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 *Калачиков А.А*
Практик *Ахпашев А.В*
Практик *Попович И.А*

Новосибирск 2025 г.

1 ЗНАКОМСТВО С БИБЛИОТЕКАМИ SOAPY SDR, LIBIIO 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.

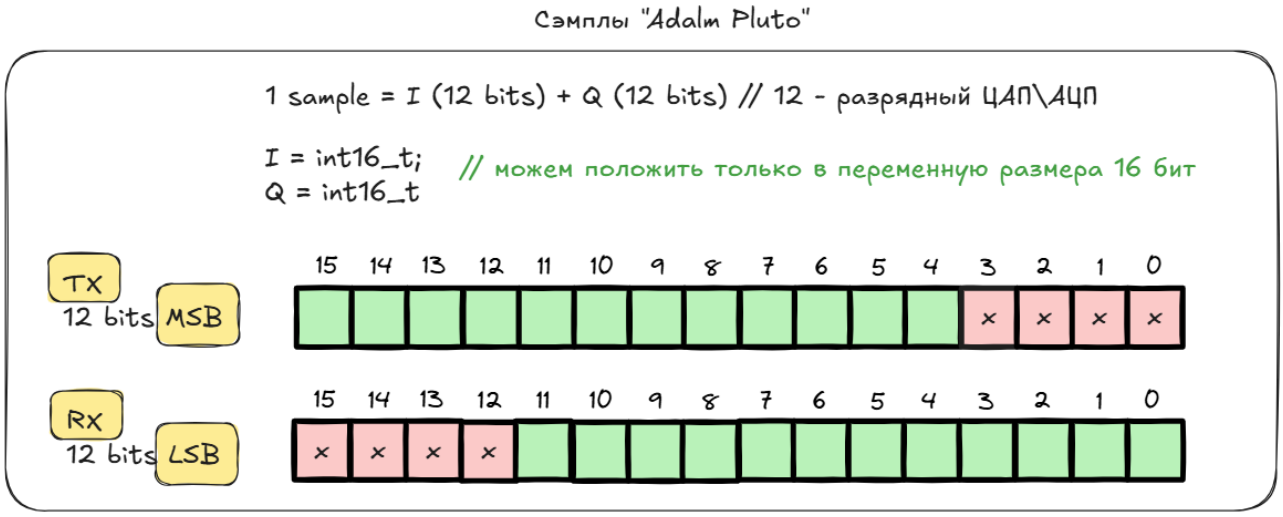

Рис. 1 — Сэмплы Adalm Pluto

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

Листинг 1.1 — Работа с буферами SDR

```
// Получение MTU
size_t rx_mtu = SoapySDRDevice_getStreamMTU(sdr, rxStream);
size_t tx_mtu = SoapySDRDevice_getStreamMTU(sdr, txStream);

// Выделениепамятиподбуферы
int16_t tx_buff[2*tx_mtu];
int16_t rx_buffer[2*rx_mtu];
```

