HELLOW

PROJECT PRESENTATION

Rupasinghe N.P.S.S. - 210549D
Madhushan I.D. - 210349N
Sirimanna N.T.W. - 210610H
Gamage S.B.P. - 210178M

OUR GOALS

Implement a point to point digital wireless secure and reliable communication system using softawear defined radios.

DESIGN METHODOLOGY

- GNU Radio for simulation purposes.
- BladeRF for Physical Implementations.

Project Requirements

Transmitter

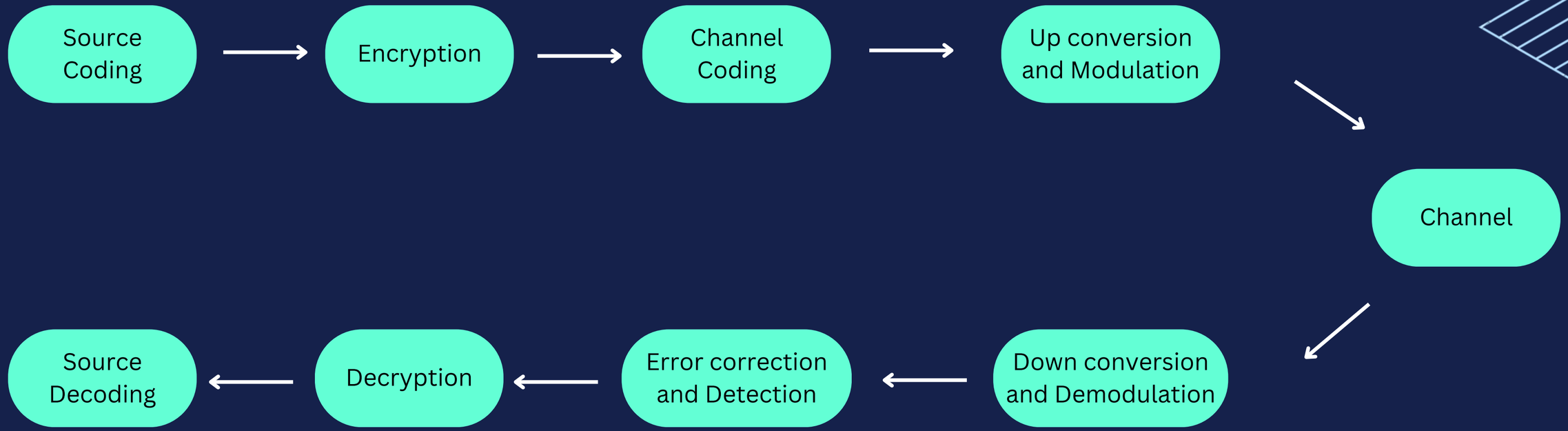
- Encoding
- Enhance Reliability
- Modulation

