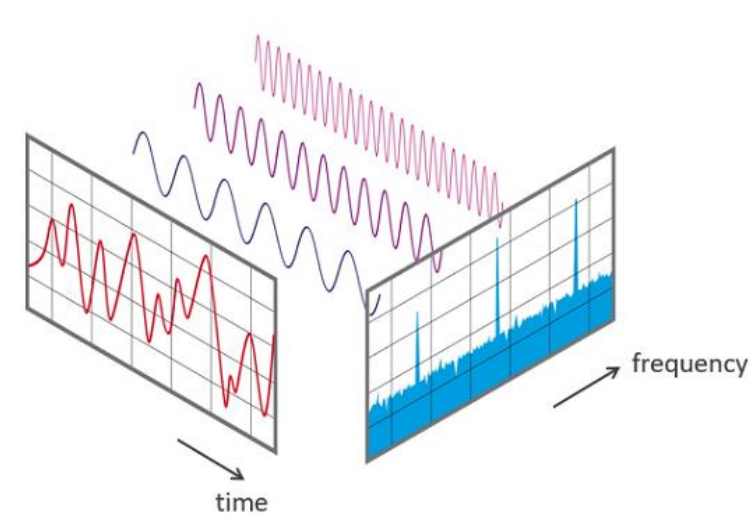
Теория: SDR, DSP, Signals

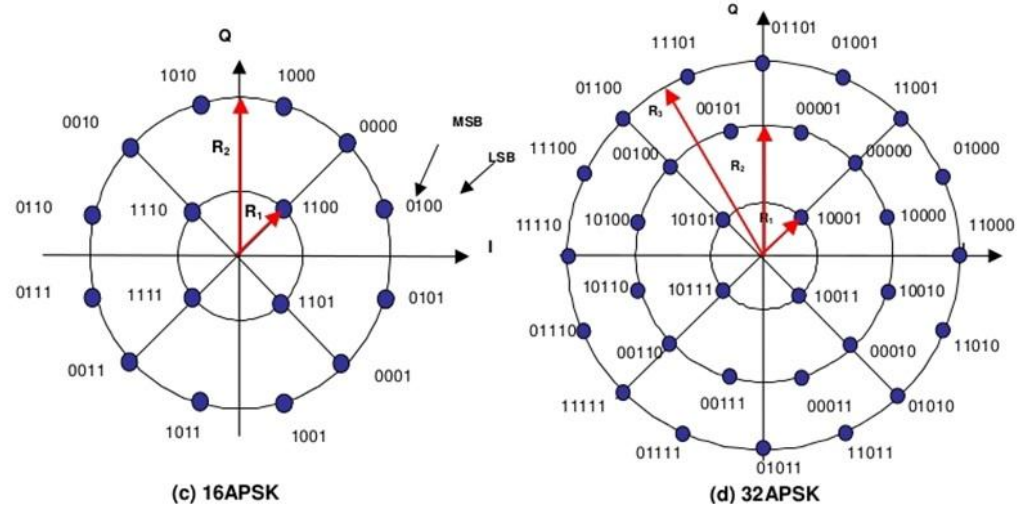
Signals – Классификация модуляции – Виды манипуляций

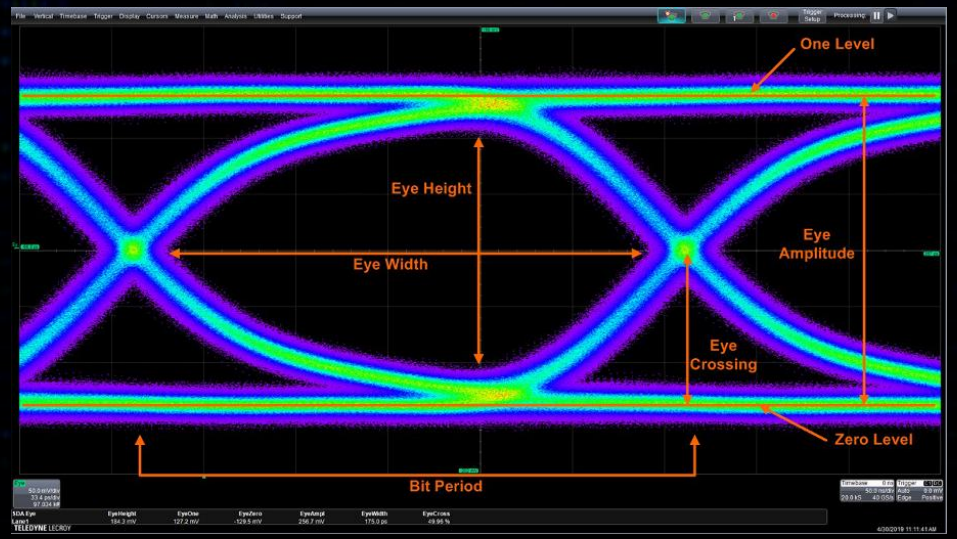


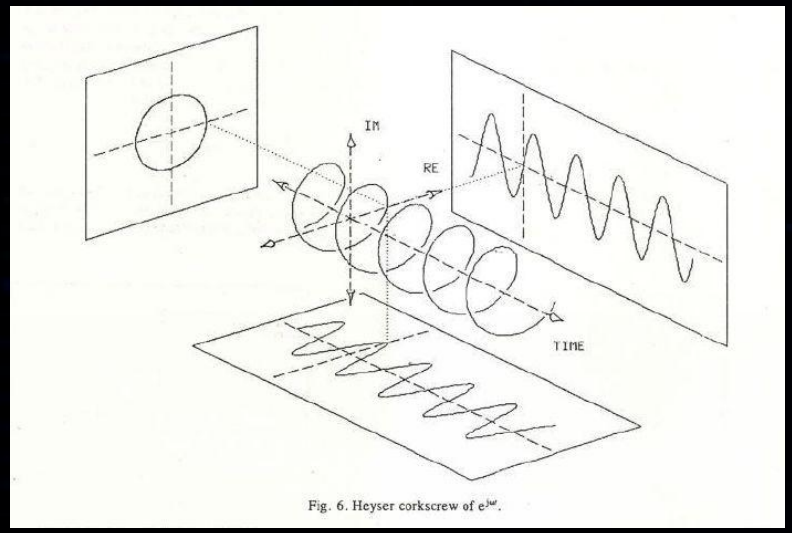
Теория: SDR, DSP, Signals

Signals – Домены представления



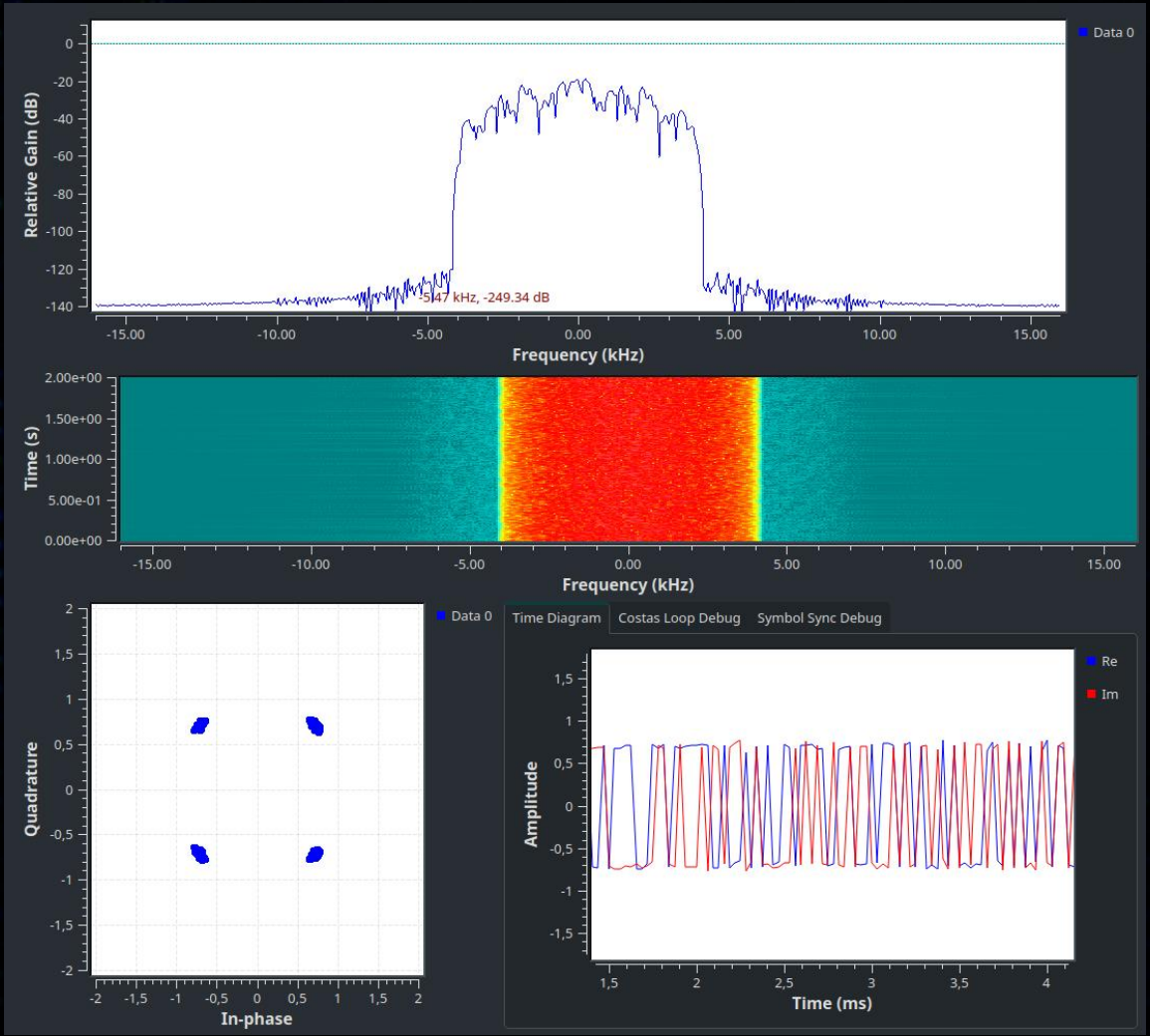






Теория: SDR, DSP, Signals

Signals – Домены представления – QPSK в разных доменах



Домены как инструмент_

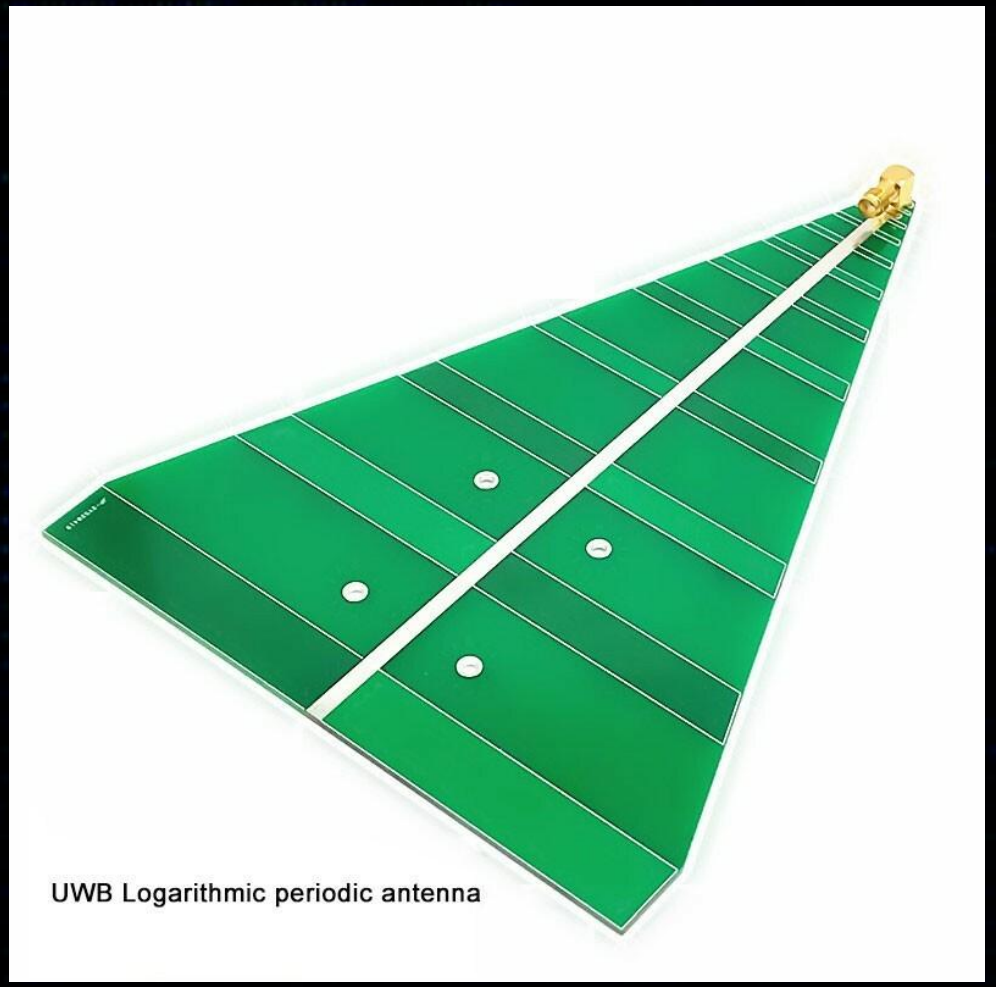
- Спектральное: поиск сигналов, анализ их характеристик по мощности и частоте, фильтрация
- Временное: анализ формы сигнала
- Фазовое: анализ фазовых изменений сигналов PSK
- Глазковая диаграмма: качество принимаемого сигнала, интерференция символов
- Все домены представления – главный инструментарий для анализа сигнала, совокупность признаков в разных доменах представления даёт достаточно информации для определения характера модуляции

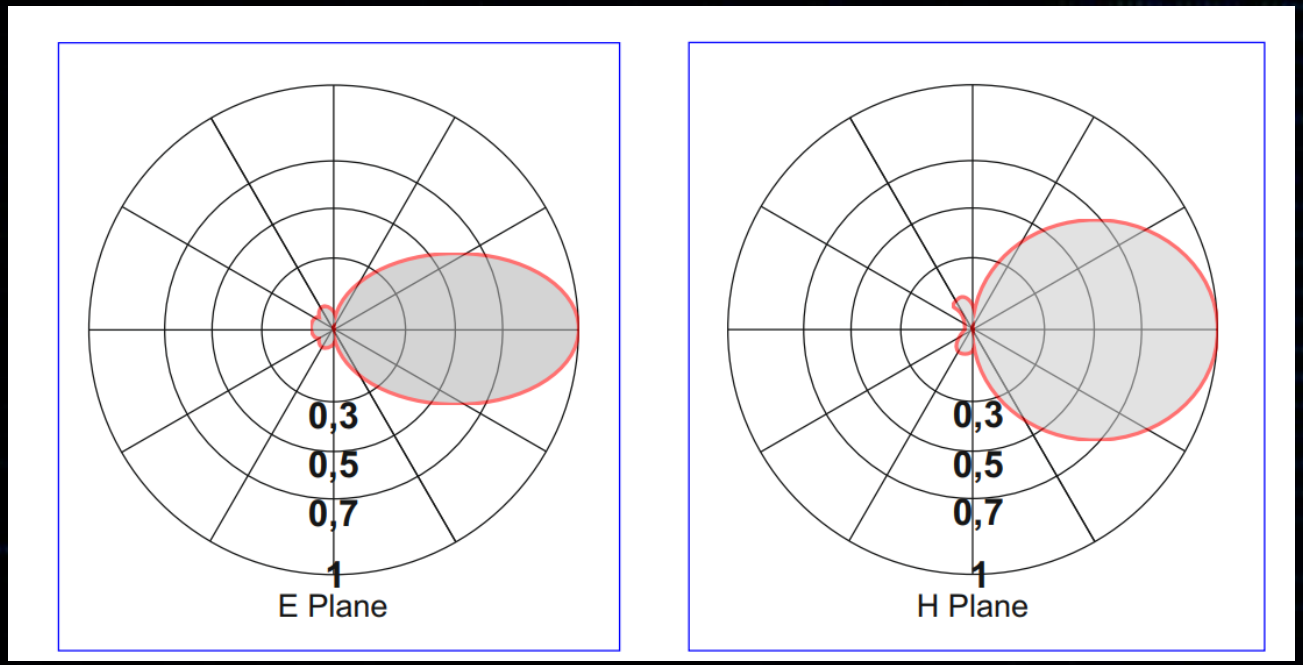
Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A

