Санкт-Петербургский государственный политехнический
университет Петра Великого

**Высшая школа интеллектуальных систем и
суперкомпьютерных технологий**

Лабораторная работа

# GNU Radio

Выполнил студент гр. 3530901/80201 И.С. Иванов

Преподаватель: Н.В. Богач

Санкт-Петербург
2021

# Содержание

**1 Передача сигнала** **5**

**2 Добавление нарушений канала** **8**

**3 Восстановление `timing`** **10**

3.1 ISI . . . . . 10

3.2 Разные `clocks` . . . . . 11

3.3 Блок синхронизации многофазных часов . . . . . 12

3.4 Использование блока многофазной синхронизации . . . . . 15

**4 Многолучевость** **17**

**5 Эквалайзеры** **19**

5.1 CMA . . . . . 19

5.2 LMS DD . . . . . 20

**6 Фазовая и точная частотная коррекция** **22**

**7 Расшифровка** **24**

**8 Выводы** **26**

# Список иллюстраций


1 Flowgraph избыточной пропускной способности . . . . . 5

2 Избыточная пропускная способность . . . . . 6

3 Flowgraph созвездия QPSK . . . . . 6

4 Созвездие QPSK . . . . . 7

5 Flowgraph с добавлением нарушения канала . . . . . 8

6 Добавление нарушения канала . . . . . 9

7 Flowgraph проблемы ISI . . . . . 10

8 Передача 4-ех символов без рассинхронизации . . . . . 10

9 Flowgraph разные clocks . . . . . 11

10 Передача 4-ех символов с рассинхронизации . . . . . 12

11 Применение функции sample . . . . . 12

12 Нет смещения часов . . . . . 13

13 Есть смещение часов . . . . . 13

14 Flowgraph с несколькими фазами . . . . . 14

15 Несколько фаз . . . . . 14

16 Flowgraph с блоком многофазной синхронизации . . . . . 15

17 Использование блока многофазной синхронизации . . . . . 16

18 Flowgraph многолучевость . . . . . 17

19 Использование блока многофазной синхронизации . . . . . 17

20 Flowgraph с эквалайзером CMA . . . . . 19

21 Использование эквалайзера CMA . . . . . 20

22 Использование эквалайзера LMS DD . . . . . 21

23 Flowgraph для исправления сдвига фазы и частоты . . . . . 22

24 Исправление сдвига фазы и частоты . . . . . 23

25 Flowgraph декодирования . . . . . 24

26 Сравнение данных . . . . . 25

# 1 Передача сигнала

Первый этап - передача сигнала QPSK. Необходимо сгенерировать поток битов и смоделировать его на сложное созвездие. Для этого мы используем блок `Constellation Modulator`.

Объект созвездия позволяет нам определить, как кодируются символы. Затем блок модулятора может использовать эту схему модуляции с дифференциальным кодированием или без него. Модулятор созвездия ожидает упакованные байты, поэтому у нас есть генератор случайного источника, предоставляющий байты со значениями 0-255.

Количество выборок на символ должно быть минимальным для обеспечения желаемой скорости передачи данных. Мы будем использовать 4, что больше, чем нам нужно. Это необходимо для лучшей визуализации сигнала в различных областях.

Наконец, мы устанавливаем значение избыточной пропускной способности. Посмотрим на полученный `flowgraph`.

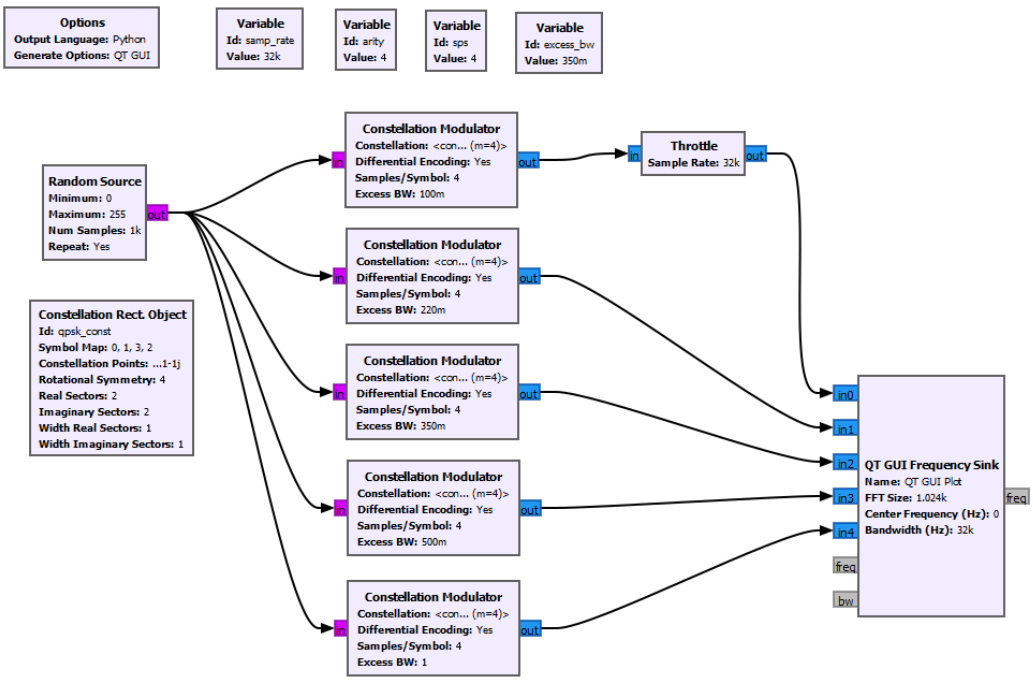


Рис. 1: Flowgraph избыточной пропускной способности

После запуска получаем график, показывающий различные значения избыточной пропускной способности. Типичные значения находятся в диапазоне от 0,2 (красная кривая) до 0,35 (зеленая кривая).

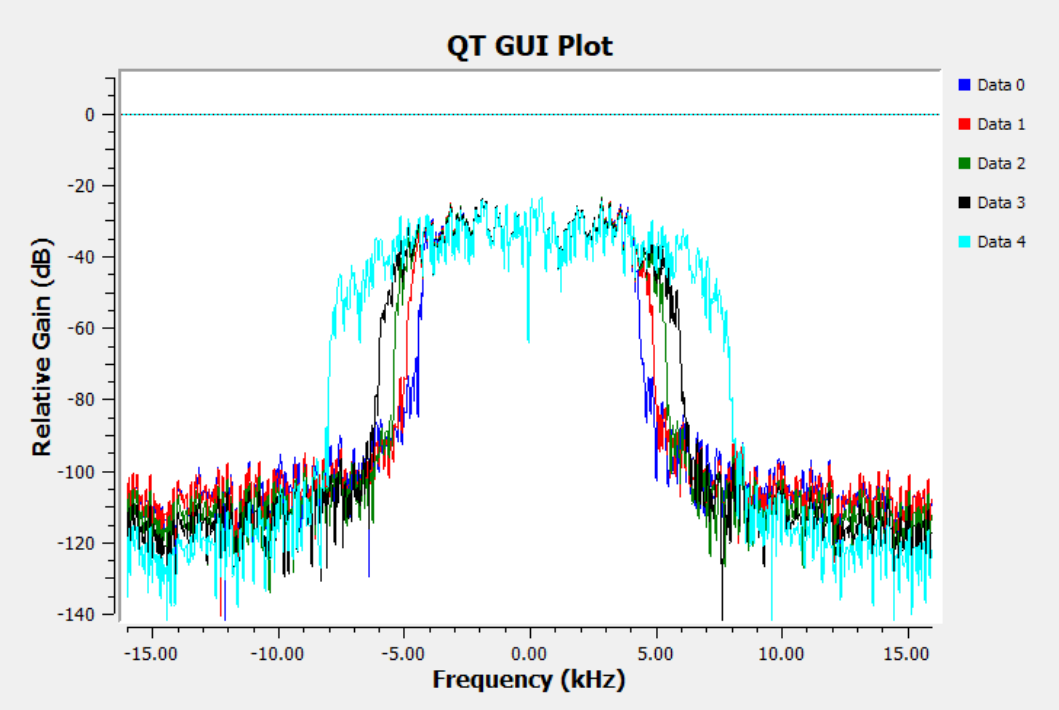


Рис. 2: Избыточная пропускная способность

Далее запустим файл `mpsk_stage1.grc`, передающий созвездие QPSK.

