Лабораторная работа №12
GNU Radio

Смирнов Никита

31 мая 2021 г.

Оглавление

Список иллюстраций

Глава 1

Передача сигнала

На данном этапе нашей задачей является передача сигнала QPSK(Quadrature Phase-Shift).Мы генерируем поток битов и моделируем его на сложное созвездие.

Рассмотрим получившуюся схему из модулей и график ниже, который визуализирует генерацию различных значений избыточной пропускной способности.

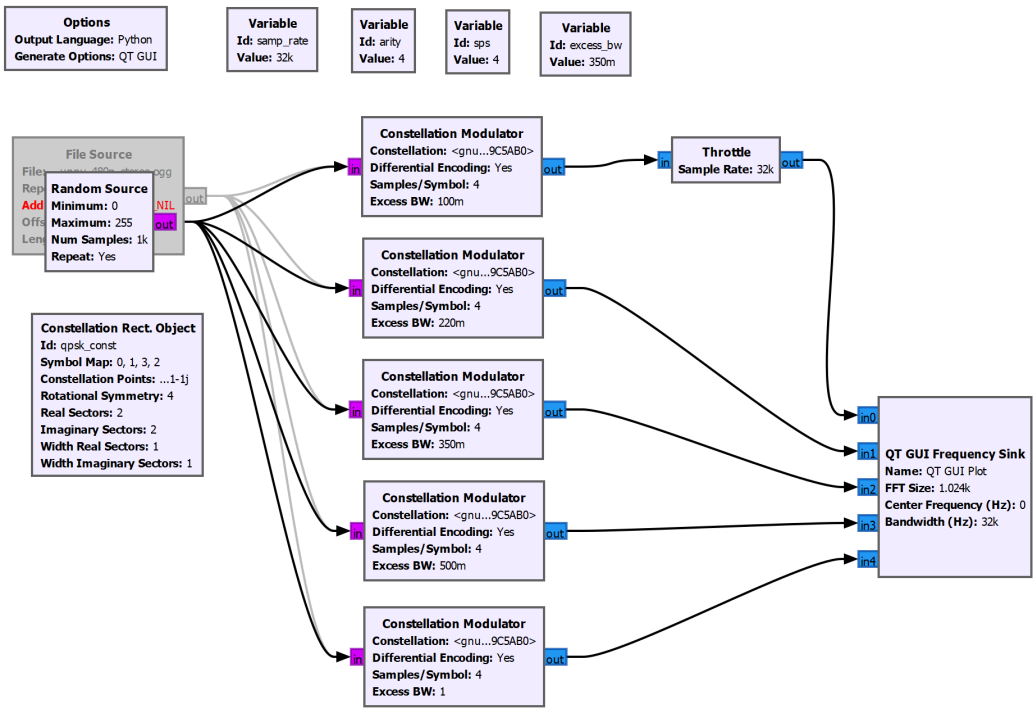

Рис. 1.1: mpsk_rrc_rolloff схема

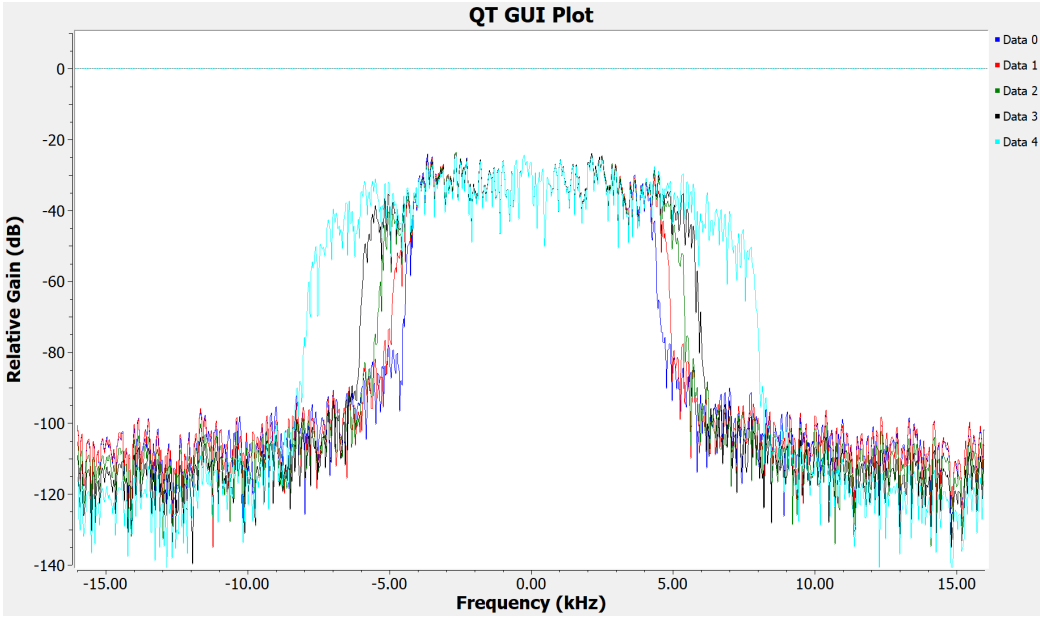

Рис. 1.2: Визуализация генерации данных

Теперь рассмотрим пример потокового графа, который передает созвездие `QPSK`. Он отображает как передаваемый сигнал, так и часть цепи приемника во времени, частоте и диаграмме созвездия.

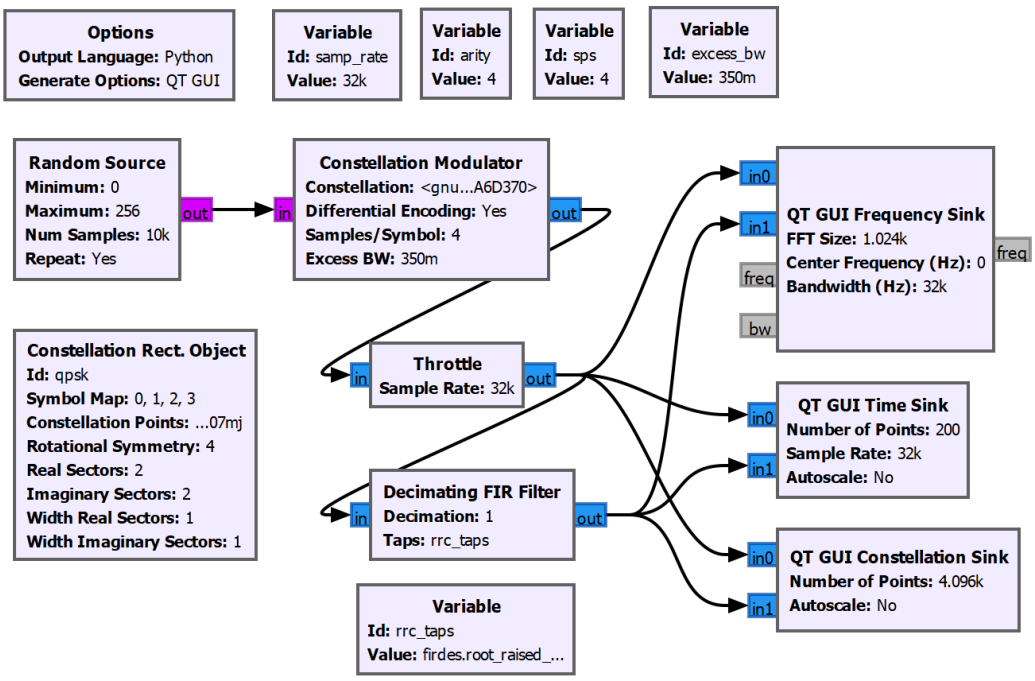

Рис. 1.3: mpsk_stage1 схема

На графике созвездия мы видим эффекты повышающей дискретизации и процесса фильтрации. Благодаря уменьшению внеполосных излучений наш сигнал теперь остается в пределах полосы пропускания нашего канала.