Технический анализ

Определение несущей частоты – Направленная антенна

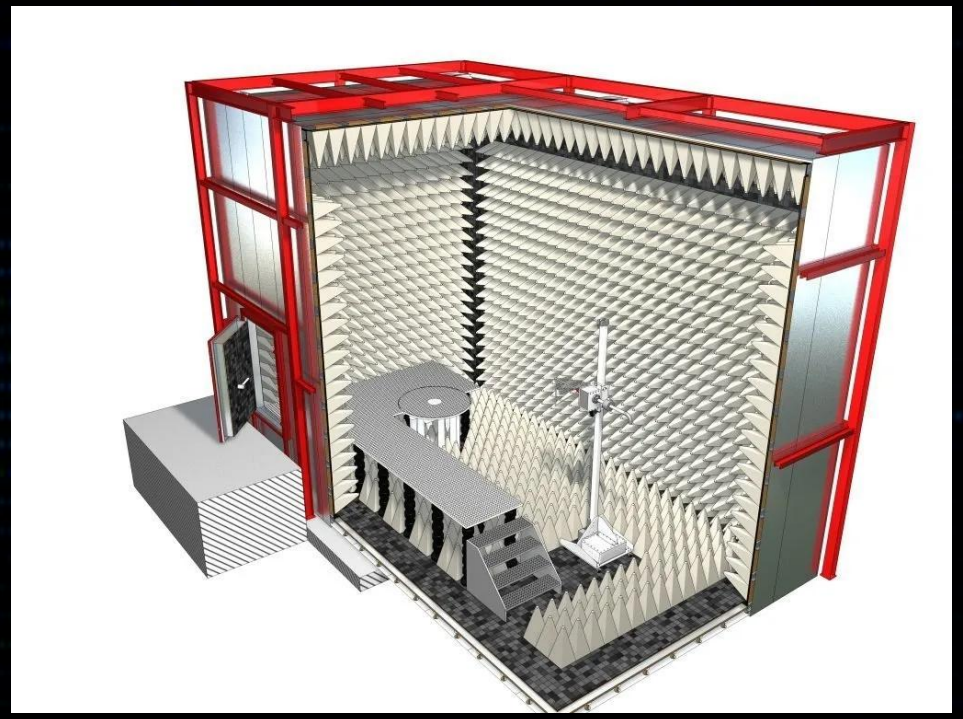




- SDRangel
- GQRX
- GNU Radio + gr-fosphor

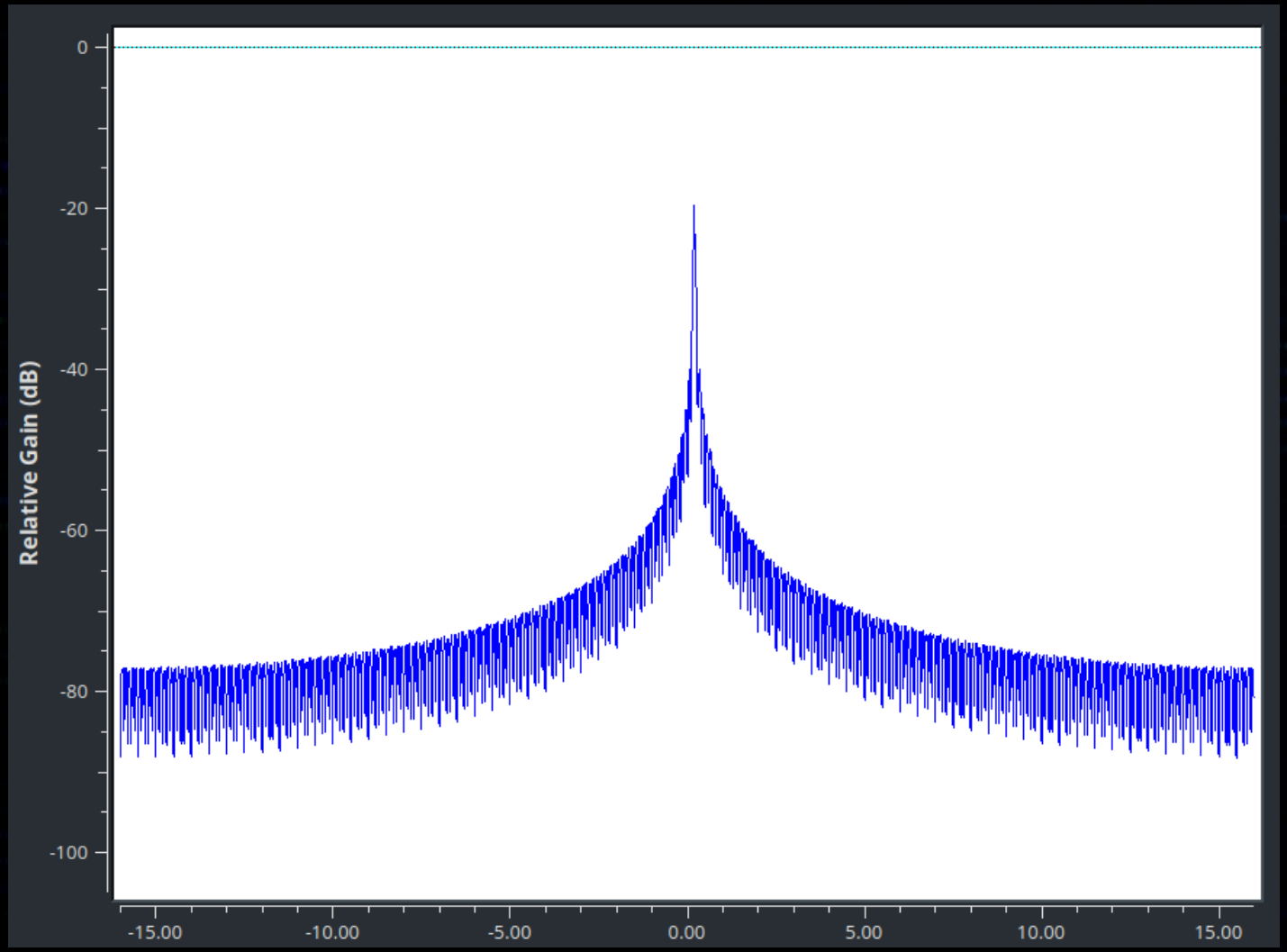
Технический анализ

Определение несущей частоты – Безэховая камера



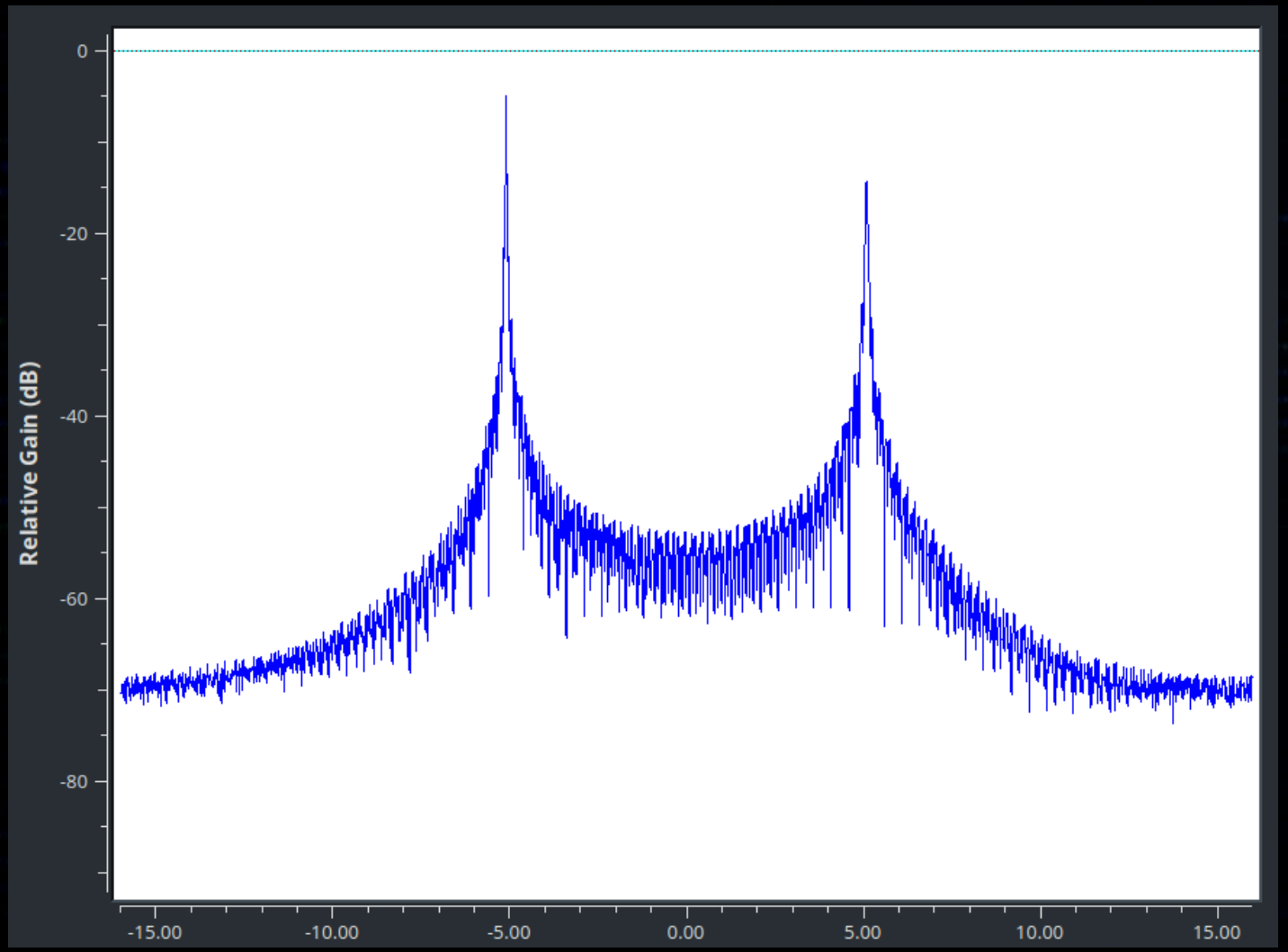
Технический анализ

Спектральный анализ – Манипуляция ASK



Технический анализ

Спектральный анализ – Манипуляция FSK



Технический анализ

Спектральный анализ – Манипуляция PSK

Технический анализ

Порядок манипуляции – Понятие

ASK (OOK)

Кодирует 0 бит отсутствием сигнала, 1 бит - наличием сигнала; поэтому так и называется - on-off shift keying, включено-выключено

FSK (2-FSK)

Кодирует 0 бит наличием сигнала меньшей частоты, 1 бит - наличием сигнала большей частоты

PSK (BPSK)

Кодирует 0 бит сигналом с фазой $180°$ (или π), 1 бит - сигналом с фазой $0°$

Символ

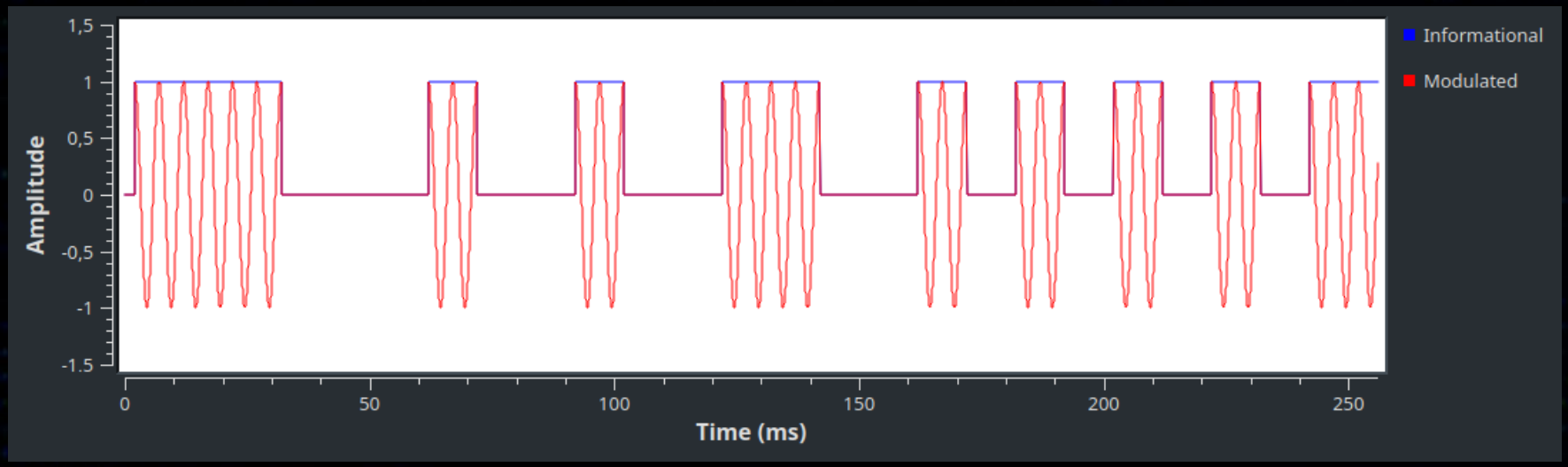
Некоторое значение Амплитуды, Частоты, Фазы или их совокупность

