

Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации
федеральное государственное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

09.03.01 Информатика и вычислительная техника
(направление подготовки/специальность)
Программное обеспечение мобильных систем
(профиль/специализация)
Очная
(форма обучения)

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ
(вид практики)

Тип практики Технологическая (проектно-технологическая) практика
на предприятии ООО «Бюро 1440»
(наименование профильной организации/структурного подразделения СибГУТИ)

ТЕМА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

Coming soon

Выполнил:

студент института информатики и вычислительной техники Любимов Кирилл Алексеевич
группа ИА-331

_____ / _____ /
«____» _____ 202__г. *(подпись)* *(ФИО)*

Проверил¹
Руководитель практики от профильной
организации _____ / _____ /
«____» _____ 202__г. *(подпись)* *(ФИО)*

Проверил:
Руководитель практики от СибГУТИ _____ / _____ /
«____» _____ 202__г. *(подпись)* *(ФИО)*

«____» _____ 202__г.
отметка ² _____ «____» _____ 202__г.

Новосибирск 2025

¹ В случае прохождения практики в профильной организации

² Заполняется во время промежуточной аттестации

**План-график проведения
Производственной практики**
вид практики
Любимов Кирилл Алексеевич
Фамилия Имя Отчество студента

института ИВТ, курса 3, гр. ИА-331

Направление: 09.03.01 Информатика и вычислительная техника

Код – Наименование направления (специальности)

Направленность (профиль)/ специализация: Программное обеспечение мобильных систем

Место прохождения практики: г. Новосибирск, ул. Бориса Богаткова, д. 51, ауд. 469

Объем практики: 360/10 часов/ЗЕ

Тип практики: Технологическая (проектно-технологическая) практика

Срок практики: с 16.09.2025 по 26.05.2026 (раз в неделю)

Содержание практики³:

Тема индивидуального задания практики Coming soon

Наименование видов деятельности	Дата (начало – окончание)
Архитектура Adalm Pluto SDR. GNU Radio. Построение радио-приёмника	16 сентября, 2025
Знакомство с библиотеками Soapy SDR, Libiio для работы с Adalm Pluto SDR. Инициализация SDR-устройства. Работа с буфером: получение цифровых IQ-отсчетов	23 сентября, 2025
Работа с библиотеками Soapy SDR, Libiio. Формирование и передача с SDR сигналов произвольной формы	30 сентября, 2025
Примеры формирования I/Q-сэмплов произвольной формы. Работа с буфером приема SDR	7 октября, 2025
Имитация аналоговой передачи звука и его прием с использованием SDR. Анализ влияния чувствительности приемника и усиления передатчика на качество принятых отсчетов сигнала (сэмплов)	14 октября, 2025 21 октября, 2025
Реализация приема и передачи BPSK-сигналов	28 октября, 2025
Дискретная свертка. Реализация приема и передачи BPSK-символов	11 ноября, 2025
Прием и фильтрация сигнала. Прямоугольный и приподнятый косинус	18 ноября, 2025
Программная реализация детектора временной ошибки (синхронизация приемника и передатчика) на SDR	25 ноября, 2025
Программная реализация детектора временной ошибки (синхронизация приемника и передатчика) на SDR. Написание функций петли (контура) синхронизации	2 декабря, 2025
	, 2025
	, 2025
	, 2025
	, 2025

³ В случае прохождения практики в профильной организации

В соответствии с рабочей программой практики
Руководитель практики от профильной
организации*
«__ » ____ 202_ г.

_____ / Андреев А. В. /
(подпись) (ФИО)

Руководитель практики от СибГУТИ
«__ » ____ 202_ г.

_____ / Брагин К. И. /
(подпись) (ФИО)

Отзыв о работе студента

(ФИО студента)

Уровень освоения компетенций

Компетенции	Уровень сформированности компетенций
ПК-1 Способен разрабатывать требования и проектировать программное обеспечение	

ПК-3 Способен осуществлять эксплуатацию и развитие транспортных сетей и сетей передачи данных, включая спутниковые системы

Уровень компетенций: высокий, средний, низкий, не сформирована

Руководитель практики от СибГУТИ:

Старший преподаватель

Кафедры ТС и ВС

должность руководителя практики подпись

Брагин К.И.

ФИО руководителя практики

« __ » июля 202__ г.

СОДЕРЖАНИЕ

1 ПРАКТИКА: АРХИТЕКТУРА SDR СИСТЕМЫ	3
2 ЗНАКОМСТВО С БИБЛИОТЕКАМИ SOAPY SDR, LIBIIO	11

1 ПРАКТИКА

АРХИТЕКТУРА SDR СИСТЕМЫ

УСТАНОВКА ПО, НАСТРОЙКА УСТРОЙСТВА

| [Ссылка на GitHub](#)

Цель практики:

Узнать, что такое SDR, изучить принципы его работы и внутреннюю архитектуру на базовом уровне. Познакомиться с инструментом GNU Radio и создать с его помощью программу для SDR, позволяющую принимать радио.

Краткие теоретические сведения

Что такое SDR?

