

МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств
(ТС и ВС)

Отчет по производственной практике
по дисциплине
SDR

по теме:

Знакомство с библиотеками Soapy SDR, Libiio для работы с Adalm Pluto SDR. Инициализация
SDR-устройства. Работа с буфером: получение цифровых IQ-отсчетов

Студент:
Группа ИА-331

К.А Любимов

Предподаватели:
Лектор
Практик
Практик

Калачиков А.А
Ахпашев А.В
Попович И.А

Новосибирск 2025 г.

1 ЗНАКОМСТВО С БИБЛИОТЕКАМИ SOAPY SDR, LIBIPU ДЛЯ РАБОТЫ С ADALM PLUTO SDR. ИНИЦИАЛИЗАЦИЯ SDR-УСТРОЙСТВА. РАБОТА С БУФЕРОМ: ПОЛУЧЕНИЕ ЦИФРОВЫХ IQ-ОТСЧЕТОВ

| Ссылка на GitHub

Цель практики:

Понять как работает приёмник. Написать на C++ программу для SDR. Передать свой сигнал и принять его, чтобы проверить как всё работает.

Краткие теоретические сведения

Начнем с описания сэмпла в рамках работы с Adalm Pluto.

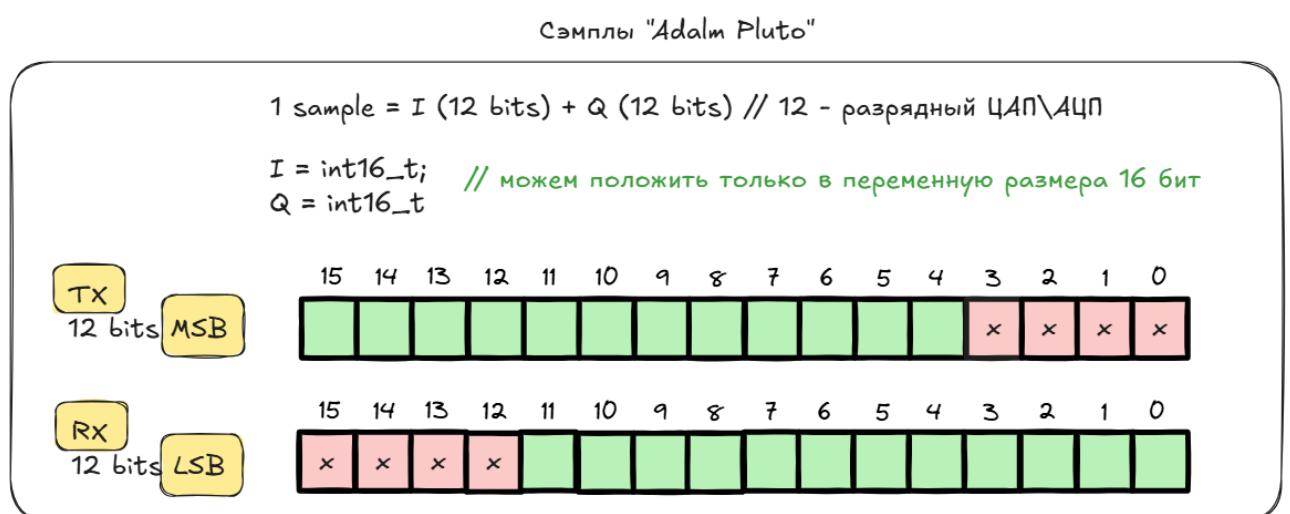


Рис. 1 — Сэмплы Adalm Pluto

Буфер, его структура, выделение памяти под буфера (для хранения сэмплов TX, RX):

Листинг 1.1 — Работа с буферами SDR

```
1 // Получение MTU
2 size_t rx_mtu = SoapySDRDevice_getStreamMTU(sdr, rxStream);
3 size_t tx_mtu = SoapySDRDevice_getStreamMTU(sdr, txStream);
4
5 // Выделение памяти под буфера
6 int16_t tx_buff[2*tx_mtu];
7 int16_t rx_buffer[2*rx_mtu];
```

