

## II. LAB

**Software Required:** NI LabVIEW 2012, NI LabVIEW 4.3 Modulation Toolkit.

**Functions and VI (Virtual Instrument) from the LabVIEW software to be used in this lab:** Case Structure (Function), Decimate 1D Array (Function), Multiply (Function), Add (Function), Quotient & Remainder (Function), Numeric Constant (Function), For Loop (Function), Index Array (Function), Symbol energy (VI), Subtract (Function), Complex To Polar (Function), Max & Min (Function), Array Constant (Function), DBL Numeric Constant (Function), and Interleave 1D Arrays (Function).

### PART-1: MODULATOR

**INPUTS:** Modulation type (string), input bit stream (1-D array of unsigned byte [8-bit integer (0 to 255)]), and symbol energy (1) (double [64-bit real (~15 digit precision)]).

**OUTPUTS:** Output symbols (1-D array of double complex [~ 15 digits precision]).

A virtual instrument template (“student\_modulation.vit”) is provided to you with these inputs and outputs. The template should be populated according to instructions. Then, newly created VIs are plugged in the simulator and transmitter USRP control files.

### PROCEDURE

In this part, we develop a LabVIEW virtual instrument as a baseband digital transmitter that can use either Binary Phase Shift Keying (BPSK) or Quadrature Phase Shift Keying (QPSK) modulation to transmit a modulated carrier  $\cos(2\pi f_c t)$  at frequency  $f_c$  (e.g., 2.47 GHz) using USRP N210 as the transmitting hardware. This corresponds to sending a random bit sequence or a predefined file. Specifically, the VI generates a random baseband signal. This signal is up-converted and transmitted.

We now detail the construction of the VI at hand in Lab View:

#### a) Setup

\* Open the LabVIEW program (Start > All Programs > National Instruments > LabVIEW 2012 > LabVIEW 2012).

\* Open “student\_modulation.vit” file (Open Existing (OR Ctrl+O) > Select “student\_modulation.vit” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

- \* Enable Functions Palette (View > Functions Palette or right click on anywhere of the block diagram window).
- \* Place a *Case Structure* box (Functions Palette > Programming > Structures > select and drag Case Structure).
- \* Move “modulation type”, “input bit stream”, “output symbols”, and “Symbol Energy (1)” inside the *Case Structure*.
- \* Connect “modulation type” output to “case selector” node on *Case Structure* box.

### **b) Modulations**

In this section, main modulation type selection cases are formed.

- \* Place a *Case Structure* box (Functions Palette > Programming > Structures > select and drag Case Structure) inside the previous *Case Structure* box. This *Case Structure* is used as the foundation of modulation selection and converts the input bit stream into decimal numbers by using the algorithm defined under each modulation case.
- \* Click on the arrow on *Case Structure* box and select “false”. Double click on the text and change it to “QPSK”. This case is used to insert the algorithm for QPSK modulation.
- \* Repeat the same step to change “true” to “BPSK”. This case is used to insert the algorithm for BPSK modulation.
- \* When BPSK is selected, right click on the name and select “Make This the Default Case”. After this step, case label reads - “BPSK”, Default -. This step makes BPSK our default modulation scheme.

### **c) BPSK mapping to levels**

In this section, incoming bits are mapped into signal levels when BPSK modulation is selected. This process is explained in section I. Overview, under “The mapping procedure for BPSK”.

While BPSK is selected:

- \* Place a *Quotient & Remainder* function and connect the input bit stream to its x input.
- \* Create a *Numeric Constant* and connect it to the y input of *Quotient & Remainder*. Set the constant value to 2.

### **d) QPSK mapping to levels**

In this section, incoming bits are mapped into signal levels when QPSK modulation is selected. This process is explained in section I. Overview, under “The mapping procedure for QPSK”.

While QPSK is selected:

- \* Place a *Decimate 1D Array* inside the *Case Structure* box (Functions Palette > Programming > Array > Decimate 1D Array). This function divides the elements of the input bit stream into two output arrays (even array elements (0, 2, 4,...) in one output and odd array elements (1, 3, 5,...) in another output).
- \* Connect the input bit stream to array input of *Decimate 1D Array*.
- \* Place a *Multiply* function inside the *Case Structure* box (Functions Palette > Programming > Numeric > Multiply).
- \* Create a *Numeric Constant* (Functions Palette > Programming > Numeric > Numeric Constant) and connect it to the x input of the multiplier. Set the constant value to 2.
- \* Connect the y input of multiplier to the “elements 0, n, 2n,...” output of *Decimate 1D Array*.
- \* Place an *Add* function (Functions Palette > Programming > Numeric > Add).
- \* Connect the x input of *Adder* to  $x*y$  output of multiplier.
- \* Connect the y input of *Adder* to the “elements 1, n+1, 2n+1,...” output of *Decimate 1D Array*.
- \* Place a *Quotient & Remainder* function and connect its x input to the output of *Multiply* (Functions Palette > Programming > Numeric > Quotient & Remainder).
- \* Connect the x input of *Quotient & Remainder* to  $x+y$  output of *Adder*.
- \* Create a *Numeric Constant* and connect it to the y input of *Quotient & Remainder*. Set the constant value to 4.

The process accomplished with above steps is explained in section I. Overview, under “The mapping procedure for QPSK”.

### e) Sequences and modulation setup

In this section, above mapping processes are repeated as the length of the incoming bit stream.

\*Create a *For Loop* inside the bigger *Case Structure* but outside the modulation *Case Structure* (Functions Palette > Structure > For Loop).

- \* Place a *Case Structure* inside this *For Loop*. This allows virtual instrument to map all elements of the input bit stream into an analog signal level.
- \* Connect “modulation type” output to “case selector” node on *Case Structure* box as we did for the previous *Case Structure* box. This step ensures that both *Case Structures* operate on the same modulation setting once the type is selected by the operator on simulation/hardware control front panel.
- \* Click on the arrow on *Case Structure* box and select “false”. Double click on the text and change it to “QPSK”.
- \* Repeat the same step to change “true” to “BPSK” and make this the default case as you did earlier for the first *Case Structure*.

#### **f) Mapping to BPSK constellation**

In this section, signal levels are mapped into constellation points when BPSK modulation is selected.

While BPSK is selected:

- \* Place an *Index Array* inside the *For Loop* but outside the *Case Structure* box (Functions Palette > Programming > Array > Index Array).
- \* Right click of the “array” input of this *Index Array* and create a constant (Right click menu > Create > Constant), then move this constant inside the *Case Structure* box.
- \* Right click on this constant and deselect “index display” (Right click menu > Visible items > Index display).
- \* Make this constant display two bits by simply (carefully) extending from its sides. Replace the 0|0 with 1|-1 and connect it to the “array” input of *Index Array*.
- \* Change the first *Case Structure* box to “BPSK, Default” and connect the “R” output of *Quotient & Remainder* to the “index” input of the *Index Array* in the *For Loop*.
- \* Select the two bits (1|-1) you placed under BPSK, copy them (Ctrl + C), go to QPSK case of the second *Case Structure*, and paste the copied constant there (Ctrl + V).

#### **g) Mapping to QPSK constellation**

In this section, signal levels are mapped into constellation points when QPSK modulation is selected.

