

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе №12**  
**Дисциплина:** Телекоммуникационные технологии  
**Тема:** GNU Radio

Работу выполнил:  
Ляшенко В.В.  
Группа: 3530901/80201  
Преподаватель:  
Богач Н.В.

Санкт-Петербург  
2021

# Оглавление

<b>1</b>	<b>Передача сигнала</b>	<b>3</b>
<b>2</b>	<b>Добавление нарушений канала</b>	<b>6</b>
<b>3</b>	<b>Восстановление времени</b>	<b>8</b>
3.1	Проблема ISI . . . . .	8
3.2	Разные часы . . . . .	9
3.3	Блок синхронизации многофазных часов . . . . .	10
3.3.1	Один фильтр . . . . .	10
3.3.2	Несколько фильтров . . . . .	11
3.4	Использование блока многофазной синхронизации . . . . .	13
<b>4</b>	<b>Многолучевость</b>	<b>15</b>
<b>5</b>	<b>Эквалайзеры</b>	<b>17</b>
5.1	Эквалайзер CMA . . . . .	17
5.2	Эквалайзер LMS DD . . . . .	18
<b>6</b>	<b>Фазовая и точная частотная коррекция</b>	<b>20</b>
<b>7</b>	<b>Расшифровка</b>	<b>22</b>
<b>8</b>	<b>Выводы</b>	<b>24</b>

# Список иллюстраций

1.1	Flowgraph избыточной пропускной способности . . . . .	3
1.2	Избыточная пропускная способность . . . . .	4
1.3	Flowgraph созвездия QPSK . . . . .	4
1.4	Созвездие QPSK . . . . .	5
2.1	Flowgraph с нарушениями канала . . . . .	6
2.2	Созвездие с нарушениями . . . . .	7
3.1	Flowgraph проблемы ISI . . . . .	8
3.2	Сравнение символов, отфильтрованных RC и RRC . . . . .	9
3.3	Flowgraph разные часы . . . . .	9
3.4	Работа разных часов . . . . .	10
3.5	Flowgraph с использованием блока . . . . .	10
3.6	Нет смещения часов . . . . .	11
3.7	Есть смещение часов . . . . .	11
3.8	Flowgraph с несколькими фазами . . . . .	12
3.9	Несколько фаз . . . . .	12
3.10	Flowgraph с блоком многофазной синхронизации . . . . .	13
3.11	Использование блока многофазной синхронизации . . . . .	13
3.12	Добавление частотного сдвига . . . . .	14
4.1	Flowgraph многолучевость . . . . .	15
4.2	Использование блока многофазной синхронизации . . . . .	15
5.1	Flowgraph с эквалайзером CMA . . . . .	17
5.2	Использование эквалайзера CMA . . . . .	18
5.3	Flowgraph с эквалайзером LMS DD . . . . .	19
5.4	Использование эквалайзера LMS DD . . . . .	19
6.1	Flowgraph для исправления сдвига фазы и частоты . . . . .	20
6.2	Исправление сдвига фазы и частоты . . . . .	21
7.1	Flowgraph декодирования . . . . .	22
7.2	Сравнение данных . . . . .	23

# Глава 1

## Передача сигнала

В данной работе мы изучим, как строится PSK модулятор/демодулятор.

Первым этапом является передача сигнала QPSK. Сначала мы генерируем поток битов и модулируем его на сложное созвездие. Для этого мы используем блок **Constellation Modulator**. Объект созвездия позволяет нам определить, как кодируются символы.

Также нам требуется генератор случайного источника, который предоставляет упакованные байты со значениями 0-255 для модулятора созвездия.

Количество выборок на символ должно быть минимальным для обеспечения желаемой скорости передачи данных. В данной работе мы используем чуть большее число (4) для лучшей визуализации сигнала.

Наконец, мы устанавливаем значение избыточной пропускной способности. Таким образом, мы получаем flowgraph показанный на рис.1.1.

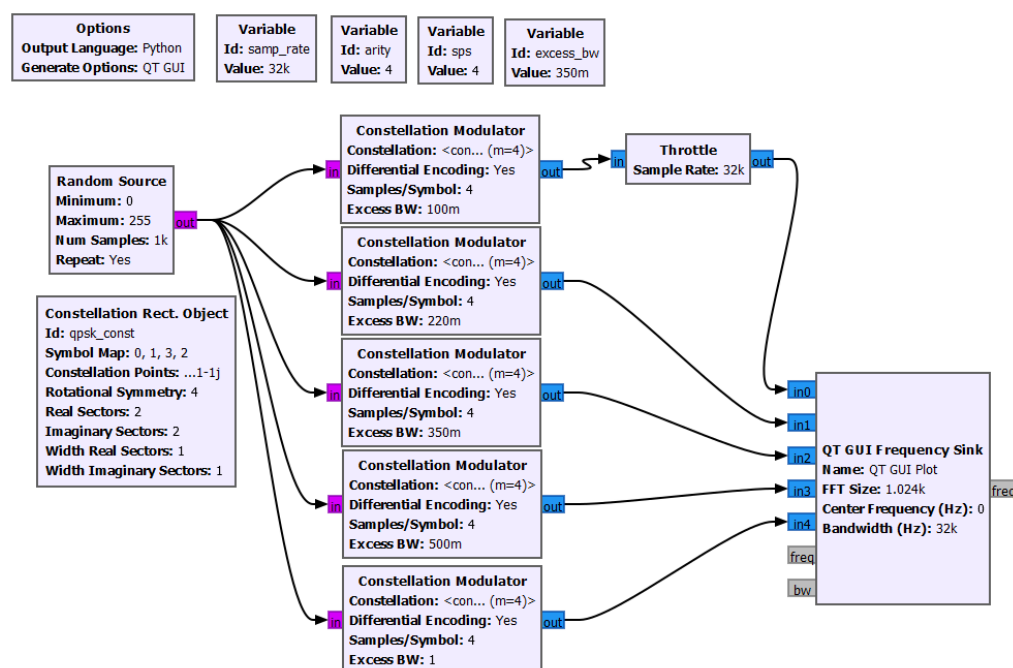


Рис. 1.1: Flowgraph избыточной пропускной способности

При запуске этого flowgraph, мы получаем график, показывающий различные значения избыточной пропускной способности (Рис.1.2). Типичные значения находятся в диапазоне от 0,2 (красная кривая) до 0,35 (зеленая кривая).

