Digital Wireless Communication Practical Laboratory Session

WIRELESS COMMUNICATIONS 371-1-1903
SPRING 2020

Part 1 – Send *"Alice"*, receive *"Bob"*

Description

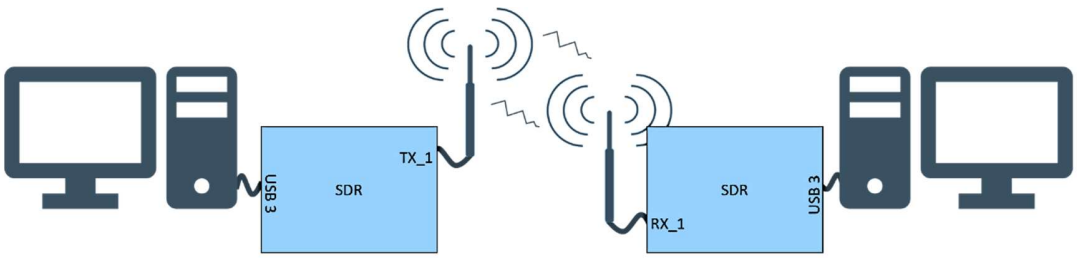
In this experiment we will construct a communication system using the GNU Radio freeware that will upgrade the ASK concept you created in the theoretical session. If you haven't performed the theoretical session you will find it difficult to perform this practical session.

Equipment needed

- 2 Linux PC with GNU Radio installed.
- 2 LimeSDR/USRP Software defined radios.
- 2 SMA to SMA RF cables
- 2 ULF-SMA Adapters.
- Attenuators box.

If any of the above equipment is missing or defective, notify the lab instructor prior to starting this session or otherwise you may obtain false results (and major frustration).

Equipment Setup



Recording

This is a practical session; thus, you most likely do not have this equipment at your disposal at home other than currently, here. Therefor it is highly recommended that you document and record your results during this session. This will assist you in completing you report at home. You may not fully complete what that is required of you during the time you have, take this into account.

Instructions

1. Create 2 new .grc files, each in its own PC. One will be the transmitter and the other will be the receiver.

2. Download Alice.txt (Alice In Wonderland textbook) from moodle.

3. Build the transmitter as following:

    a. Set the "samp_rate" to $192KHz$.

    b. Add a "File Source" and set the file path to where Alice.txt is. Set Repeat to "Yes".

    c. Add a "Varicode Encoder" and connect its input port to the output port of the "File Source".

        - **Varicode** is a self-synchronizing code used for text compression. It converts Byte stream to Binary stream and supports all ASCII characters, but the characters used most frequently in English have shorter codes. The space between characters is indicated by a 00 sequence, an implementation of Fibonacci coding. Originally created for speeding up real-time keyboard-to-keyboard exchanges over low bandwidth links.

    d. Add a "Chunks to Symbols", set the symbol table: "$0,1$" with dimensions: "1". Connect its input port to the output port of the "Varicode Encoder".

    e. Add "Import" component and enter "$import\ numpy$ " in the Import field.

        - This allows us to use Python libraries in our code.

    f. We need a convolution FIR filter. Add "Interpolating FIR filter" and set the Interpolation to: 20. To set a filter characteristic, insert the following in the Taps field:

    ```
    numpy.convolve(numpy.array(filter.firdes.gaussian(1, 20, 1.0, 4*20)),numpy.array((1,) * 20))
    ```

    connect its input port to the output port of the "Chunks to Symbols" component.

    g. Add "Multiply const" with a constant of 0.9. Connect its input port to the output port of the "Interpolating FIR filter" component.

    h. Add a "Rational Resampler" for changing the sample rate by 10. Enter a Interpolation value of 10. Connect its input port to the output port of the "Multiply const" component.

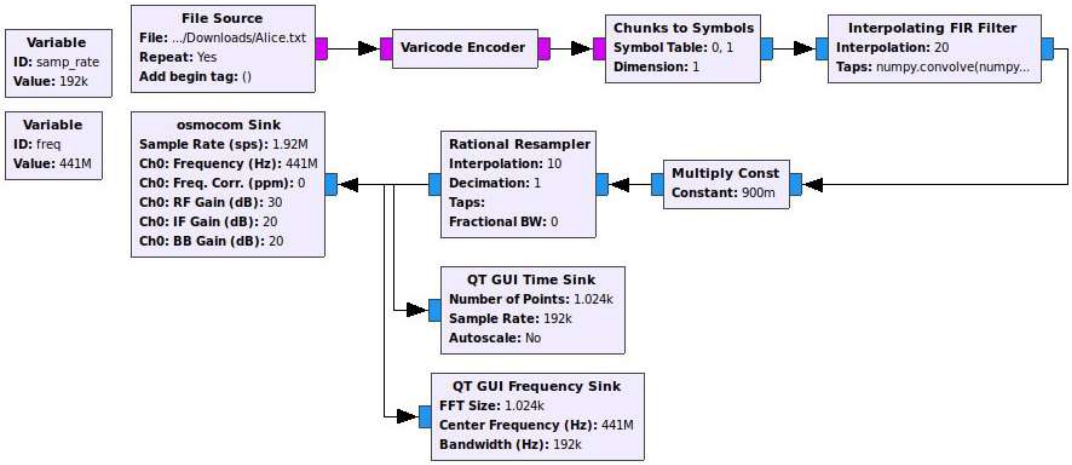
    i. To the "Rational Resampler" connect a "QT GUI Time Sink" and a "QT GUI Frequency Sink" (both with the control panel set to "Yes")

j. Create a Variable for the carrier frequency. Set it to:

$$400\text{MHz}+\text{"your pair number"}*20\text{MHz}$$

k. Add a “osmocom Sink” and set its frequency as your defined variable. Set the sample rate to $10*\text{samp\_rate}$. Set the gain to 30dB. Connect it to the output port of the “Rational Resampler” component.