Порядок манипуляции

Определяет количество бит на символ

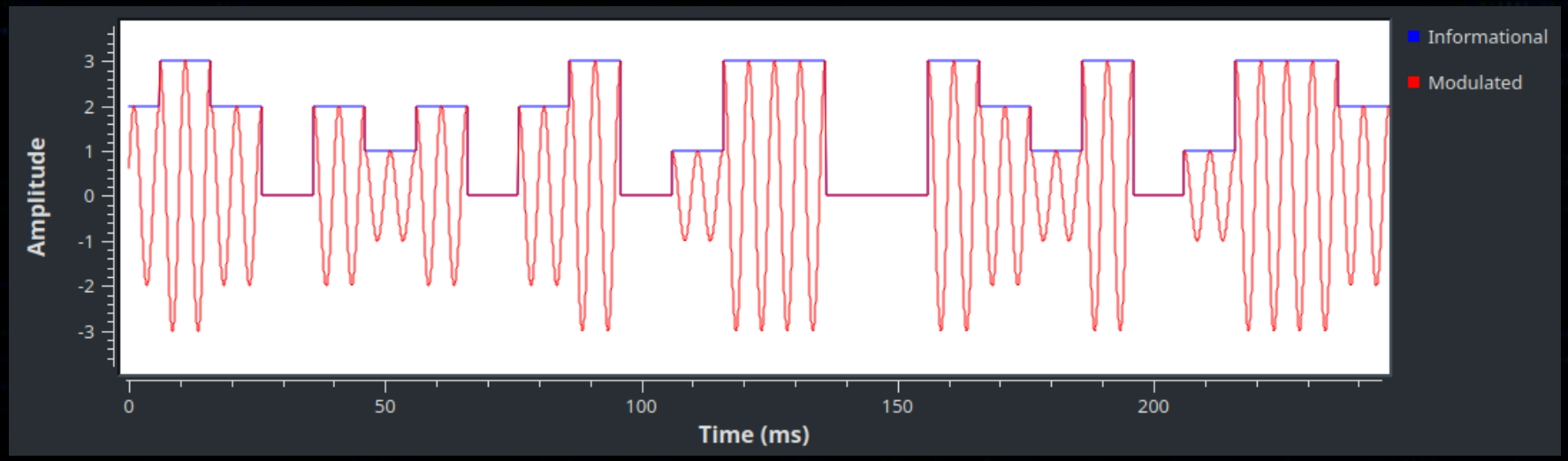
Технический анализ

Порядок манипуляции – ASK – Временное представление



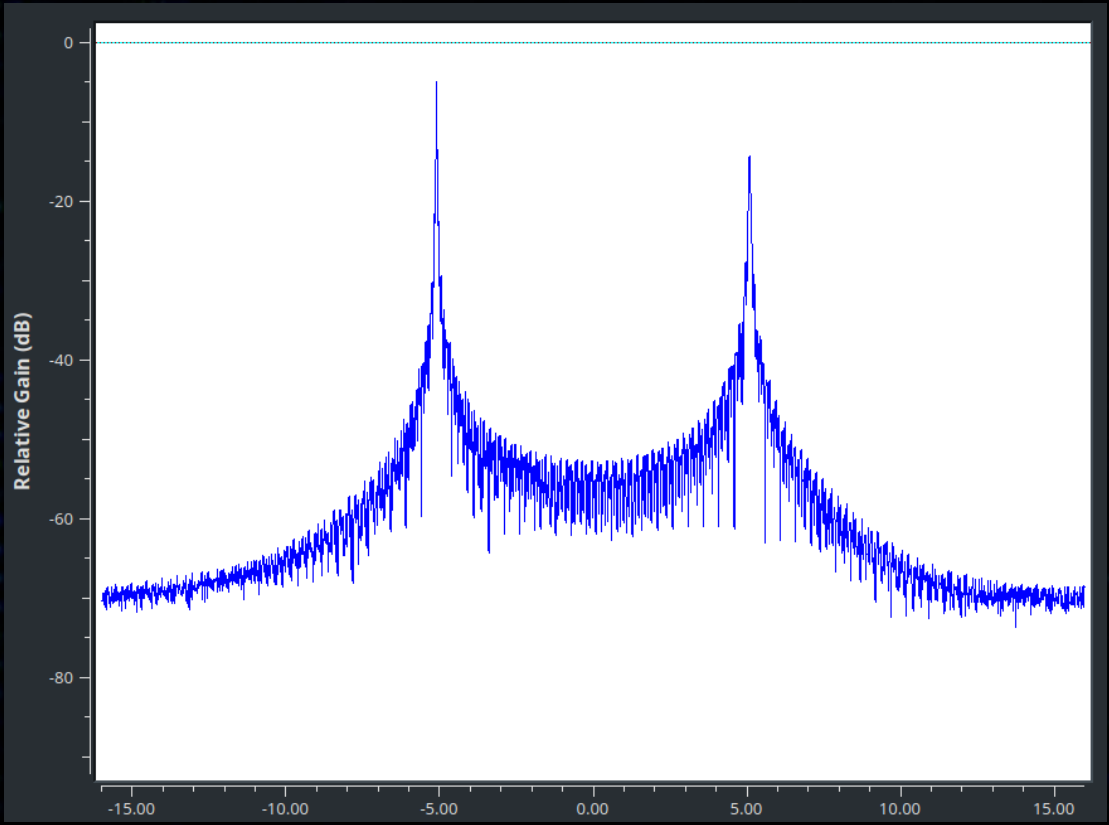
OOK

4-ASK



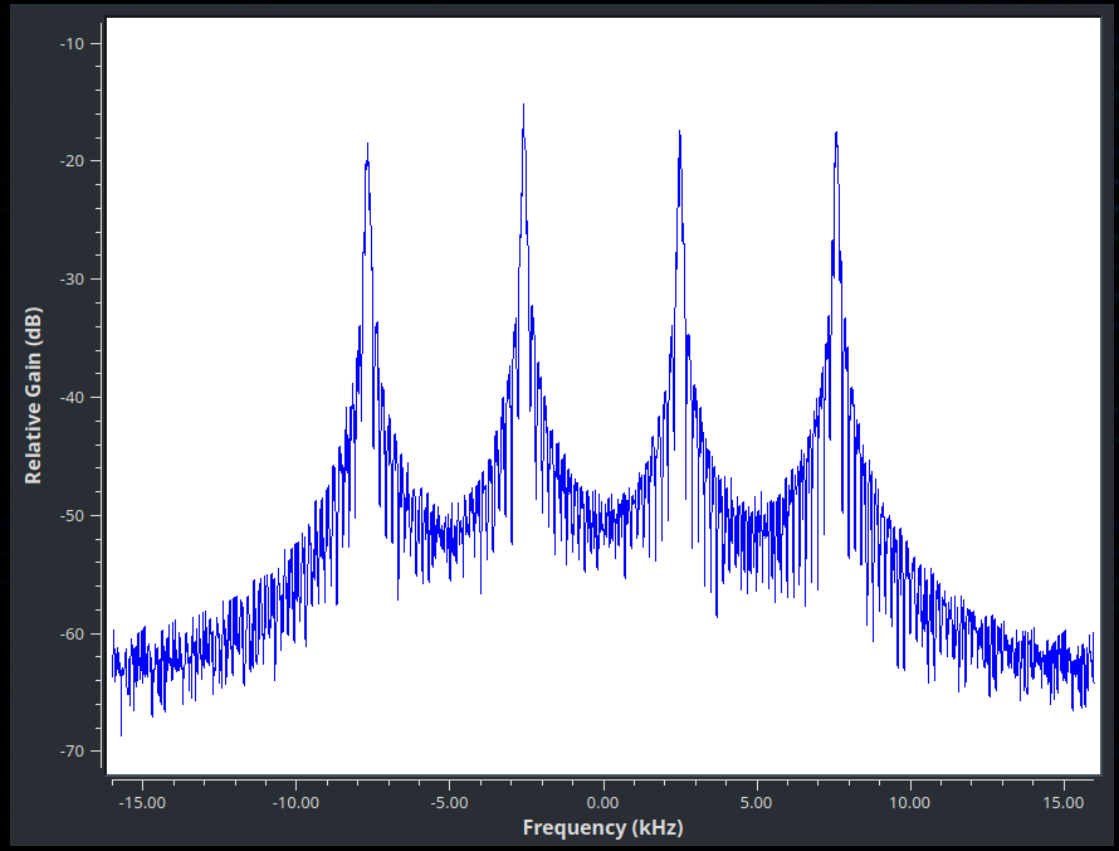
Технический анализ

Порядок манипуляции – FSK – Спектральное представление



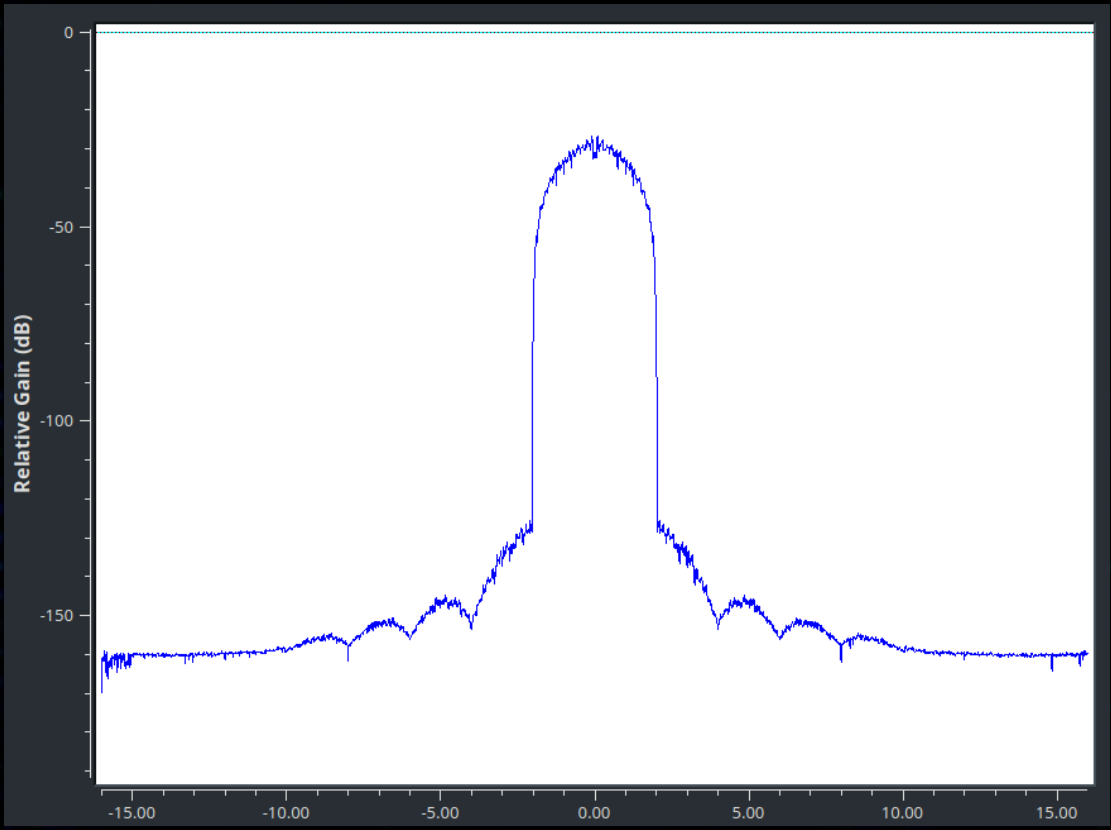
2-FSK

4-FSK



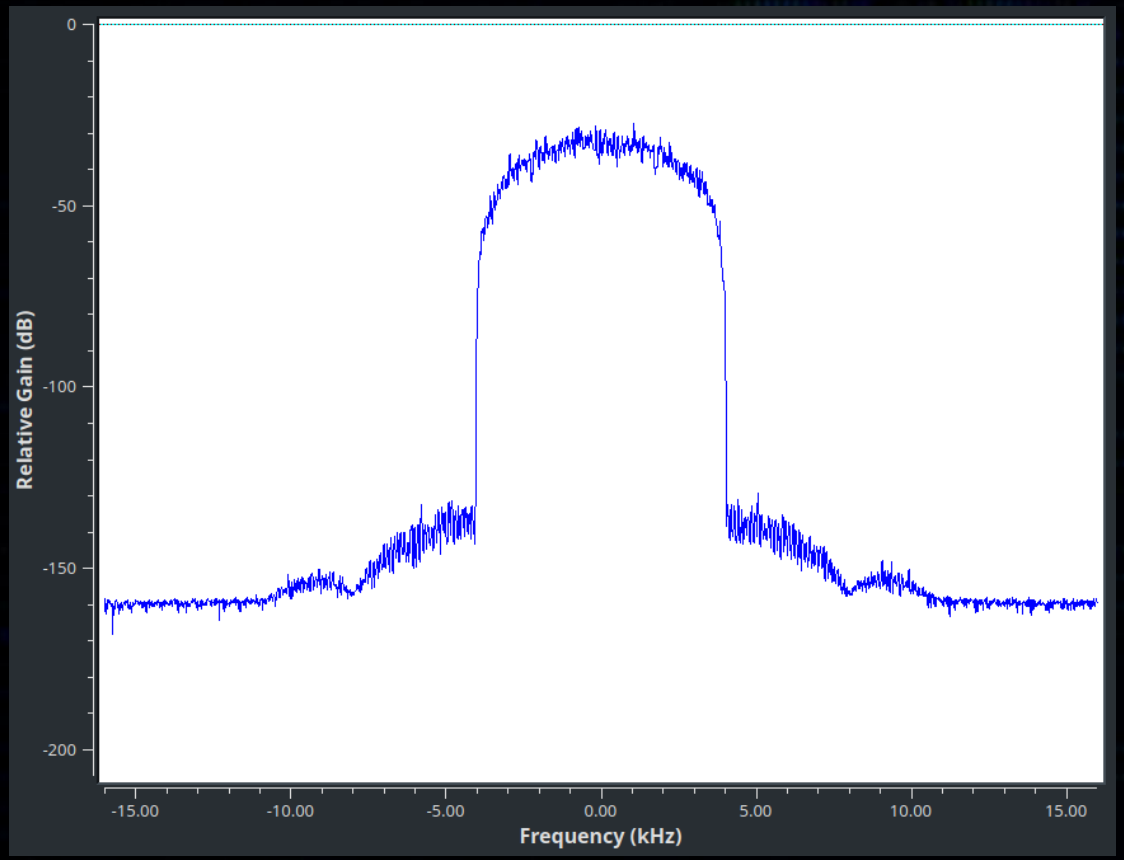
Технический анализ

Порядок манипуляции – PSK – Спектральное представление



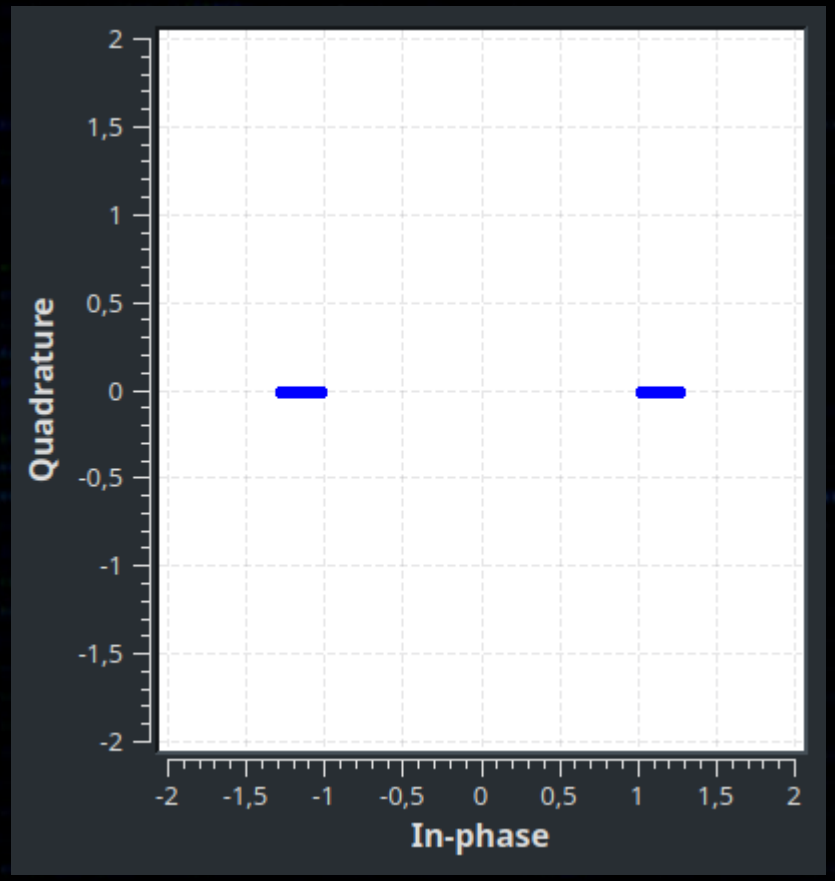
2-PSK (BPSK)

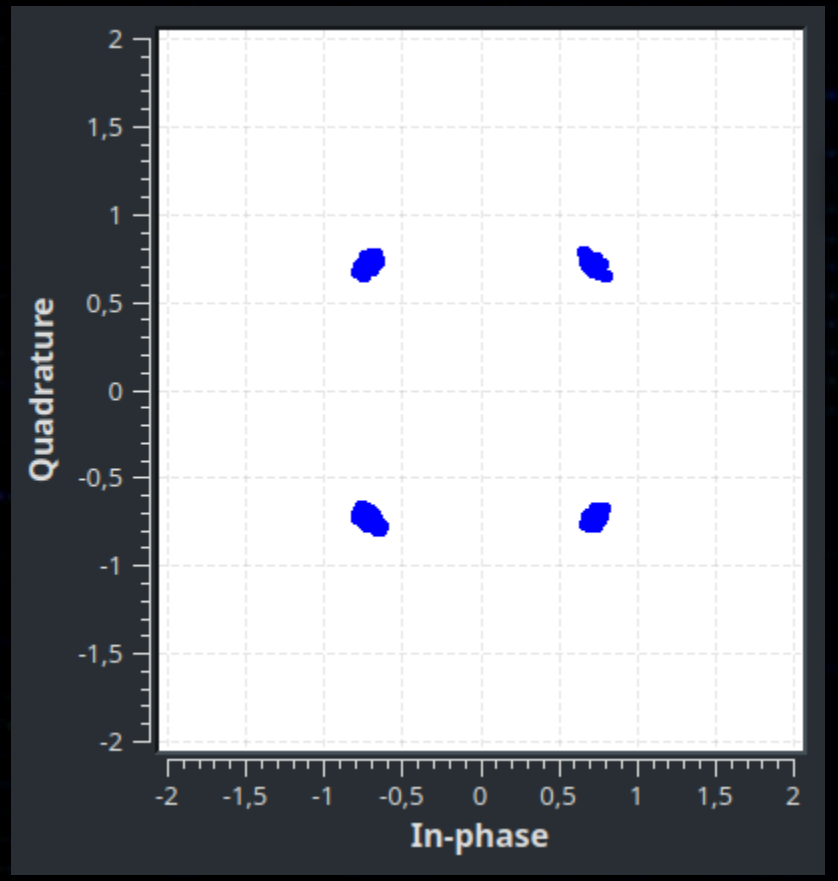
4-PSK (QPSK)
x2 symbol speed



Технический анализ

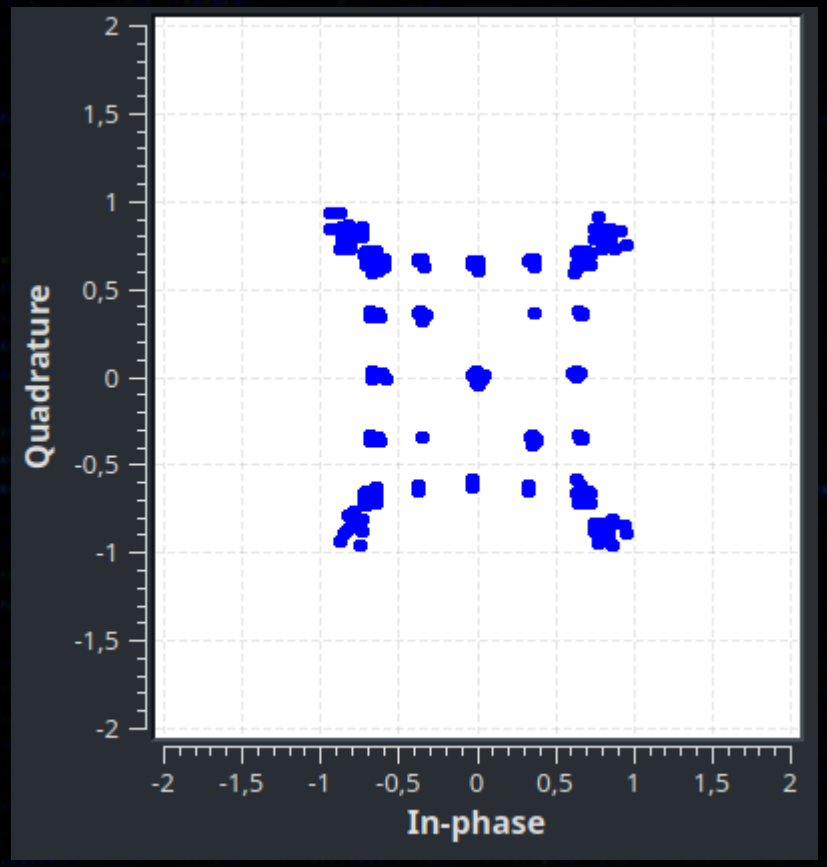
Порядок манипуляции – PSK – Фазовое представление #1

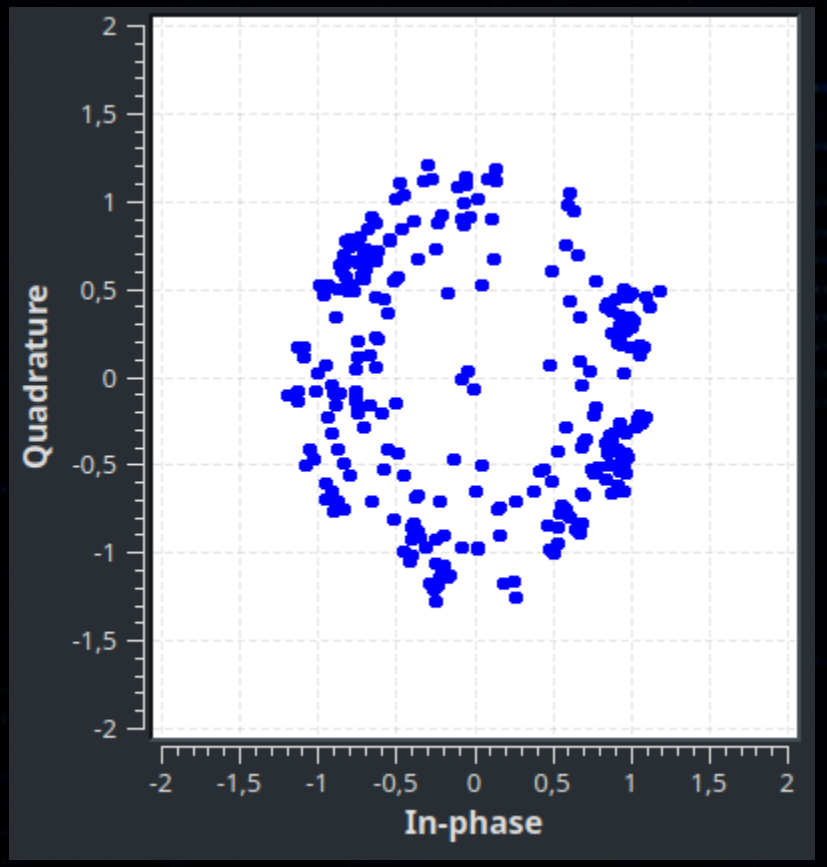




Технический анализ

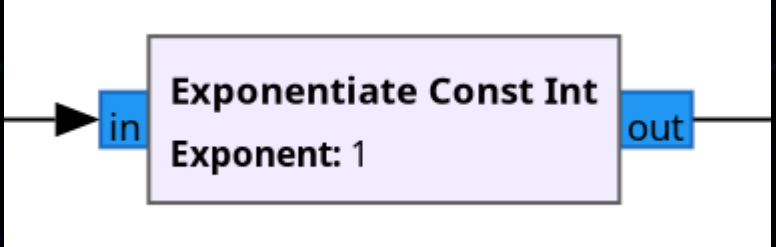
Порядок манипуляции – PSK – Фазовое представление #2