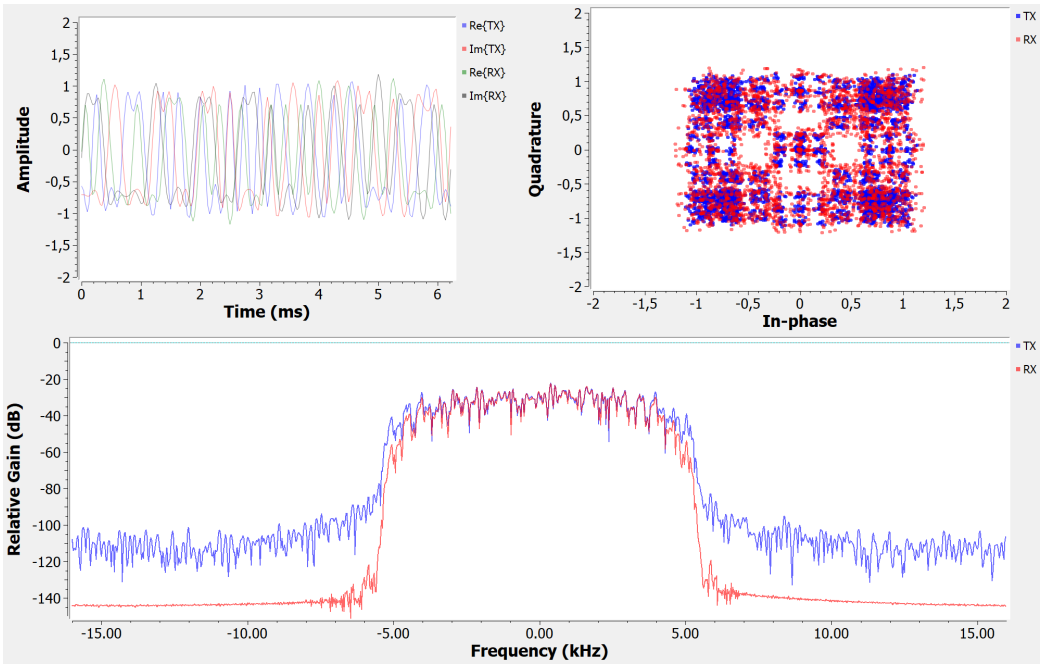

Рис. 1.4: Визуализация генерации данных для схемы mpsk_stage1

Глава 2

Добавление искажений канала

Теперь на данном этапе рассмотрим влияние канала, а также искажение сигнала во время передачи, и когда мы видим его в приёмнике. Для этого мы будем использовать блок `Channel Model`, который позволит смоделировать проблему появления шума, а также различие clock'ов, которые определяют частоту радиомодулей. Благодаря этому мы можем поиграть с эффектами шума, сдвига частоты и временного сдвига. Для следующего графика выставлены следующие параметры: шум - 0,2, смещение частоты - 0,025, смещение времени - 1,0005.

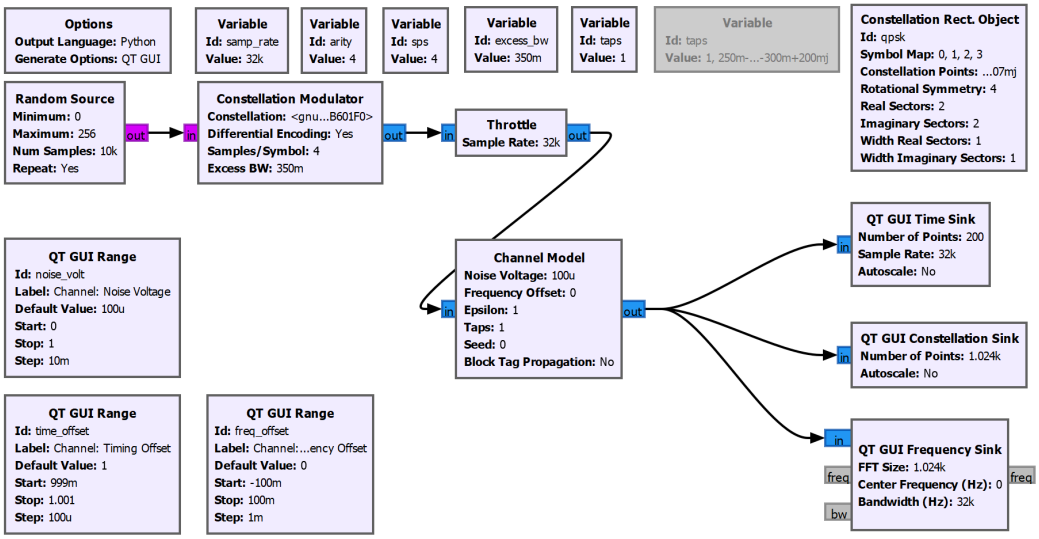

Рис. 2.1: mpsk_stage2 схема

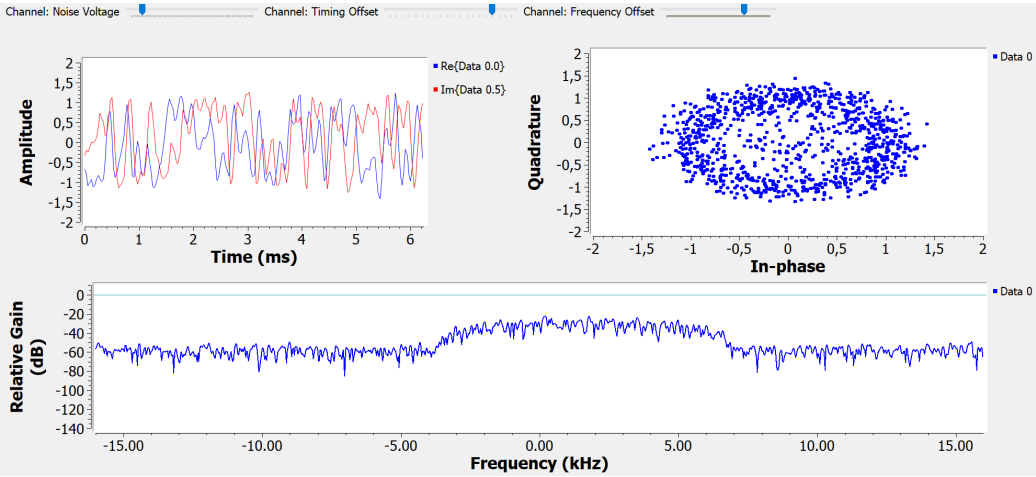

Рис. 2.2: Визуализация генерации данных

В итоге мы получили график созвездия, выглядящий гораздо хуже, чем тот, что был на первом этапе.

Глава 3

Временное восстановление

3.1 Алгоритм восстановления многофазных clock'ов

Теперь рассмотрим процесс восстановления и начнем с временного восстановления. 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`). Для того, чтобы проиллюстрировать проблему `ISI` на примере потокового графа, просто создадим четыре отдельных символа из единиц в потоке, а затем отфильтруем их.

