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:

ВВЕДЕНИЕ В АРХИТЕКТУРУ SDR-УСТРОЙСТВ. ЗНАКОМСТВО С БИБЛИОТЕКАМИ SOAPY SDR, LIBIO ДЛЯ РАБОТЫ С ADALM PLUTO SDR. ИНИЦИАЛИЗАЦИЯ SDR-УСТРОЙСТВА. РАБОТА С БУФЕРОМ: ПОЛУЧЕНИЕ ЦИФРОВЫХ IQ-ОТСЧЕТОВ.

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

На прошлом занятии мы рассматривали архитектуру простого передатчика и то, как в нем формируется и отправляется сигнал.

Сейчас разберем архитектуру приемника и то, как он принимает сигнал и преобразует его в данные.

Сигнал поступает на антенну приемника и первым делом нам нужно выяснить амплитуду sin и cos. Нужно выделить символы. Для этого подаем сигнал на демодулятор. Внутри него выполняются следующие действия: перемножает приемный сигнал на несущие сигналы (sin и cos), потом это подается на фильтр и на выходе получаем I и Q, но уже слегка измененные из-за помех сигнала. Далее эти I и Q поступают на ацп, который разобьет сигнал I и Q на семплы и в этих отчетах нужно выполнить символьную синхронизацию - выделить из этих семплов символы, которые потом нужно подать на демаппер (тот же маппер, но с обратной задачей)

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.