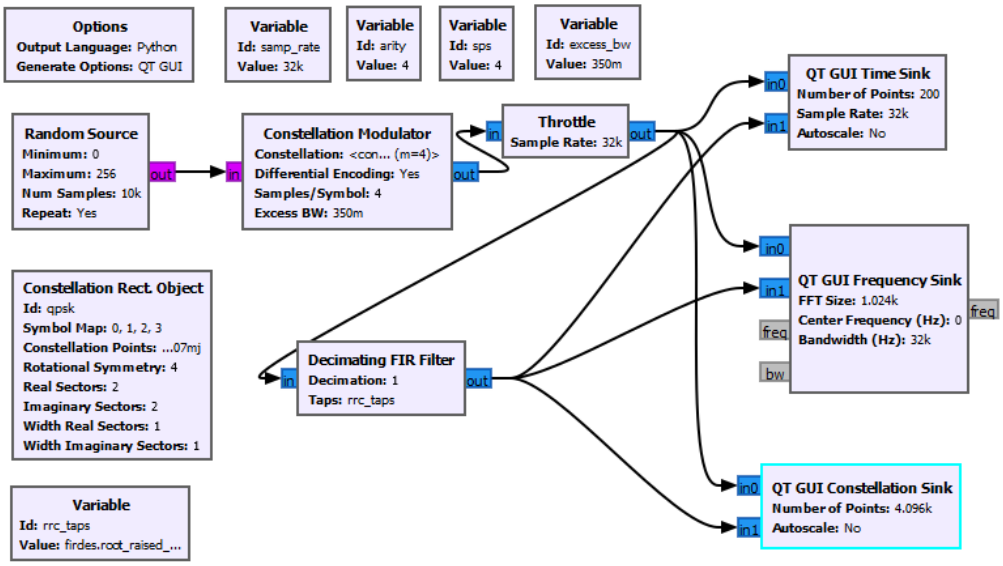


Рис. 3: Flowgraph созвездия QPSK

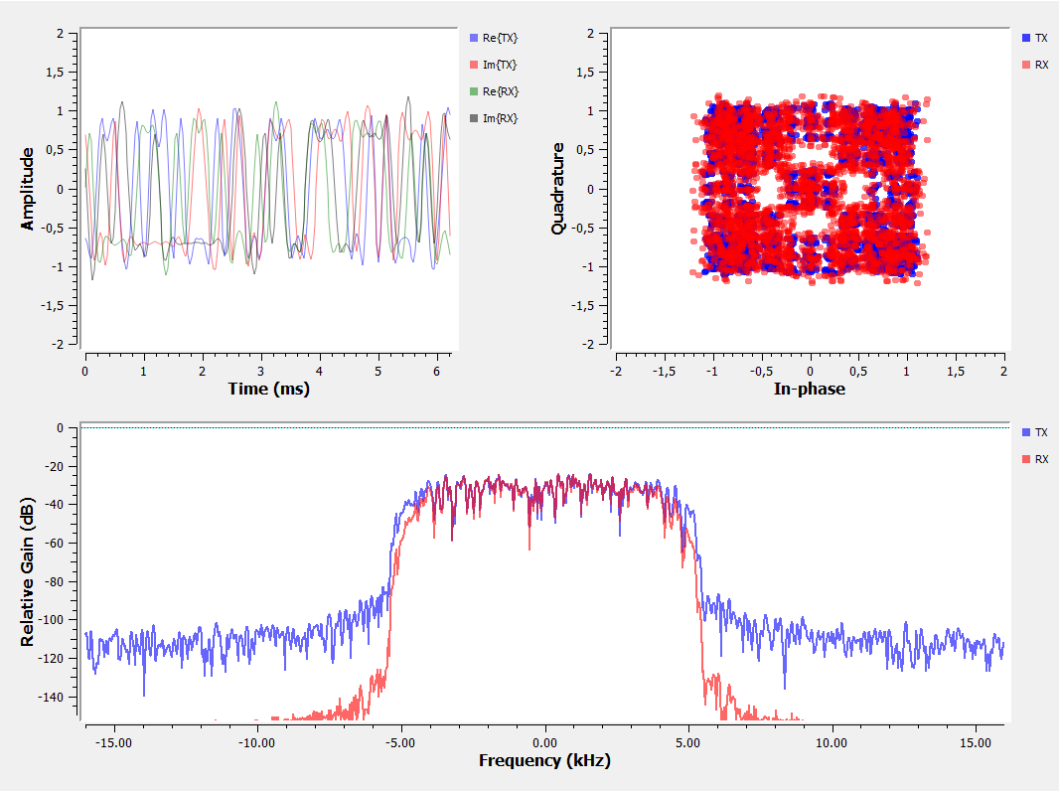


Рис. 4: Созвездие QPSK

На графике созвездия мы видим эффекты повышающей дискретизации и процесс фильтрации. В этом случае фильтр RRC добавляет преднамеренные самоинтерференции(ISI).

На стороне приема мы избавляемся от ISI с помощью другого фильтра RRC. В результате, мы сворачиваем два фильтра RRC вместе, и получаем фильтр с приподнятым косинусом.

Фильтрация представляет собой просто свертку, поэтому выходной сигнал RRC-фильтра на приемной стороне представляет собой сигнал в форме приподнятого косинусоидального импульса с минимизированным ISI. Другое преимущество состоит в том, что при отсутствии эффектов канала мы используем согласованный фильтр на приемнике.

# 2 Добавление нарушений канала

Теперь мы рассмотрим влияние канала на то, как сигнал искажается между передачей и приёмом. Для начала мы будем использовать самый простой блок модели канала GNU Radio, который позволит нам смоделировать несколько основных проблем связанных с приёмником.

Проблемы:

1. Шум. В нашем приемнике вызывается аддитивный белый гауссовский шум. Мы устанавливаем мощность этого шума, регулируя значение напряжения шума модели канала.
2. `clock`, определяющие частоту радиомодулей. Два радиомодуля имеют разные `clock`. Одно радио работает на частоте `fc + f_delta_1`. Другое радио имеет другие `clock` и его реальная частота равна `fc + f_delta_2`.
3. Таким образом, полученный сигнал будет `f_delta_1 + f_delta_2`.
4. Идеальная точка выборки. Идеальная точка выборки связана с проблемой часов. Два радиомодуля работают с разной скоростью, поэтому идеальная точка выборки неизвестна.

Результат моделирования позволяет нам поработать с аддитивным шумом, сдвигом частоты и временным сдвигом.

