Hands-on 9

Parte 01: Transmissão/Recepção PSK sem fio em canais banda limitados em banda

(baseado no material disponibilizado pelo Tom Rondeau)

Introdução Teórica

Modulação PSK

A modulação PSK (*Phase Shift Keying*) é um esquema de modulação digital que tem por finalidade transmitir dados através da alteração da fase de uma onda portadora. Por ser um esquema de modulação com implementação simples e robusta, é muito utilizado em sistemas de comunicações sem fio. Neste esquema de modulação associa-se cada fase a um determinado conjunto de bits, ou seja, cada conjunto de bits representará um símbolo, como mostrado na figura 1. O caso mais simples de modulação PSK é o BPSK (*Binary Phase Shift Keying*), que transmite um bit (0 ou 1) através da mudança de fase da portadora em dois valores, 0°e 180º. A figura 2 ilustra a representação em diagrama de constelação de um sinal BPSK. Embora seja uma modulação robusta ao ruído aditivo, a taxa de transmissão obtida com esse esquema de modulação não é alta, já que associa-se apenas 1 bit por símbolo.

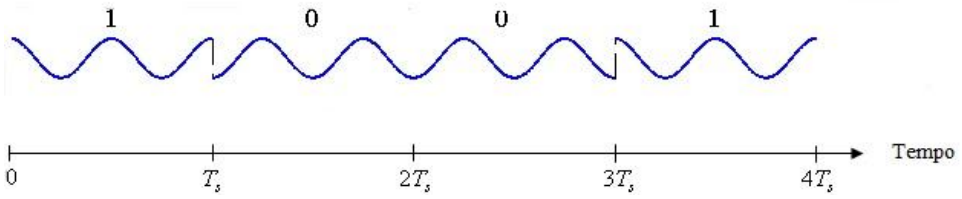


Figura 1: Bits e suas representações no sistema BPSK.

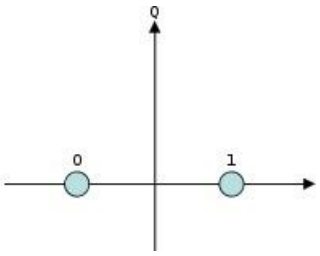


Figura 2: Representação em Diagrama de Constelação do BPSK.

Decisão e erro

Como na modulação BPSK a informação binária é atribuída às mudanças de fase da portadora, na recepção, a decisão é feita observando-se a fase do sinal recebido. Diz-se que ocorreu erro de transmissão, quando no envio de um bit “0” o receptor decide pelo bit “1”, e vice-versa. Tal equívoco na decisão é ocasionado por interferências de diversas origens (e.g. AWGN). Assim, os sistemas de transmissão digitais têm como um dos principais parâmetros para análise de desempenho a taxa de erro, pois um serviço de transmissão digital precisa de um certo patamar de taxa de erro para garantir a qualidade de serviço apropriada.

De forma a se obter uma estimativa do comportamento do sistema são utilizados modelos matemáticos para o cálculo da probabilidade de erro, a qual é dependente do tipo de canal. Tomando como base um canal sob influência de ruído branco gaussiano (canal AWGN), podemos calcular a probabilidade de erro ao utilizar a modulação BPSK como:

$$P_e = erfc\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

sendo E_b a energia de bit; e N_0 a densidade espectral de potência do ruído.

Referências

[1] http://en.wikipedia.org/wiki/Phase-shift_keying#Binary_phase-shift_keying_.28BPSK.29 – Acesso em 20/01/2014

[2] http://en.wikipedia.org/wiki/Raised-cosine_filter#Roll-off_factor – Acesso em 20/01/2014

[3] Proakis, Salehi, Bauch; Modern Communication Systems Using Matlab® ; 3ªedição; Cengage Learning.

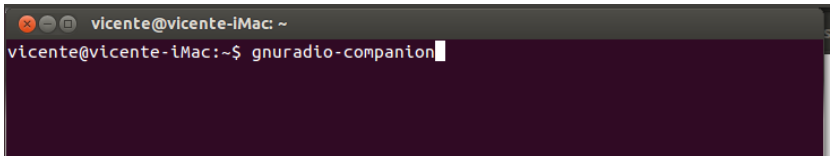
[4] Dayan, Rausley; Transmissão Digital, Princípios e Aplicações; 1ª edição; Editora Érica.

[5] http://www-ee.uta.edu/dip/Courses/EE4330/comparison%20of%20modulation%20methods.pdf

Exercício

OBJETIVO: Repetir os loopbacks realizados pelo Tom Rondeau, um dos principais desenvolvedores do GNU Radio. Tratam-se de experimentos de uma cadeia de transmissão/recepção digital usando modulação PKS, simulados num ambiente com a presença de ruído AWGN, multipercurso e offsets de tempo e frequência. Com base nas características do canal, são estabelecidos métodos mitigadores para a recuperação da fase e frequência do sinal.

1. Caso ainda não esteja aberto, inicialize o GNU Radio Companion.
    a. Abra um terminal digitando CRTL+ALT+t
    b. Digite: gnuradio-companion e pressione ENTER

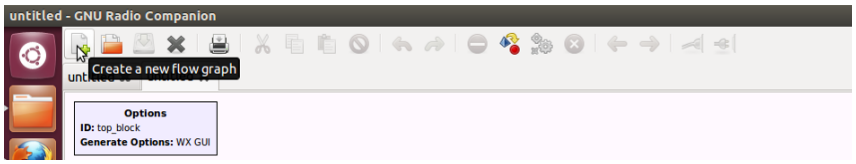


Alternativa:

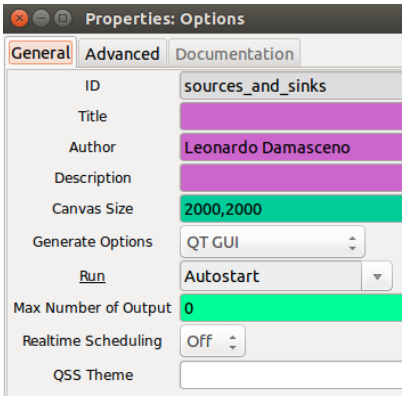
a. Clique em Dash Home

b. Digite gnuradio e clique no ícone correspondente ao GRC

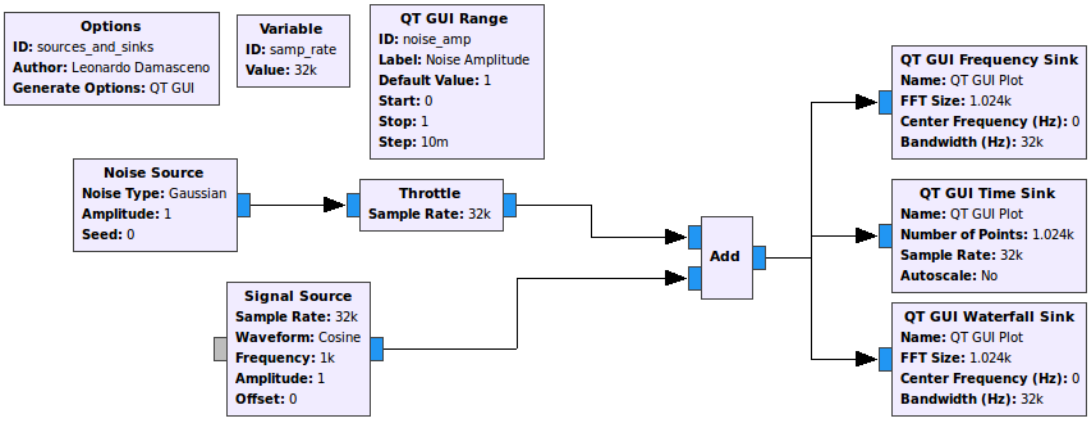
2. Com o GNU RADIO COMPANION aberto, crie um novo projeto



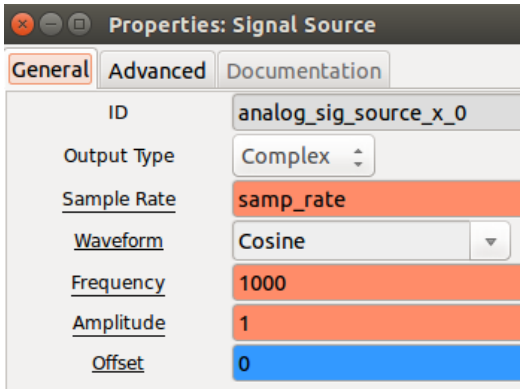
3. Clique duas vezes no Bloco ***Options***. Esse bloco configura alguns parâmetros gerais do *flowgraph*. Mantenha o ID como top_block. Digite um título para o projeto e um autor. Selecione *Generate Options* como *QT GUI*, *Run* para *Autostart* e *Realtime Scheduling* para *Off*. Então, feche a janela de propriedades.



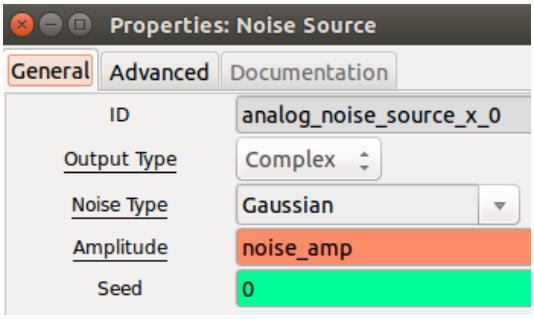
4. A princípio será simulado o comportamento de uma fonte de ruído adicionada a uma onda cossenoidal, e o resultado será visualizado no domínio do tempo e da frequência. Para isso, construa seu projeto utilizando os blocos ***Noise Source***, ***Signal Source***, ***Throttle***, ***Add***, ***QT GUI Frequency Sink***, ***QT GUI Time Sink***, ***QT GUI Waterfall Sink*** e ***QT GUI Range***. Mantenha o valor do *Sample Rate* do bloco ***Variable*** em *32000,* e altere em todos os blocos o campo *Type* para *Complex*. Com isso, interligue os blocos de forma que sua área de trabalho fique similar à figura a seguir.



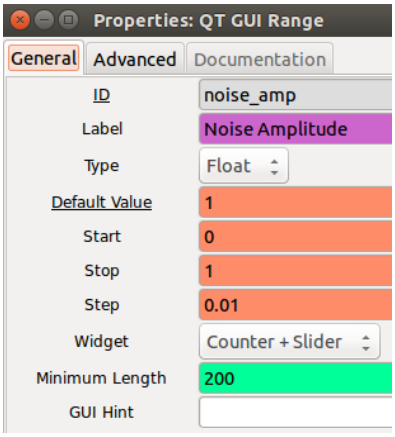
5. Primeiro, configure o bloco ***Signal Source***, atribuindo-o uma forma de onda cossenoidal (*Waveform* selecionado em *Cosine*), com Frequência de 1000 Hz, *Amplitude* igual a 1 e *Offset* igual a 0. Assim, o bloco deve estar com os parâmetros configurados como o da figura abaixo.



