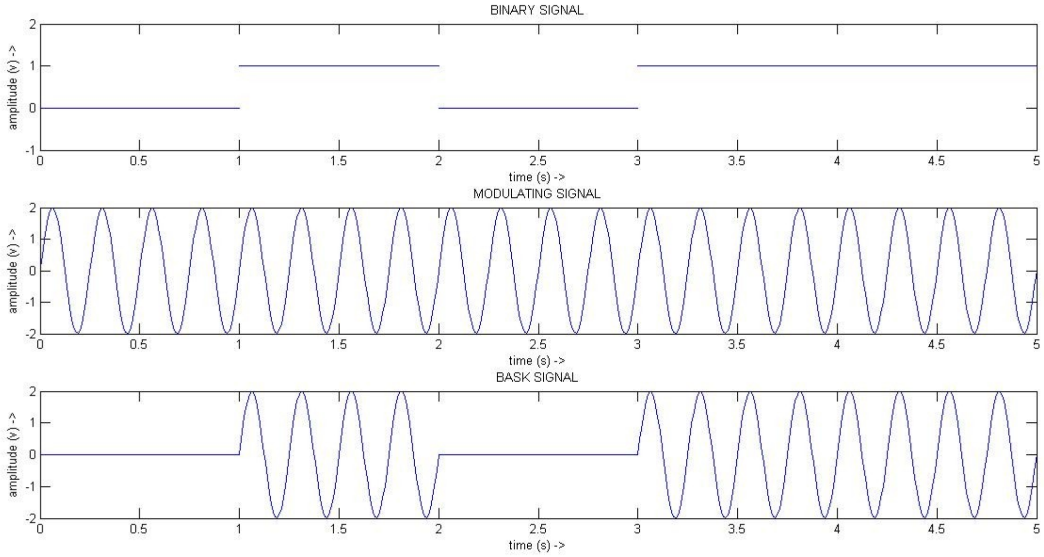
Name: Pintu Kumar	ID: 11640650

Communication Lab Report
Experiment - 05

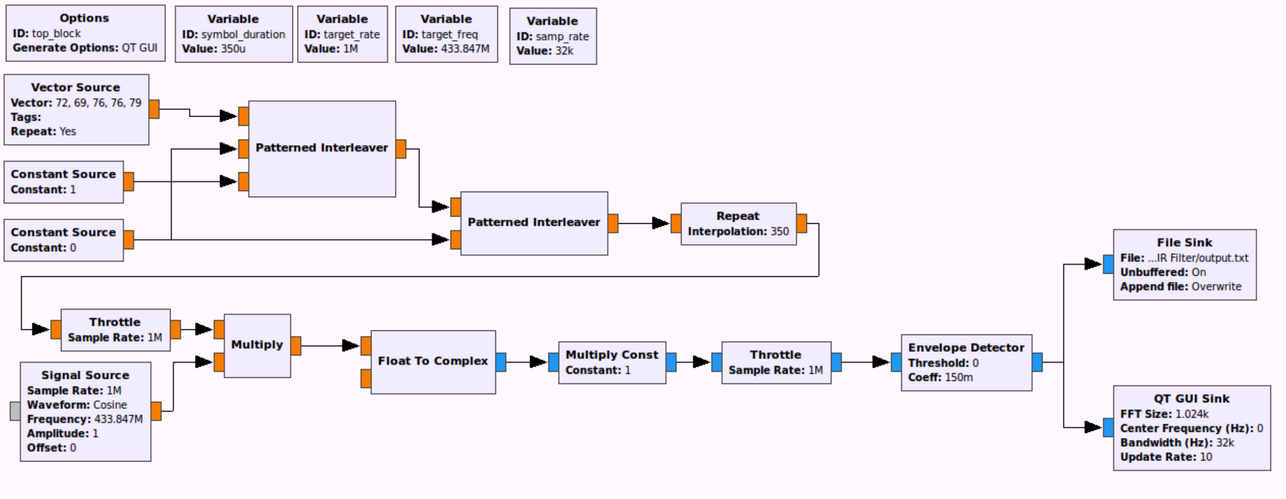
BASK Modulation Technique Using Embedded Python Block

Introduction

In BASK, the amplitude of the sinusoidal carrier signal is changed according to the message level (“0” or “1”), while keeping the phase and frequency constant.