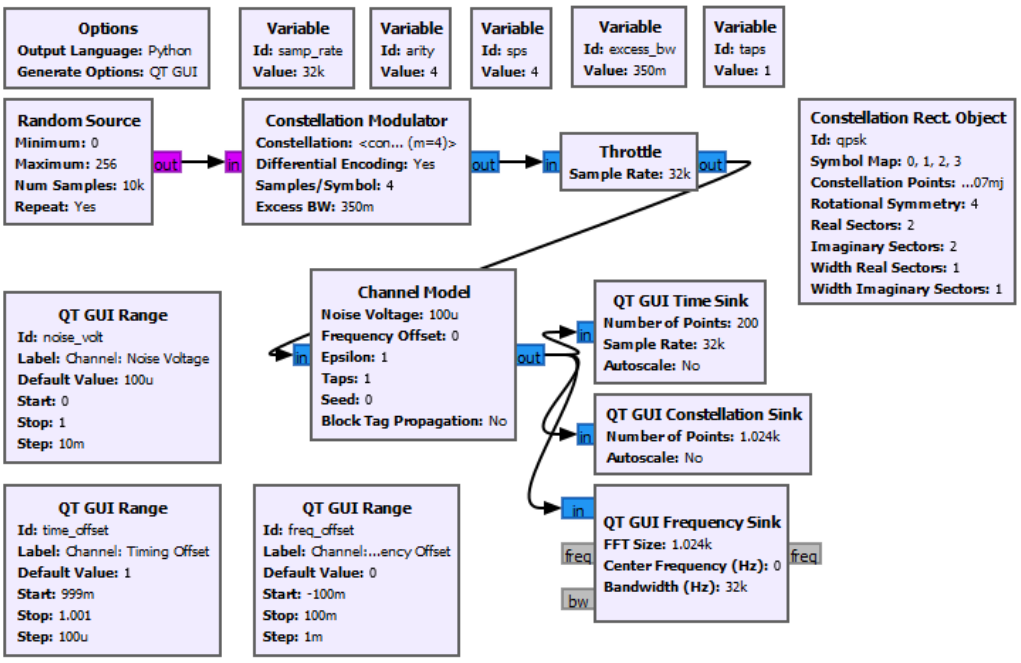


Рис. 5: Flowgraph с добавлением нарушения канала

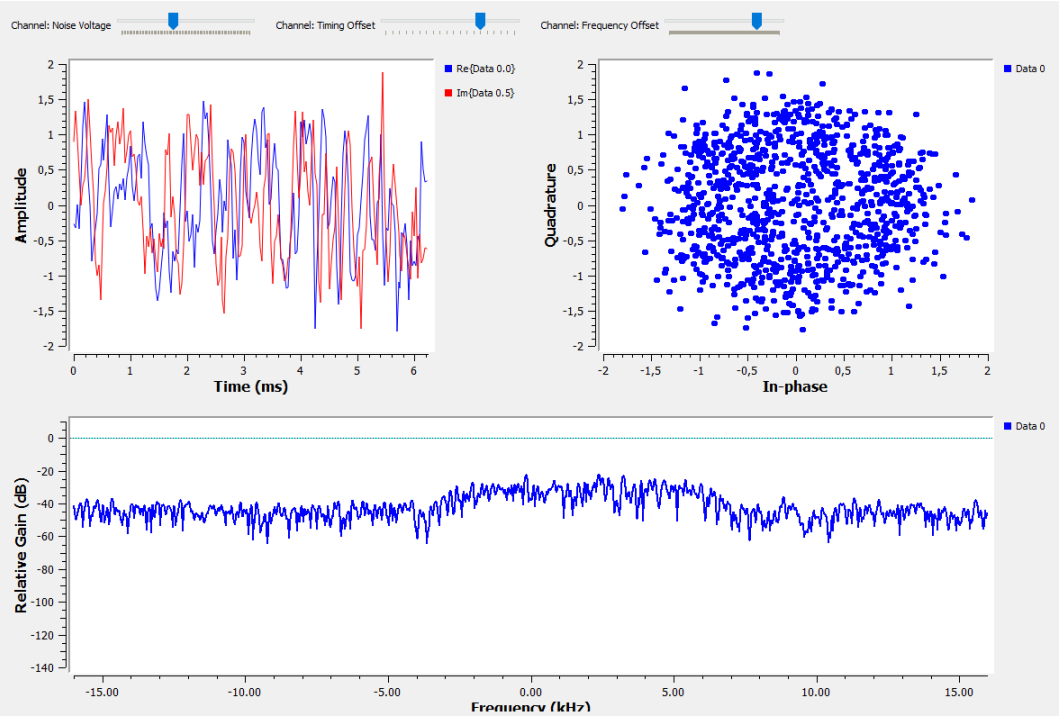


Рис. 6: Добавление нарушения канала

График созвездия показывает нам облако образцов, намного хуже того, с чего мы начали на последнем этапе. Теперь, исходя из полученного сигнала, мы должны отменить все эти эффекты.

# 3 Восстановление `timing`

Теперь перейдём к процессу восстановления. Здесь мы будем использовать алгоритм восстановления многофазных `clocks`. Мы начнем с восстановления `timing`. Мы пытаемся найти наилучшее время для дискретизации входящих сигналов, что позволит максимизировать отношение сигнал/шум (SNR) каждой выборки, а также уменьшить влияние межсимвольных помех (ISI).

## 3.1 ISI

Проблему ISI мы можем увидеть на примере flowgraph `symbol_sampling.grc`, где мы создаем четыре отдельных символа, а затем фильтруем их.

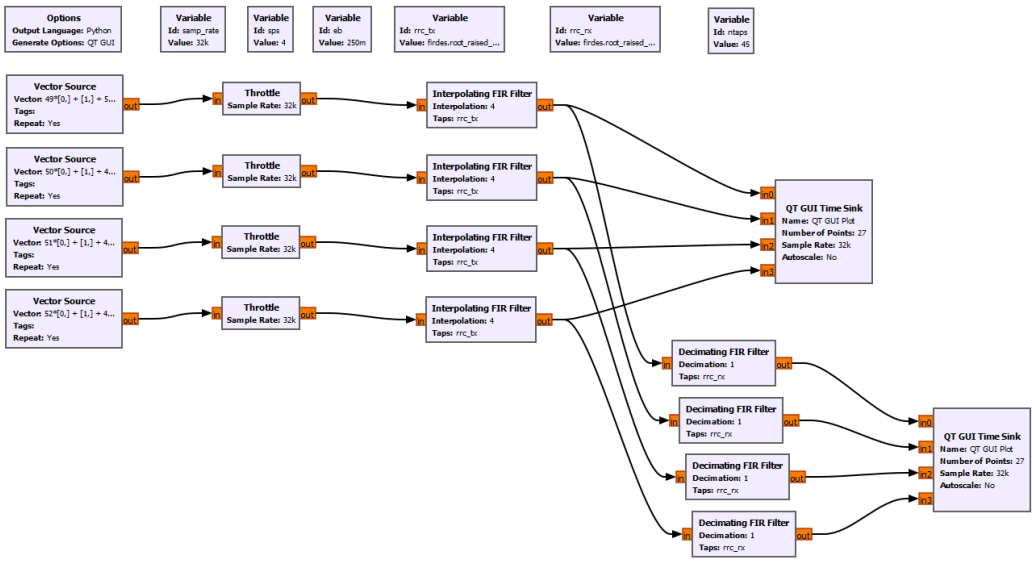


Рис. 7: Flowgraph проблемы ISI

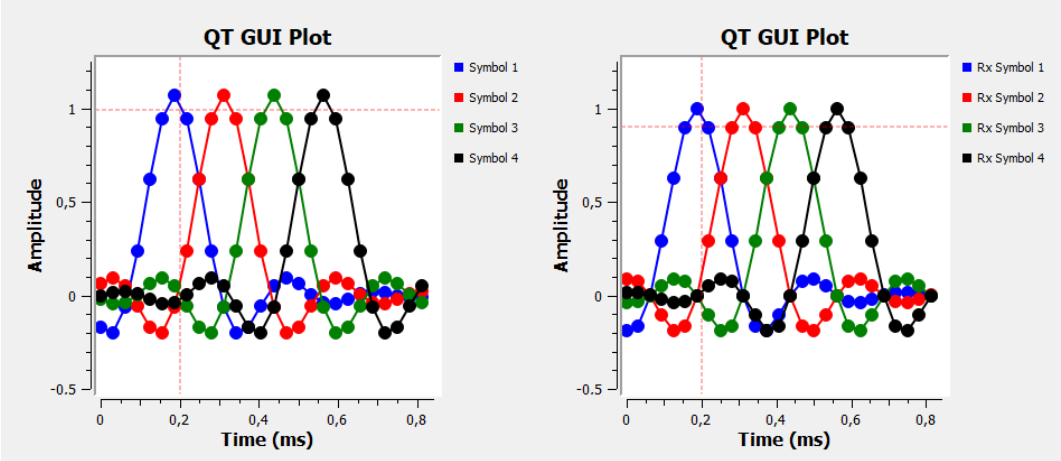


Рис. 8: Передача 4-ех символов без рассинхронизации

На графике видны различия между символами, отфильтрованными RC и RRC. Без фильтрации Найквиста (RRC - справа) мы можем увидеть, как в идеальной точке выборки каждого символа другие символы имеют некоторую энергию. Если мы суммируем эти символы вместе (как в непрерывном потоке), то энергия других выборок складывается вместе и искажает символ в этой точке. И наоборот, на выходе с RC-фильтром энергия от других выборок равна 0 в идеальной точке дискретизации для данного символа во времени. Это означает, что если мы делаем выборку точно в правильной точке выборки, мы получаем энергию только от текущего символа без помех от других символов в потоке. Это моделирование позволяет нам легко настроить количество выборок на символ, избыточную полосу пропускания фильтров RRC и количество ответвлений.

## 3.2 Разные `clocks`

Затем мы посмотрим, как разные часы влияют на точки выборки между передатчиком и приемником. Используя пример потокового графа в `symbol_sampling_diff.g` мы моделируем влияние разных часов в передатчике и приемнике.