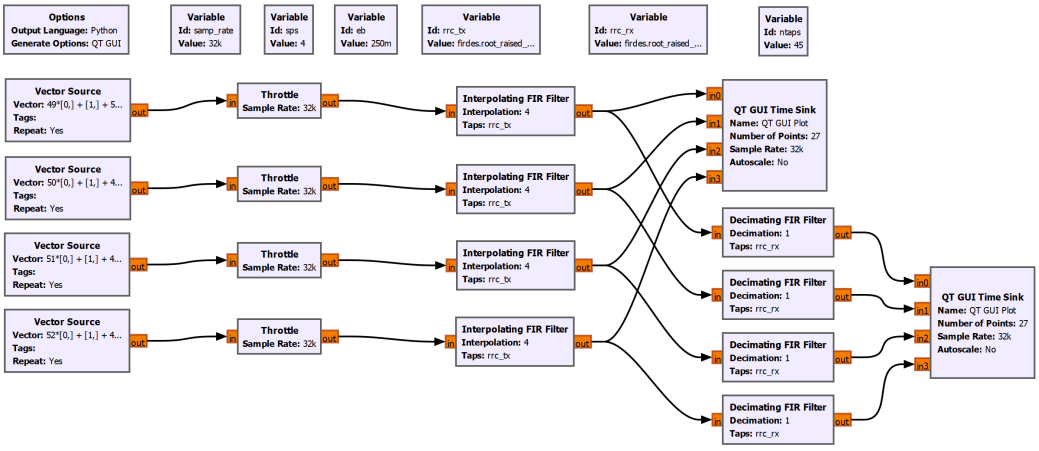

Рис. 3.1: symbol_sampling схема

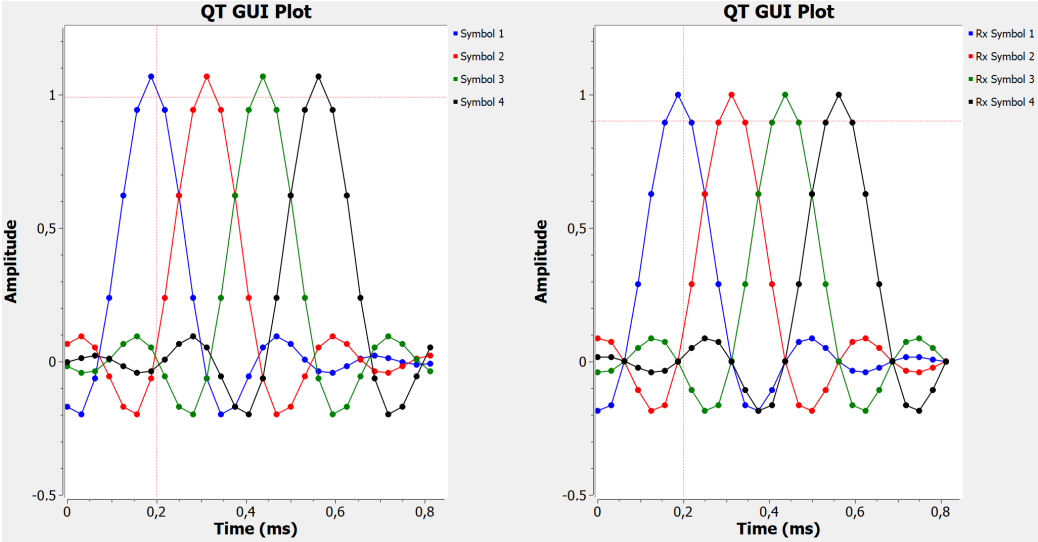

Рис. 3.2: Визуализация потокового графа

Это моделирование позволяет нам легко настраивать такие вещи, как количество выборок на символ, избыточную полосу пропускания фильтров RRC и количество ответвлений. После этого мы можем поиграть с этими различными значениями, чтобы увидеть, как они влияют на поведение точки выборки.

Теперь рассмотрим влияния разных clock'ов на точки выборки между передатчиком и приёмником. Для модуляции этого, мы добавлям `Resampler`, который немного регулирует время выборки символа между переданным сигналом и приёмником.

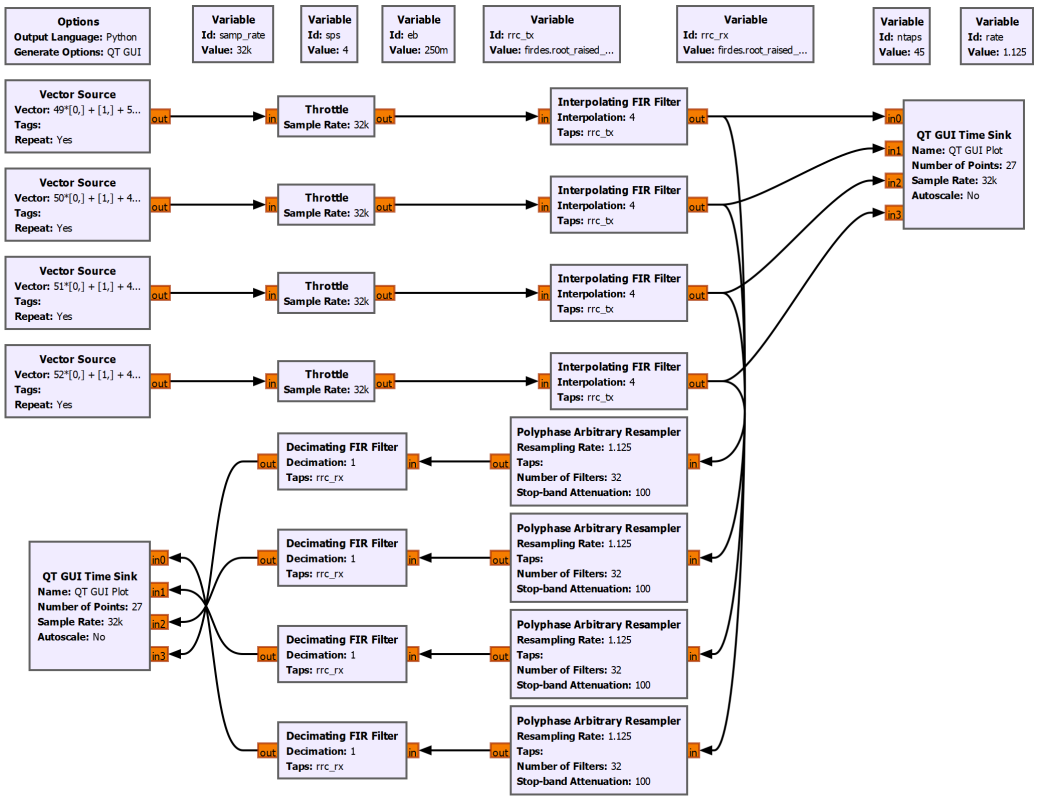

Рис. 3.3: symbol_sampling_diff схема

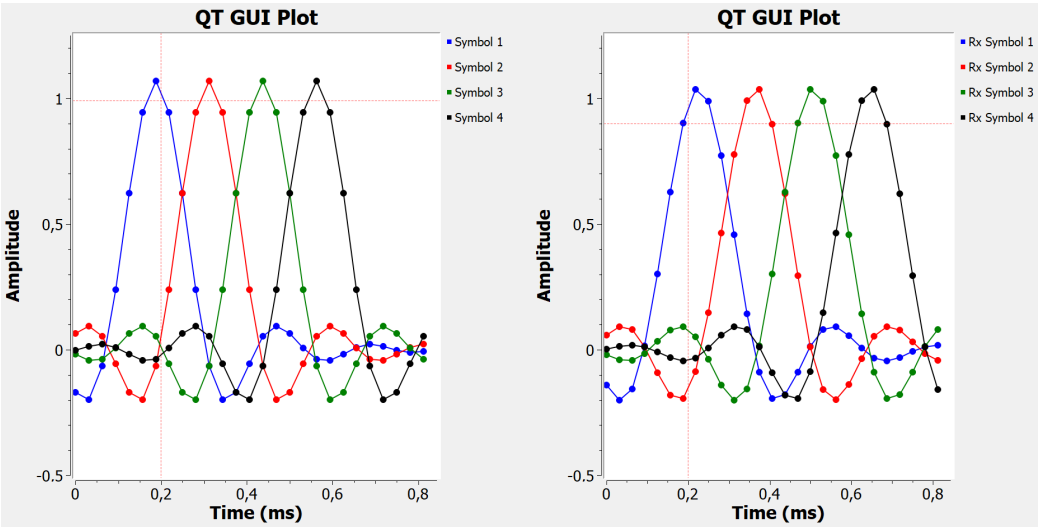

Рис. 3.4: Визуализация потокового графа

3.2 Подробная информация о блоке синхронизации многофазных clock'ов

Блок синхронизации многофазных clock'ов, во-первых, выполняет восстановление clock'ов. Во-вторых, он выполняет согласованный фильтр приёмника для устранения проблемы `ISI`. В-третьих, он выполняет понижающую дискретизацию сигнала и производит выборки со скоростью 1 sps.

Блок работает, вычисляя первый дифференциал входящего сигнала, который будет связан с его смещением тактовой частоты. 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, что когда выход дифференциального фильтра равен 0, то мы нашли оптимальную точку выборки.