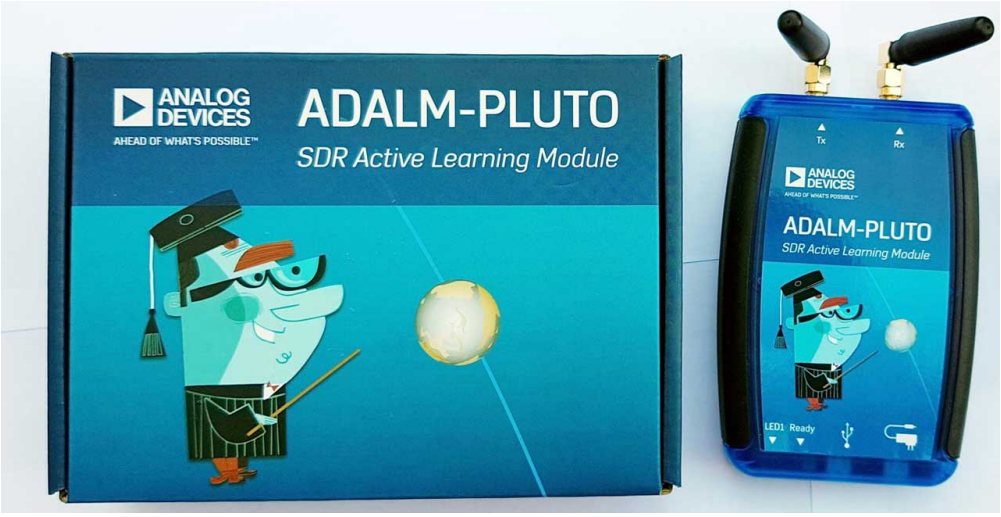

Рисунок 1 — Внешний вид Adalm Pluto

Архитектура Adalm Pluto SDR

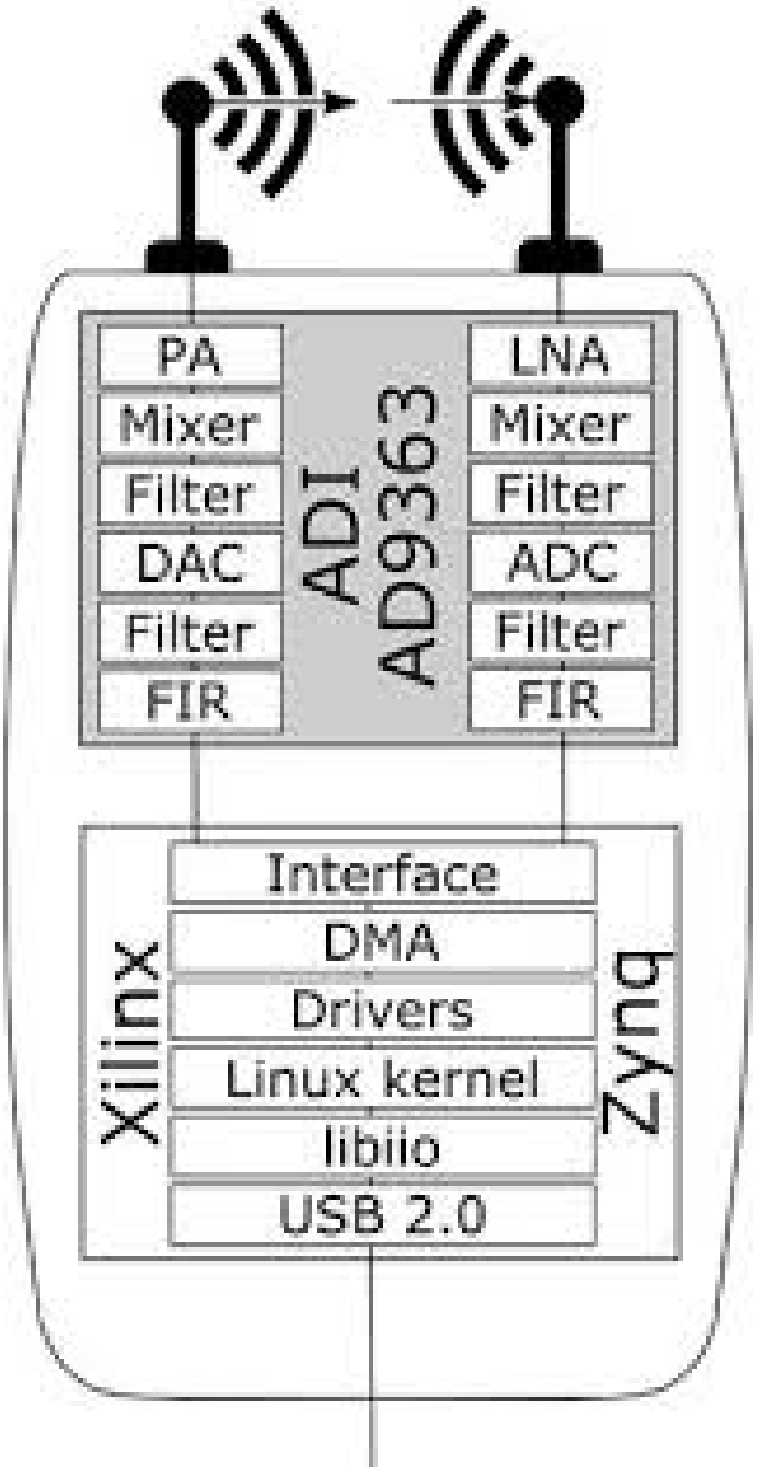

Рисунок 2 — Архитектура Adalm Pluto

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 3 — 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