Project Requirements

Receiver

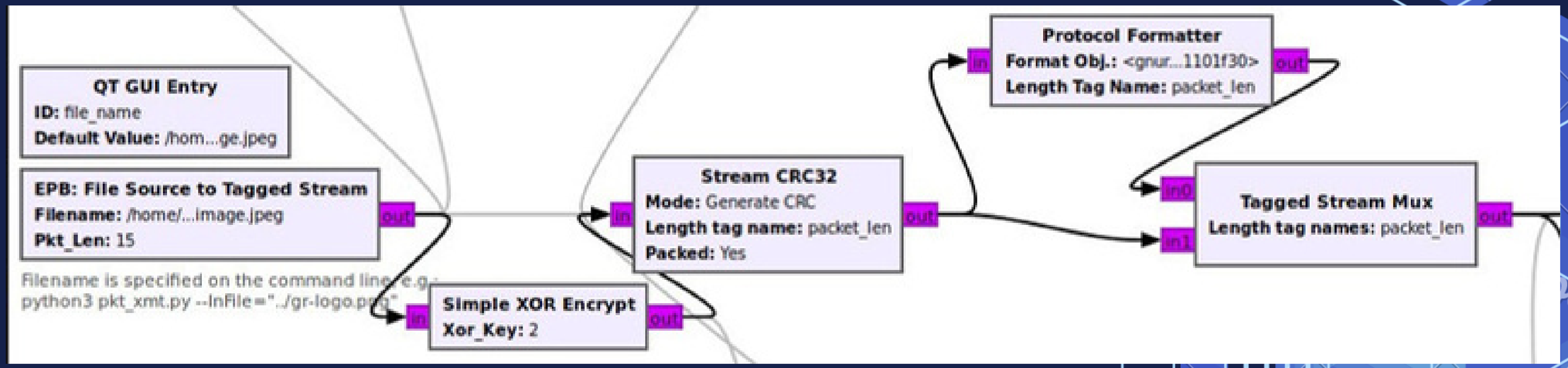
- Encoding
- Demodulation

BLOCK DIAGRAM

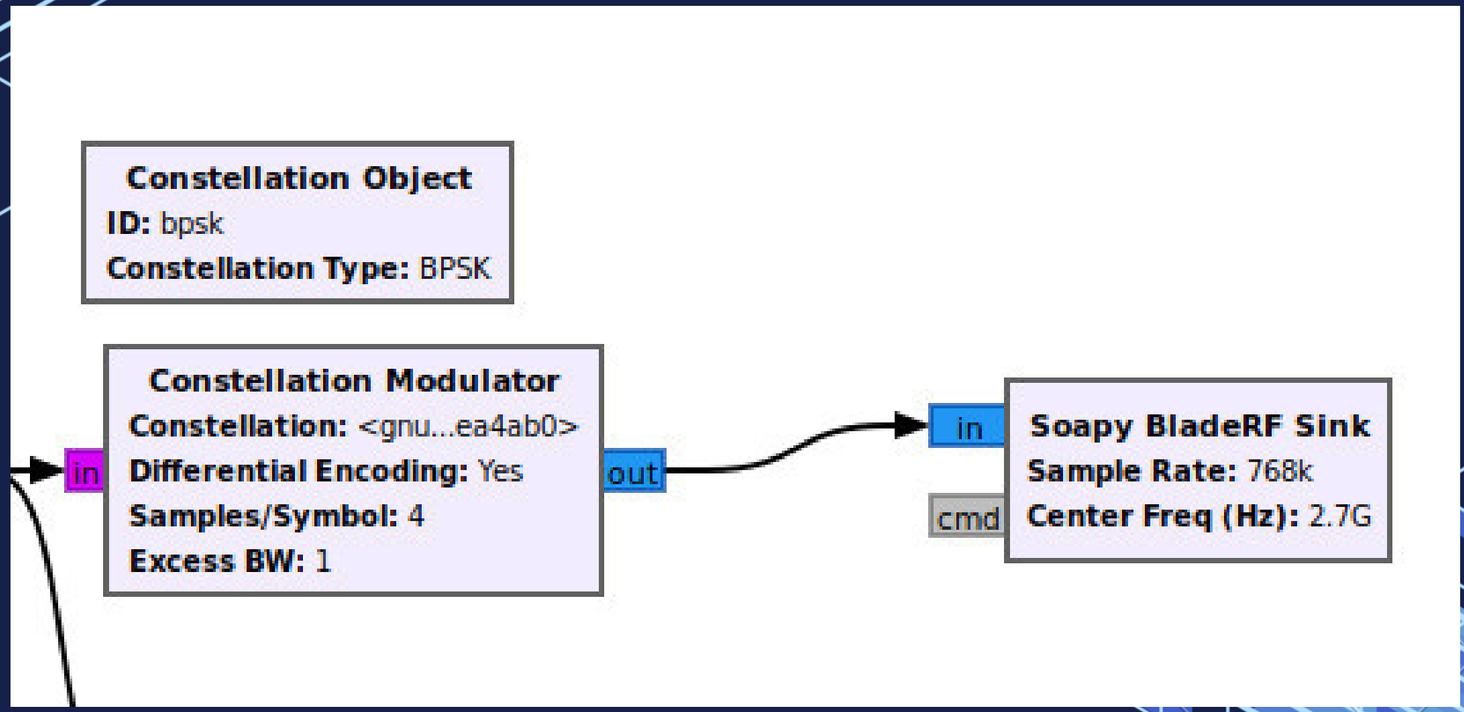


We identified from our previous knowledge what tasks we need to do in order to meet the given design requirements

TRANSMITTER



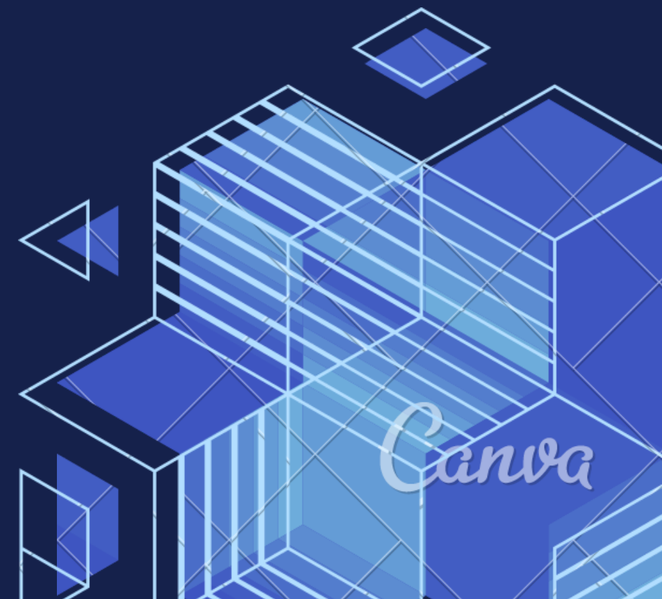
Modulator



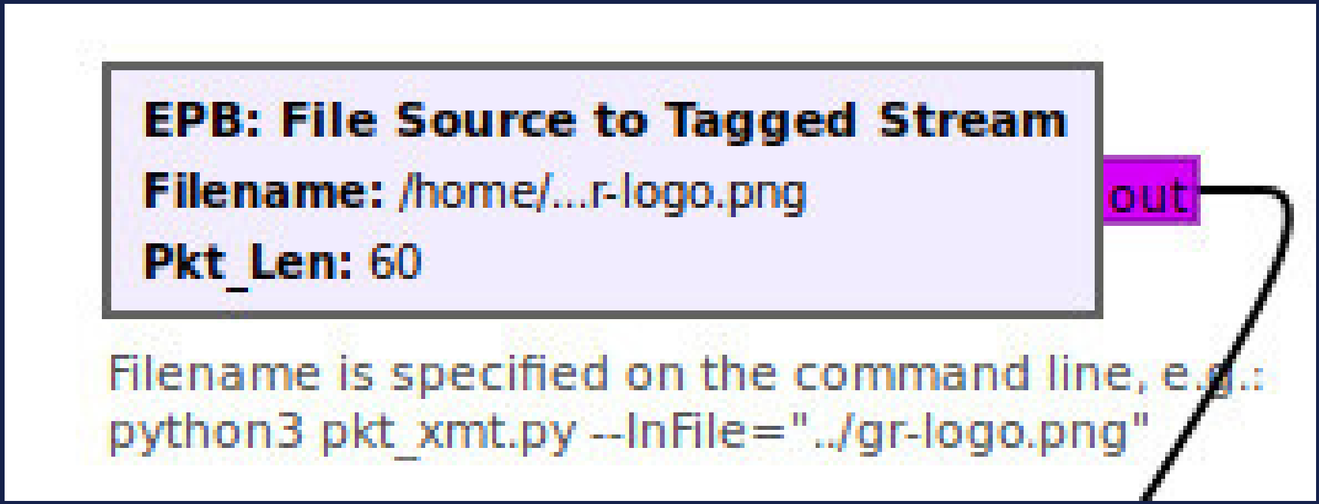
Differential encoding - (YES) - Instead of encoding each symbol independently, the difference between adjacent symbols is encoded.