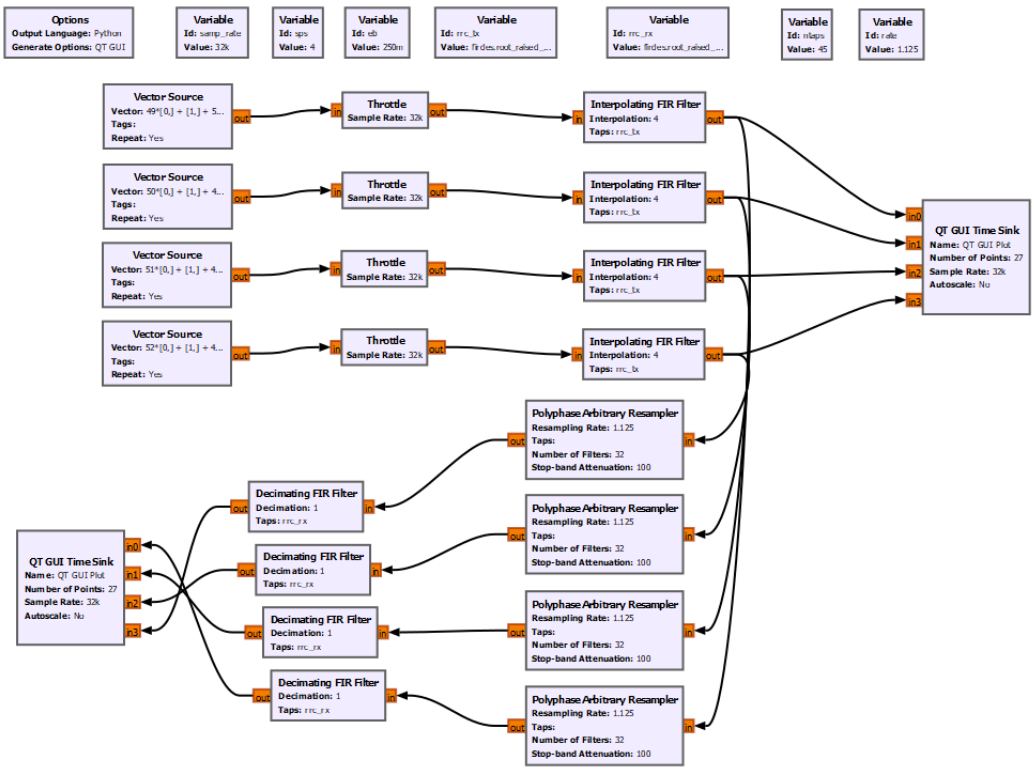


Рис. 9: Flowgraph разные clocks

Все часы несовершенные, поэтому они:

1. начнут свою работу в разные моменты времени;

2. дрейфуют относительно других часов.

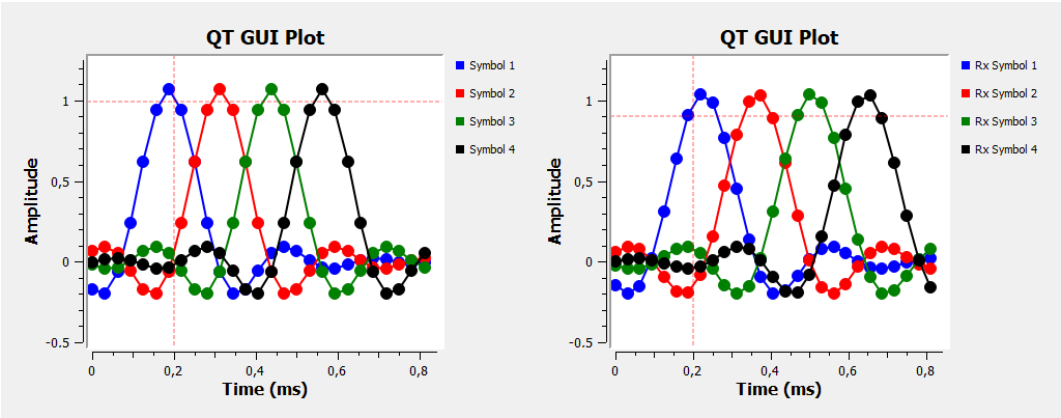


Рис. 10: Передача 4-ех символов с рассинхронизации

Наша задача - синхронизировать часы передачи и приемника, используя только информацию в приемнике из входящих выборок. Это задача известна как восстановление часов или времени.

## 3.3 Блок синхронизации многофазных часов

Блок вычисляет первый дифференциал входящего сигнала, который будет связан с его смещением тактовой частоты. В примере flowgraph `symbol_differential_f` можно увидеть графики с параметром скорости 1 (т.е. нет смещения часов).

Нам нужен образец 0,22 мс. Фильтр разности ([-1, 0, 1]) генерирует дифференциал символа, и выходной сигнал этого фильтра в правильной точке выборки равен 0. Затем мы можем инвертировать этот оператор и вместо этого сказать, когда выход дифференциального фильтра равен 0, мы нашли оптимальную точку выборки.

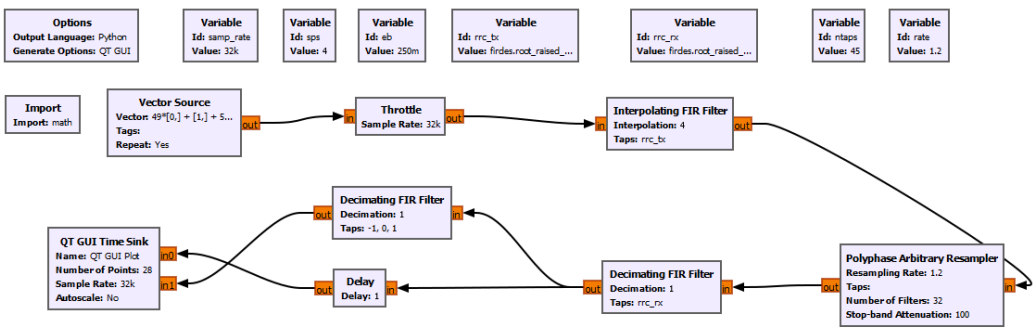


Рис. 11: Применение функции sample

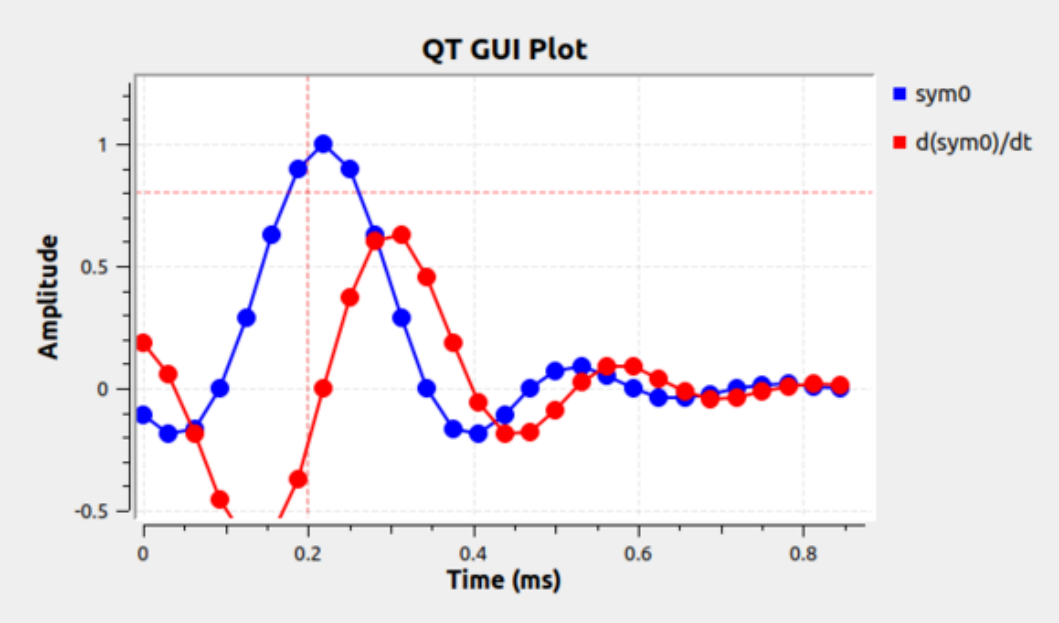


Рис. 12: Нет смещения часов

У нас есть временное смещение там, где пик символа выключен, а производный фильтр не показывает нам нулевую точку.

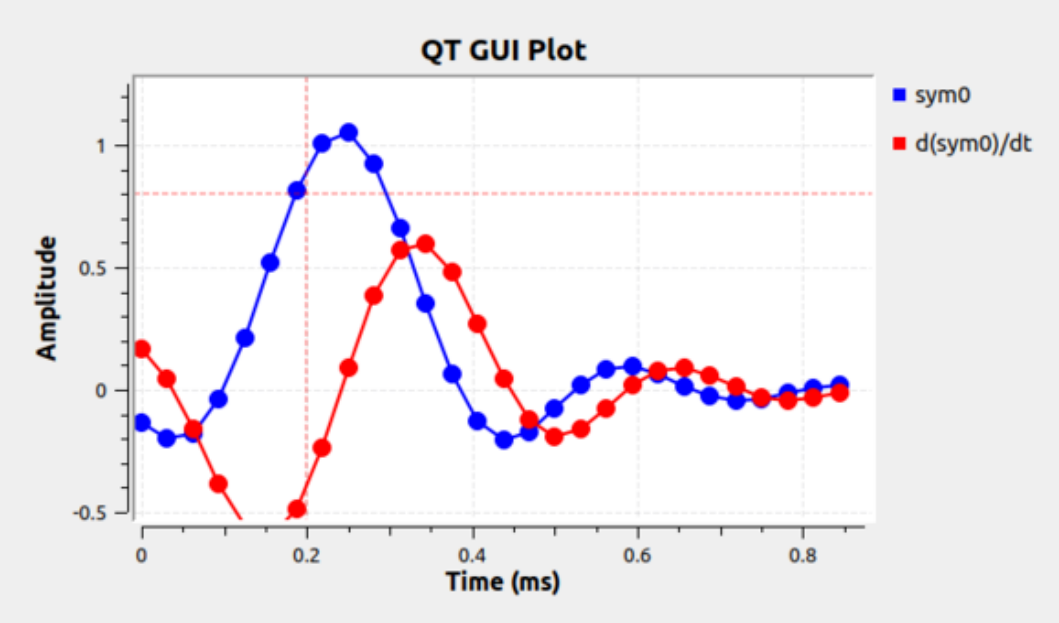


Рис. 13: Есть смещение часов

Вместо использования одного фильтра мы можем создать серию фильтров, каждый с разной фазой. Если у нас достаточно фильтров на разных фазах, один из них имеет правильную фазу фильтра, которая даст нам желаемое значение синхронизации. Посмотрим на симуляцию, которая строит 5 фильтров, что означает 5 различных фаз.