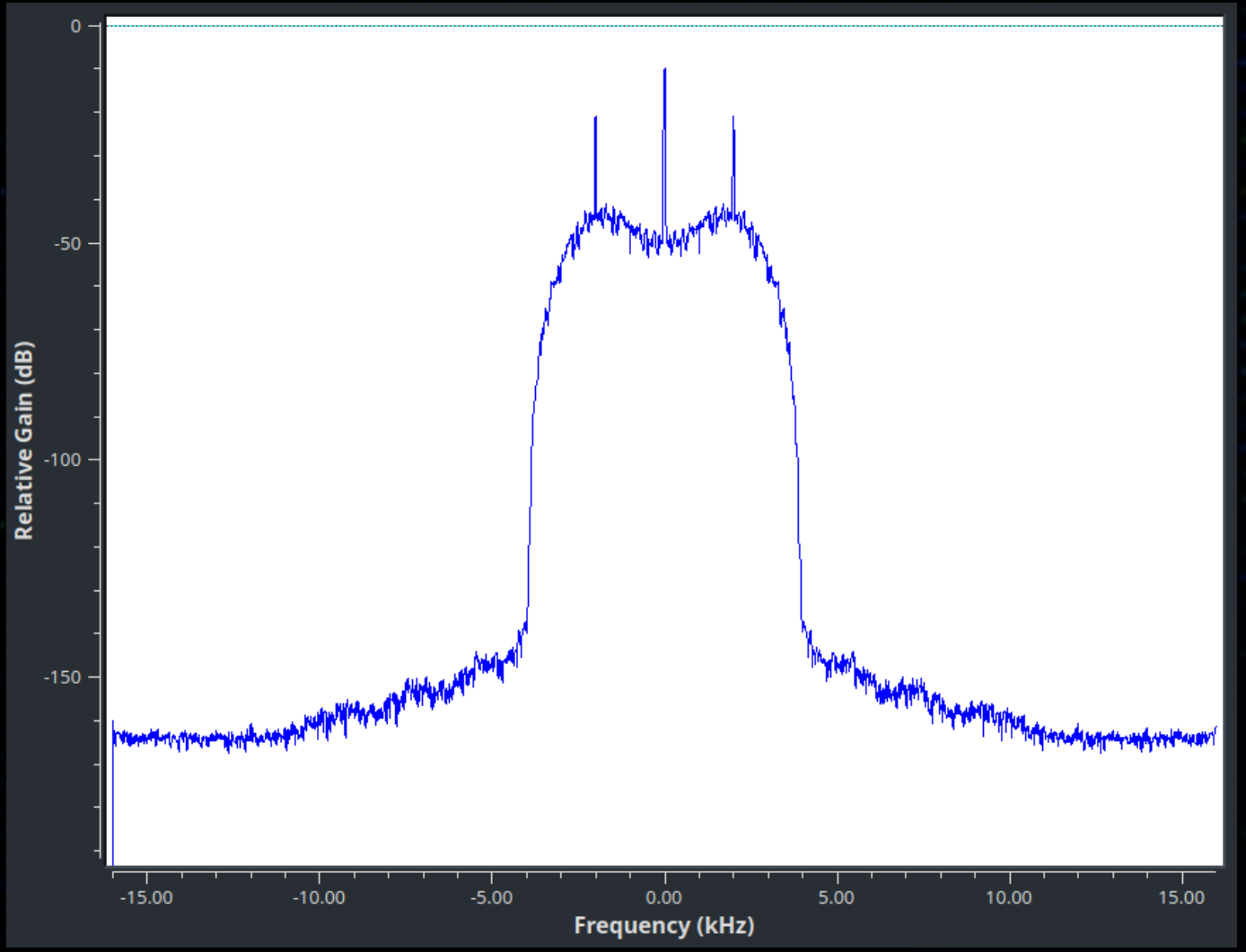


Технический анализ

Порядок манипуляции – M-th Power Method – BPSK

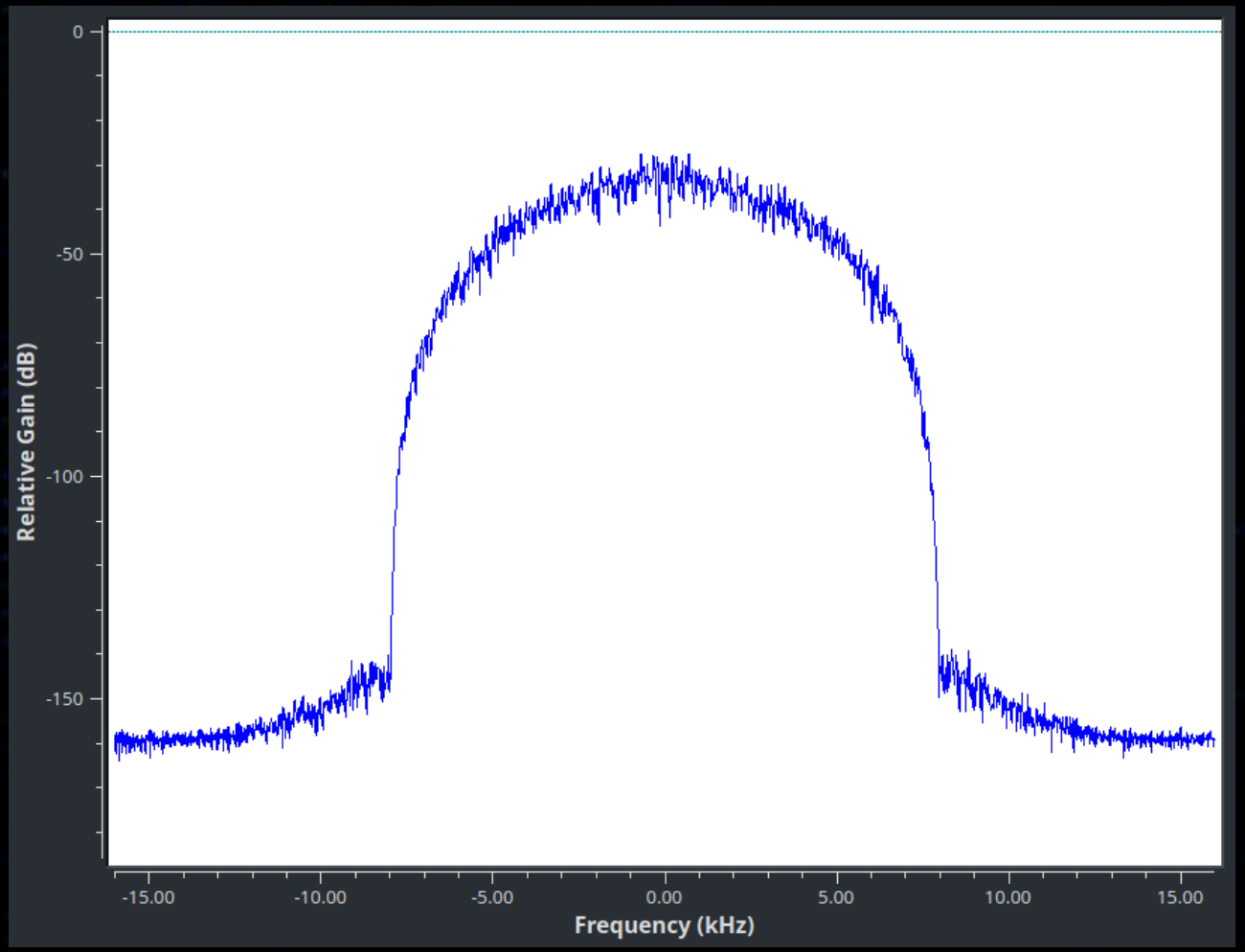



```
>>> (-1+0j)**2
(1+0j)
>>> (1+0j)**2
(1+0j)
```



Технический анализ

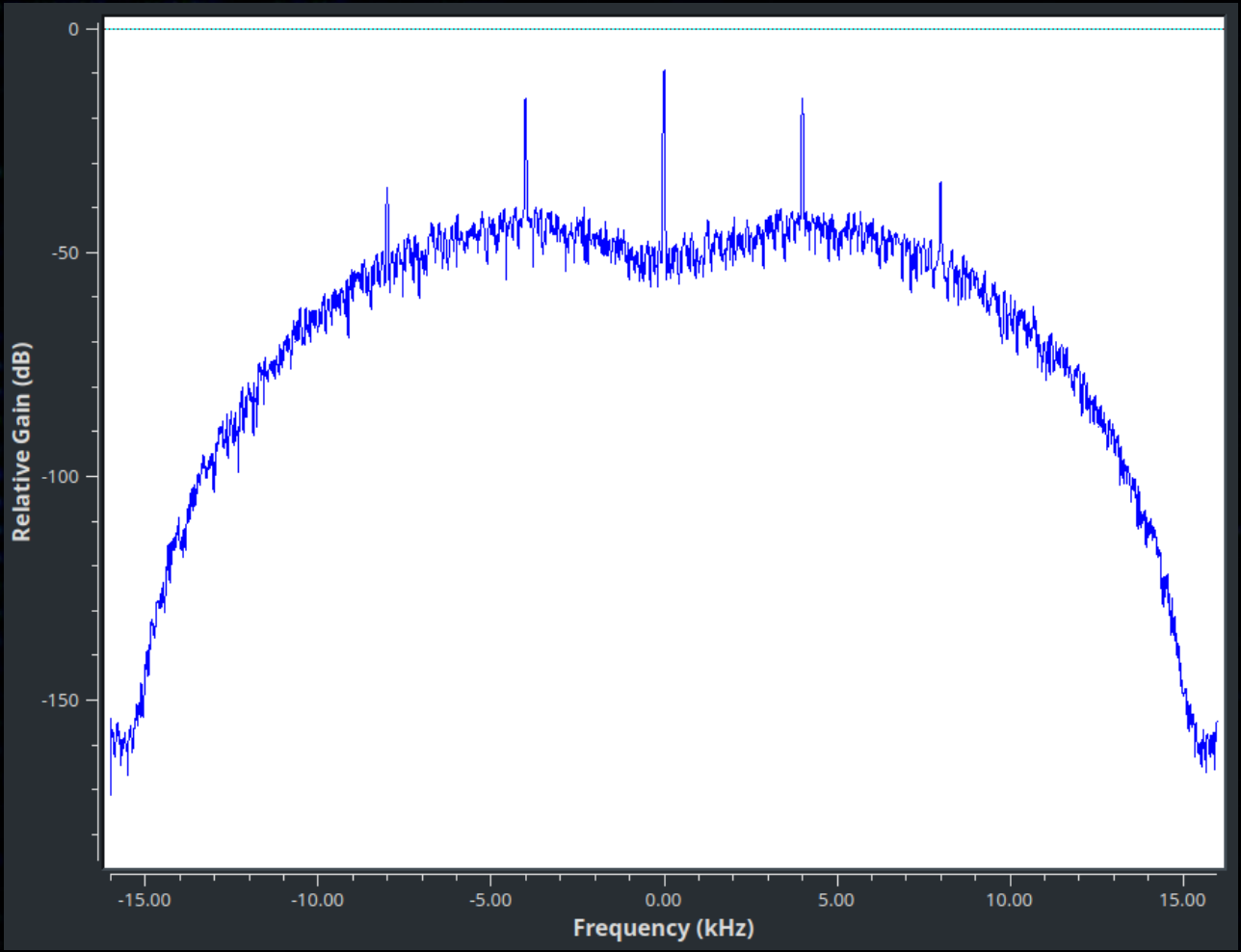
Порядок манипуляции – M-th Power Method – QPSK ^2




```
>>> (1+1j)**2
2j
>>> (1-1j)**2
-2j
>>> (-1+1j)**2
-2j
>>> (-1-1j)**2
2j
```

Технический анализ

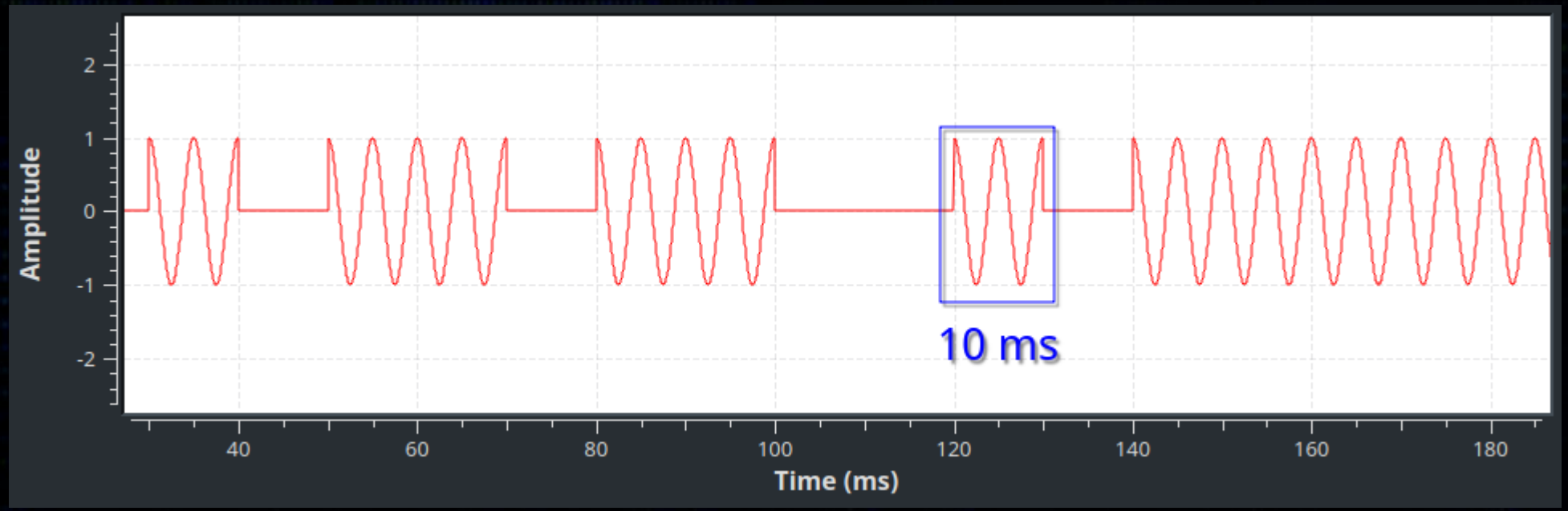
Порядок манипуляции – M-th Power Method – QPSK ^4




```
>>> (1+1j)**4
(-4+0j)
>>> (1-1j)**4
(-4-0j)
>>> (-1+1j)**4
(-4-0j)
>>> (-1-1j)**4
(-4+0j)
```