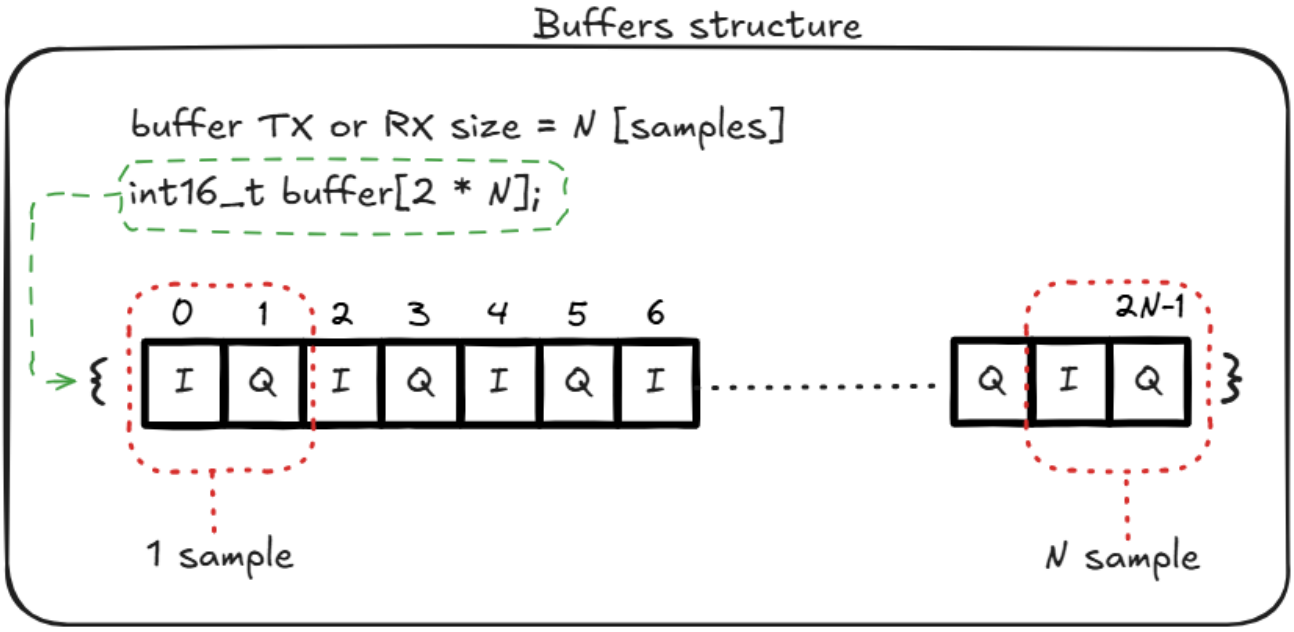

Рис. 2 — Структура буфера

Как происходит чтение сэмплов с SDR

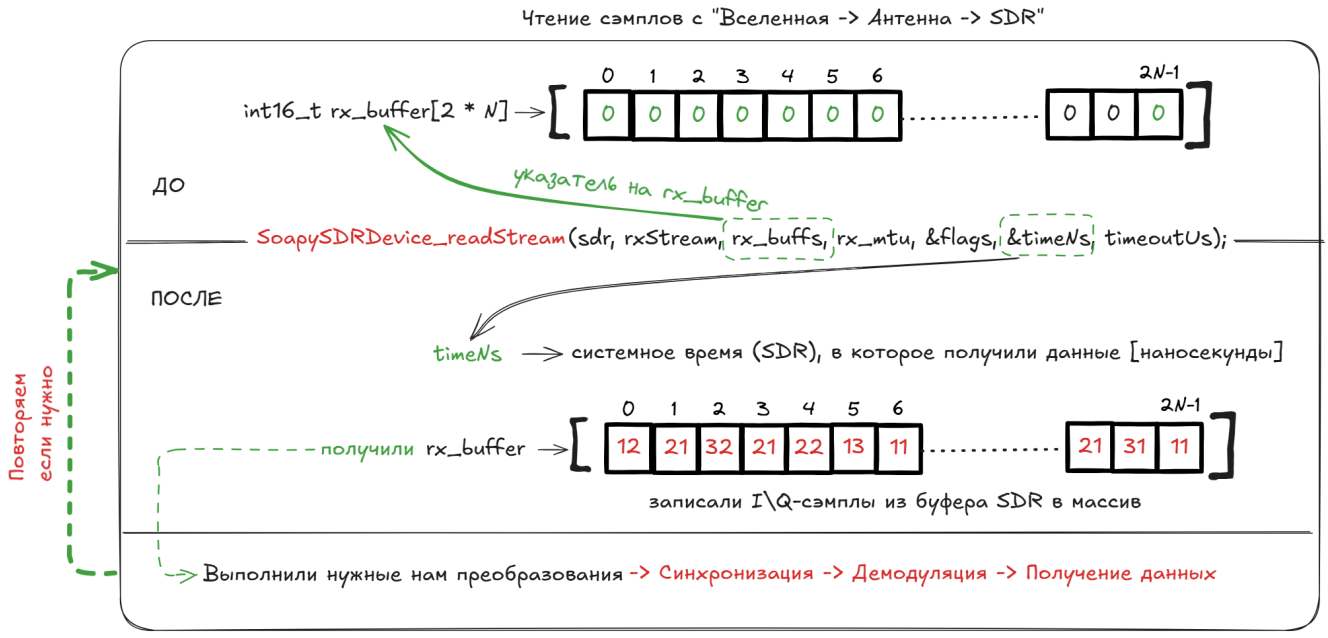

Рис. 3 — Чтение сэмплов

Листинг 1.2 — получения сэмплов с SDR

```
const long  timeoutUs = 400000;
long long last_time = 0;
size_t iteration_count = 10;

for (size_t buffers_read = 0; buffers_read < iteration_count; buffers_read++)
{
    void *rx_buffs[] = {rx_buffer};
    int flags;        // flags set by receive operation
    long long timeNs; //timestamp for receive buffer
```

