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

Новосибирск 2025 г.

**СОДЕРЖАНИЕ**

# 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 традиционно аппаратные компоненты (фильтры, модуляторы, демодуляторы, детекторы, кодеры и т.п.) реализованы на программном уровне, с использованием универсального оборудования и цифровой обработки сигналов.

### Базовая архитектура системы радиосвязи

Картинка1

### 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**. **TX** 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 преобразуется в набор нулей и единиц (сообщение, которое отправлял **TX**). 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 TX**

картинка2

**Coder**

Устройство или программный модуль, предназначенный для **преобразования исходных данных в кодированное представление**.

Основные функции кодера:

- **Преобразование данных** - перевод информации (например, текста, аудио, видео, сигналов) в определённый формат или код.
- **Сжатие** - уменьшение объёма данных за счёт использования специальных алгоритмов кодирования.
- **Обеспечение помехоустойчивости** - добавление избыточности в данные для защиты от ошибок при передаче по каналу связи.
- **Совместимость** - приведение данных к стандартному виду, который может быть правильно обработан устройством-приёмником.

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

**Mapper**

Устройство или программный модуль, выполняющий **отображение исходных данных на множество символов или сигналов** в соответствии с выбранной схемой модуляции или кодирования.

Основные функции маппера:

- **Отображение битов на символы** - преобразование двоичной последовательности в набор сигналов (например, QPSK, QAM).
- **Подготовка к модуляции** - формирование последовательности комплексных чисел, которые могут быть использованы модулятором.
- **Оптимизация передачи** - использование схем отображения, минимизирующих вероятность ошибки при передаче (например, Gray-код).
- **Гибкость** - возможность выбора различных карт маппинга в зависимости от условий канала и требований к системе.

В нашем самом простом случае mapper будет выполнять единсвтенную задачу - отображать исходные биты на множество символов.

Символ - элемент сиганльного множества.

Сигнальное мн-во - 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 соответсвующие им символы. Если в блоке 8 бит, то всего должно быть 256 символов (на каждую возможную комбинацию).

Пример таблицы

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 (амплитуды и фазы), и изображается в виде точки на диаграмме.

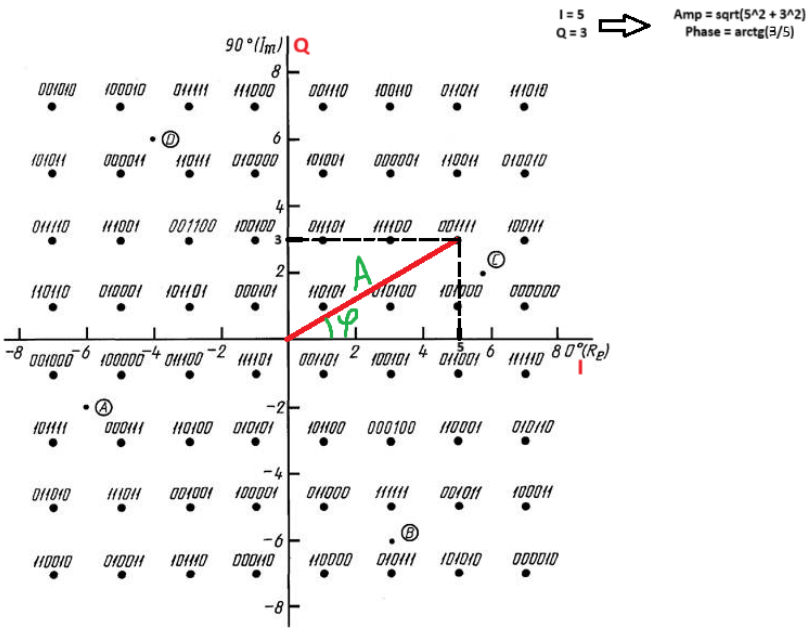


Рисунок 1 — Пример созвездия символов

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

## Немного про I/Q семплы

Запишем сигнал и распишем с помощью формулы аргументов косинусов.

$$S_c(t) = A(t)cos(2\pi f_c t + \phi(t)) = A(t)cos\phi(t) * cos(2\pi f_c t) - A(t)sin\phi(t) * sin(2\pi f_c t)$$

Здесь $A(t)cos\phi(t)$ - синфазная часть сигнала (I), а $A(t)sin\phi(t)$ - квадратурная часть (Q).

Таким образом, зная I и Q можно создать сигнал, и mapper как раз нам эти I/Q и дает.

Все это называется QAM-модуляцией (Quadrature Amplitude Modulation) - технология передачи цифрового информационного потока в виде аналогового сигнала. Это достигается путем разделения несущей волны на две несущие одинаковой частоты сдвинутые относительно друг-друга на 90 градусов.

**Формирующий фильтр**

Устройство или программный модуль в системе передачи данных, предназначенный для преобразования последовательности символов в непрерывный сигнал с заданной временной формой, оптимальной для передачи по каналу связи.

**Математическое описание:**

$$s(t) = \sum_{k=-\infty}^{\infty} a_k \, h(t - kT)$$

где:

- $s(t)$ — сигнал на выходе формирующего фильтра,
- $a_k$ — последовательность передаваемых символов,
- $h(t)$ — импульсная характеристика фильтра,
- $T$ — период передачи символа.

**Функции формирующего фильтра:**

1. Формирование импульсной формы сигнала для минимизации межсимвольной интерференции (ISI).
2. Ограничение спектра передаваемого сигнала для уменьшения влияния шумов и помех.
3. Обеспечение соответствия сигнала требованиям канала связи (например, полосе пропускания).

4. Подготовка сигнала к последующей модуляции и передаче.

