Hands-on 4

Loopback transmissão e recepção BPSK utilizando o GNURadio

**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) por símbolo 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.

|  |
| --- |
| PSK  Figura 1: Bits e suas representações no sistema BPSK. |

|  |
| --- |
| const_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. ruído AWGN).Assim, os sistemas de transmissão digitais têm como um dos principais parâmetros para análise de desempenho a taxa de erro (BER), pois um serviço de transmissão digital precisa de um certo patamar de BER 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:

sendo a energia de bit; e 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:**Utilizando os conhecimentos adquiridos nos exercícios passados e por meio de uma breve introdução sobre modulação digital, vamos projetar um *loopback* de um sistema de transmissão BPSK, bem como analisar algumas características do sinal modulado (sinal transmitido) e recebido.

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

|  |
| --- |
|  |

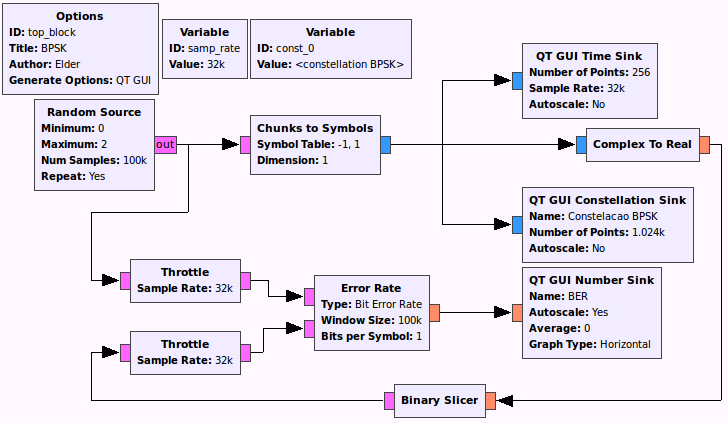
1. Com o GNU RADIO COMPANION aberto, crie um novo projeto

|  |
| --- |
| fig04.png |

1. Clique duas vezes no Bloco ***Options***, esse bloco configura alguns parâmetros gerais do flowgraph. Mantenha o ID como top\_block. Se desejar, digite um título para o projeto e um autor. Selecione *Generate Options* como *QT GUI*, *Run* para *Autostart* e *Real time Scheduling* para *Off*. Então, feche a janela de propriedades. A figura a seguir mostra um exemplo dessa janela de configuração.

|  |
| --- |
| C:\Users\GppCom\Desktop\blocos_bpsk.png |

1. Construa um projeto utilizando os seguintes blocos*:* dois blocos***Variable****,* dois blocos ***Throttle****,* um bloco ***Random Source****,* um bloco ***Chunk to Symbols****,* um bloco ***QT GUI Time Sink***, um bloco ***QT GUI Constellation Sink****,* um bloco ***QT GUI NumberSink****,* um bloco ***Error Rate****,* um bloco ***Complexto Real****,* um bloco ***Binary Slicer***. Conecte os blocos de forma similar ao da figura a seguir.



1. 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 *const\_0* e o campo *Value* para *(digital.constellation\_bpsk())*. Essa última configuração nos faz perceber a utilização de funções de um módulo do GNU Radio (aqui, um módulo específico para BPSK). Você pode conferir como ficará cada bloco nas figuras a seguir.

|  |  |
| --- | --- |
| C:\Users\Cleiton\Desktop\Screenshot from 2014-02-17 09^%02^%20.png | C:\Users\Cleiton\Desktop\Screenshot from 2014-02-17 09^%02^%49.png |

1. Agora no bloco ***Random Source****,* edite os campos: *Output Type* para *Byte; Minimum* para *0, Maximum* para 2*, Num Samples* para *100000* e a opção *Repeat* para *Yes.* Esse bloco gera uma fonte de informação aleatória para análise. Observe a figura abaixo.

|  |
| --- |
| C:\Users\GppCom\Desktop\random sorce.png |

1. Agora passemos a configuração do bloco ***Chunk to Symbols***. Para tanto, abra o bloco e edite os campos *Input Type* para *Byte*; *Output Type* para *Complex*; *Symbol Table* para *const\_0.points()*; *Dimension* para 1; *Num ports* para 1. Essa alteração utiliza o modulo digital do GNU Radio definido anteriormente no bloco *variable* cujo *ID* é *const\_0*, a função *.points()* retorna o mapeamento bit/símbolo da modulação BPSK. A configuração deve estar igual à figura a seguir.

