

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:
АРХИТЕКТУРА ADALM PLUTO SDR. GNU RADIO. ПОСТРОЕНИЕ
РАДИО-ПРИЁМНИКА

Студент:
Группа ИА-331

Я.А Гмыря

Предподаватели:
Лектор
Семинарист
Семинарист

Калачиков А.А
Ахнашев А.В
Попович И.А

Новосибирск 2025 г.

СОДЕРЖАНИЕ

1	ЦЕЛЬ И ЗАДАЧИ	3
2	ЛЕКЦИЯ	4
3	ПРАКТИЧЕСКАЯ ЧАСТЬ	10
4	ВЫВОД	22

ЦЕЛЬ И ЗАДАЧИ

Цель: 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.

Задачи:

1. Прослушать и законспектировать лекцию, познакомиться с основами SDR-систем.
2. На основе полученных знаний создать в GNU Radio программу для SDR, позволяющую принимать радио.

ЛЕКЦИЯ

Что такое SDR?

Software-Defined Radio (SDR) - радиосистема, в которой часть аппаратных компонентов (фильтры, модуляторы и т.п.) реализованы на программном уровне.

Базовая архитектура системы радиосвязи

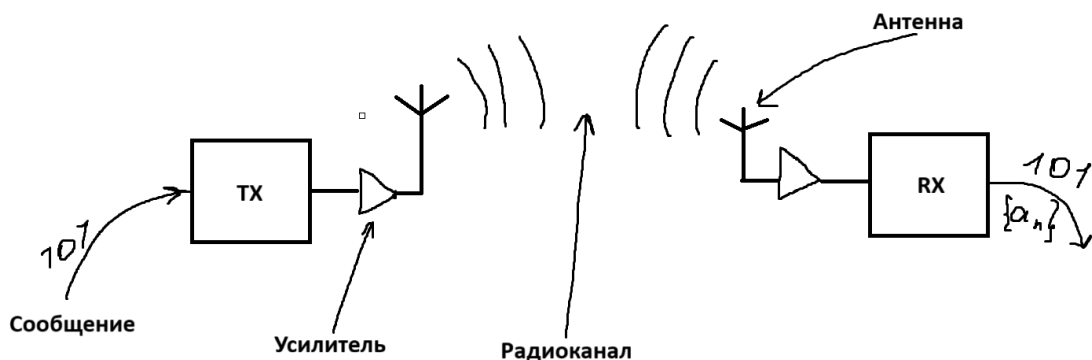


Рисунок 1 — Архитектура системы радиосвязи в целом

Описание компонентов архитектуры

Базовая архитектура состоит из **передатчика (TX)** и **приемника (RX)**. Между ними находится **радиоканал - среда**, в которой распространяется сигнал. У **TX** и **RX** есть **антенна - устройство**, которое излучает или принимает электромагнитные волны, и преобразует их в электрический ток и обратно, в самом простом случае это просто кусок проволоки. Также и у **TX** и у **RX** есть **усилитель**, который усиливает отправляемый/принимаемый сигнал.

Описание процесса обмена данными

На стороне **TX** формируется **сообщение**, которое необходимо передать. Это сообщение поступает в передатчик в виде набора **нулей и единиц**.

ТХ преобразует нули и единицы определенным образом в электрические колебания, которые через **антенну** излучаются в виде электромагнитных колебаний в радиоканал.

В этом же радиоканале находится **приемник**, **антенна** которого принимает эти электромагнитные колебания и преобразует в электрический ток. После этого электрические колебания определенным образом преобразуются в набор нулей и единиц (сообщение, которое отправлял **ТХ**). Стоит отметить, что **прием сообщения** намного сложнее, чем отправка. Это связано с изменениями, которым подвергается сигнал во время прохождения через **радиоканал**. Сигнал изменяется случайным образом, поэтому точно сказать, как изменится сигнал, мы не можем, мы можем это только предположить с какой-то точностью. Эта проблема решается путем добавления в исходный сигнал **избыточности**, которая позволяет с более высокой точностью принять сигнал на стороне приемника. Такой избыточностью может быть **контрольная сумма - число**, которое вычисляется по определенному алгоритму, который учитывает позицию бита и его значение, т.е. если хоть в какой-нибудь позиции изменится значение бита, то **контрольная сумма** будет уже другой, что сигнализирует об искажении сигнала.