На этом этапе необходимо превратить символы (из I и Q) в длительные (передаваемые) символы (символы, растянутые по по времени с длиной $T_s$)

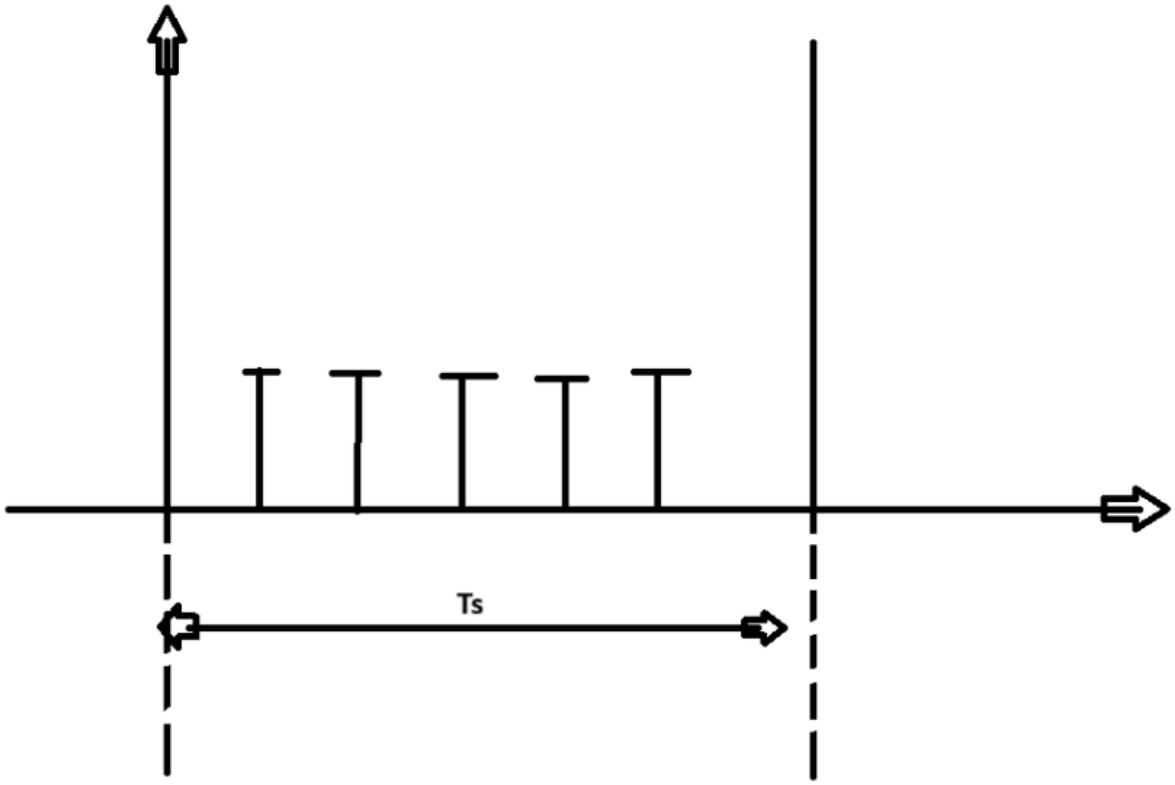


Рисунок 2 — Пример длительного символа

До этого момента все выполнялось программно, т.е это зона ответственности программиста, всё, что будет дальше - работа самой SDR.

Далее происходит двухканальная модуляция, где и 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** — компактная и автономная портативная SDR-платформа, разработанная компанией **Analog Devices** для обучения основам SDR и радиочастотных технологий, а также подходящая для радиолюбительских экспериментов.

Она сочетает в себе **приемопередатчик AD9363** и **процессор Xilinx Zynq**, позволяя генерировать и измерять аналоговые радиочастотные сигналы в широком диапазоне частот.