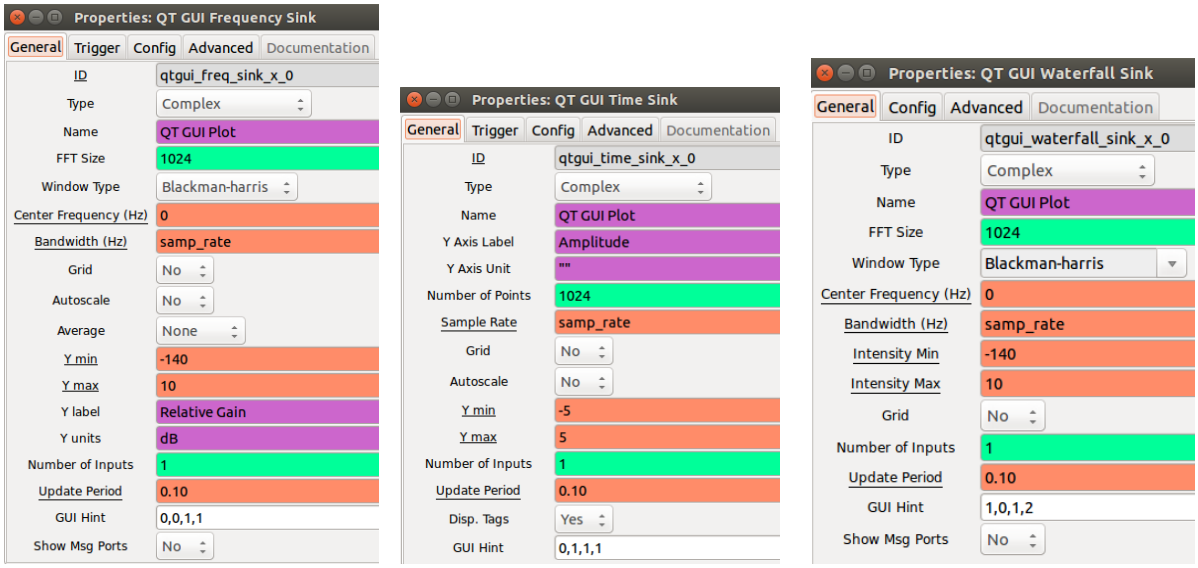
6. O bloco ***Noise Source*** deve ser configurado com *Output Type* selecionado para *Complex*, *Noise Type* como *Gaussian* (o popular ruído AWGN), no campo da *Amplitude* digite *noise_amp*, que é um parâmetro que será configurado pelo bloco ***QT GUI Range*** para fazer o controle da amplitude do ruído, enquanto a simulação é executada. Por fim, atribua 0 para o parâmetro *Seed*.



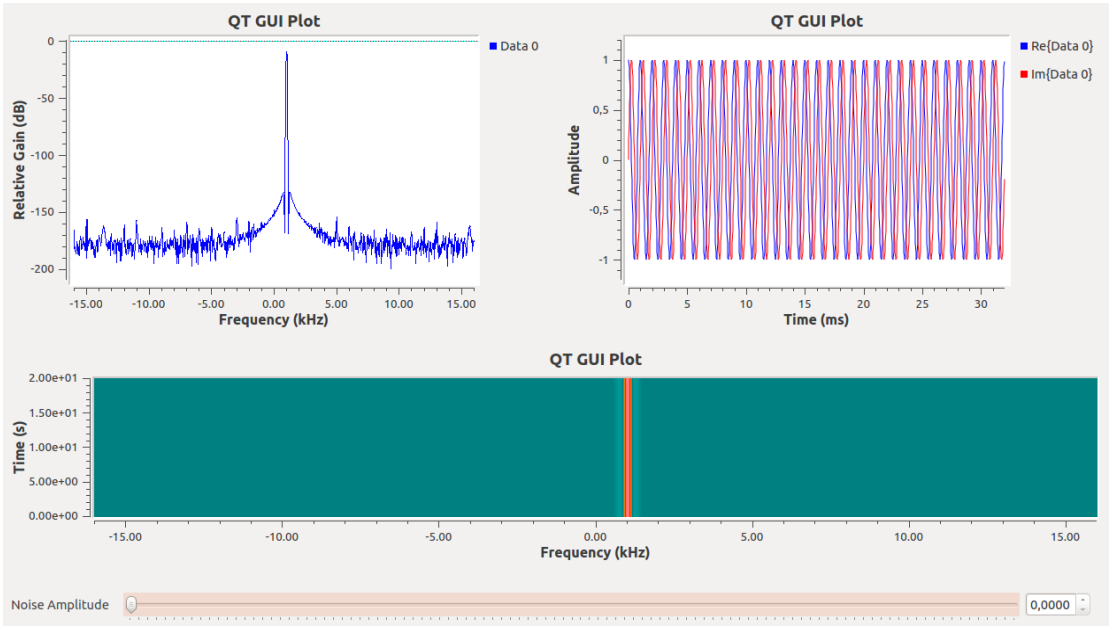
7. Como já comentado no item anterior, configure agora o bloco ***QT GUI Range*** com os seguintes parâmetros: *ID* para *noise_amp*, *Label* para *Noise Amplitude* e *Type* para *Float*. Determine o *Default Value* para 1, *Start* em 0, *Stop* igual a 1 e o *step* de 0.01. Para uma melhor visualização do parâmetro de controle do ***QT GUI Range***, selecione *Widget* para *Counter + Slider* e *Minimum Length* igual a 200.

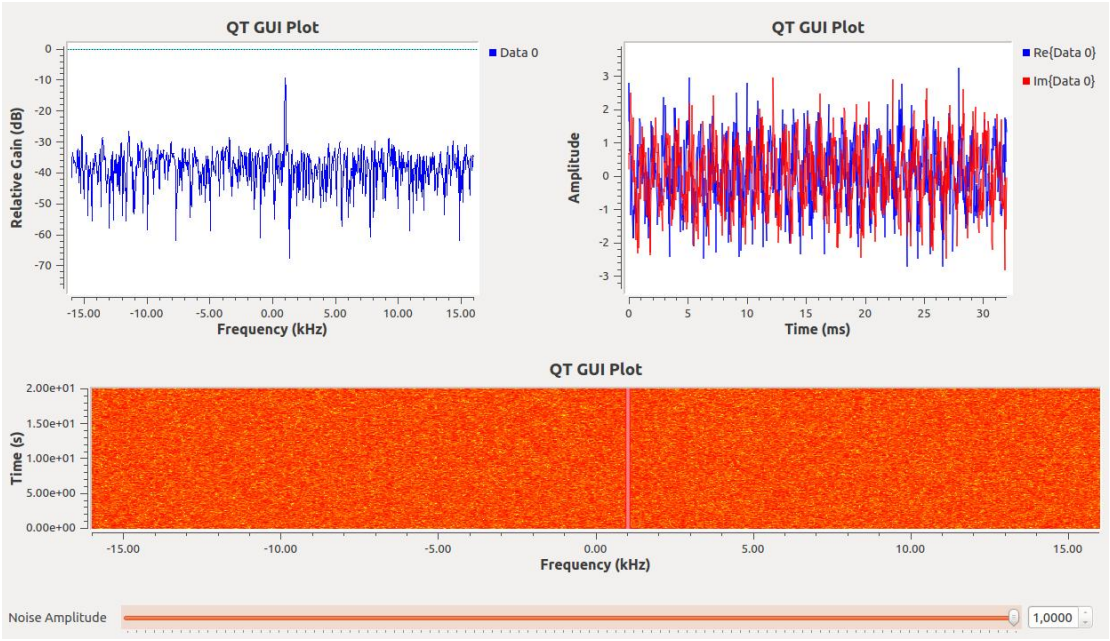


8. Para finalizar as configurações, edite os parâmetros dos blocos ***QT GUI Frequency Sink***, ***QT GUI Time Sink***, ***QT GUI Waterfall Sink*** conforme às figuras abaixo. Esses blocos são responsáveis por mostrar a onda cossenoidal no domínio da frequência, do tempo e através de um espectrograma, respectivamente.

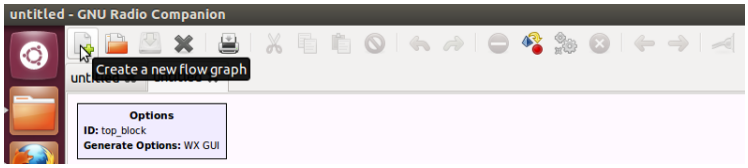


9. Feito isso gere o gráfico e execute-o. Os resultados das figuras a seguir representam a onda cossenoidal para dois cenários: sem ruído (*Noise Amplitude* igual a 0) e com a amplitude máxima do ruído (*Noise Amplitude* igual a 1).

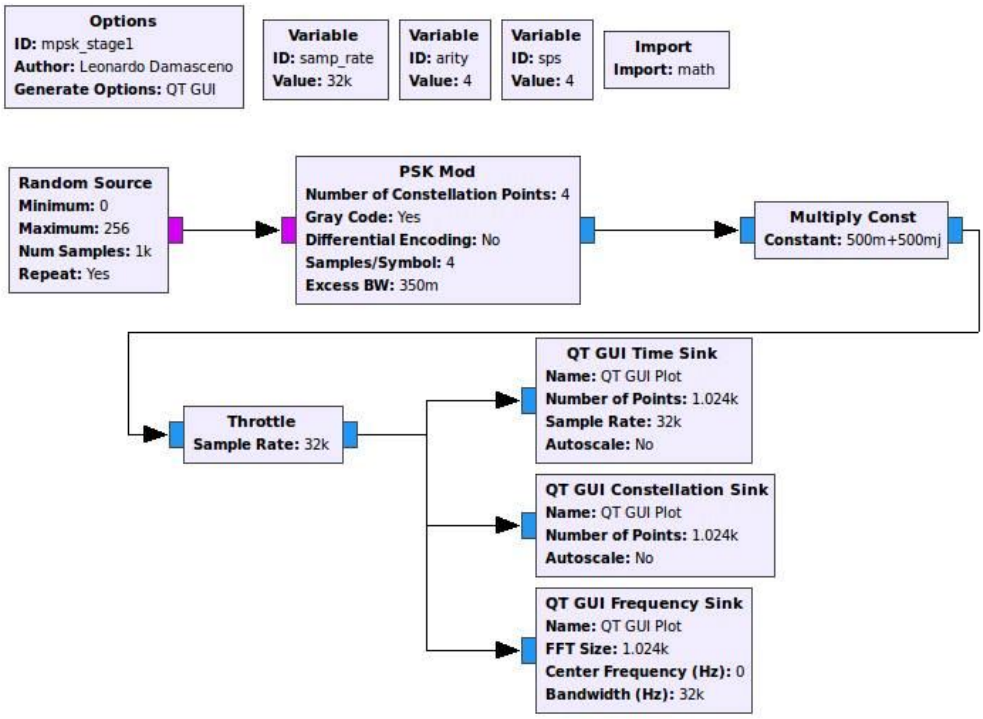




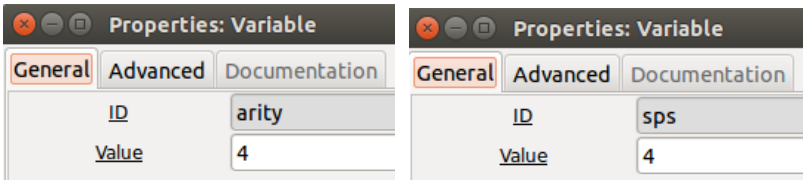
10. Agora crie um novo projeto ou edite o primeiro projeto criado neste hands-on.