Buffers structure

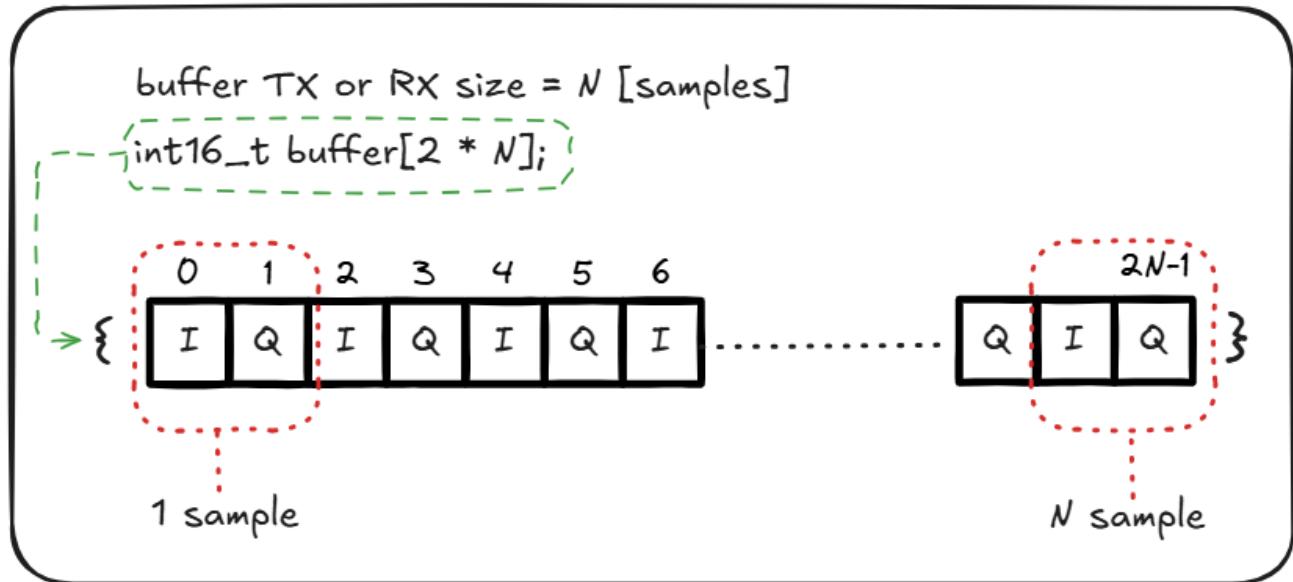


Рис. 2 — Структура буфера

Как происходит чтение сэмплов с SDR

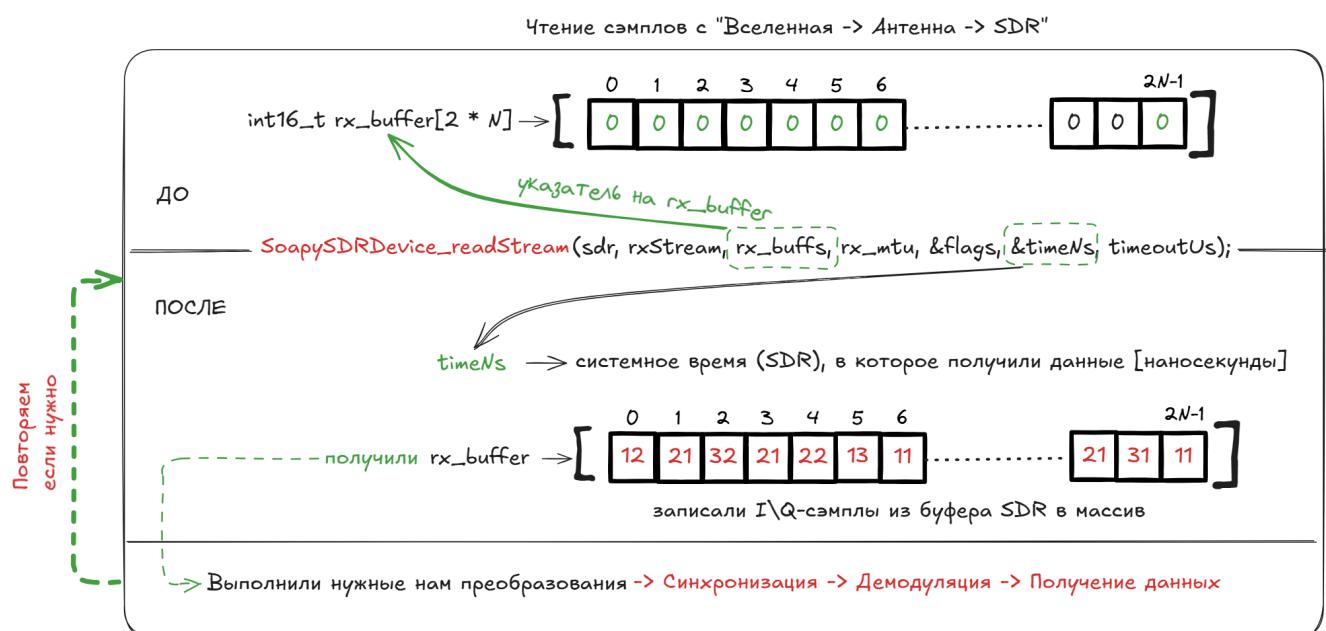


Рис. 3 — Чтение сэмплов

Листинг 1.2 — получения сэмплов с SDR

```

1 const long timeoutUs = 400000;
2 long long last_time = 0;
3 size_t iteration_count = 10;
4
5 for (size_t buffers_read = 0; buffers_read < iteration_count; buffers_read++)
6 {
7     void *rx_buffs[] = {rx_buffer};
8     int flags;           // flags set by receive operation
9     long long timeNs; // timestamp for receive buffer

```

```

10
11     int sr = SoapySDRDeviceReadStream(sdr, rxStream, rx_buffs, rx_mtu, &flags,
12                                         timeNs, timeoutUs);
13
14     printf("Buffer: %lu - Samples: %i, Flags: %i, Time: %lli, TimeDiff: %lli\n",
15            buffers_read, sr, flags, timeNs, timeNs - last_time);
}