While QPSK is selected:

- \* Right click on the constant and change its representation to CDB (Right click menu > Representation > CDB). This displays the constant value as complex double.
- \* Make it four bits by stretching from sides and change the values as follows (with the same order from left to right):  $1+1i$  |  $1-1i$  |  $-1+1i$  |  $-1-1i$
- \* Connect this constant to “array” input of the *Index Array*.
- \* Change both *Case Structures* to “QPSK” and connect the “R” output of *Quotient & Remainder* to the “index” input of the *Index Array* in the *For Loop*.
- \* Place the “symbol\_energy.vi” outside the *For Loop* (Function Palette > Select a VI > digital\_comm.llb (under .../labs – student/ lab 2.1) > “symbol\_energy.vi”). Click on “OK” if a warning pops up.
- \* Connect “element” output of the *Index Array* to “input symbols” input of symbol\_energy.vi.
- \* Connect “output symbols” output of the symbol\_energy.vi to the input of output symbols element.
- \* Connect the output of Symbol Energy (1) element to “symbol energy (1)” input of the “symbol\_energy.vi”

#### **h) Necessary connections**

In this section, necessary error in – error out connections are made in order to finalize VI design.

- \* Connect the error in (no error) element to “case selector” of the big *Case Structure* box (“True” changes into “No error” and box color becomes light green). Connect this case selector to the error out element.
- \* Change the case to “Error” and connect the error in (no error) element to the error out element through the *Case Structure* box.

At the end, your block diagram should look like the one below in Fig 1 and your front panel should look like the one in Fig 2:

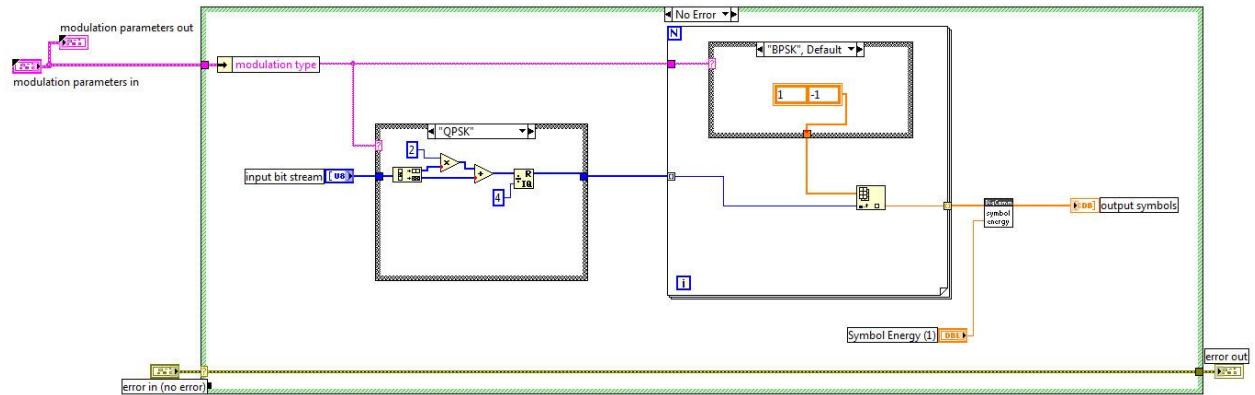


Fig 1 – Complete block diagram

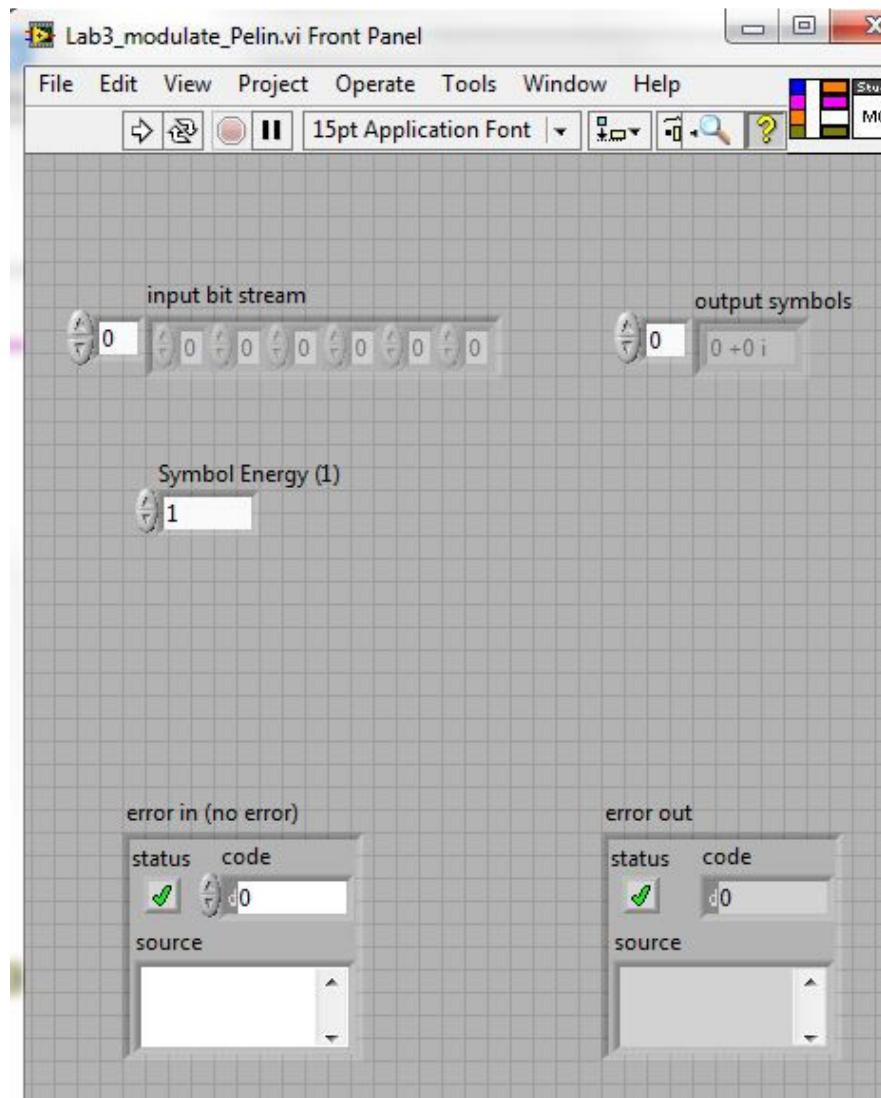


Fig 2 – Complete front panel

For visual reference, check this video out (note to Gil: ***please add your video here***): [See the video: How to build Lab3\\_modulate\\_student.vi](#)

\* In order to check if your file has errors click on Operate > Run (OR Ctrl+R). If no error window is generated, then your file is good.

\* SAVE it AS a new file by adding your name at the end (example: if your name is John, then the new file name is “Lab3\_modulate\_John.vi”).

## **PART-2: DECODER**

**INPUTS:** Modulation type (string) and input symbols (1-D array of double complex [ $\sim$  15 digits precision]).

**OUTPUTS:** Decoded bit stream (1-D array of unsigned byte [8-bit integer (0 to 255)]).

A virtual instrument template (“student\_decode.vit”) is provided to you with these inputs and outputs. The template should be populated according to instructions. Then, newly created VIs are plugged in to the simulator and transmitter USRP control files.