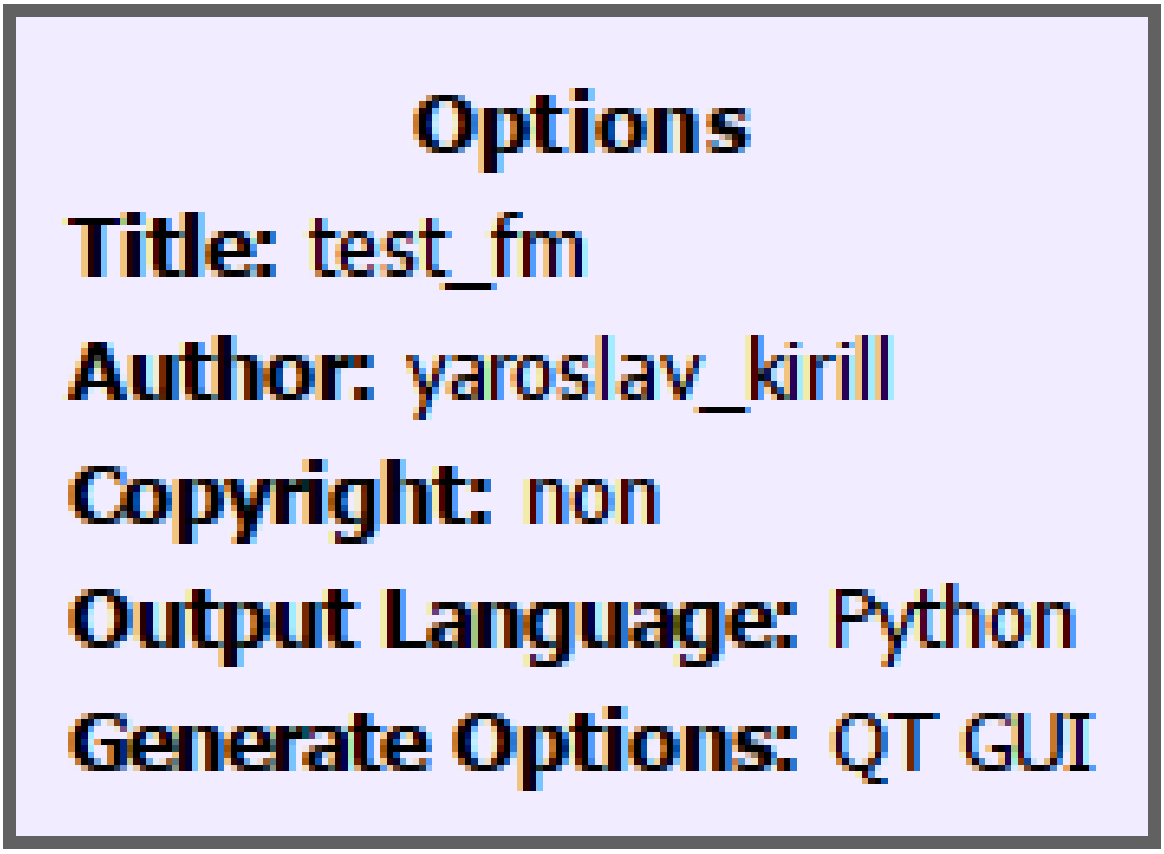

Рисунок 4 — Блок options

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