|  |
| --- |
| C:\Users\Cleiton\Desktop\Screenshot from 2014-02-17 09^%08^%34.png |

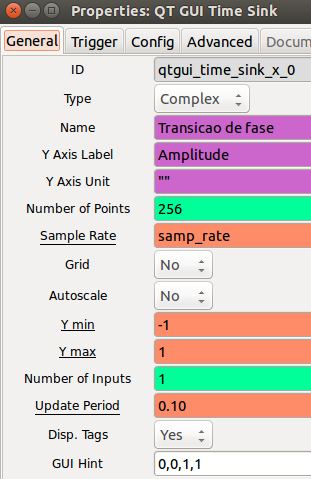
1. Agora clique no bloco ***QT GUI Constellation Sink***, configure o parâmetro *Name* para *Constelacao BPSK* e *Number of Points* para 1024 e *GUI Hint* como 0,1,1,1.

|  |
| --- |
| D:\Dropbox\digital_transmission\short_courses\introduction_to_GNU_and_USRP\New_hands_on\hands_on_4\Prints\qt_constellation.png |

1. A atividade atual irá realizar uma análise de erros de bits que será visualizada durante a simulação através do bloco ***QT GUI Number Sink***, o qual retorna uma saída numérica. Tal bloco pode ser configurado da seguinte forma: *Type* como *Float*; *Name* definido como *BER(%)*; *Min* como *-100*; *Max Value* como *100*. Defina *Number of Inputs* igual a 1 e *Update Period* para 0.10. O bloco deverá ficar similar a figura a seguir.

|  |
| --- |
| C:\Users\GppCom\Desktop\qt number sink.png |

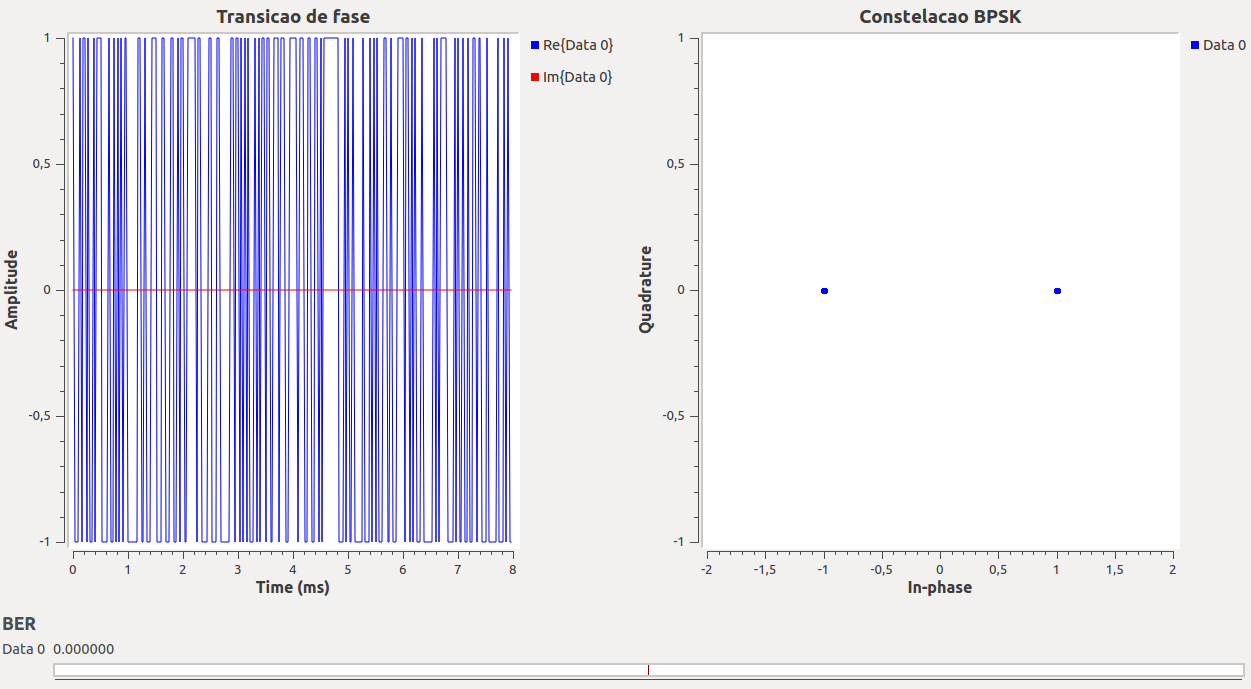
1. Agora vamos editar o bloco ***QT GUI Time Sink***, no campo *Name* edite “*Transicao de fase*”, altere o parâmetro *Number of Points* para 256 e *GUI Hint* para 0,0,1,1.



