Communication design project EN 2130

Group 04

BANDARA H.M.C.N.K 200063X

KAUSHALYA K.W.K. 200298V

LUCKSHAN G.W.C.M. 200358G

PASQUAL A.C. 200445V

Content

- Objective
- Requirements
- Block Diagram of a Communication System
- GNU Radio
- Modulation
- Encoding
- Encryption
- Simulation
- Real world Transmission

Objective

Implement a point-to-point digital wireless secure and reliable communication system using software defined radios

Requirements

Transmitter design

Encoding

Security implementation

Reliability enhancement

Jamming protection

modulation

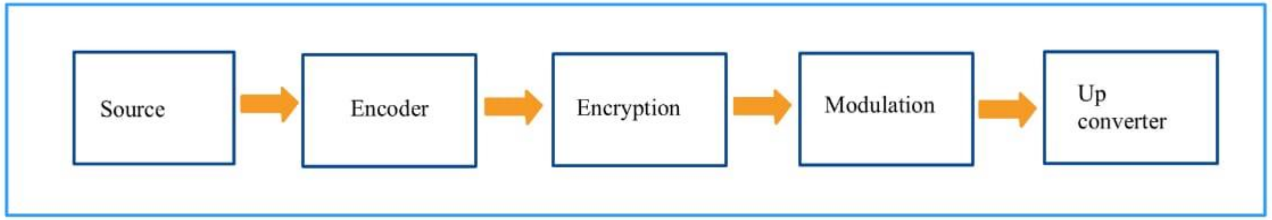
Receiver design

Reliability enhancement

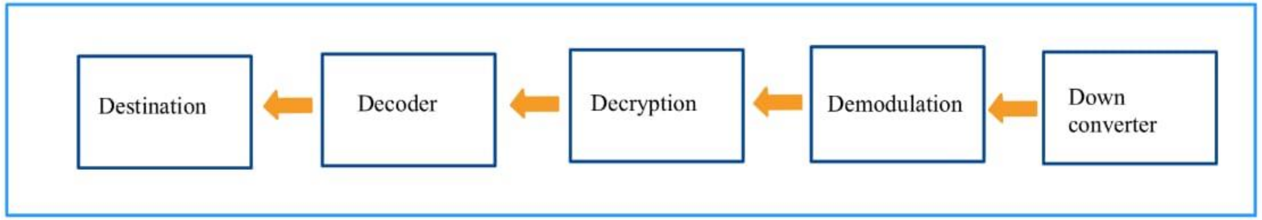
Demodulation

decoding

Block Diagram



Transmitter



Receiver

GNU Radio

- Version 3.10
- Free and open-source software development toolkit
- Offers signal processing building blocks to create software defined radios
- Can be utilized in a simulation like environment without hardware

Modulation

BPSK – Binary Phase Shift Keying

Comparing Phase vs Frequency modulation Phase modulation is more immune to noise effects

Therefore, it is more suited for data transmission

Modulation

QPSK – Quadrature Phase Shift Keying

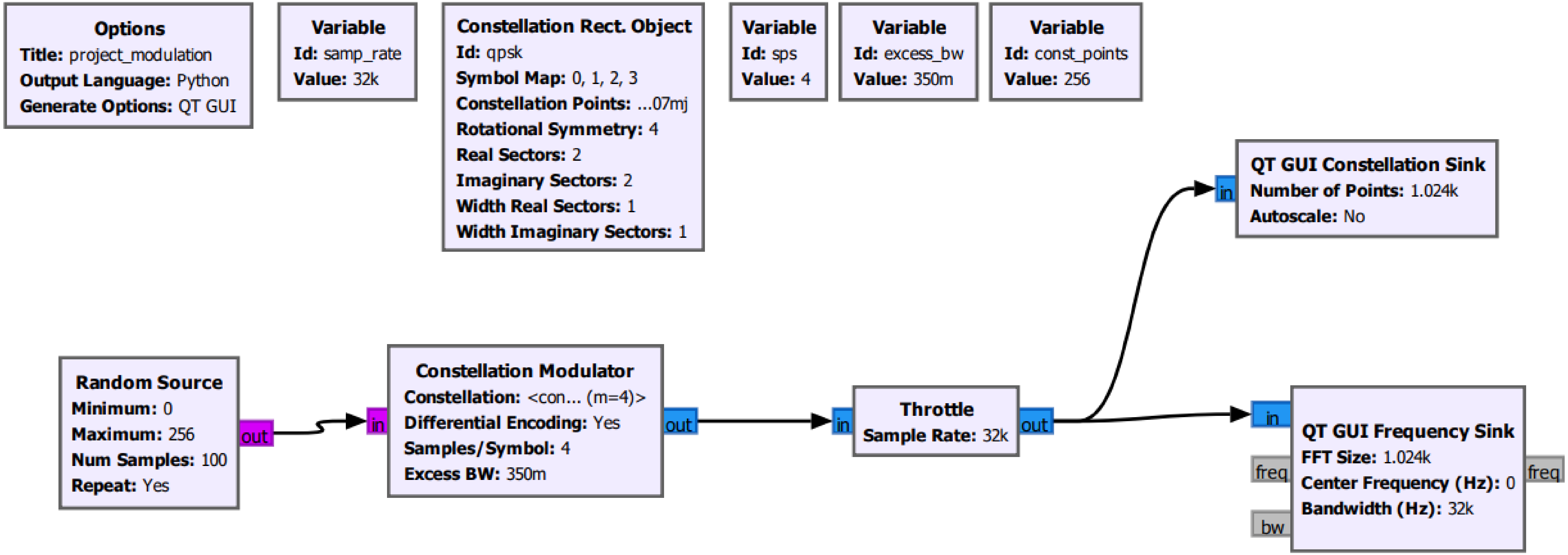
Comparing Phase vs Frequency modulation Phase modulation is more immune to noise effects

