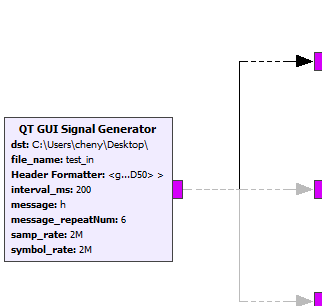
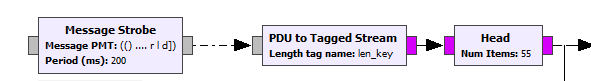
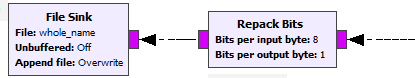
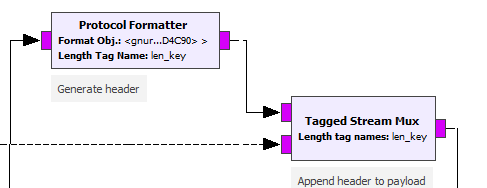
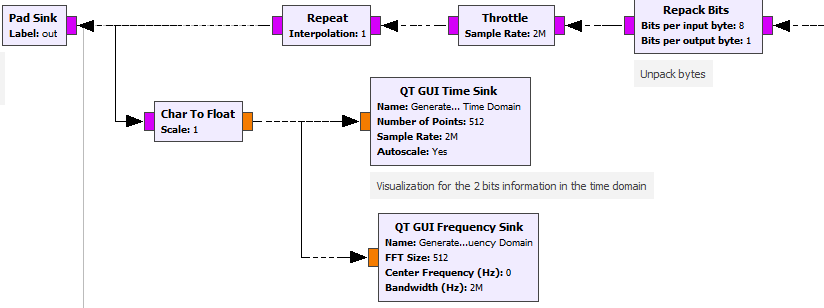
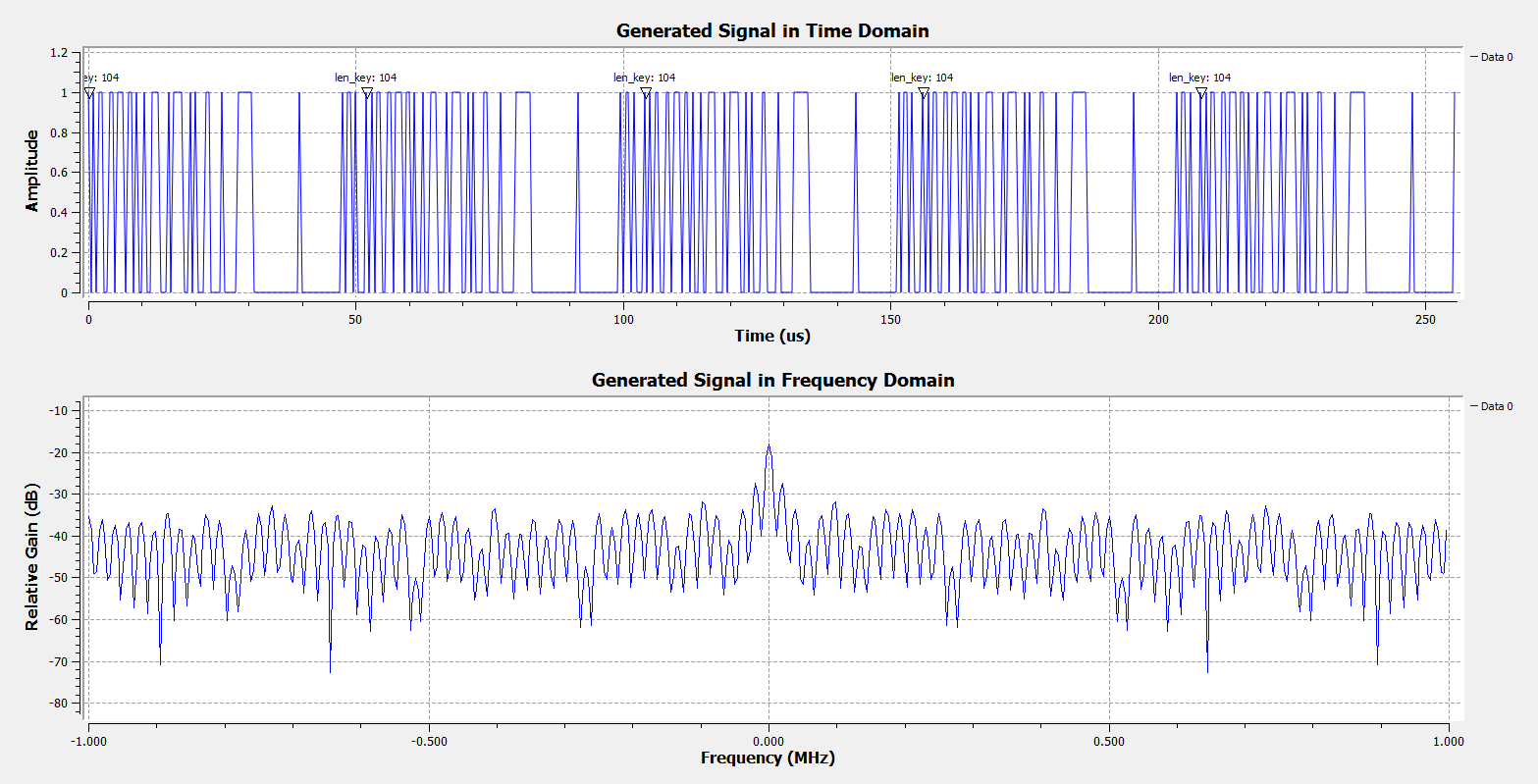