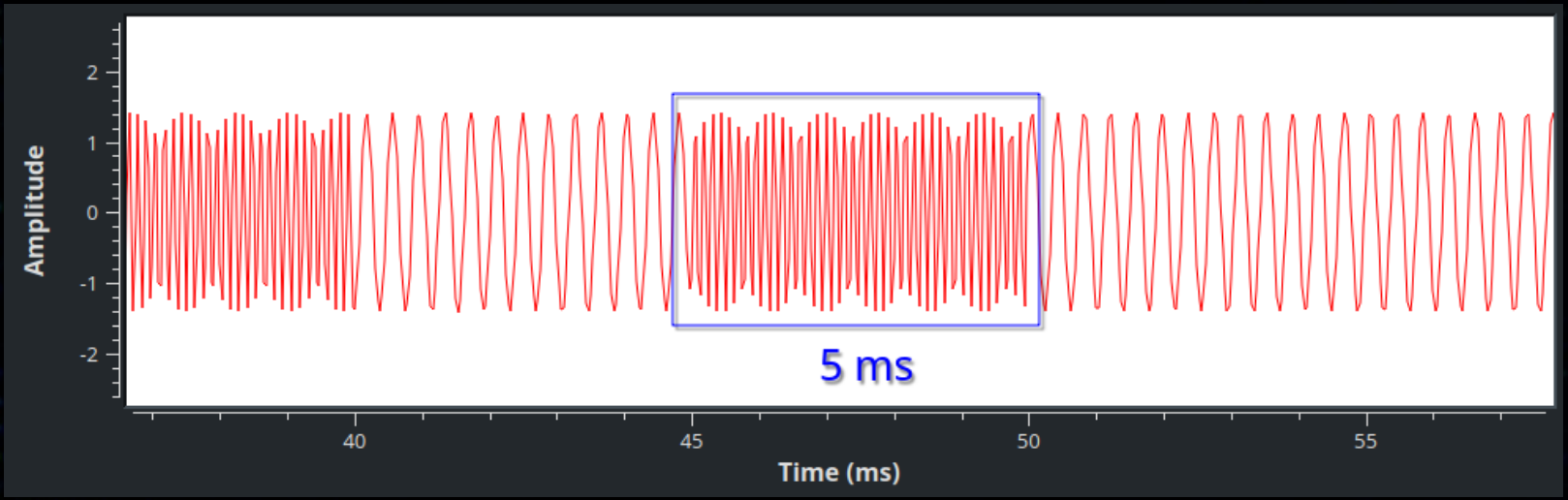
Технический анализ

Символьная скорость – Длительность символа ASK



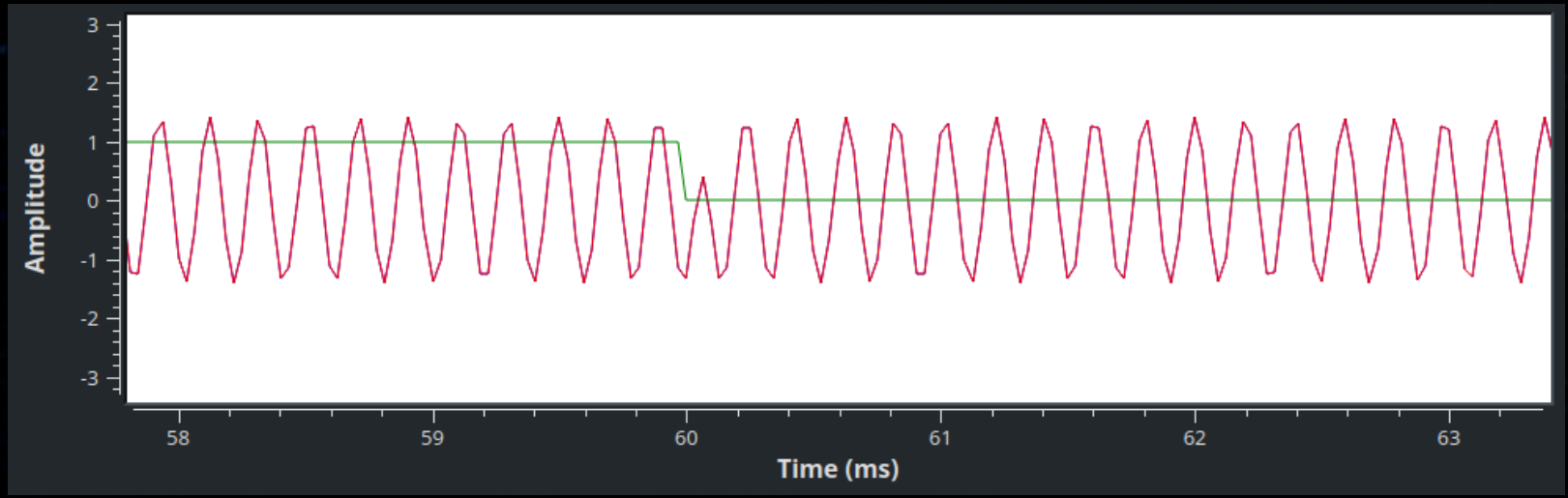
Технический анализ

Символьная скорость – Длительность символа FSK



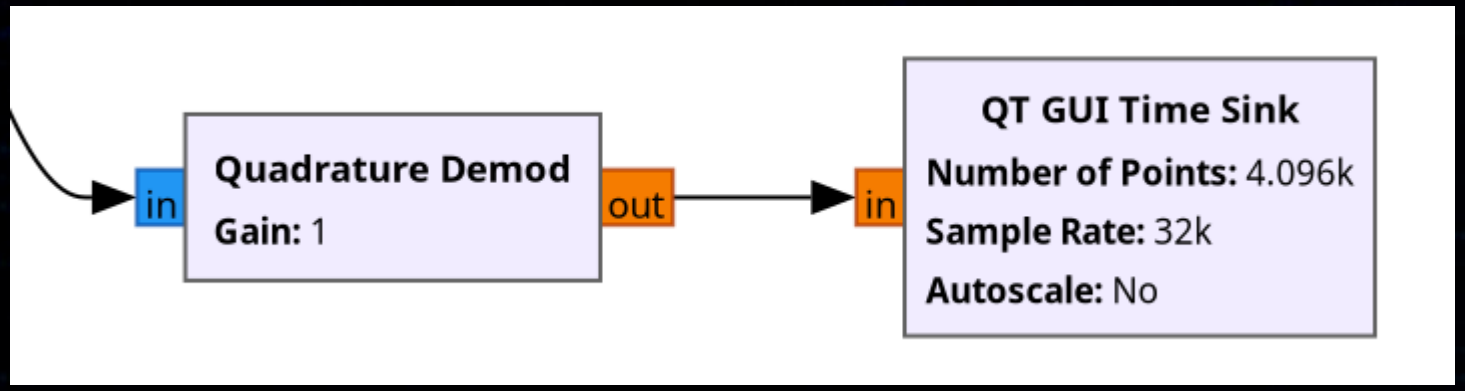
Классический 2-FSK

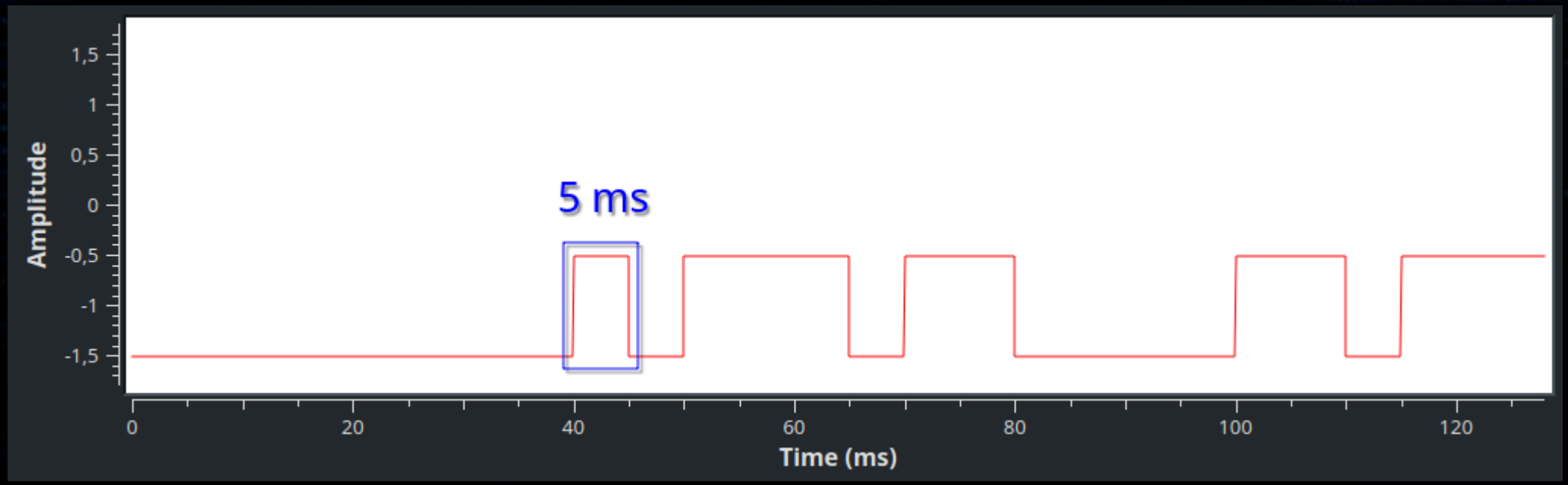
Граница символа GFSK



Технический анализ

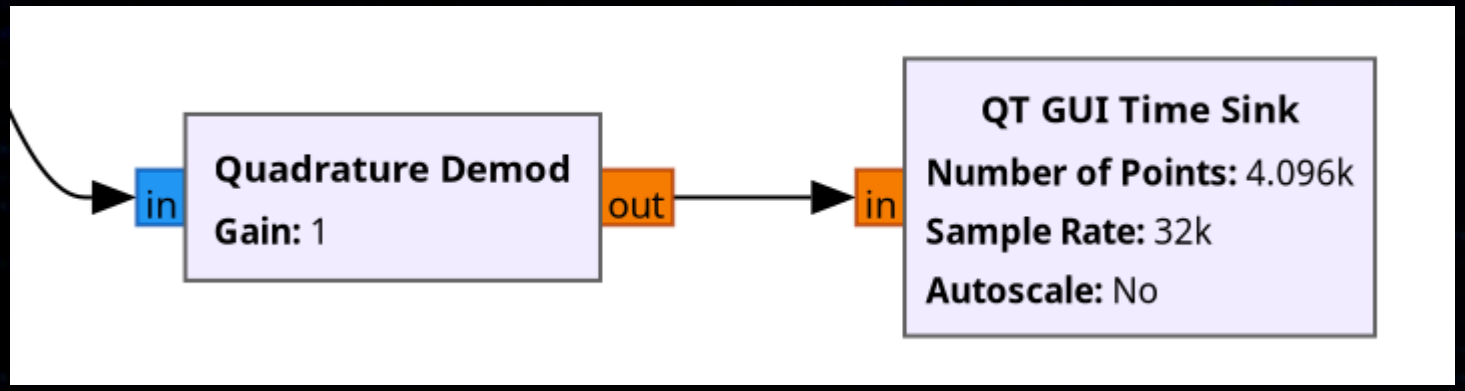
Символьная скорость – FSK после Quadrature Demod

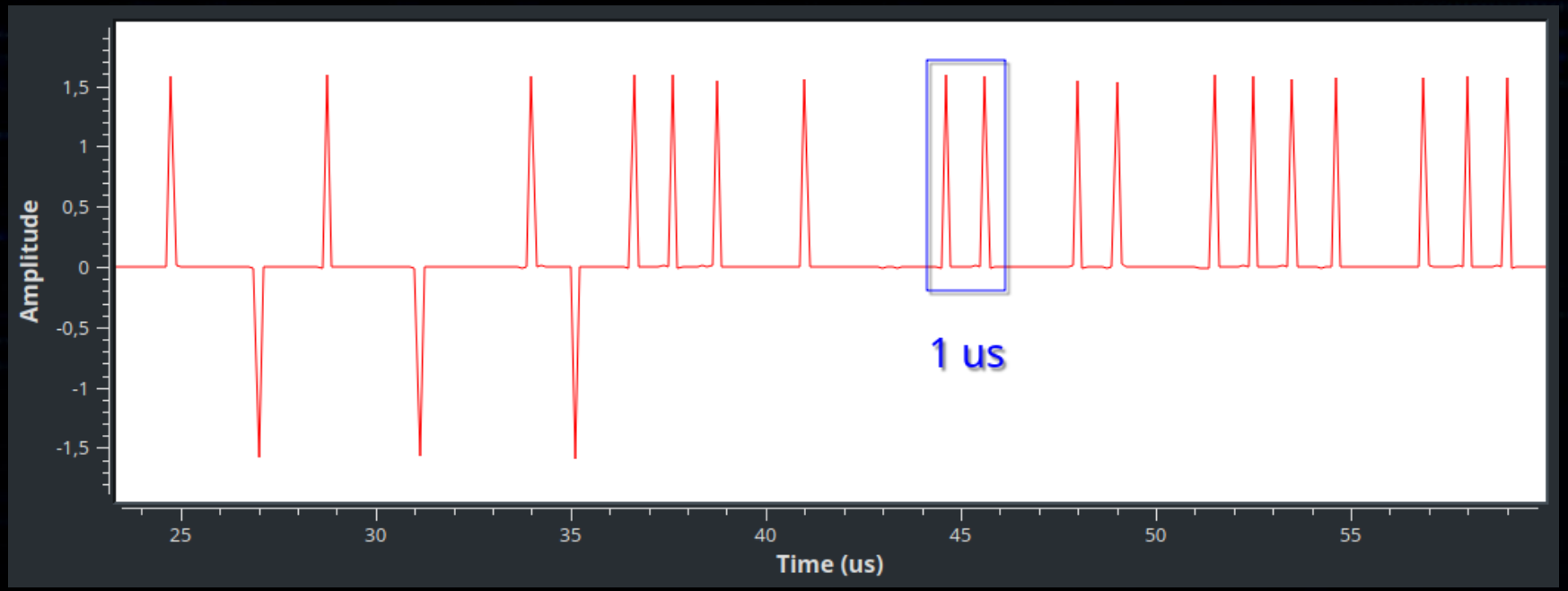




Технический анализ

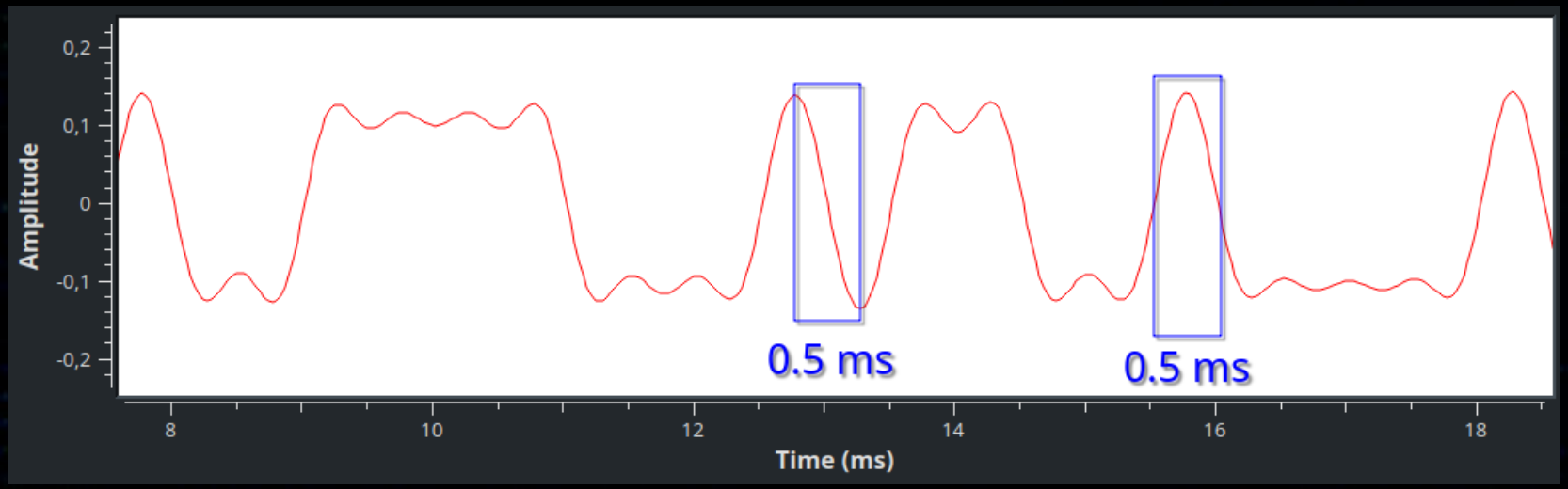
Символьная скорость – GFSK после Quadrature Demod





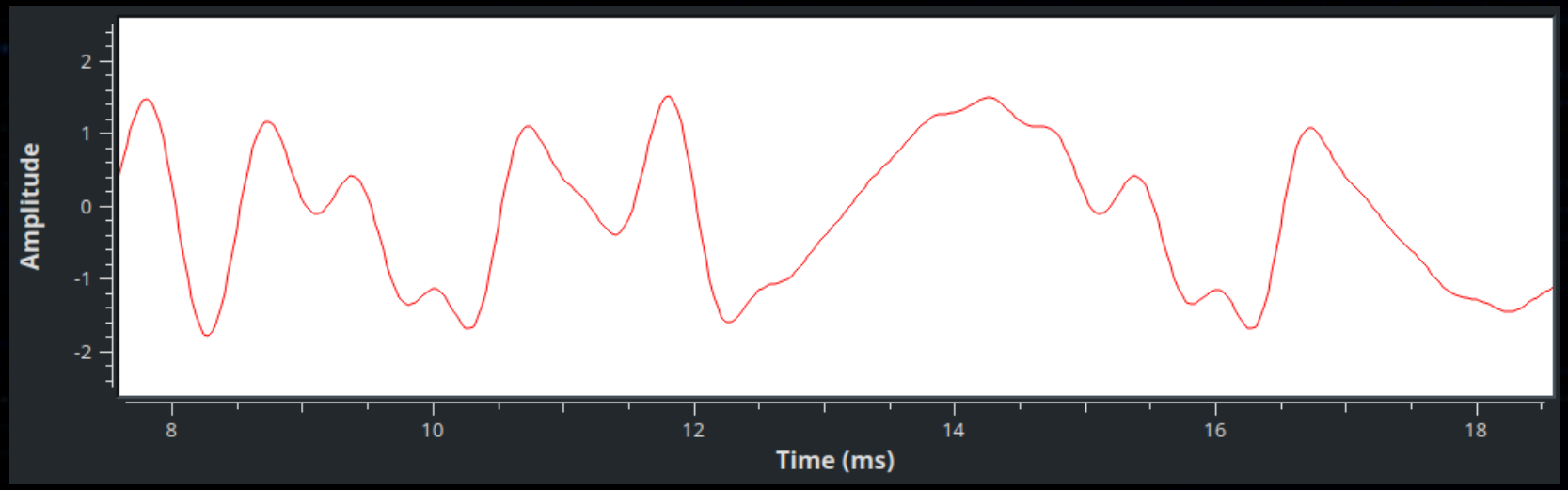
Технический анализ

Символьная скорость – Длительность символа BPSK



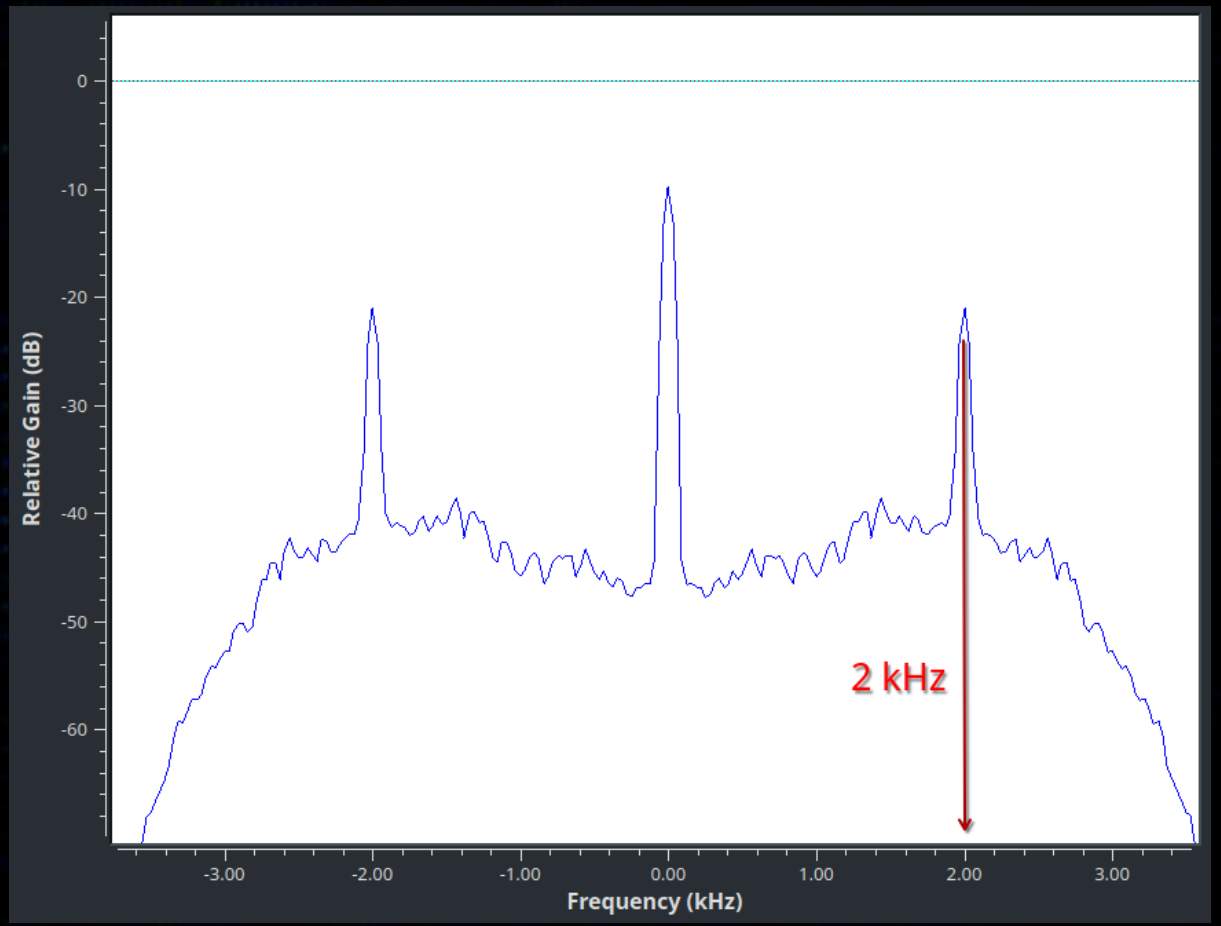
Сгенерированный сигнал без помех и фазового сдвига

