Octave and LimeSDR

Danny Webster




Table of Contents

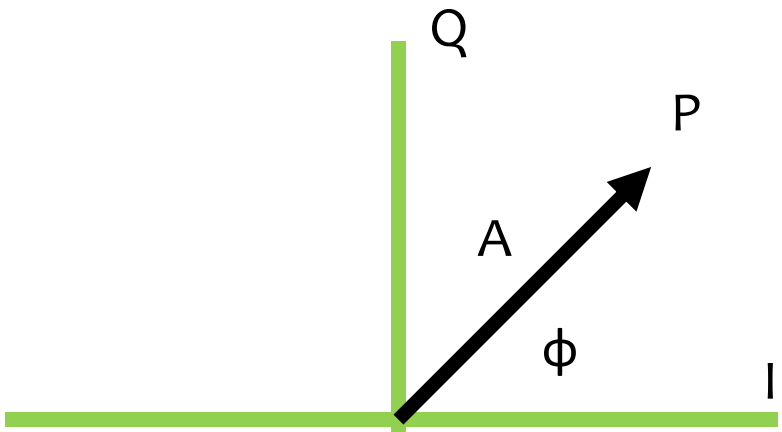
Getting started with Octave

- **Install Octave 4.0.x**
- **Build Octave-LimeSDR**
  - Windows
    - octave
    - run build.m
  - Linux
    - cd~/LimeSuite/octave/
    - make
- **Copy files into each project**
  - LoadLimeSuite.m
  - LimeSuite.oct
  - LimeSuite.dll (windows)
- **Start Octave**
  - Change directory
  - run build.m % do for each
  - run SSB.m
  - run ASK.m
  - run FSK.m
  - run NumptyText.m
  - run NumptyText2.m
  - run NumptyText3.m

Lime microsystems

Representing Modulation in Octave

- **AM, FM and PM can be represented by**
- **Cartesian co-ordinates {x,y}**
  - In Radio, called Inphase and Quadrature
  - Real, Imaginary (Complex Numbers).
  - Programming real() imag()
  - Note i and j mean the same.
- **Polar co-ordinates (Vector Modulation)**
  - Natural language for describing Amplitude and Phase modulation.
  - Amplitude A, Phase φ
  - Phase is usually in radians not degrees.
  - Programming abs() arg() exp(i*)
  - Frequency is a phase that changes at a constant rate.



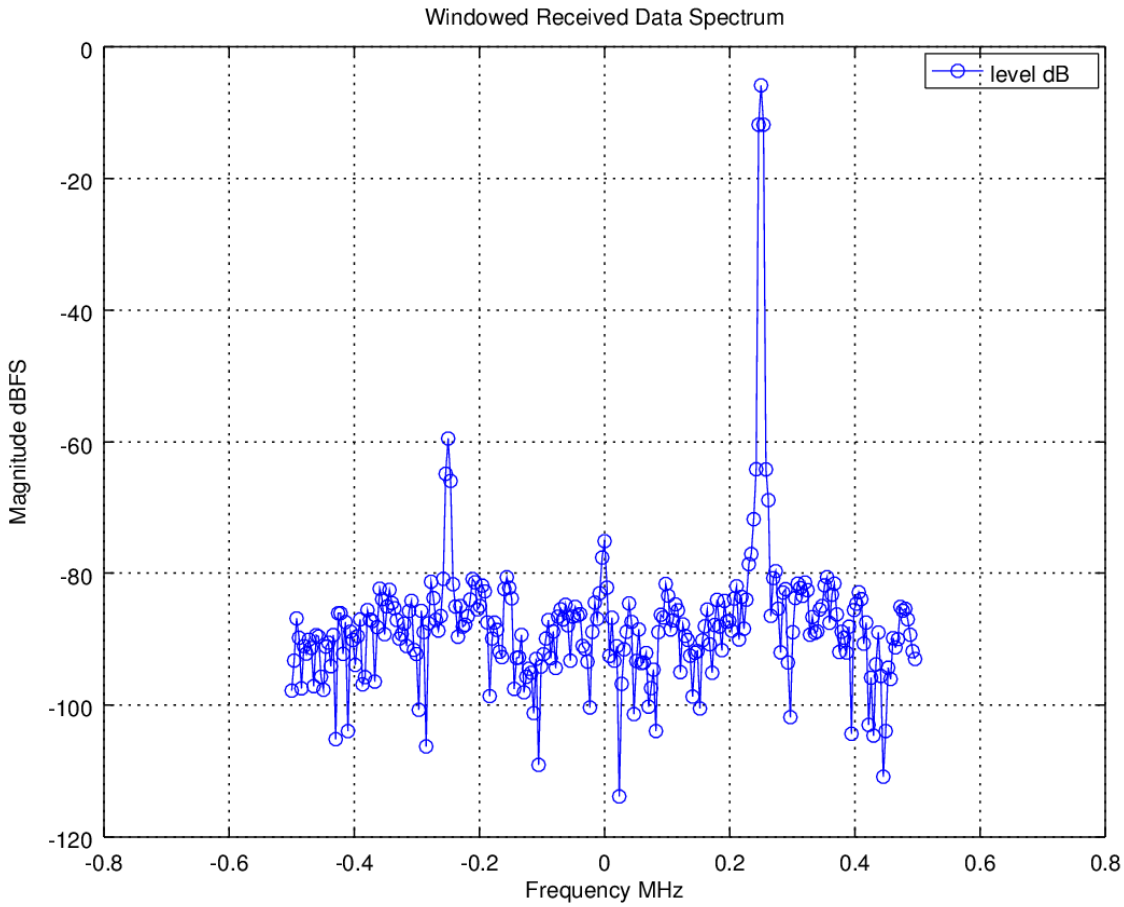
$$P(t) = I(t) + jQ(t) = A(t)exp\left(j\emptyset(t)\right)$$