Therefore, it is more suited for data transmission

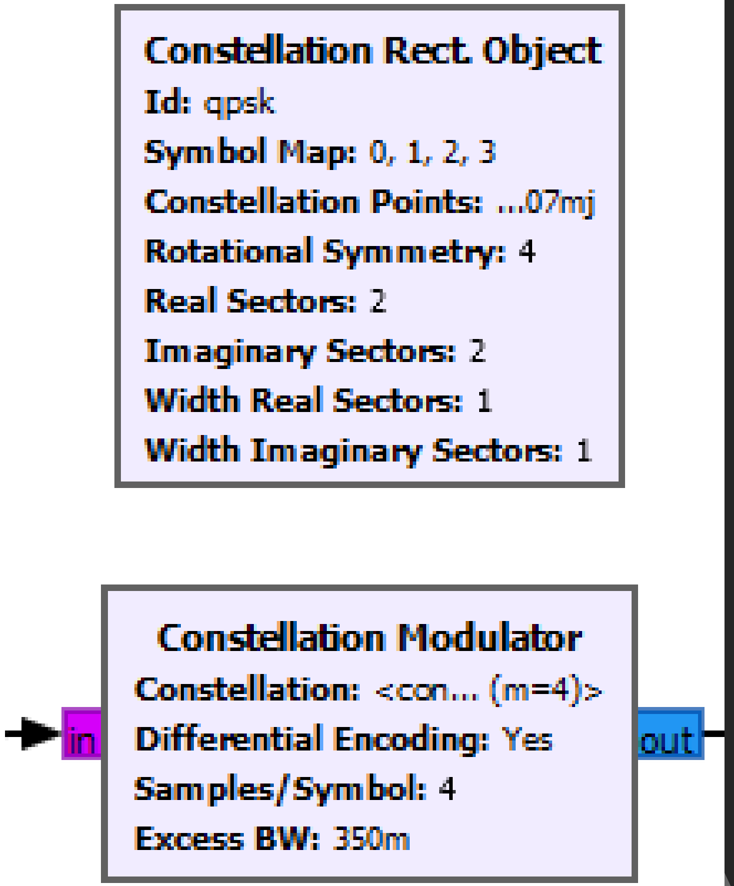
- Why BPSK ?
  - Comparing to QPSK, information transmission rate of BPSK is lower due to the high bandwidth
  - But BPSK has low error probability
  - BPSK modulated data can travel long distances
  - BPSK modulation is more power efficient
  - Therefore, BPSK was used as the modulation technique

- Why QPSK ?
  - Spectrally efficient
  - For the same BER bandwidth required is half as BPSK
  - Because of the low bandwidth, information transmission rate is higher than BPSK