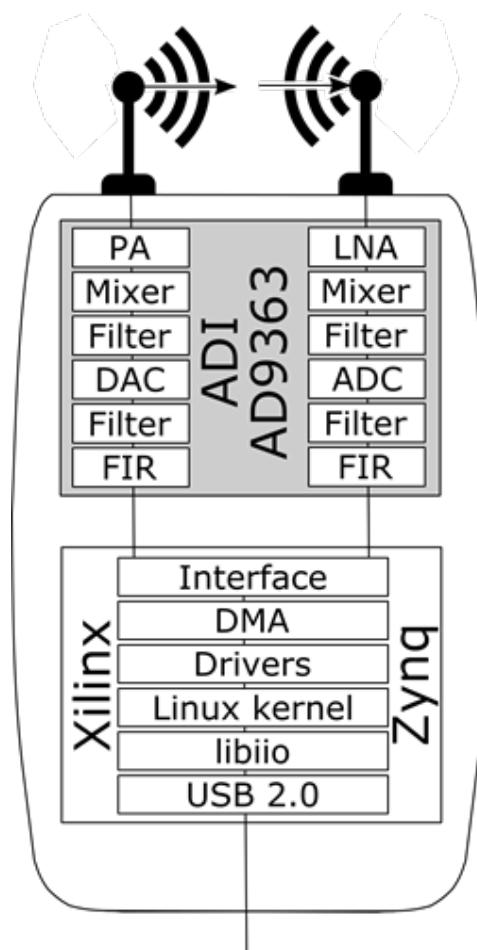


Рис. 1 — ADALM Pluto

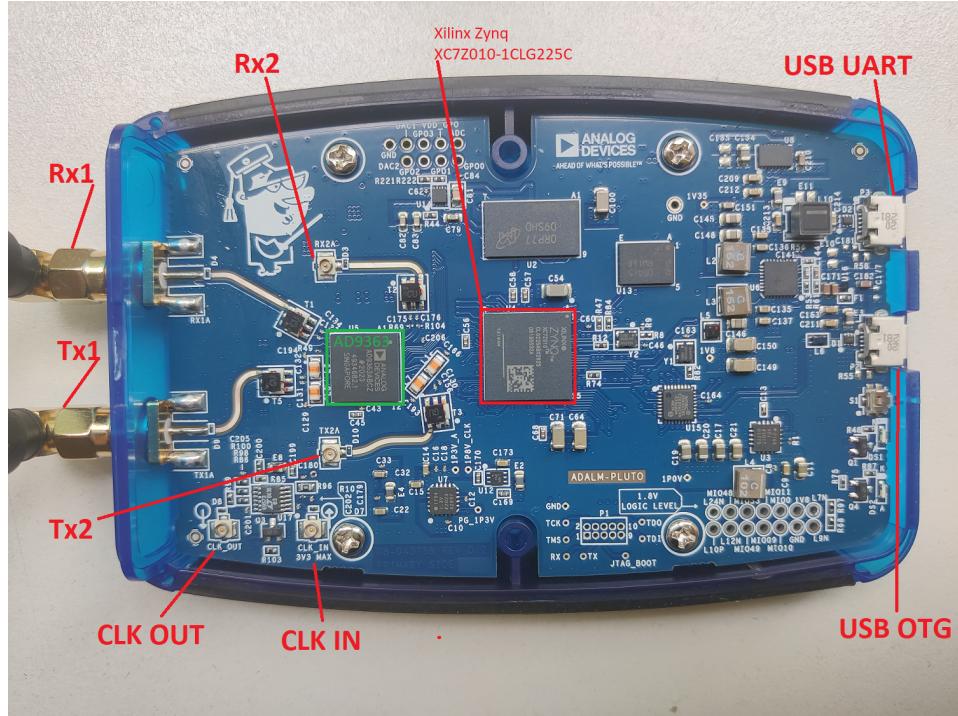


Рис. 2 — ADALM Pluto

Обучающая платформа PlutoSDR может взаимодействовать с:

1. Matlab, Simulink
2. GNU Radio
3. C, C++ при помощи дополнительных библиотек
4. C#
5. Среда языка Python

В начале данного курса мы наладим взаимодействие PlutoSDR с языком Python. Далее напишем программы под другие платформы, сравним разницу по времени обработки сигналов, простоте написания кода и редактирования.

Чип AD9363

Программируемый РЧ приемопередатчик, возможности которого позволяют использовать его для построения микро- (фемто-) сот мобильной связи 3G, 4G и 5G (в некоторых конфигурациях).