1. Para a “demodulação” usaremos o bloco ***Binary Slicer***. Ele não precisa de configuração. Se a entrada for *x*, a saída será 0, se x < 0; ou será 1, se x >= 0.
2. Para visualizarmos a variação da Taxa de Erro de Bits (BER) durante a simulação, é preciso comparar os bits gerados com os bits “demodulados”. Isso é feito através do bloco ***Error Rate***. Sua configuração é simples. Defina o campo *Type* como *Bit Error Rate*, *Window Size* como *1000000*, e o campo *Bits per Symbol* com o valor *const\_0.bits\_per\_symbol()*. Confira com a figura a seguir como deverá ficar o bloco. Lembre-se de conectar o bloco ***Error Rate*** através de blocos ***Throttle***, que precisam ter o campo *Type* configurado para *Byte*.

|  |
| --- |
|  |

1. Finalizado esses procedimentos salve e execute o projeto. A figura abaixo apresenta as duas saídas dos blocos ***QT GUI Time Sink*** e ***QT GUI Number Sink*** em um mesmo gráfico. É possível notar ainda que os pontos no diagrama de constelação são quase imperceptíveis, mas estão centrados em -1 e 1. Esse resultado deve-se à ausência de ruído no processamento do sinal. Logo, não há dispersão dos pontos no diagrama.Note também que observamos claramente a transição de fase durante o tempo.



1. Agora vamos adicionar um canal com ruído. Utilizaremos basicamente o projeto anterior com algumas modificações. Salve o projeto com outro nome, exclua o bloco ***QT GUI Time Sink*** e adicione os blocos ***Add****,* ***import****,* ***Noise source*** e ***QT GUI Range****.* Conecte-os de forma similar ao da figura a seguir.

|  |
| --- |
|  |

1. Abra o bloco ***Noise Source*** e mude o campo *Amplitude* para *1.0 / math.sqrt(10\*\*(float(EbN0)/10))*. Esse é o valor da amplitude do ruído para um dado valor de Eb/N0. O valor de Eb/N0 será controlado dinamicamente no bloco ***QT GUI Range***, o qual será configurado posteriormente. Para melhor compreender o valor da amplitudedo ruído será feita sua dedução a seguir e logo depois terminaremos a configuração do bloco ***Noise Source****.*

A Eb/N0 linear (*γ*linear) pode ser calculada a partir da Eb/N0 em dB(*γ*dB) por:

|  |  |
| --- | --- |
|  | (1) |

Sabemos que a potência do ruído branco (de média zero) é igual a sua variância

|  |  |
| --- | --- |
|  | (2) |

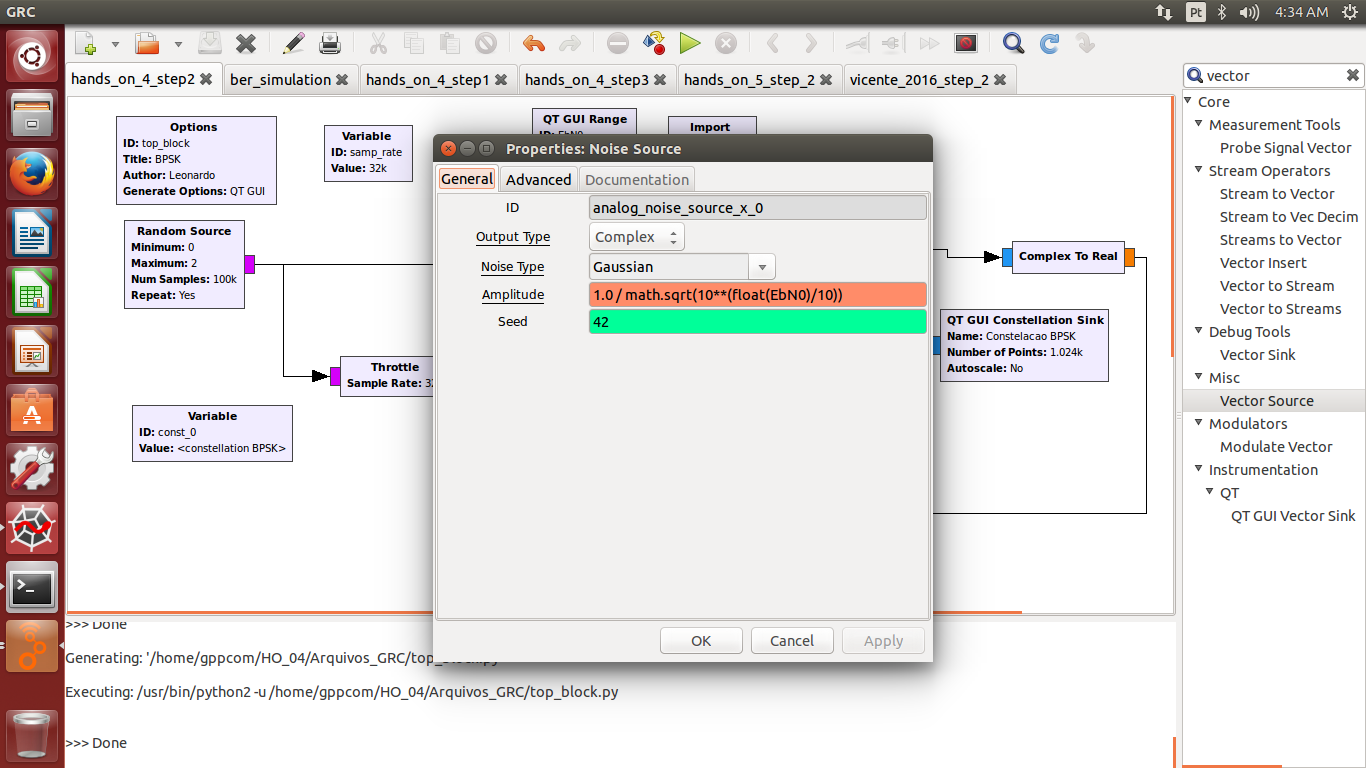
Geralmente, normalizamos a energia do bit *E*b para 1 e mudamos a potência do ruído (*σ*2) para variar o valor da Eb/N0. Assim, a amplitude do ruído (seu desvio padrão) pode ser calculada em função da Eb/N0 como a seguir.