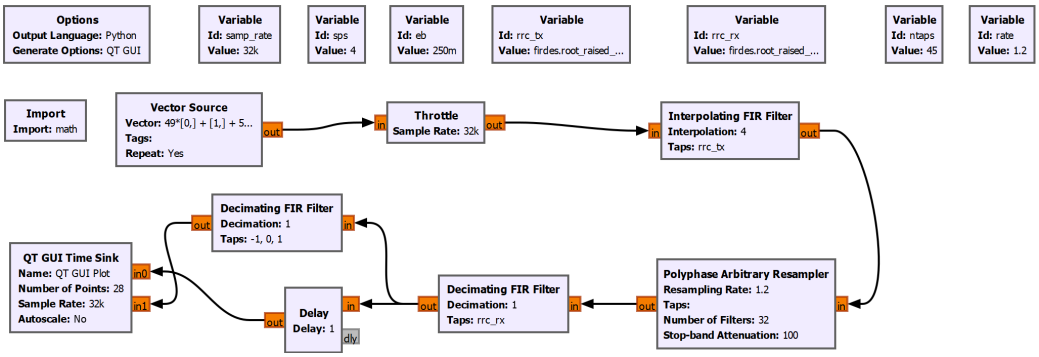

Рис. 3.5: symbol_differential_filter схема

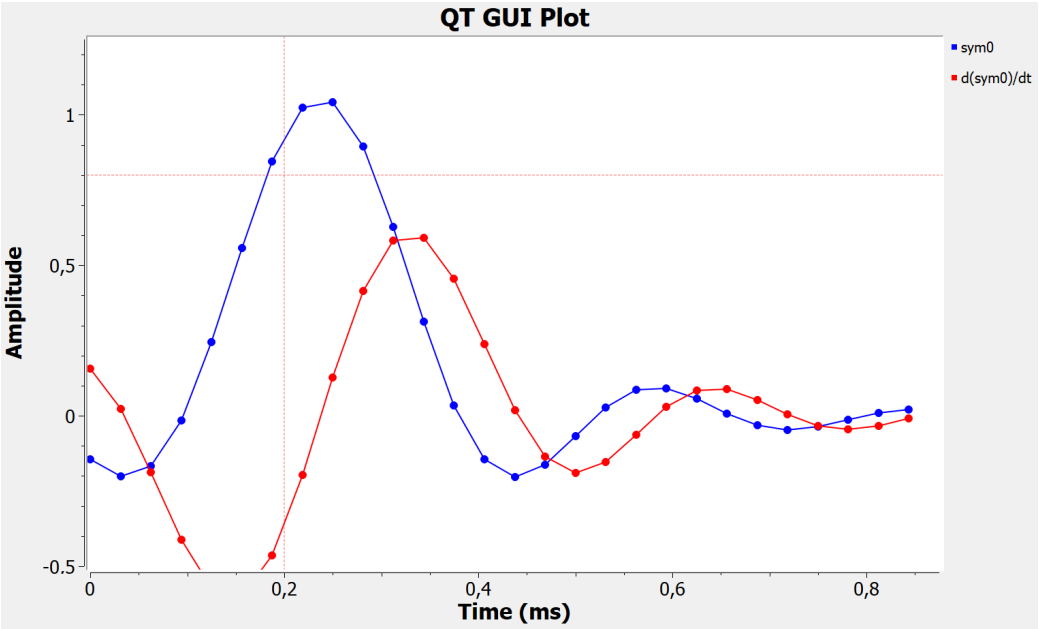

Рис. 3.6: Визуализация потокового графа

Но что происходит, когда у нас есть тактовое смещение? В пиковой точке дифференциальный фильтр уже не будет давать 0, как показано ниже.

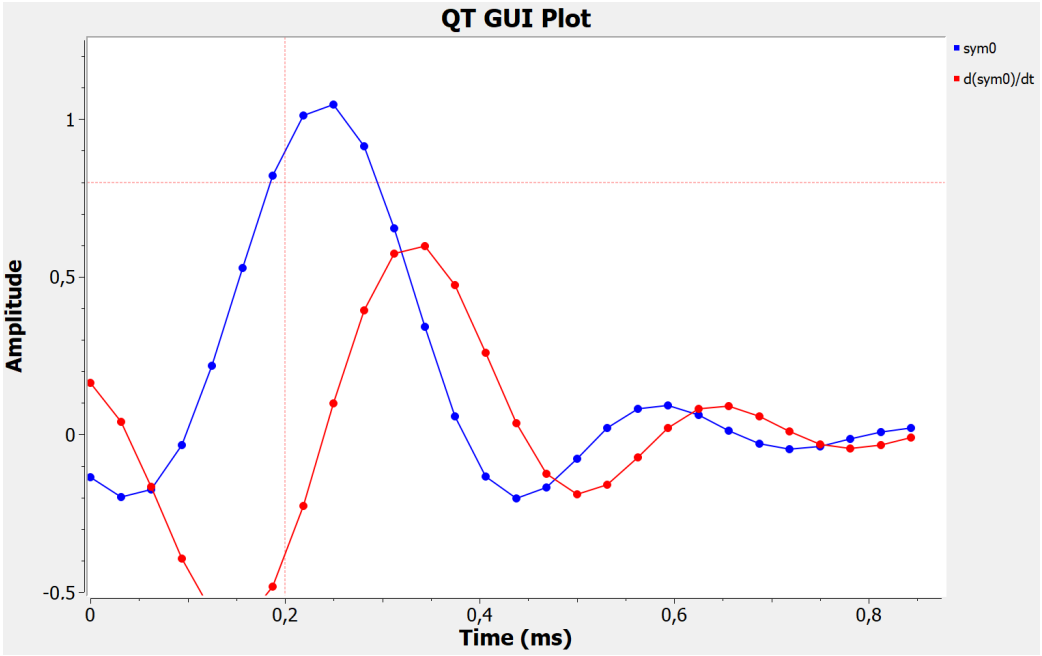

Рис. 3.7: Визуализация потокового графа со смещением

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

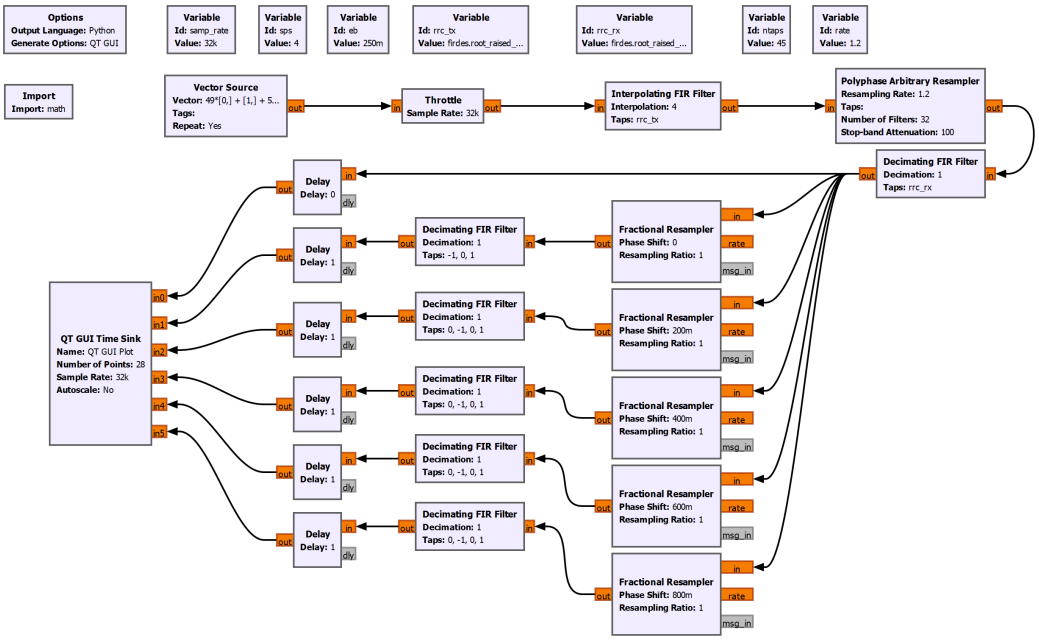

Рис. 3.8: symbol_differential_filter_phases схема

Видно, что сигнал, помеченный как `d(sym0)/dt+phi3` в правильной точке выборки равен 0. Это говорит о том, что наша идеальная точка выборки возникает при этом сдвиге фазы. Поэтому, если мы возьмем фильтр `RRC` нашего приёмника и настроим его фазу на $3 * \frac{2}{5}\pi$, то мы сможем исправить несоответствие синхронизации и выбрать идеальную точку выборки.