$$P(t) = A(t)\left(cos\left(j\emptyset(t)\right) + jsin\left(j\emptyset(t)\right)\right)$$

$$f = \frac{1}{2\pi}\frac{d\emptyset(t)}{dt}$$

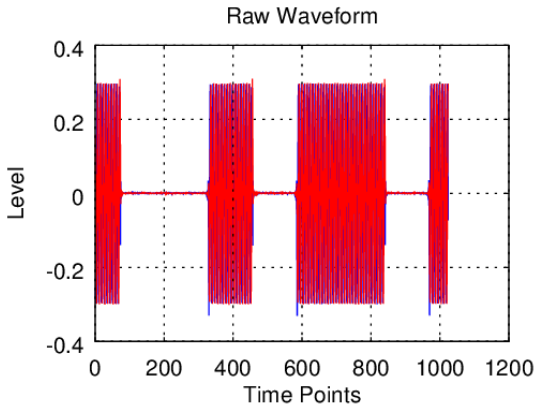
Single Sideband Demo

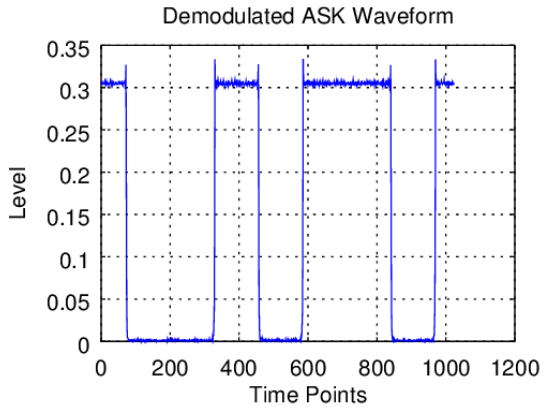
- **Demonstrates**
  - Octave link with LimeSDR
  - SSB Generation
  - FFT with Hanning Window
- **Settings File**
  - .ini generated in LimeSuite
  - 866MHz -50dBm for table top license exempt transmission.