|  |  |
| --- | --- |
|  | (3) |

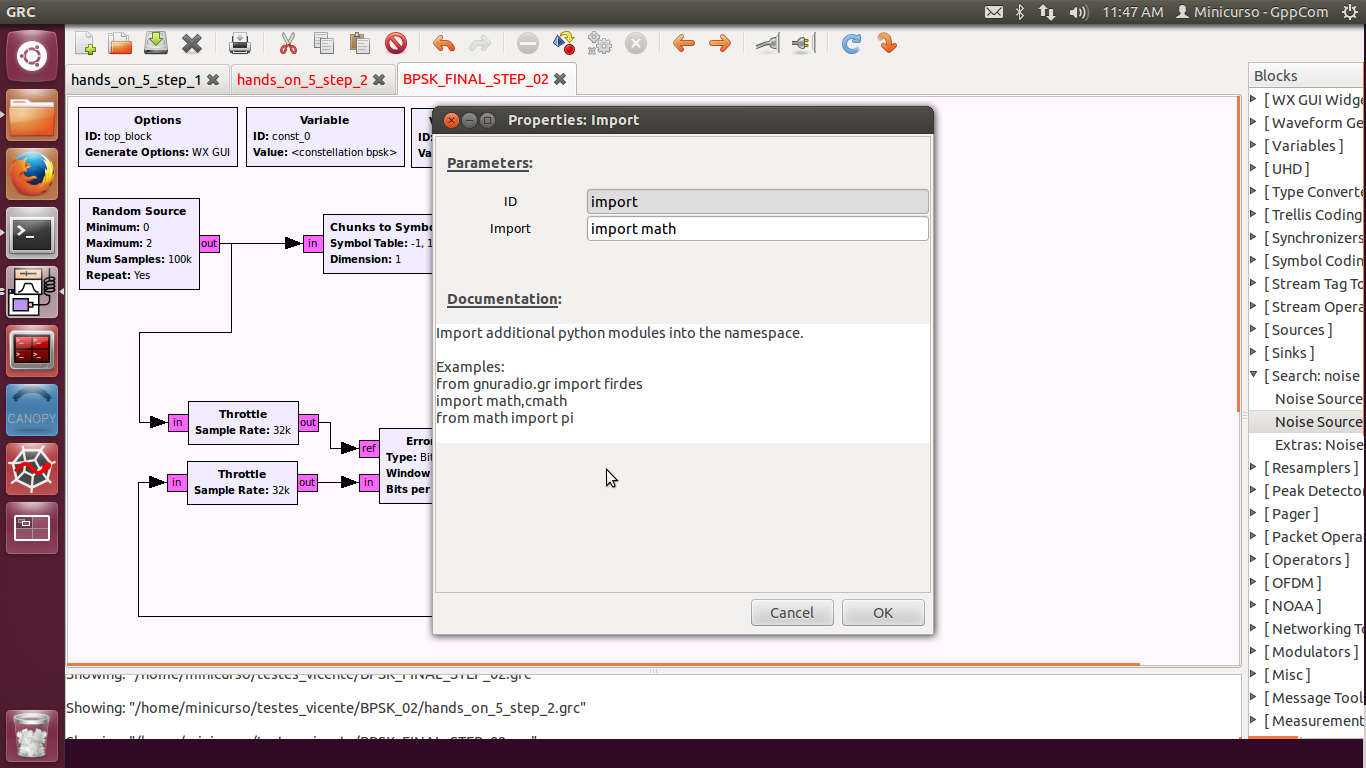
Escrevendo (3) em função da Eb/N0 em dB (*γ*dB), temos



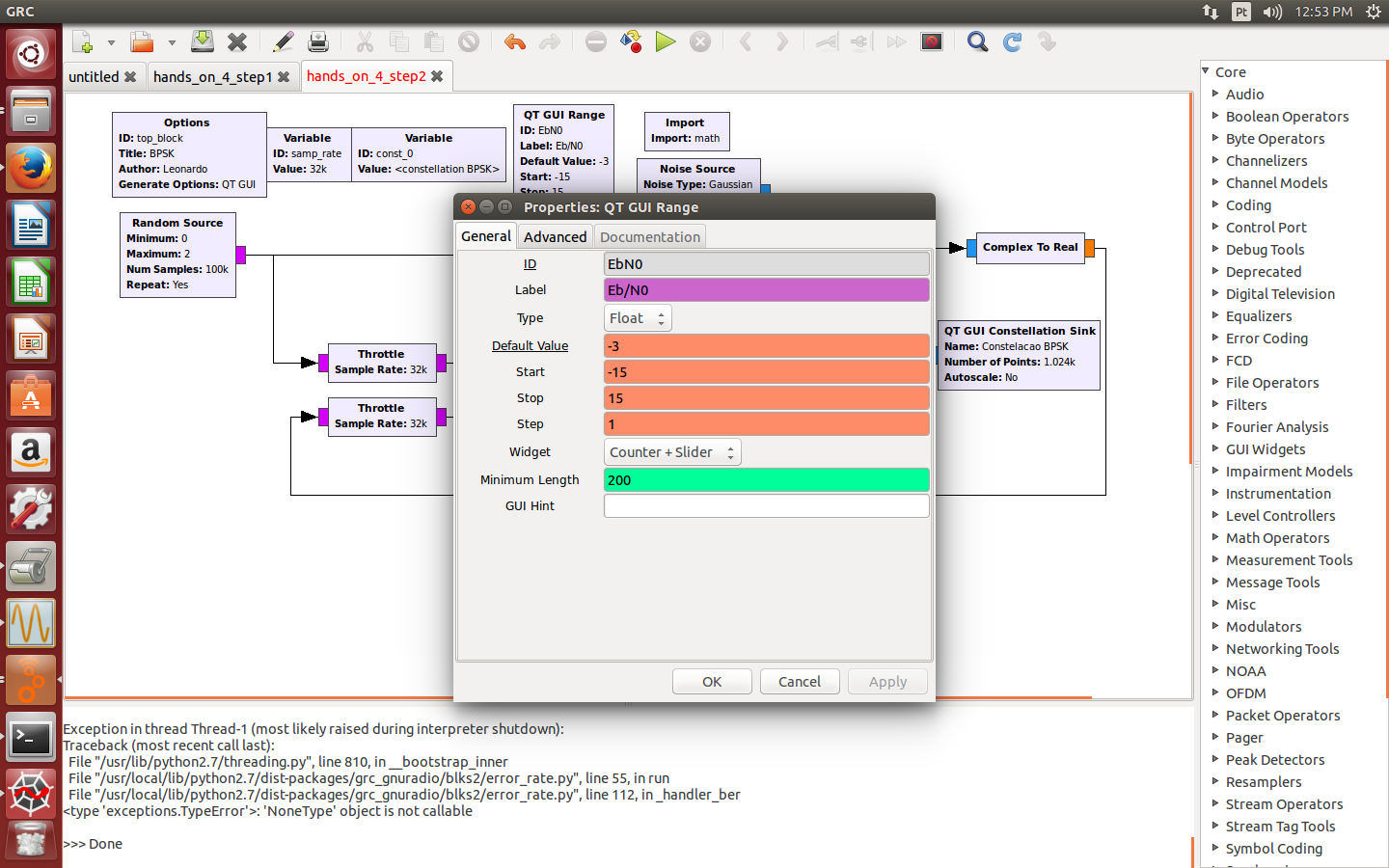
Por isso colocamos o valor *1.0 / math.sqrt(10\*\*(float(EbN0)/10))* no campo *Amplitude* do bloco ***Noise Source*,** pois ele é complexo e o bloco divide o valor de amplitude por dois*.* O bloco deve ficar como na figura a seguir.