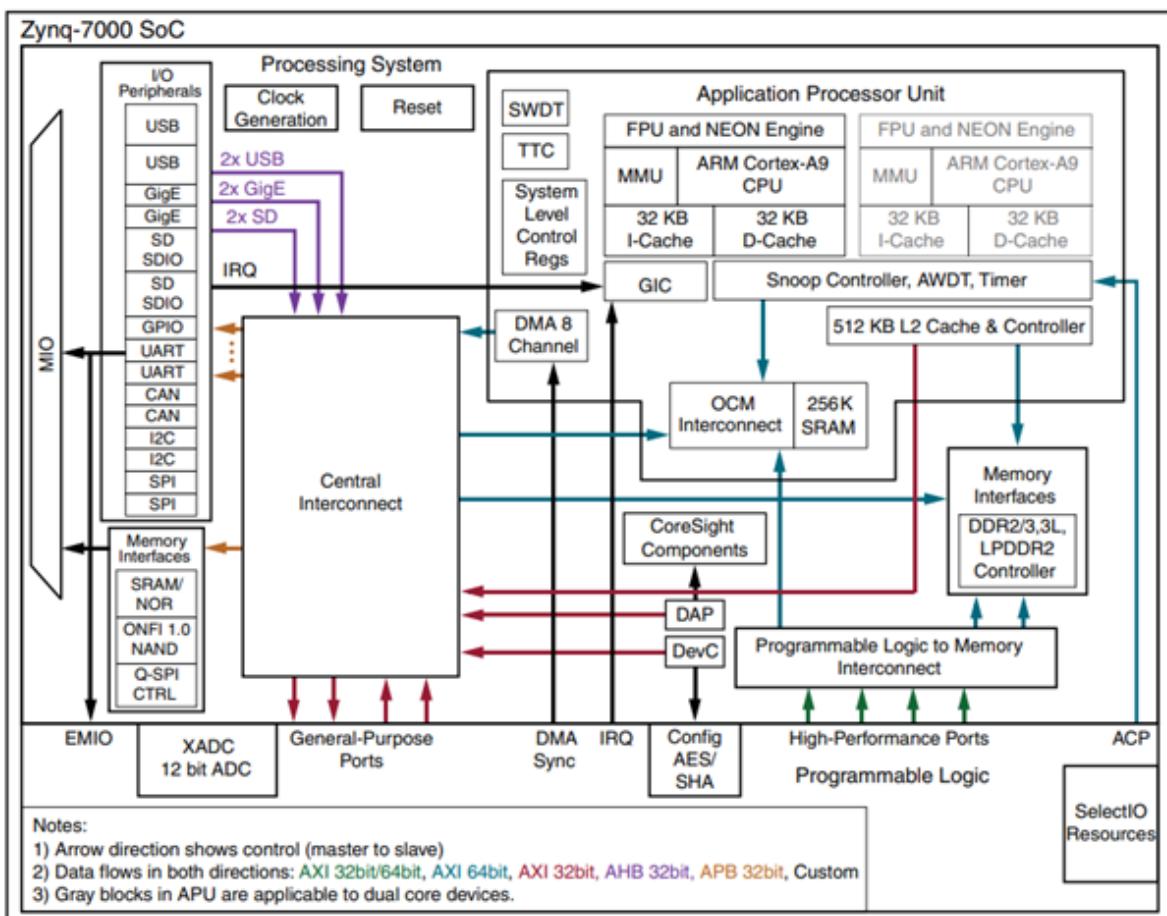


Рис. 3 — ПЛИС Xilinx Zynq

Основные характеристики

- **12-битный ЦАП/АЦП** (Цифро-аналоговый/Аналого-цифровой Преобразователь)
- Поддерживаемые несущие частоты от 90 [МГц] до **3.8** [ГГц]
- Поддерживает временной и частотный дуплекс (**TDD, FDD**)
- Ширина полосы частот: **20** [МГц]
- Шумы в приемнике: **3** [dB]
- EVM (Error Vector Magnitude): **-34** [dB]
- Tx noise: < **-157** [dBm/Hz]
- **2 Rx, 2 Tx**

Структурная схема Zynq

Каждый Zynq состоит из одного или двух ядер ARM Cortex-A9 (ARM v7), кэш L1 у каждого ядра свой, кэш L2 общий. Поддерживаемая оперативная память имеет стандарты DDR3, DDR3L, DDR2, LPDDR-2. Максимальный объем оперативной памяти равен 1 Гбайт (2 микросхемы по 4 Гбит). Максимальная тактовая частота оперативной памяти 525 МГц. Операционные системы: Standalone

(bare-metal) и Petalinux. Процессорный модуль общается с внешним миром и программируемой логикой с помощью портов, объединенных в группы:

- MIO (multiplexed I/O)
- EMIO (extended multiplexed I/O)
- GP (General-Purpose Ports)
- HP (High-Performance Ports)
- ACP (Accelerator coherency port)

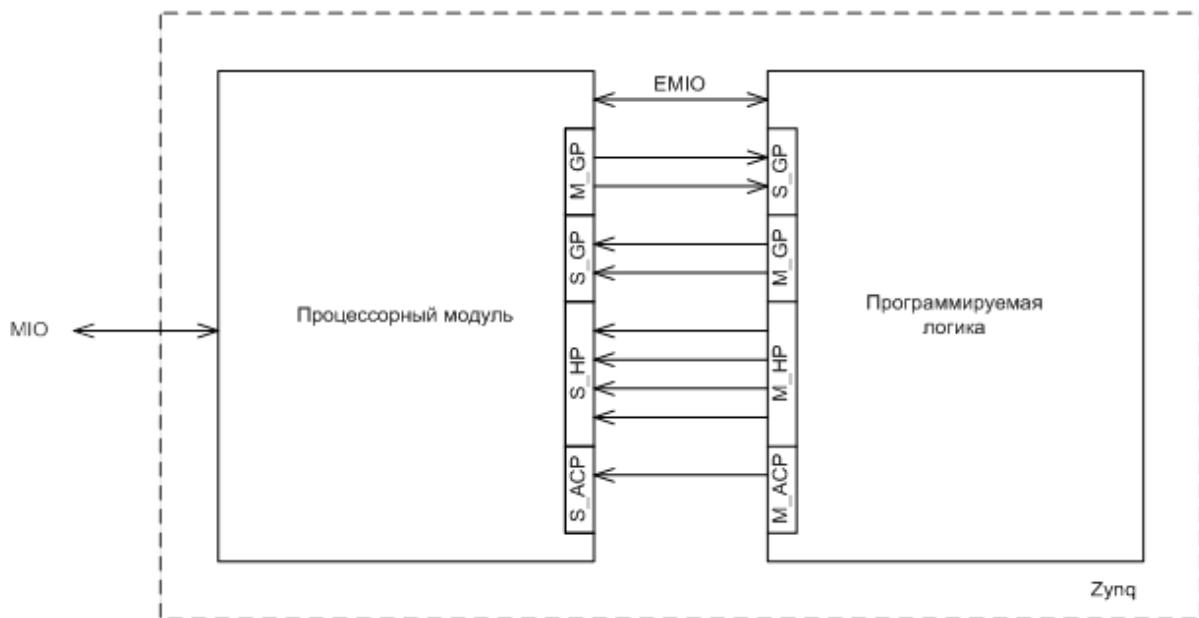


Рис. 4 — Структурная схема Zynq

Схема интерфейсов Zynq

*Буквы *S* и *M* у порта обозначают соответственно *Slave* и *Master*.

Так как, в одном корпусе Zynq реализованы и процессорный модуль и программируемая логика, есть выводы, которые относятся к процессорному модулю и выводы, которые относятся к программируемой логике.