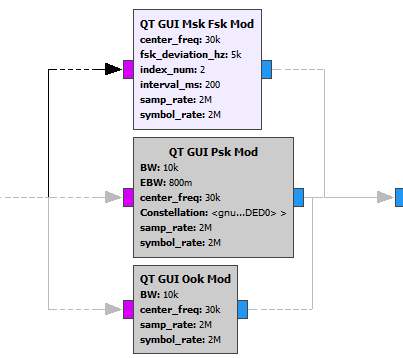
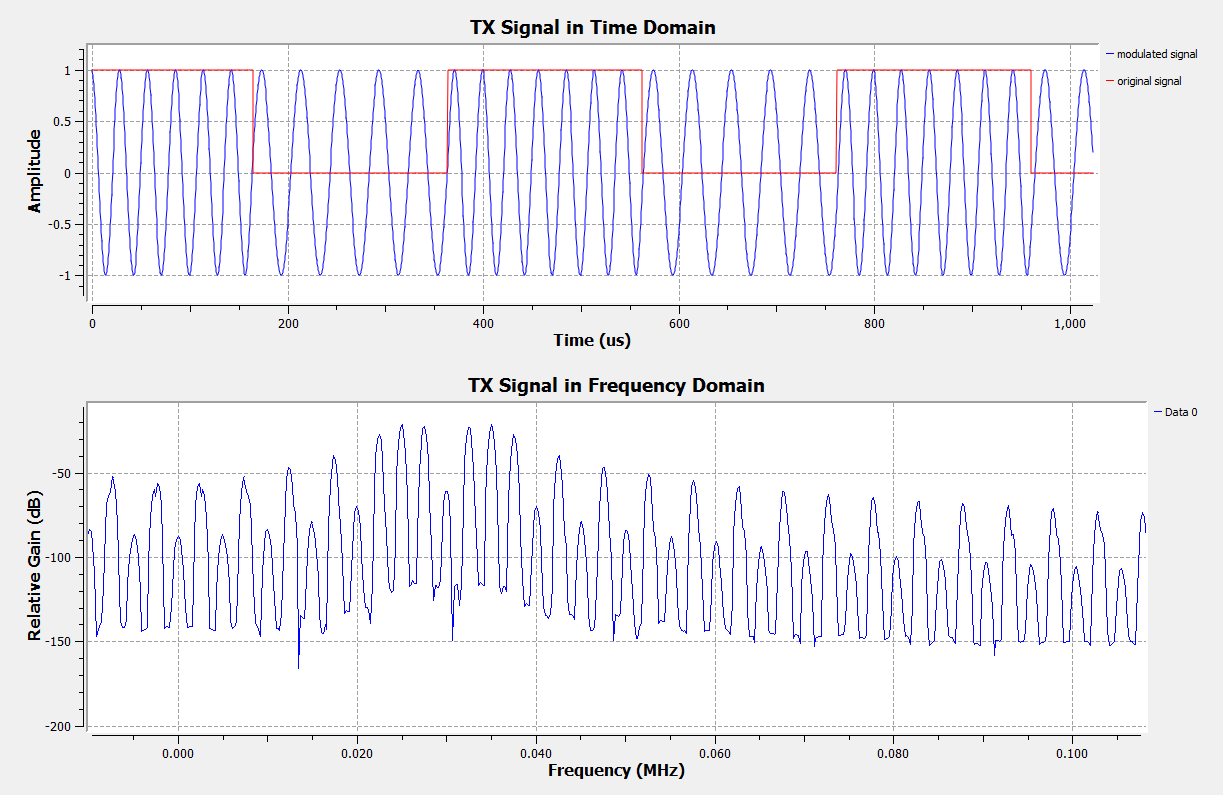
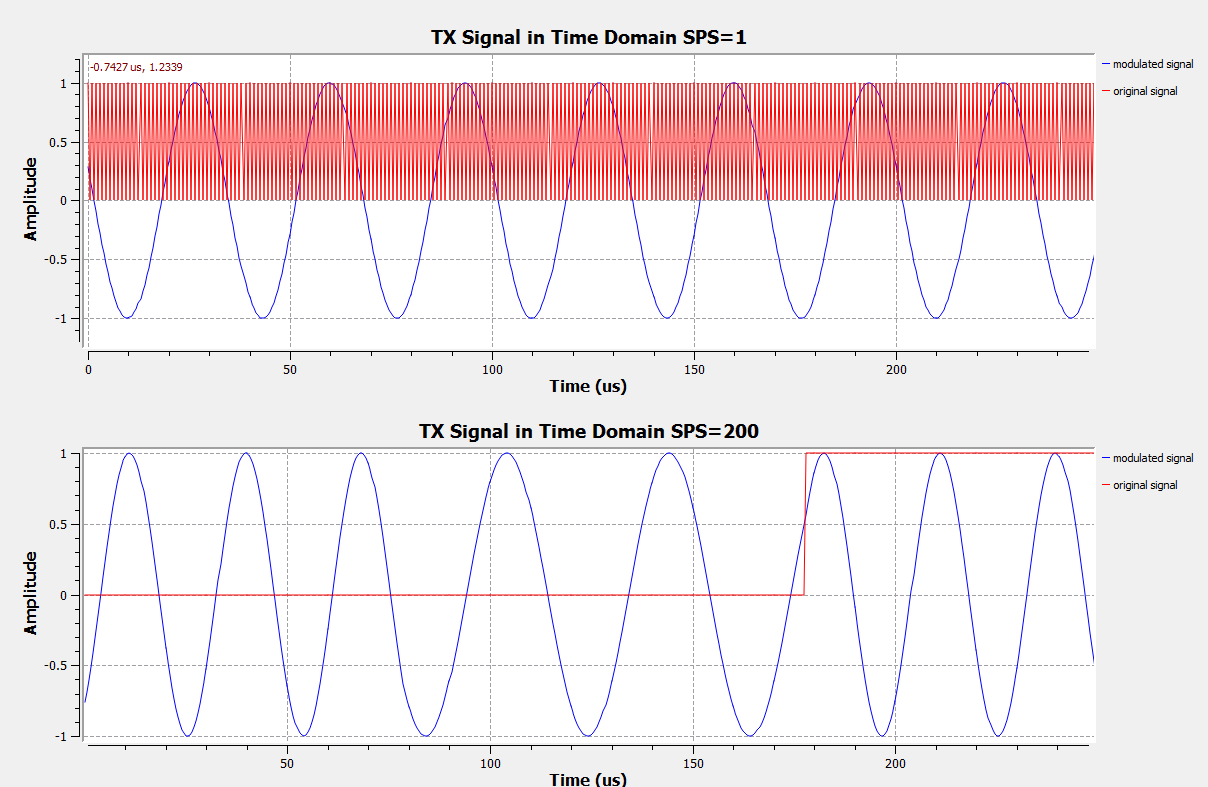
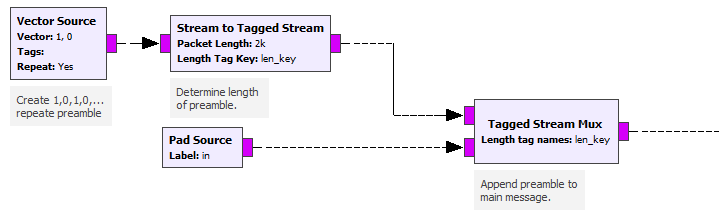
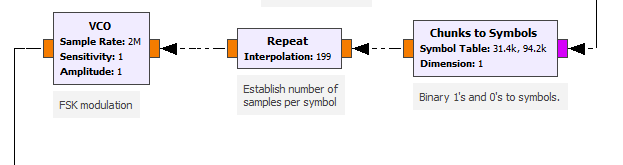
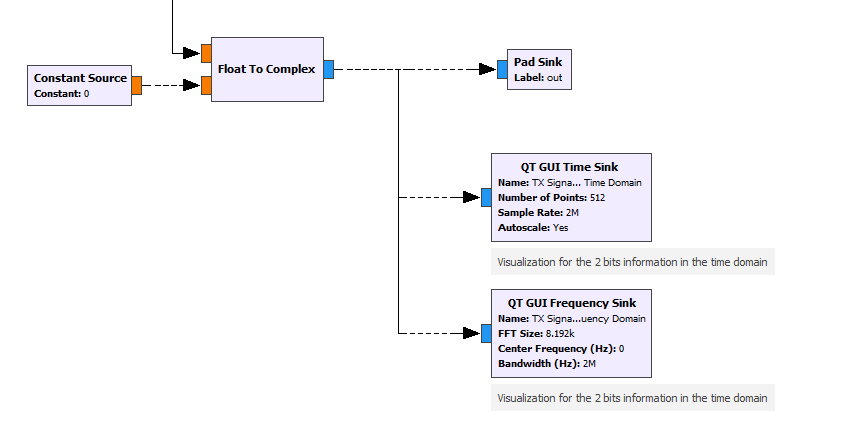
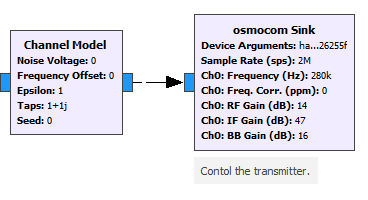