1. O bloco ***Noise Source*** ainda ficará vermelho e acusando que não consegue executar o cálculo digitado. Isso devido a falta da biblioteca *math*. Para importá-la precisamos do bloco ***import***. Configure o bloco digitando *import math* no campo *Import*. O bloco deverá ficar como na figura a seguir.

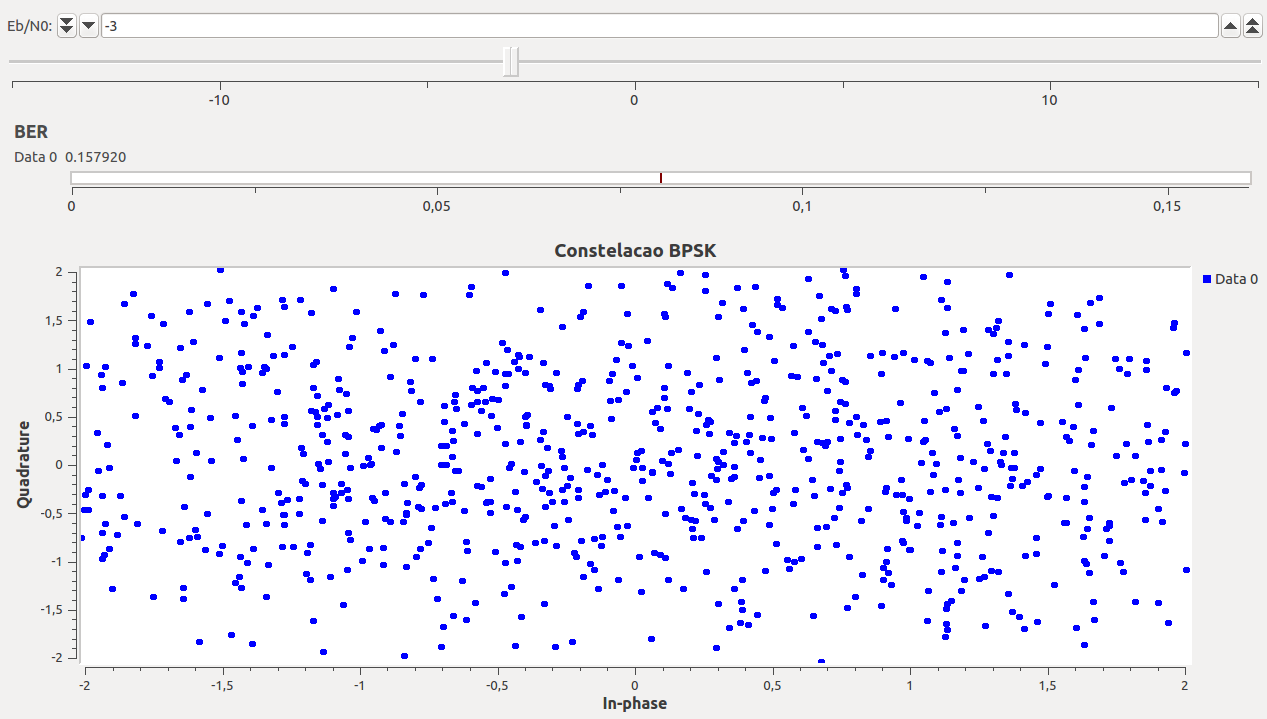


1. Agora vamos configurar o bloco ***QT GUI Range****.* Altere o campo *ID* para EbN0 e o *Label* para *Eb/N0 (dB)*. Altere o valor do campo *Default Value* para -3; *Start* para -15; *Stop* para 15, *Step* para 1 e *Widget* para Counter + Slider. O bloco deve ficar como na figura a seguir.



1. Salve e execute o projeto. Altere o valor do ruído e observe como ficará a constelação. Você perceberá que ao diminuirmos o nível da relação sinal ruído a constelação tende a se espalhar. É necessário aumentar o número de divisões do eixo vertical para visualizarmos o espalhamento de uma forma mais clara. Abaixo será mostrada uma tabela contendo valores teóricos da probabilidade de erro para valores específicos de Eb/N0. Para um número grande de amostras, se espera que o valor da BER convirja para o valor da *P*e. Utilize um dos valores de Eb/N0 da tabela, aguarde um momento e compare o valor da *P*e da tabela com o valor obtido na simulação. Você pode conferir um saída para EB/N0 de - 3dB na figura a seguir.

|  |
| --- |
| **Tabela 1- Alguns valores de BER teórica para valores de Eb/N0 (dB).** |



1. Como último exercício, objetivando coletar os valores de BER obtidos pela simulação do GNU Radio, iremos fazer uso de um procedimento para salvar saídas desejadas. Para tal, vamos utilizar o último projeto realizado e fazer algumas alterações. Inicialmente adicione os blocos ***Head*** e ***File Sink*** à sua área de trabalho. Eles devem ser conectados de acordo com a figura a seguir (na saída do bloco ***Error rate***).