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

Как происходит подготовка TX буфера и передача сэмплов на SDR

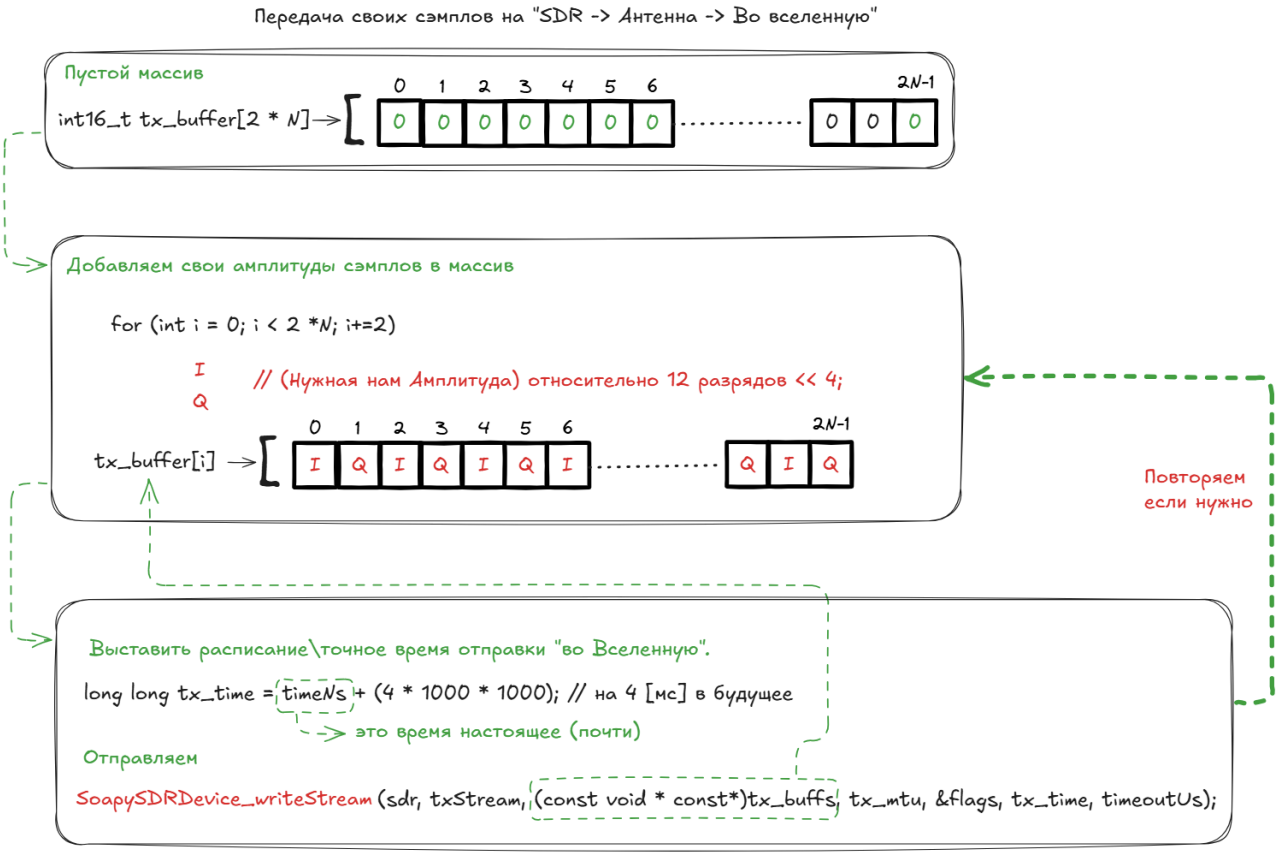

Рис. 4 — Передача сэмплов на SDR

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
            [TS_5]00 [TS_6]00 [TS_7]00 FFFF FFFF
8       //that is a flag (FFFF FFFF) followed by the 64 bit timestamp, split into 8 bytes
            and packed into the lsb of each of the DAC words.
9       //DAC samples are left aligned 12-bits, so each byte is left shifted into place
10      for(size_t i = 0; i < 2; i++)
11      {
12          tx_buff[0 + i] = 0xffff;
13          // 8 x timestamp words
14          tx_buff[10 + i] = 0xffff;
15      }
16
```