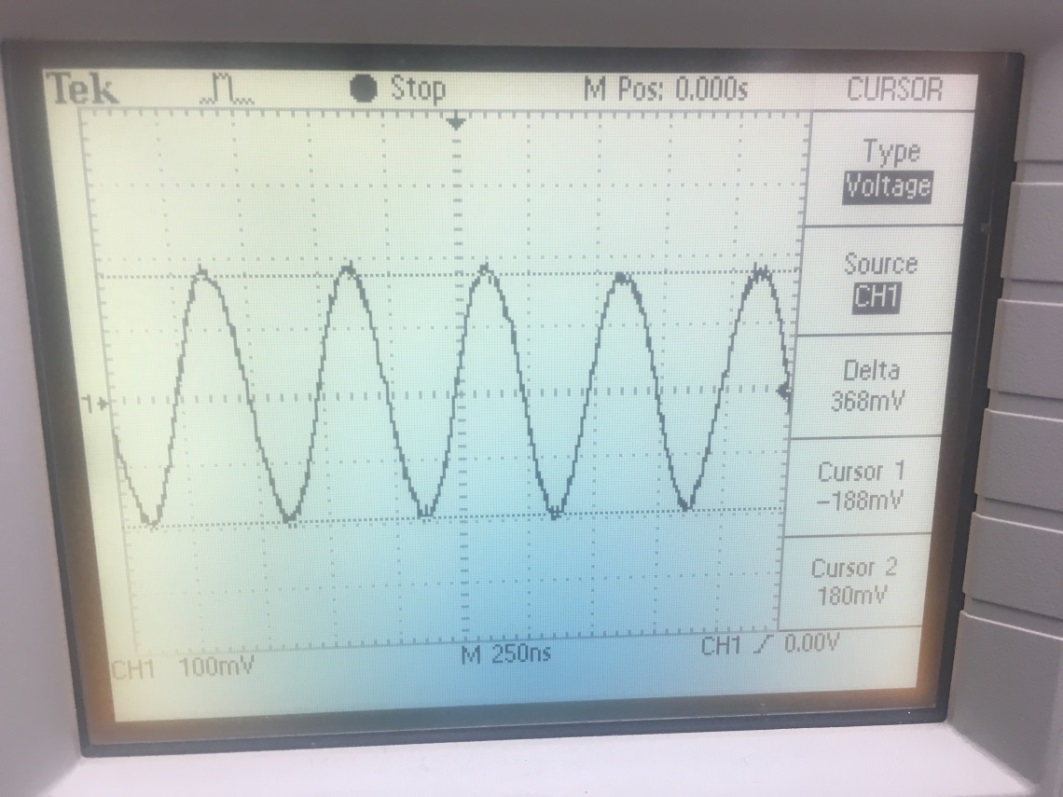
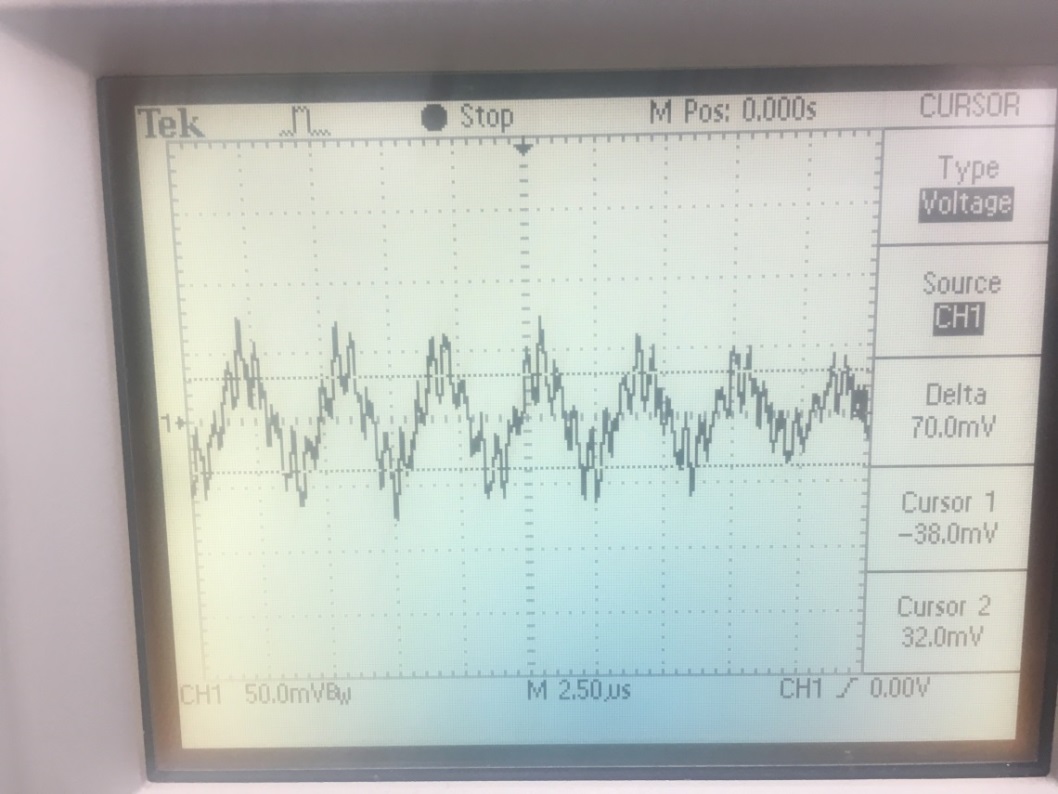
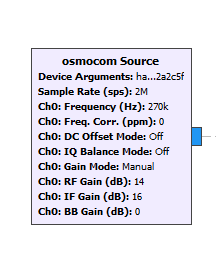
* **Signal Generator:**
  + 
  + Parameters:
    - **Dst**: Designation folder of all saved signal files
    - **Filename:** the name of the original digital signal transformed from test message
    - **Header Format:** Used to generate the access code attached to the beginning of each signal packet. During actual transmission mode, computer at the receiver side will check the access code first. If the error of access code is below the threshold. Then computer will regarded the following signal as target message. During the testing mode, access code will help to synchronize the reference signal and the received signal for bit error comparison.