Modulation

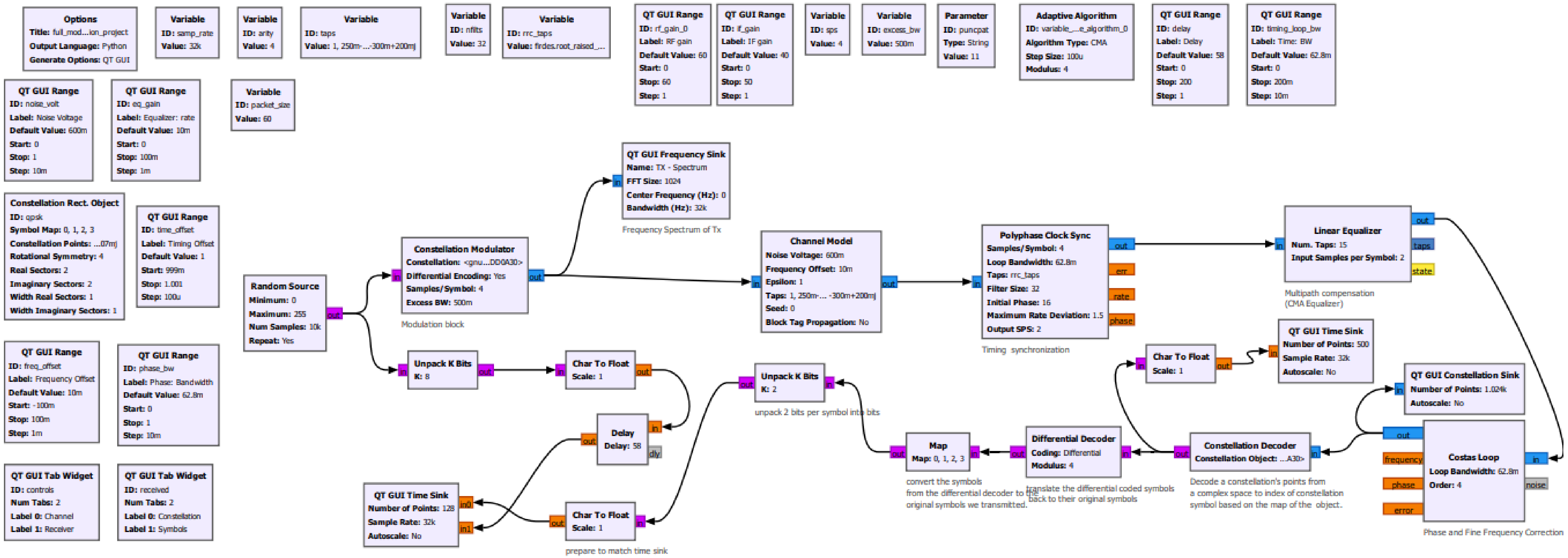


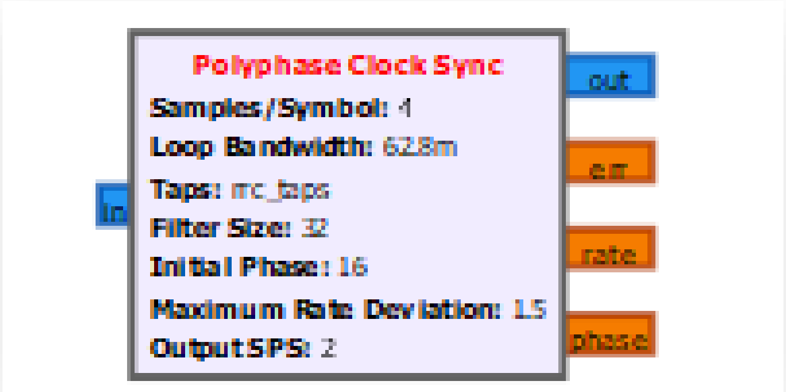
Options
Title: project_modulation
Output Language: Python
Generate Options: QT GUI
Variable
Id: samp_rate
Value: 32k
Constellation Rect. Object
Id: qpsk
Symbol Map: 0, 1, 2, 3
Constellation Points: ...07mj
Rotational Symmetry: 4
Real Sectors: 2
Imaginary Sectors: 2
Width Real Sectors: 1
Width Imaginary Sectors: 1
Variable
Id: sps
Value: 4
Variable
Id: excess_bw
Value: 350m
Variable
Id: const_points
Value: 256
QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No
in
Random Source
Minimum: 0
Maximum: 256
Num Samples: 100
Repeat: Yes
out
in
Constellation Modulator
Constellation: <con... (m=4)>
Differential Encoding: Yes
Samples/Symbol: 4
Excess BW: 350m
out
in
Throttle
Sample Rate: 32k
out
in
QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 32k
freq
bw
freq



- Constellation Modulator
  - A digital signal processing block that performs modulation by mapping digital symbols to points in a complex plane
  - Differential Encoding : Yes
    - Instead of encoding each symbol independently, the difference between adjacent symbols is encoded
    - Reduces the sensitivity of the receiver to phase shifts in the signal, which can be caused by noise or fading.
    - Improves the reliability of the data transmission

Full modulation Diagram





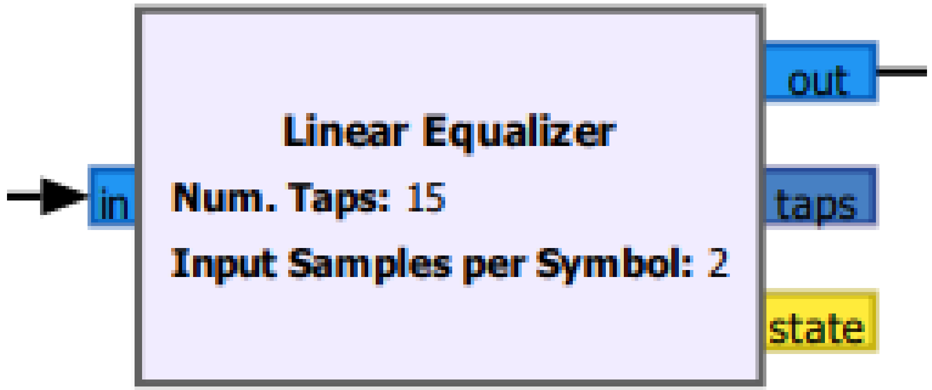
Polyphase Clock Sync

- A clock recovery technique
- Removes ***Inter-Symbol interference (ISI)*** and down samples the signal to produce samples at a desired rate
- Useful in digital communication systems where the transmitted signal may experience phase and timing offsets due to channel distortion or other factors