## **PROCEDURE**

In this part, we develop a LabVIEW virtual instrument as a receiver that can capture the transmitted analog signal using USRP N210 and decode/demodulate it based on the known modulation type information. This corresponds to sending a random bit sequence or a predefined file. Specifically, the analog signal is received, down-converted to baseband and demodulated by the URSP and the VI translates these signal levels into binary bit sequences and stores them into decoded bit stream output.

We now detail the construction of the VI at hand in Lab View:

### **a) Setup**

\* Open the LabVIEW program (Start > All Programs > National Instruments > LabVIEW 2012 > LabVIEW 2012)

\* Open “student\_decode.vit” file (Open Existing (OR Ctrl+O) > Select “student\_decode.vit” > OK). A basic graphic user interface (GUI) opens.

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

\* Enable Functions Palette (View > Functions Palette).

\* Place a *Case Structure* box (Functions Palette > Programming > Structures > select and drag

Case Structure).

- \* Move “modulation type”, “input symbols”, and “decoded bit stream” inside the *Case Structure*.

### b) Modulations

In this section, main modulation type selection cases are formed.

- \* Place another *Case Structure* box inside the first *Case Structure*. We use this *Case Structure* box for modulation type definitions.
- \* Change the “True” of the second *Case Structure* box to “QPSK” and “False” to “BPSK”. Then, make BPSK the default case.
- \* Connect “modulation type” output to “case selector” node on *Case Structure* box.

### c) Decoding constellation to BPSK levels

In this section, received constellation points are decoded into signal levels when BPSK modulation is selected.

While BPSK is selected:

- \* Place a *For Loop* inside the second *Case Structure*.
- \* Add a *Subtract* function (Functions Palette > Programming > Numeric > Subtract).
- \* Place a *Complex To Polar* function inside the *For Loop* (Functions Palette > Programming > Numeric > Complex > Complex to Polar). This function breaks a complex number into its polar components (i.e., magnitude  $r$  and angle  $\theta$ ).
- \* Place a *Max & Min* function inside the *For Loop* (Functions Palette > Programming > Array > Max & Min). This function returns maximum and minimum values of the array at its input along with their indexes.
- \* Place an *Array Constant* function inside the *For Loop* (Functions Palette > Programming > Array > Array Constant). This function supplies a constant array value to the block diagram.
- \* Place a *DBL Numeric Constant* inside the big *Array Constant* box (Functions Palette > Programming > Numeric > DBL Numeric Constant). This function is used to pass a double-precision, floating-point numeric value to the block diagram.

- \* Right click on this constant and deselect “index display” (Right click menu > Visible items > Index display). Then change its representation to CDB.
- \* Make this constant display two bits by simply (carefully) extending from its sides. Replace the  $0+0i|0+0i$  with  $1+0i|-1+0i$  and connect it to the x input of *Subtract* function.
- \* Connect the “input symbols” to the y input of the *Subtract* function.
- \* Connect the x-y output of *Subtract* function to the z input of *Complex To Polar* function.
- \* Connect the r output of *Complex To Polar* function to array input of array *Max & Min* function.
- \* Connect the min index(es) output of array *Max & Min* function to decoded bit stream. This output gives the index for the first minimum value.

#### d) Decoding constellation to QPSK levels

In this section, received constellation points are decoded into signal levels when QPSK modulation is selected.

While QPSK is selected:

- \* Place a *For Loop* inside the second *Case Structure* (like you did for BPSK case).
- \* Copy the two bits ( $1+0i|-1+0i$ ) you placed under BPSK case (Ctrl + C), go to QPSK case of the second *Case Structure*, and paste the copied constant there (Ctrl + V).
- \* Copy *Subtract*, *Complex To Polar*, and array *Max & Min* functions (Hint: You can copy several multiple elements at once by holding the <Shift> key while selecting them.) and paste them under the QPSK case.
- \* Make the constant four bits by stretching from sides and change the values as follows (with the same order from left to right):  $1+1i | 1-1i | -1+1i | -1-1i$  and connect it to the x input of *Subtract* function.
- \* Connect the “input symbols” to the y input of the *Subtract* function.
- \* Connect the x-y output of *Subtract* function to the z input of *Complex To Polar* function.
- \* Connect the r output of *Complex To Polar* function to array input of array *Max & Min* function.

- \* Place a *Quotient & Remainder* function outside the *For Loop* but inside the second *Case Structure*.
- \* Place an *Interleave 1D Arrays* function next to the *Quotient & Remainder* function (Functions Palette > Programming > Array > Interleave 1D Arrays). This function interleaves corresponding elements from the input arrays into a single output array. For example: If your first input array is [1 2 3 4] and the second input array is [5 6 7] then the interleaved output array becomes [1 5 2 6 3 7]. As you would notice, last element of the first array is not used since the shortest array defines the size of the interleaved 1D output array.
- \* Connect the min index(es) output of array *Max & Min* function to the x input of *Quotient & Remainder* function.
- \* Connect a *Numeric Constant* to the y input of *Quotient & Remainder* function and set its value to 2.
- \* Connect the R output of *Quotient & Remainder* function to the array 1 input of *Interleave 1D Arrays* function.
- \* Connect the IQ output of *Quotient & Remainder* function to the array 0 input of *Interleave 1D Arrays* function.
- \* Connect the interleaved array output of *Interleave 1D Arrays* function to decoded bit stream.

#### e) Detection

In this section, Maximum Likelihood detection algorithm is implemented.

- \* The detection method used here is called Maximum Likelihood (ML) detection. Briefly the detection algorithm we built in this VI works as follows:

- A complex double symbol from “input symbols” enters the *For Loop* and is subtracted from each element of the constant (2-bit for BPSK or 4-bit for QPSK). Result is processed and separated into its magnitude and phase components by *Complex To Polar* function.
- Next, array *Max & Min* function selects the index of the array element which resulted in minimum difference as this is most likely the wanted symbol.
- Quotient & Remainder* function is used only for QPSK case as the index can be 0, 1, 2, or 3 but the algorithm calls for 0 or 1.
- Lastly, the index is directly used in BPSK and *Quotient & Remainder* output is used after going through the *Interleave 1D Arrays* function in QPSK as the decoded bit stream.

#### f) Necessary connections

In this section, necessary error in – error out connections are made in order to finalize VI design.

- \* Connect the error in (no error) element to “case selector” of the big *Case Structure* box (“True” changes into “No error” and box color becomes light green). Connect this case selector to the error out element.
- \* Change the case to “Error” and connect the error in (no error) element to the error out element through the *Case Structure* box.