Why BPSK

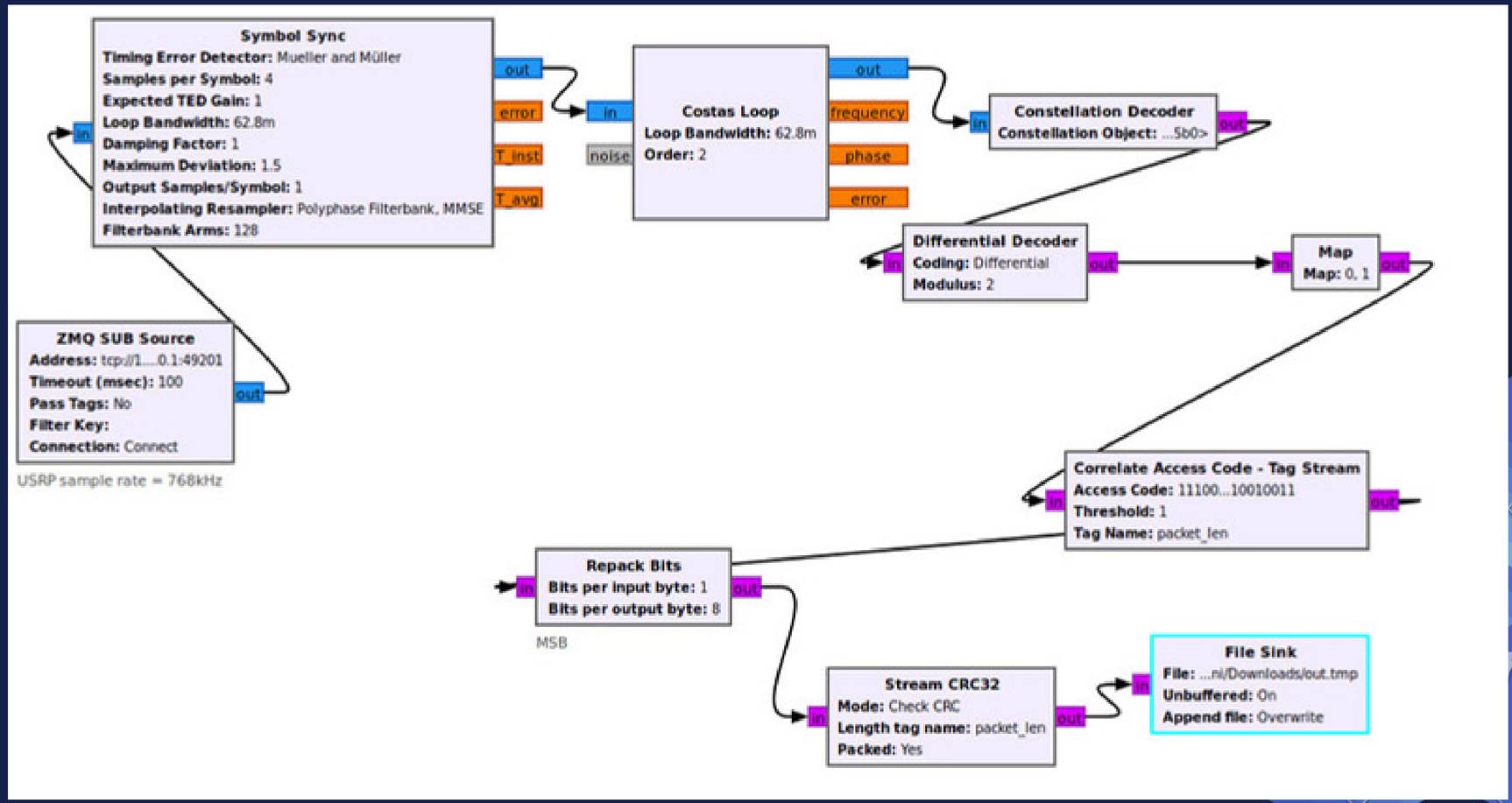
- Comparing to QPSK, information transmission rate is lower due to the high bandwidth.
- But bpsk has low error probability.
- BPSK is more power efficient.
- BPSK modulated data can travel long distance.