Порты

Порты МИО представляют собой многофункциональные порты ввода-вывода, непосредственно подключенные к выводам процессорной системы (Processing System, PS). Ключевые характеристики:

- **Количество:** 54 порта (в большинстве конфигураций)
- **Назначение:** подключение периферийных устройств PS к внешним выводам кристалла
- **Особенности:** мультиплексирование функций на одних и тех же физических выводах

MIO

Порты MIO подключены к выводам процессора. С помощью MIO могут быть подключены следующие периферийные устройства процессорного модуля:

- USB-контроллер – 2 шт
- Gigabit Ethernet контроллер – 2 шт
- SD/SDIO контроллер – 2 шт
- UART – 2 шт
- CAN – 2 шт
- I2C – 2 шт
- SPI – 2 шт
- GPIO. Все выводы можно использовать как выводы общего назначения

Так же, к MIO могут быть подключены следующие устройства памяти процессорного модуля:

- QSPI контроллер
- ONFI контроллер
- SRAM/NOR контроллер

Количество MIO портов равно 54 (за исключением некоторых микросхем в корпусе CLG225, там еще меньше). Поэтому все сразу включить не удастся. Для решения этой проблемы существует группа портов EMIO.

Выполнение

Практическая работы выполнялась в программе GNU Radio. GNU Radio - это открытая платформа для разработки программных решений в области SDR. Её ключевое преимущество — визуальное проектирование радиоэлектронных систем с помощью готовых блоков, что избавляет от необходимости программировать. Собранные проекты работают непосредственно с SDR-оборудованием, таким как Adalm-Pluto или LimeSDR.

Библиотека GNU Radio содержит обширный набор компонентов для цифровой обработки сигналов (ЦОС). Сами модули разработаны на C++ для обеспечения высокой производительности, а их соединение и управление проектом осуществляется с помощью Python. Разработку можно вести как программно, используя API, так и визуально — в графической среде GNU Radio Companion (GRC).

Создание схемы в GNU Radio

Блок Options (Параметры проекта)

Данный блок определяет основные настройки всего проекта. Наиболее важные параметры:

- **Output Language (Язык выходного кода)**: определяет язык генерации исполняемой программы (Python или C++)
- **Generate Options (Опции генерации)**: задаёт тип используемого интерфейса

Блок Variable (Переменная)

Служит для объявления переменных. Переменная имеет имя (ID) и значение. В проекте часто определяется:

Variable ID: samp_rate
 Value: 2.4M

Блок QT GUI Range (Ползунок)

Создает регулируемый ползунок для динамического изменения параметров во время работы программы.

- **Default Value**: значение при запуске
- **Start/Stop**: диапазон значений
- **Step**: шаг изменения

Блок PlutoSDR Source (Источник)

Обеспечивает подключение к SDR-модулю ADALM-Pluto и управление его параметрами.

Основные параметры:

- **IIO context URI**: IP-адрес устройства
- **Sample Rate**: частота дискретизации АЦП
- **LO Frequency**: частота гетеродина
- **Buffer size**: размер буфера

Блок Low Pass Filter (ФНЧ)

Фильтр нижних частот для ограничения полосы пропускания сигнала.

Параметры:

- **Decimation**: коэффициент уменьшения частоты дискретизации
- **Cutoff Freq**: частота среза
- **Sample Rate**: исходная частота дискретизации

Блок QT GUI Frequency Sink (Спектр)

Обеспечивает визуализацию спектра сигнала в реальном времени.

Блок QT GUI Time Sink (Осциллограф)

Отображает сигнал во временной области.

Блок WBFM Receive (FM-демодулятор)

Выполняет демодуляцию широкополосного FM-сигнала.

Блок Audio Sink (Выход на звуковую карту)

Направляет аудиопоток на звуковую карту компьютера.

Сборка системы

Все блоки соединяются в графическом редакторе GRC, формируя законченную схему FM-приёмника. Программа позволяет в реальном времени:

- Принимать радиопередачи
- Наблюдать спектр и форму сигнала
- Регулировать частоту настройки