```

Как происходит подготовка TX буфера и передача сэмплов на SDR

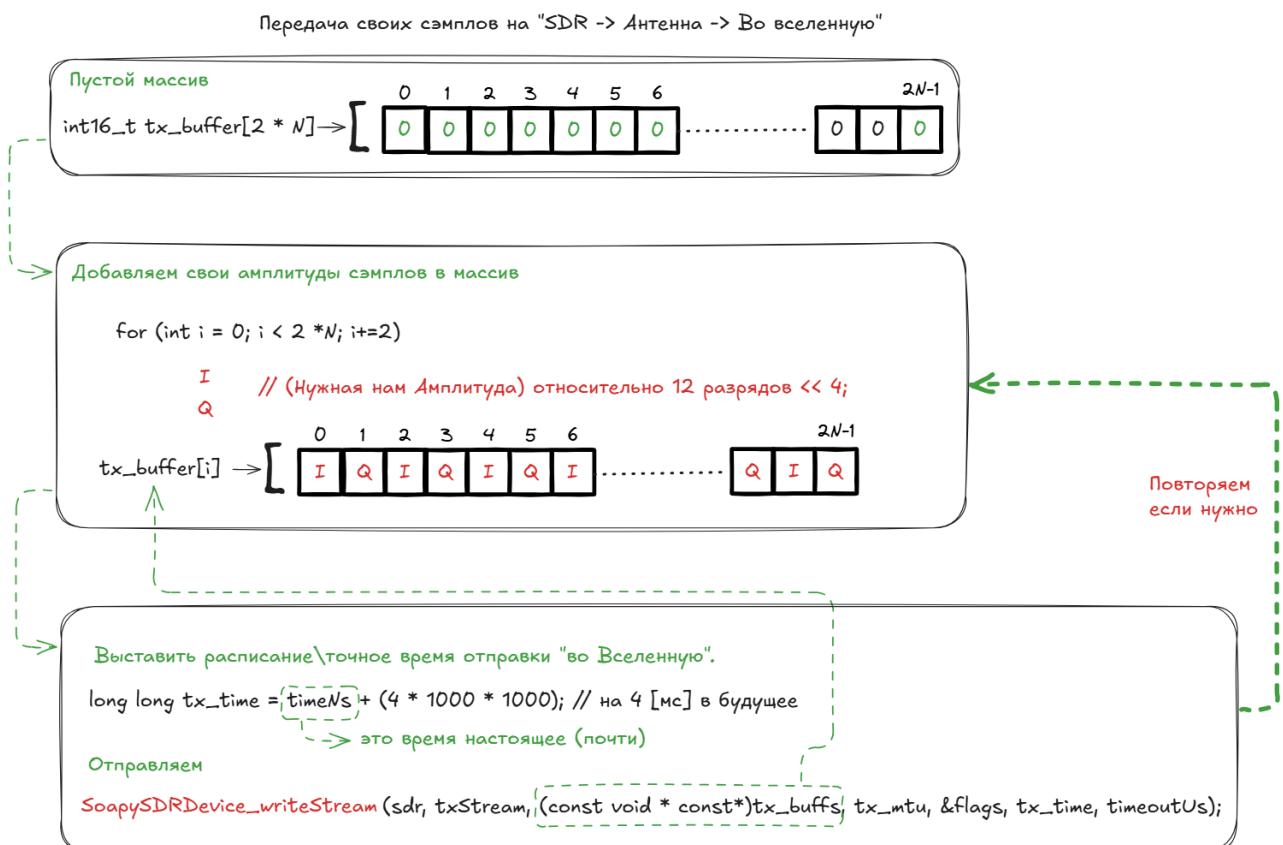


Рис. 4 — Передача сэмплов на SDR

Листинг 1.3 — Подготовка массива для передачи (TX buffer):

```

1  for (int i = 2; i < 2 * tx_mtu; i+=2)
2  {
3      tx_buff[i] = 1500 << 4;    // I
4      tx_buff[i+1] = 1500 << 4; // Q
5  }
6  //prepare fixed bytes in transmit buffer
7  //we transmit a pattern of FFFF FFFF [TS_0]00 [TS_1]00 [TS_2]00 [TS_3]00 [TS_4]00
8  // [TS_5]00 [TS_6]00 [TS_7]00 FFFF FFFF
9  //that is a flag (FFFF FFFF) followed by the 64 bit timestamp, split into 8 bytes
10 // and packed into the lsb of each of the DAC words.
11 //DAC samples are left aligned 12-bits, so each byte is left shifted into place
12 for(size_t i = 0; i < 2; i++)
13 {
14     tx_buff[0 + i] = 0xffff;
15     // 8 x timestamp words
16     tx_buff[10 + i] = 0xffff;
}

```

```

17 last_time = timeNs;
18
19 long long tx_time = timeNs + (4 * 1000 * 1000); // на 4 мс\[\\] в будущее
20
21 for(size_t i = 0; i < 8; i++)
22 {
23     uint8_t tx_time_byte = (tx_time >> (i * 8)) & 0xff;
24     tx_buff[2 + i] = tx_time_byte << 4;
25 }
26
27 void *tx_buffs[] = {tx_buff};
28 if( (buffers_read == 2) ){
29     printf("buffers_read: %d\n", buffers_read);
30     flags = SOAPY_SDR_HAS_TIME;
31     int st = SoapySDRDevice_writeStream(sdr, txStream, (const void * const*)
32                                         tx_buffs, tx_mtu, &flags, tx_time, timeoutUs);
33     if ((size_t)st != tx_mtu)
34     {
35         printf("TX Failed: %i\n", st);
36     }
}

```

Одновременный прием и передача. Ниже представлена схема работы буферов RX и TX:

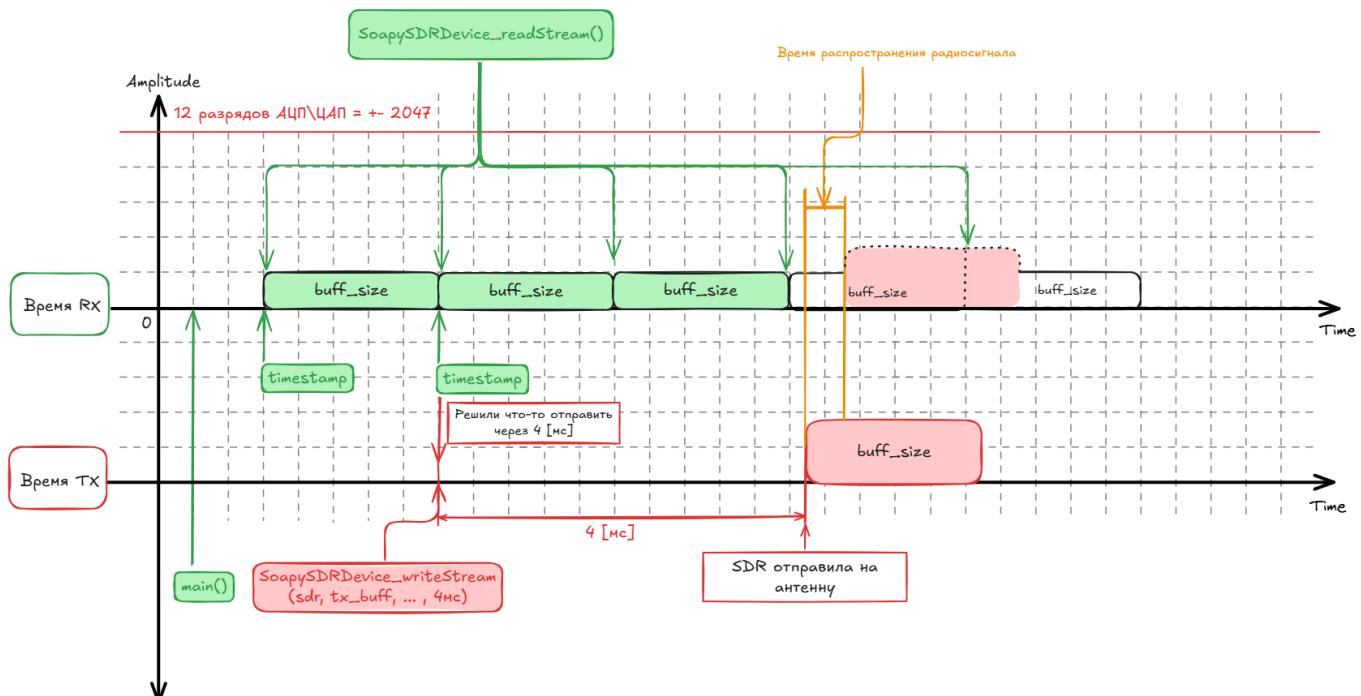


Рис. 5 — Одновременный прием и передача

Код, отражающий работу схемы:

Листинг 1.4 — Подготовка массива для передачи (TX buffer):

```

1 const long timeoutUs = 400000;
2 long long last_time = 0;
3 size_t iteration_count = 10;
4
5 for (size_t buffers_read = 0; buffers_read < iteration_count; buffers_read++)
6 {
7     void *rx_buffs[] = {rx_buffer};
8     int flags;           // flags set by receive operation

```

```

9  long long timeNs; //timestamp for receive buffer
10
11 int sr = SoapySDRDeviceReadStream(sdr, rxStream, rx_buffs, rx_mtu, &flags, &
12   timeNs, timeoutUs);
13
14 printf("Buffer: %lu - Samples: %i, Flags: %i, Time: %lli, TimeDiff: %lli\n",
15   buffers_read, sr, flags, timeNs, timeNs - last_time);
16 last_time = timeNs;
17
18 long long tx_time = timeNs + (4 * 1000 * 1000); // на 4 мс[] в будущее
19
20 for(size_t i = 0; i < 8; i++)
21 {
22     uint8_t tx_time_byte = (tx_time >> (i * 8)) & 0xff;
23     tx_buff[2 + i] = tx_time_byte << 4;
24 }
25
26 void *tx_buffs[] = {tx_buff};
27 if( (buffers_read == 2) ){
28     printf("buffers_read: %d\n", buffers_read);
29     flags = SOAPY_SDR_HAS_TIME;
30     int st = SoapySDRDeviceWriteStream(sdr, txStream, (const void * const*)
31                                         tx_buffs, tx_mtu, &flags, tx_time, timeoutUs);
32     if ((size_t)st != tx_mtu)
33     {
34         printf("TX Failed: %i\n", st);
35     }
36 }
37
38 }