Внутренняя архитектура ТХ

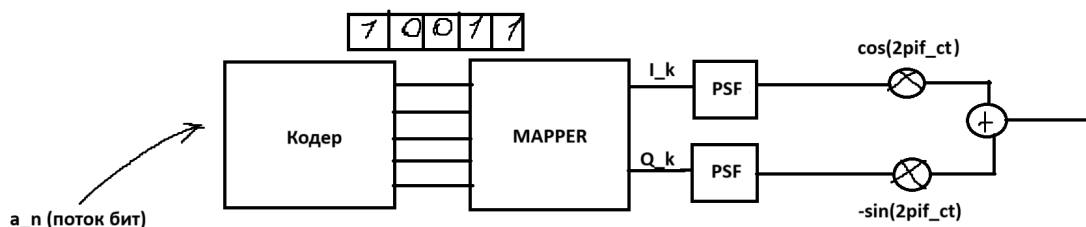


Рисунок 2 — ТХ архитектура

Coder

В нашем случае кодер будет выполнять единственную задачу - формировать из потока бит блоки (допустим по 8 бит) и направлять их в Mapper.

Mapper

Устройство, выполняющее **отображение исходных данных на множество символов или сигналов** в соответствии с выбранной схемой модуляции.

Символ - элемент сигнального множества.

Сигнальное множество - набор состояний радиосигнала.

Иными словами: mapper берет блок битов (у нас это 8 бит) и сопоставляет его сигналу с определенными характеристиками, для этого в mapper хранится таблица с комбинациями битов и соответствующие им символы. Если в блоке 8 бит, то всего должно быть 256 символов (на каждую возможную комбинацию).

b0	b1	state
0	0	state1
0	1	state2
1	0	state3
1	1	state4

Рисунок 3 — Пример таблицы в маппере

Также для визуализации данного процесса используется созвездие СИМВОЛОВ

Созвездие символов - это графическое представление множества **возможных символов модуляции** в комплексной плоскости. Каждый символ соответствует определённой комбинации параметров сигнала и изображается в виде точки на диаграмме.

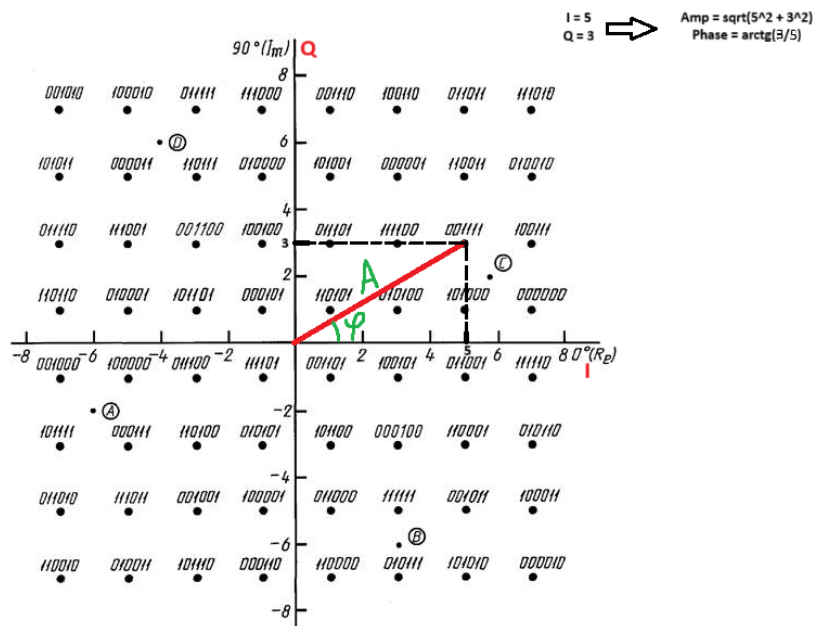


Рисунок 4 — Пример созвездия символов

Каждой точке соответствуют координаты (I, Q) , где I - действительная составляющая, Q - мнимая. Зная координаты, можем вычислить длину радиус-вектора до этой точки, это будет амплитудой этого сигнала, угол между действительной осью и радиус-вектором - фаза сигнала.

От таргет идет 2 выхода, один для I составляющей, другой для Q составляющей, которые поступают на вход формирующего фильтра.