Сгенерированный сигнал без помех, но с фазовым сдвигом $\pi/64$

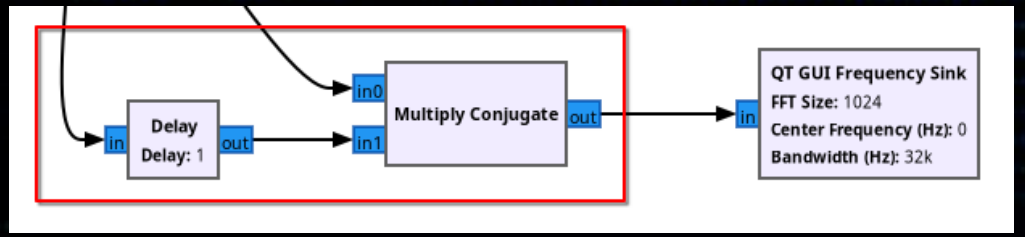


Технический анализ

Символьная скорость – Циклостационарный анализ – BPSK

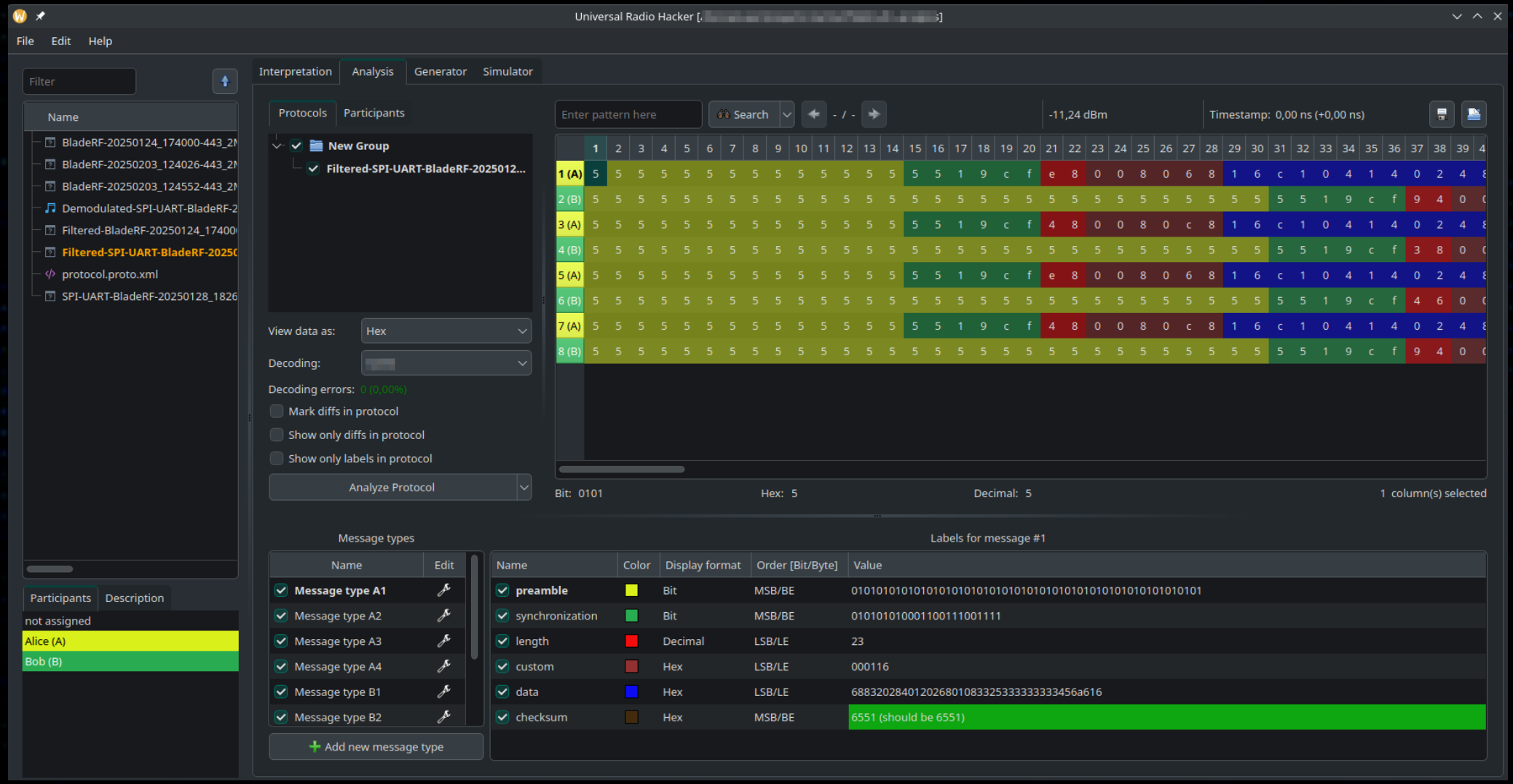


1 Hz = 1 cycle / sec
1 / 2000 = 0.5 ms



Технический анализ

Восстановление структуры кадра (URH)

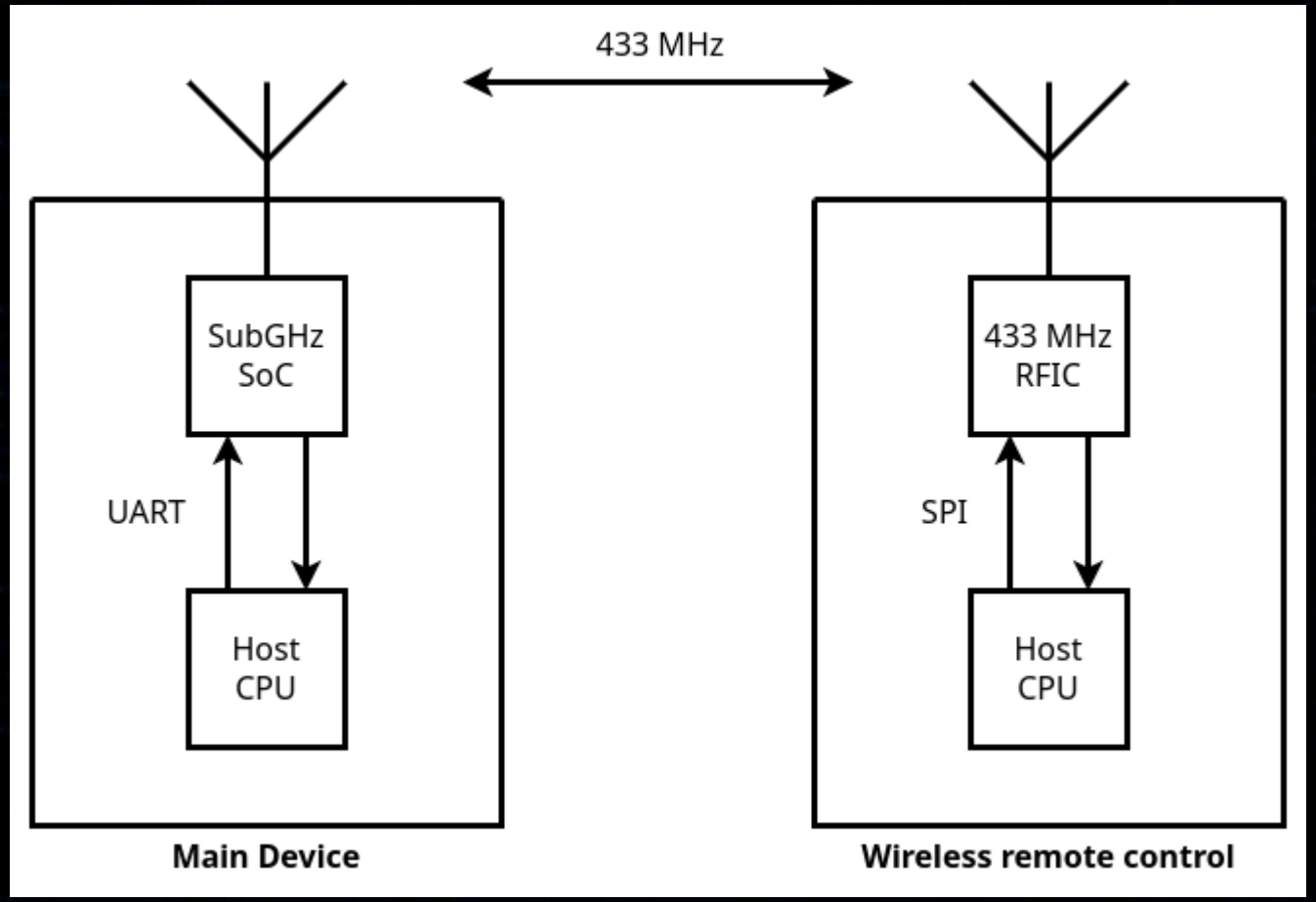


Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A

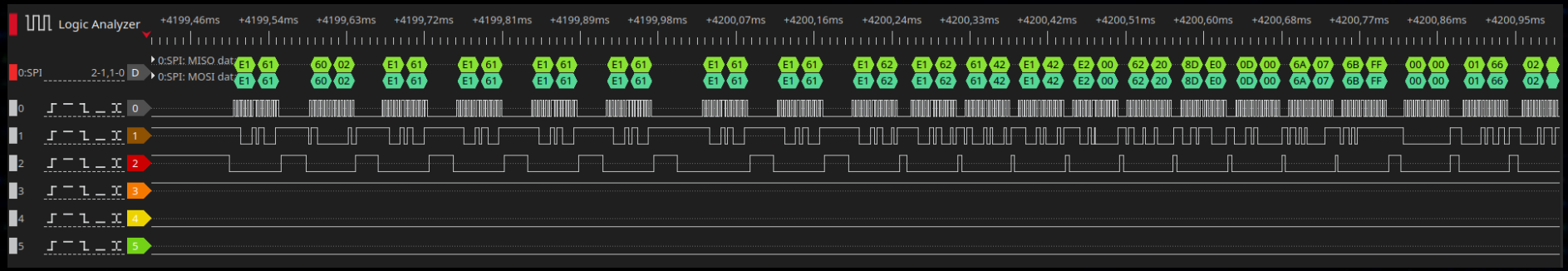
Реализация трансивера на боевом примере

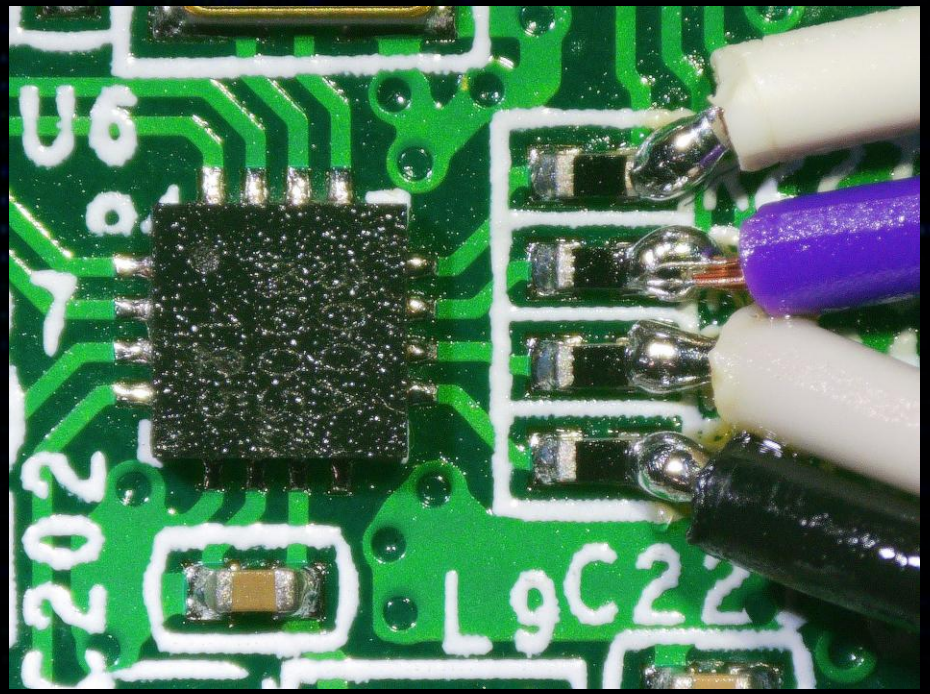
Подопытный



Реализация трансивера на боевом примере

Подопытный – Инициализация по SPI





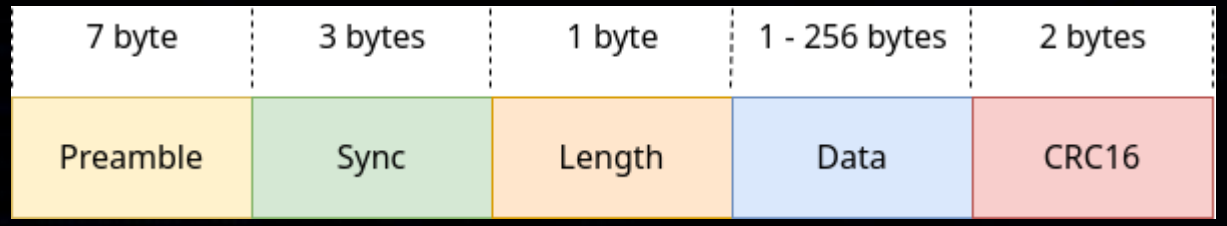
```
Command(type=write, reg_addr=38, value=00010010)
Command(type=write, reg_addr=39, value=00000111)
Command(type=write, reg_addr=3a, value=00000000)
Command(type=write, reg_addr=3b, value=01010101)
Command(type=write, reg_addr=3c, value=00000100)
Command(type=write, reg_addr=3d, value=00000000)
Command(type=write, reg_addr=3e, value=00000000)
Command(type=write, reg_addr=3f, value=00000000)
Command(type=write, reg_addr=40, value=00000000)
Command(type=write, reg_addr=41, value=00000000)
Command(type=write, reg_addr=42, value=11001111)
...
```