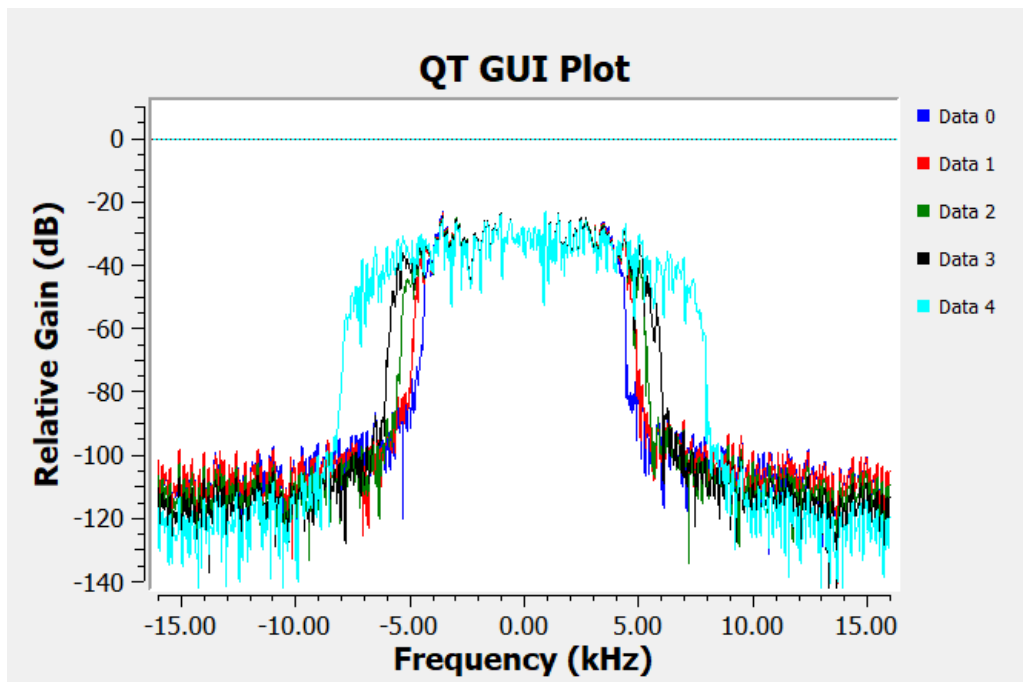


Рис. 1.2: Избыточная пропускная способность

Теперь запустим flowgraph `mpsk_stage1.grc`, передающий созвездие QPSK (Рис.1.3).

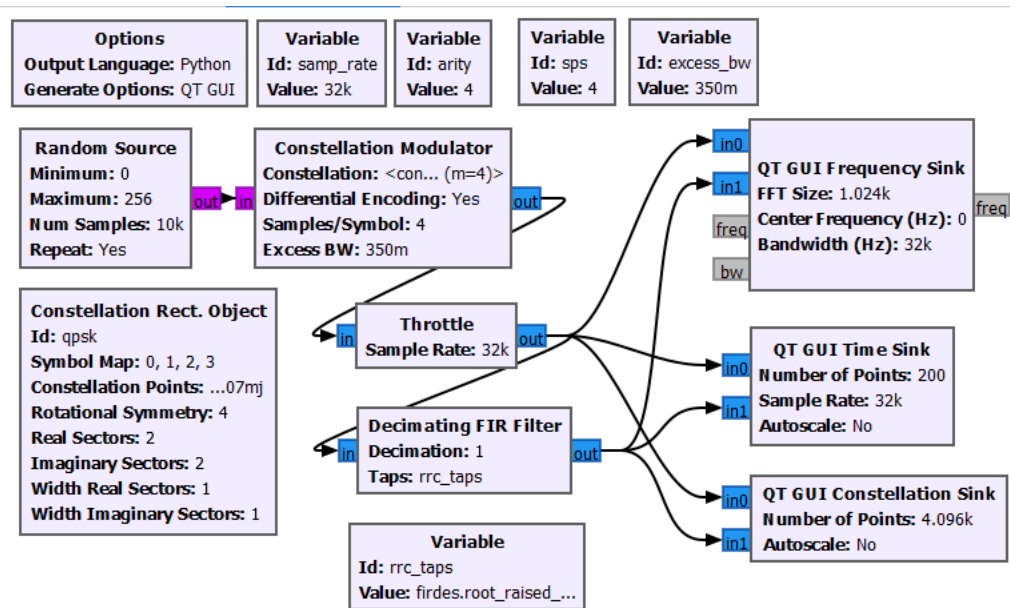


Рис. 1.3: Flowgraph созвездия QPSK

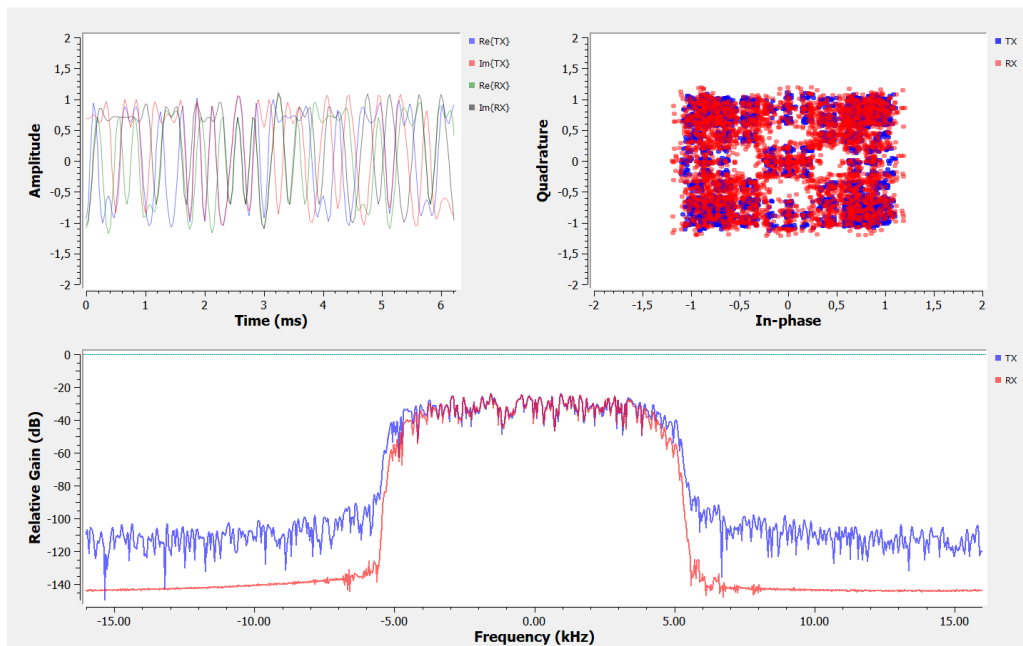


Рис. 1.4: Созвездие QPSK

На рис.1.4 мы видим эффекты повышающей дискретизации и процесс фильтрации.

На передатчике мы используем фильтр RRC, который добавляет межсимвольные помехи (ISI). На приёмнике мы используем другой фильтр RRC для устранения ISI. Таким образом, мы сворачиваем два фильтра RRC вместе и получаем фильтр с приподнятым косинусом.

Фильтрация представляет собой свертку, поэтому выходной сигнал RRC-фильтра на приемной стороне представляет собой сигнал в форме приподнятого косинусоидального импульса с минимизированным ISI.

Получается, что на приёмнике мы используем согласованный фильтр.

Согласованный фильтр — линейный оптимальный фильтр, предназначенный для выделения сигналов известной формы на фоне шумов.

## Глава 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. Часы, определяющие частоту радиомодулей.

Два радиомодуля имеют разные часы. Одно радио работает на частоте  $f_c + f\_delta\_1$ . Другое радио имеет другие часы и его реальная частота равна  $f_c + f\_delta\_2$ . Таким образом, полученный сигнал будет  $f\_delta\_1 + f\_delta\_2$ .

### 3. Идеальная точка выборки.

Идеальная точка выборки связана с проблемой часов. Два радиомодуля работают с разной скоростью, поэтому идеальная точка выборки неизвестна.

В результате моделирования, полученный flowgraph позволяет нам поработать с аддитивным шумом, сдвигом частоты и временным сдвигом (Рис.2.1).