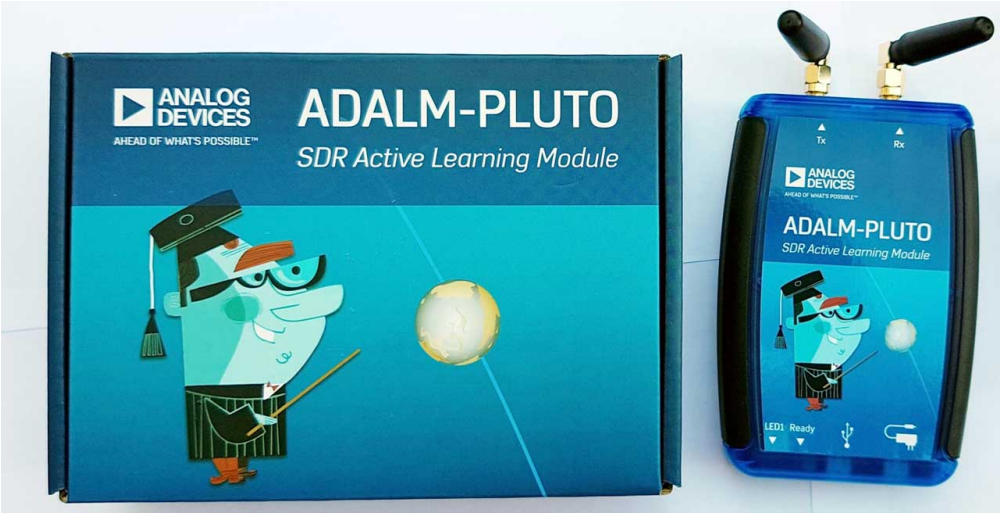


Рисунок 3 — Внешний вид Adalm Pluto

**Архитектура Adalm Pluto SDR**

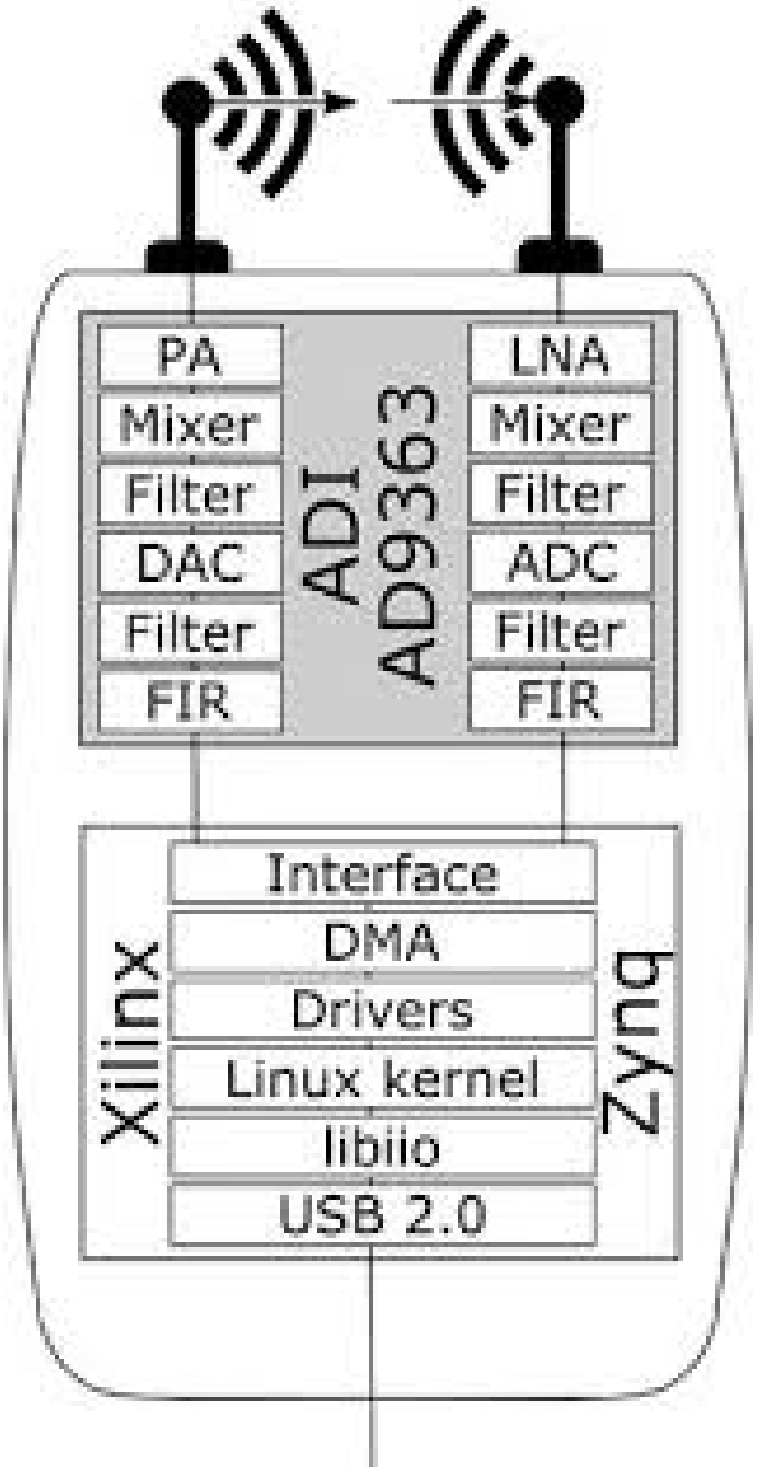


Рисунок 4 — Архитектура Adalm Pluto

**Описание основных блоков Adalm Pluto**

**PA (Power Amplifier)**

**Функция:** Усиление сигнала до уровня, требуемого для излучения.

**LNA (Low Noise Amplifier)**

**Функция:** Усиление слабого приёмного сигнала с минимальным добавлением шума перед его дальнейшей обработкой.

**ADC / DAC**

**ADC (Analog-to-Digital Conversion):** оцифровка аналогового сигнала в цифровой поток для последующей цифровой обработки.
**DAC (Digital-to-Analog Conversion):** преобразование цифровых семплов в аналоговый сигнал перед микшированием.

**FIR (Finite Impulse Response)**

**Функция:** детальная фильтрация, коррекция формы спектра, компенсация искажений.

**Mixer**

**TX:** перенос низкочастотного сигнала (baseband) на несущую частоту (subcarrier) для отправки в эфир.
**RX:** перенос высокочастотного сигнала на низкочастотный для дальнейшей обработки.

**Filter**

**TX:** фильтрация выходного сигнала перед передачей, подавление лишних гармоник.

**RX:** фильтрация принимаемого сигнала, подавление внеполосных помех перед оцифровкой.

**libiio**

**Описание:** библиотека, которая облегчает работу с устройствами ввода/вывода в **Linux**, особенно с радиочипами серии **AD936x**. Она даёт **API** для обмена данными и управления устройствами.

**Linux Kernel**

**Описание:** ядро **Linux**, управляющее процессором, памятью и устройствами ввода-вывода.

**DMA (Direct Memory Access)**

**Описание:** механизм, который автоматически переносит большие объёмы данных между устройством и памятью, разгружая процессор.

**Drivers**

**Описание:** инструкции для ядра, объясняющие, как правильно пользоваться конкретным оборудованием.

**Xilinx Zynq**

**Описание:** семейство микросхем от компании Xilinx. На одном кристалле объединены ARM-процессор (обычно на 2 ядра), который запускает Linux, управляет периферией, и ПЛИС (FPGA) для реализации аппаратных блоков (фильтры, ускорители обработки сигналов) и для более скоростных вычислений в реальном времени.