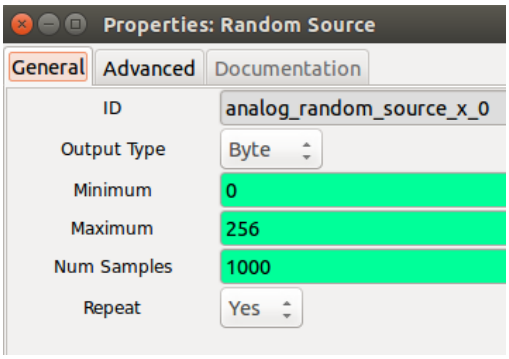
11. Construa (edite) este novo projeto utilizando três blocos ***Variable***, um bloco ***Import***, um bloco ***Random Source***, um bloco ***PSK Mod***, um bloco ***Multiply Const***, um bloco ***Throttle***, um bloco ***QT GUI Frequency Sink***, ***QT GUI Time Sink*** e ***QT GUI Waterfall Sink***. Conecte os elementos de forma que sua área de trabalho fique igual à figura a seguir.



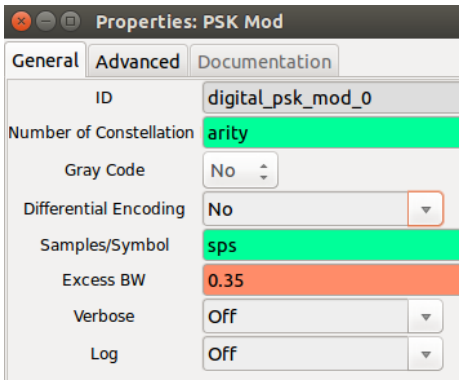
12. Feita as conexões vamos iniciar a configuração dos parâmetros de cada bloco, começando pelos blocos ***Variable***. Clique duas vezes no bloco cujo *ID* é *samp_rate* e altere o campo *Value* para 32000. Agora abra outro bloco ***Variable*** e edite seu *ID* para *arity* e o campo *Value* para 4, no último bloco ***Variable***, edite o ID para *sps* e o campo *Value* para 4. Confira como deve ficar a configuração dos blocos a seguir.



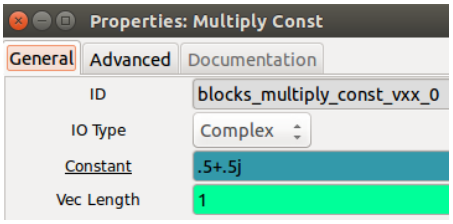
13. Agora no bloco ***Random Source***, edite os campos: *Output Type* para *Byte; Minimum* para *0, Maximum* para *256, Num Samples* para *1000* e a opção *Repeat* para *Yes.* Esse bloco gera uma fonte de informação aleatória para análise, com 1000 amostras. Observe a figura abaixo.



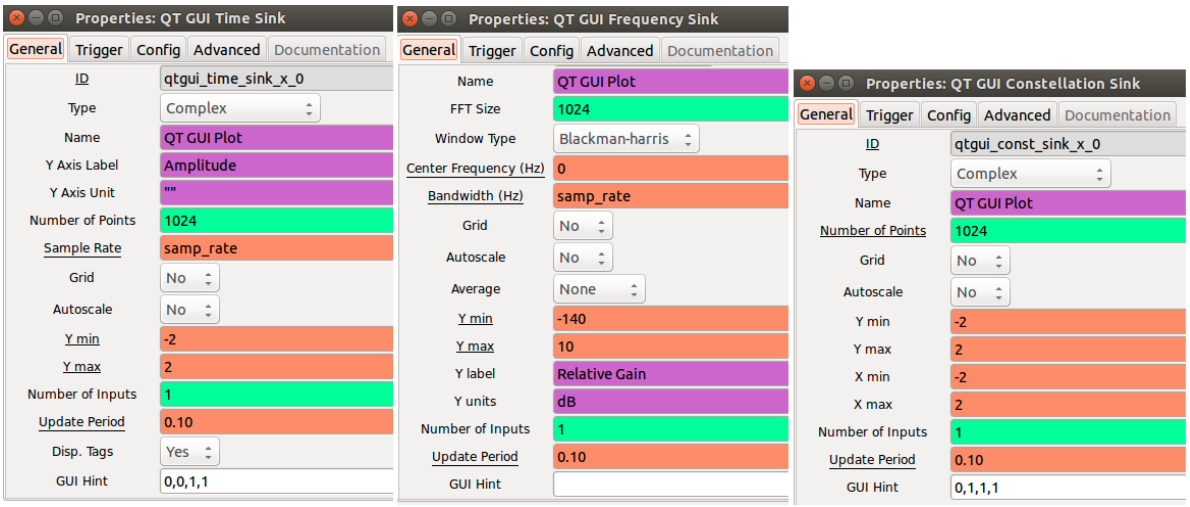
14. Agora configure o bloco ***PSK Mod***, responsável por realizar a modulação da mensagem em *Byte*. Edite o campo *Number of Constellation* para *arity*, selecione *Gray Code* e *Differential Encoding* para *No* e *Samples/Symbol* para *sps*. Inicialmente, atribua no campo *Excess BW* o valor de 0.35 (depois será simulado por valores de 0.1 e 0.7), este parâmetro representa o fator de *roll-off* do sistema. Por fim, selecione *Off* tanto no campo *Verbose* quanto *Log*. Neste caso, a modulação utilizada será a QPSK, transmitindo 4 bits/símbolo. Veja como deve ficar a configuração do bloco na figura a seguir.



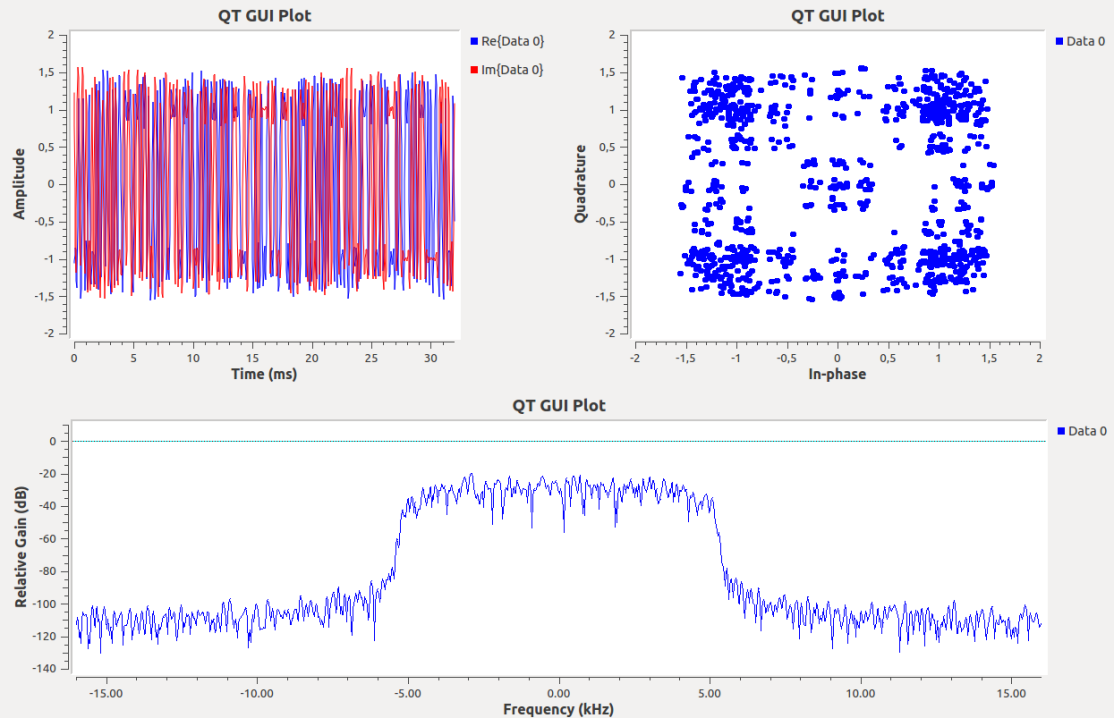
15. Configure o bloco ***Multiply Const***, alterando o campo *IO Type* para *Complex*, *Constant* para *.5+.5j* e *Vec Length* igual a 1. Este bloco funciona como um amplificador, em que o parâmetro *Constant* é multiplicado pela amplitude do sinal de entrada.



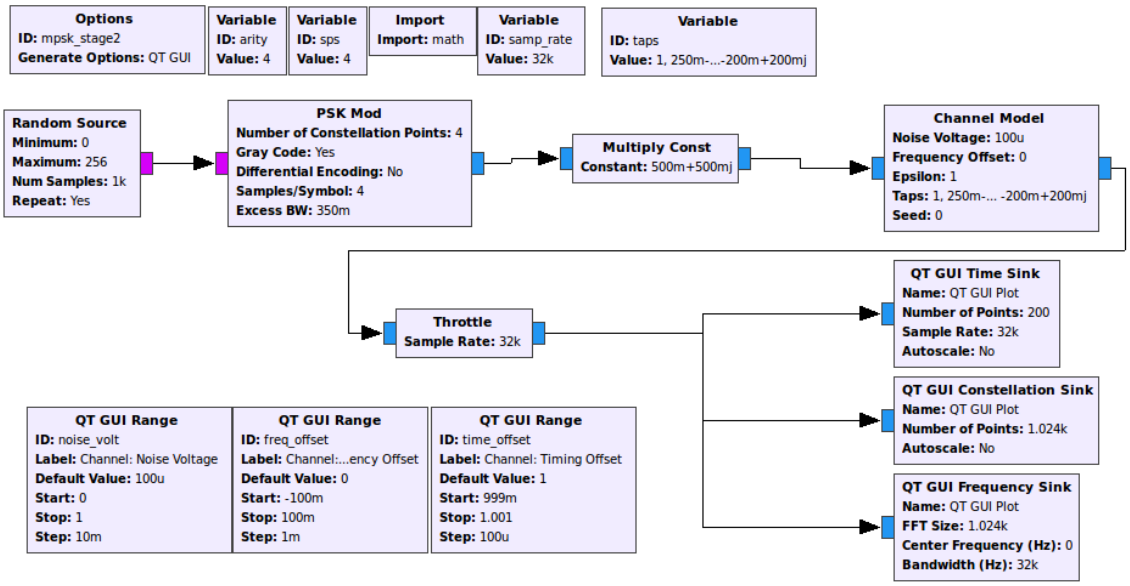
16. Por fim, edite os parâmetros dos blocos ***QT GUI Frequency Sink***, ***QT GUI Time Sink***, ***QT GUI Waterfall Sink*** conforme às figuras abaixo.