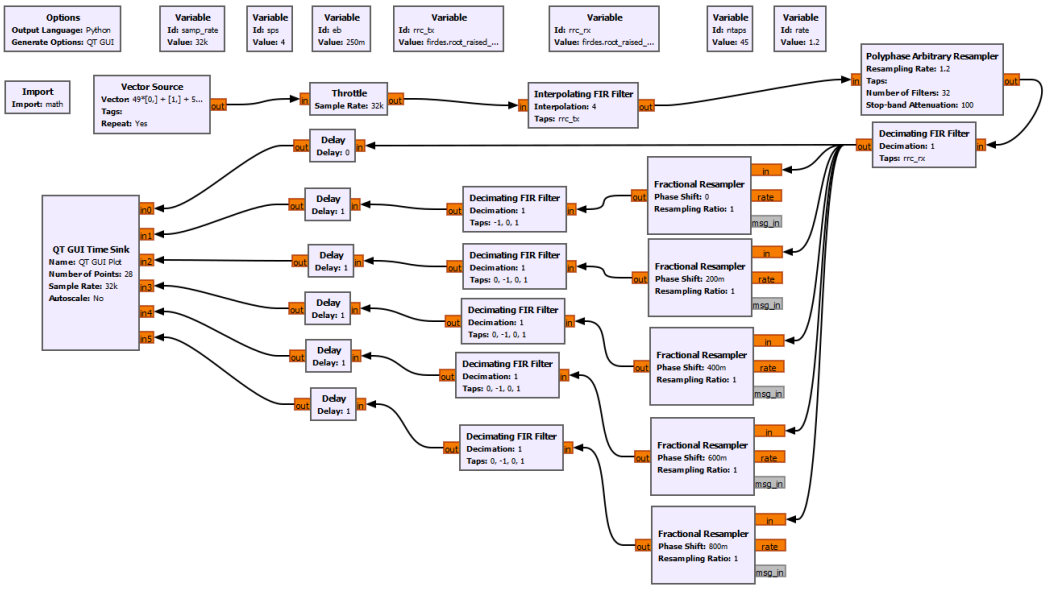


Рис. 14: Flowgraph с несколькими фазами

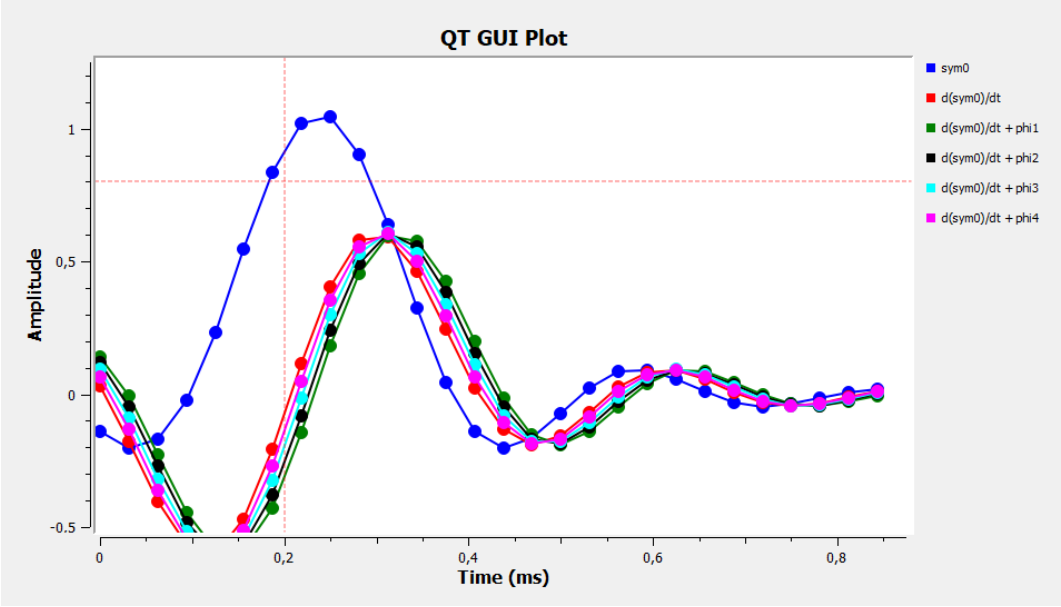


Рис. 15: Несколько фаз

На графике мы можем видеть, что сигнал, помеченный как $d(sym0)/dt + phi3$, имеет точку отсчета в 0. Это говорит нам о том, что наша идеальная точка дискретизации находится при этом фазовом сдвиге. Следовательно, если мы возьмем RRC-фильтр нашего приемника и настроим его фазу на $phi3 = 3 * 2\pi/5$, то мы сможем исправить рассогласование по времени и выбрать идеальную точку дискретизации.

Итак, в данном примере мы использовали 5 фильтров в качестве идеальной точки выборки. Но этого недостаточно. Любое смещение выборки между этими

фазами по-прежнему приведет к несвоевременной выборке с добавленными ISI, как мы исследовали ранее.

Поэтому мы используем больше фильтров (32), чтобы получить максимальный коэффициент шума ISI, который меньше шума квантования 16-битного значения. То есть мы получим точность 16 бит. Для большей точность потребуется больше фильтров.

Затем мы используем контур управления 2-го порядка, который требуется, чтобы получить как правильную фазу фильтра, так и разницу в скорости между двумя тактовыми сигналами.

## 3.4 Использование блока многофазной синхронизации

Теперь мы используем этот блок в нашей симуляции. Пример flowgraph `mpsk_stage` принимает выходные данные модели канала и передает их через наш блок.