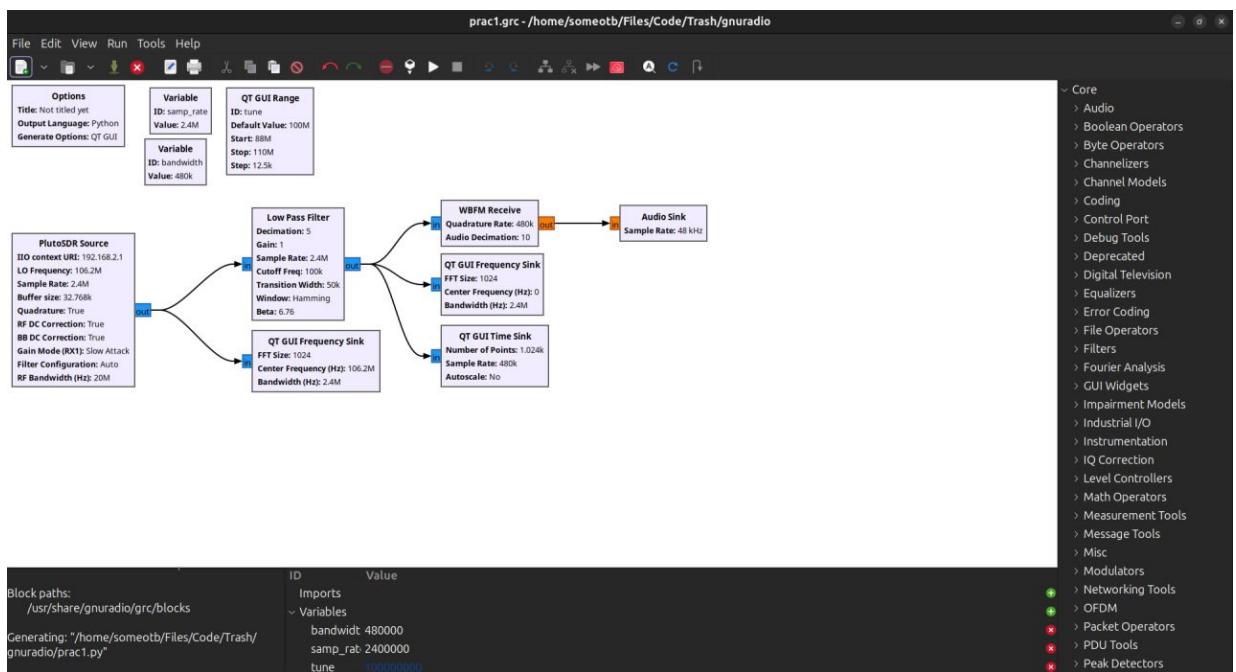


Рис. 5 — Собранная модель

Код собирается автоматически исходя из модели собранной из блоков в интерфейсе GNU Radio. И с помощью данной модели получилось поймать сигнал одной радио станции, которую можно было слушать, на других частотах звук был сильно искажен. На радио станции пока слушал, понравилась одна песня: **Девочка в платьице белом - Мюзикола.**

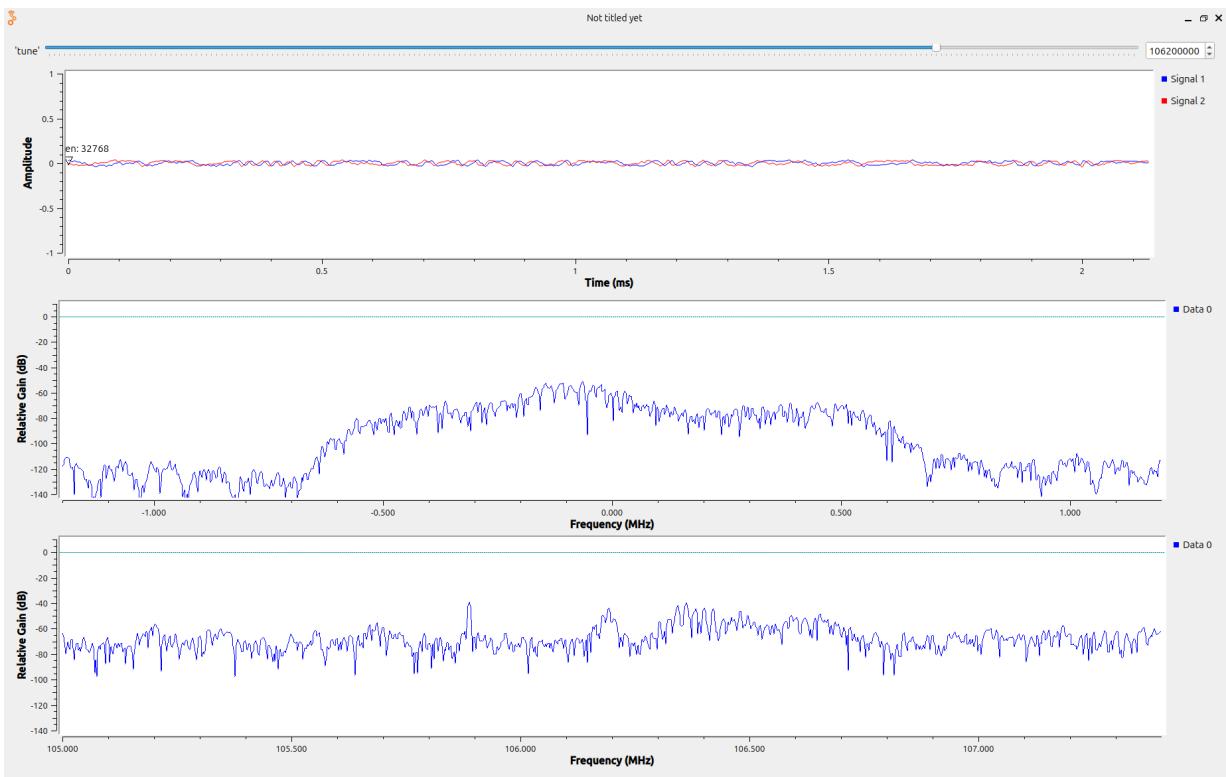


Рис. 6 — Результаты

Вывод

В результате работы были изучены принципы SDR, исследована архитектура ADALM Pluto и освоен инструмент GNU Radio. Разработана программа для приёма и воспроизведения передач в FM-диапазоне.

2 ЗНАКОМСТВО С БИБЛИОТЕКАМИ SOAPY SDR, LIBPPO 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.