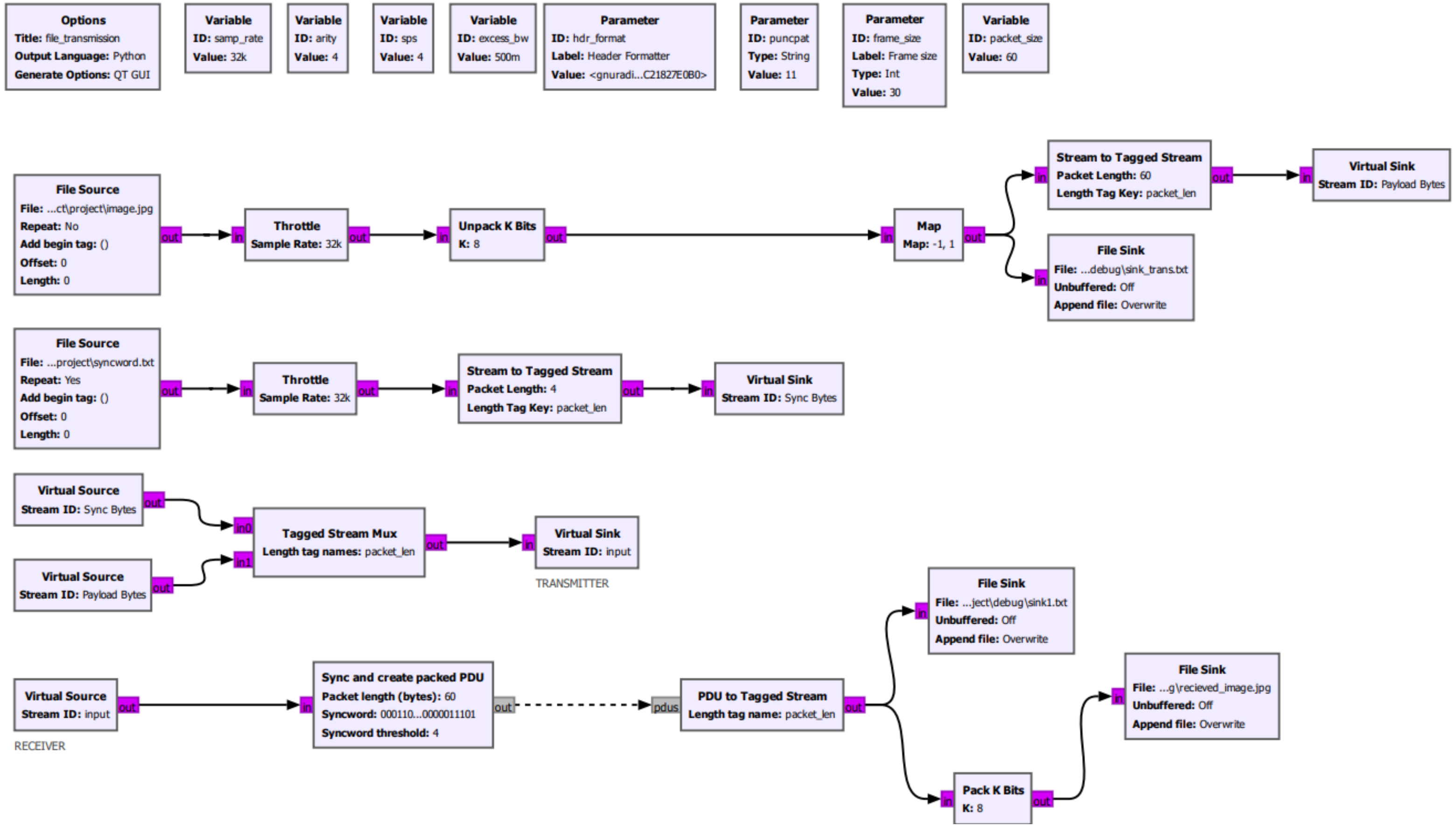
Costas Loop

- Used for ***carrier phase synchronization*** such that it synchronize a receiver’s local oscillator with the received carrier signal
- Uses a phase error detector to measure the difference between the phase of the received signal and the phase of the local oscillator.
- This phase error is then used to adjust the phase of the local oscillator so that it tracks the phase of the received signal.
- Simple and effective

- Linear Equalizer
  - The Linear Equalizer block equalizes the incoming signal using an FIR filter
  - Used in a receiver chain to compensate for the effects of multipath fading, which can cause time- and frequency-selective distortion of the received signal.
  - Increases the gain of the needed signal and decreases the gain of the distortions