```
    last_time = timeNs;

    long long tx_time = timeNs + (4 * 1000 * 1000); // на 4 мс\[\] вбудущее

    for(size_t i = 0; i < 8; i++)
    {
        uint8_t tx_time_byte = (tx_time >> (i * 8)) & 0xff;
        tx_buff[2 + i] = tx_time_byte << 4;
    }

    void *tx_buffs[] = {tx_buff};
    if( (buffers_read == 2) ){
        printf("buffers_read: %d\\n", buffers_read);
        flags = SOAPY_SDR_HAS_TIME;
        int st = SoapySDRDevice_writeStream(sdr, txStream, (const void * const*)
           tx_buffs, tx_mtu, &flags, tx_time, timeoutUs);
        if ((size_t)st != tx_mtu)
        {
            printf("TX Failed: %i\\n", st);
        }
    }
```

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

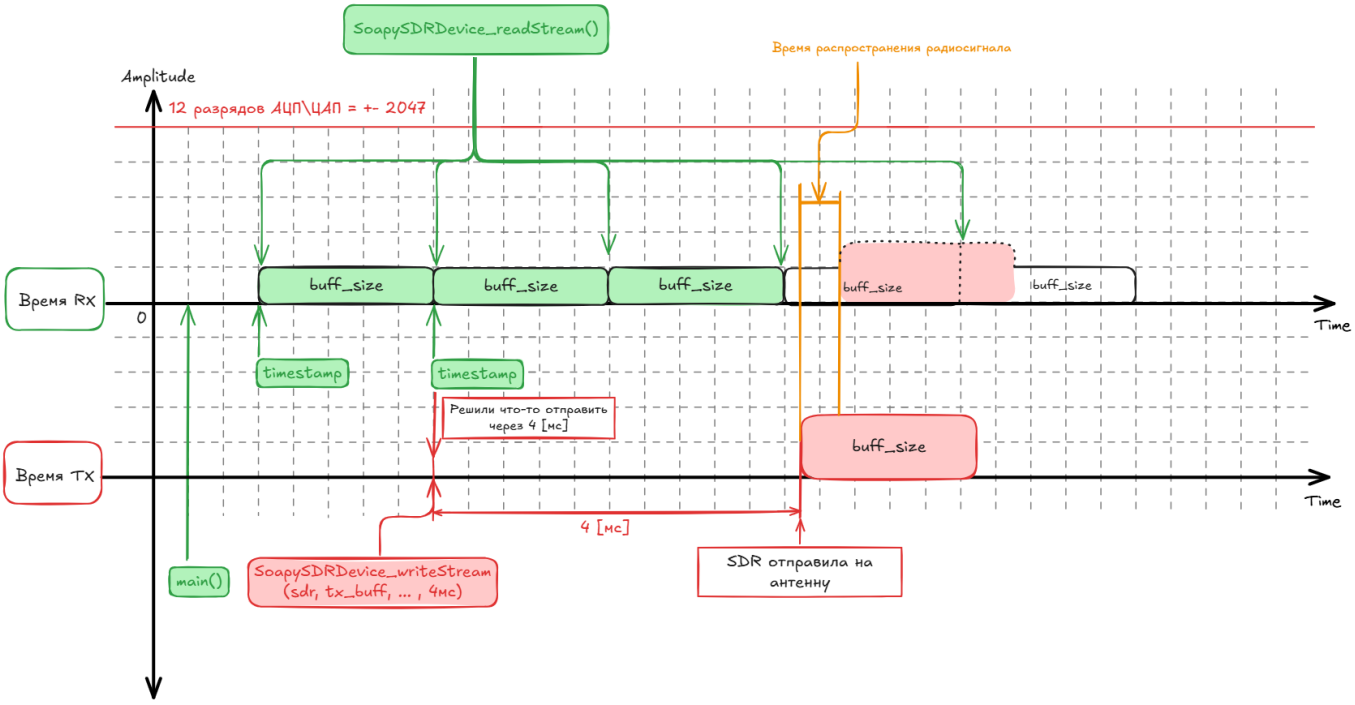

Рис. 5 — Одновременный прием и передача

Код, отражающий работу схемы:

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