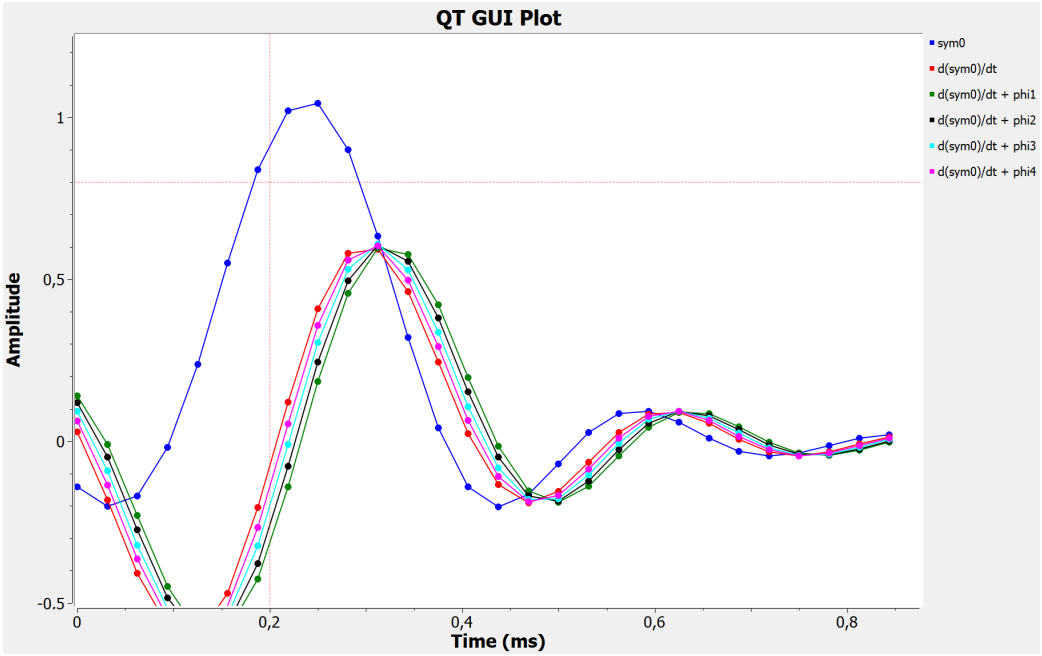

Рис. 3.9: Визуализация потокового графа с фильтрами

Но это лишь моделируемое приближение, в действительности выборки каждого фильтра не будут происходить в один и тот же момент времени. Чтобы действительно увидеть такое поведение, мы должны увеличить частоту дискретизации в количество раз, равное количеству фильтров.

3.3 Использование блока синхронизации многофазных clock'ов в нашем приёмнике

Теперь применим этот блок в нашей симуляции. Блок настроен с 32 фильтрами и полосой пропускания петли $\frac{2}{100}\pi$.

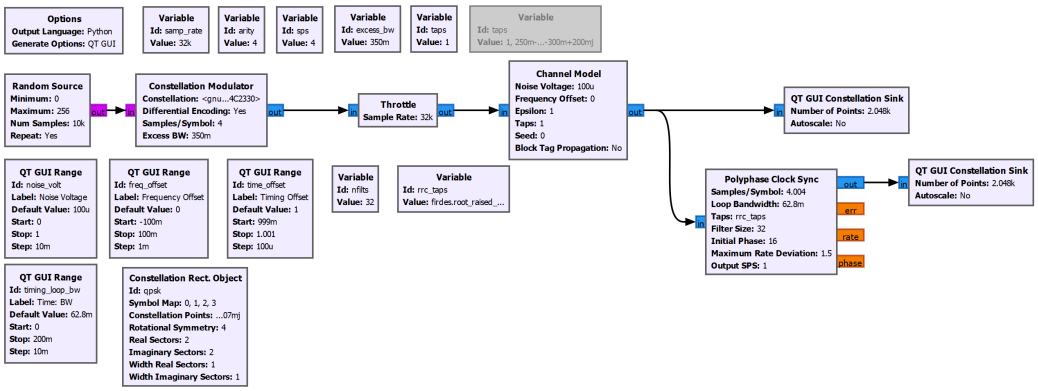

Рис. 3.10: mpsk_stage3 схема

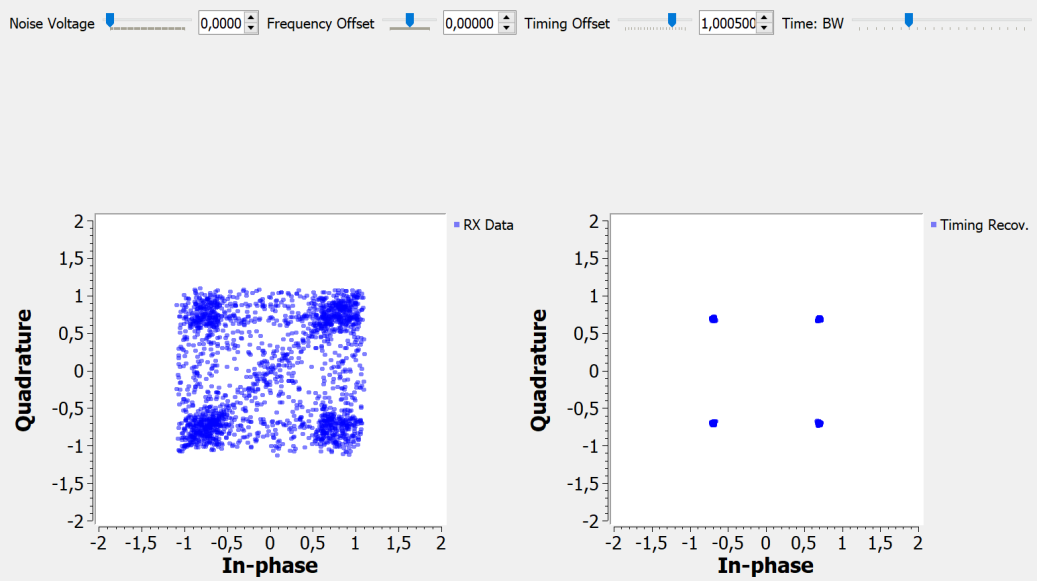

Рис. 3.11: Визуализация графа

На графике слева - сигнал до синхронизации, справа - после. Он всё ещё немного зашумлён из-за `ISI`, но этот шум быстро поглощается другим шумом при установке `Noise Voltage` для каналов отличным от нуля:

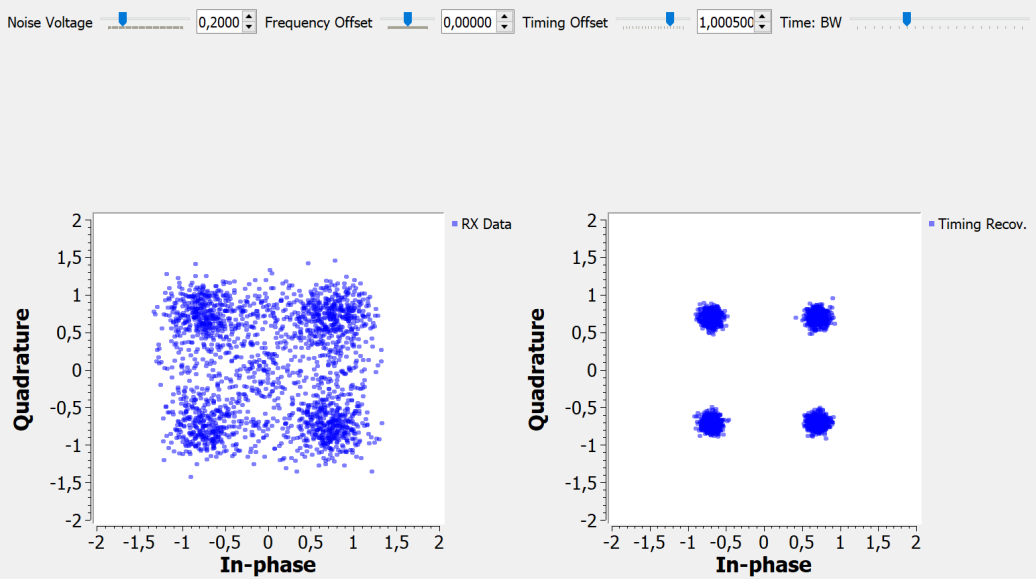

Рис. 3.12: Удаление шума

Если добавить смещение частоты, то созвездие станет окружностью. Созвездие всё ещё будет находится на единичной окружности, что означает, что тактовая синхронизация сохраняется, но блок не позволяет нам корректировать смещение частоты.