      * GNUradio has build-in function to generate the access code *‘digital.header\_format\_default(digital.packet\_utils.default\_access\_code, 0)’*
    - **Interval\_ms:**  The interval time between two messages. The unit is ms so ‘200’ means 200ms.
      * In the heart rate simulation it is usually the interval time between two beats. So ….
    - **Message:**  The actual text message you want to send. One character is 1 byte long signal. For example, send “Helloworld” is actually sending 80 bits signals.
    - **Message\_RepeatTime:** How many times to you want to send the same message. To get an accurate Bit Error rate, we usually repeat the message several times and calculate the average Bit Error Rate. Here I repeat 5 times.
    - **Sample Rate**: The sample rate of generated signal., means number of discrete samples the computer will process per seconds.
      * Require to be >=2\*maximum frequency of processing signals.
      * In the function generator …
      * Sample rate should not within the chip capacity, usually smaller than clock rate
    - **Symbol Rate:** Number of symbols per second. For example, ‘a’ can be decode as ‘01100001’, if the symbol rate is 100 then each ‘1’/’0’ should last for 0.01 seconds.
      * The symbol rate should > actual need.
      * The symbol rate should < = sample rate, otherwise the computer is unable to process.
  + Workflow Details:
    - 
      * Generate text message for each Interval \_ms *(Interval \_ms = 200)* and convert that to digital stream. Use head to only transmit ‘len(message)\*(message\_repeatNum) ‘ digits. *(message\_repeatNum = 5, and len(message) = 11)*
    - 
      * Repack the 1 byte symbol into 8 bots and save the original signal file as reference when do bit error comparison *(whole\_name = dst + filename)*
    - 
      * Attach the access code to the message.
    - 
      * Repack the access-code-attached message symbol into 8 bits. Send that to the next stage in Pad Sink and also visualize them in Time Domain and Frequency Domain.
  + Example Result Screenshot: 
    - In the example screenshot I send one byte signal ‘H’. The length of access code is always 96. So the total length is 104 and the same pattern repeated for 5 times as you can see from the time domain.
* **Modulation/FSK:** 
  + 
  + Parameter: 
    - **Center\_freq:** The center of f0 ( ‘0’ frequency) and f1 ( ‘1’ frequency). For example, in the picture, I use 25KHz to represent 0 and 35KHz to represent 1, so the center\_freq=30KHz
    - **Fsk\_deviation:** |f1-f0|/2, in the above example, fsk\_deviation = 5KHz
      * In the FSK algorithm, required *|f1-f0|=symbol\_rate\*index\_num/2* to keep orthogonality.
      * Ideally, |f1-f0| is the minimum requirement for sample rate. In the practice, we want center\_freq to be some positive number in order to get a better gain. So the minimum requirement for sample\_rate is *fsk\_deviation + center\_freq*
    - **Index\_num**: …. Must be an positive integer
      * When index\_num = 1, the algorithm is MSK (minimum shift key). Here index\_num =2 achieved best result.
    - **Interval\_ms**: refer to the signal generator section. Must be the same
    - **Sample\_rate:** refer to the signal generator section. Must be the same
    - **Symbol\_rate:** refer to the signal generator section. Must be the same.
  + Dependent Variables:
    - **SPS: *=( symbol\_rate/fsk\_deviation\_hz/4)\*(index\_num).* Must be integer*.***
      * Sample per symbol. As I mentioned above, 2\*fsk\_deviation = *|f1-f0|=symbol\_rate\*index\_num/2.* However the symbol rate it required to fulfill the fsk\_deviation needs might be too low for us to capture. So we repeat the each sample for **SPS** times to generate some “pseudo symbol rate”. For example the original symbol rate is 2MHz which means each symbol last *1/2M = 0.5e-6 s*, by repeating SPS=200 times, each symbol last *1/2M\*200 = 1e-4 s,* which means the “pseudo symbol rate” *2M/200 = 10KHz.*In the above picture you can realize without SPS, the modulated wave is unable to catch such fast sample rate.