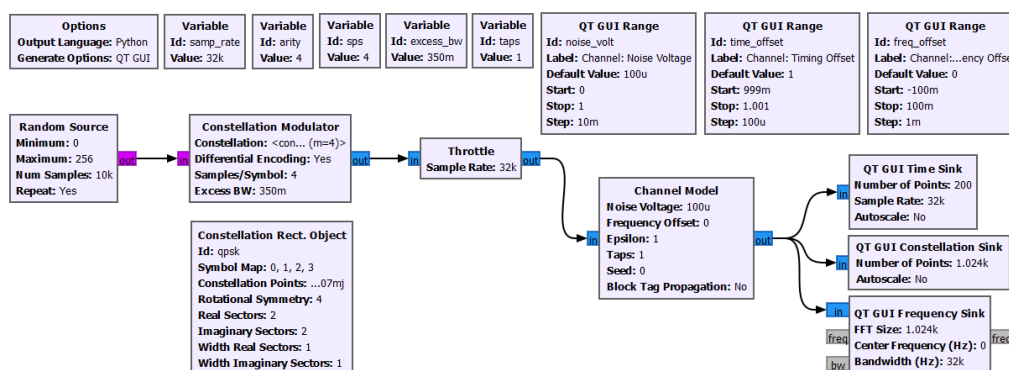


Рис. 2.1: Flowgraph с нарушениями канала

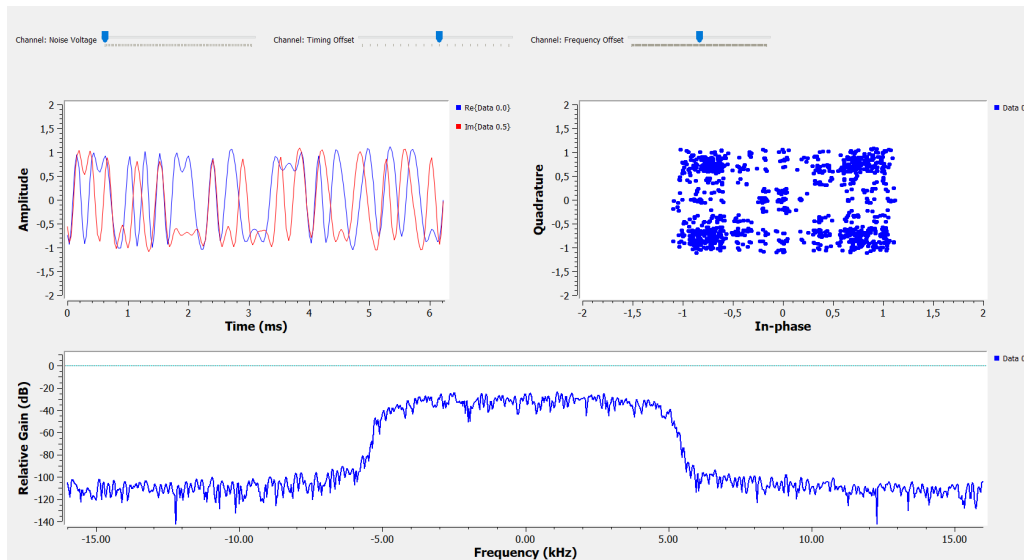


Рис. 2.2: Созвездие с нарушениями

Как мы можем видеть на рис.2.2, график созвездия показывает нам облако образцов хуже, чем на предыдущем этапе. Теперь мы должны отменить все эти эффекты.



## Глава 3

# Восстановление времени

Теперь перейдём к процессу восстановления. Здесь мы будем использовать алгоритм восстановления многофазных часов.

Мы начнем с восстановления времени. 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 Проблема ISI

Проблему ISI мы можем увидеть на примере flowgraph `symbol_sampling.grc`, где мы создаем четыре отдельных символа, а затем фильтруем их (Рис.3.1).

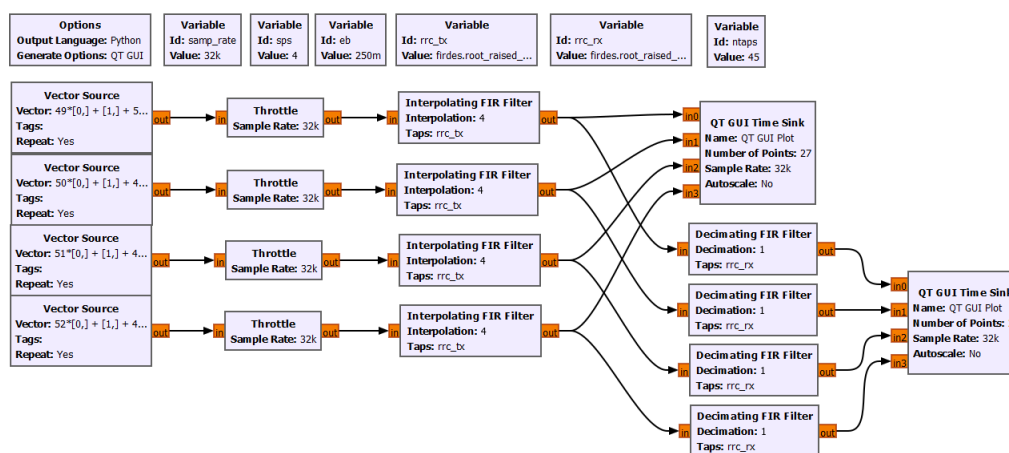


Рис. 3.1: Flowgraph проблемы ISI

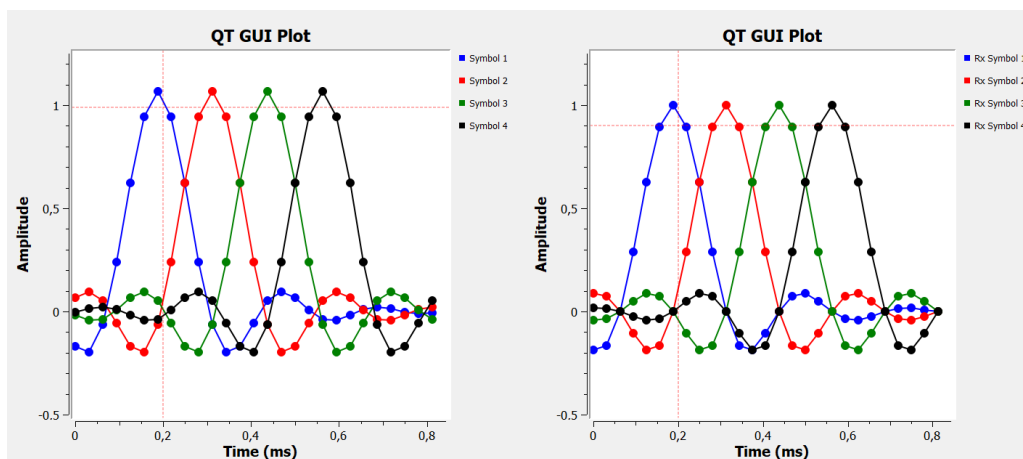


Рис. 3.2: Сравнение символов, отфильтрованных RC и RRC

На рис.3.2 мы видим выходные данные, которые показывают 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 часы

Затем мы посмотрим, как разные часы влияют на точки выборки между передатчиком и приемником. Используя пример потокового графа в `symbol_sampling_diff.grc`, мы моделируем влияние разных часов в передатчике и приемнике (Рис.3.3).