**USB 2.0**

**Описание:** порт для связи хоста и Adalm Pluto.

**GNU Radio**


Рисунок 5 — 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**

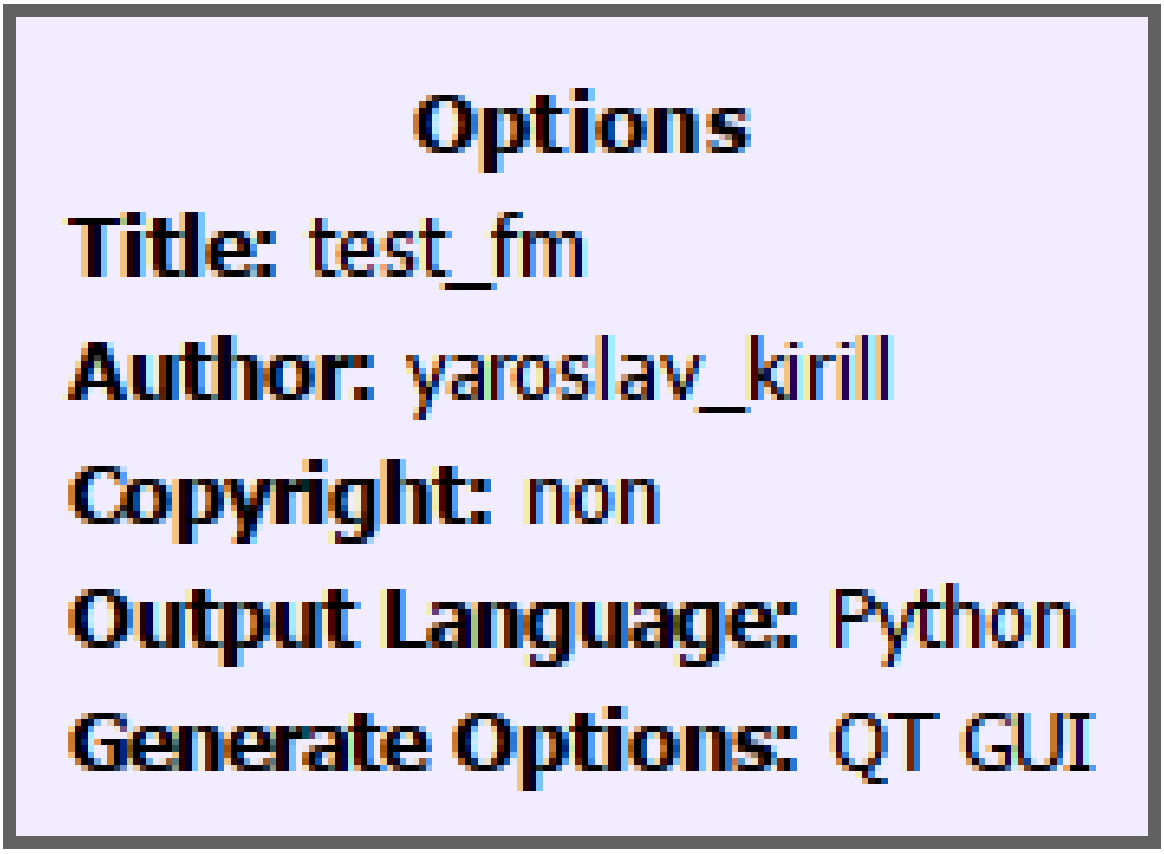


Рисунок 6 — Блок options

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

**Output Language** — язык, на котором будет сгенерирован код программы (у меня это Python).

**Generate Options** — используемый графический интерфейс (у меня это QT).

**Блок variable**

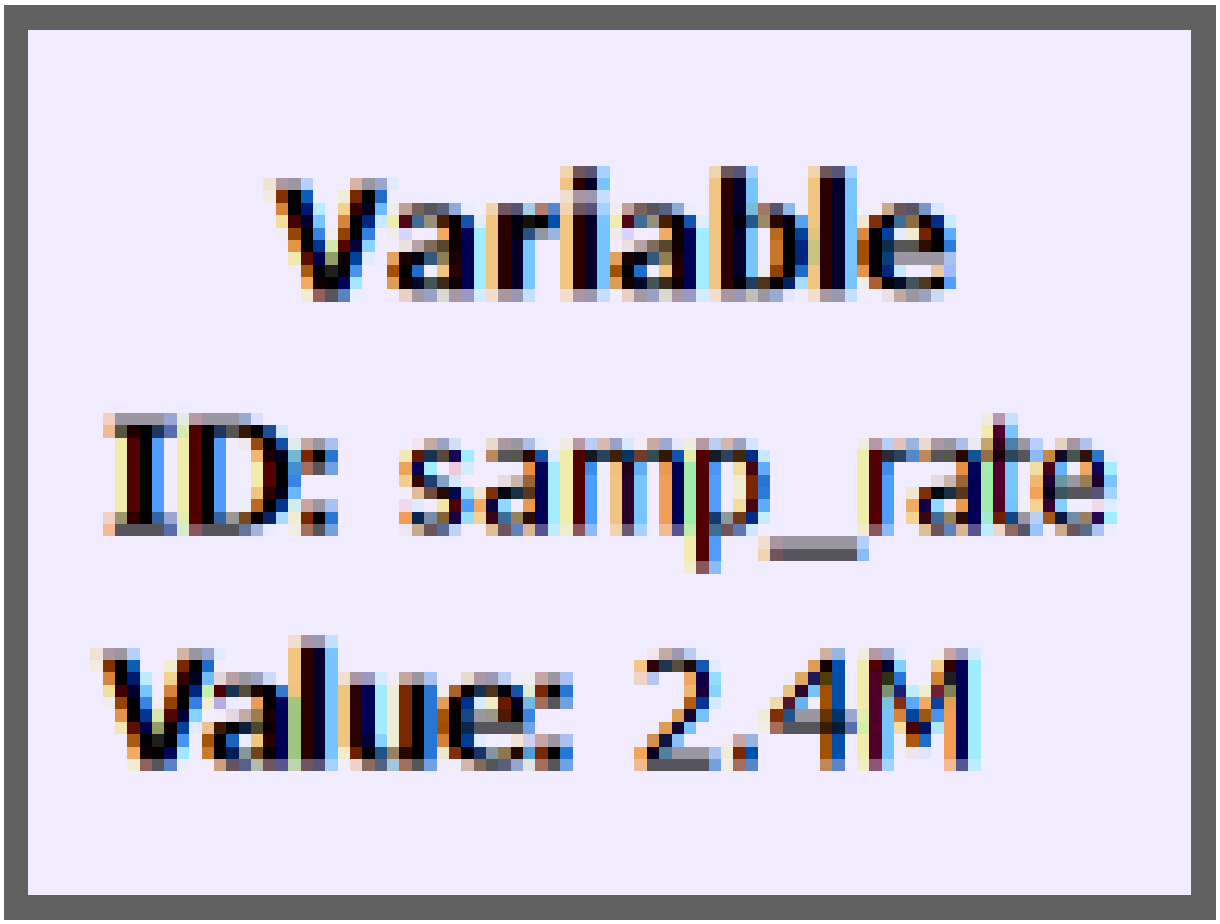