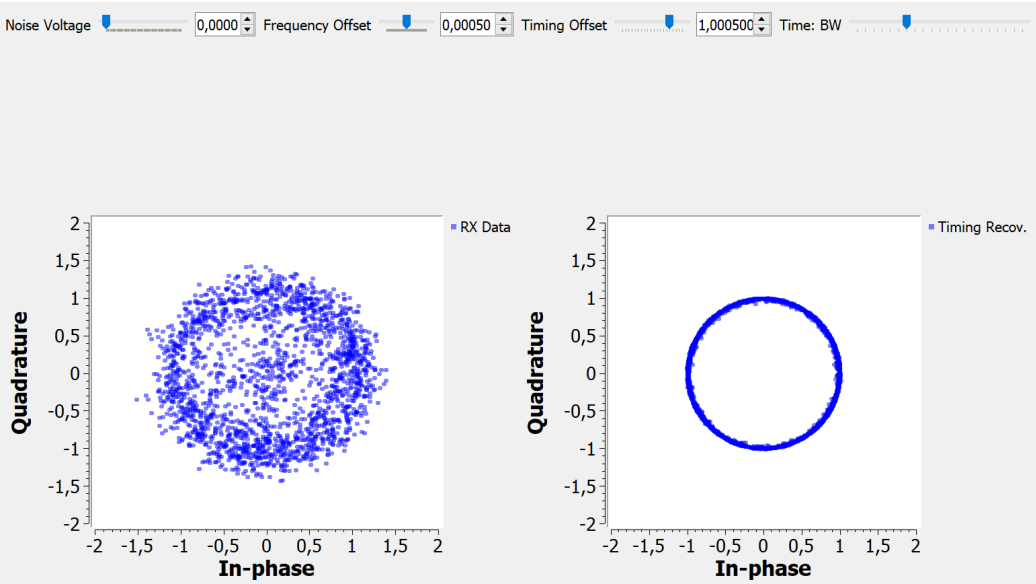

Рис. 3.13: Добавление смещения частоты

Глава 4

Многолучевое распространение

Воздействие комбинации сигналов на приёмник при многолучевом распространении - искажение сигнала. Если разница во времени между отражениями достаточно мала по сравнению с шириной символа, искажение может быть внутри символа - внутри-символьная интерференция. Если время отражения превышает время символа, отражение от одного символа будет влиять на следующие сигналы - ещё одна причина межсимвольной интерференции.

Чтобы исправить это поведение, мы можем использовать механизм, очень похожий на стереоэквалайзер. С помощью стереофонического эквалайзера мы можем изменить усиление определенных частот, чтобы либо подавить, либо усилить эти сигналы, наиболее распространенными из которых являются низкие и высокие частоты.

К сожалению, по каким-то причинам библиотеки scipy не было в файлах gnuradio, поэтому пришлось добавить её вручную.

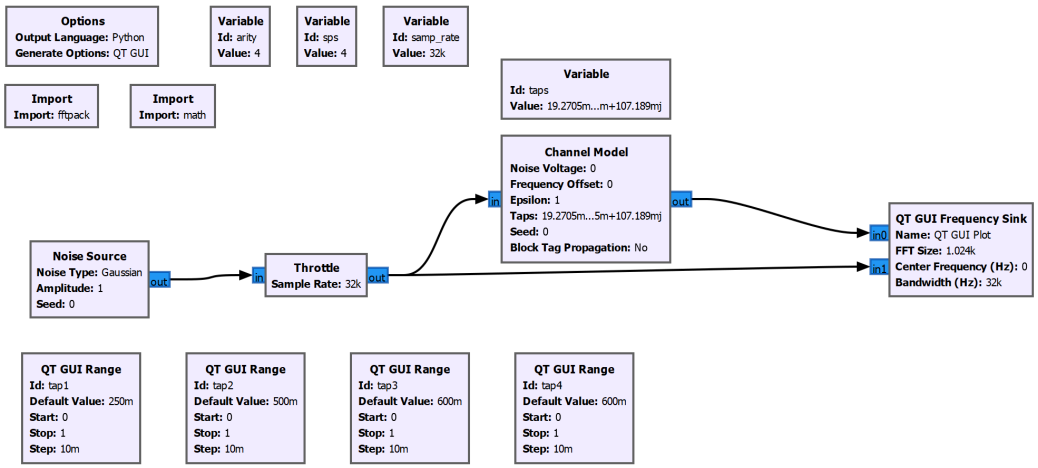

Рис. 4.1: multipath_sim схема

Это моделирование настраивает модель канала, чтобы предоставить каналу пять элементов управления эквалайзером, четыре из которых мы можем изменить. Хотя в этом примере мы явно контролируем частотную область, на самом деле мы играем с возможностью создать эквалайзер, который может корректировать или регулировать частотную характеристику принятого сигнала. В конечном итоге цель показана на рисунке ниже, где многолучевой канал создает некоторые искажения в сигнале, как показано в частотной области. Задача эквалайзера - инвертировать этот канал.

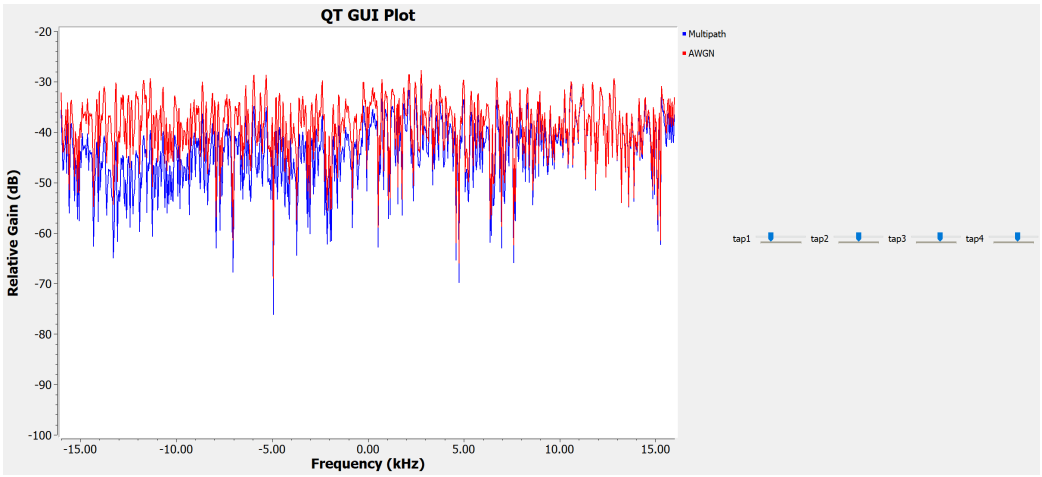

Рис. 4.2: Визуализация графа

Глава 5

Эквалайзеры

GNU Radio поставляется с двумя легко используемыми эквалайзерами, `CMA` и `DD LMS`. Мы сначала будем использовать `CMA (Constant Modulus Algorithm)`, а затем `DD LMS (Decision-Directed Least Mean Squared)`. CMA или алгоритм постоянного модуля - это слепой эквалайзер, но он работает только с сигналами с постоянной амплитудой или модулем. В примере ниже мы используем алгоритм CMA с 11 taps.