```
const long  timeoutUs = 400000;
long long last_time = 0;
size_t iteration_count = 10;

for (size_t buffers_read = 0; buffers_read < iteration_count; buffers_read++)
{
    void *rx_buffs[] = {rx_buffer};
    int flags;        // flags set by receive operation
```

```
    long long timeNs; //timestamp for receive buffer

    int sr = SoapySDRDevice_readStream(sdr, rxStream, rx_buffs, rx_mtu, &flags, &
        timeNs, timeoutUs);

    printf("Buffer: %lu - Samples: %i, Flags: %i, Time: %lli, TimeDiff: %lli\n",
        buffers_read, sr, flags, timeNs, timeNs - last_time);
    last_time = timeNs;

    long long tx_time = timeNs + (4 * 1000 * 1000); // на 4 мс[] вбудущее

    for(size_t i = 0; i < 8; i++)
    {
        uint8_t tx_time_byte = (tx_time >> (i * 8)) & 0xff;
        tx_buff[2 + i] = tx_time_byte << 4;
    }

    void *tx_buffs[] = {tx_buff};
    if( (buffers_read == 2) ){
        printf("buffers_read: %d\n", buffers_read);
        flags = SOAPY_SDR_HAS_TIME;
        int st = SoapySDRDevice_writeStream(sdr, txStream, (const void * const*)
            tx_buffs, tx_mtu, &flags, tx_time, timeoutUs);
        if ((size_t)st != tx_mtu)
        {
            printf("TX Failed: %i\n", st);
        }
    }

}
```

Выполнение

Сперва необходимо установить все необходимые библиотеки

Листинг 1.5 — Установка SoapySDR

```
sudo apt-get install python3-pip python3-setuptools
sudo apt-get install cmake g++ libpython3-dev python3-numpy swig python3-matplotlib

git clone --branch soapy-sdr-0.8.1 https://github.com/TelecomDep/SoapySDR.git

cd SoapySDR
mkdir build && cd build

cmake ../

make -j\`nproc\` # nproc - количествопотоков , например make -j16
sudo make install
sudo ldconfig
```

Листинг 1.6 — Установка Libiio

```
sudo apt-get install libxml2 libxml2-dev bison flex libcdk5-dev cmake
sudo apt-get install libusb-1.0-0-dev libaio-dev pkg-config
sudo apt install libavahi-common-dev libavahi-client-dev

git clone --branch v0.24 https://github.com/TelecomDep/libiio.git

```

```
cd libiio
mkdir build && cd build
cmake ../
make -j`nproc` # nproc - количествопотоков , например make -j16
sudo make install
```

Листинг 1.7 — Установка LibAD9361

```
git clone --branch v0.3 https://github.com/TelecomDep/libad9361-iio.git
cd libad9361-iio

mkdir build && cd build

cmake ../

make -j\`nproc\` # nproc - количествопотоков , например make -j16
sudo make install
sudo ldconfig
```

Листинг 1.8 — Установка SoapyPlutoSDR

```
git clone --branch sdr_gadget_timestamping https://github.com/TelecomDep/
    SoapyPlutoSDR.git
cd SoapyPlutoSDR

mkdir build && cd build

cmake ../

make -j\`nproc\` # nproc - количествопотоков , например make -j16
sudo make install
sudo ldconfig
```

Соответственно подключить их в самом коде

Листинг 1.9 — Подключение бибилиотек

```
#include <SoapySDR/Device.h>   // Инициализацияустройства
#include <SoapySDR/Formats.h>  // Типыданных , используемыхдлязаписисэмплов
#include <stdio.h>             //printf
#include <stdlib.h>            //free
#include <stdint.h>
#include <complex.h>
```

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