Рисунок 7 — Блок variable

В этом блоке можно задать переменную (почти как в языке программирования). Переменная имеет ID (имя) и значение. Я таким образом задаю переменную `samp_rate` (частоту дискретизации), равную $2.4 \times 10^6$ Hz.

**QT GUI Range**

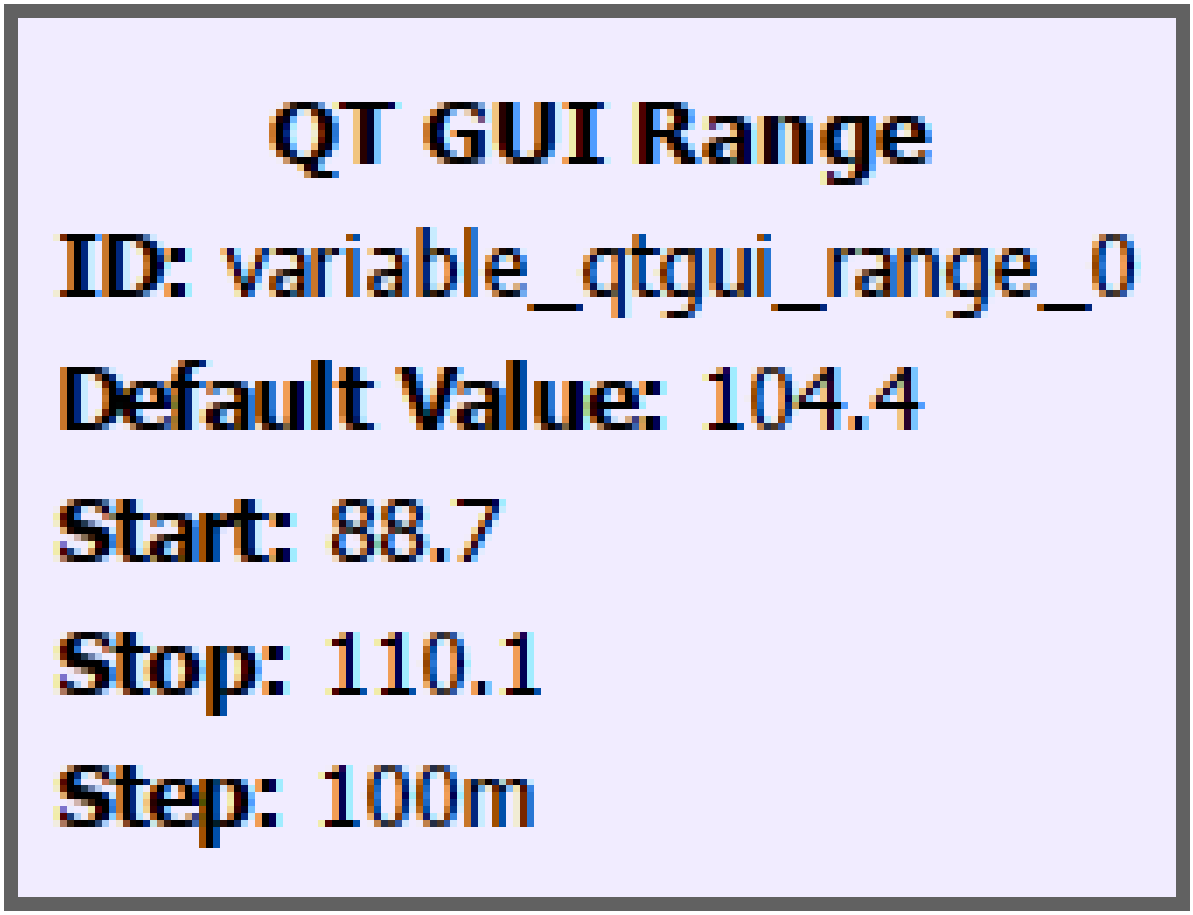


Рисунок 8 — Блок QT GUI Range

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. Я таким образом задаю 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**