Реализация трансивера на боевом примере

Подопытный – Параметры канала

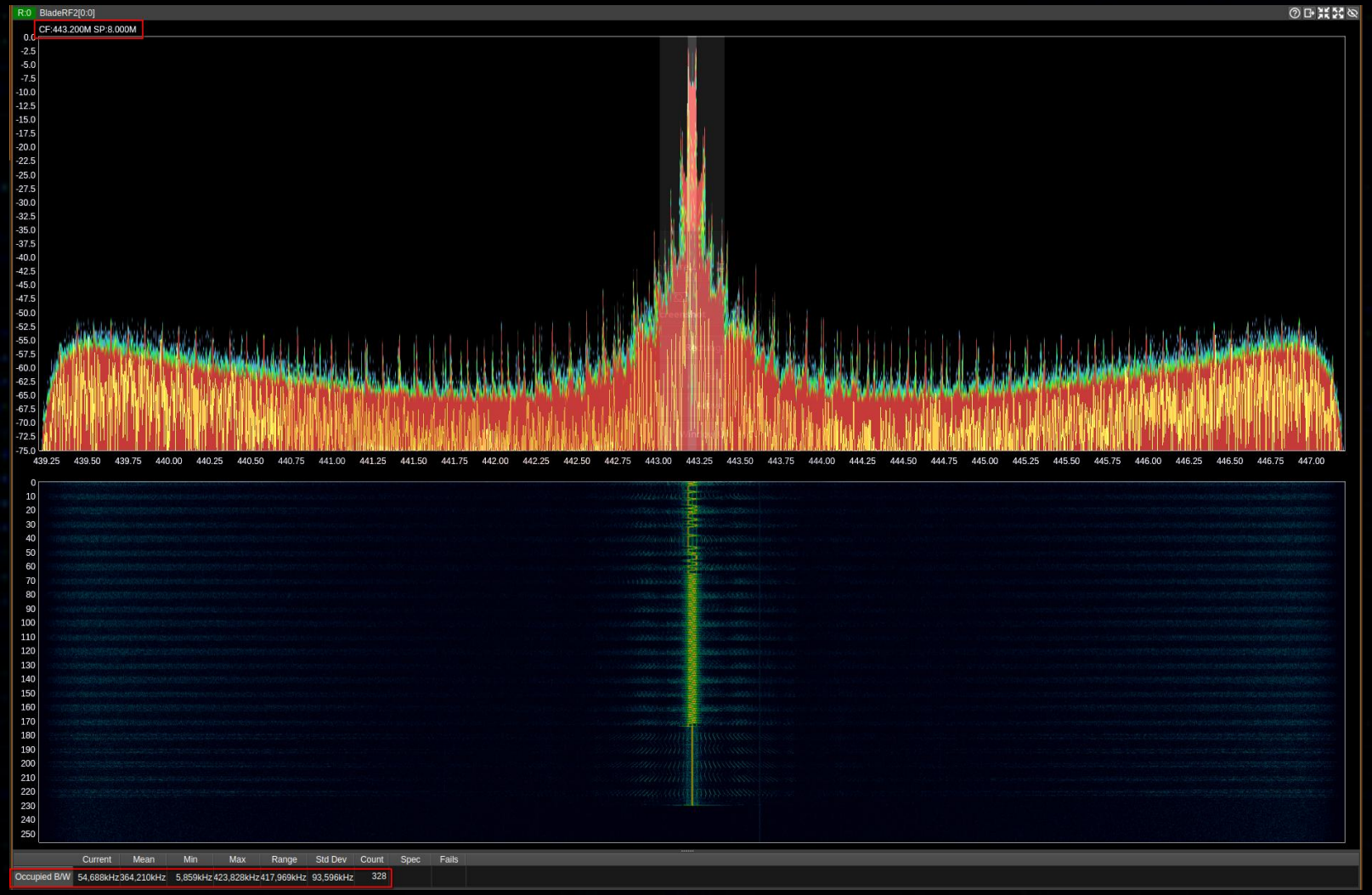
Addr	R/W	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Function
0x3B	R/W	CUS_PKT3			RX_PREAM_			PREAM_LENG_UNIT	DATA_MODE[1:0]		Baseband Bank
0x39	R/W	CUS_PKT2				TX_PREAM_SIZE[7:0]					
0x3A	R/W	CUS_PKT3				TX_PREAM_SIZE					
0x3B	R/W	CUS_PKT4				[15:8]					
0x3C	R/W	CUS_PKT5	RESV		SYNC_TOL[2:0]			SYNC_SIZE[2:0]		SYNC_MAN_EN	
0x3D	R/W	CUS_PKT6				SYNC_VALUE [7:0]					
0x3E	R/W	CUS_PKT7				SYNC_VALUE [15:8]					
0x3F	R/W	CUS_PKT8				SYNC_VALUE [23:16]					
0x40	R/W	CUS_PKT9				SYNC_VALUE [31:24]					
0x41	R/W	CUS_PKT10				SYNC_VALUE [39:32]					
0x42	R/W	CUS_PKT11				SYNC_VALUE [47:40]					
0x43	R/W	CUS_PKT12									
0x44	R/W	CUS_PKT13				SYNC_VALUE [63:56]					
0x45	R/W	CUS_PKT14	RESV		PAYLOAD_LENG		AUTO_ACK_EN	NODE_LENG_POS_SEL	PAYLOAD_BIT_ORDER	PKT_TYPE	
0x46	R/W	CUS_PKT15				PAYLOAD_LENG[7:0]					
0x47	R/W	CUS_PKT16	RESV	RESV	NODE_FIRST_EN	FNODE_DIFF_MASK		NODE_SIZE[1:0]		NODE_DET_MODE	
0x48	R/W	CUS_PKT17				NODE_VALUE[7:0]					
0x49	R/W	CUS_PKT18				NODE_VALUE[15:8]					
0x4A	R/W	CUS_PKT19				NODE_VALUE[23:16]					
0x4B	R/W	CUS_PKT20				NODE_VALUE[31:24]					
0x4C	R/W	CUS_PKT21	FEC_TYPE	FEC_EN	CRC_BYTE_SWAP	CRC_BIT_INV	CRC_RANGE	CRC_TYPE[1:0]		CRC_EN	
0x4D	R/W	CUS_PKT22				CRC_SEED[7:0]					
0x4E	R/W	CUS_PKT23				CRC_SEED [15:8]					
0x4F	R/W	CUS_PKT24	CRC_BIT_SWDER	WHITEN_SEED[6]	WHITEN_SEED_TYPE	WHITEN_TYPE[1:0]		WHITEN_EN	MANCH_TYPE	MANCH_EN	
0x50	R/W	CUS_PKT25				WHITEN_SEED[7:0]					
0x51	R/W	CUS_PKT26	RESV	RESV	RESV	RESV	RESV	RESV	TX_PREFIX_TYPE		
0x52	R/W	CUS_PKT27				TX_PKT_NUM [7:0]					
0x53	R/W	CUS_PKT28				TX_PKT_GAP [7:0]					
0x54	R/W	CUS_PKT29	FIFO_AUTO_RES_EN			FIFO_TH[6:0]					

- Frequency: 433 MHz
- Modulation: GFSK
- Symbol speed: 1200 Hz (baud)
- FEC: не используется
- Scrambling: не используется
- Preamble: [0x55] * 7
- Sync: 0x5519cf



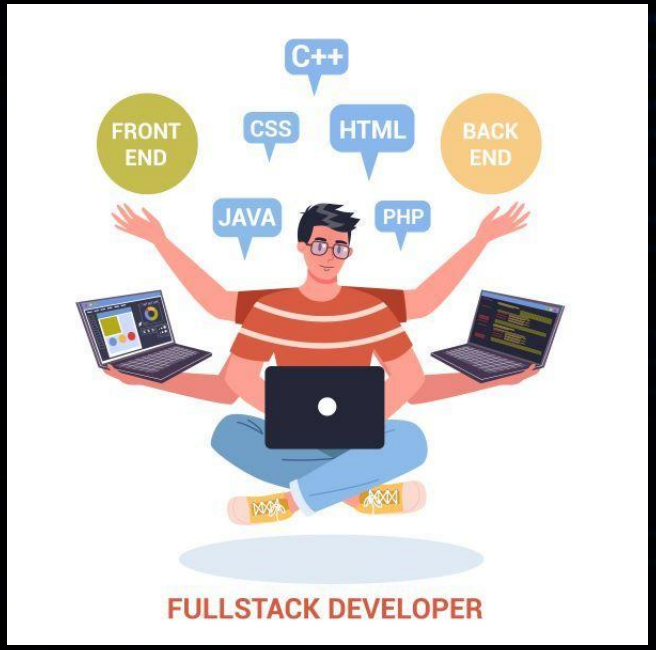
Реализация трансивера на боевом примере

Подопытный – Реальная рабочая частота



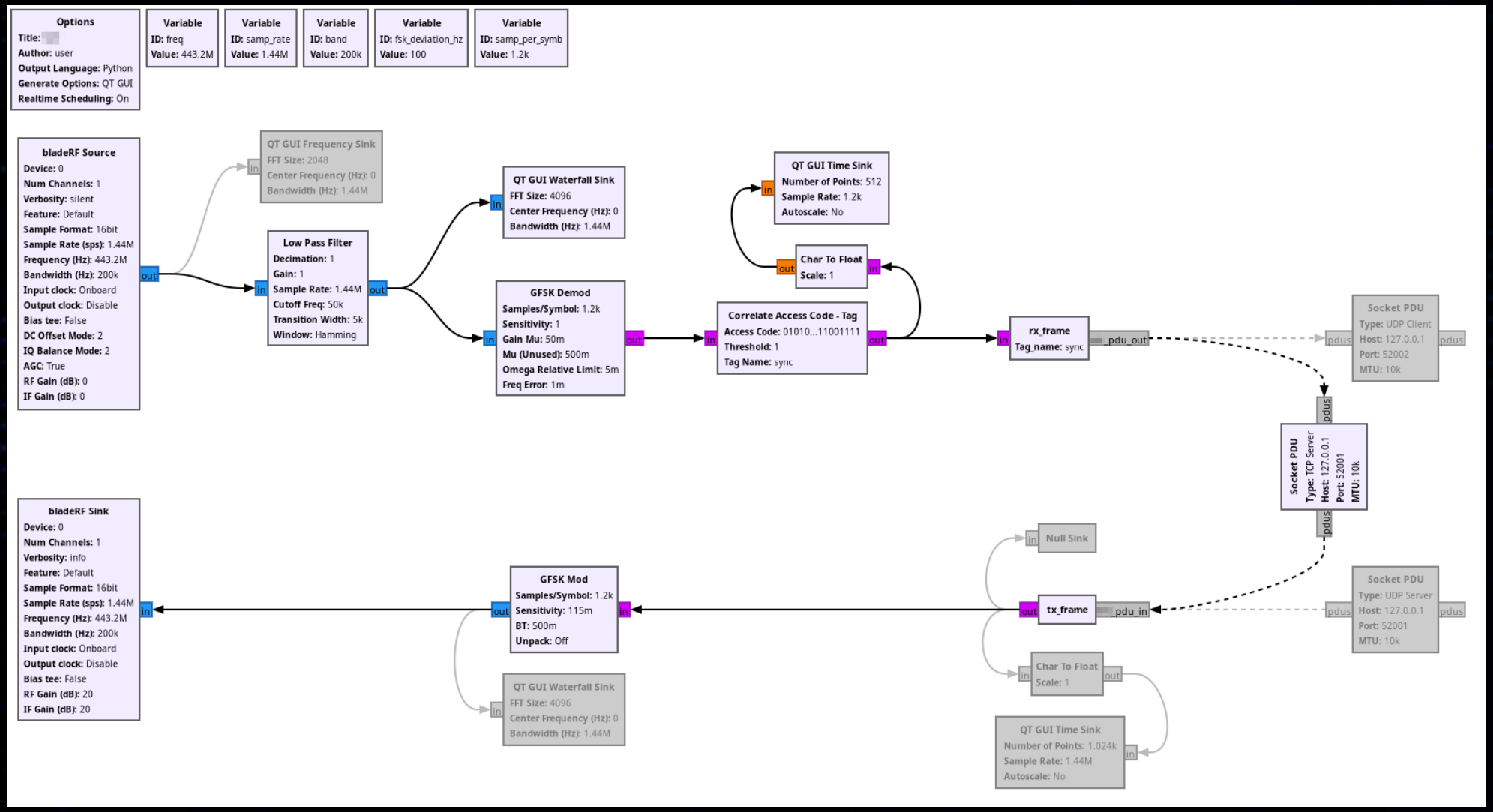
~~433,05 - 434,79 MHz~~

443.2 MHz



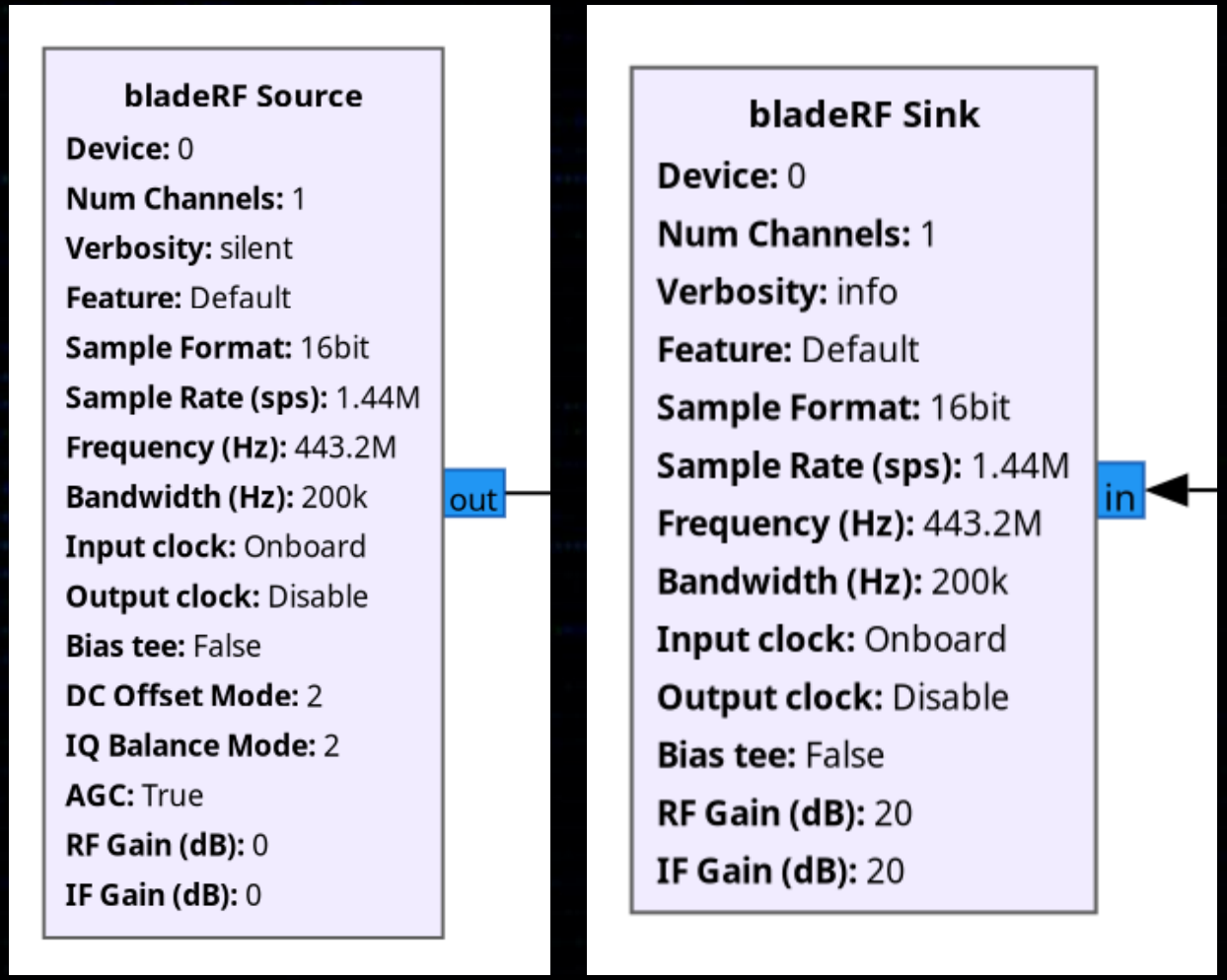
Реализация трансивера на боевом примере

Проект GNU Radio



Реализация трансивера на боевом примере

GNU Radio – Blocks Classification



По расположению_

- Source
- Sink
- Intermediate

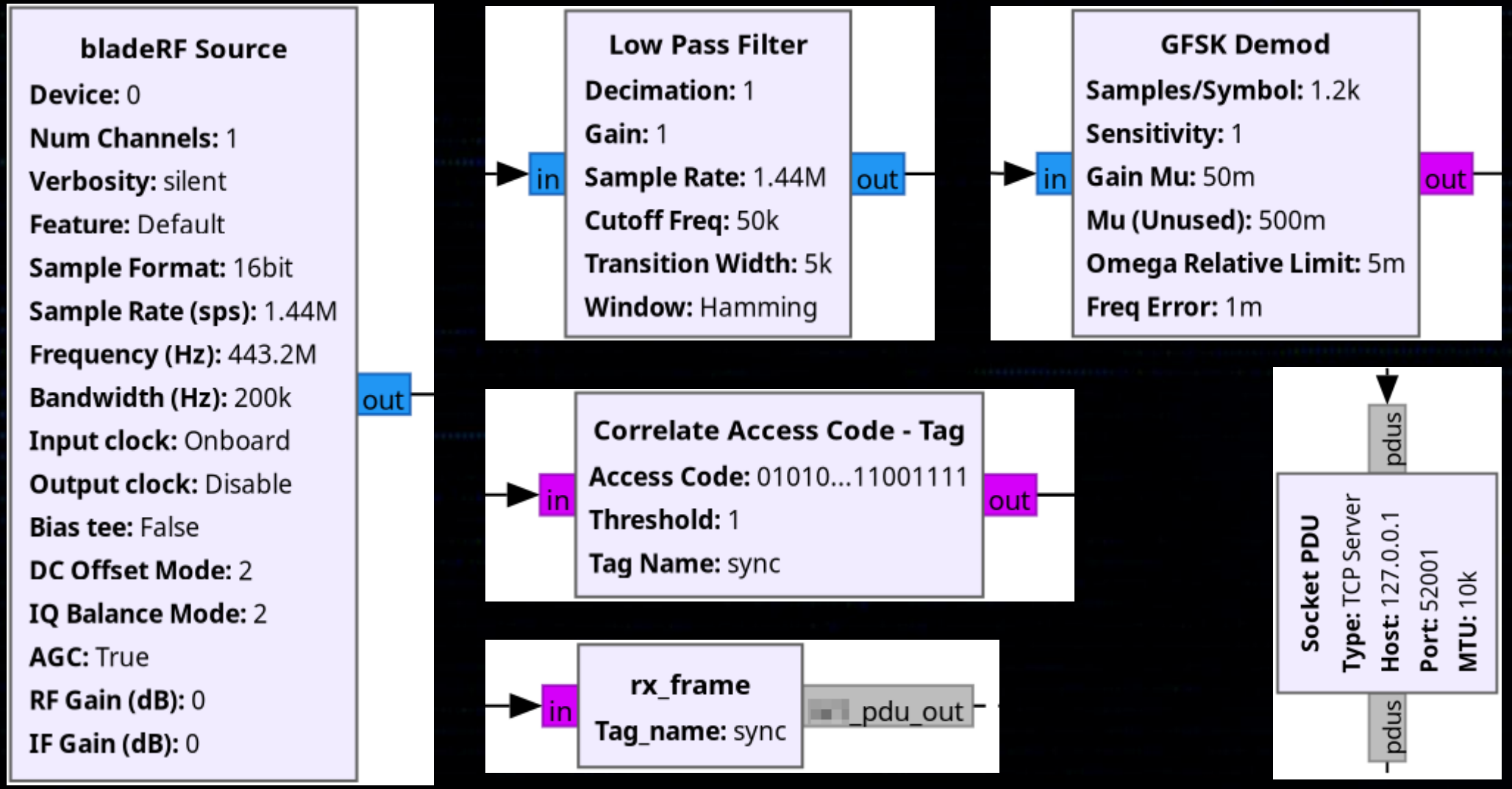
По синхронности_

- Basic (a.k.a. General) Blocks (N:M)
- Synchronous Blocks (1:1)
- Decimation Blocks (N:1)
- Interpolation Blocks (1:M)