```

Выполнение

Сперва необходимо установить все необходимые библиотеки

Листинг 1.5 — Установка SoapySDR

```

1 sudo apt-get install python3-pip python3-setuptools
2 sudo apt-get install cmake g++ libpython3-dev python3-numpy swig python3-matplotlib
3
4 git clone --branch soapy-sdr-0.8.1 https://github.com/TelecomDep/SoapySDR.git
5
6 cd SoapySDR
7 mkdir build && cd build
8
9 cmake ../
10
11 make -j`nproc` # nproc - количество потоков , например make -j16
12 sudo make install
13 sudo ldconfig

```

Листинг 1.6 — Установка Libiio

```

1 sudo apt-get install libxml2 libxml2-dev bison flex libcdk5-dev cmake
2 sudo apt-get install libusb-1.0-0-dev libaio-dev pkg-config
3 sudo apt install libavahi-common-dev libavahi-client-dev
4
5 git clone --branch v0.24 https://github.com/TelecomDep/libiio.git
6

```

```

7 cd libiio
8 mkdir build && cd build
9 cmake ../
10 make -j`nproc` # nproc - количество потоков , например make -j16
11 sudo make install

```

Листинг 1.7 — Установка LibAD9361

```

1 git clone --branch v0.3 https://github.com/TelecomDep/libad9361-iio.git
2 cd libad9361-iio
3
4 mkdir build && cd build
5
6 cmake ../
7
8 make -j`\nproc\` # nproc - количество потоков , например make -j16
9 sudo make install
10 sudo ldconfig

```

Листинг 1.8 — Установка SoapyPlutoSDR

```

1 git clone --branch sdr_gadget_timestamping https://github.com/TelecomDep/
  SoapyPlutoSDR.git
2 cd SoapyPlutoSDR
3
4 mkdir build && cd build
5
6 cmake ../
7
8 make -j`\nproc\` # nproc - количество потоков , например make -j16
9 sudo make install
10 sudo ldconfig

```

Соответственно подключить их в самом коде

Листинг 1.9 — Подключение библиотек

```

1 #include <SoapySDR/Device.h> // Инициализация устройства
2 #include <SoapySDR/Formats.h> // Типы данных , используемых для записи сэмплов
3 #include <stdio.h>           //printf
4 #include <stdlib.h>          //free
5 #include <stdint.h>
6 #include <complex.h>

```

Листинг 1.10 — Аргументы(в рамках библиотки SoapySDR имеют формат ключ:значение)

```

1 SoapySDRKwargs args = {};
2 SoapySDRKwargs_set(&args, "driver", "plutosdr");
3 if (1) {
4     SoapySDRKwargs_set(&args, "uri", "usb:");
5 } else {
6     SoapySDRKwargs_set(&args, "uri", "ip:192.168.2.1");
7 }
8 SoapySDRKwargs_set(&args, "direct", "1");
9 SoapySDRKwargs_set(&args, "timestamp_every", "1920");
10 SoapySDRKwargs_set(&args, "loopback", "0");
11 SoapySDRDevice /*sdr = SoapySDRDevice_make(&args);
12 SoapySDRKwargs_clear(&args);

```

Мы подключаем SDR к нашему ПК или ноутбуку по USB2.0, в момент запуска программы SDR начинает выполнять запрограммированные действия, в нашем случае мы отправляем десять буферов, которые записываются в бинарный файл с расширением .pcm(Pulse-Code Modulation -

импульсно-кодовая модуляция), который мы далее визуализируем с помощью python, и его библиотек matplotlib, numpy