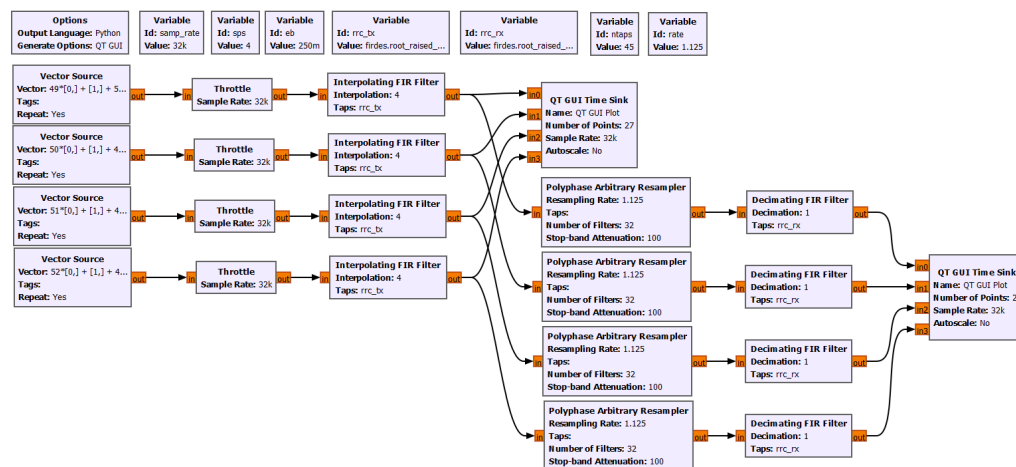


Рис. 3.3: Flowgraph разные часы

Все часы несовершенные, поэтому они (Рис.3.4):

1. начнут свою работу в разные моменты времени;
2. дрейфуют относительно друг друга.

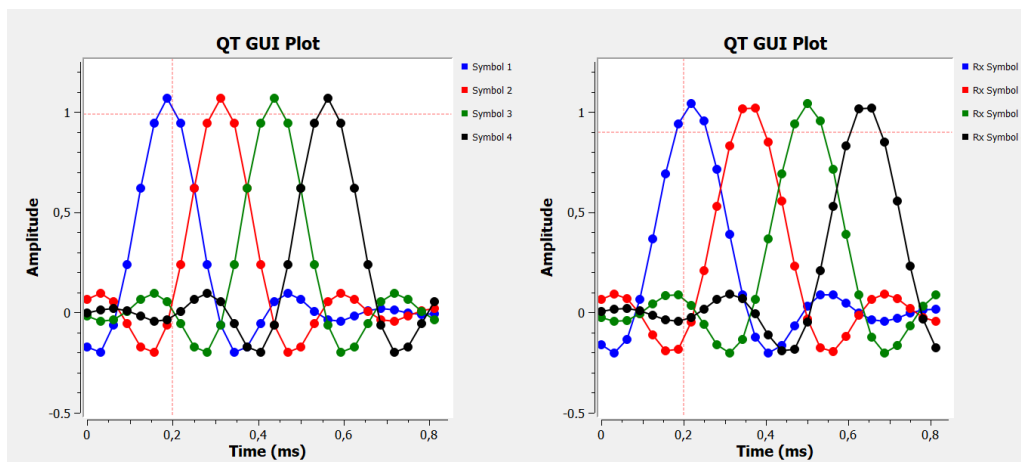


Рис. 3.4: Работа разных часов

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

Изучим как работает блок синхронизации многофазных часов перед его применением.

#### 3.3.1 Один фильтр

Блок вычисляет первый дифференциал входящего сигнала, который будет связан с его смещением тактовой частоты. Используя пример flowgraph `symbol_differential_filter.grc` (Рис.3.5), мы можем увидеть графики с параметром скорости 1 (т.е. нет смещения часов).

Очевидно, что нам нужен образец 0,22 мс. Фильтр разности  $([-1, 0, 1])$  генерирует дифференциал символа, и, как показано на рис.3.6, 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.

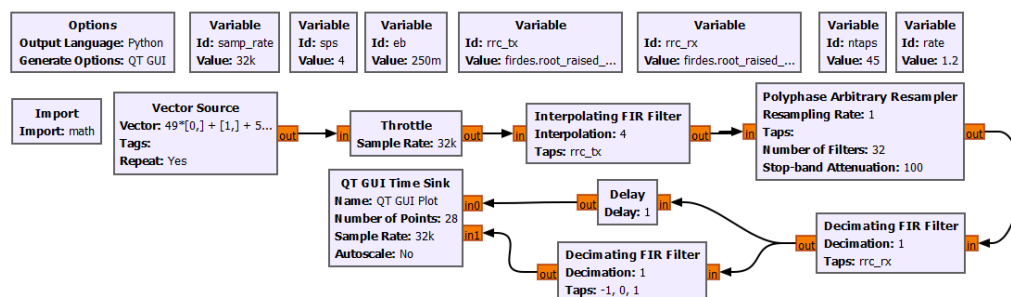


Рис. 3.5: Flowgraph с использованием блока

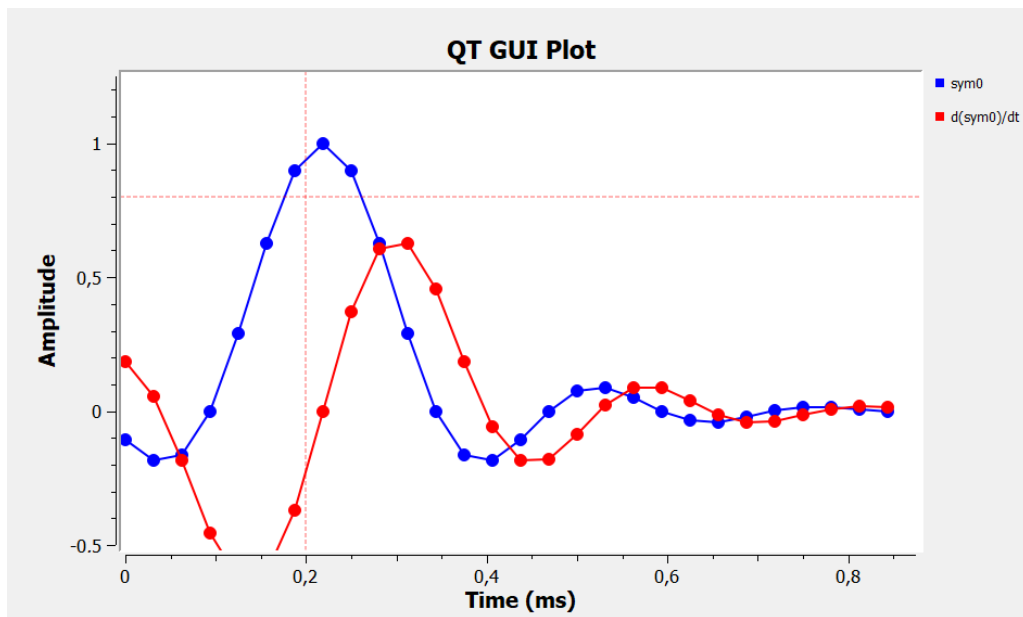


Рис. 3.6: Нет смещения часов

У нас есть временное смещение там, где пик символа выключен, а производный фильтр не показывает нам нулевую точку (Рис.3.7).

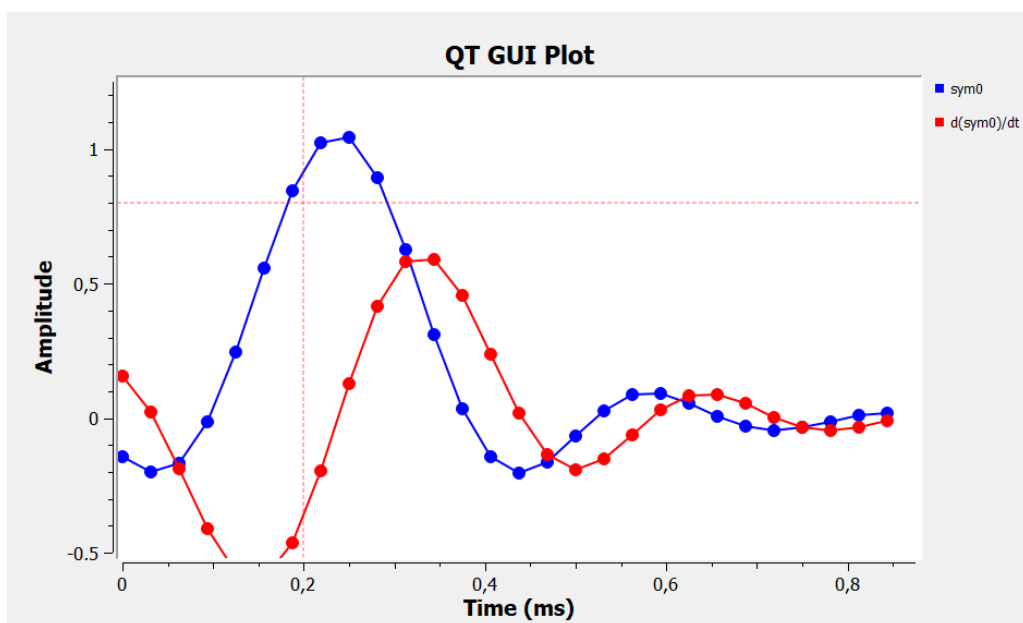


Рис. 3.7: Есть смещение часов

### 3.3.2 Несколько фильтров

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.3.8).