Формирующий фильтр

Устройство, преобразующее последовательности символов в непрерывный сигнал с заданной формой. Этот фильтр должен превратить символы (I и Q) в длительные (передаваемые) символы (символы, растянутые по времени с длиной T_s)

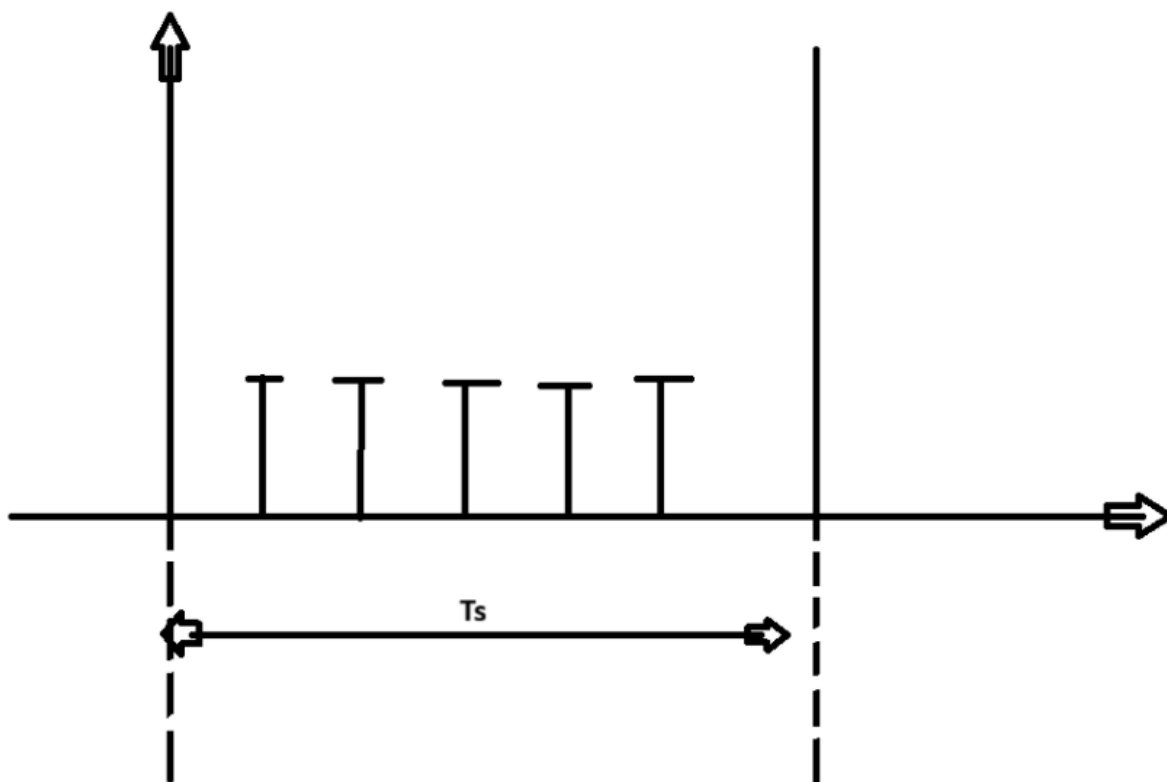


Рисунок 5 — Пример длительного символа

До этого момента все выполнялось программно. Всё, что будет дальше - работа самой SDR.

Далее происходит генерация непрерывного сигнала. Математически этот процесс можно записать следующим образом:

$$s(t) = I(t) \cos(2\pi f_c t) - Q(t) \sin(2\pi f_c t)$$

f_c здесь - несущая частота (высокочастотное колебание).

ПРАКТИЧЕСКАЯ ЧАСТЬ

Adalm Pluto SDR

Adalm Pluto SDR - модель SDR, разработанная компанией **Analog Devices** для обучения основам SDR.