|  |
| --- |
|  |

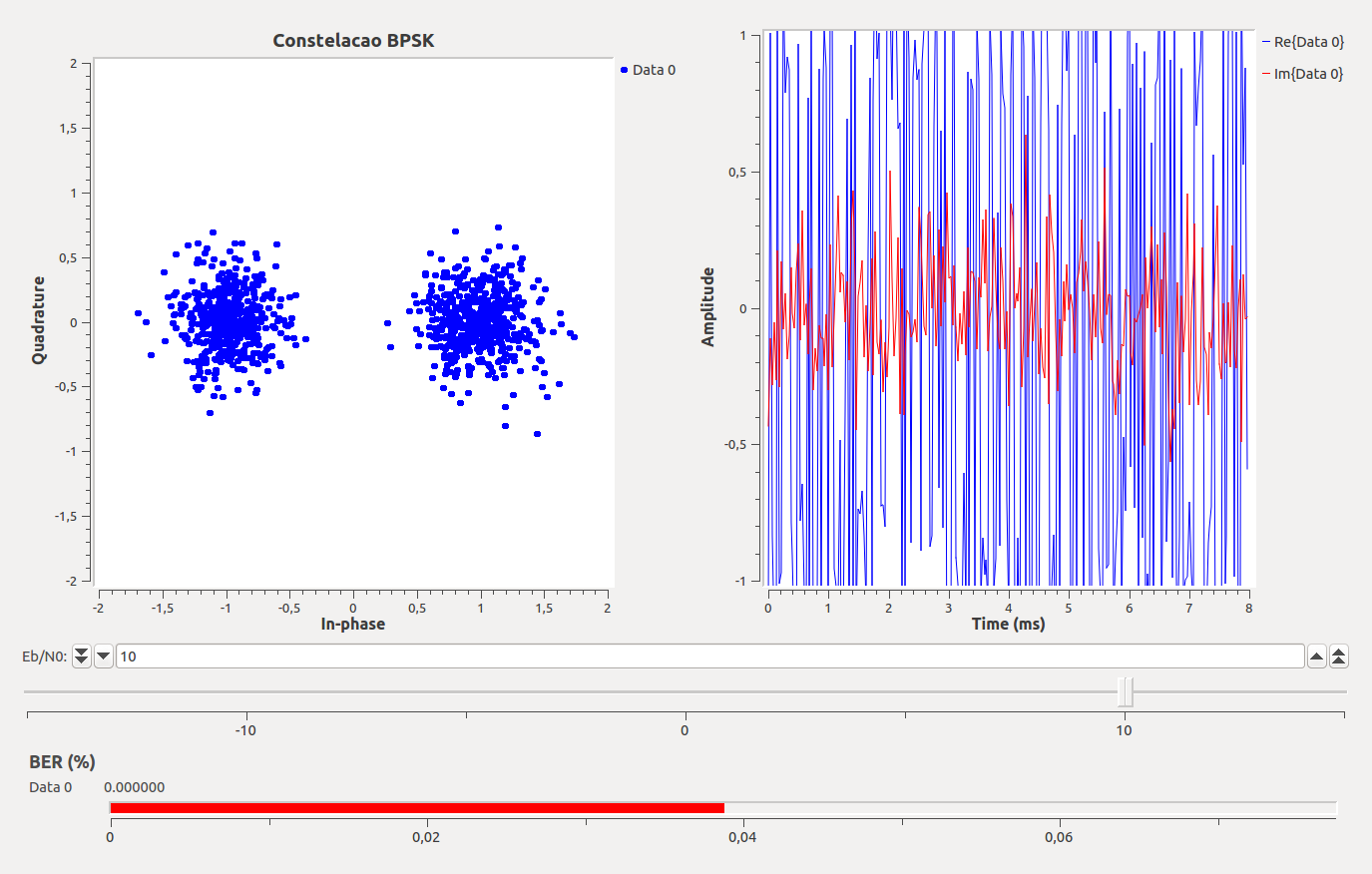
1. O bloco ***Head*** serve para limitar certa quantidade de amostras na saída. No nosso caso, a quantidade de amostras que passará ao bloco ***File Sink***. Agora altere os parâmetros do bloco ***Head*** com os seguintes valores: *Type* para *Float*; *Num items* para *1000000*; *Vec Length* para *1*. Sua configuração deve estar similar a figura a seguir.

|  |
| --- |
|  |

1. No bloco ***File Sink*** selecione o caminho para o salvamento (e.g. ***/home/alunos/Documentos/)*** e nomeie o arquivo como ***BER.bin*** (se essa pasta não existir, por favor, a crie). Feito isso, abra o bloco ***Error rate*** e altere o campo *Window Size* para *100000.* Sua configuração deve estar similar a figura a seguir.

|  |
| --- |
| D:\Dropbox\digital_transmission\short_courses\introduction_to_GNU_and_USRP\New_hands_on\hands_on_4\Prints\file sink.png |

1. Salve e execute o projeto. Escolha um valor de EB/N0 de acordo com a tabela 1 e espere a estabilização da BER. O resultado deve ser semelhante similar a figura a seguir.



1. Aguarde um momento após a estabilização e finalize seu projeto fechando a janela aberta ou clicando no “X” na barra de ações, observe a figura a seguir.

|  |
| --- |
|  |

1. Com o arquivo de dados salvo, podemos abri-lo e observar os dados da simulação. Demonstraremos como abrir este arquivo no ***Python***. Em um terminal (CRTL+ALT+t para abrir), digite os comandos a seguir.

*~$ python*

*>>>filefile=open('/home/gppcom/HO\_04/BER.bin', 'r')*

*>>>import numpy as np*

*>>>Arquivo\_final=np.fromfile(file,dtype=np.float32)*

|  |
| --- |
| D:\Dropbox\digital_transmission\short_courses\introduction_to_GNU_and_USRP\New_hands_on\hands_on_4\Prints\pythonTerminal.png |