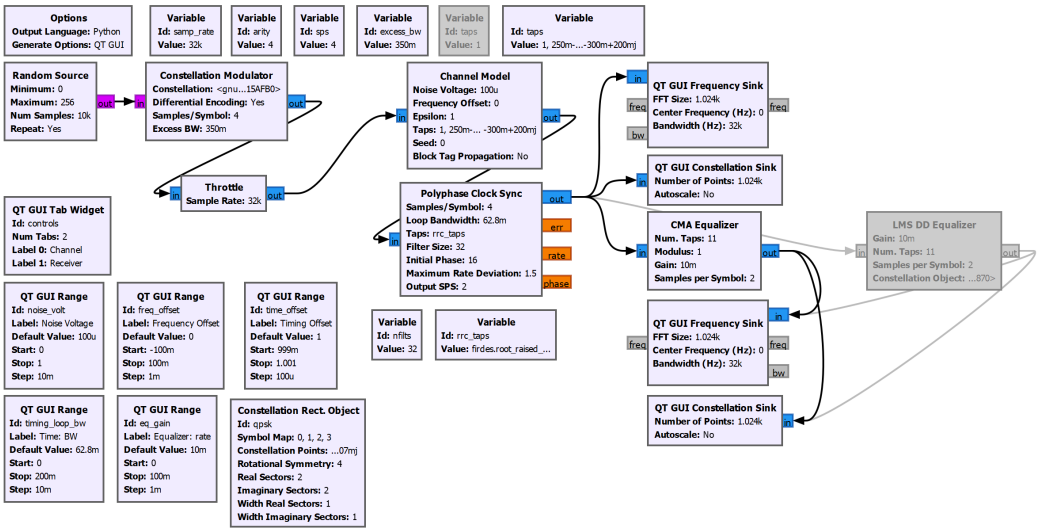

Рис. 5.1: mpsk_stage4 граф с `CMA`

Мы можем наблюдать сходимость алгоритма `CMA`. Перед эквалайзером у нас очень некрасивый сигнал даже без шумов. Эквалайзер прекрасно понимает, как инвертировать и отменить этот канал, чтобы у нас снова был хороший, чистый сигнал. Мы также можем видеть сам канал и то, как он красиво выравнивается после эквалайзера.

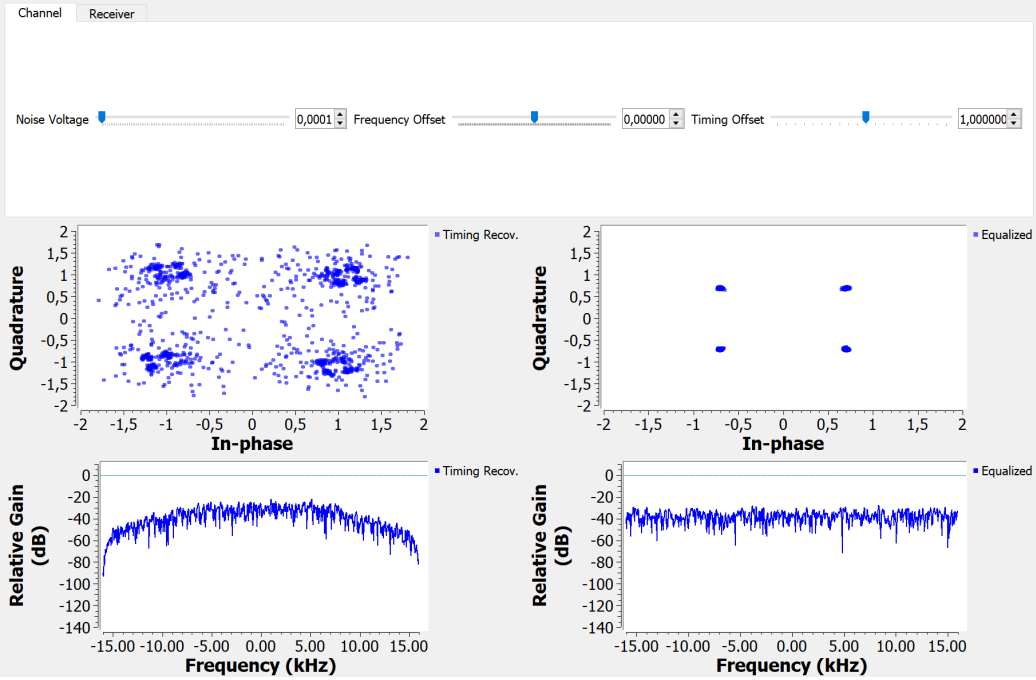

Рис. 5.2: Визуализация данных графа

Теперь рассмотрим `DD LMS`. `CMA` и `DD LMS` довольно похожи в плане параметров, за исключением одной важной особенности. `DD LMS`, в отличие от `CMA` – не слепой эквалайзер, он требует информации о принимаемом сигнале. Ему необходимо знать точки созвездия. Этот эквалайзер отлично подходит для сигналов, которые не соответствуют требованию постоянного модуля алгоритма `CMA`. С другой стороны, если `SNR` достаточно плох, принимаемые решения будут неправильными, что может ухудшить производительность приемника. Кроме того, `DD LMS` более сложен в вычислительном отношении. Для `DD LMS` рассматривать графики не будем, так как результат на них максимально близок с теми, что мы рассмотрели ранее.

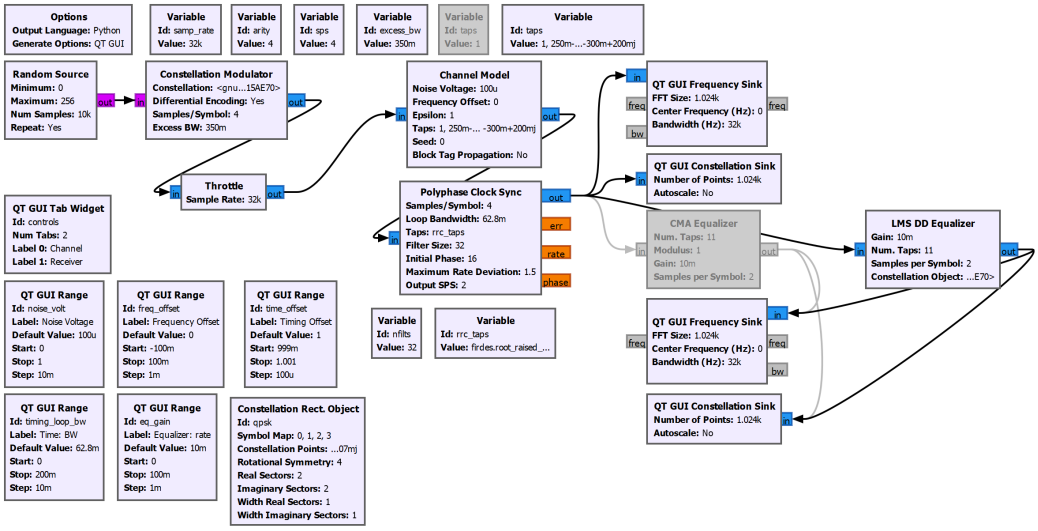

Рис. 5.3: mpsk_stage4 граф с `DD LMS`

Глава 6

Фазовая и точная частотная коррекция

Учитывая, что мы выровняли канал, у нас всё ещё есть проблема смещения фазы и частоты. Эквалайзеры, как правило, не адаптируются быстро, поэтому смещение частоты может быть легко за пределами возможностей эквалайзера. Кроме того, если мы просто запускаем эквалайзер `CMA`, всё, о чем он заботится, - это схождение к единичной окружности. Он ничего не знает о созвездии, поэтому, когда он блокируется, он блокируется на любой заданной фазе. Теперь нам нужно исправить любой сдвиг фазы, а также любой сдвиг частоты.

Для этой задачи мы собираемся использовать `Costas Loop` в примере. Блок `Costas Loop` может синхронизировать `BPSK`, `QPSK` и `8PSK`. Приёмник созвездия будет привязан к любому заданному объекту созвездия, хотя в зависимости от созвездия функция принятия решения может быть более или менее сложной.