Block Diagram For BASK


Options
ID: top_block
Generate Options: QT GUI

Variable
ID: symbol_duration
Value: 350u

Variable
ID: target_rate
Value: 1M

Variable
ID: target_freq
Value: 433.847M

Variable
ID: samp_rate
Value: 32k

Vector Source
Vector: 72, 69, 76, 76, 79
Tags:
Repeat: Yes

Constant Source
Constant: 1

Constant Source
Constant: 0

Patterned Interleaver

Patterned Interleaver

Repeat
Interpolation: 350

Throttle
Sample Rate: 1M

Signal Source
Sample Rate: 1M
Waveform: Cosine
Frequency: 433.847M
Amplitude: 1
Offset: 0

Multiply

Float To Complex

Multiply Const
Constant: 1

Throttle
Sample Rate: 1M

Envelope Detector
Threshold: 0
Coeff: 150m

File Sink
File: ...IR Filter/output.txt
Unbuffered: On
Append file: Overwrite

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 32k
Update Rate: 10

Blocks:

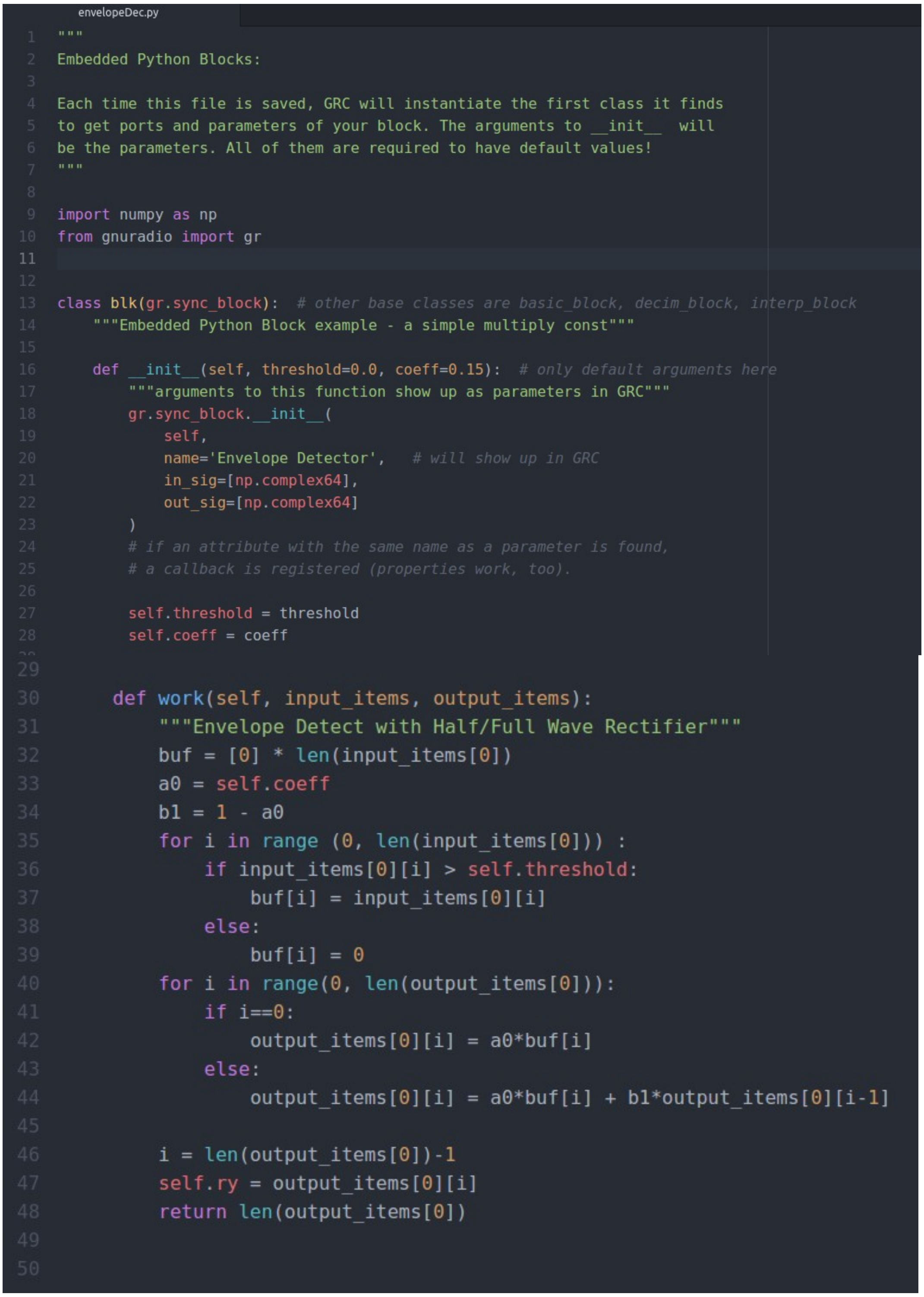
Vector-source - This block produces a stream of samples based on an input vector .We can also send a string using the vector source but first we need to create an array of ASCII values of each character that we are sending. We define the vector “HELLO”. The ASCII values are 72, 69, 76, 76, 79 respectively of “HELLO” word.