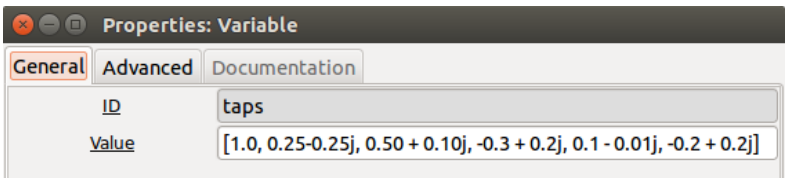
17. Feito isso, execute o *flowgraph* e observe como deve ficar os gráficos iniciais do experimento.



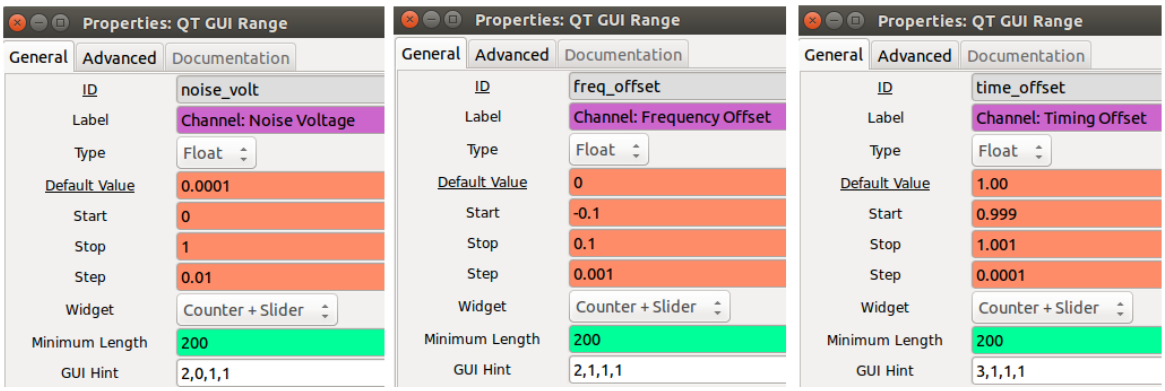
18. Agora deve ser adicionado o canal no sistema, com o intuito de simular uma situação de transmissão real. Para isso, adicione mais um bloco ***Variable***, um bloco ***Channel Model*** *e três blocos* ***QT GUI Range*** ao *flowgraph*. Interligue os blocos de maneira semelhante à figura abaixo.



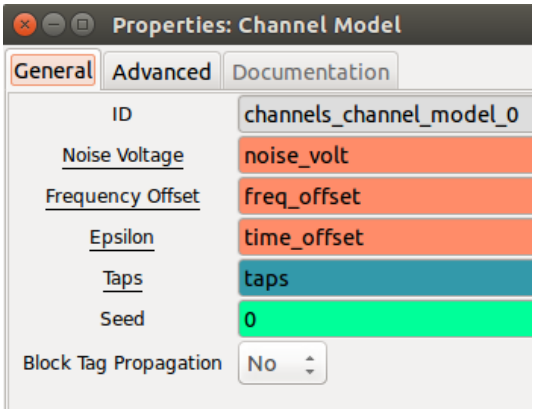
19. Configure o novo bloco ***Variable*** de maneira similar à figura a seguir. Este bloco é responsável por adicionar as características de multipercurso do canal, nos quais os parâmetros configurados no campo *Value* representam os *taps* do canal, funcionando como um filtro atenuador.



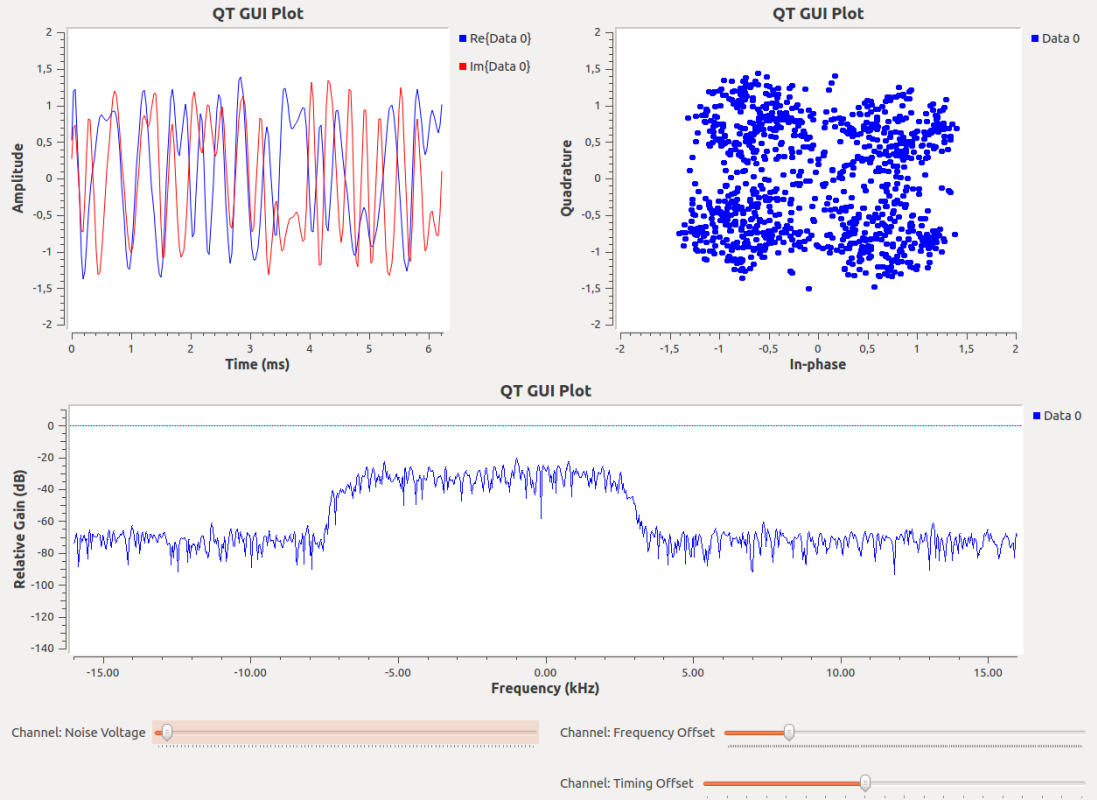
20. Em seguida, configure os três blocos ***QT GUI Range*** de acordo com as figuras ilustradas a seguir. Os blocos serão configurados para controlar e adicionar a intensidade do ruído, *offset* de frequência e *offset* de tempo, respectivamente, do canal.



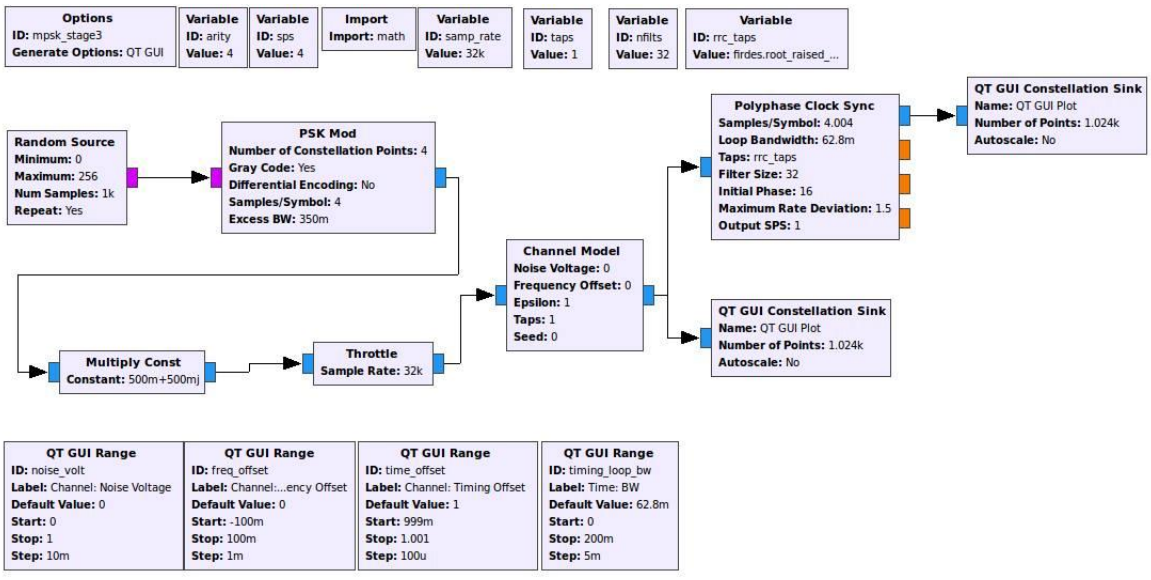
21. Depois de executar os itens 19 e 20, configure o bloco ***Channel Model*** de forma que os parâmetros iniciais do mesmo sejam preenchidos pelos *ID's* atribuídos nos blocos ***QT GUI Range*** e ***Variable***. Assim, o campo *Noise Voltage* será preenchido pelo parâmetro *noise_volt*, *Frequency Offset* pelo *freq_offset*, *Epsilon* pelo *time_offset* e *Taps* por *taps*. Atribua 0 para o campo *Seed* e selecione *No* para *Block Tag Propagation*.



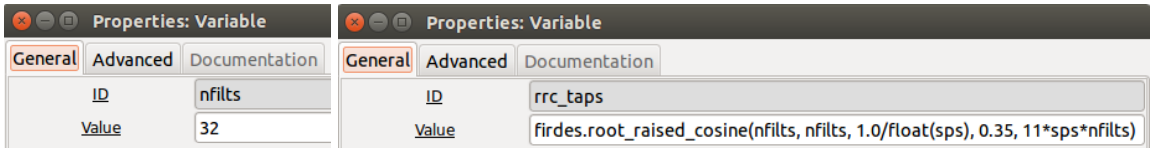
22. Depois de configurar todos os blocos descritos, gere o gráfico e execute-o. Os gráficos devem se comportar conforme é ilustrado na figura a seguir. Observe os efeitos do canal no sinal recebido tanto na constelação, como no domínio do tempo e da frequência. A constelação do demodulador encontra-se espelhadas entre as regiões de decisão, apresentando características de problemas ocasionados pelo processo de formatação de pulso do modulador, provocando ISI e falta de sincronismo de tempo.