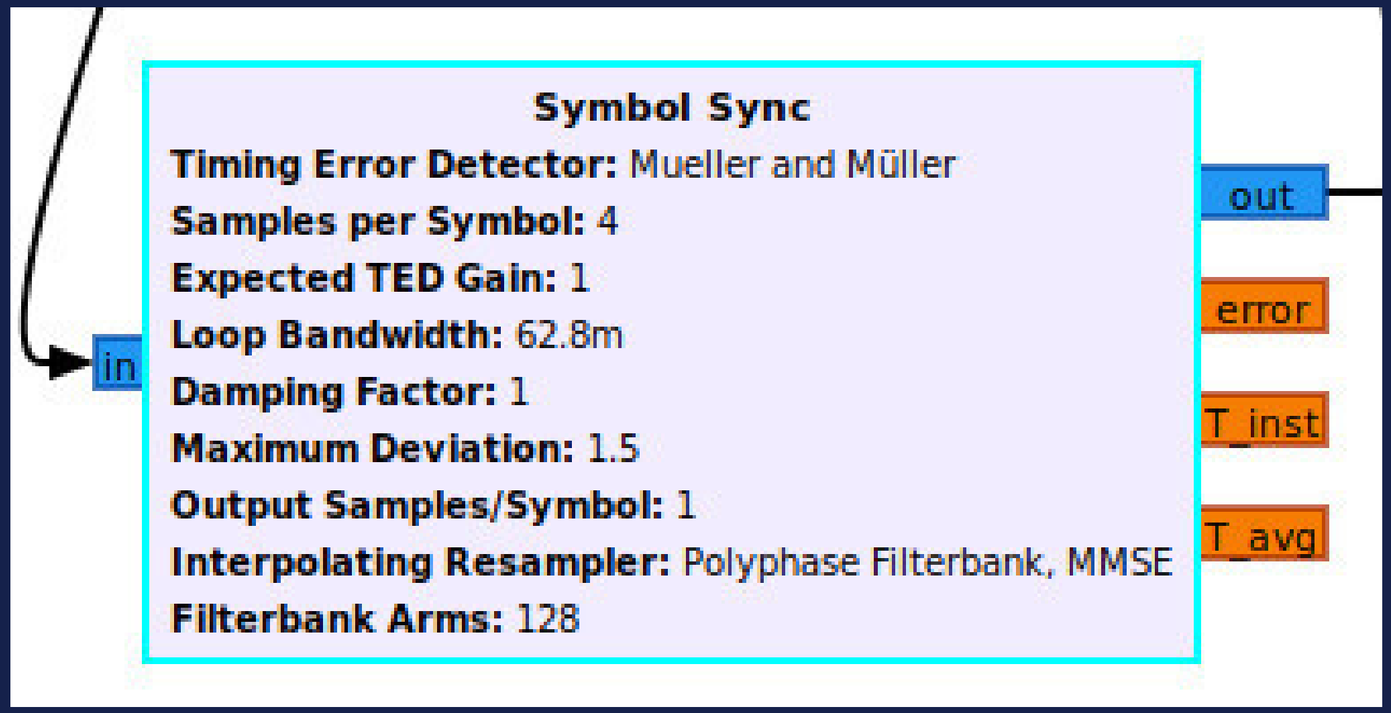
EPB File to tagged stream



RECEIVER

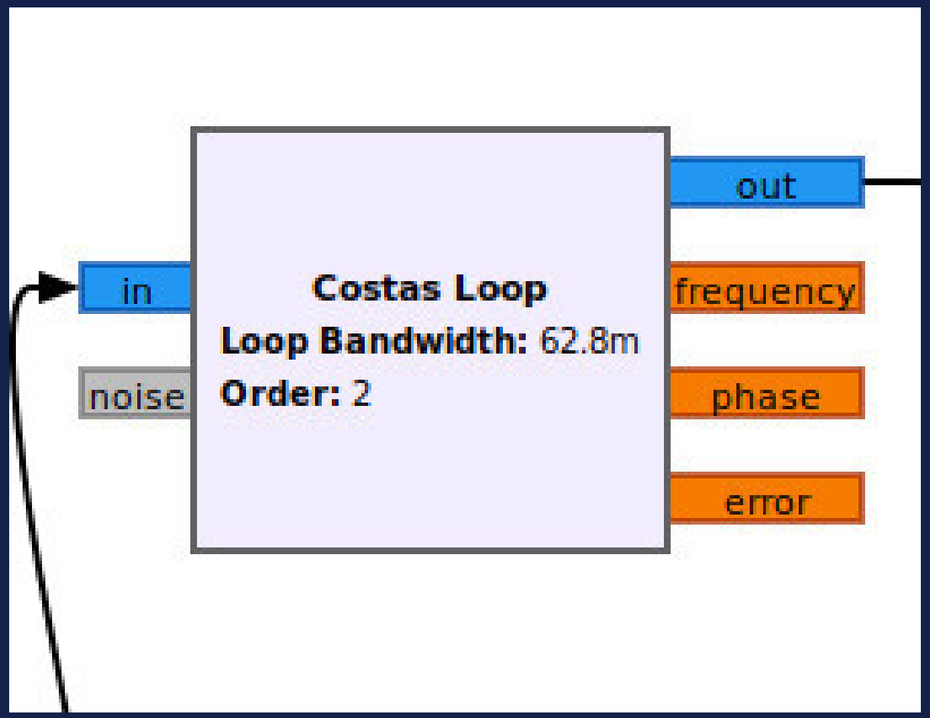

Symbol Sync
Timing Error Detector: Mueller and Müller
Samples per Symbol: 4
Expected TED Gain: 1
Loop Bandwidth: 62.8m
Damping Factor: 1
Maximum Deviation: 1.5
Output Samples/Symbol: 1
Interpolating Resampler: Polyphase Filterbank, MMSE
Filterbank Arms: 128
in
out
error
T_inst
T_avg
Costas Loop
Loop Bandwidth: 62.8m
Order: 2
noise
frequency
phase
Constellation Decoder
Constellation Object: ...5b0>
Differential Decoder
Coding: Differential
Modulus: 2
Map
Map: 0, 1
ZMQ SUB Source