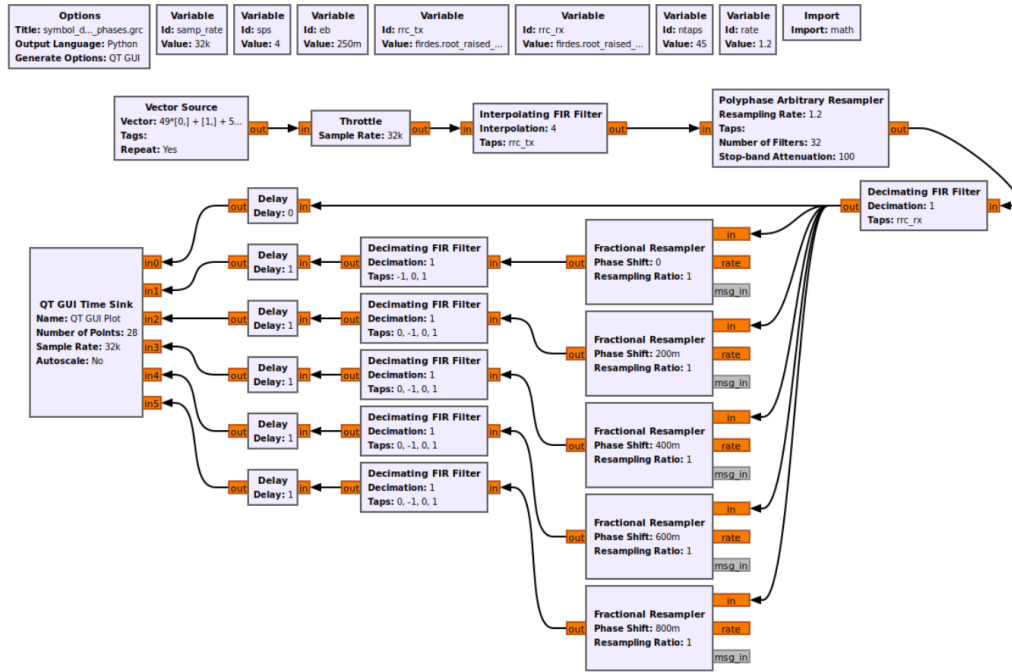


Рис. 3.8: Flowgraph с несколькими фазами

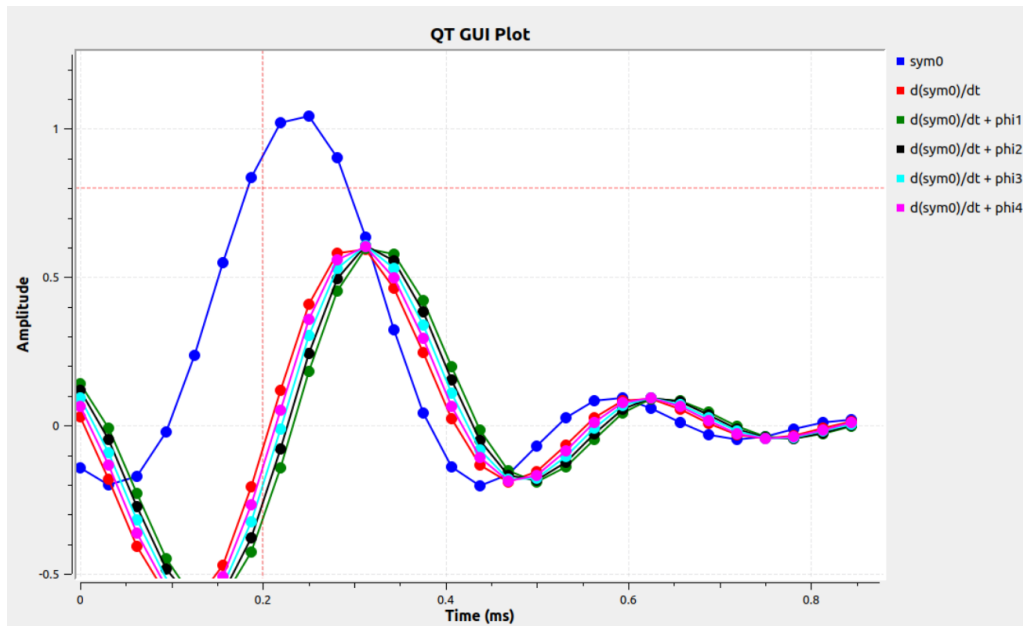


Рис. 3.9: Несколько фаз

На рис.3.9 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 точности 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_stage3.grc` принимает выходные данные модели канала и передает их через наш блок (Рис.3.10).