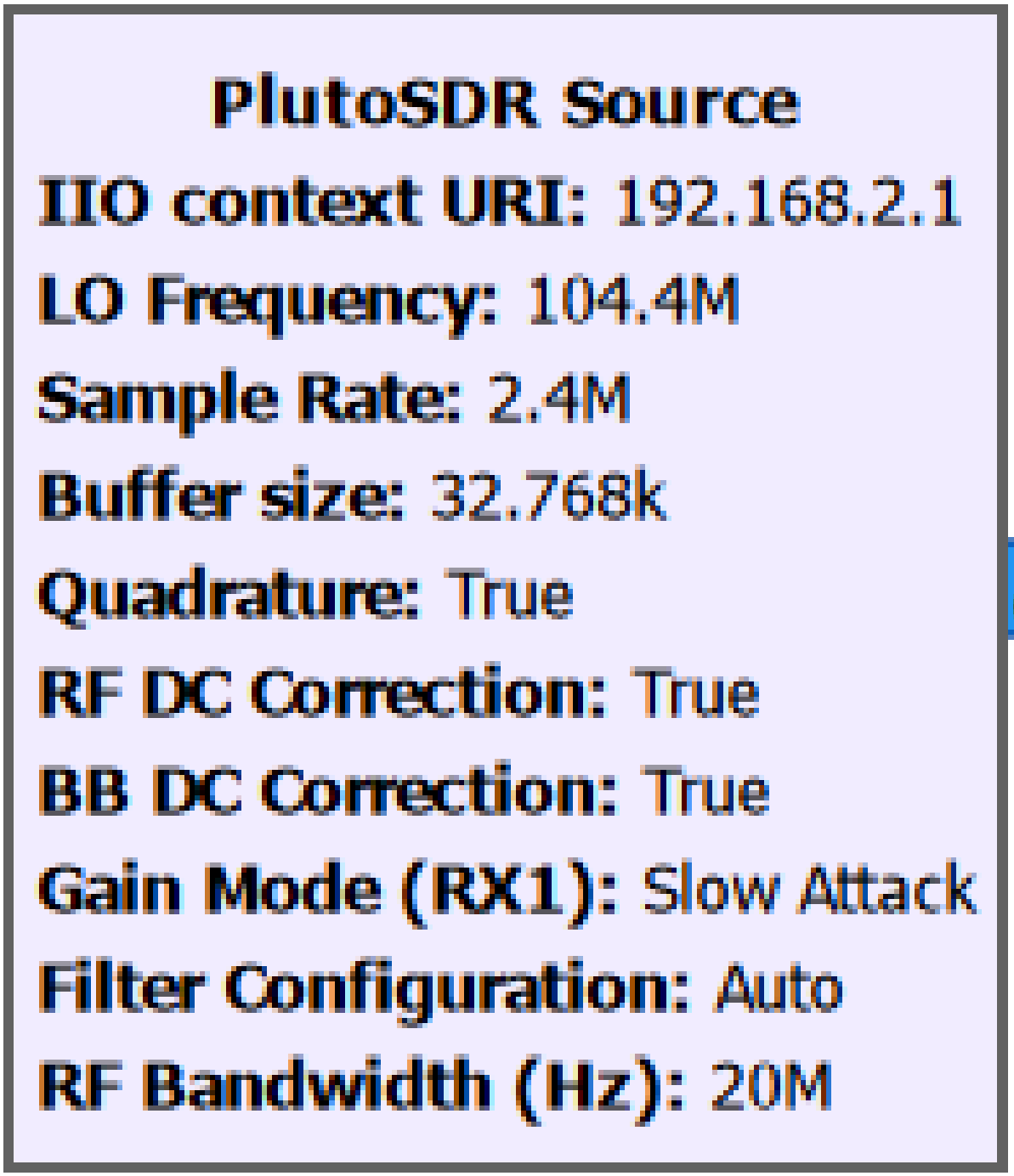


Рисунок 9 — Блок PlutoSDR Source

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

**Параметры:**

- **IIO context URI** — IP адрес Adalm Pluto, нужен, потому что PlutoSDR может подключаться по USB или сети (Ethernet/USB-Ethernet);
- **LO (Local Oscillator)** — локальный генератор частоты или центральная частота приёма, т.е. радиостанция, которую хотим слушать;
- **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;
- **RF DC Correction** — исправляет постоянную составляющую (DC offset), которая может появляться из-за несовершенства тракта;
- **BB DC Correction** — исправляет смещение в baseband-сигнале;
- **Gain Mode (RX1)** — режим автоматической регулировки усиления (AGC). Slow Attack — плавное изменение усиления;
- **Filter Configuration** — выбор полосовых фильтров в тракте SDR. В режиме Auto плата сама подбирает оптимальную конфигурацию фильтров;
- **RF Bandwidth** — полоса пропускания приёмного тракта.