Address: tcp://1....0.1:49201
Timeout (msec): 100
Pass Tags: No
Filter Key:
Connection: Connect
USRP sample rate = 768kHz
Correlate Access Code - Tag Stream
Access Code: 11100...10010011
Threshold: 1
Tag Name: packet_len
Repack Bits
Bits per input byte: 1
Bits per output byte: 8
MSB
Stream CRC32
Mode: Check CRC
Length tag name: packet_len
Packed: Yes
File Sink
File: ...ni/Downloads/out.tmp
Unbuffered: On
Append file: Overwrite

SYMBOL SYNC



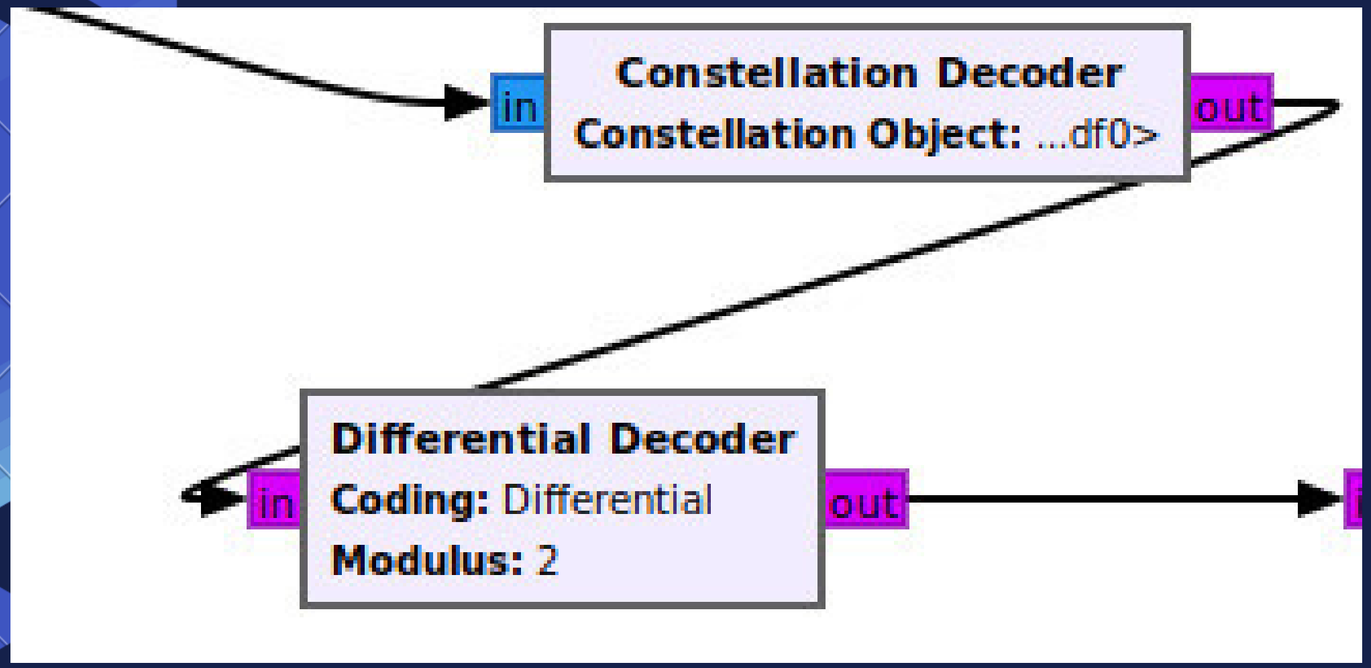
- It handles timing and frequency offset.
- It initially consists of a linear equalizer(To handle multipath propagation).

