**Lab Assignment 1**

**Lab Location**

JEC 5312

**Lab hours**

Wed, Thu, Friday: 11am-1pm, 4pm-6pm

**Part 1 - Introduction**

In this lab, you will use GNU Radio to implement the binary phase shift keying (BPSK) and quaternary phase shift keying (QPSK) modulation schemes. You will learn:

* how BPSK and QPSK works and how to use Maximum Likelihood method to decode;
* how Bit Error Rate (BER) changes when the signal-to-noise ratio (SNR) increases;
* using USRP and Fourier Transformation block to see the spectrum for WiFi.

**GNU Radio:**

GNU Radio is a free & open-source software development toolkit that provides signal processing blocks to implement software-defined radios. Software-Defined radio is a radio system which performs the required signal processing in software instead of using dedicated integrated circuits in hardware. The GNU Radio performs all signal processing operations. Moreover, it is accompanied by a GUI called GNU Radio Companion (GRC) which allows the user to create signal processing applications by drag-and-drop [1]. Each ‘file’ in a GRC is called a *flowgraph.* Each signal processing operation in a flowgraph is performed using a designated ‘block’.

**Starting up GRC:**

You can access the GRC from one of the designated desktops present in the IoT lab with Ubuntu system and login password rpi@123.

Download the following two files to PC.

* BPSK\_BER\_simulation.grc:

<http://homepages.rpi.edu/~wangy52/PersonalWebsite/build/html/_downloads/4cf91d3012d2695bd03d448ecf139d5d/BPSK_BER_simulation.grc>

* QPSK\_BER\_simulation.grc:

<http://homepages.rpi.edu/~wangy52/PersonalWebsite/build/html/_downloads/fb532d6490ef1cf229b64348c2a10f90/QPSK_BER_simulation.grc>

By simply entering the following command in the terminal:

gnuradio-companion &

An untitled GRC window should appear. Click ‘Open’ to access the ‘**BPSK\_BER\_simulation.grc**’ or ‘**QPSK\_BER\_simulation.grc**’ flowgraph to be used in this lab.

**Part 2 - The BPSK and QPSK Modulation Schemes**

BPSK and QPSK are two simple digital modulation schemes. BPSK maps every bit symbol to one output symbol, and QPSK maps every two bits to one output symbol. The mapping relationships are listed in the following table.

|  |  |  |
| --- | --- | --- |
| Modulation | Data Bit(s) | Symbol |
| BPSK | 0 | -1 |
| 1 | 1 |
| QPSK | 00 | -1-j |
| 01 | 1-j |
| 10 | -1+j |
| 11 | 1+j |

*For undergraduate students*, your tasks are:

1. coming up with a decoding method to decode **BPSK** code;
2. implementing your decoding method with Python2.7 in the block named “BPSK Decoder”;
3. finishing the “BER” block to measure the Bit-Error-Rate under certain SNR.
4. run the flowgraph and filling in the table in Appendix A

***For graduate students***, your tasks are:

1. coming up with a decoding method to decode **QPSK** code;
2. implementing your decoding method with Python2.7 in the block named “**QPSK** Decoder”;
3. finishing the “BER” block to measure the Bit-Error-Rate under certain SNR.
4. run the flowgraph and filling in the table in Appendix B

Hints:

1. The input to the BPSK/QPSK Decoder block is of data type numpy.array(numpy.complex). You may need to deal with the real and imaginary part of each number;
2. For BPSK, BER is equivalent to Symbol-Error-Rate. You can simply check how many decoded symbols are different from the ground truth symbols. For QPSK, you have to do a bit-wise comparison. For example, 2 and 3 are different in symbols, but their corresponding binary bits differ in only one bit. Similarily, 1 and 4 differ in two bits.
3. To make the BER measurement more precise, we adopt the idea of *sliding window*. I create a member variable named self.window in the block with length 1,000,000. The self.window always stores the most recent 1,000,000 comparison results. Therefore, BER is computed out from relatively large amount of data.

For example, if the block stores 200 results in self.window[0:199], when the block needs to store another 300 results, the corresponding location is self.window[200: 499].

**Details of the flowgraph:**





The flow graph uses a ‘Random Source’ block to generate the original signal. The ‘Chunks to Symbols’ block maps the incoming data to a symbol according to the following mapping table:

The ‘Noise Source’ introduces the Gaussian noise of a predetermined amplitude to the original signal. This source models the process of a signal becoming noisy as it is transmitted through a channel.

The ‘BPSK Decoder’ or ‘QPSK Decoder’ decodes the incoming BPSK-modulated signal to bytes. This stage represents the receiver. The block named “BER” is used to compute the Bit-Error-Rate. It takes in two inputs: ***the original signal*** and ***the decoded signal***. Its purpose is to determine the number of bit-errors introduced by the noisy channel within a fixed window size.

Run the flowgraph by clicking on the ‘Run’ button on the toolbar. This will open a new window displaying the signal constellation of the noisy signal and the BER. You should also be able to increase or decrease the SNR value by moving the bar to the left or to the right.



***Refer to Part 4 - Question 4 for recording relevant data.***

**Part 4 - Deliverables:**

1. **Describe and explain** the effect on the BPSK (for undergrad) or QPSK (for graduate) signal constellations when the SNR value is increased or decreased.
2. Simply **describe** your decoding method.
3. **Record the BER readings** using tables in the Appendix (correct to 3 significant figures) for the SNR values given in the table. Describe and explain the observed trend.
4. **Generate plots from the readings you collect**. Use the ‘*semilogy*’ command if you are using MATLAB or python.matplotlib to plot the data,.
5. Determine the SNR values beyond which the symbols can be decoded with great accuracy (SER < 0.0001) for both BPSK and QPSK schemes.
6. (for graduate students) Should BER vs SNR curve for BPSK and QPSK the same? Why or why not?

**Appendix A**

|  |  |
| --- | --- |
| **SNR (dB)** | **SER for BPSK** |
| -15 |  |
| -12 |  |
| -9 |  |
| -6 |  |
| -3 |  |
| -2 |  |
| -1 |  |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

**Appendix B**

|  |  |
| --- | --- |
| **SNR (dB)** | **SER for QPSK** |
| -15 |  |
| -12 |  |
| -9 |  |
| -6 |  |
| -3 |  |
| -2 |  |
| -1 |  |
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

The documentation for GNU Radio may be found in [2].

If you have any questions, please turn to or email one of the lab TAs:

**· Syed Ahsan Raza Naqvi (naqvis2@rpi.edu)**

**· Yu Wang (****wangy52@rpi.edu****)**

[1] <https://wiki.gnuradio.org/index.php/What_is_GNU_Radio%3F>

[2] <https://www.gnuradio.org/doc/doxygen/>