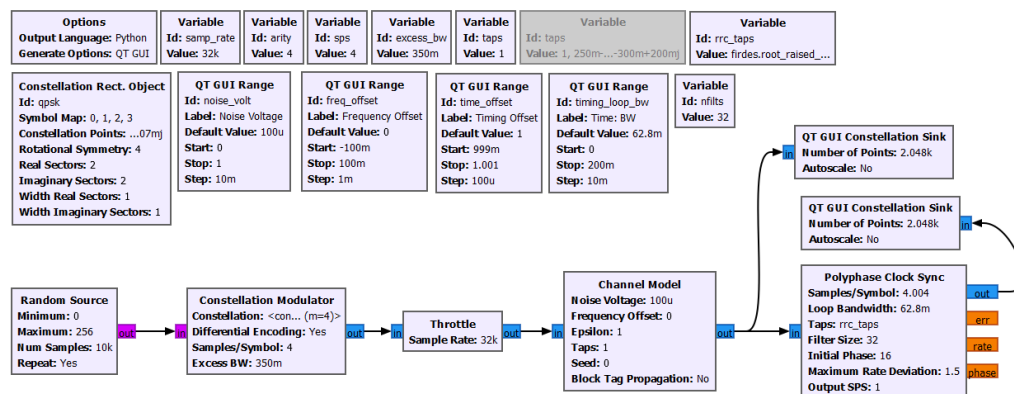


Рис. 3.10: Flowgraph с блоком многофазной синхронизации

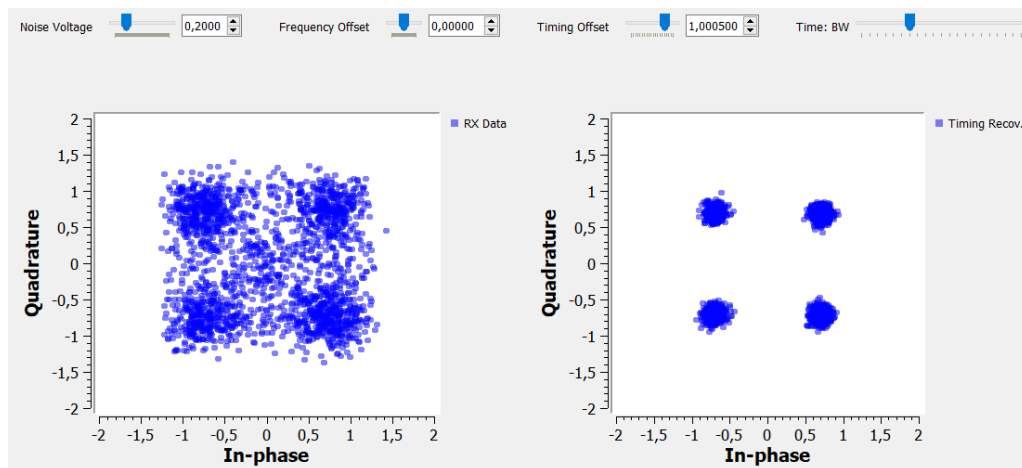


Рис. 3.11: Использование блока многофазной синхронизации

На рис.3.11 мы видим два созвездия: слева - полученный сигнал до восстановления синхронизации и справа - после восстановления синхронизации.

Мы можем изменять разные параметры и наблюдать за изменением созвездия. Например, когда мы добавляем частотный сдвиг, мы видим, что созвездие становится кругом (Рис.3.12).

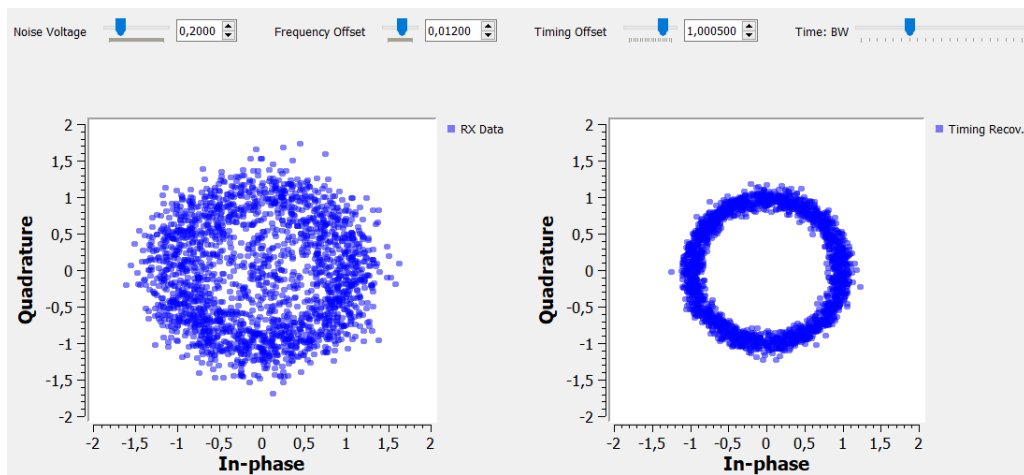


Рис. 3.12: Добавление частотного сдвига

## Глава 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.4.1).

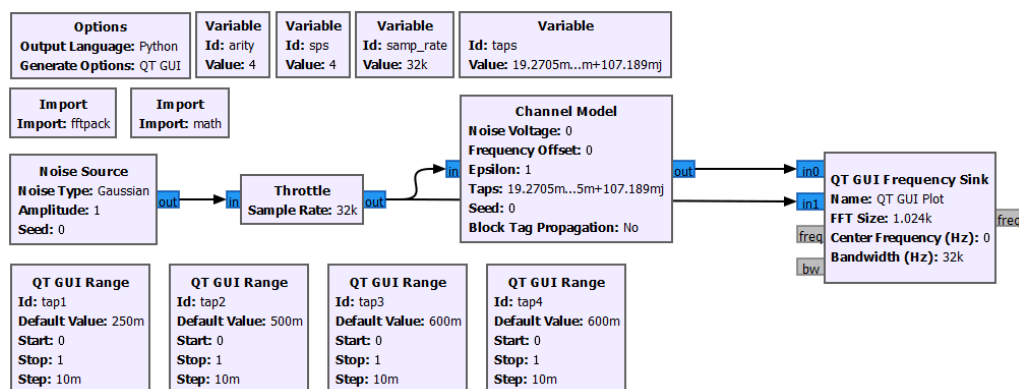


Рис. 4.1: Flowgraph многолучевость

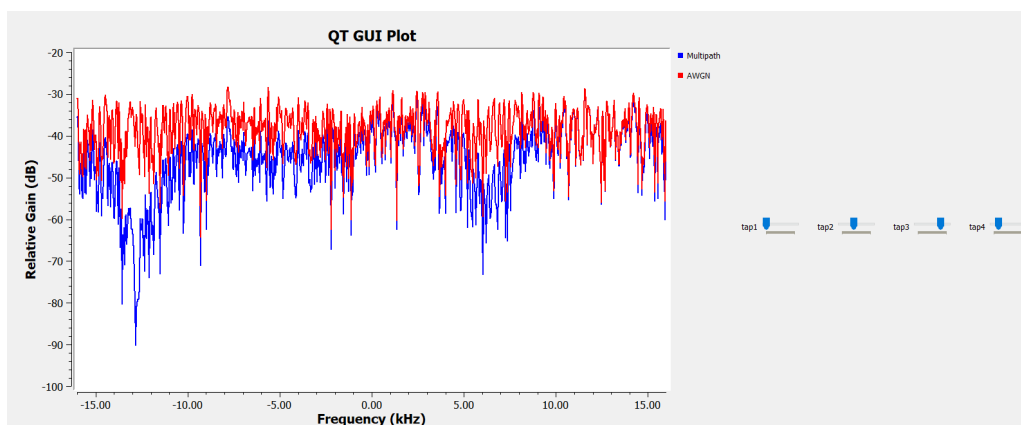


Рис. 4.2: Использование блока многофазной синхронизации



Как мы можем видеть на рис.4.2, многолучевой канал создает некоторые искажения в сигнале. Задача эквалайзера - отменить искажение, вызванное каналом, 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.

# Глава 5

## Эквалайзеры

GNU Radio имеет два эквалайзера.

### 5.1 Эквалайзер СМА

СМА 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 СМА с 11 taps (Рис.5.1). Изменим параметры и посмотрим, как это влияет на производительность как с точки зрения вычислений, так и с точки зрения сигналов.