At the end, your block diagram should look like the one below in Fig 3 and your front panel should look like the one in Fig 4:

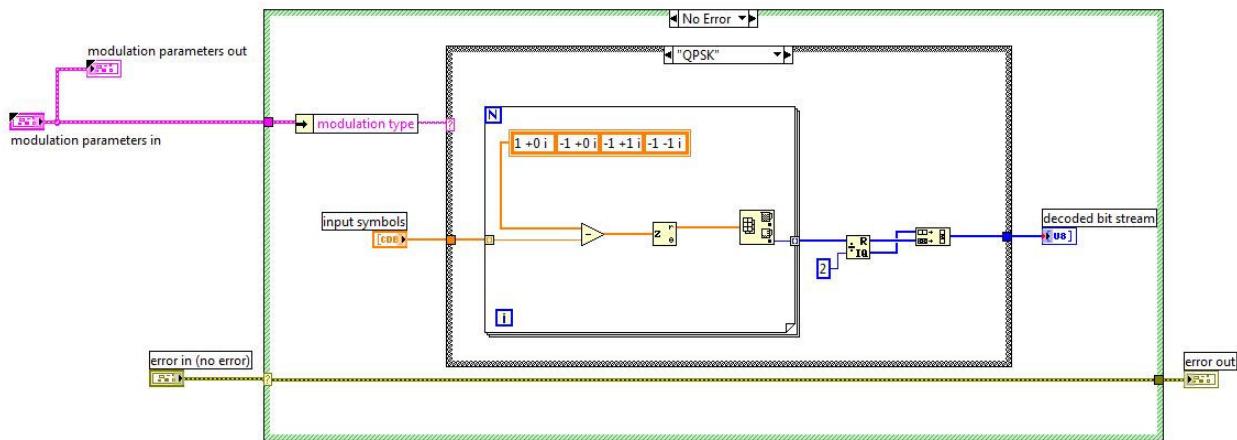


Fig 3 – Complete block diagram

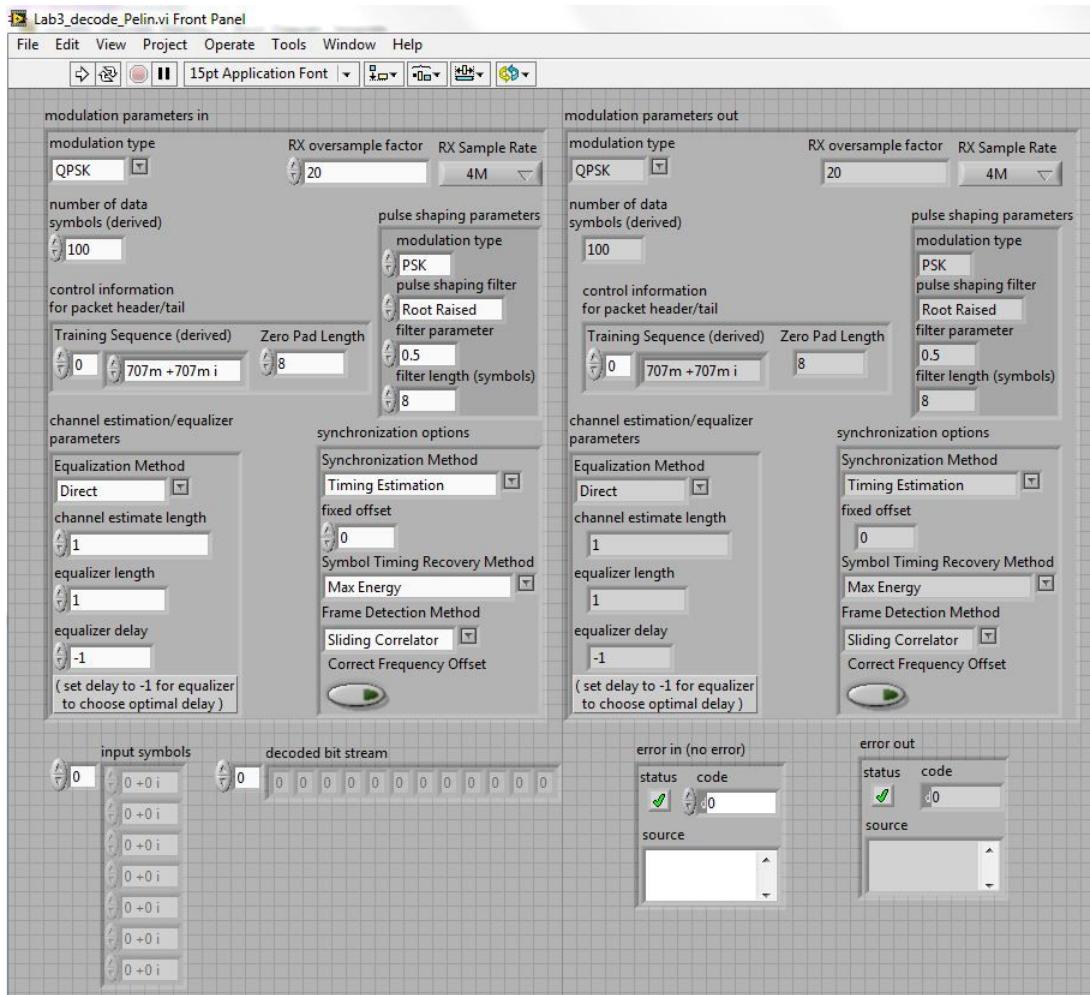


Fig 4 – Complete front panel

For visual reference, check this video out (note to Gil: ***please add your video here***): [See the video: How to build a decoder.vi](#)

\* In order to check if your file has errors click on Operate > Run (OR Ctrl+R). If no error window is generated, then your file is good.

\* SAVE it AS a new file by adding your name at the end (example: if your name is John, then the new file name is “Lab3\_decode\_John.vi”).

### **PART-3: SIMULATION**

In this part we use the provided transmitter and receiver templates to plug in our VI files and test their operation before using them with the USRP boards.

In order to simulate your modulator and decoder designs, please follow these steps:

\* Open the LabVIEW program (Start > All Programs > National Instruments > LabVIEW 2012 > LabVIEW 2012).

### **a) Transmitter setup**

\* Open “transmitter.vi” file (Open Existing (OR Ctrl+O) > Select “transmitter.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

\* Locate the “source.vi (source) in the block diagram and replace it with the source file you created in Lab 1 “student\_source\_<name>.vi” file (Right click on the “source” box > Replace > All Palettes > Select a VI...>insert your “student\_source\_<name>.vi” file).

\* Locate the “modulate.vi (MOD) in the block diagram and replace it with your newly created “student\_modulate\_<name>.vi” file (Right click on the “MOD” box > Replace > All Palettes > Select a VI...>insert your “student\_modulate\_<name>.vi” file).

### **b) Receiver setup**

\* Open “receiver.vi” file (Open Existing (OR Ctrl+O) > Select “receiver.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

\* Locate the “error\_detect.vi (error detect) in the block diagram and replace it with the source file you created in Lab 1 “student\_error\_detect\_<name>.vi” file (Right click on the “error detect” box > Replace > All Palettes > Select a VI...>insert your “student\_error\_detect\_<name>.vi” file).

\* Locate the “decode.vi in the block diagram and replace it with your newly created “student\_decode\_<name>.vi” file (Right click on the “decode” box > Replace > All Palettes > Select a VI...>insert your “student\_decode\_<name>.vi” file).

### **c) Simulator setup**

\* Open “simulator.vi” file (Open Existing (OR Ctrl+O) > Select “simulator.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Click on “Run Continuously” button (two looping arrows).

### **d) Data collection**

Bring up the simulator front panel and perform the following measurements shown on Fig 5, 6, and 7. Detailed information on front panel objects can be found in Appendix-A of this document.