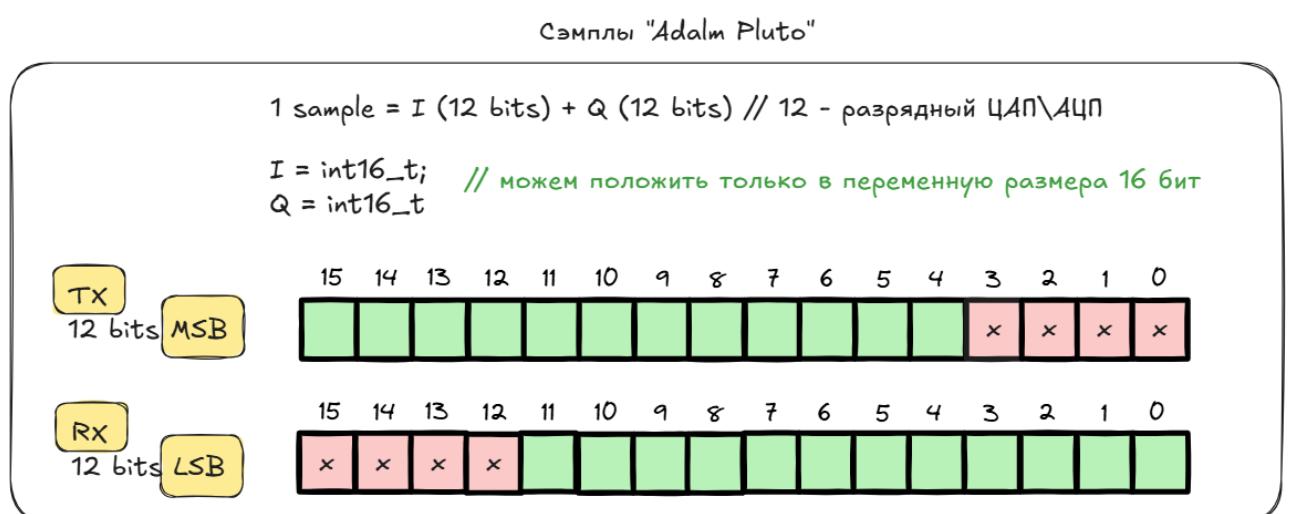


Рис. 7 — Сэмплы Adalm Pluto

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

Листинг 2.1 — 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

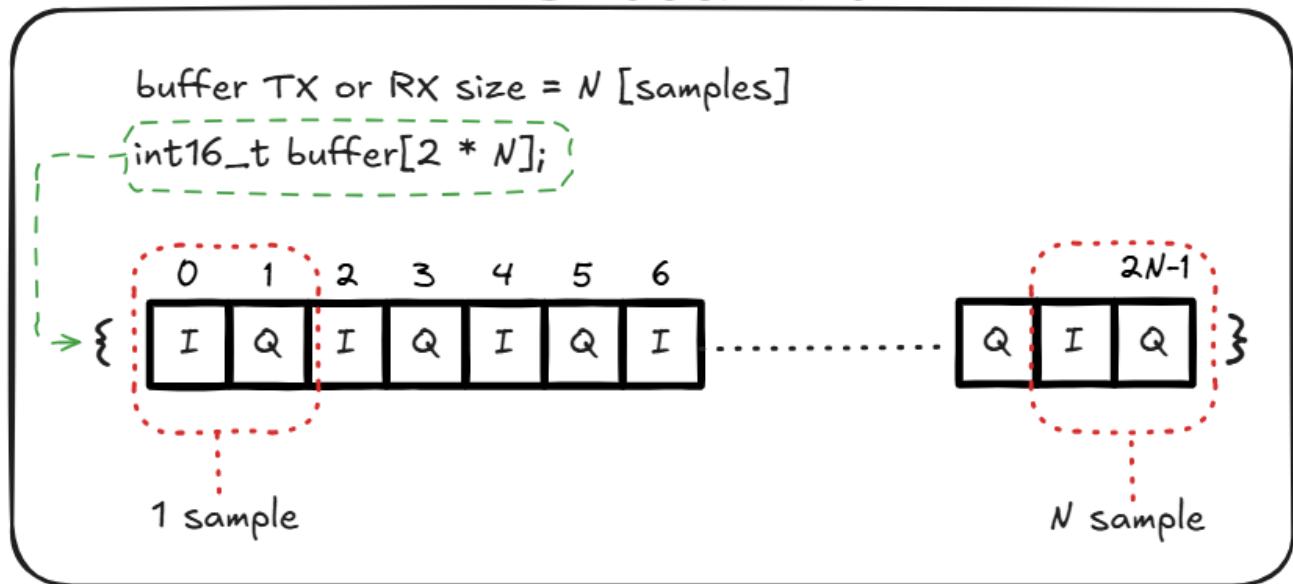


Рис. 8 — Структура буфера

Как происходит чтение сэмплов с SDR

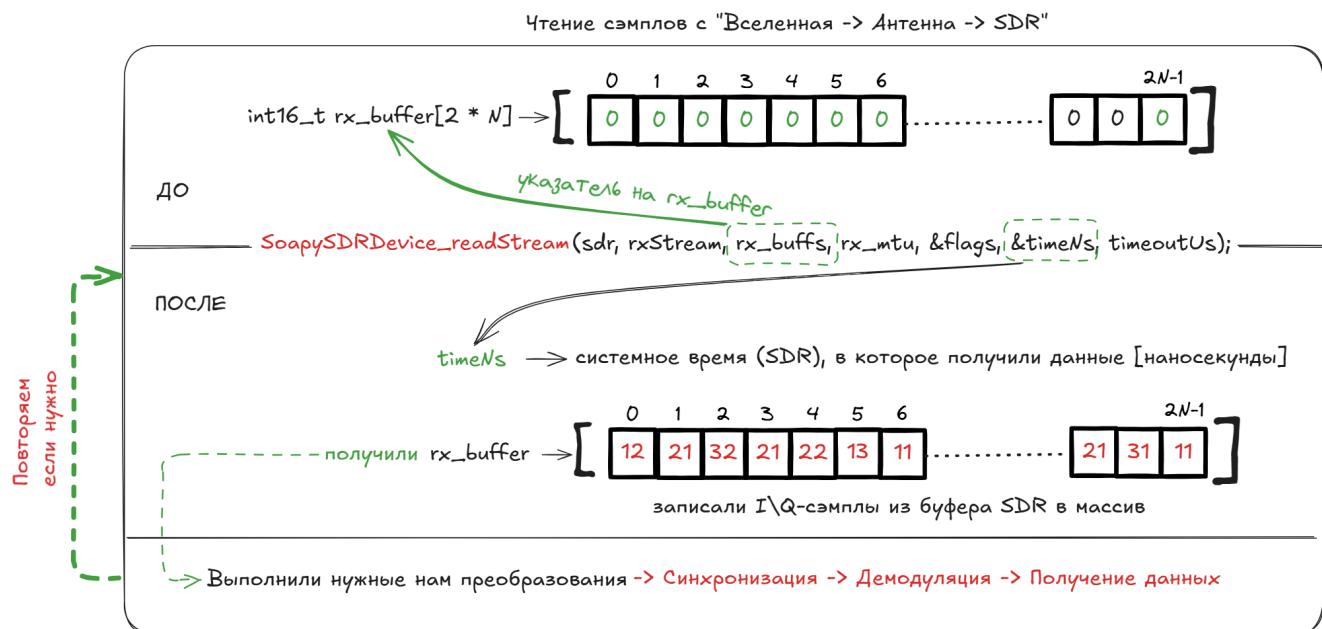


Рис. 9 — Чтение сэмплов

Листинг 2.2 — 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",
    buffers_read, sr, flags, timeNs, timeNs - last_time);
}

```

Как происходит подготовка TX буфера и передача сэмплов на SDR