**Low Pass Filter**

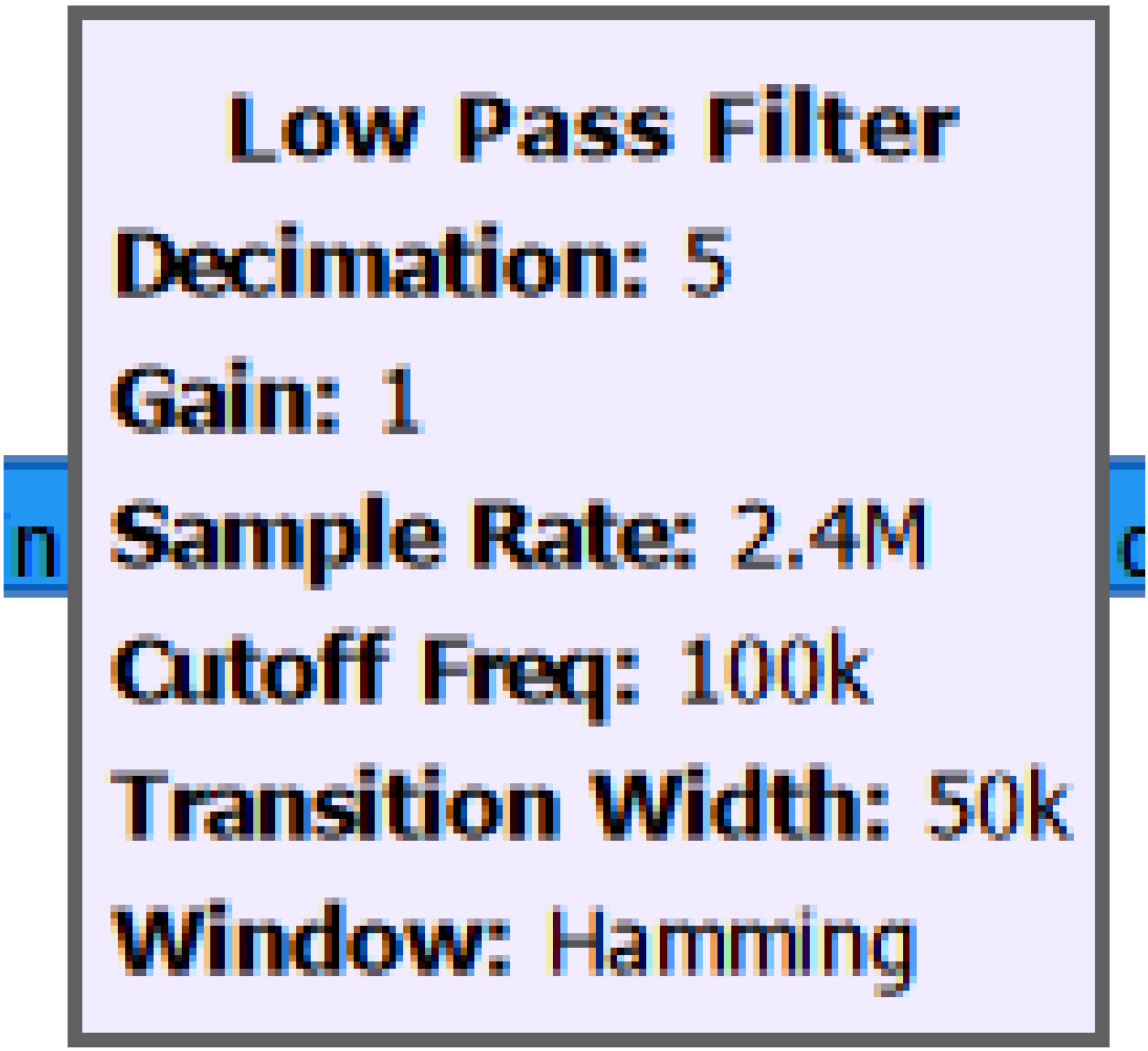


Рисунок 10 — Блок Low Pass Filter

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**

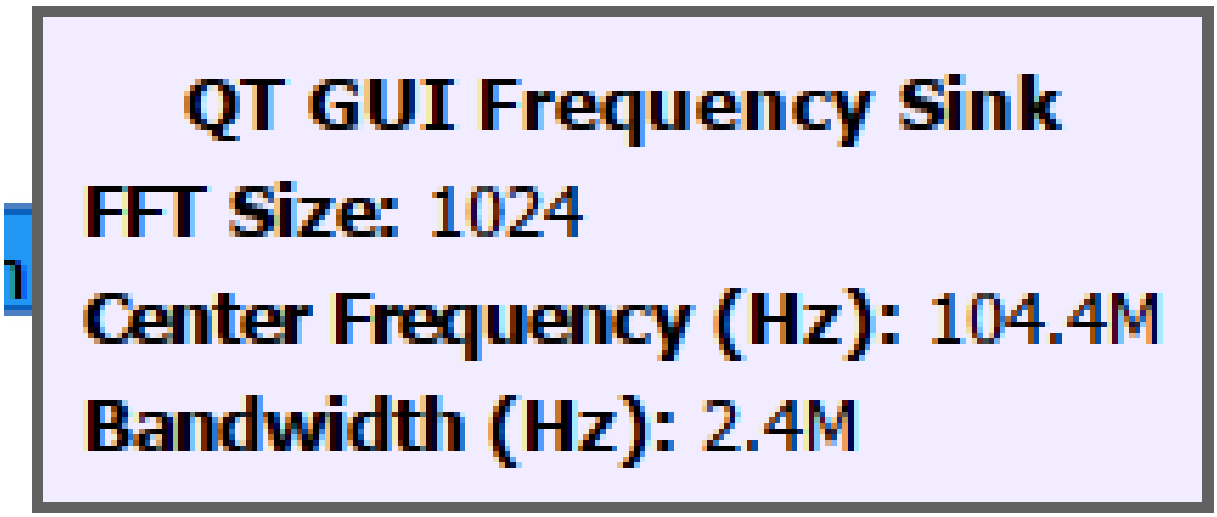


Рисунок 11 — Блок QT GUI Frequency Sink

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**

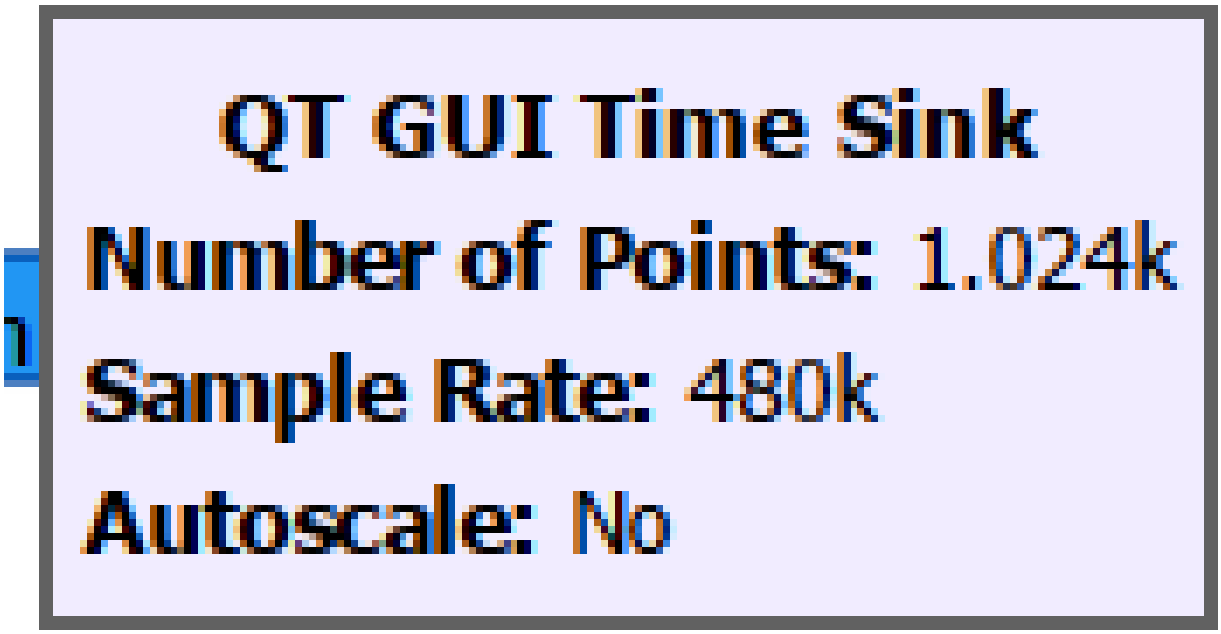


Рисунок 12 — Блок QT GUI Time Sink