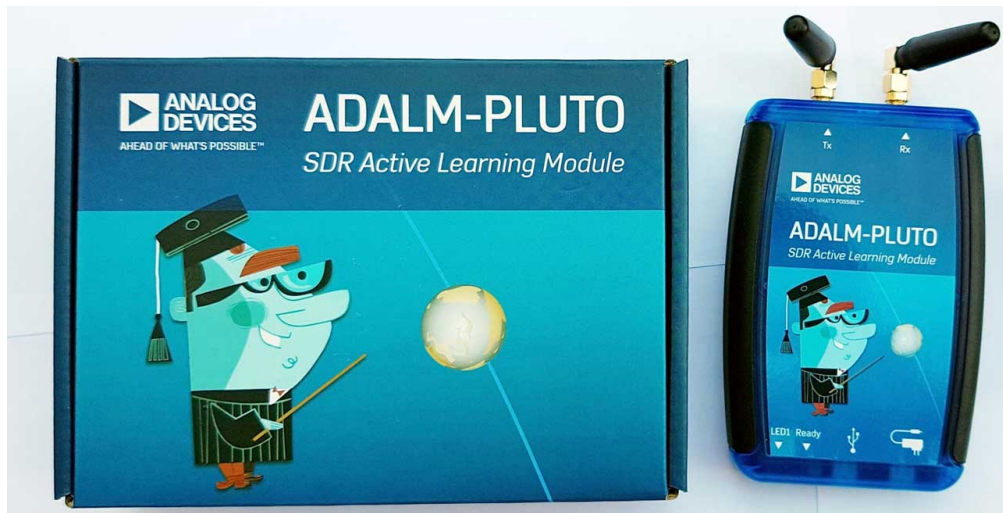


Рисунок 6 — Внешний вид Adalm Pluto

Архитектура Adalm Pluto SDR

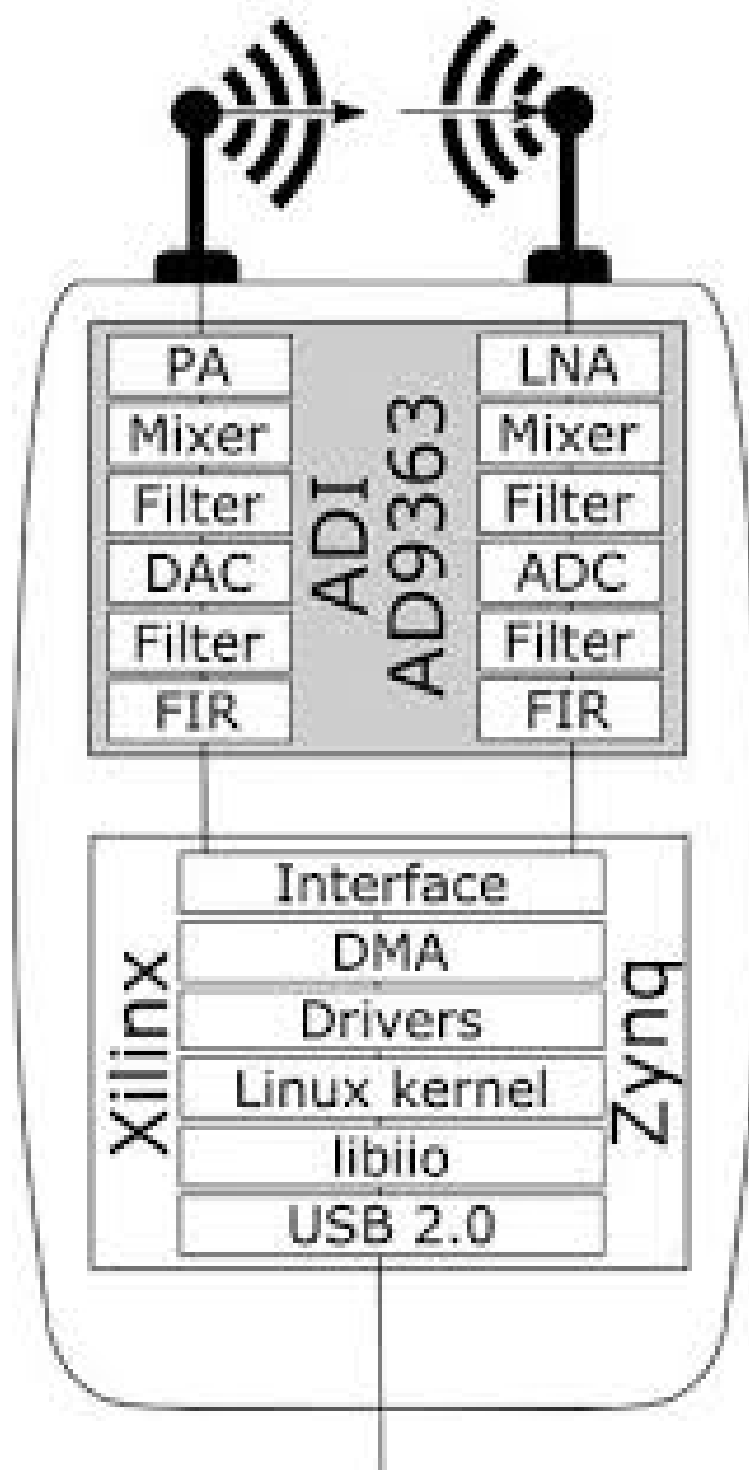


Рисунок 7 — Архитектура Adalm Pluto

Описание основных блоков Adalm Pluto

PA (Power Amplifier)

Функция: Усиление сигнала.

LNA (Low Noise Amplifier)

Функция: Усиление слабого приёмного сигнала с минимальным добавлением шума.