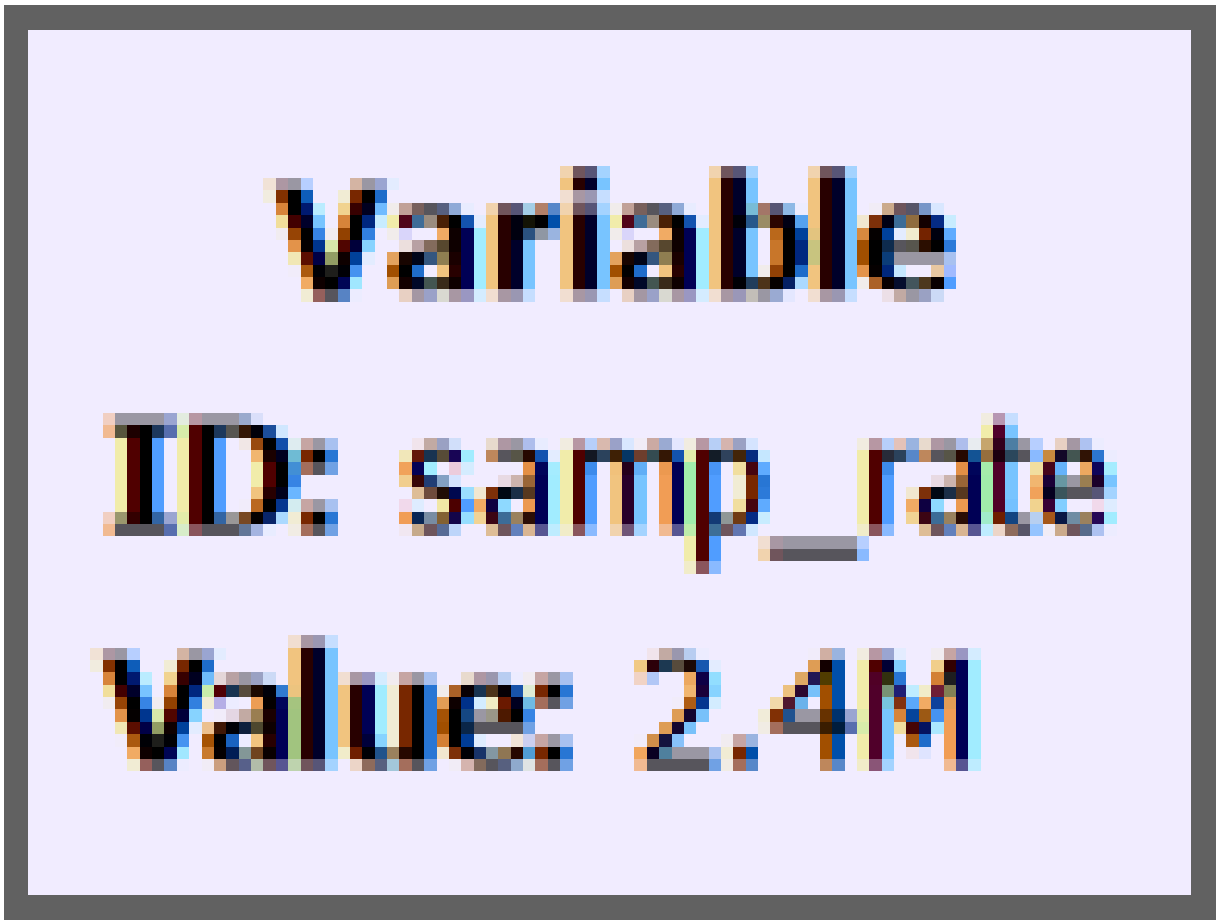

Рисунок 5 — Блок variable

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 2.4×10^6 Hz.