Costas Loop

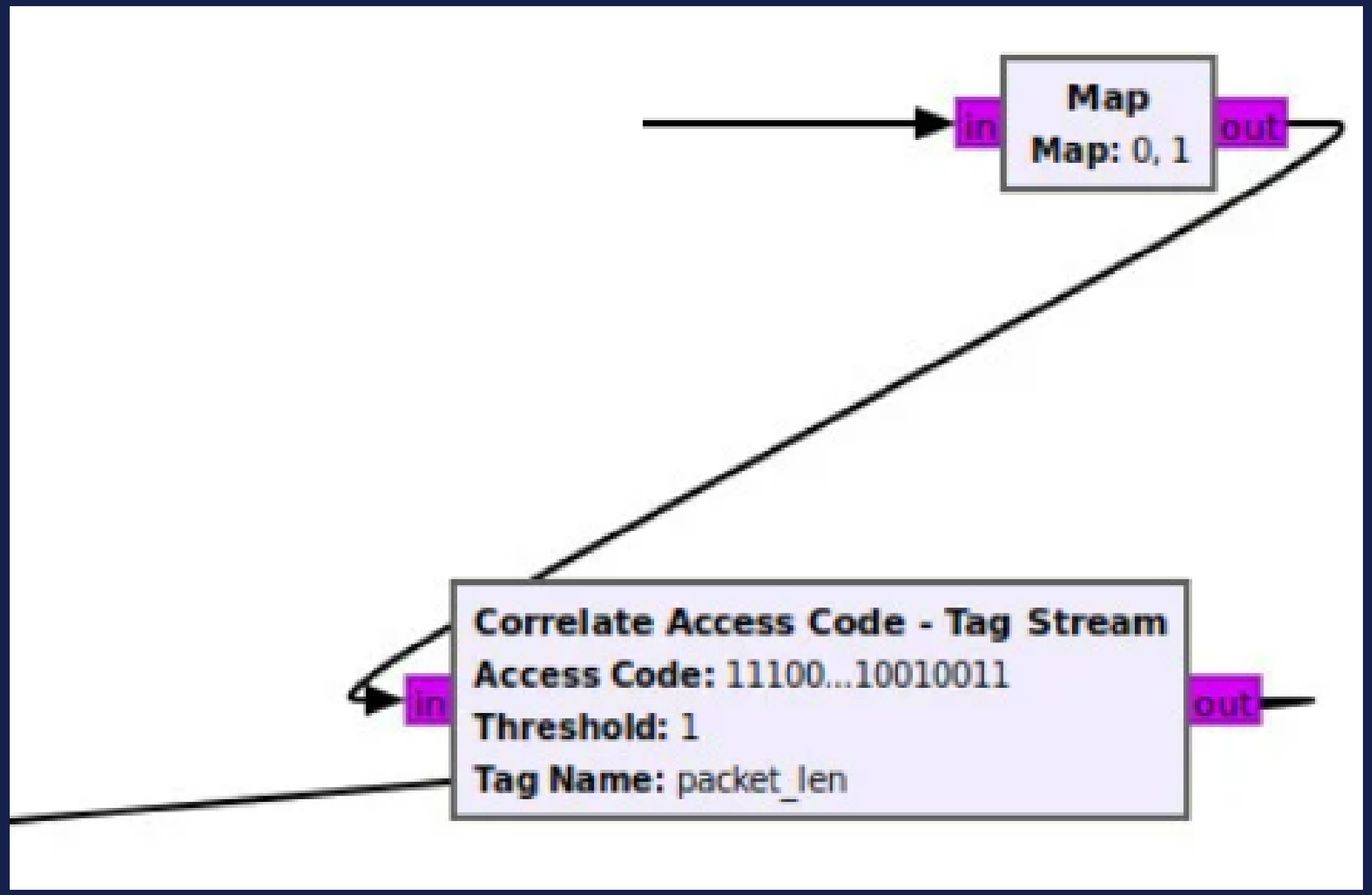


This block converts broadband into baseband

Constellation decoder and Differential Decoder

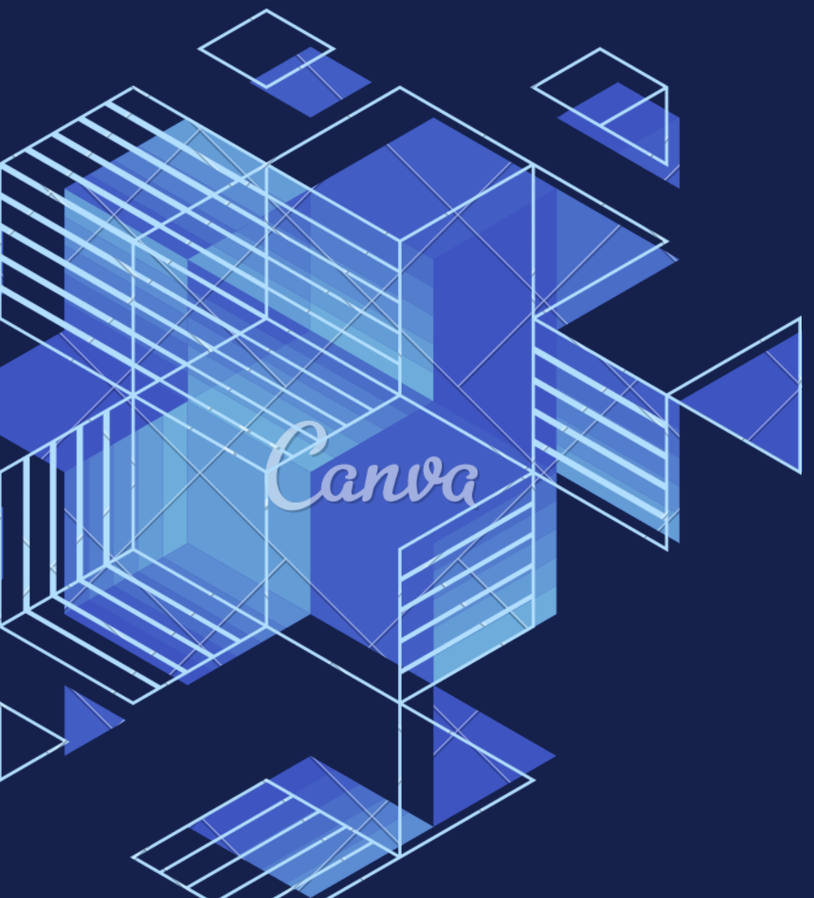


CAC Block


Map
Map: 0, 1
in
out
Correlate Access Code - Tag Stream
Access Code: 11100...10010011
Threshold: 1
Tag Name: packet_len
in
out

What we try to do

- Encryption
- CRC
- FEC



Encryption

- Symmetric Key encryption - XOR
- Transmitter and Receiver know 8-bit key
- Transmitter XOR bytes with that key and that is encryption
- The receiver then XOR the received bytes with that same key and that is recovery.