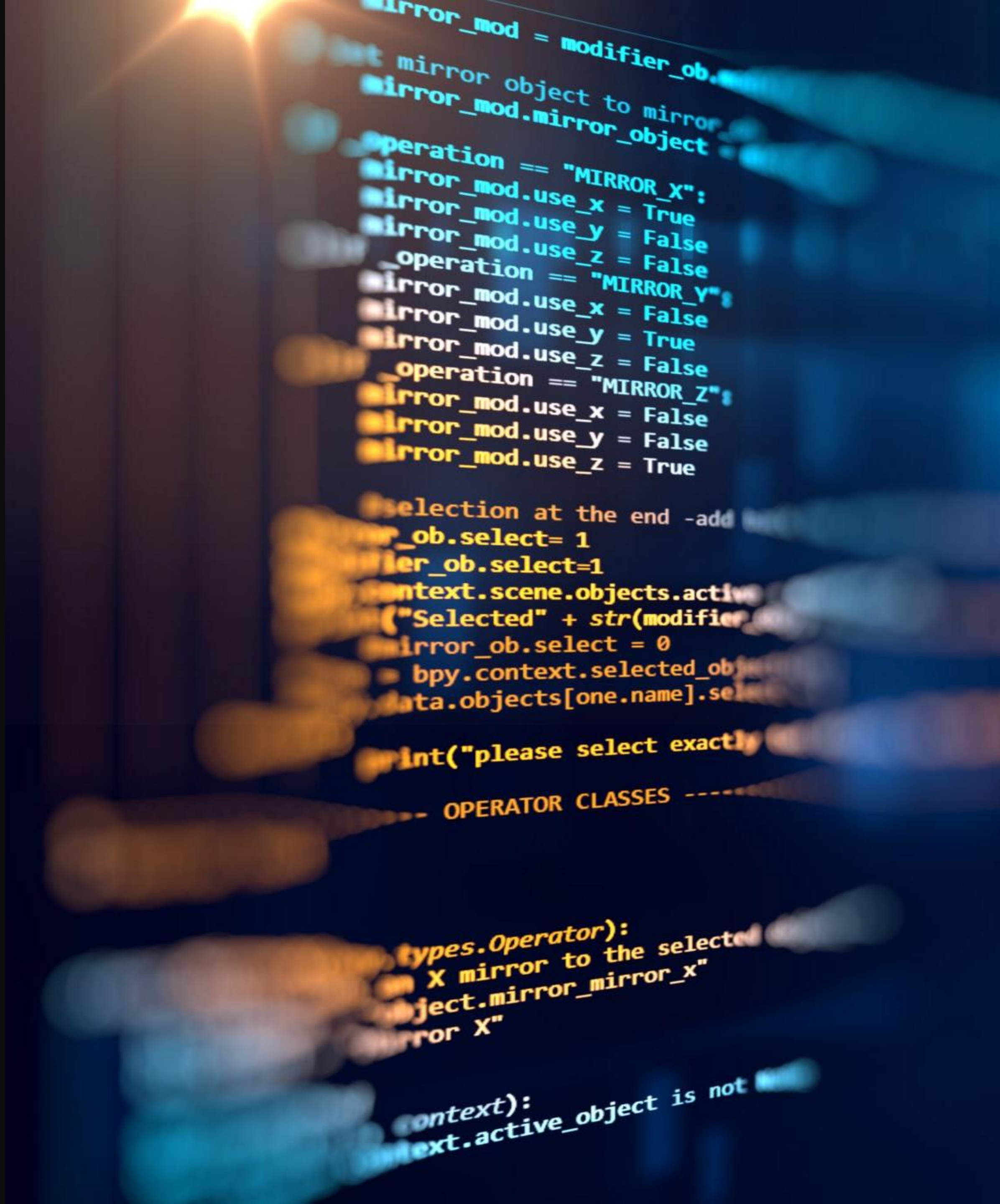


File Transmission

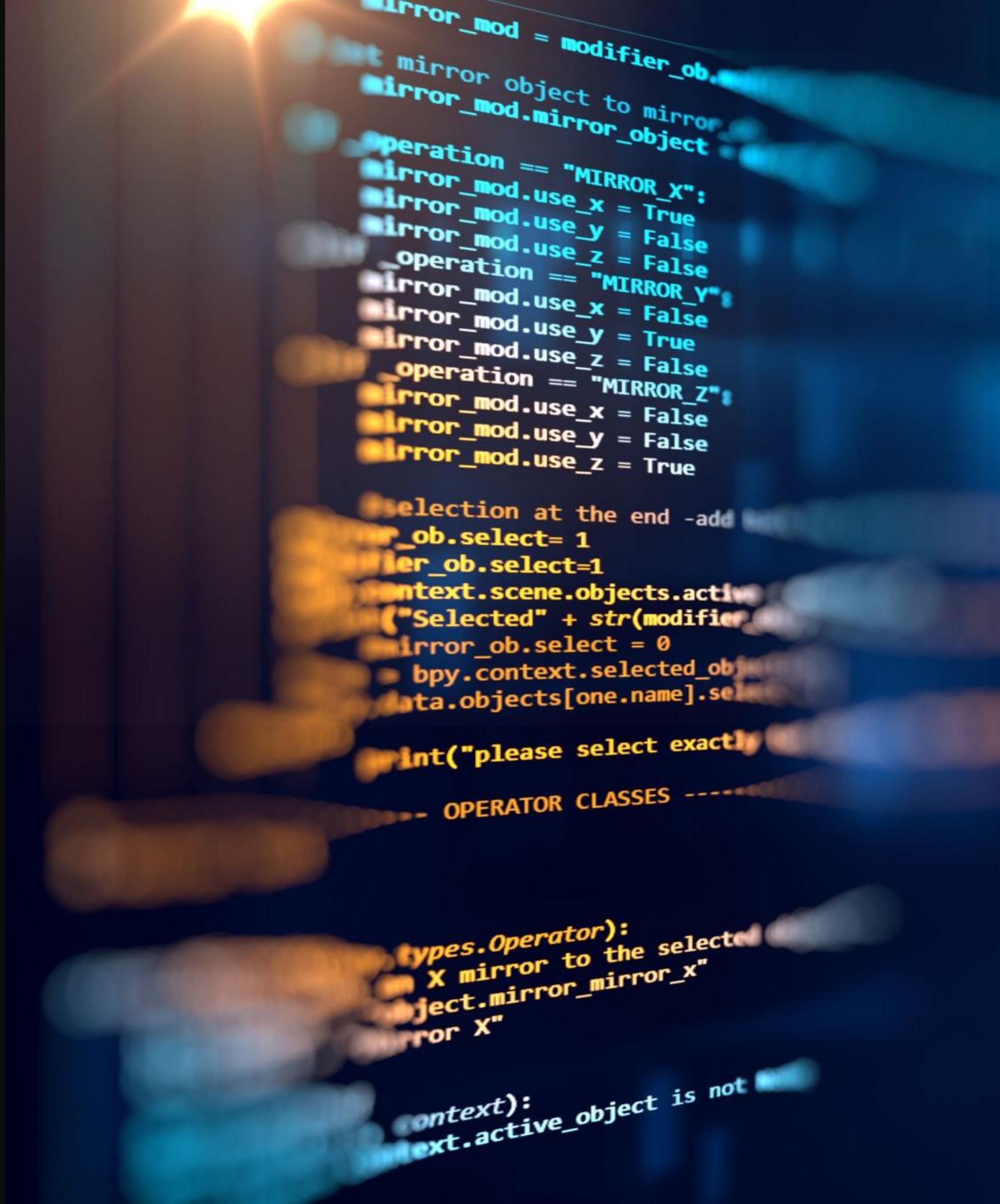


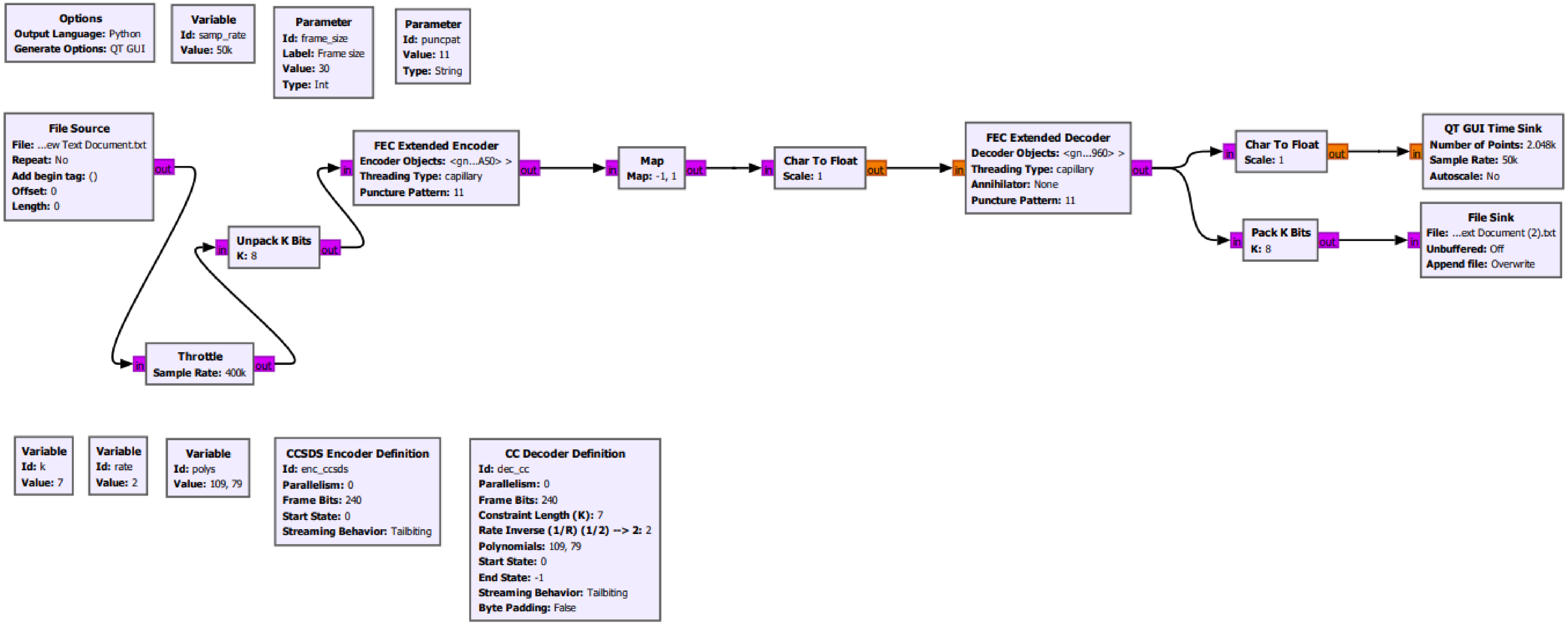
Channel coding (Forward Error Correction)

- Since a feedback can’t be obtained, FEC is used as the error correction method
- Transmitter encodes the data with an error correcting code and sends the coded message
- Channel decoder in transmitter accepts message bits and adds redundancy according to prescribed rule
- Channel decoder in receiver exploits the redundancy to decide which message bits were transmitted
- 2 FEC methods
- Block coding and Convolution coding