If you preformed all the stages correctly, your system’s code should look like this:



The modulation itself takes place in the SDR (by mixing the signal generated in Gnuradio with the carrier signal – field “CH0 Frequecny” in OsmocomSink)

4. Now, Build the receiver as following:

a. Set the “samp_rate” to $1.2MHz$.

b. Create a Variable for the carrier frequency. Set it like your transmitter:

$$400\text{MHz}+\text{"your pair number"}*20\text{MHz}$$

c. Add a “QT GUI Range” for the gain, from 0dB to 70dB in steps of 1dB and with a default value of $50dB$.

d. Add a “osmocom Source” and set its frequency and RF gain as your defined variable and range accordingly.

e. Add a “Low Pass Filter” and set its properties as follows:

- Decimation: `5`
- Gain: `1`
- Cutoff Frequency: `50KHz`
- Transition Width: `25KHz`

Connect its input port to the “osmocom Source” output port.

f. Add a “Complex to Mag” component and connect its input port to the output port of the “Low Pass Filter” component.

g. Add a float “DC Blocker” and set the Length to 320.

- This will ensure that our data is centered at 0V DC so that we can decode it.

connect its input port to the output port of the “Complex to Mag” component.

h. To the “DC Blocker” connect a float “QT GUI Time Sink” and a float “QT GUI Frequency Sink” (both with the control panel set to “Yes”)

i. Place a float “Clock Recovery MM” - this block is needed to recover a symbol form input signal (a clock allows us to know what is the symbol duration):

- in field “Omega” type `(samp_rate/5/9600)`

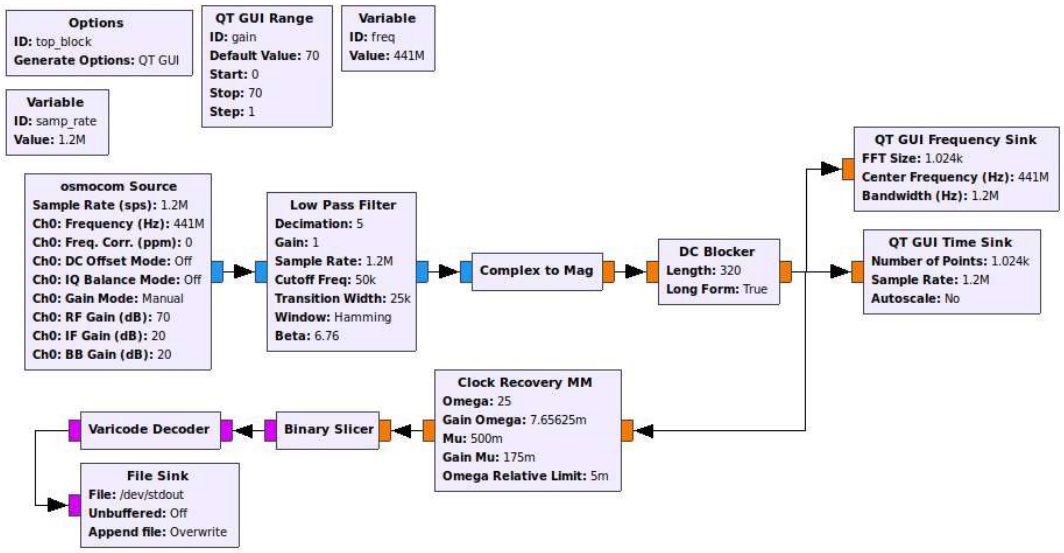
Connect its input port to the output port of the “DC Blocker” component.

j. Add a “Binary Slicer” to slice the float signal to a binary stream. Connect its input port to the output port of the “Clock Recovery MM” component.

k. Add a “Varicode Decoder” and connect its input port to the output port of the “Binary Slicer”.

l. Add a “File Sink” and in the file path write: `/dev/stdout`
This will print the decoded data to the console.

If you preformed all the stages correctly, your system's code should look like this:



Save your code and add it to your submission!

5. Go ahead and run your code. You should see the text being transferred between the computers.

Report

Make sure your system is working correctly prior to filling this report

Include snapshots and explanations of your results in your submission.

Power vs distance graph

a. Place both antennas closely together. The printed text should be errorless.
b. Measure and document the voltage of the data in the receiver side.
c. Slowly move the antennas apart and away from each other and measure and document the voltage of the data in the receiver side.
d. Repeat several times (at least 5 different distances) until you can clearly see errors in the printed text.

Plot and add a Power vs distance graph to your submission. Include a short explanation to this phenomenon and your measured results.

Corona! Can't do this remotely…

1. **Power vs distance graph**
   Instead of empirically, plot a Power vs Distance graph based on what you would expect to have measured and add it to your submission. Include a short explanation to this phenomena (justification).

2. **Thresholds**
   Change the RF gain of the transmitter and/or the receiver until errors appear, you can add a gain slider.
   a. Find and record the threshold in which errors begin to accrue.
   b. Find and record the threshold in which **NO** data is decoded at all. You can use the gain in your SDRs to achieve a sufficient attenuation. If the data keeps streaming, explain why.
   c. What are the causes to these thresholds?
   d. How can we improve these thresholds? Give at least 2 approaches.