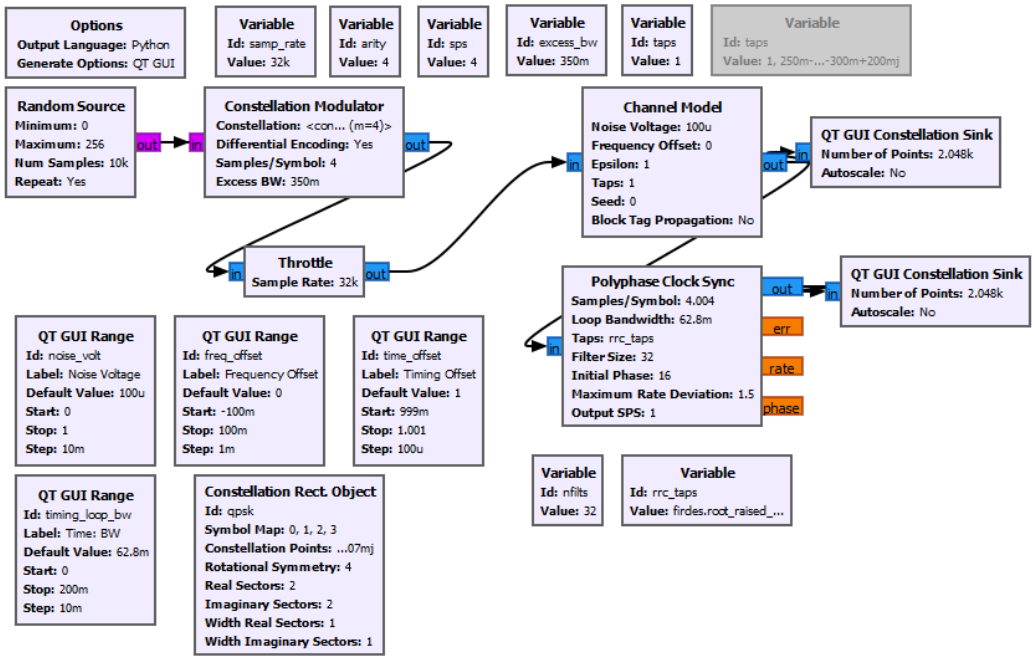


Рис. 16: Flowgraph с блоком многофазной синхронизации

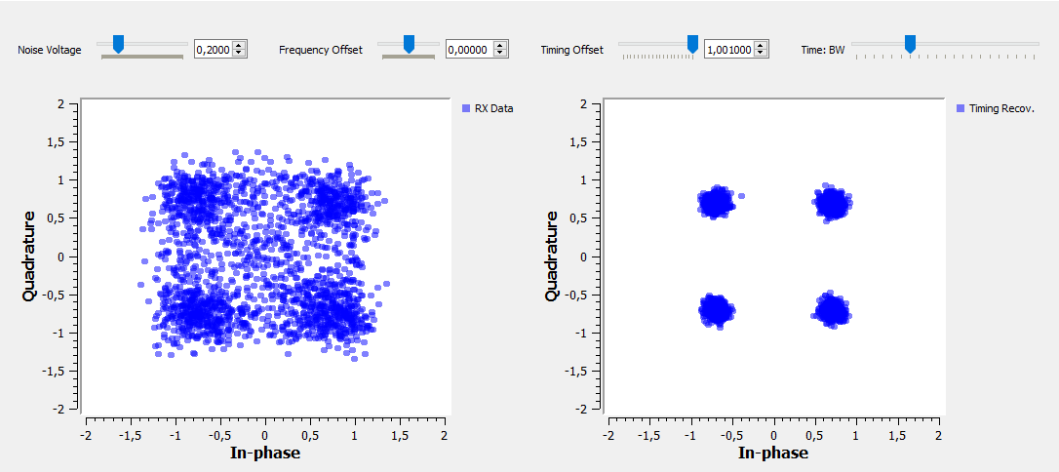


Рис. 17: Использование блока многофазной синхронизации

Видно два созвездия: слева - полученный сигнал до восстановления синхронизации и справа - после восстановления синхронизации.

# 4 Многолучевость

Сначала мы разберемся, что такое многолучевость. В большинстве коммуникационных сред у нас нет единственного пути для прохождения сигнала от передатчика к приемнику. Сигналы отражаются от различных поверхностей и приходят на приёмник в разное время в зависимости от длины пути. Их суммирование в приемнике вызывает искажения. Эти искажения вызывают внутрисимвольную и межсимвольную интерференции.

Нам нужно исправить это поведение, и мы можем сделать это, используя механизм, очень похожий на стереоэквалайзер. С помощью стереофонического эквалайзера мы можем изменить усиление определенных частот, чтобы либо подавить, либо усилить эти сигналы.

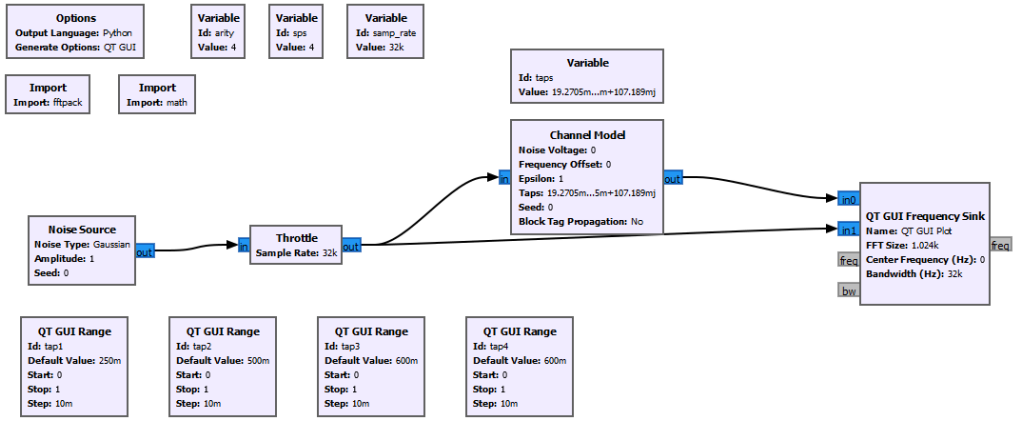


Рис. 18: Flowgraph многолучевость

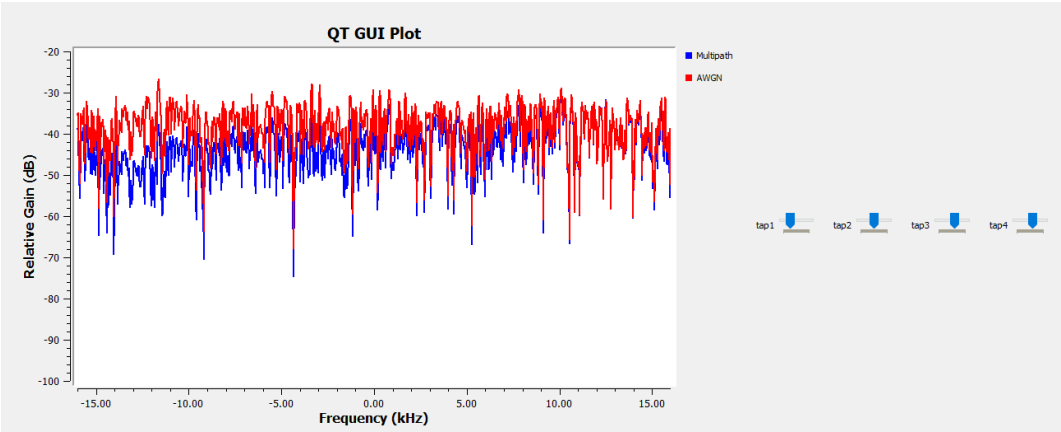


Рис. 19: Использование блока многофазной синхронизации

Как мы можем видеть на графике, многолучевой канал создает некоторые искажения в сигнале. Задача эквалайзера - отменить искажение, вызванное ка-

налом, чтобы выходной сигнал эквалайзера был ровным. Но вместо того, чтобы настраивать каждый tap вручную, мы используем алгоритмы, которые обновляют эти taps за нас. Наша задача - использовать правильный алгоритм эквалайзера и настроить параметры.

Одним из важных параметров здесь является количество taps в эквалайзере. Как мы видим в нашем моделировании, пять taps дают довольно грубый контроль над частотной характеристикой. Чем больше taps, тем больше времени требуется как на вычисление ответвлений, так и на запуск эквалайзера против сигнала.

# 5 Эквалайзеры

GNU Radio имеет два эквалайзера.

## 5.1 CMA

CMA или алгоритм постоянного модуля - это слепой эквалайзер, который работает только с сигналами с постоянной амплитудой или модулем. В примере `mpsk_stage4.grc` мы используем алгоритм CMA с 11 taps. Изменим параметры и посмотрим, как это влияет на производительность как с точки зрения вычислений, так и с точки зрения сигналов.