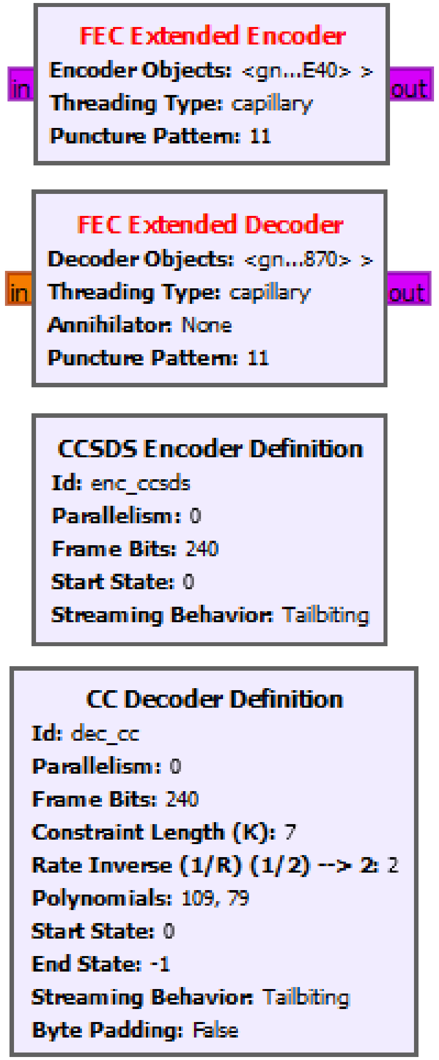


- Block coding : Check block by block and add parity
- Convolution coding : Check bit by bit and add parity
- Convolution vs Block coding
  - Block coding is easier to implement and simpler to understand
  - But Convolution coding has better error controlling correction capabilities
  - Therefore, convolution coding is used




Options
Output Language: Python
Generate Options: QT GUI
Variable
Id: samp_rate
Value: 50k
Parameter
Id: frame_size
Label: Frame size
Value: 30
Type: Int
Parameter
Id: puncpat
Value: 11
Type: String
File Source
File: ...ew Text Document.txt
Repeat: No
Add begin tag: ()
Offset: 0
Length: 0
Throttle
Sample Rate: 400k
Unpack K Bits
K: 8
FEC Extended Encoder
Encoder Objects: <gn...A50> >
Threading Type: capillary
Puncture Pattern: 11
Map
Map: -1, 1
Char To Float
Scale: 1
FEC Extended Decoder
Decoder Objects: <gn...960> >
Threading Type: capillary
Annihilator: None
Puncture Pattern: 11
Char To Float
Scale: 1
QT GUI Time Sink
Number of Points: 2.048k
Sample Rate: 50k
Autoscale: No
Pack K Bits
K: 8
File Sink
File: ...ext Document (2).txt
Unbuffered: Off
Append file: Overwrite
Variable
Id: k
Value: 7
Variable
Id: rate
Value: 2
Variable
Id: polys
Value: 109, 79
CCSDS Encoder Definition
Id: enc_ccsds
Parallelism: 0
Frame Bits: 240
Start State: 0
Streaming Behavior: Tailbiting
CC Decoder Definition
Id: dec_cc
Parallelism: 0
Frame Bits: 240
Constraint Length (K): 7
Rate Inverse (1/R) (1/2) --> 2: 2
Polynomials: 109, 79
Start State: 0
End State: -1
Streaming Behavior: Tailbiting
Byte Padding: False

FEC with modulation



- FEC Extended Encoder Block
  - Used to implement the forward error correcting method
  - Threading type : capillary
    - Applying FEC to small segments of data instead to the entire block
    - Detect and correct errors in real time instead of waiting for the entire block to be transmitted
    - Therefore, faster error correction and detection compared to other threading types

  - Encoder object : CCSDS Encoder
    - Streaming behavior : Tailbiting
      - Split the data into multiple blocks and add a CRC code to the end of each block
      - If an error is detected the decoder uses the redundant data at the end of the previous block to correct the error
      - Useful for streaming data transmission
      - Ensures that the errors in the transmitted data are corrected in real time without the retransmission of the entire blocks of data

Encryption

- ❑ Data file was encrypted, and the encryption key was sent through a secured channel
- The data file was encrypted using a symmetric key cryptography technique.
- AES is used
- AES vs DES
  - AES is the most commonly used symmetric cryptography technique
  - AES is more mathematically efficient
  - AES is significantly faster
  - AES is exponentially stronger than DES due to high number of bit keys
    - AES – 128,192,256-bit keys
    - DES – 56-bit keys