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 13 — 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 14 — 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.

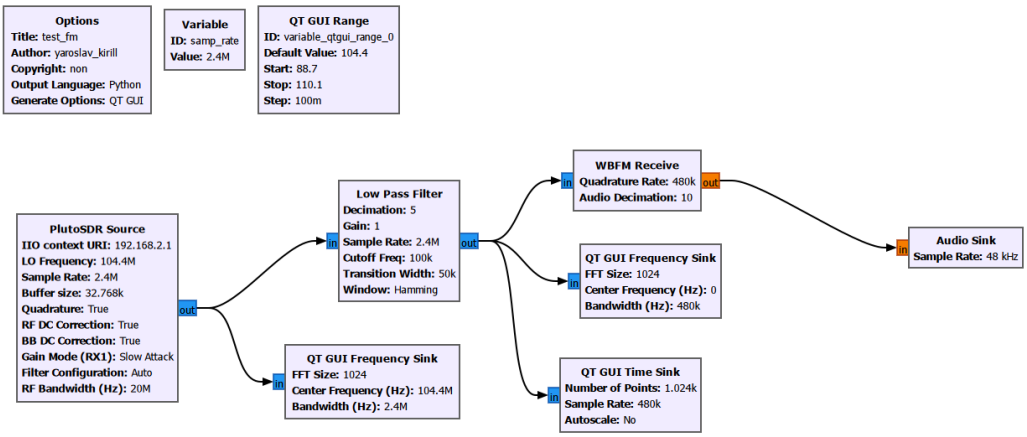


Рисунок 15 — Пример простой радиосистемы в GNURadio

Пример работы программы:

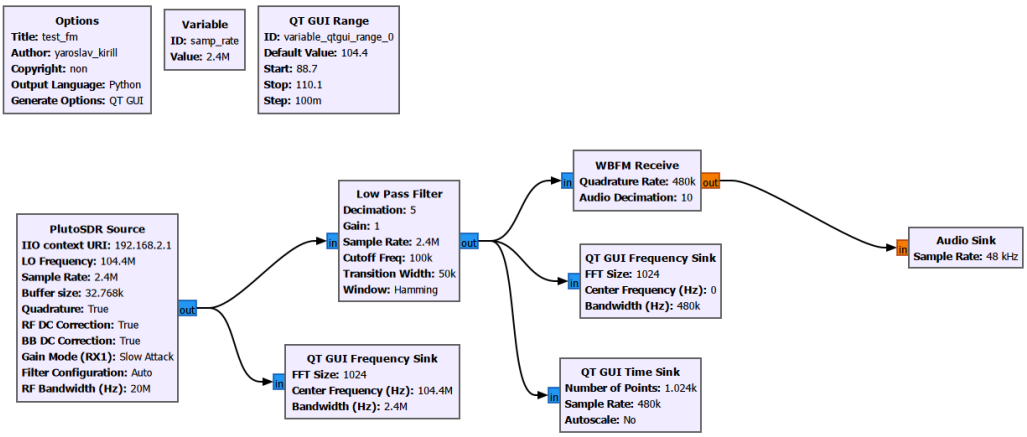


Рисунок 16 — Пример работы программы

# ВЫВОД

В ходе проделанной работы с помощью полученных знаний 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.