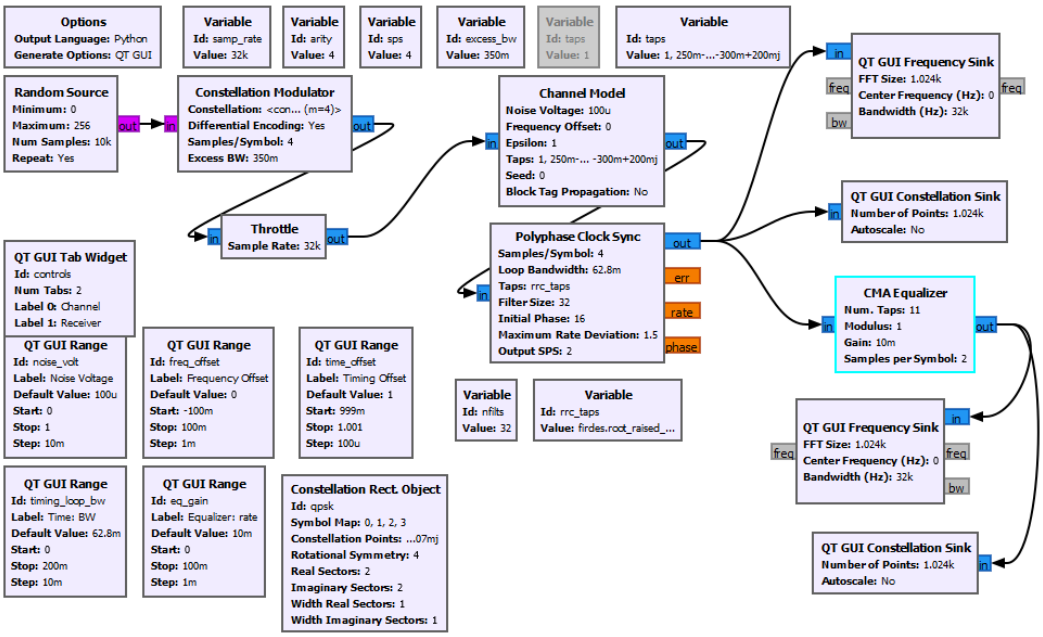


Рис. 20: Flowgraph с эквалайзером CMA

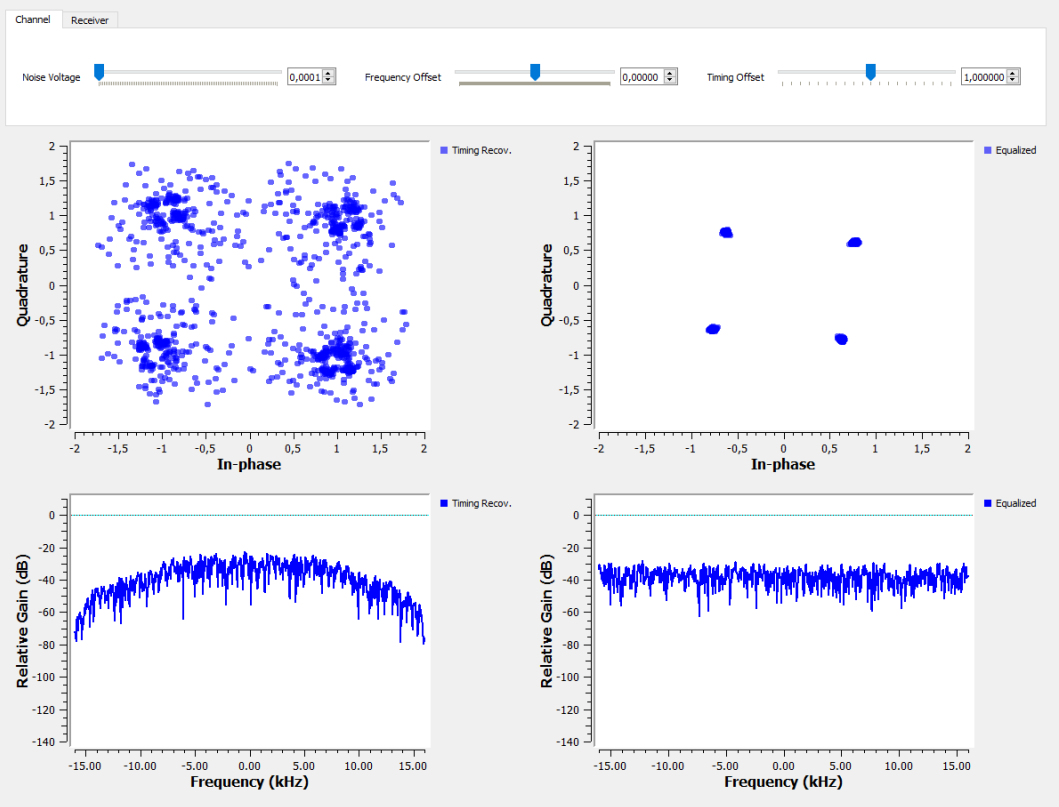


Рис. 21: Использование эквалайзера CMA

На графиках мы можем увидеть эффект синхронизированного по времени многолучевого сигнала до и после эквалайзера. До эквалайзера у нас очень некрасивый сигнал даже без шумов. Эквалайзер понимает, как инвертировать и сократить канал, чтобы у нас снова был хороший, чистый сигнал. Мы также можем видеть сам канал и то, как он красиво выравнивается после эквалайзера.

## 5.2 LMS DD

Хорошей практикой является использование блока эквалайзера LMS DD. LMS или алгоритм наименьших квадратов требует знания принимаемого сигнала. Эквалайзеру необходимо знать точки созвездия для корректировки, и он использует решения о выборках, чтобы сообщить, как обновлять ответвления для эквалайзера.

Заменим эквалайзер CMA на LMS DD.

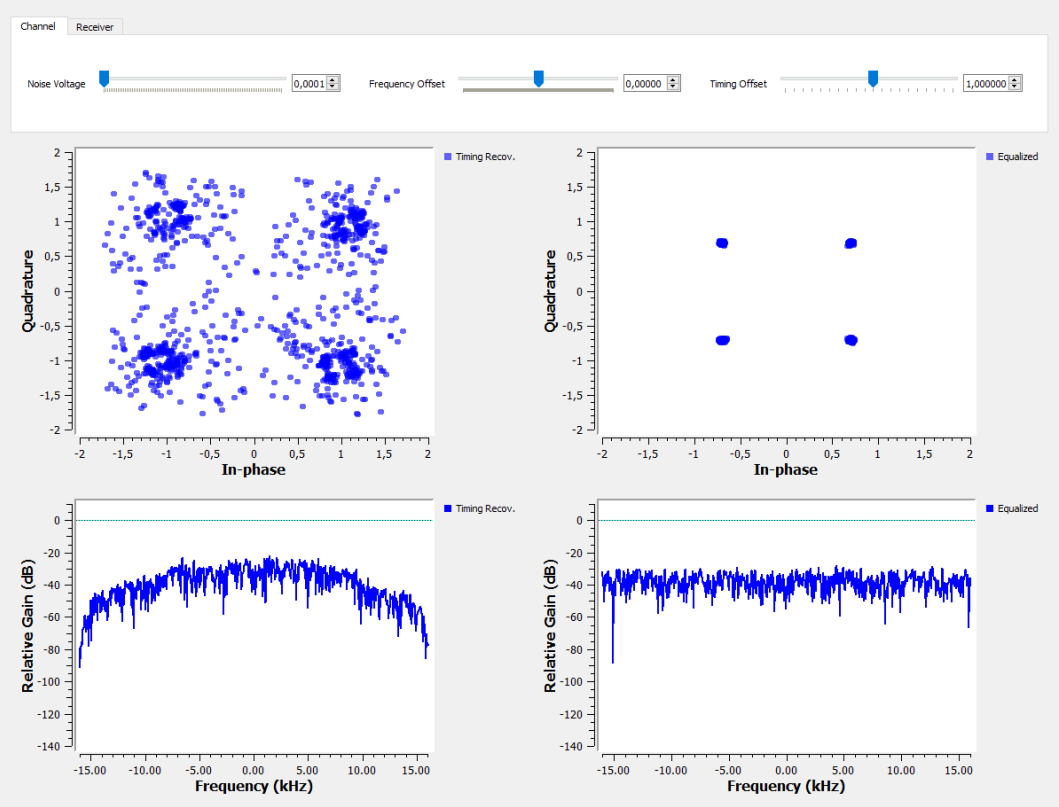


Рис. 22: Использование эквалайзера LMS DD

# 6 Фазовая и точная частотная коррекция

Мы выровняли канал, но у нас все еще есть проблема смещения фазы и частоты. Она выходит за пределами возможностей эквалайзера. Поэтому нам нужно исправить любой сдвиг фазы, а также любой сдвиг частоты.

На этом этапе мы будем использовать цикл второго порядка, чтобы мы могли отслеживать фазу и частоту. Тип восстановления, который мы здесь рассмотрим, предполагает, что мы выполняем точную частотную коррекцию. Поэтому мы должны находиться в приличном диапазоне идеальной частоты, чтобы цикл сошёлся.

Для этой задачи мы собираемся использовать цикл Костаса в примере `mpsk_stage5`.