ADC / DAC

ADC (Analog-to-Digital Conversion): оцифровка аналогового сигнала.

DAC (Digital-to-Analog Conversion): преобразование цифровых семплов в аналоговый сигнал.

FIR (Finite Impulse Response)

Функция: детальная фильтрация, коррекция формы спектра.

Mixer

TX: перенос низкочастотного сигнала на несущую частоту.

RX: перенос высокочастотного сигнала на низкочастотный.

Filter

TX: фильтрация выходного сигнала перед передачей.

RX: фильтрация принимаемого сигнала.

libiio

Описание: библиотека, которая облегчает работу с устройствами ввода/вывода в **Linux**. Она даёт **API** для обмена данными и управления устройствами.

Linux Kernel

Описание: ядро **Linux**. Управляет железом.

Drivers

Описание: обеспечение корректной работы **Linux** с устройствами.

Xilinx Zynq

Описание: семейство микросхем от компании Xilinx. На одном кристалле объединены ARM-процессор, на котором работает **Linux**, и ПЛИС для более скоростных вычислений.

GNU Radio



Рисунок 8 — Логотип GNU Radio

GNU Radio - это инструмент с открытым исходным кодом для разработки программного обеспечения в сфере программно-определяемого радио.

Он позволяет при помощи «строительных блоков» создавать конфигурации радиоустройств, не написав ни одной строчки кода, и запускать программы непосредственно с использованием SDR-модулей, например:

Adalm-Pluto, LimeSDR и др.

В библиотеке имеется широкий спектр функций для цифровой обработки сигналов. Модули написаны на **C++**, а их взаимодействие реализовано на **Python**. Приложения можно строить как через **API GNU Radio**, так и посредством графического интерфейса **GNU Radio Companion (GRC)**.

Построение схемы в GNU Radio

Блок options

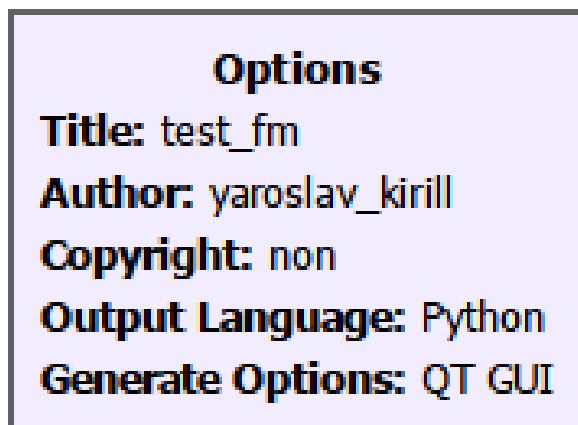


Рисунок 9 — Блок options

Этот блок задает настройки проекта. Самое важное здесь: **Output Language** и **Generate Options**.

Output Language — язык, на котором будет сгенерирован код программы.

Generate Options — используемый графический интерфейс.

Блок variable

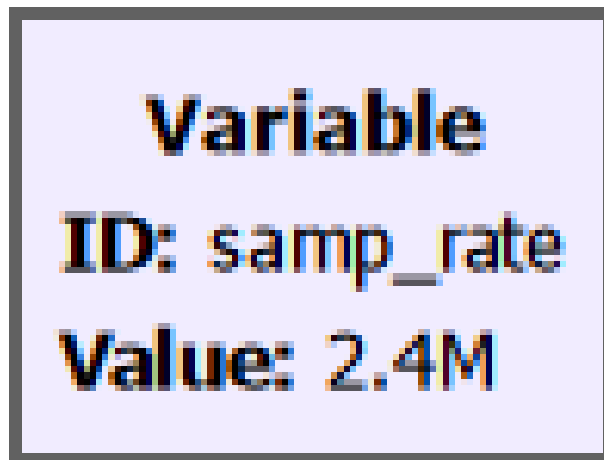


Рисунок 10 — Блок variable

В этом блоке можно задать переменную (почти как в языке программирования). Переменная имеет ID (имя) и значение. Здесь задается `samp_rate` (частота дискретизации), равная 2.4×10^6 Hz.