QT GUI Range

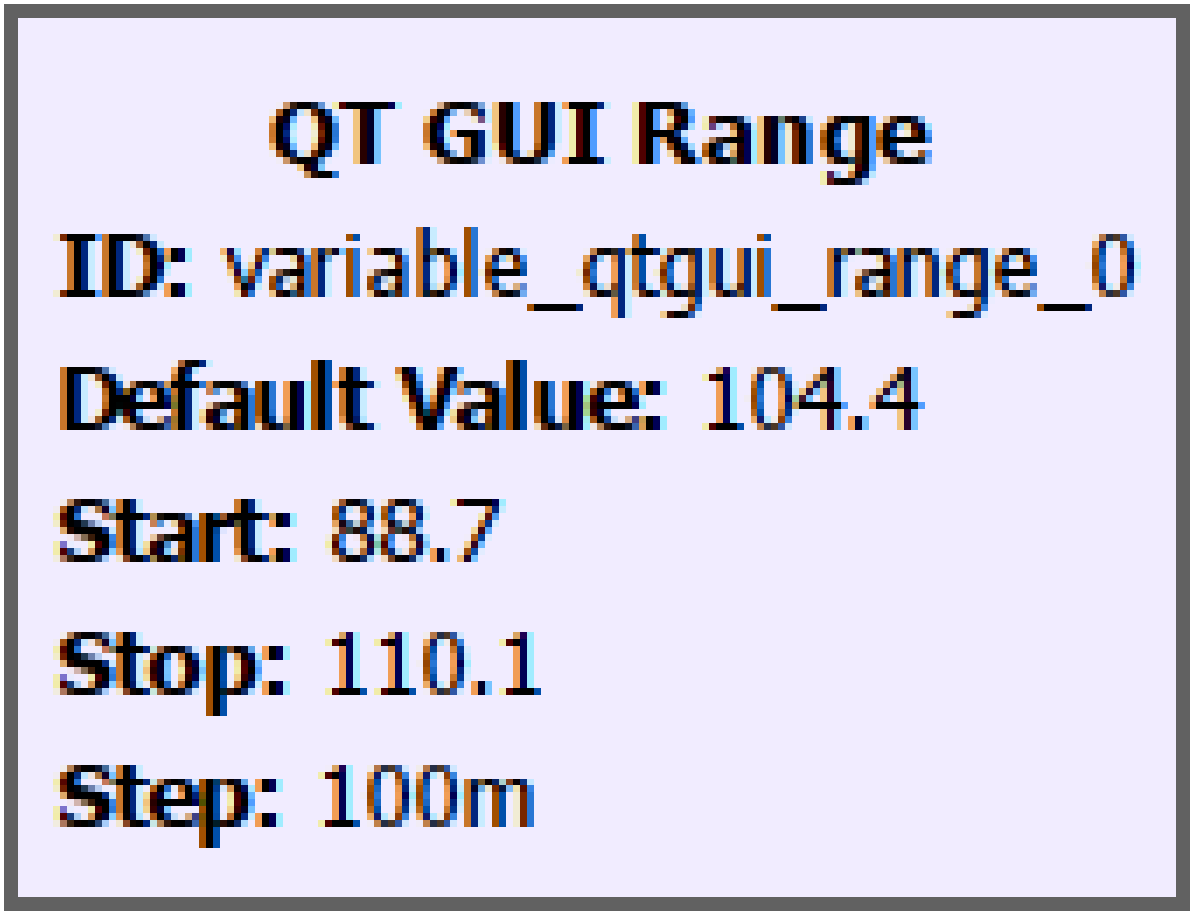

Рисунок 6 — Блок QT GUI Range

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

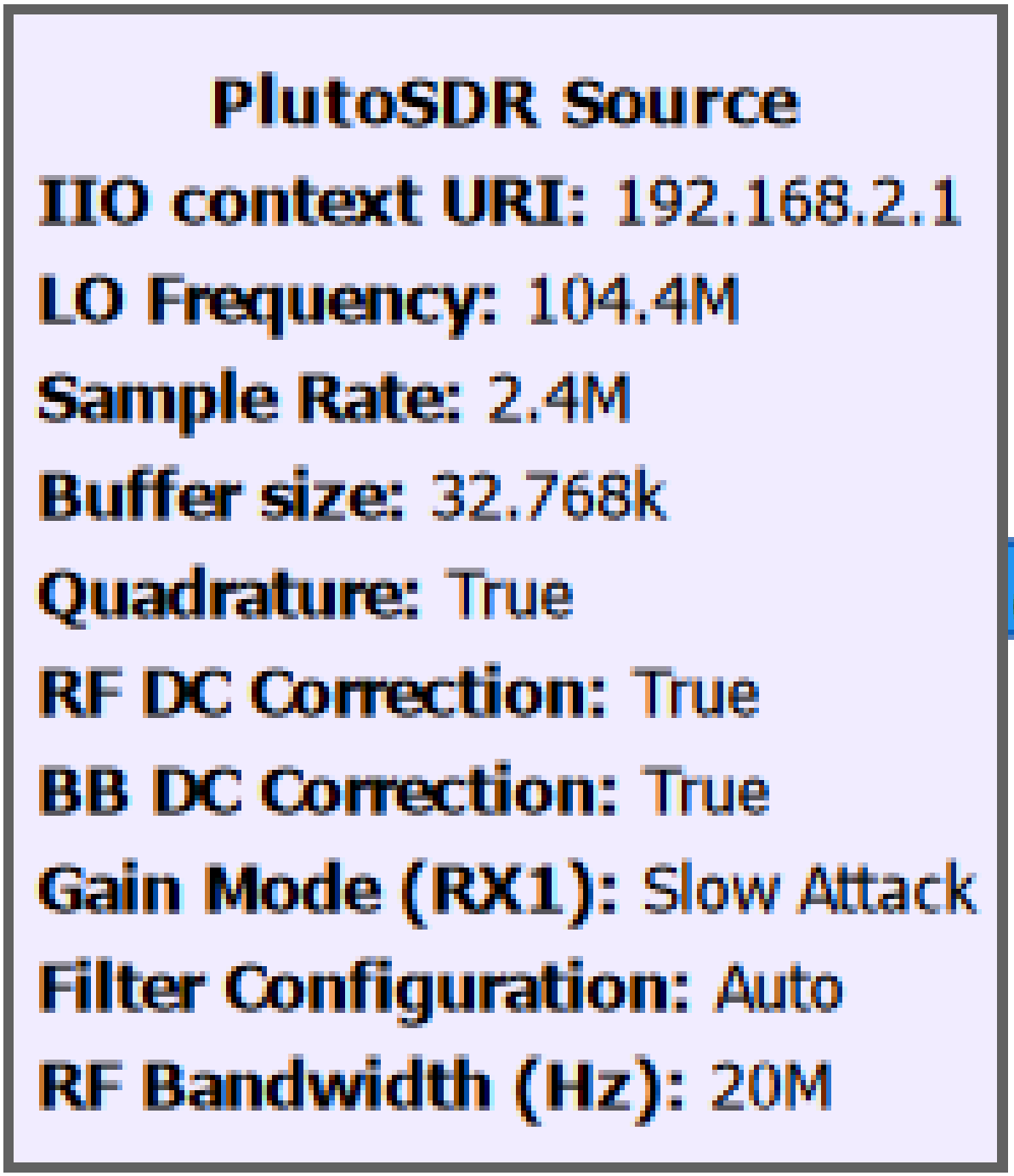

Рисунок 7 — Блок PlutoSDR Source

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