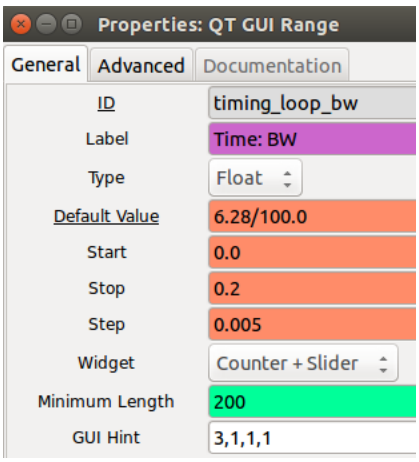
23. A partir de agora serão utilizados métodos para recuperar o tempo de bit e corrigir as distorções de fase do sinal, assim como melhorar seu ganho. Para iniciar esse processo de recuperação do sinal, adicione os seguintes blocos na sua área de trabalho: dois blocos ***Variable***, um bloco ***QT GUI Range***, um bloco ***Poliyphase Clock Sync*** e um bloco ***QT GUI Constellation Sink*** (com isso, desabilite ou remova os blocos ***QT GUI Frequency Sink*** e ***QT GUI Time Sink***). Mantenha a interligação dos blocos semelhante à figura abaixo.



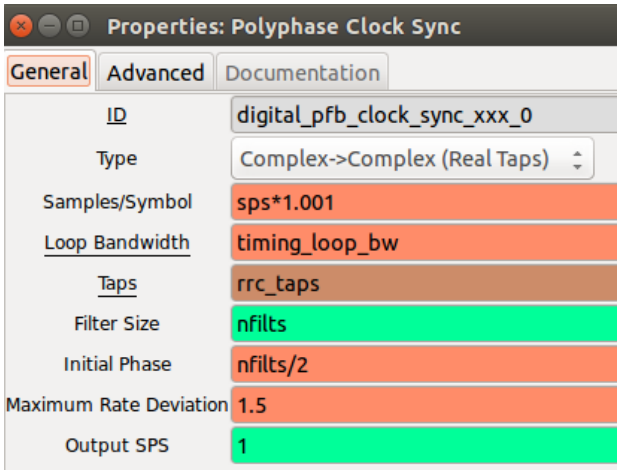
24. Agora configure os blocos ***Variable*** de acordo com a ilustração das figuras a seguir. Ambos são configurados para trabalhar em conjunto com o bloco ***Poliyphase Clock Sync***, sendo o primeiro bloco responsável por determinar o número de filtros derivadores e o segundo responsável por realizar a operação do filtro casado.



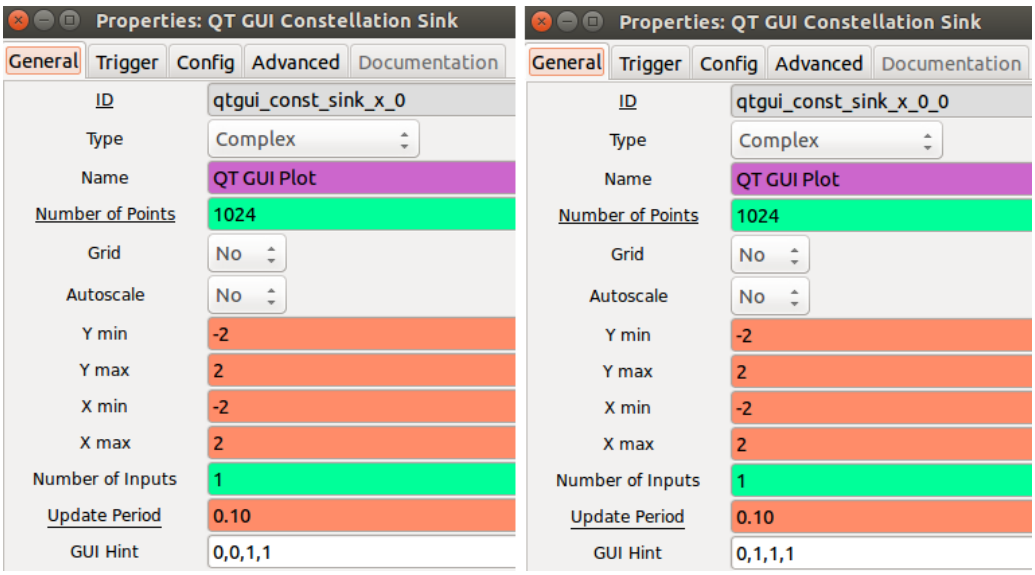
25. Depois disso, configure o bloco ***QT GUI Range*** de acordo com a figura abaixo. Esta configuração é feita para realizar o controle da largura de banda do sistema.



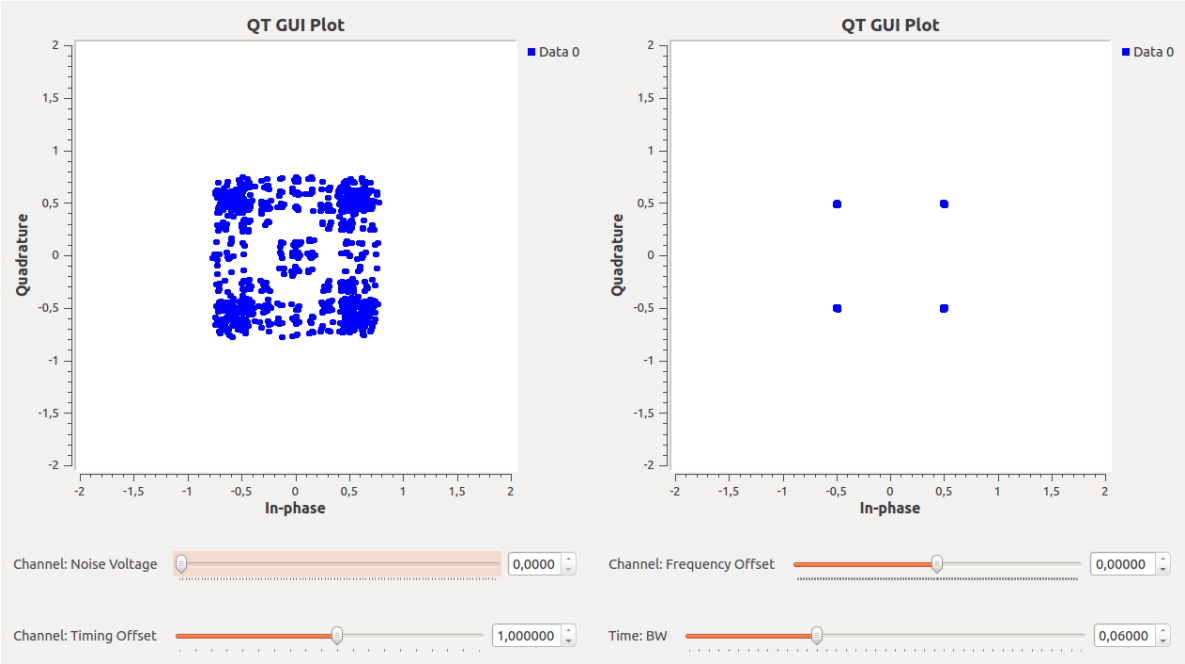
26. Agora configure o bloco ***Poliyphase Clock Sync*** com os parâmetros iguais aos editados na figura abaixo. Este bloco resolve o problema do sincronismo de tempo, implementando o PFB (Banco de Filtros Polifásicos) e a operação do filtro casado.



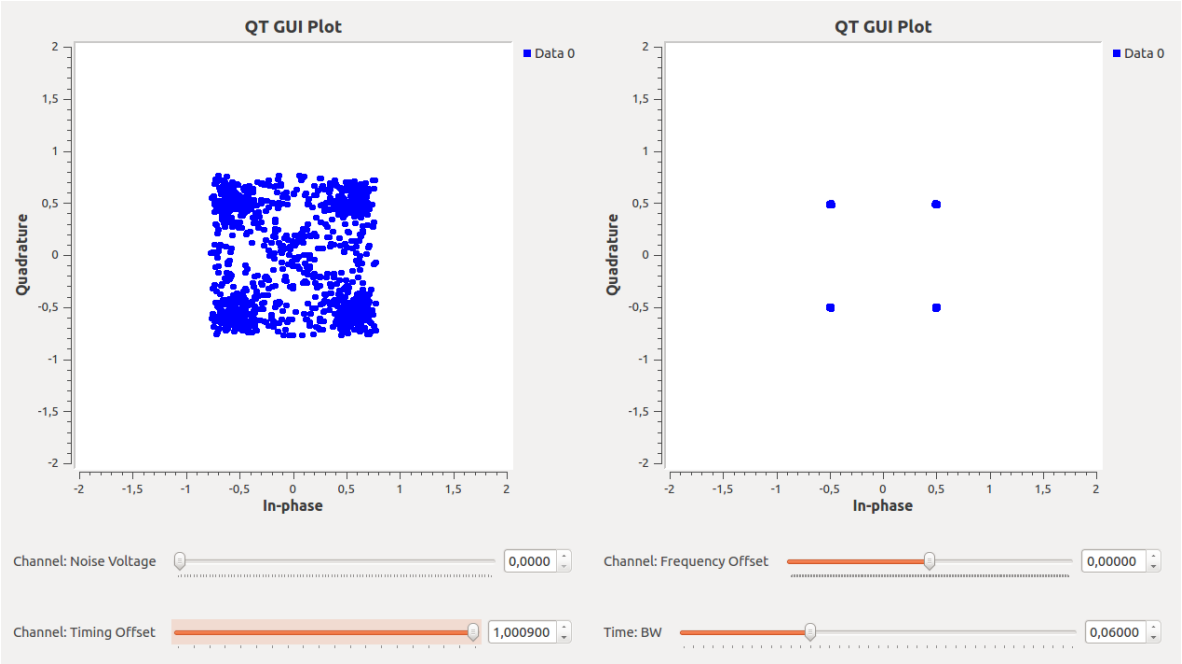
27. Para finalizar, configure os dois blocos ***QT GUI Constellation Sink***. As figuras a seguir ilustram a configuração do bloco antes e depois do bloco ***Poliyphase Clock Sync***, respectivamente.