Patterned Interleaver – This block takes data from the input, whose index is first in the pattern, then the data from the input whose index is specified second.

File Sink- show the way before the file which will contain the result of the work of the “Patterned Interleaver” block, as well as turn off the buffering during the output, plus we specify to overwrite our file.

Envelope Detector- It takes a (relatively) high-frequency amplitude modulated signal as input and provides an output which is the envelope of the original signal. It is created using “Embedded Python Block”. It has Rectifier + Low Pass filter.
A low-pass filter (LPF) is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The exact frequency response of the filter depends on the filter design.

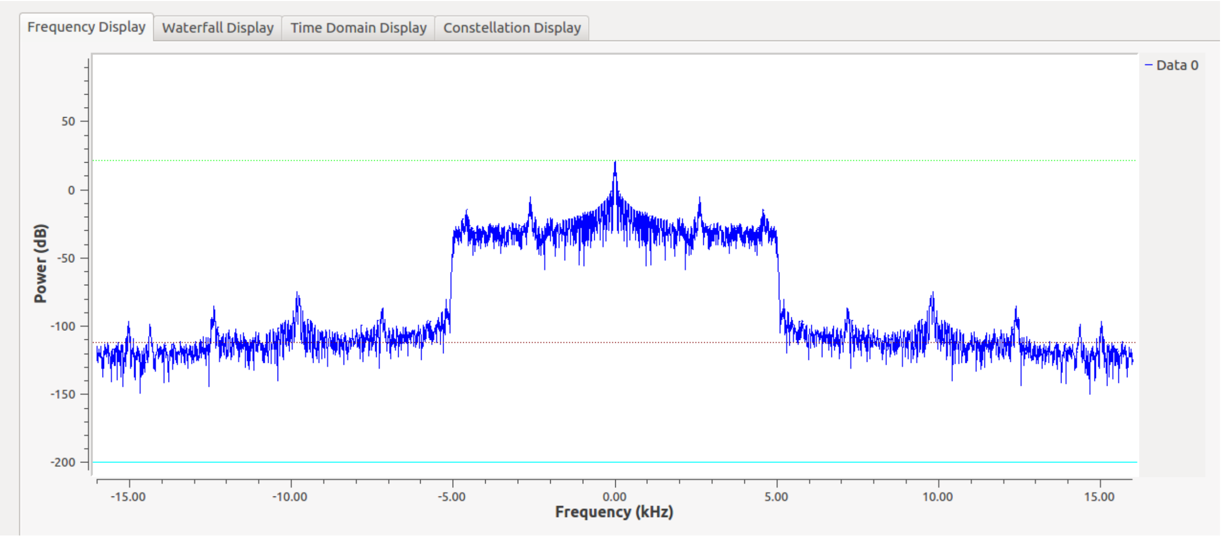
Python Code (Low Pass Filter)