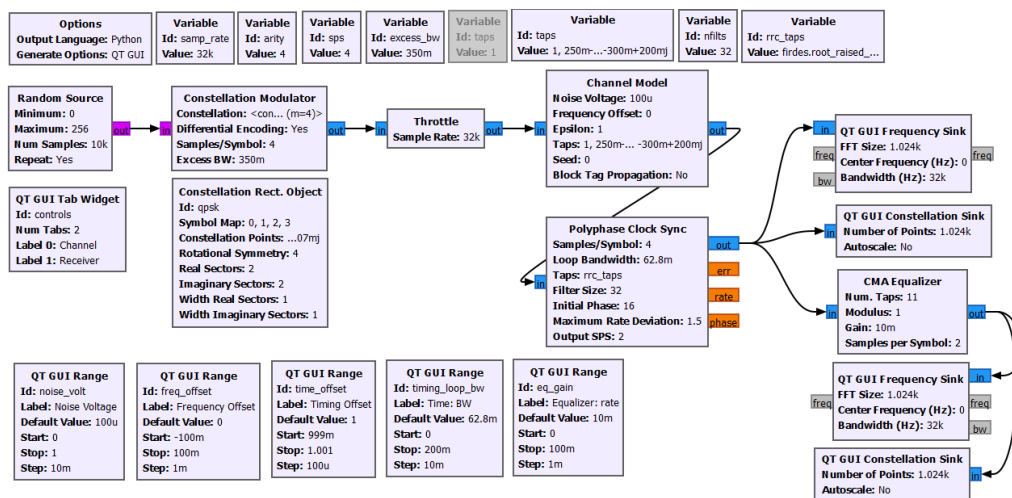


Рис. 5.1: Flowgraph с эквалайзером СМА

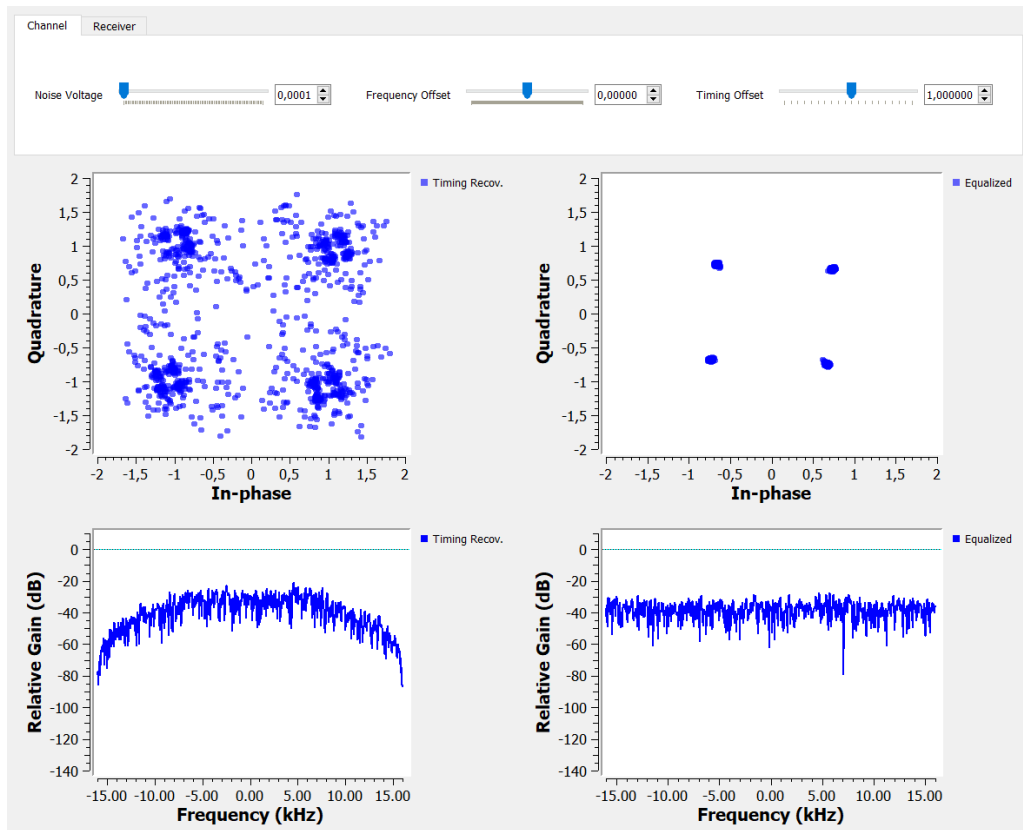


Рис. 5.2: Использование эквалайзера СМА

На рис.5.2 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 Эквалайзер 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 СМА на LMS DD (Рис.5.3).

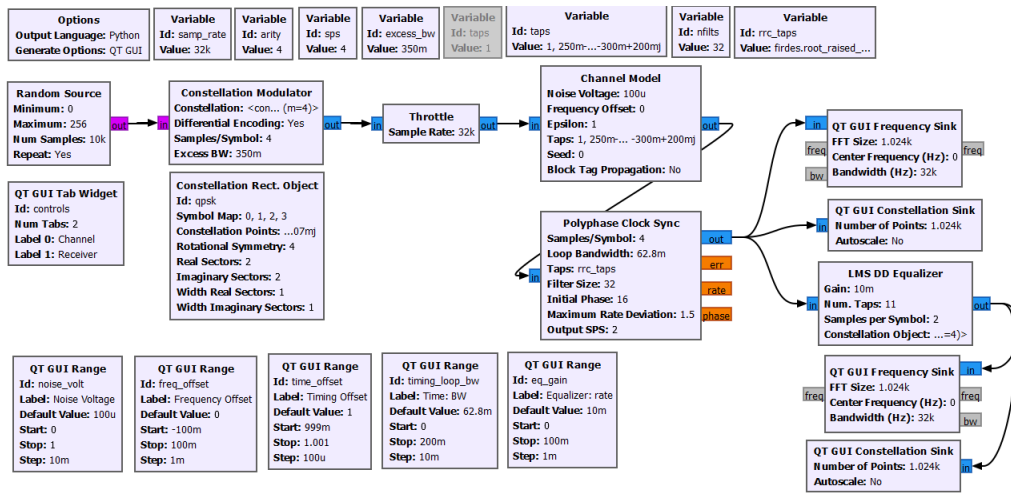


Рис. 5.3: Flowgraph с эквалайзером LMS DD

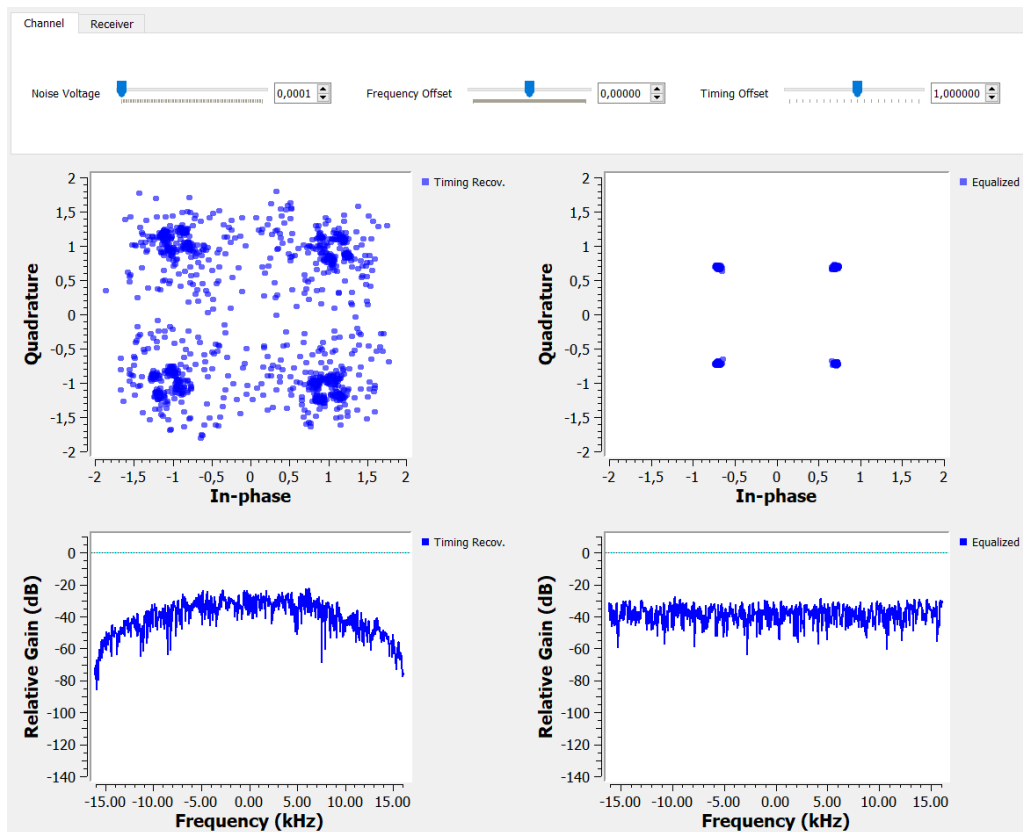


Рис. 5.4: Использование эквалайзера LMS DD

Результат работы этого эквалайзера мы можем видеть на рис.5.4.