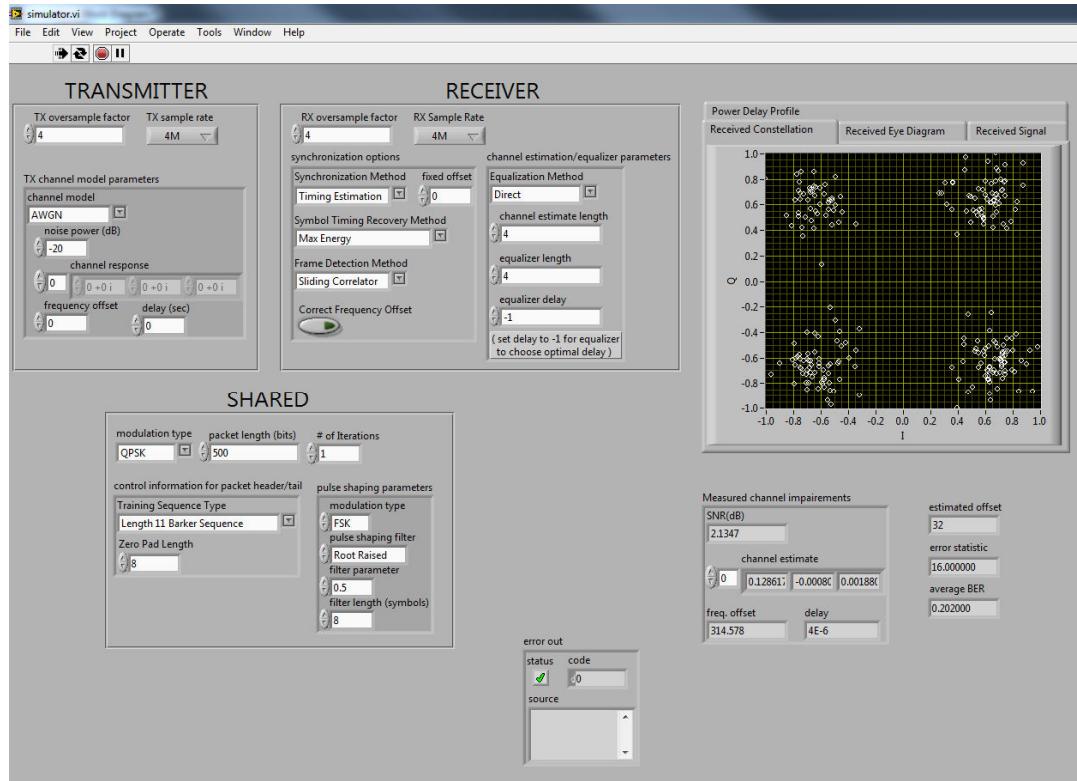


Fig 5 – Simulator front panel – received constellation view

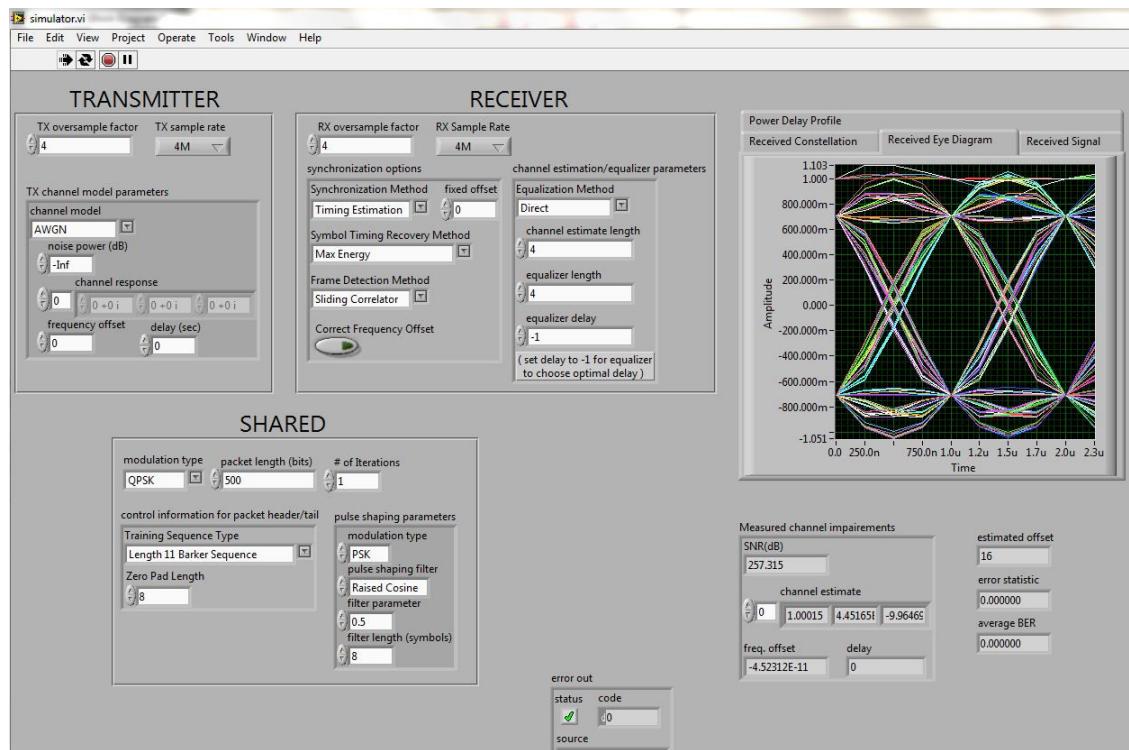


Fig 6 – Simulator front panel – eye diagram view with ideal noise power setting (-infinity)

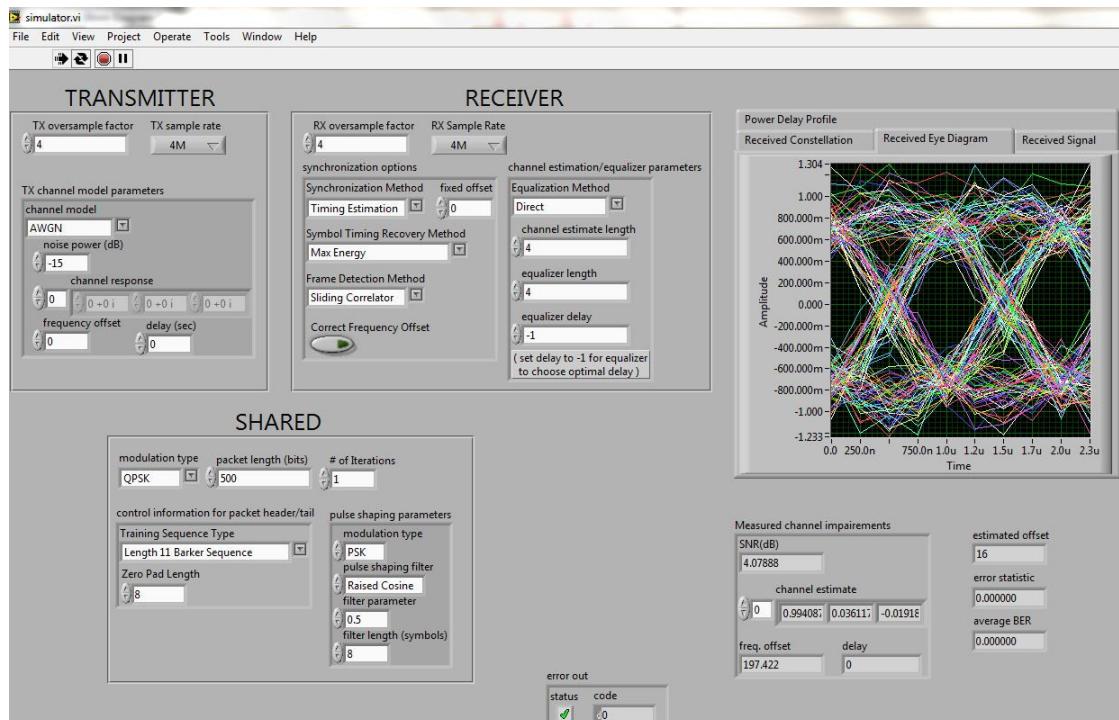


Fig 7 – Simulator front panel – eye diagram view with more realistic noise power setting (-15 dB)

Repeat the simulation using following noise power settings “-INF”, “-20”, and “0” dB for both BPSK and QPSK and collect screenshots of both constellation and eye diagram views. They should be used for report preparation.