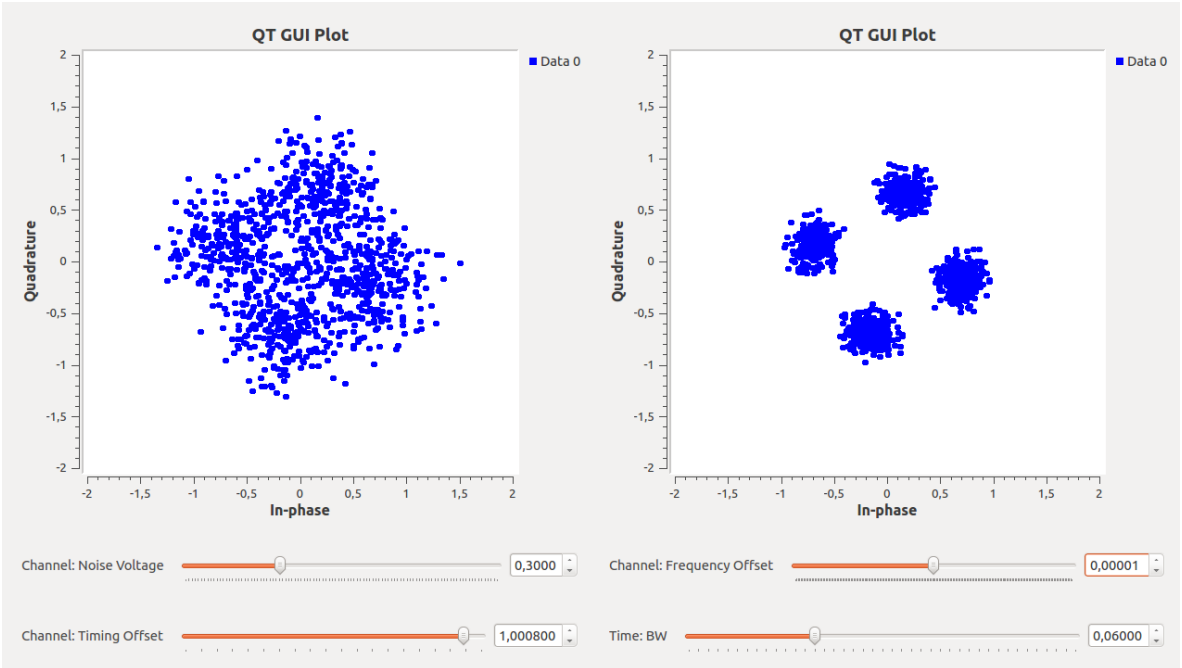
28. Feito isso, gere o *flowgraph* e execute-o. O resultado gerado deve ser similar ao da figura abaixo. O gráfico à esquerda mostra a constelação antes do bloco ***Poliyphase Clock Sync*** e a constelação à direita é o sinal tratado após o uso do mesmo.



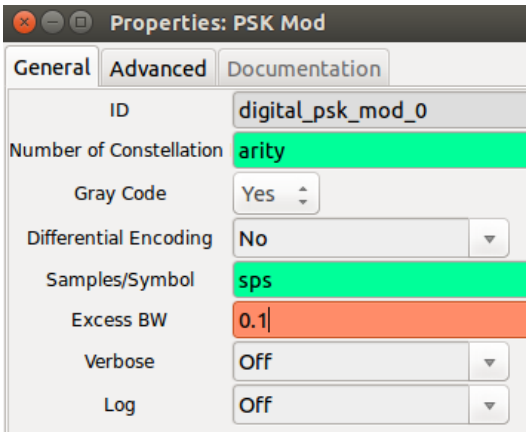
29. Observe na figura seguinte, que mesmo aumentando o parâmetro do *offset* de tempo (*Timing Offset*) para o valor máximo, consegue-se obter a recuperação do sinal.



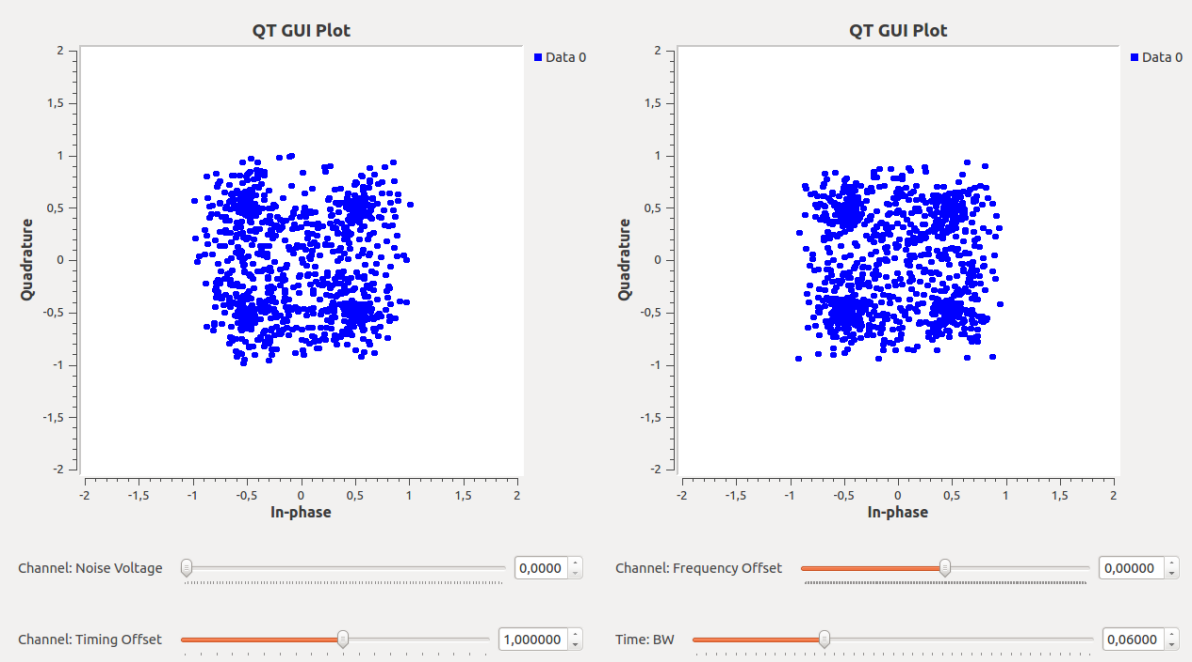
30. Agora serão mostrados dois cenários nos quais o bloco ***Poliyphase Clock Sync*** não consegue recuperar a constelação corretamente. No primeiro cenário, é a ocorrência do *offset* de frequência. Veja na figura abaixo, que com um *offset* (deslocamento) de $1x10^{-5}$ na frequência do sinal, a constelação entra em rotação e perde o sincronismo de fase.



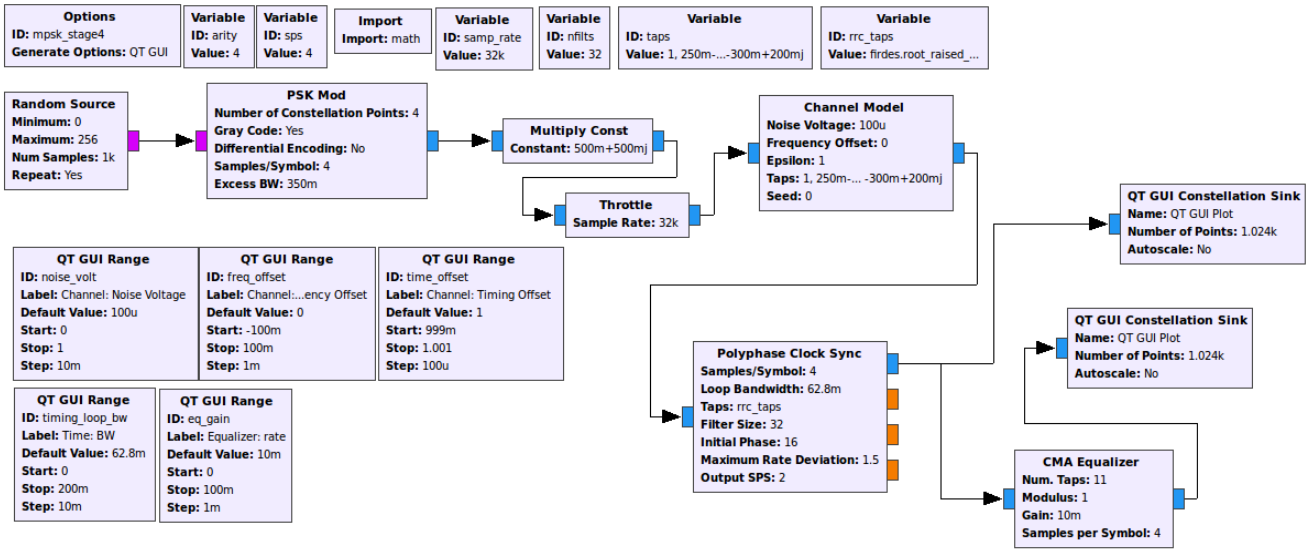
31. No outro cenário, deve ser alterado o fator de *roll-off* que foi configurado no bloco ***PSK Mod***. Para isso, altere o valor do parâmetro *Excess BW* de *0.35* para *0.1*. Esta mudança causa o efeito de ISI no sistema. Veja abaixo como deve ficar as configurações do bloco.



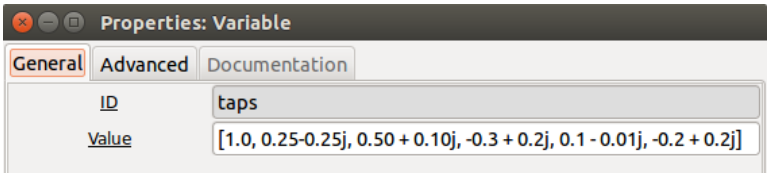
32. Feito isso, gere e execute o *flowgraph*. Observe na figura a seguir que, com a ocorrência da ISI, a constelação encontra-se espalhadas na região de decisão, dificultando extremamente a recuperação dos bits transmitidos.



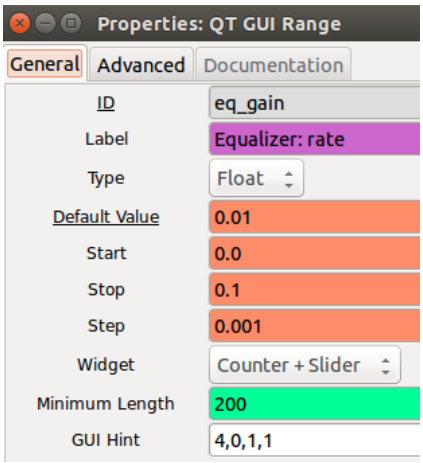
33. Mantendo as mesmas configurações do *flowgraph* atual, acrescente o bloco ***CMA Equalizer*** e mais um bloco ***QT GUI Range***. Com isso, desconecte o bloco ***QT GUI Constellation Sink*** que estar na saída do bloco ***Channel Model*** e conecte-o na saída do ***Poliyphase Clock Sync***. E o outro bloco ***QT GUI Constellation Sink*** deve ser conectado na saída do bloco ***CMA Equalizer***. Interligue os blocos de maneira que sua área de trabalho fique igual à figura abaixo.