    - **Pream\_vec\_len: *= samp\_rate\*interval\_ms\*1e-3/SPS***
      * Length of preamble vector. During the interval time, there is only noise existing instead of message. So large error will occur when jump from the noise to received message. In order to avoid such error, I send 1,0,1,0 … during the interval time.
  + Workflow Details:
    - 
      * Attach the preamble 1,0,1,0… signal to the encoded signal packet created from **Signal Generator (from pad source).**
    - 
      * Match the binary digits to its representative frequency, repeat that for **SPS** times to generate “pseudo symbol” and use vco to actually generate the cosine wave with corresponding frequency
    - 
      * Because signal in reality is always in complex form. So change float to complex by inserting 0 imaginary part. Then send it to the next stage as well as visualize it in the frequency/time domain.
* **Interface to transmitter:**
  + 
  + Parameter:
    - **Device Argument:** The address number of hackrf. Type command “hackrf\_info” to check the hackrf you connected with.
    - **Sample Rate:** refer to the signal generator section. Must be the same
    - **RF Gain (dB):** 0 or 14 dB; amplifier for the final transmitting frequncty
      * Here I choose the maximum 14
      * …
    - **IF Gain (dB):** 0 to 47 dB in 1dB steps.
      * Here I choose the maximum 47dB; amplifier for intermediate frequency
      * …
    - **BB Gain:**  Not used in TX side.
    - **In the TX side, there are not much difference in RF and IF gain.**
    - **Noise Voltage:** Used to simulate the noise voltage in channel
      * If you are using the physical plug in noise, just set it to 0.
    - **Frequency offset:** The normalized frequency offset. 0 is no offset; 0.25 would be, for a digital modem, one quarter of the symbol rate.
      * Usually set to 0
    - **epsilon** : The sample timing offset to emulate the different rates between the sample clocks of the transmitter and receiver. 1.0 is no difference.
      * Usually set to 0
    - **taps :** Taps of a FIR filter to emulate a multipath delay profile.
      * Usually set to 0
    - **noise\_seed** : A random number generator seed for the noise source.
      * Usually set to 0
  + Dependent Variable:
    - **Ch0 Frequency: *= carrier\_freq – center\_freq (see the Modulation section)***
      * …
  + Workflow Details:
    - Channel Model is used to simulate the channel environment and Osmocom Sink is the interface to control Hackrf as transmitter.
  + Example Result Screenshot:
    - This is what the transmitted signal looks like when the frequency is in the recommended range, for example Fs=2M,Fc=2.01M,Amp: 252mV-632mV/2, , (0dB RF Gain, 16dB IF Gain) 
    - This is what the transmitted signal looks like when the frequency is OUT OF the recommended range, for example Fs=2M, Fs=270K,Fc=280K, Amp: 60mV/2, (0dB RF Gain, 16dB IF Gain) 
    - Other in/out of range signals have similar behavior as these two examples.
* **Interface to Receiver**
  + 
  + Parameter:
    - **Device Argument:** refer to the Interface to Transmitter
    - **Sample Rate:** refer to the signal generator section.
      * Can be the same or smaller than the sample rate of transmitter.
      * If smaller than the Hackrf act as a low pass filter. For example when the sample rate of the transmitter is 2M, which include 0-1MHz signals. 1M sample rate for the receiver will exclude all high frequency signals that are > 0.5MHz.
      * Here I choose the same sample rate as the transmitter. Please refer to … for more details.
    - **RF Gain (dB):** 0 or 14 dB; amplifier for the final transmitting frequncty
      * Here I choose the maximum 14
      * …
    - **IF Gain (dB):** 0 to 40 dB in 8dB steps. amplifier for intermediate frequency
      * Here I choose the 16dB because the bigger gain will introduce additional more strong low frequency noise;
      * …
    - **BB Gain:**  0 to 62 in 2dB steps; amplifier at base band stages
      * **…**
      * Here I choose 0dB because…
    - **In the RX side, RF gain should be the first choice.**
  + Dependent Variable:
    - **Ch0 Frequency: *= carrier\_freq – center\_freq (see the Modulation section)***
    - In order to recover the modulated Signals, this hould be exactly the same as transmitter (unless there is frequency offset in the channel which is unusual).