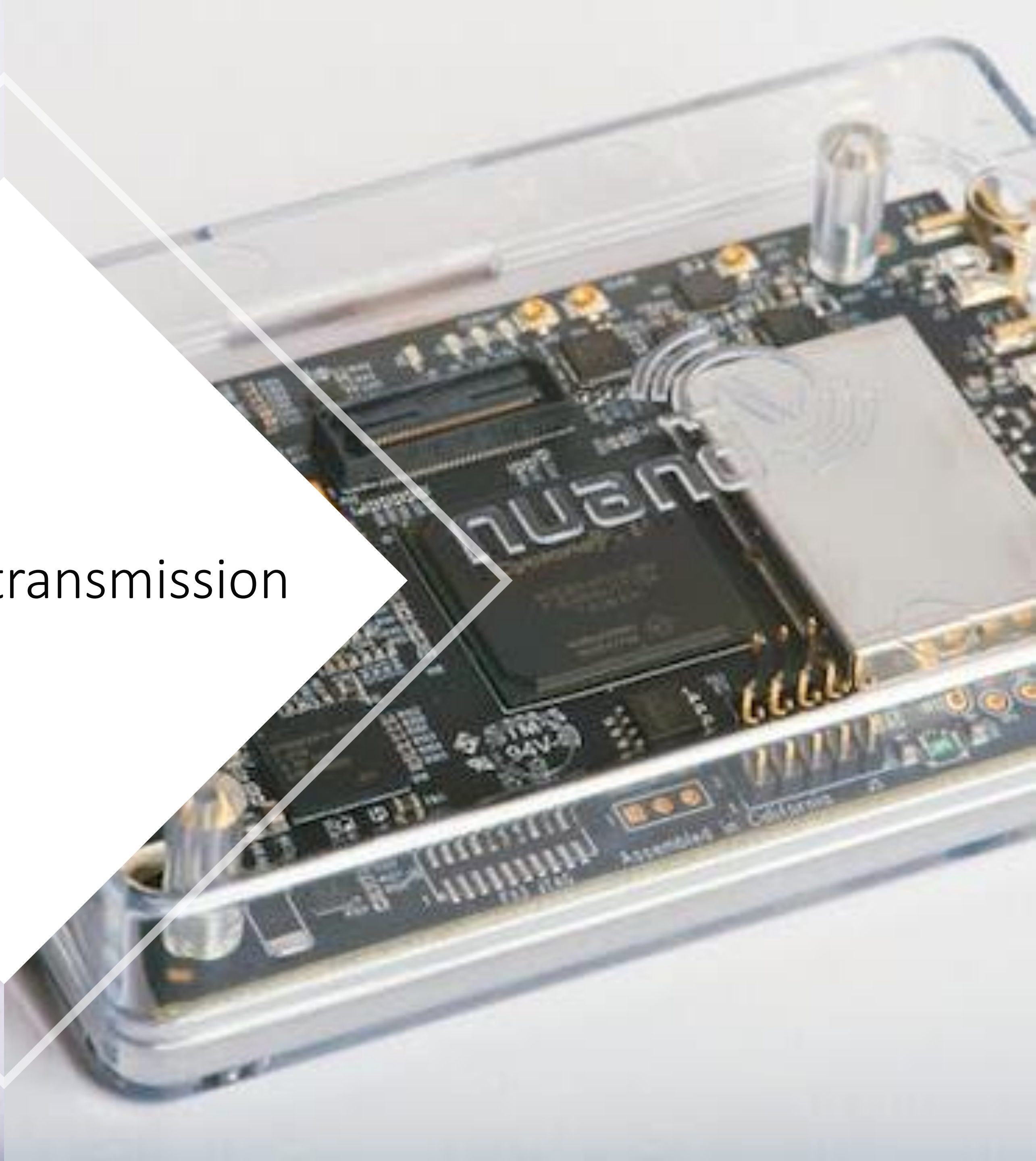
- The encryption key was sent through an asymmetric key encrypted channel
- RSA is used
- Since encryption key doesn't have to be transmitted regularly, RSA is used even though it's slow

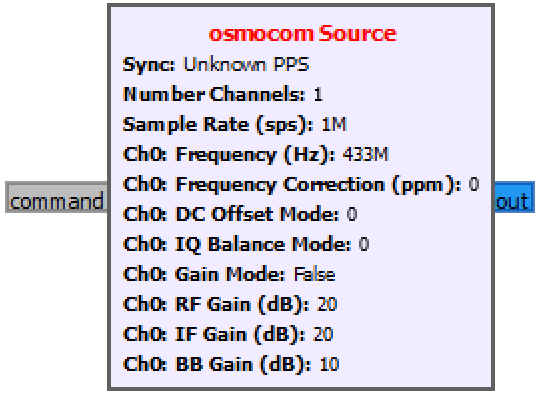
- RSA is more secure than AES
- But RSA channel isn’t capable of sending a file with large number of bits.
- Hence the data file can’t be sent through RSA even though it’s more secured
- Therefore, the encryption key was sent through RSA while the file was sent through AES

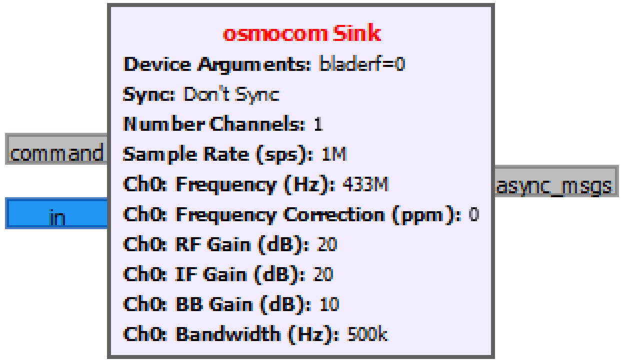
Encryption blocks

Simulation

Real world transmission







- Osmocom Source
  - provides an interface to receive data from various software-defined radio (SDR) devices
- Osmocom Sink
  - Osmocom Sink block provides a powerful and flexible way to send data over the air using a range of RF transceivers

THANK YOU!