34. Antes de configurar os novos blocos adicionados, clique no bloco ***Variable*** cujo *ID* é *taps* e altere o campo *Value* para os mesmos valores que foram editados no item 19. Veja como deve ficar os valores digitados na figura abaixo.

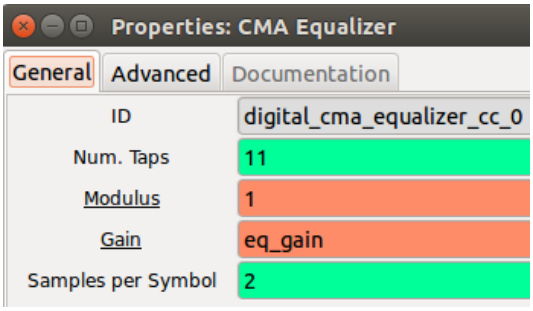


35. Feito isso, configure o novo bloco ***QT GUI Range*** com os mesmos parâmetros ilustrados na figura a seguir. Este bloco vai ser responsável pelo controle do ganho do Equalizador, podendo ser alterado durante o tempo de execução.

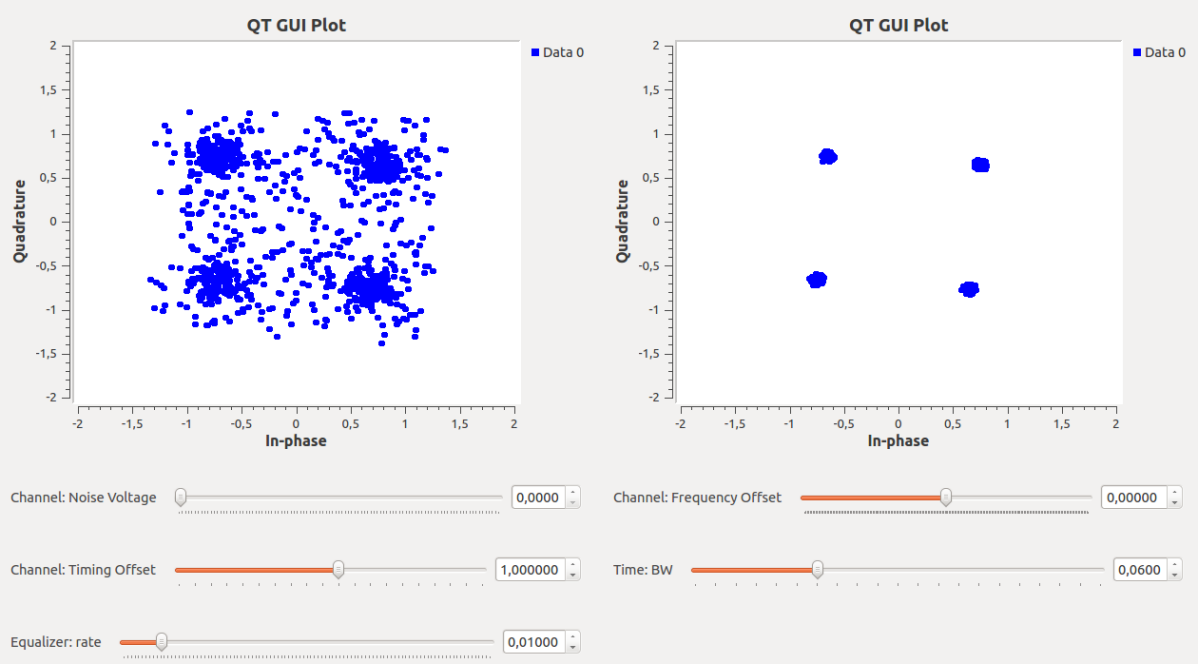


36. Para finalizar, configure o bloco ***CMA Equalizer*** atribuindo *Num. Taps* igual a 11 (esse valor é estabelecido pelo autor, com base nas características do canal), *Modulus* igual 1, *Gain* para *eq_gain* (parâmetro configurado no bloco ***QT GUI Range*** para o controle do ganho) e *Samples per Symbol* igual a 2. Este bloco serve para combater o problema de ISI, ocasionado no experimento

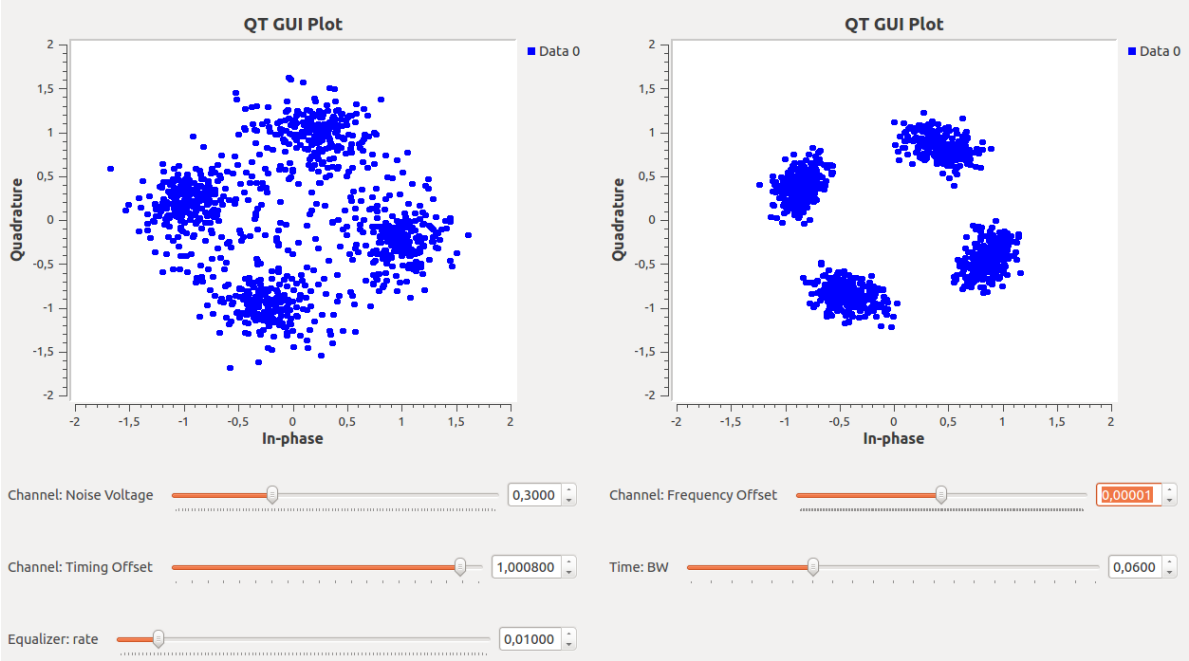
anterior. O bloco deve estar configurado conforme a figura a seguir.



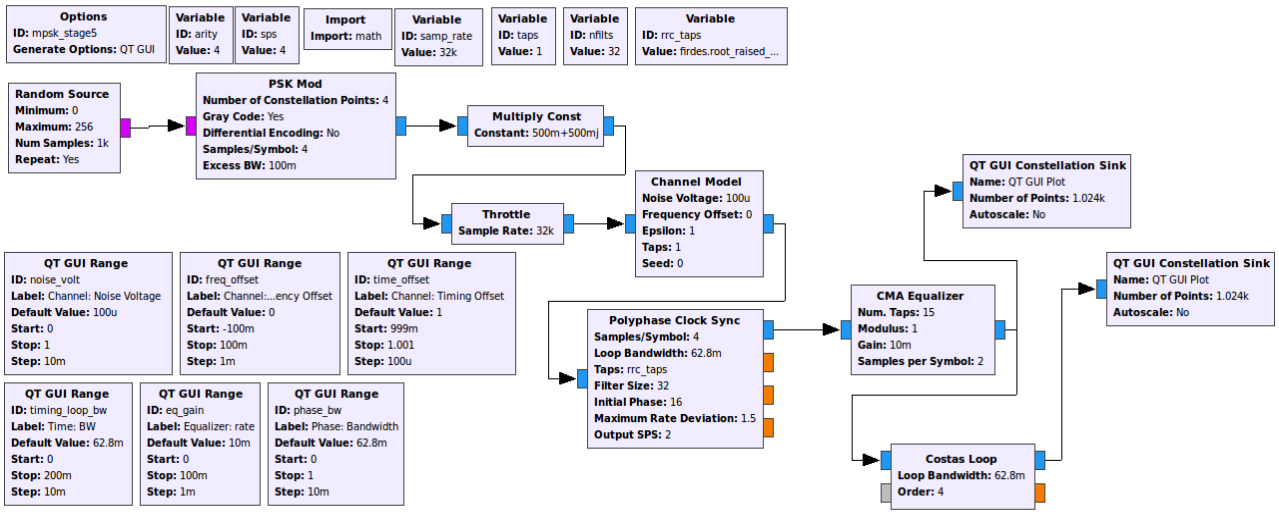
37. Gere e execute o *Flowgraph*. Observe que mesmo com fator de *roll-off (Excess Band)* configurado para 0.1, o equalizador consegue evitar a ISI.



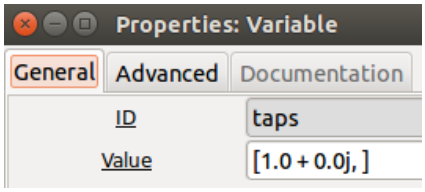
38. Portanto, um dos cenários propostos foi resolvido adicionando um Equalizador na cadeia de recepção. Agora, no mesmo gráfico em execução, gere um *offset* de $1x10^{-5}$ na frequência e observe como a constelação perde totalmente o sincronismo da fase.



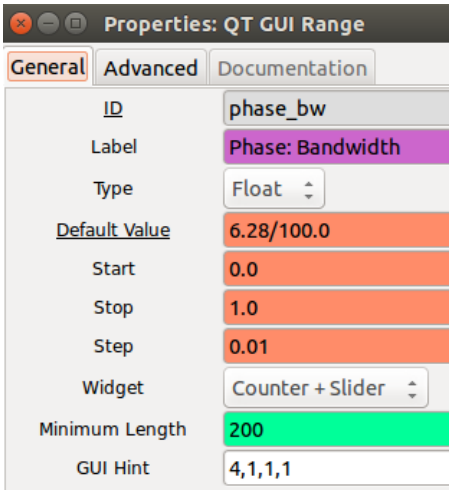
39. Para resolver este problema de sincronismo de fase gerado pelo *offset* de frequência, usaremos um dos métodos mais eficazes para realizar sincronismo de fase e frequência, que é o algoritmo *Costas Loop*. Assim, adicione na sua área de trabalho o bloco ***Costas Loop*** e mais um bloco ***QT GUI Range***. Interligue os blocos de maneira que o *flowgraph* fique semelhante à figura abaixo.