Листинг 1.11 — Код для визуализации полученных данных с SDR

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 rx = np.fromfile("/home/plutoSDR/sdr/pluto/dev/rx.pcm", dtype=np.int16)
5
6 # Преобразование в комплексные отсчеты (I + j*Q)
7 samples = []
8 for x in range(0, len(rx), 2):
9     samples.append(rx[x] + 1j * rx[x+1])
10
11 samples = np.array(samples)
12
13 ampl = np.abs(samples)
14 phase = np.angle(samples)
15 time = np.arange(len(samples))
16
17 plt.figure(figsize=(10, 8))
18
19 plt.subplot(3, 1, 1)
20 plt.plot(time, ampl)
21 plt.title("Amplitude")
22 plt.ylabel("Amplitude")
23 plt.grid(True)
24
25 plt.subplot(3, 1, 2)
26 plt.plot(time, phase)
27 plt.title("Phase")
28 plt.ylabel("Phase (rad)")
29 plt.grid(True)
30
31 plt.subplot(3, 1, 3)
32 plt.plot(time, np.real(samples), label='I (Real)')
33 plt.plot(time, np.imag(samples), label='Q (Imag)')
34 plt.title("I and Q components")
35 plt.ylabel("Value")
36 plt.xlabel("Time (samples)")
37 plt.legend()
38 plt.grid(True)
39
40 plt.tight_layout()
41 plt.show()
```

Полученные Результаты приведены на графике, через subplot, на первом subplot показана амплитуда полученного сигнала, на втором subplot показана фаза полученного сигнала, на третьем I - реальная и Q - мнимая части.

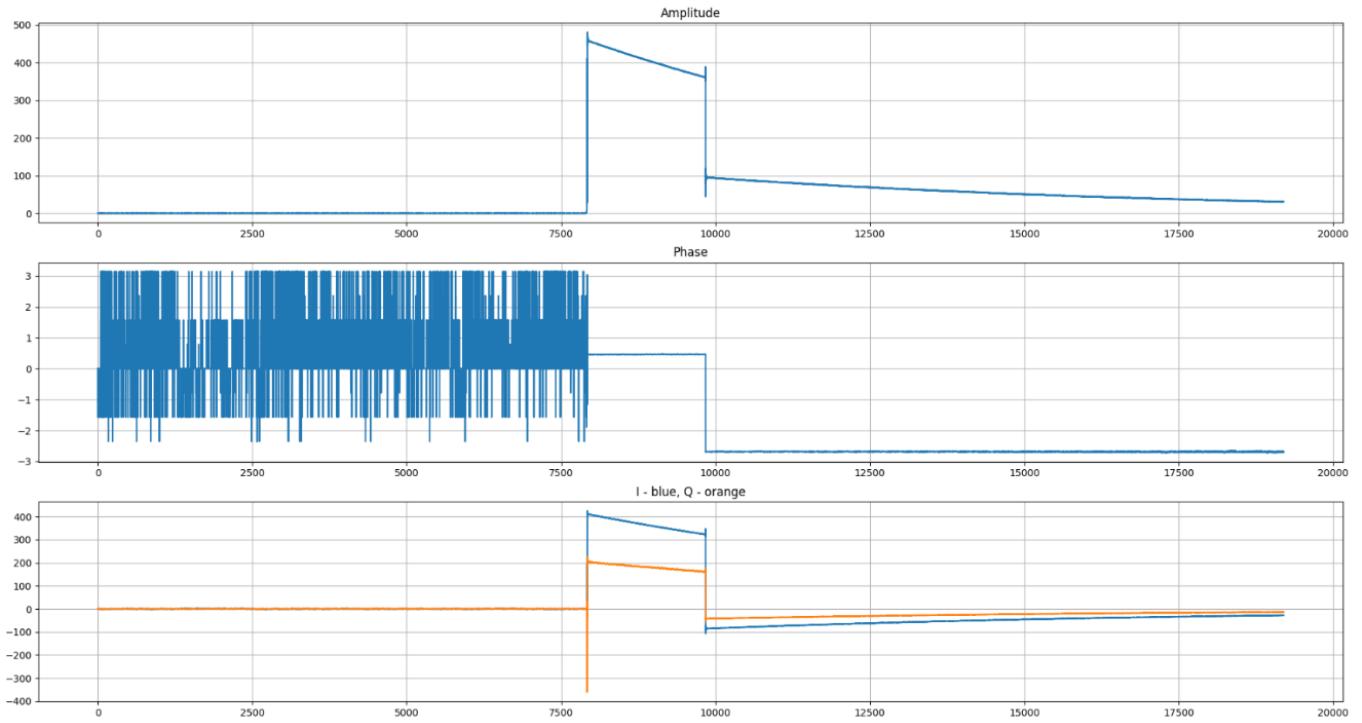


Рис. 6 — Результаты, которые мы получили с SDR

Вывод

В ходе работы я познакомился с минимальной схемой работы Pluto SDR, установил и подключил нужные библиотеки(LibAD9361, Libiio, SoapySDR, SoapyPlutoSDR), отправил сэмплы и получил их, вывел характеристики полученного сигнала на график.