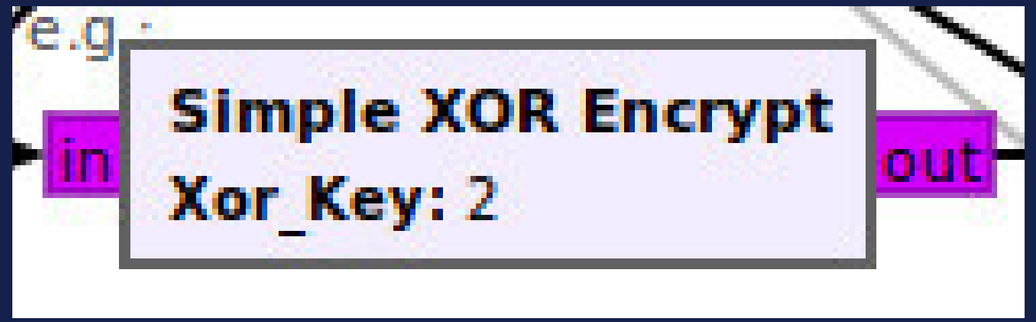
Open

epy_block_0_itecwotw.py
/tmp

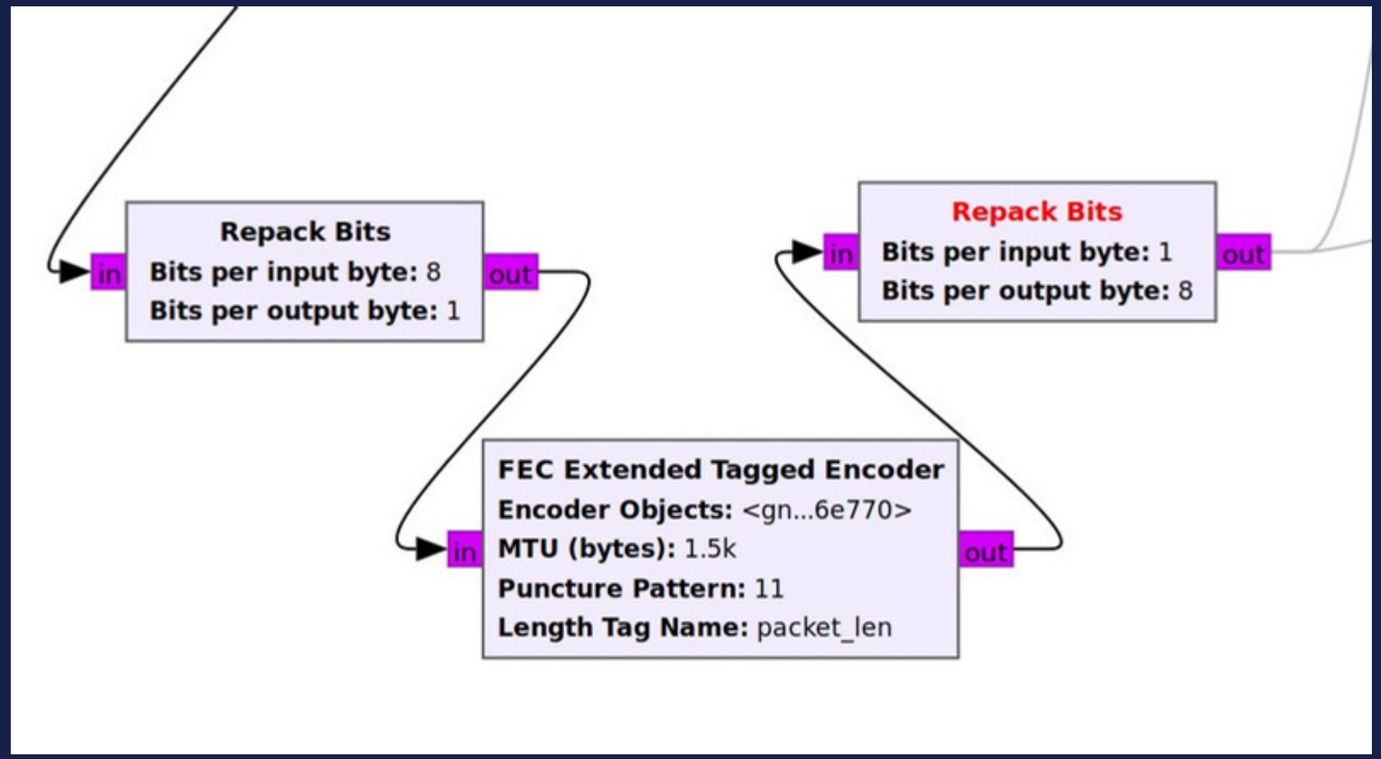
Save

```python
import numpy as np
from gnuradio import gr


class SimpleXOREncrypt(gr.sync_block):
    """Embedded Python Block example - simple XOR-based encryption"""

    def __init__(self, xor_key=0xAA):
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Simple XOR Encrypt',   # will show up in GRC
            in_sig=[np.uint8],
            out_sig=[np.uint8]
        )

        # XOR key for encryption
        self.xor_key = xor_key

    def work(self, input_items, output_items):
        """simple XOR-based encryption for every byte"""
        input_data = np.frombuffer(input_items[0], dtype=np.uint8)

        # Encrypt every byte using XOR
        encrypted_data = input_data ^ self.xor_key

        # Copy the encrypted data to the output buffer
        np.copyto(output_items[0], encrypted_data)

        return len(output_items[0])
```