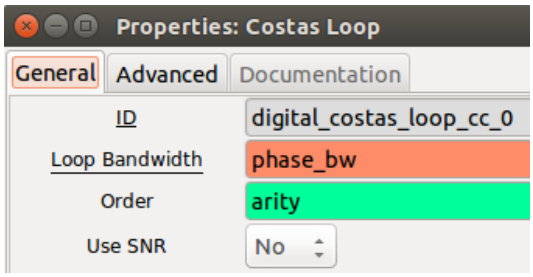
40. Antes de configurar os novos blocos, faça uma alteração novamente no bloco ***Variable*** cujo *ID* é *taps*, substituindo os parâmetros de multipercurso para *1*. Verifique como deve ficar o bloco.



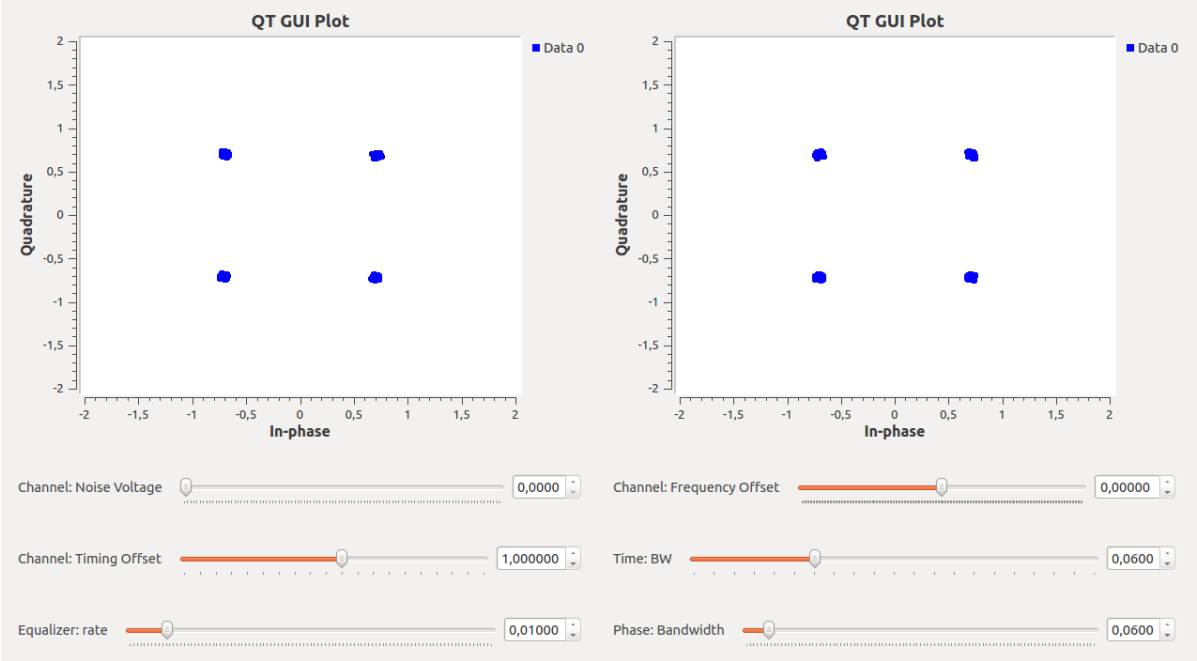
41. Agora, configure o bloco ***QT GUI Range***, conforme é ilustrado na figura abaixo. Este bloco atribui os parâmetros de fase do sistema, que será controlado pelo bloco ***Costas Loop***.



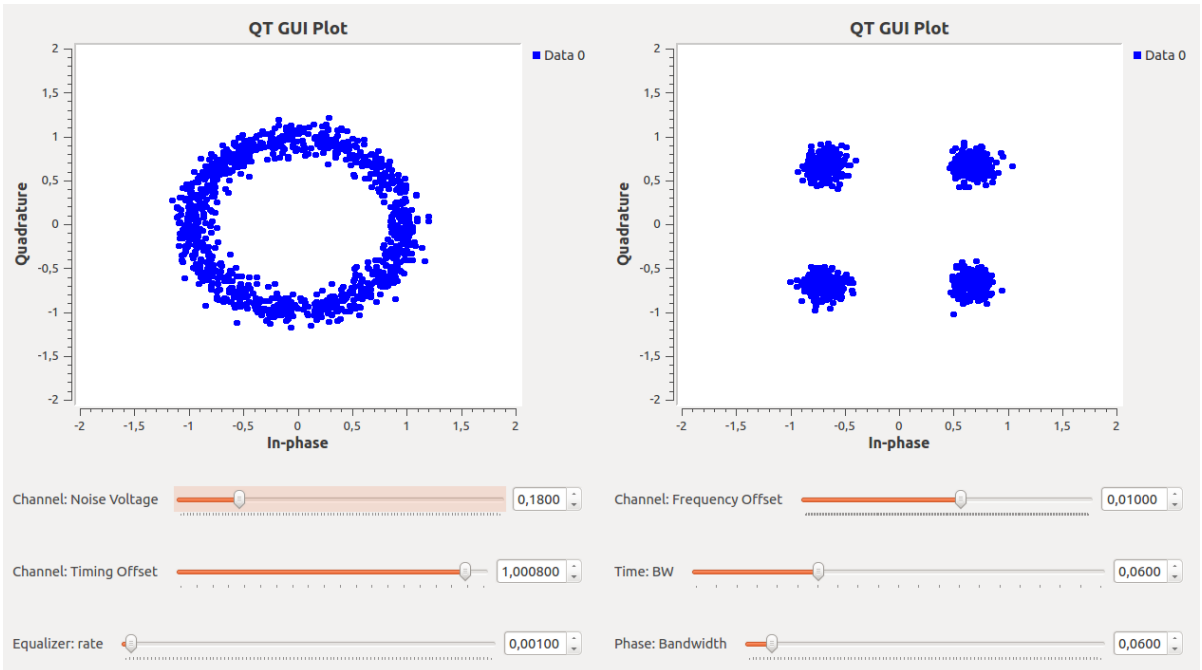
42. Para finalizar, configure o bloco ***Costas Loop***, atribuindo o *ID* do bloco ***QT GUI Range*** configurado no item anterior no campo *Loop Bandwidth*, no campo *Order* digite a variável *arity* (configurada anteriormente em um dos blocos ***Variable***, que representa o número 4, ou seja, o bloco ***Costas Loop*** terá ordem 4, devido ser QPSK) e selecione *No* para o campo *Use SNR*. Este bloco serve para estimar a fase e frequência no receptor, assim como apresenta robustez na presença de ruído. Veja como deve ficar a configuração deste bloco na figura abaixo.



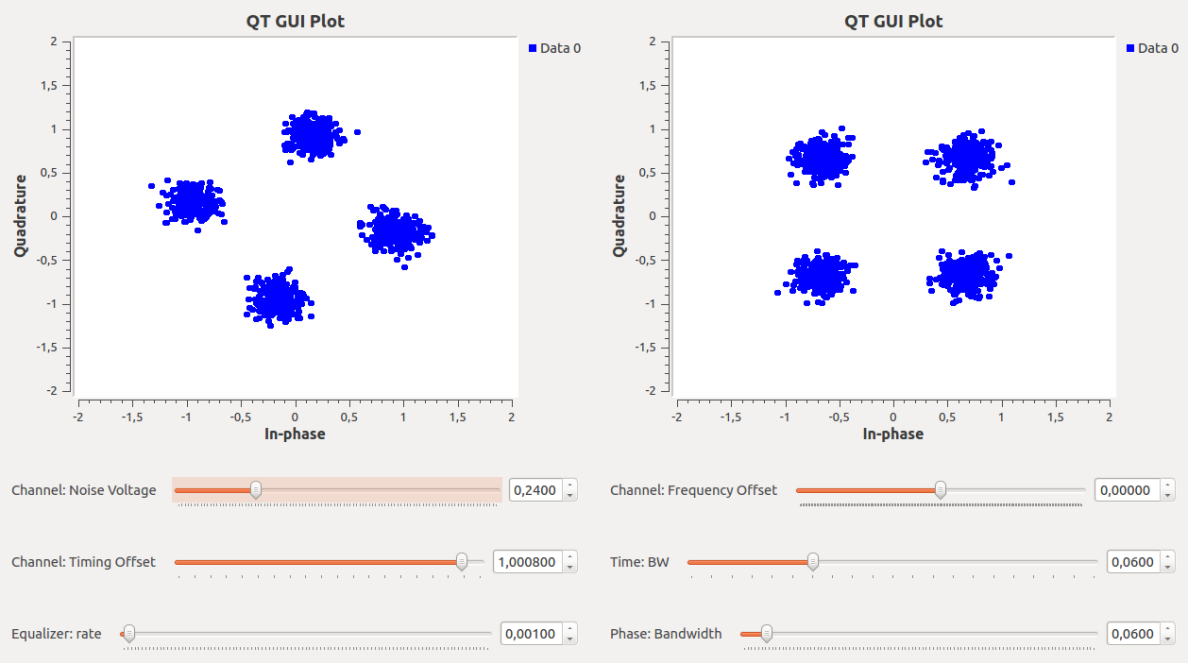
43. Gere e execute o *flowgraph*. A constelação à esquerda da figura abaixo representa a saída do sinal depois do equalizador, e a constelação à direita é depois do uso do algoritmo *Costas Loop*.



44. Agora, com o gráfico em execução, atribua um *offset* de frequência de *0.01* e veja que o ***Costas Loop*** ainda consegue estimar a fase. Já a constelação na saída do ***CMA Equalizer*** entra em rotação, perdendo totalmente o sincronismo.



45. Como já mencionado no item 42, além de conseguir trabalhar com *offset* de frequência, o *Costas Loop* também é robusto a ruído. Para comprovar a teoria, com o gráfico em execução, selecione novamente o *range Frequency Offset* para 0 e aumente o *range* de *Noise Voltage* para uma amplitude de *0.24*. Observe que com um tempo de simulação, a constelação à esquerda entra em rotação e a constelação do *Costas Loop* permanece travada.



46. Vale salientar, que existem outros equalizadores que conseguiria suportar a amplitude de ruído selecionada no item anterior. O ***CMA Equalizer*** é um equalizador “cego”, que não tem um parâmetro de referência (piloto) para comparar com o valor esperado. Assim, se o canal estiver severo, o equalizador obterá um mau desempenho.