#### **PART-4: USRP OPERATION**

In this part we use the provided top\_tx and top\_rx templates to plug in our VI files and use them with the USRP boards.

In order to operate using USRPs, please follow these steps:

\* Open the LabVIEW program (Start > All Programs > National Instruments > LabVIEW 2012 > LabVIEW 2012).

##### **a) Transmitter setup**

\* Open “transmitter.vi” file (Open Existing (OR Ctrl+O) > Select “transmitter.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

\* Locate the “source.vi (source) in the block diagram and replace it with the source file you created in Lab 1 “student\_source\_<name>.vi” file (Right click on the “source” box > Replace > All Palettes > Select a VI...>insert your “student\_source\_<name>.vi” file).

\* Locate the “modulate.vi (MOD) in the block diagram and replace it with your newly created “student\_modulate\_<name>.vi” file (Right click on the “MOD” box > Replace > All Palettes > Select a VI...>insert your “student\_modulate\_<name>.vi” file).

\* Open “top\_tx.vi” file (Open Existing (OR Ctrl+O) > Select “top\_tx.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

\* Set the following values in Table 1 on top\_tx front panel, HW parameters tab:

LABEL	VALUE
USRP IP Address <IP> from NI-USRP configuration utility	
Active antenna	TX1
Carrier frequency (in HZ)	2G
Gain (in dB)	10
Generation Mode	Continuous

Table 1 – Top\_tx HW parameters settings

\* Go to “Modulation parameters” tab of the same front panel, change the TX sample rate value to “2M” and packet length (bits) to “100”.

\* Click on “Run Continuously” button (two looping arrows).

### b) Receiver setup

On another computer with a connected USRP:

\* Open the LabVIEW program (Start > All Programs > National Instruments > LabVIEW 2012 > LabVIEW 2012)

\* Open “receiver.vi” file (Open Existing (OR Ctrl+O) > Select “receiver.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Open Block Diagram (Window > Show Block Diagram (OR Ctrl+E)).

\* Locate the “error\_detect.vi (error detect) in the block diagram and replace it with the source file you created in Lab 1 “student\_error\_detect\_<name>.vi” file (Right click on the “error detect” box > Replace > All Palettes > Select a VI...>insert your “student\_error\_detect\_<name>.vi” file).

\* Locate the “decode.vi in the block diagram and replace it with your newly created “student\_decode\_<name>.vi” file (Right click on the “decode” box > Replace > All Palettes > Select a VI...>insert your “student\_decode\_<name>.vi” file).

\* Open “top\_rx.vi” file (Open Existing (OR Ctrl+O) > Select “top\_rx.vi” > OK). A basic graphic user interface (GUI) opens. (NOTE: If a warning window pops up, click on “Ignore” option.)

\* Set the following values in Table 2 on top\_rx front panel, HW parameters tab:

LABEL	VALUE
USRP IP Address <IP> from	NI-USRP configuration utility
Active antenna	RX1
Carrier frequency (in HZ)	2G
Gain (in dB)	20

Table 2 – Top\_rx HW parameters settings

\* Go to “Modulation parameters” tab of the same front panel, and change the RX sample rate value to “2M”.

\* Click on “Run Continuously” button (two looping arrows).

For visual reference, check this video out: [See the video: How to build USRP operation](#)

### c) Data collection

Your top\_tx (which we use to control the transmitting USRP) window should look like the ones in Fig 8 and Fig 9 below:

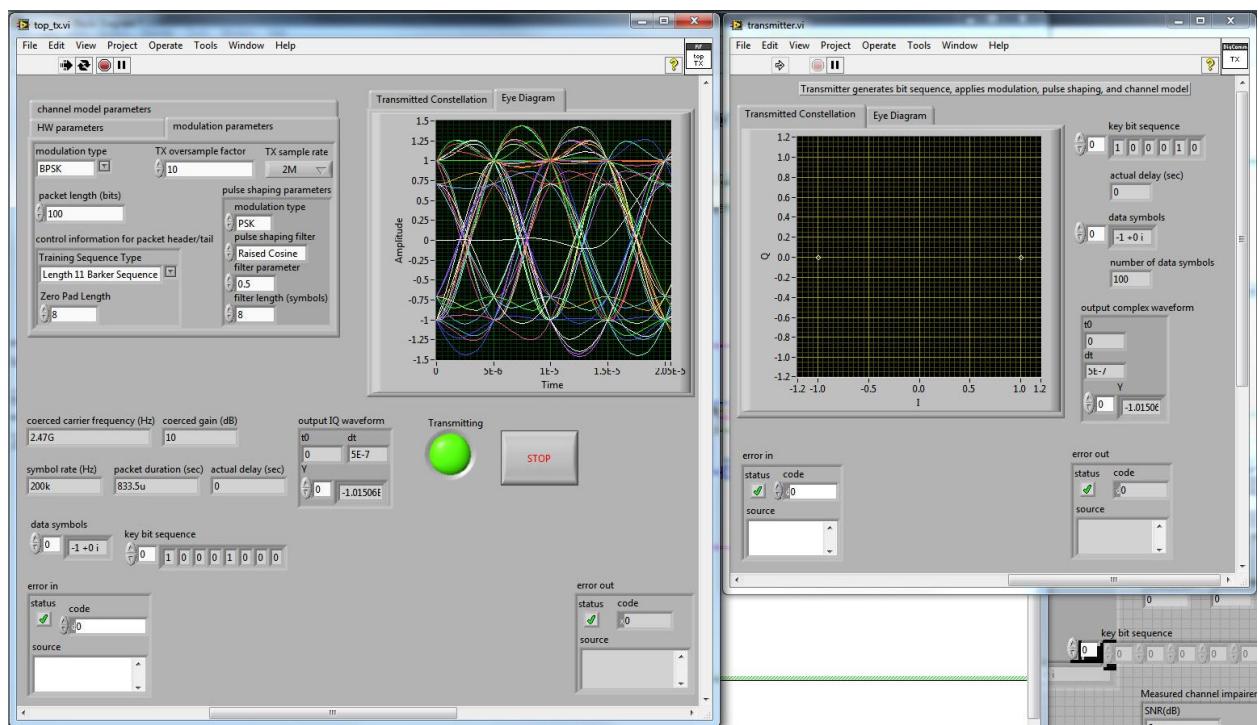


Fig 8 – Top\_tx front panel when BPSK is selected as modulation type

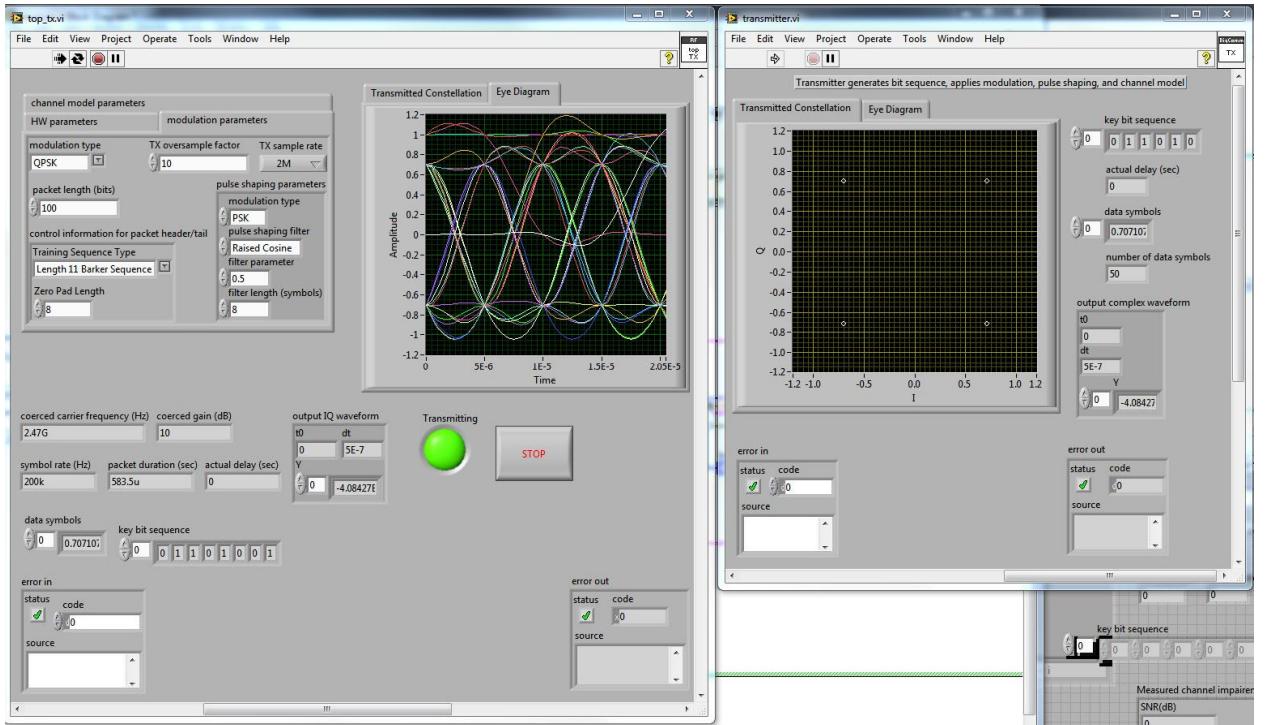


Fig 9 – Top\_tx front panel when QPSK is selected as modulation type

Your top\_rx (which we use to control the receiving USRP) window should look like the ones in Fig 10 and Fig 11 below:

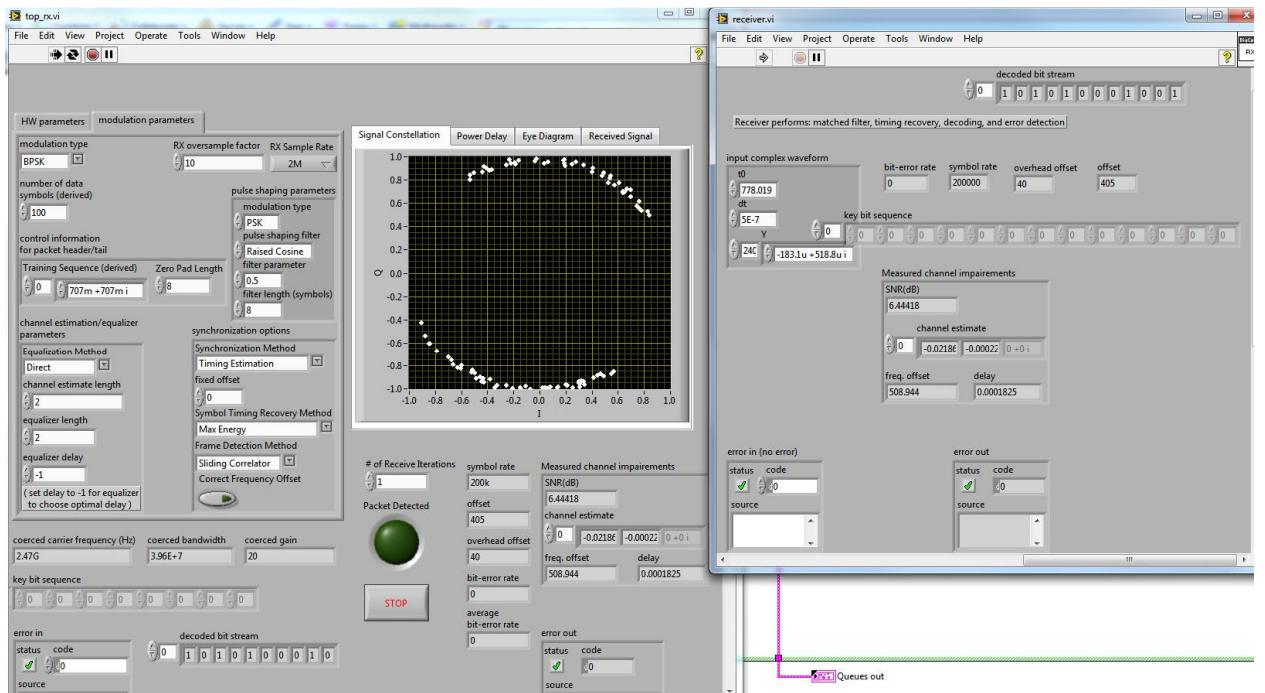


Fig 10 – Top\_rx front panel when BPSK is selected as modulation type

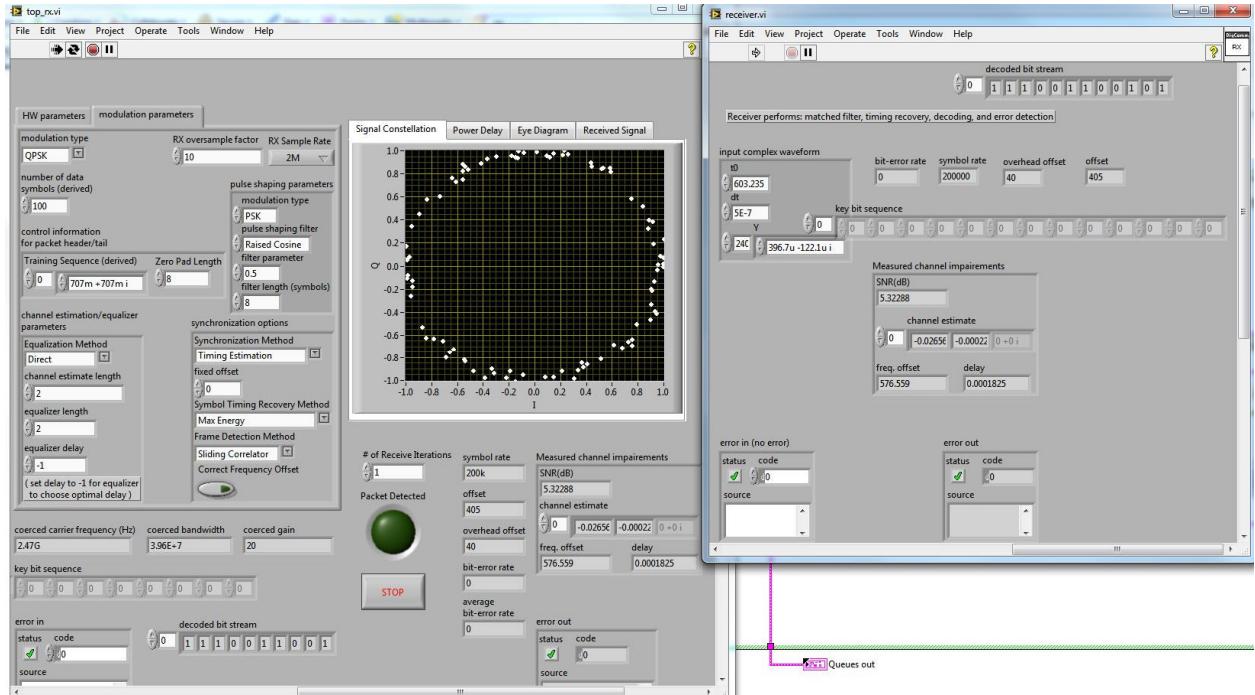


Fig 11 – Top\_rx front panel when QPSK is selected as modulation type

Detailed information on front panel objects can be found in Appendix-A of this document.

Bring up the top\_tx and top\_rx front panels and perform the measurements shown on Fig 8, 9, 10, 11 above.

Click on “Correct Frequency Offset” button on top\_rx front panel to see its effect on the results.

Repeat the USRP operation using following Rx Gain settings “60” and “15” dB for both BPSK and QPSK (Tx Gain is 0 dB) and collect screenshots of both top\_tx and top\_rx front panels. They should be used for report preparation.

# APPENDIX A

## PART-1: SIMULATION

On “simulator.vi” front panel you can manipulate the following values:

### TRANSMITTER SECTION

- a) **TX oversample factor:** Data type is long [32-bit integer (-2147483648 to 2147483647)]. This specifies the number of desired samples per symbol for the pulse shaping filter. It must be an even number larger than 2.
- b) **TX sample rate:** Data type is double [64-bit real (~15 digit precision)]. This specifies the transmitter sampling rate in Hertz (or samples per second). It must be specified between 195.312k and 100M.
- c) **Noise power:** Data type is double [64-bit real (~15 digit precision)]. This specifies the level of noise power in dB. It can be any value between –Infinity (lInf) and +lInf.

### RECEIVER SECTION

- d) **RX oversample factor:** Data type is long [32-bit integer (-2147483648 to 2147483647)]. This specifies the number of desired samples per symbol for the pulse shaping filter. It must be an even number larger than 2. It should be same as TX oversample factor.
- e) **RX sample rate:** Data type is double [64-bit real (~15 digit precision)]. This specifies the transmitter sampling rate in Hertz (or samples per second). It must be specified between 195.312k and 100M.
- f) **Correct frequency offset:** Data type is Boolean [TRUE or FALSE]. This compensates the effect of the frequency offset between transmitted and received signals. It can be set as on or off.

### SHARED SECTION

- g) **Modulation type:** Data type is string. This chooses the modulation scheme to be used in both transmitter and receiver. It can be either BPSK or QPSK.
- h) **Packet length:** Data type is long [32-bit integer (-2147483648 to 2147483647)]. This specifies the packet length to be transmitted and received in bits.
- i) **Iterations:** Data type is long [32-bit integer (-2147483648 to 2147483647)]. This changes how many times the simulation is performed. Higher number of iterations increases the accuracy, but decreases the processing speed.

In the graph field you see four different tabs:

- 1) **Received constellation:** This graph shows continuously changing I (in-phase or real) and Q (quadrature or imaginary) values of the constellation points as symbols received. These constellation points are the ones detected by the ML detector we constructed in Part-2.
- 2) **Received eye diagram:** An “eye diagram” is an oscilloscope display in which a digital data signal from a receiver is repetitively sampled and applied to the vertical input, while the data rate is used to trigger the horizontal sweep. It is called so because its shape resembles an eye. An open eye pattern corresponds to minimal signal distortion (due to intersymbol interference and noise).
- 3) **Received signal:** These two graphs show the Amplitude (in Volts) change of the received signal I and Q components in time domain.
- 4) **Power delay profile:** This graph gives the amplitude of the signal received through a multipath channel as a function of time delay (sec). The time delay is the difference in travel time between multipath arrivals.

## **PART-2: USRP OPERATION**

On “Top\_tx.vi” and “Top\_rx.vi” front panels you can manipulate the following common values:

### **HW PARAMETERS TAB**

- a) **Device name:** Data type is IVI session of class niUSRP. This area is to enter the IP address of the connected USRP.
- b) **Active antenna:** Data type is string. This area is to enter the antenna name to be used.
- c) **Carrier frequency:** Data type is double [64-bit real (~15 digit precision)]. This area is to enter the frequency of RF signal in Hertz (Hz).
- d) **Gain:** Data type is double [64-bit real (~15 digit precision)]. This area is to enter the aggregate gain applied to the RF signal in decibel (dB).

### **MODULATION PARAMETERS TAB**

- e) **Modulation type:** Data type is string. This chooses the modulation scheme to be used in both transmitter and receiver. It can be either BPSK or QPSK.
- f) **TX (or RX) oversample factor:** Data type is long [32-bit integer (-2147483648 to 2147483647)]. This specifies the number of desired samples per symbol for the pulse shaping filter. It must be an even number larger than 2.
- g) **TX (or RX) sample rate:** Data type is double [64-bit real (~15 digit precision)]. This specifies the transmitter sampling rate in Hertz (or samples per second). It must be specified between 195.312k and 100M.

The following item value is only on top\_rx.vi:

- h) **Correct frequency offset:** Data type is Boolean [TRUE or FALSE]. This compensates the effect of the frequency offset between transmitted and received signals. It can be set as on or off.

In the graph fields of “top\_tx.vi” and “top\_rx.vi” you see the following graphs:

- 1) **(top\_tx) Transmitted constellation:** This graph shows the theoretical/ideal constellation points for the selected modulation scheme (two constellation points one at  $(I,Q)=(+1,0)$  and the other one at  $(-1,0)$  for BPSK; four constellation points one at  $(I,Q)=(+0.707,+0.707)$ , second at  $(+0.707,-0.707)$ , third at  $(-0.707,+0.707)$ , and fourth at  $(-0.707,-0.707)$  for QPSK).
- 2) **(top\_rx) Signal constellation:** This graph shows continuously changing I (in-phase or real) and Q (quadrature or imaginary) values of the constellation points as symbols received.
- 3) **(top\_tx and top\_rx) Eye diagram:** It is shown on both transmitting and receiving window. The transmitting window diagram is static, while receiving window diagram is dynamic.
- 4) **Received signal:** These two graphs show the Amplitude (in Volts) change of the received signal I and Q components in time domain.

In order to export the values of a graph or array on the simulation window, follow this path and pick the output format you need: Right click on the graph/array > Export > Export (clipboard/excel/simplified image).

In order to copy the current value of an indicator, right click on the indicator while simulation is running and select “Copy Data”. This copies the current value as an image.

Also, if you need to run the simulator only for one loop, you can click on the single arrow (“Run”).

---

---