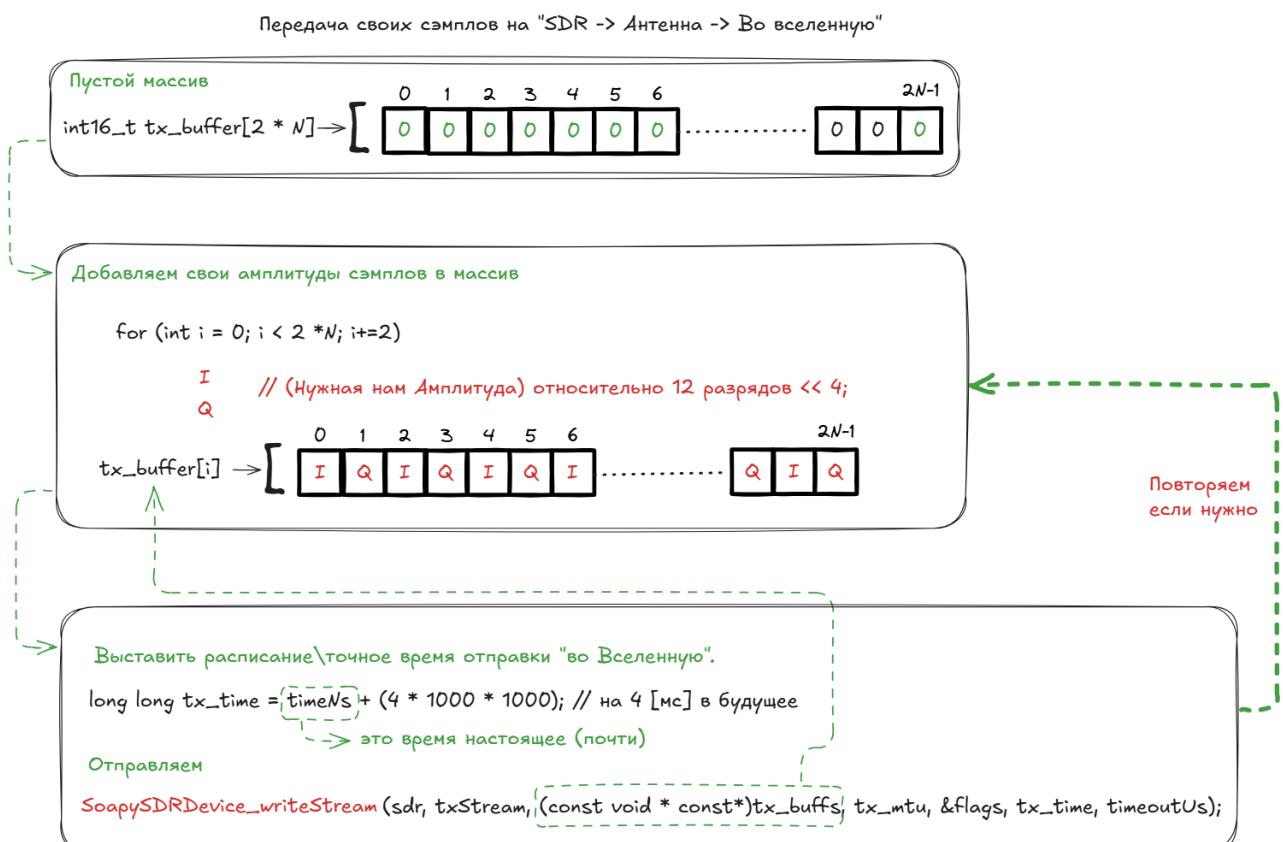


Рис. 10 — Передача сэмплов на SDR

Листинг 2.3 — 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
//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
17 last_time = timeNs;
18
19 long long tx_time = timeNs + (4 * 1000 * 1000); // на 4 мс\[] вбудущее
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:

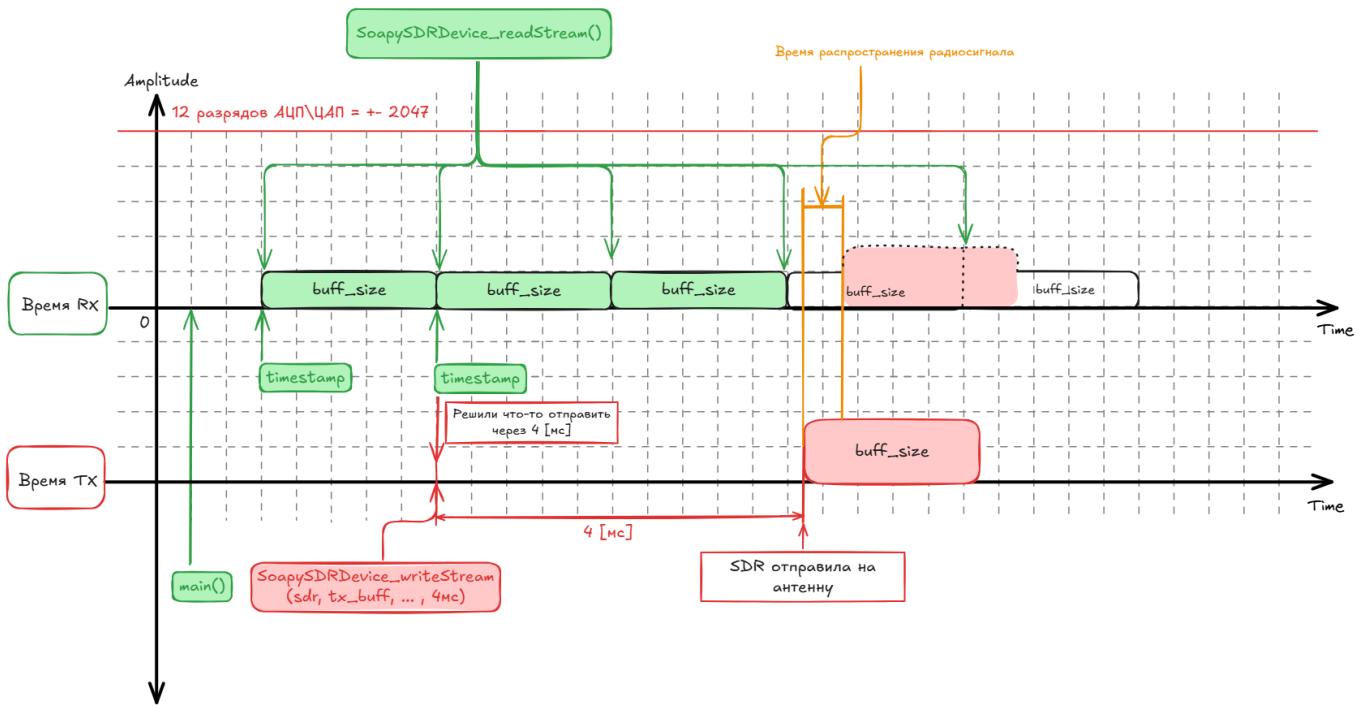


Рис. 11 — Одновременный прием и передача

Код, отражающий работу схемы:

Листинг 2.4 — 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    }
  