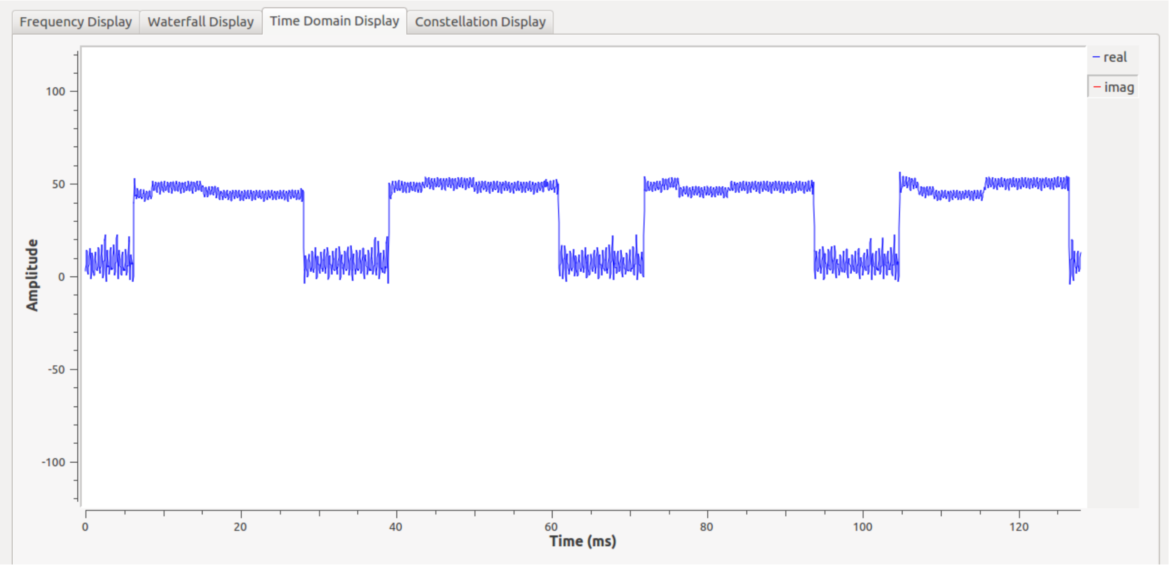


envelopeDec.py

```python
"""
Embedded Python Blocks:

Each time this file is saved, GRC will instantiate the first class it finds
to get ports and parameters of your block. The arguments to __init__  will
be the parameters. All of them are required to have default values!
"""

import numpy as np
from gnuradio import gr


class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    """Embedded Python Block example - a simple multiply const"""

    def __init__(self, threshold=0.0, coeff=0.15):  # only default arguments here
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Envelope Detector',   # will show up in GRC
            in_sig=[np.complex64],
            out_sig=[np.complex64]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too).

        self.threshold = threshold
        self.coeff = coeff

    def work(self, input_items, output_items):
        """Envelope Detect with Half/Full Wave Rectifier"""
        buf = [0] * len(input_items[0])
        a0 = self.coeff
        b1 = 1 - a0
        for i in range (0, len(input_items[0])) :
            if input_items[0][i] > self.threshold:
                buf[i] = input_items[0][i]
            else:
                buf[i] = 0
        for i in range(0, len(output_items[0])):
            if i==0:
                output_items[0][i] = a0*buf[i]
            else:
                output_items[0][i] = a0*buf[i] + b1*output_items[0][i-1]

        i = len(output_items[0])-1
        self.ry = output_items[0][i]
        return len(output_items[0])


```

Lowpass Filter Output in Frequency & Time Domain





Observation

- Any modulated signal has a high frequency carrier. The binary signal when ASK modulated, gives a zero value for Low input while it gives the carrier output for High input.
- A envelope detector block is made using python programming and an Embedded Python Block.
- We could not make a perfectly ideal low pass filter.
- The text sent was successfully recovered and since this is a simulation, there were no errors when no noise was added.
- The received message signal was routed to a text file on disk.
- It combines the incoming bytes in chunks of 8 and emits a series of bytes consisting of the LSB of each input byte. So, 00000001 00000000 00000001 00000001 00000000 becomes 10110001. This is the recovered signal.

Conclusion

The signal is successfully modulated using BASK and recovered. But on using our custom low-pass filter block, we observed distortion in the received signal. Since our low-pass filter is only an approximation of ideal LPF using FIR filter, so we are only able to decrease the gain of high frequency components instead of completely removing them.