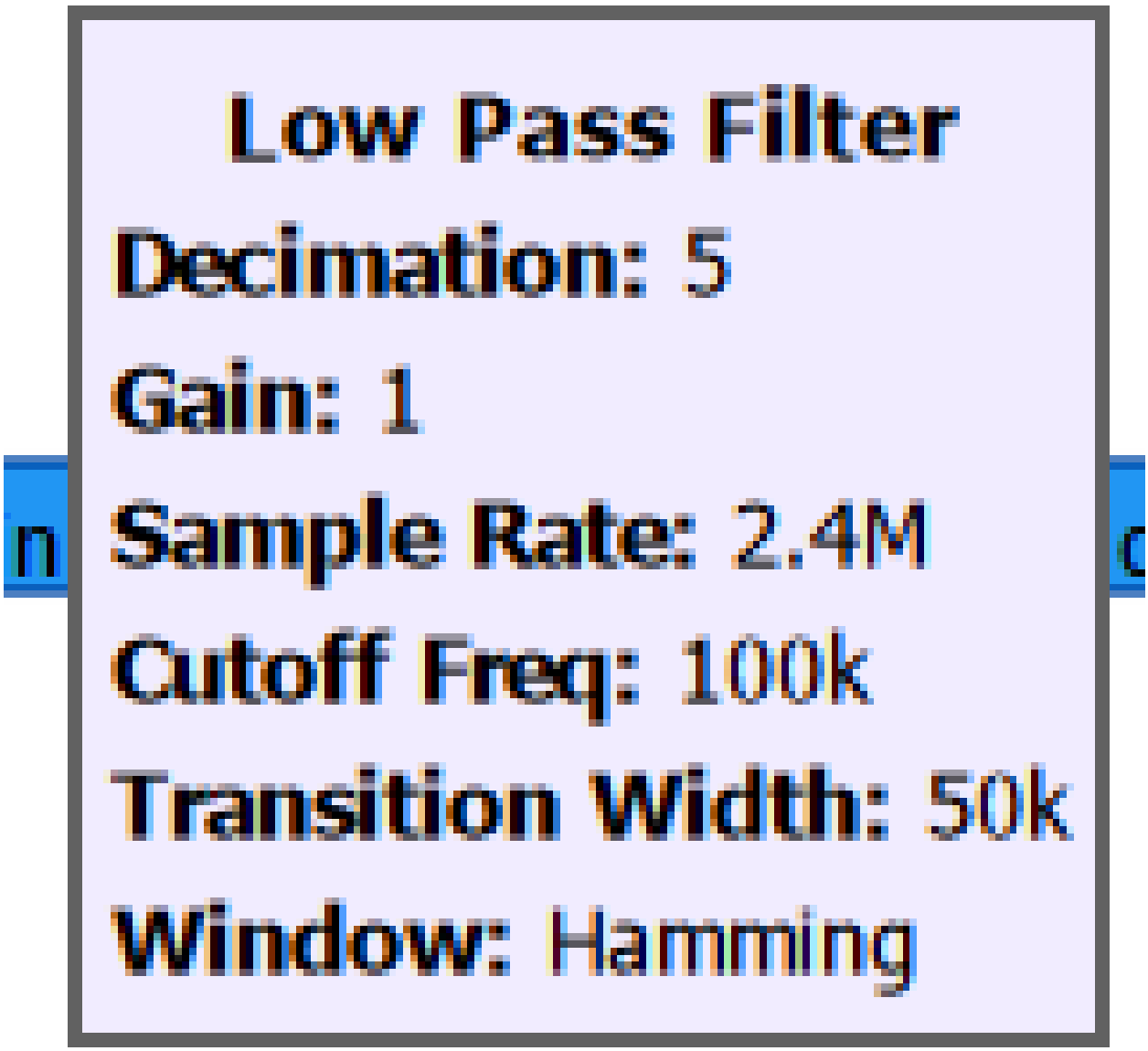

Рисунок 8 — Блок Low Pass Filter

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

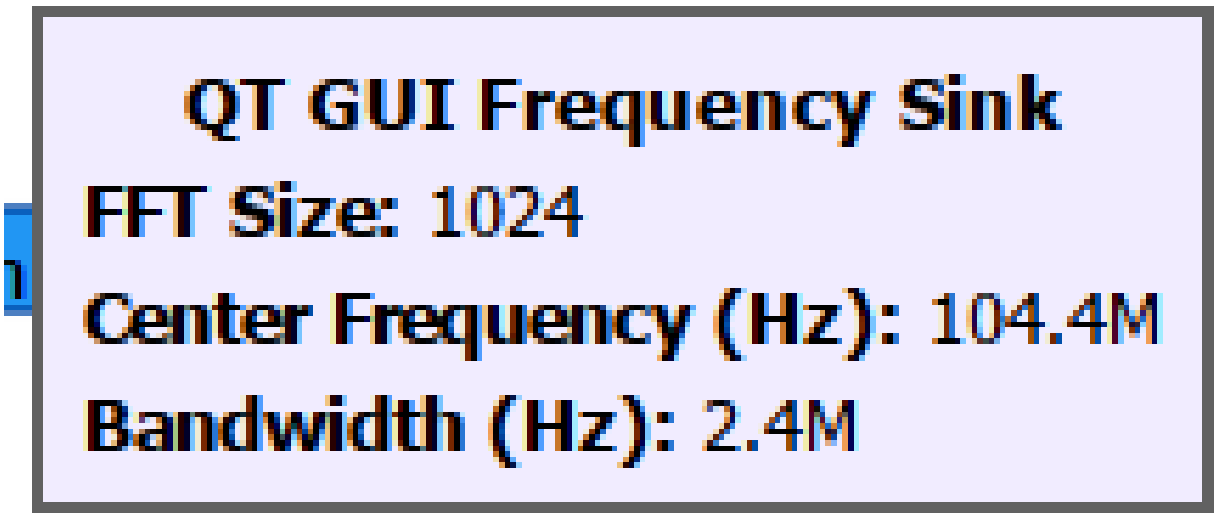

Рисунок 9 — Блок QT GUI Frequency Sink

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