```

```

24 void *tx_buffs[] = {tx_buff};
25 if( (buffers_read == 2) ){
26     printf("buffers_read: %d\n", buffers_read);
27     flags = SOAPY_SDR_HAS_TIME;
28     int st = SoapySDRDevice_writeStream(sdr, txStream, (const void * const*)
29                                         tx_buffs, tx_mtu, &flags, tx_time, timeoutUs);
30     if ((size_t)st != tx_mtu)
31     {
32         printf("TX Failed: %i\n", st);
33     }
34 }
35 }
```

Выполнение

Сперва необходимо установить все необходимые библиотеки

Листинг 2.5 — 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 2.6 — 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 2.7 — 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 2.8 — 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 2.9 — 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 2.10 — 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 2.11 — 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.

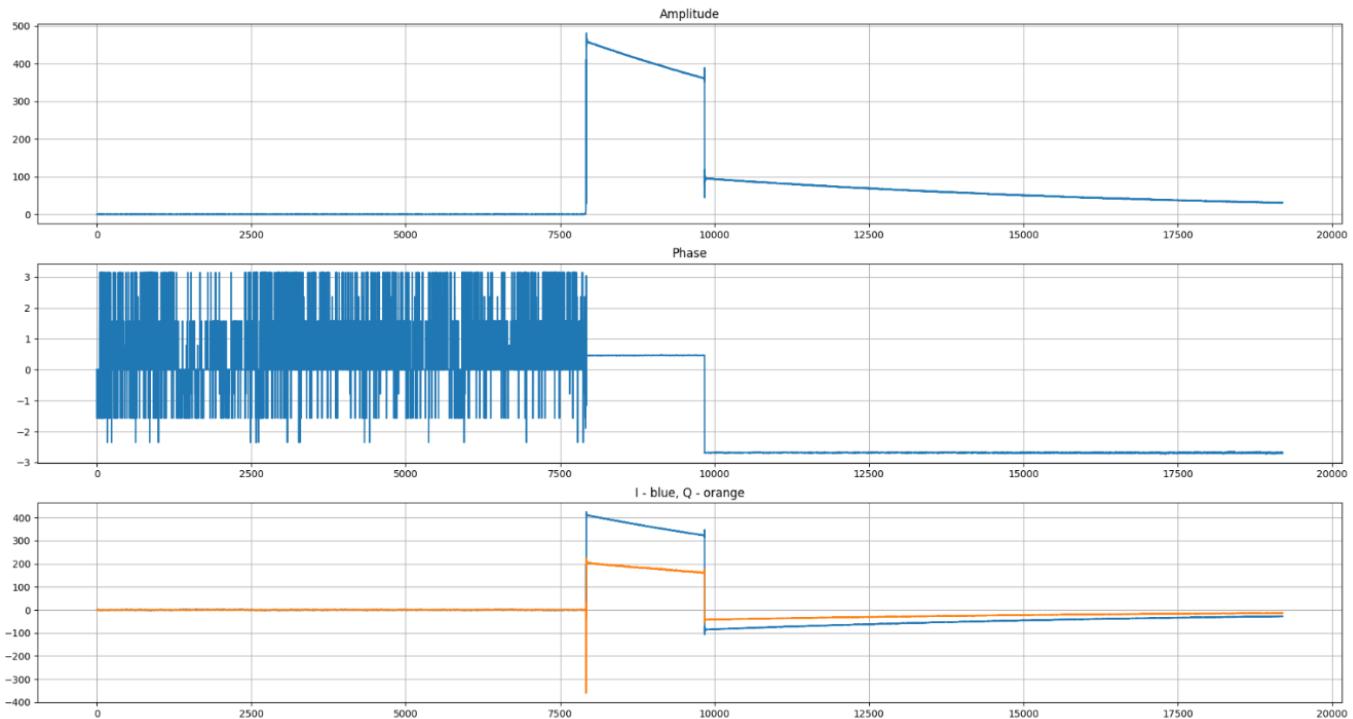


Рис. 12 — Результаты, которые мы получили с SDR

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.