```
SoapySDRKwargs args = {};
SoapySDRKwargs_set(&args, "driver", "plutosdr");
if (1) {
    SoapySDRKwargs_set(&args, "uri", "usb:");
} else {
    SoapySDRKwargs_set(&args, "uri", "ip:192.168.2.1");
}
SoapySDRKwargs_set(&args, "direct", "1");
SoapySDRKwargs_set(&args, "timestamp_every", "1920");
SoapySDRKwargs_set(&args, "loopback", "0");
SoapySDRDevice \*sdr = SoapySDRDevice_make(&args);
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

```
import matplotlib.pyplot as plt
import numpy as np

rx = np.fromfile("/home/plutoSDR/sdr/pluto/dev/rx.pcm", dtype=np.int16)

# Преобразованиевкомплексныеотсчеты    (I + j*Q)
samples = []
for x in range(0, len(rx), 2):
    samples.append(rx[x] + 1j * rx[x+1])

samples = np.array(samples)

ampl = np.abs(samples)
phase = np.angle(samples)
time = np.arange(len(samples))

plt.figure(figsize=(10, 8))

plt.subplot(3, 1, 1)
plt.plot(time, ampl)
plt.title("Amplitude")
plt.ylabel("Amplitude")
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(time, phase)
plt.title("Phase")
plt.ylabel("Phase (rad)")
plt.grid(True)

plt.subplot(3, 1, 3)
plt.plot(time, np.real(samples), label='I (Real)')
plt.plot(time, np.imag(samples), label='Q (Imag)')
plt.title("I and Q components")
plt.ylabel("Value")
plt.xlabel("Time (samples)")
plt.legend()
plt.grid(True)

plt.tight_layout()
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.

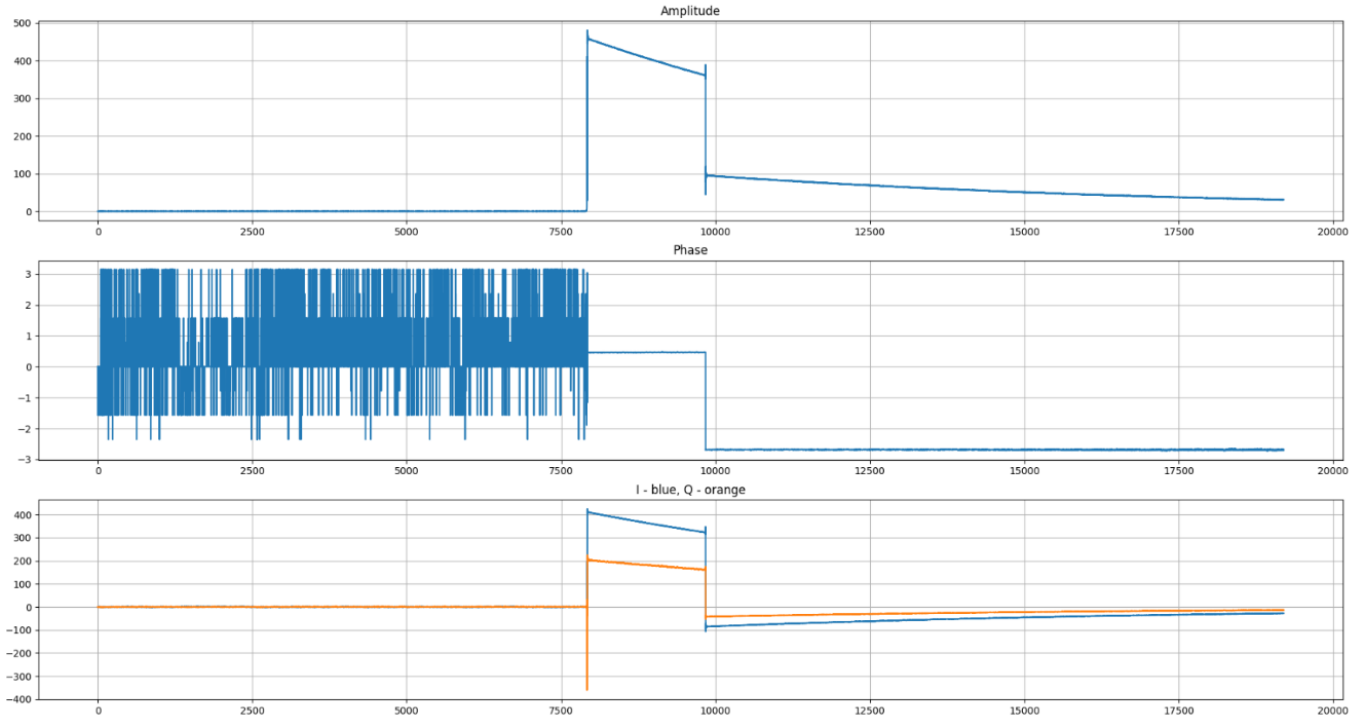

Рис. 6 — Результаты, которые мы получили с SDR

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.