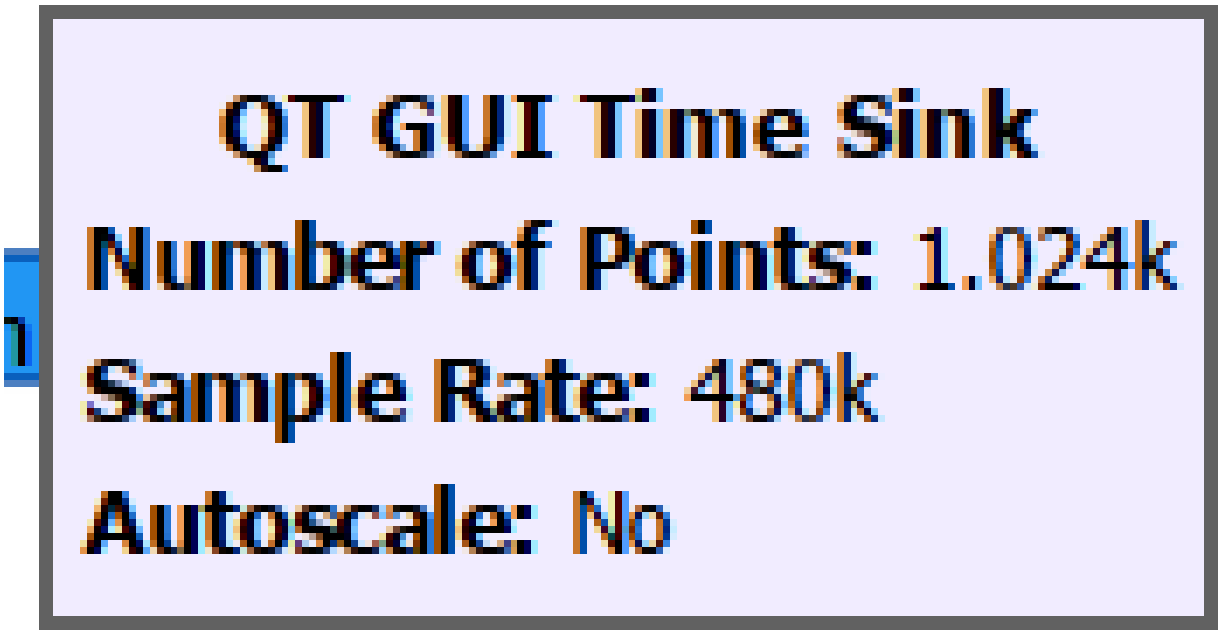

Рисунок 10 — Блок QT GUI Time Sink

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 11 — 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 12 — 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.