QT GUI Range

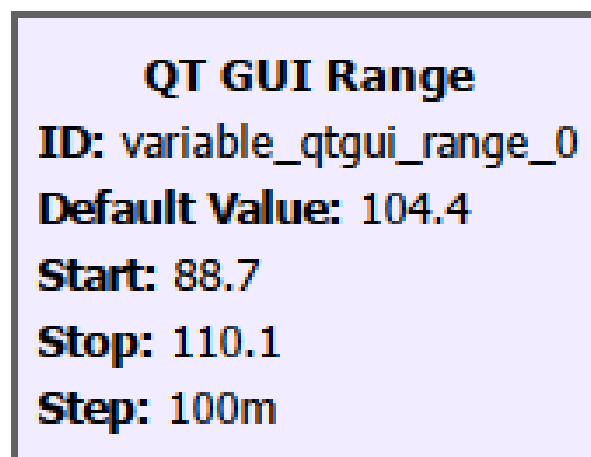


Рисунок 11 — Блок QT GUI Range

Этот блок задает ползунок из QT, позволяющий удобно менять значение переменной во время работы программы. Это позволяет не перезапускать программу, когда нам требуется поменять какое-либо значение. Здесь задается ползунок для настройки частоты приема FM волны.

Основные параметры блока:

- **Default Value** — значение, которое будет устанавливаться при запуске программы;
- **Start** — минимальное значение;
- **Stop** — максимальное значение;
- **Step** — шаг изменения при сдвиге ползунка.

PlutoSDR Source

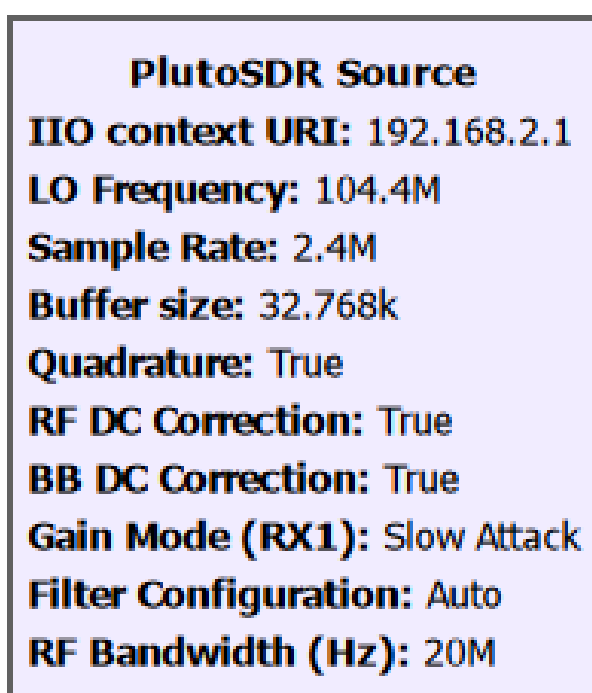


Рисунок 12 — Блок PlutoSDR Source

Этот блок отвечает за приём данных от устройства ADALM-Pluto (PlutoSDR). Он подключается к SDR, управляет настройками, получает поток отсчётов.

Параметры:

- **ИО context URI** — IP адрес Adalm Pluto, нужен, потому что PlutoSDR может подключаться по USB или сети (Ethernet/USB-Ethernet);

- **Sample Rate** — частота дискретизации АЦП внутри PlutoSDR. Определяет, с какой частотой будут делаться отсчеты при оцифровке;
- **Buffer Size** — встроенный буфер для временного хранения данных перед их передачей в компьютер;
- **Quadrature** — задаём представление сигнала в виде I/Q семплов;

Low Pass Filter

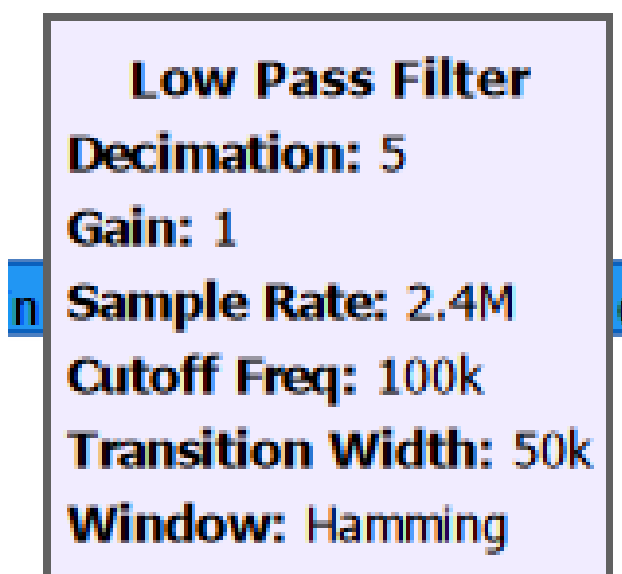


Рисунок 13 — Блок Low Pass Filter

Ограничивает полосу сигнала, выделяя только FM-станцию.