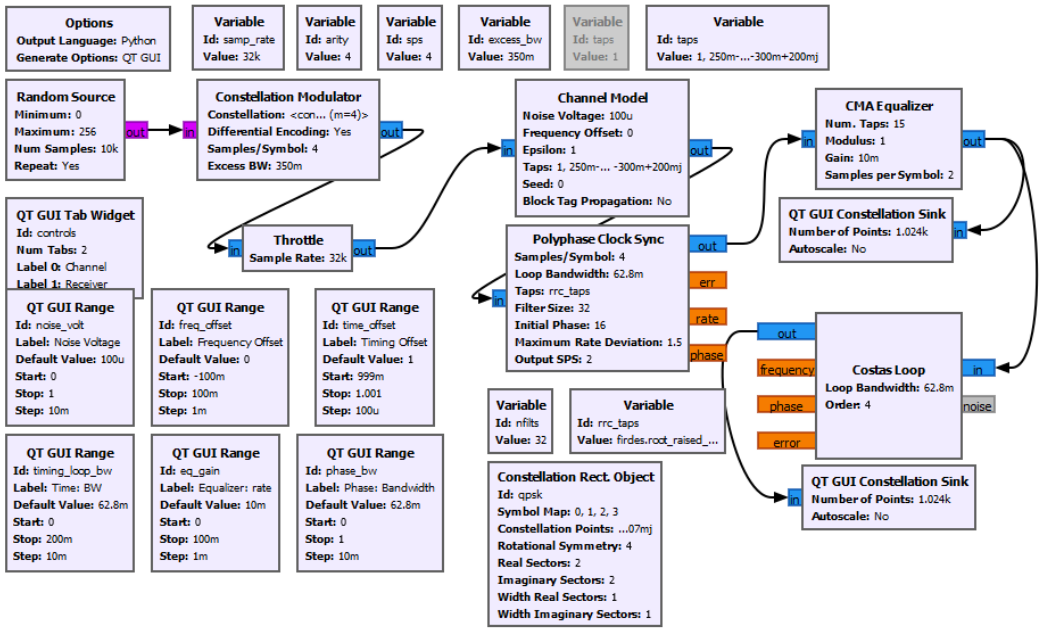


Рис. 23: Flowgraph для исправления сдвига фазы и частоты

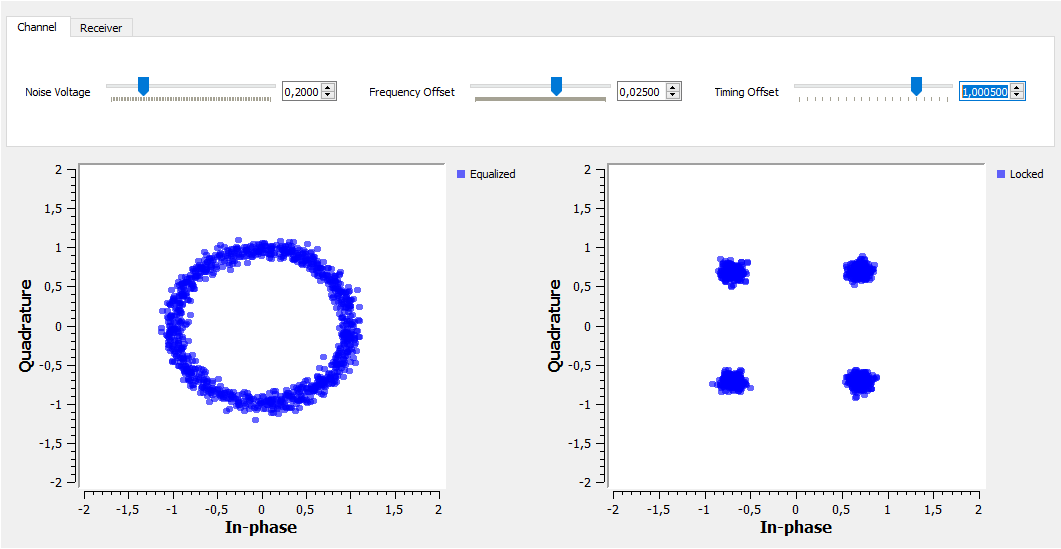


Рис. 24: Исправление сдвига фазы и частоты

Мы установили шум, временной сдвиг, простой многолучевой канал и частотный сдвиг. После эквалайзера мы видим, что все символы находятся на единичном круге, но вращаются из-за смещения частоты, которое еще не исправлено. На выходе блока цикла Костаса мы можем видеть заблокированное созвездие и дополнительный шум.

# 7 Расшифровка

Теперь мы можем декодировать сигнал. Используя пример flowgraph `mpsk_stage6.g` мы вставляем `Constellation Decoder` после цикла Костаса, но наша работа еще не завершена. Мы передаём 4 дифференциальных символа, но мы не можем быть уверены, что у нас есть такое же отображение символов на точки созвездия, которое мы делали при передаче.

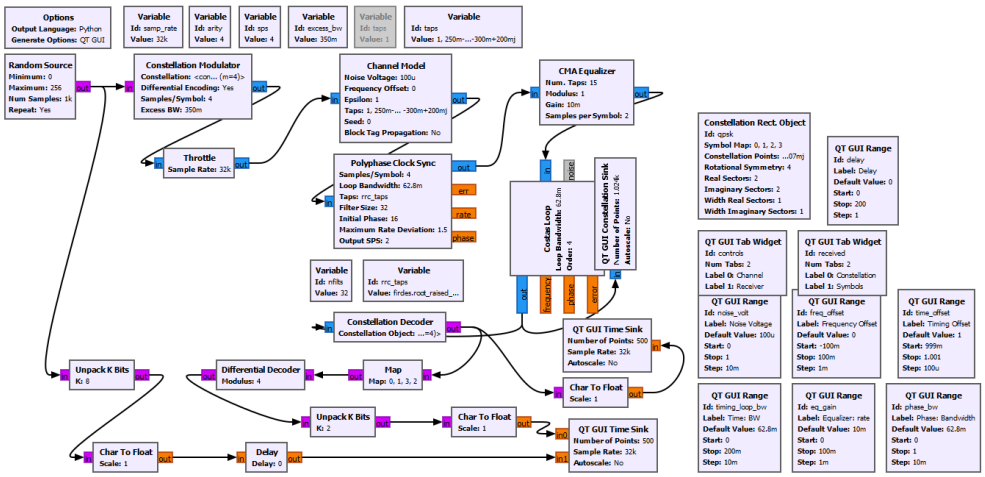


Рис. 25: Flowgraph декодирования

Flowgraph использует блок дифференциального декодера для преобразования кодированных дифференциальным кодом символов обратно в их исходные символы. Мы используем блок Map для преобразования символов из дифференциального декодера в исходные символы, которые мы передали. На данный момент у нас теперь есть исходные символы от 0 до 3, поэтому мы распакуем эти 2 бита на символ в биты, используя блок unpack bits. Теперь у нас есть исходный битовый поток данных.

Но как мы узнаем, что это исходный битовый поток? Для этого нам нужно сравнить переданные данные с входным битовым потоком. Однако прямое сравнение ничего не даст. Так как в цепочке приемника много блоков и фильтров, которые задерживают сигнал, принятый сигнал отстает на некоторое количество бит. Чтобы это исправить, необходимо добавить задержку `Delay`.

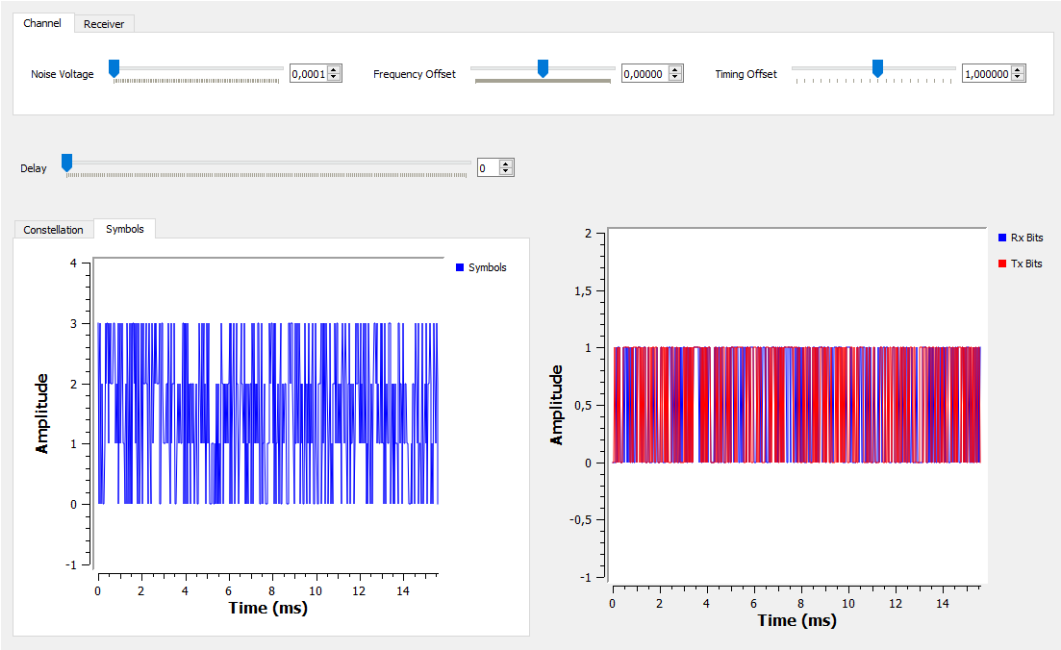


Рис. 26: Сравнение данных

# 8 Выводы

В результате выполнения данной работы мы изучили основные этапы, необходимые для восстановления сигналов, а также научились строить PSK модулятор/демодулятор для работы с сигналом.