## Глава 6

# Фазовая и точная частотная коррекция

Мы выровняли канал, но у нас все еще есть проблема смещения фазы и частоты. Она выходит за пределы 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.grc` (Рис.6.1).

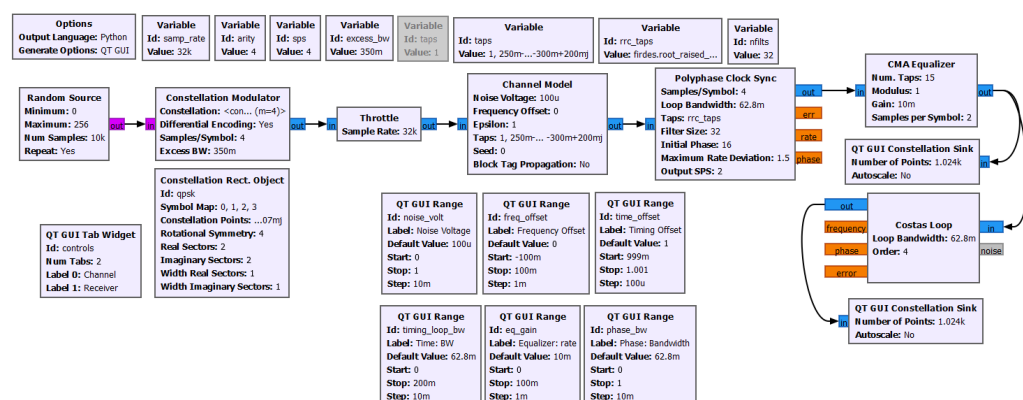


Рис. 6.1: Flowgraph для исправления сдвига фазы и частоты

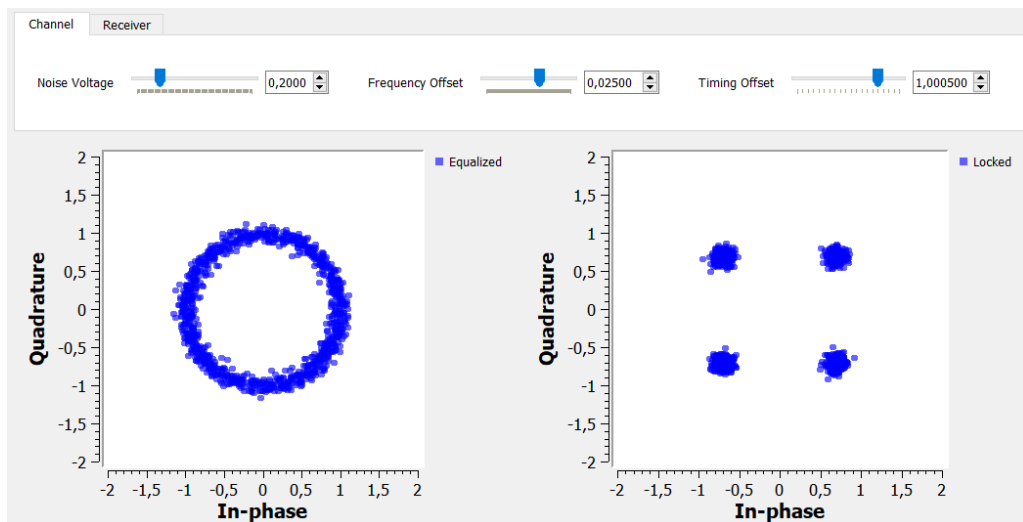


Рис. 6.2: Исправление сдвига фазы и частоты

На рис.6.2 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.

## Глава 7

# Расшифровка

Теперь мы можем декодировать сигнал. Используя пример flowgraph `mpsk_stage6.grc` (Рис.7.1), 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.

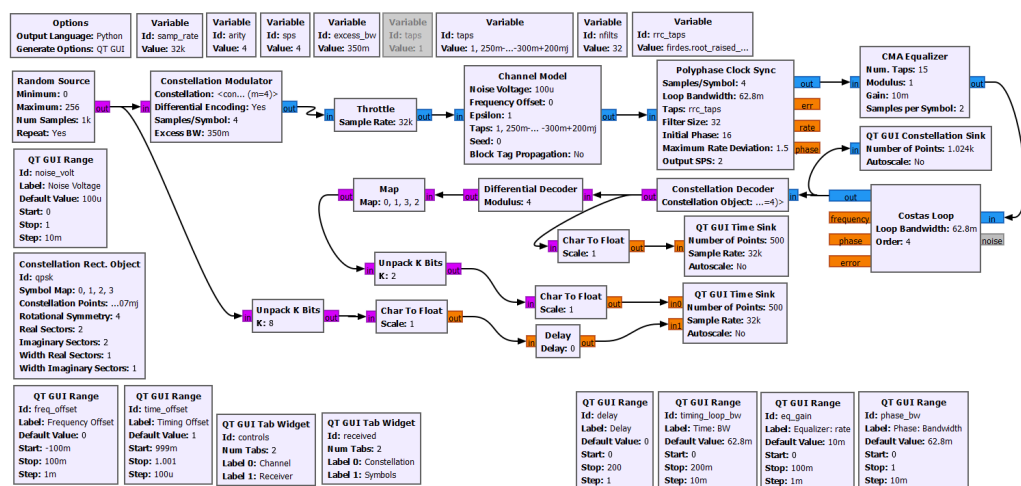


Рис. 7.1: Flowgraph декодирования

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.7.2).

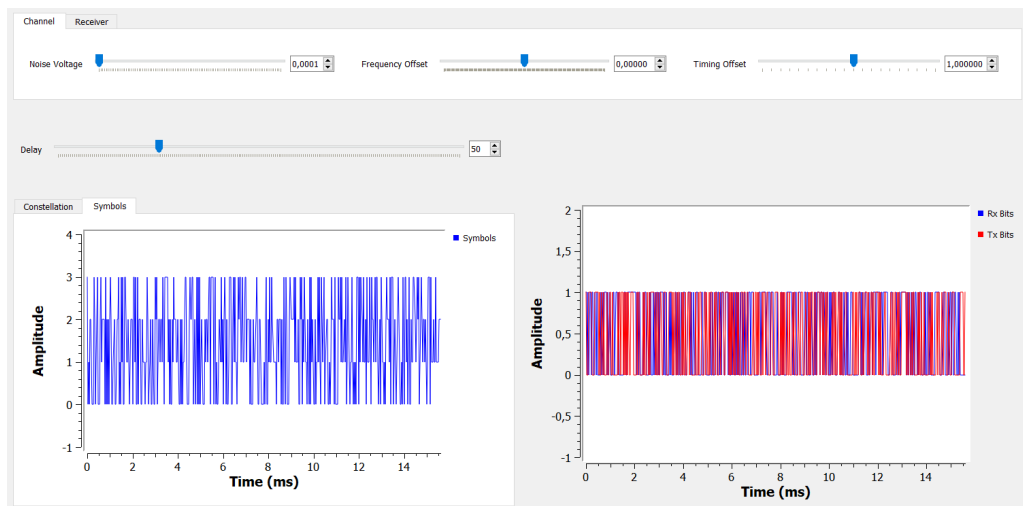


Рис. 7.2: Сравнение данных



# Глава 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.