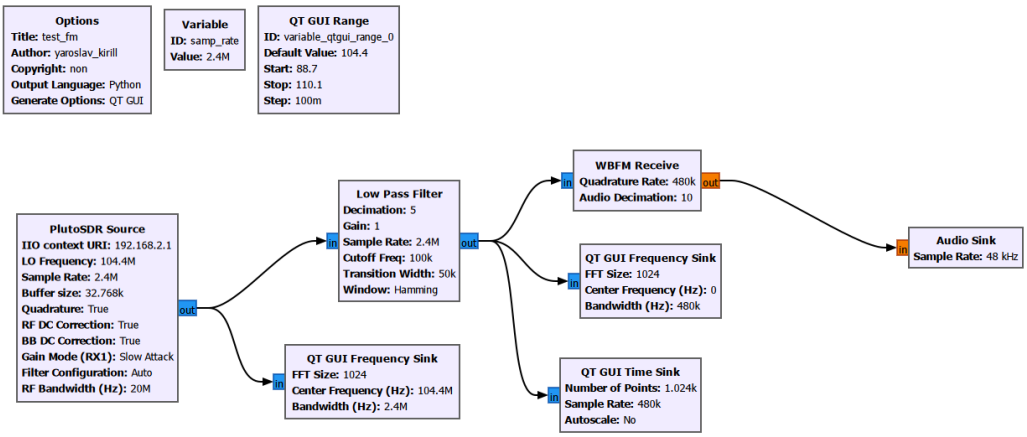

Рисунок 13 — Пример простой радиосистемы в GNURadio

Пример работы программы:

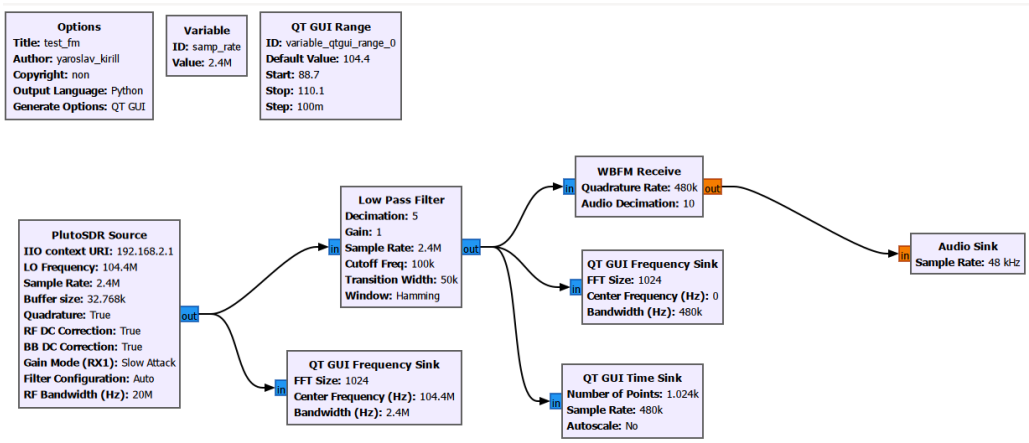

Рисунок 14 — Пример работы программы

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.