Параметры:

- **Decimation** — снижение частоты дискретизации в n раз;
- **Gain** — усиление амплитуды после фильтрации;
- **Sample Rate** — дефолтная частота дискретизации;
- **Cutoff Freq** — полоса 100 кГц (ширина FM сигнала).

QT GUI Frequency Sink

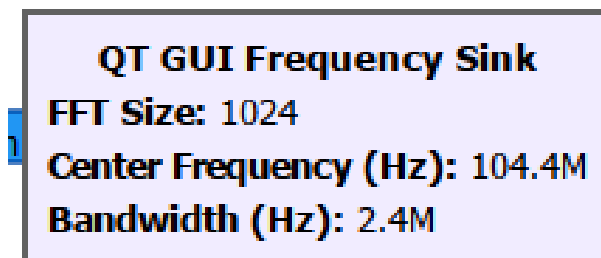


Рисунок 14 — Блок QT GUI Frequency Sink

Этот блок при помощи QT задает спектральное представление сигнала, которое меняется в реальном времени. Таких блоков 2: до фильтра (напрямую из блока source) и после фильтра. Первый показывает весь эфир, а второй — захваченный сигнал (именно FM частоту).

Параметры:

- **FFT Size** — кол-во точек для спектра;
- **Center Frequency** — центральная частота захвата;
- **Bandwidth** — полоса частот захвата.

QT GUI Time Sink

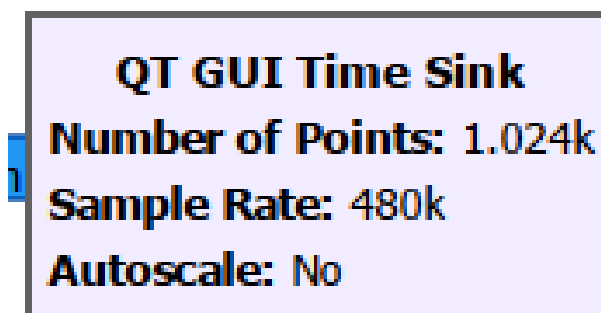


Рисунок 15 — Блок QT GUI Time Sink

Этот блок при помощи QT задает временное представление сигнала, которое меняется в реальном времени.

Параметры:

- **Number of Points** — кол-во точек, отображаемых в каждый момент времени;
- **Sample Rate** — частота дискретизации при отрисовке;
- **Autoscale** — нужно ли масштабировать сигнал по вертикали.

WBFM Receive

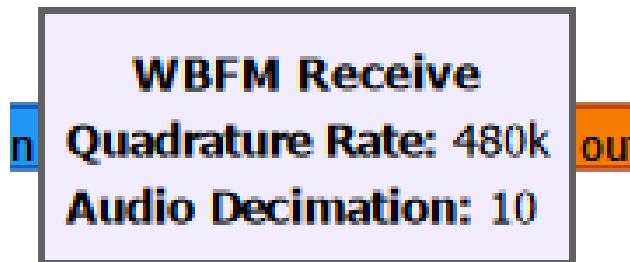


Рисунок 16 — Блок WBFM Receive

Блок демодуляции FM-сигнала.

Параметры:

- **Quadrature Rate** — входная частота дискретизации (после фильтра и децимации);
- **Audio Decimation** — уменьшение дискретизации для звука (в моем случае до 48k, чего вполне достаточно для звука).

На выходе — звуковой сигнал.

Audio Sink

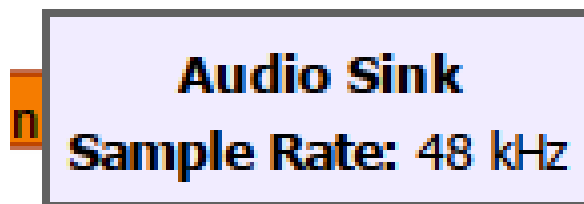


Рисунок 17 — Блок Audio Sink

От **WBFM Receive** звук идет на блок **Audio Sink** — блок, который выводит звуковой поток на аудиокарту хоста.

Параметры:

– **Sample Rate** — стандартная частота звука.

Соединить блоки нужно следующим образом, тогда у нас получится работающая радиосистема, которая будет принимать FM радио.