https://wiki.gnuradio.org/index.php/Types_of_Blocks

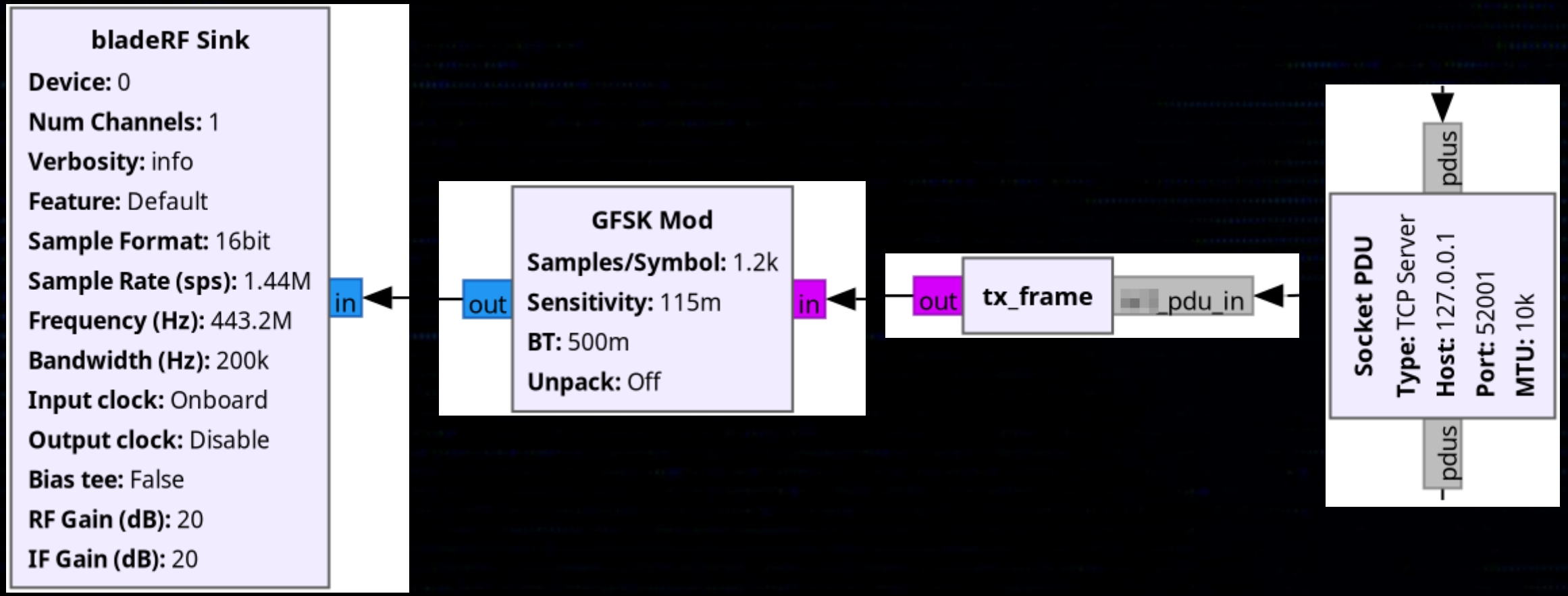
Реализация трансивера на боевом примере

Проект GNU Radio – Приёмный тракт



Реализация трансивера на боевом примере

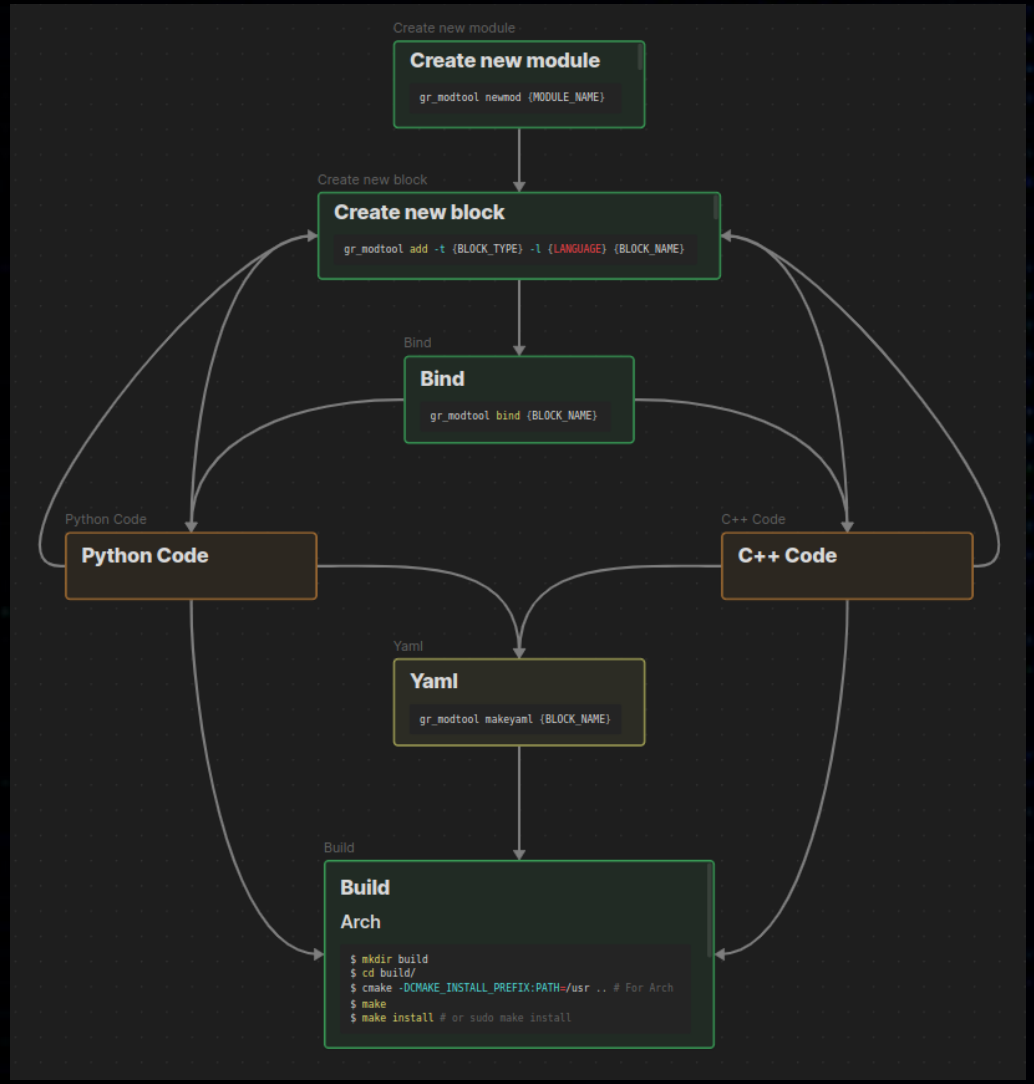
Проект GNU Radio – Передающий тракт



Реализация трансивера на боевом примере

OutOfTreeModules – gr_modtool

- gr_modtool newmod {MODULE_NAME} - создаст структуру проекта с директориями и `CMakeLists.txt`
- gr_modtool add -t {BLOCK_TYPE} -l {LANGUAGE} {BLOCK_NAME} - создаст исходные файлы блока с уже описаным шаблоном, в котором прям в комментах указано, что необходимо добавить минимально, чтобы все успешно скомпилировалось; добавит исходные файлы в `CMakeLists.txt`
- gr_modtool bind {BLOCK_NAME} - создаст бинды для Python с помощью библиотеки pybind11, чтобы блок, написанный на C++ мог быть вызван из Python кода; если блок изначально на Python, то в данном шаге нет необходимости
- Пишем реализацию своего блока - подробнее об этом далее
- gr_modtool makeyaml {BLOCK_NAME} - создаст yaml файл с правильной структурой, в котором необходимо будет объявить параметры вашего нового блока для корректного отображения в GNU Radio Companion
- сборка `cmake` и установка, как любого другого дополнительного модуля, написанного сообществом



Реализация трансивера на боевом примере

Class rx_frame_impl

```
17  enum rx_state {SYNC, RECV_LEN, RECV_DATA, RECV_CRC};
18
19  class rx_frame_impl : public rx_frame
20  {
21  private:
22      pmt::pmt_t d_tag_name;
23      rx_state state = SYNC;
24      int frame_size;
25      std::vector<uint8_t> frame_data;
26      int frame_crc;
27
28  public:
29      rx_frame_impl(const std::string& tag_name);
30      ~rx_frame_impl();
31
32      // Where all the action really happens
33      int work(int noutput_items,
34               gr_vector_const_void_star& input_items,
35               gr_vector_void_star& output_items);
36  };
```

Реализация трансивера на боевом примере

Class rx_frame_impl - Constructor

```
25  rx_frame_impl::rx_frame_impl(const std::string& tag_name)
26      : gr::sync_block("rx_frame",
27                       gr::io_signature::make(
28                           1 /* min inputs */, 1 /* max inputs */, sizeof(input_type)),
29                       gr::io_signature::make(0,0,0)) // No regular output
30  {
31      d_tag_name = pmt::string_to_symbol(tag_name);
32
33      message_port_register_out(pmt::mp("    _pdu_out")); // Create message port
34
35      frame_size = 0;
36      frame_data.clear();
37      frame_crc = 0;
38  }
```

Реализация трансивера на боевом примере

Class rx_frame_impl – Method work()

```
45  int rx_frame_impl::work(int noutput_items,
46                          gr_vector_const_void_star& input_items,
47                          gr_vector_void_star& output_items)
48  {
49      auto in = static_cast<const input_type*>(input_items[0]);
50      std::vector<tag_t> tags;
51      std::string hex_str;
52
53      switch (state)
54      {
55  >   case SYNC: {···
82
83  >   case RECV_LEN: {···
103
104 >   case RECV_DATA: {···
132
133 >   case RECV_CRC: {···
177
178     default: {
179         break;
180     }
181     }
182
183     // Tell runtime system how many output items we produced.
184     return noutput_items;
185 }
```

7 byte	3 bytes	1 byte	1 - 256 bytes	2 bytes
Preamble	Sync	Length	Data	CRC16

Реализация трансивера на боевом примере

Class tx_frame_impl

```
17  class tx_frame_impl : public tx_frame
18  {
19  private:
20      uint32_t d_itemsize;
21      std::list<pmt::pmt_t> d_pdu_queue;
22      uint32_t d_max_queue_size;
23      uint32_t d_drop_ctr;
24
25  public:
26      tx_frame_impl();
27      ~tx_frame_impl();
28
29      // Where all the action really happens
30      int work(int noutput_items,
31               gr_vector_const_void_star& input_items,
32               gr_vector_void_star& output_items);
33
34      void store_pdu(pmt::pmt_t msg);
35  };
```

Реализация трансивера на боевом примере

Class tx_frame_impl – Constructor

```
23  tx_frame_impl::tx_frame_impl()
24      : gr::sync_block("tx_frame",
25                       gr::io_signature::make(0, 0, 0),
26                       gr::io_signature::make(
27                           1 /* min outputs */, 1 /*max outputs */, sizeof(output_type))),
28                       d_max_queue_size(64),
29                       d_itemsize(sizeof(output_type))
30  {
31      size_t output_buf_size = 0;
32      output_buf_size += 256;  // max frame size (is the frame len field realy 1 byte?)
33      output_buf_size += 2;    // CRC size
34      output_buf_size += 3;    // Syncronization size
35      output_buf_size += 15;   // Preamble size
36
37      this->set_min_output_buffer(8 * output_buf_size);
38
39      this->d_pdu_queue.clear();
40
41      this->message_port_register_in(pmt::mp("[redacted]_pdu_in"));
42      this->set_msg_handler(pmt::mp("[redacted]_pdu_in"), [this](pmt::pmt_t msg) { this->store_pdu(msg); });
43  }
```

Реализация трансивера на боевом примере

OutOfTreeModules – Обобщение и рекомендации

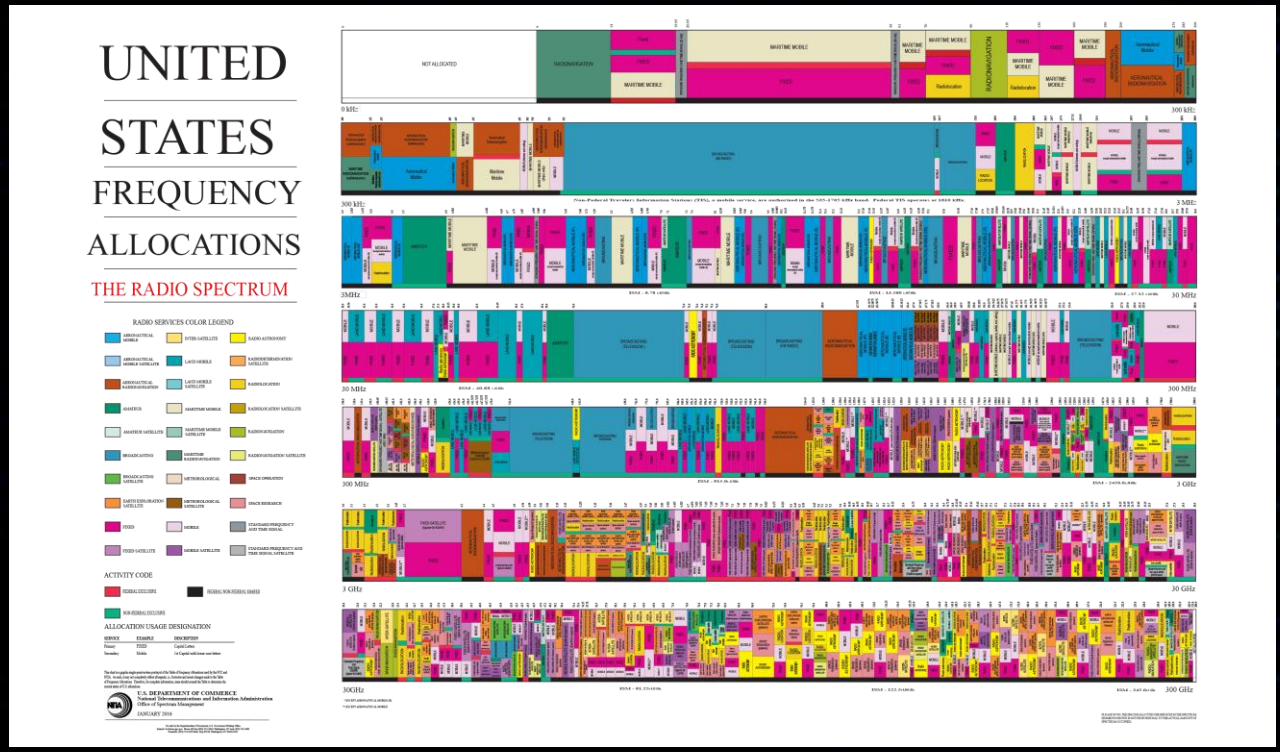
- Многое уже сделано до вас, используйте готовые блоки по-максимуму
- Все односложные модуляции можно принять и отправить штатными блоками GNU Radio без каких-либо заморочек
- Некоторые комплексные модуляции, при должной сноровке, тоже можно обрабатывать штатными блоками
- При добавлении собственных блоков не надо писать свой код с нуля, вместо этого генерируйте шаблоны с помощью утилиты gr_modtool
- При генерации с помощью gr_modtool выбирайте наиболее подходящий тип блока в зависимости от его предполагаемого положения на общей схеме: начальный, конечный или промежуточный с одинаковым или разным соотношением потребления и генерации данных
- При реализации rx_frame блоков используйте тип Sink со стандартным входным FIFO и выходным Message портом
- При реализации tx_frame блоков используйте тип Source с входным Message портом и стандартным выходным FIFO
- Это позволит вам отдавать и забирать данные снаружи GNU Radio с помощью блока "Socket PDU" и тем самым легко интегрировать популярные инструменты, например, Scapy
- При реализации rx_frame используйте state-машину в методе work() и не плодите лишних буферов, вместо этого используйте входной FIFO, выравнивая данные от стейта к стейту
- Из FIFO не обязательно забирать всё сразу, планировщик достаточно шустрый - успеете
- При реализации tx_frame в обработчике message порта копируйте данные в очередь; в методе work() забирайте из очереди, формируйте кадр, разворачивайте в битовую последовательность и отдавайте на выходной FIFO
- По возможности абстрагируйтесь от Phy уровня в GNU Radio, все что выше, начиная с Link уровня, отдавайте в сокет; это позволит не закопаться в реализацию и даст больше гибкости при атаках на сам протокол

Agenda

- Мотивация
- Как общаться с «пациентом»
- Теория: SDR, DSP, Signals
- Технический анализ сигналов
- Реализация трансивера на боевом примере
- Идеи для исследований
- Q&A

Идеи для исследований

- Internet of Things (IoT)
- Задавайте правильные вопросы заказчикам, смотрите в спектр и по сторонам
- 433 МГц – это не только keyfob от ворот и машины
- Нелицензируемые диапазоны для SRD устройств – только вершина айсберга
- fccid.io
- r/FCCID
- СМОТРИТЕ В СПЕКТР, само устройство вам не всегда нужно ;)



OFFZONE
2025

Q&A