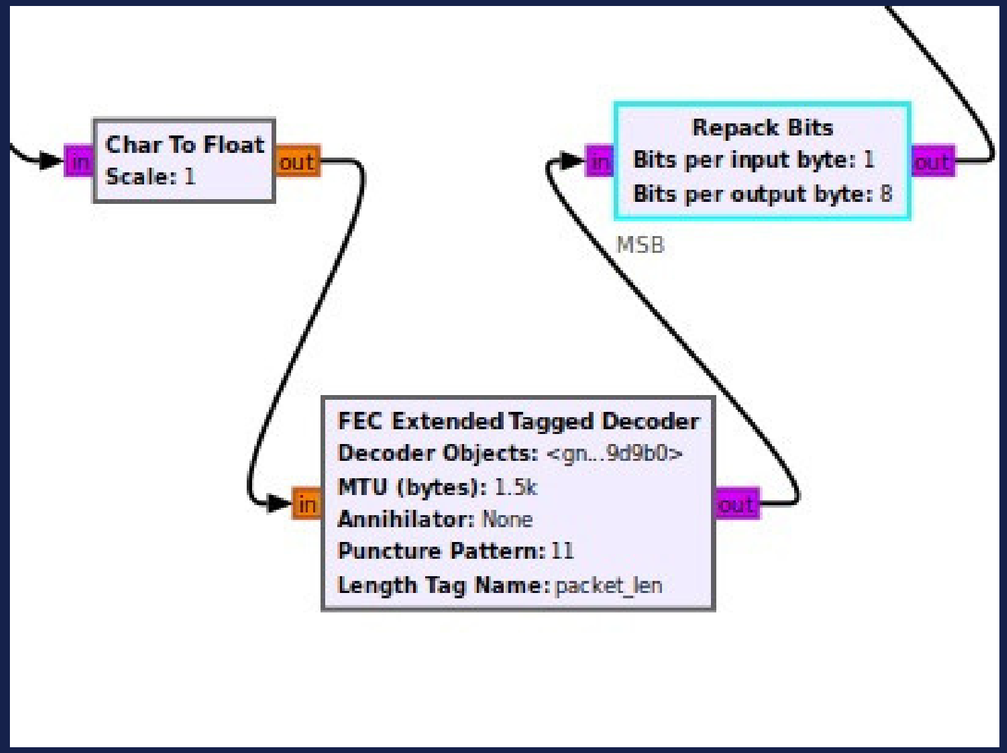
Loading file "/tmp/epy_block_0_itecwotw.py"... Python 2 Tab Width: 8 Ln 1, Col 1 INS



Forward Error Correction

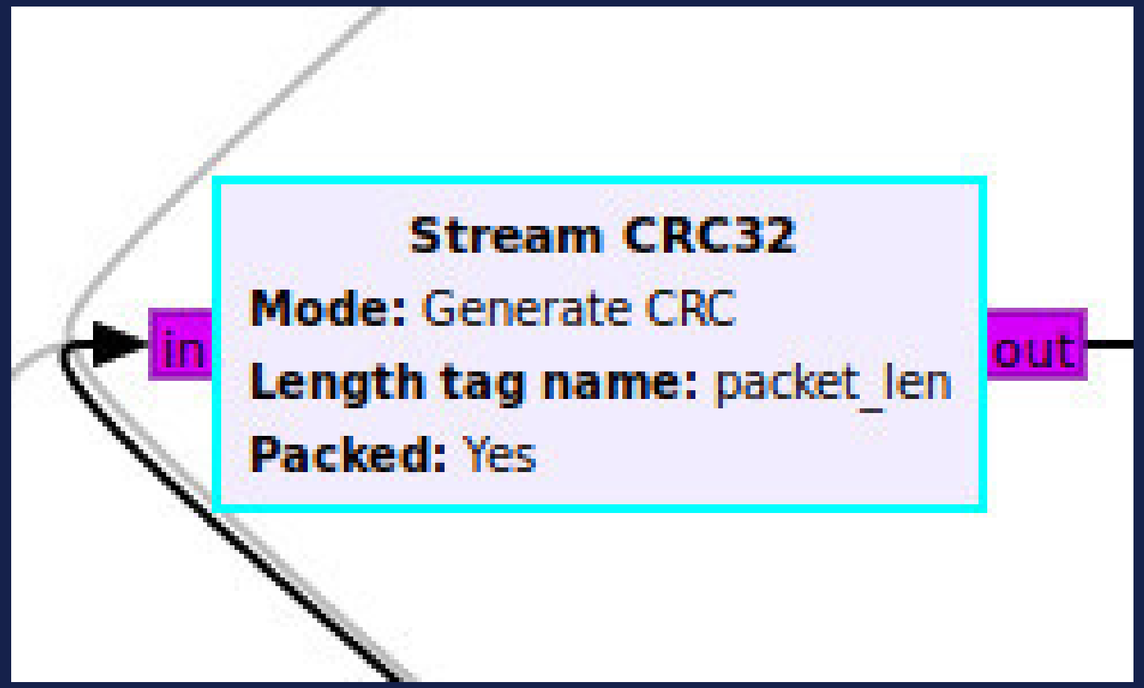


Transmitter side



Receiver side

CRC


Stream CRC32
Mode: Generate CRC
Length tag name: packet_len
Packed: Yes
in
out

Thank You !