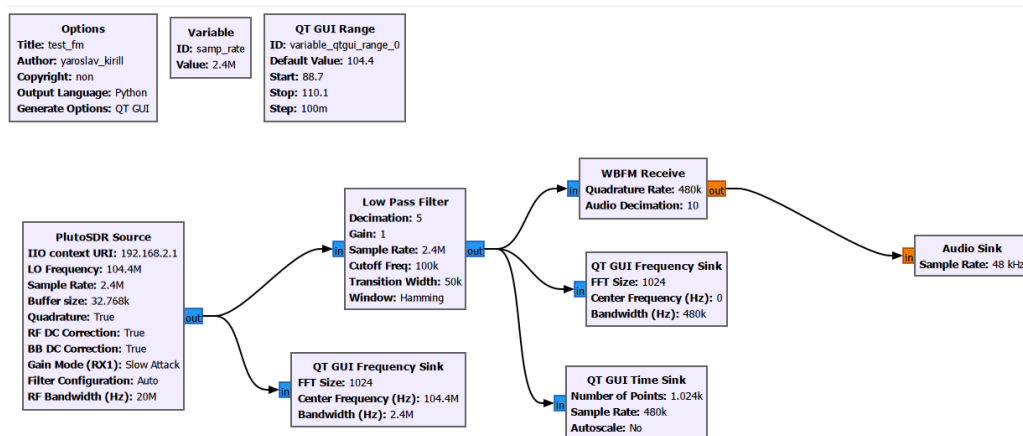


Рисунок 18 — Пример простой радиосистемы в GNURadio

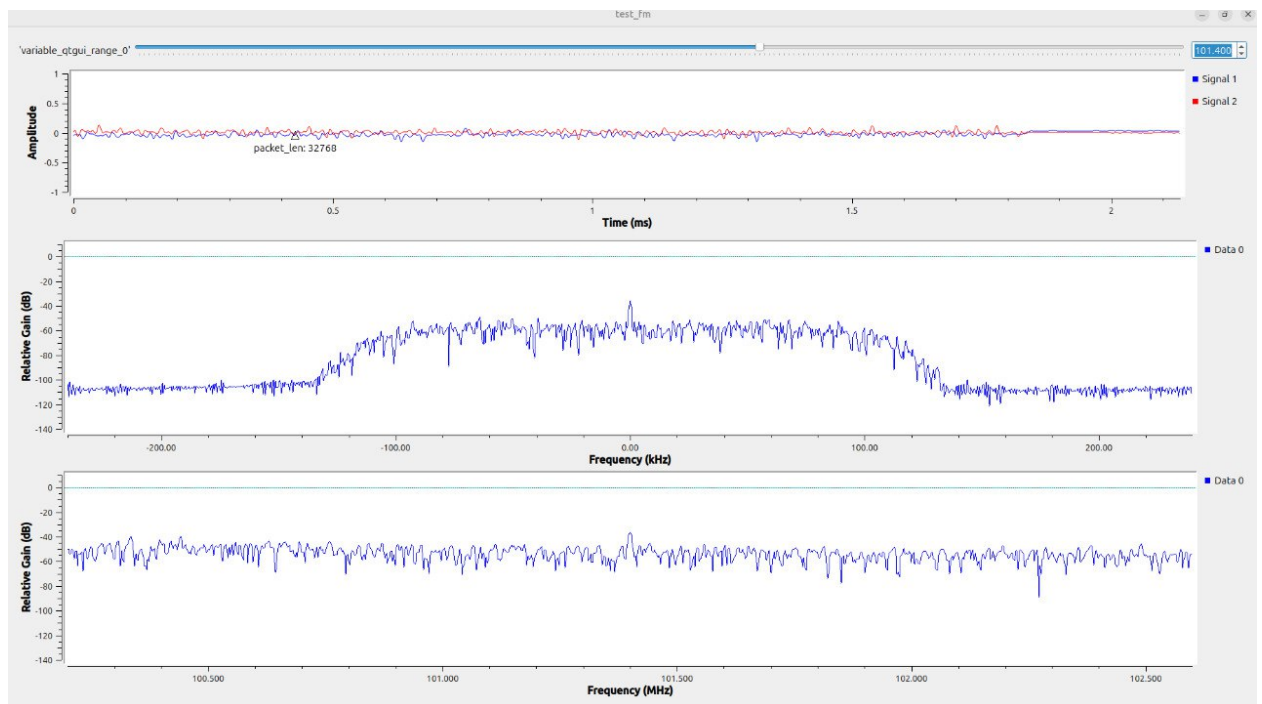


Рисунок 19 — Пример работы программы

ВЫВОД

В ходе проделанной работы я узнал, что такое SDR, изучил принципы его работы и внутреннюю архитектуру на базовом уровне. Познакомился с инструментом GNU Radio и создал с его помощью программу для SDR, позволяющую принимать FM радио.