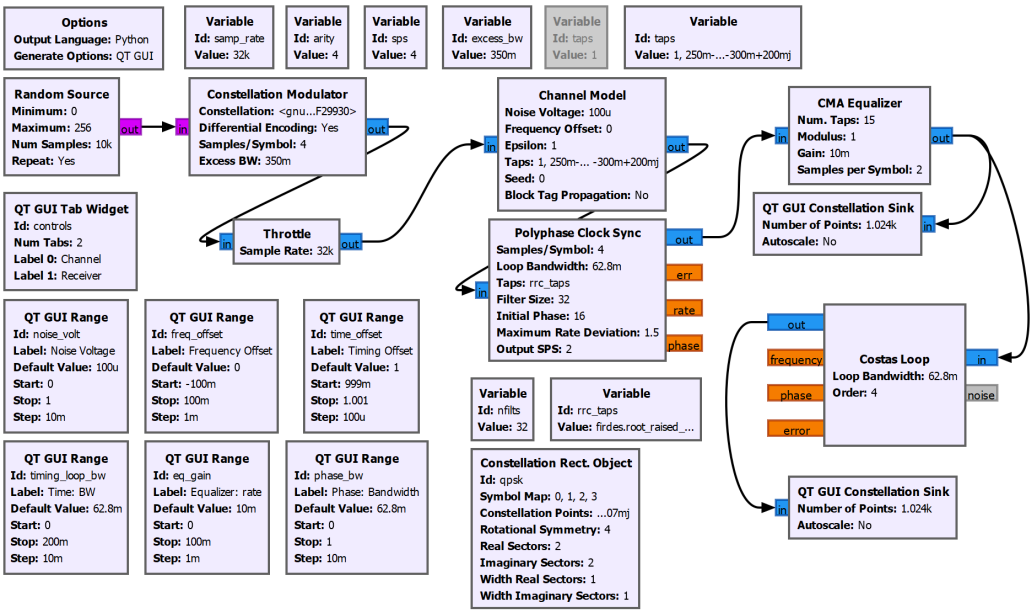

Рис. 6.1: mpsk_stage5 граф

Установим шум, временной сдвиг, простой многолучевой канал и частотный сдвиг. После эквалайзера мы видим, что все символы находятся на единичном круге, но вращаются из-за сдвига частоты, который еще ничего не исправляет. На выходе блока `Costas Loop` мы можем видеть заблокированное созвездие, как мы начали, плюс дополнительный шум, с которым мы ничего не можем поделать.

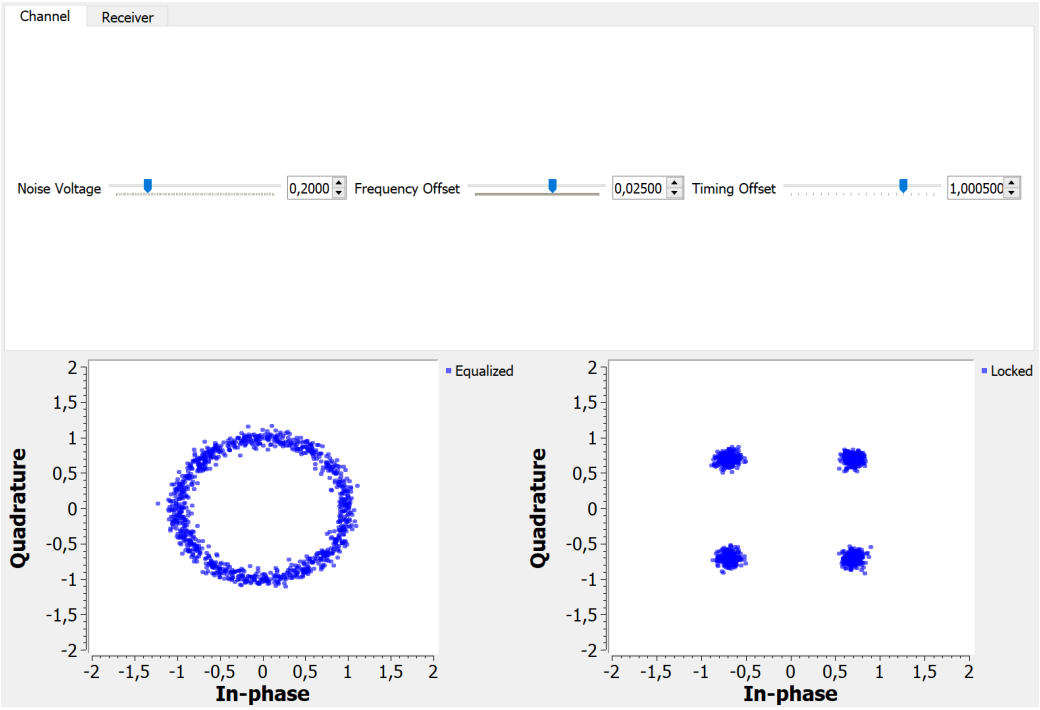

Рис. 6.2: Визуализация данных графа

Глава 7

Декодирование

Теперь мы можем декодировать сигнал. Для этого мы вставляем `Constellation Decoder` после `Costas Loop`, а также изменяем параметр `Differential` в блоке `Constellation Modulator` значение на False.

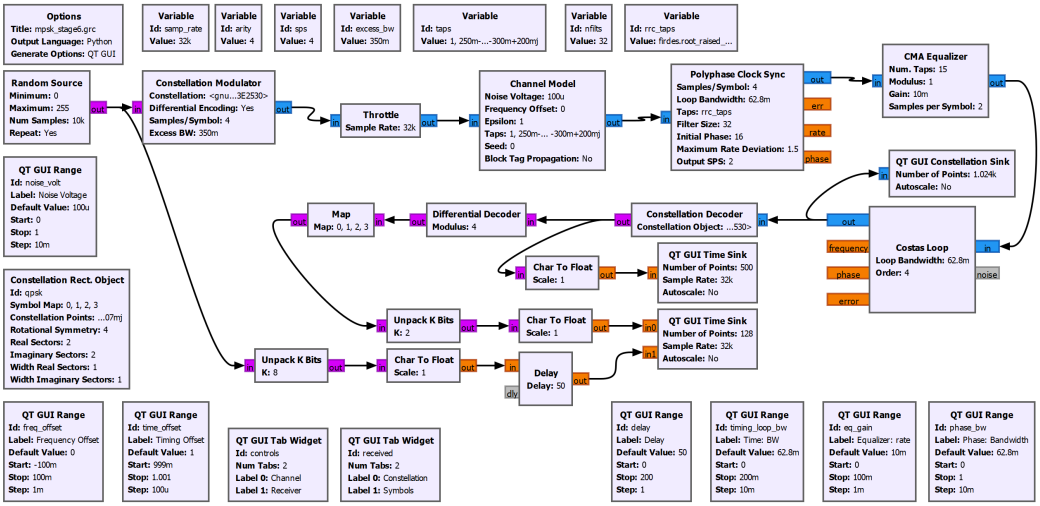

Рис. 7.1: mpsk_stage6 граф

Здесь мы можем настроить задержку, чтобы найти правильное значение и посмотреть, как синхронизируются биты. Вы также можете вычесть один сигнал из другого, чтобы увидеть, когда они синхронизированы, так как на выходе будет 0. Добавление шума и других влияний на канал можно легко увидеть как битовые ошибки, если этот сигнал не равен 0.

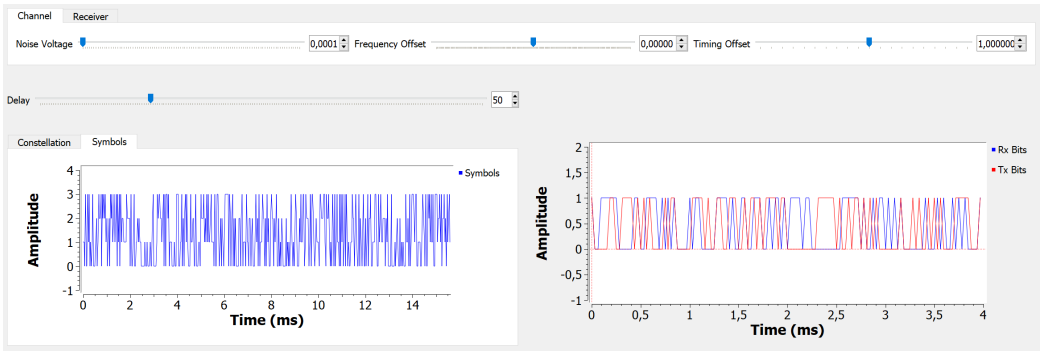

Рис. 7.2: Визуализация данных графа

Глава 8

Выводы

В процессе выполнения работы мы разобрались с тем, как произвести установку и настройку GNU Radio, а также рассмотрели множество явлений, связанныйх с передачей QPSK-сигнала. Симулируя настоящие условия передачи данных, мы столкнулись с рядом возможных трудностей, а также рассмотрели и пути их решения, что поможет нам в будущем при организации настоящих каналов передач.