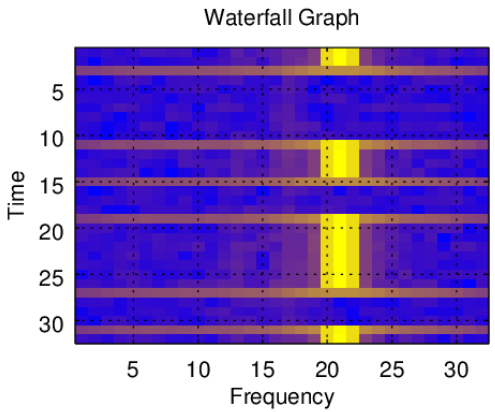


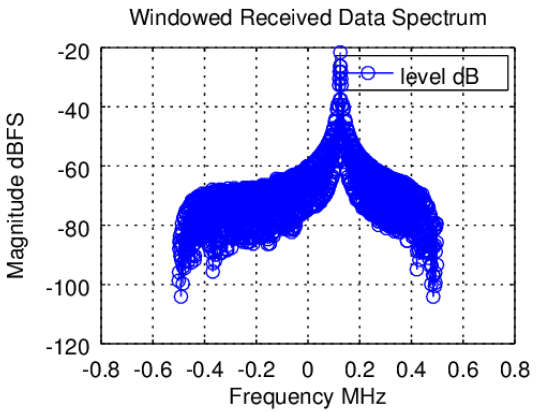
Amplitude Shift Keying Demo

- **Demonstrates**
  - Octave link with LimeSDR
  - ASK generation
  - AM/ASK demodulation.
  - Waterfall type graphs
- **Settings File**
  - .ini generated in LimeSuite
  - 866MHz -50dBm for table top license exempt transmission.



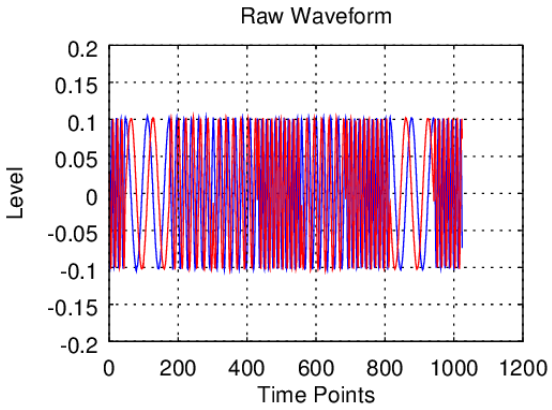


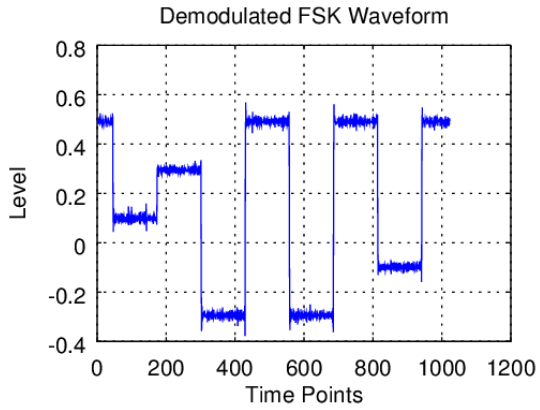


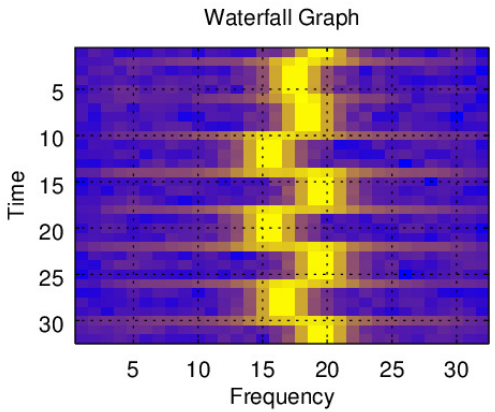


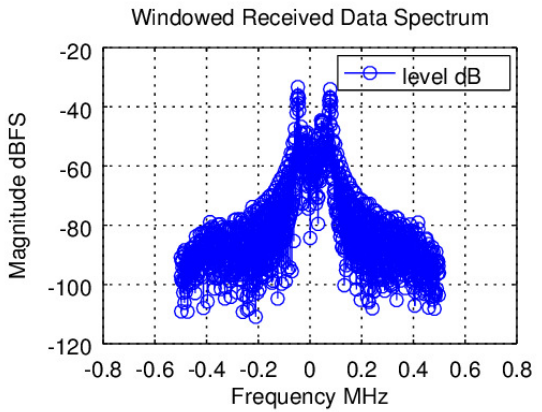
Frequency Shift Keying Demo

- **Demonstrates**
  - Octave link with LimeSDR
  - FSK generation
  - FM/FSK demodulation.
  - Waterfall type graphs
- **Settings File**
  - .ini generated in LimeSuite
  - 866MHz -50dBm for table top license exempt transmission.



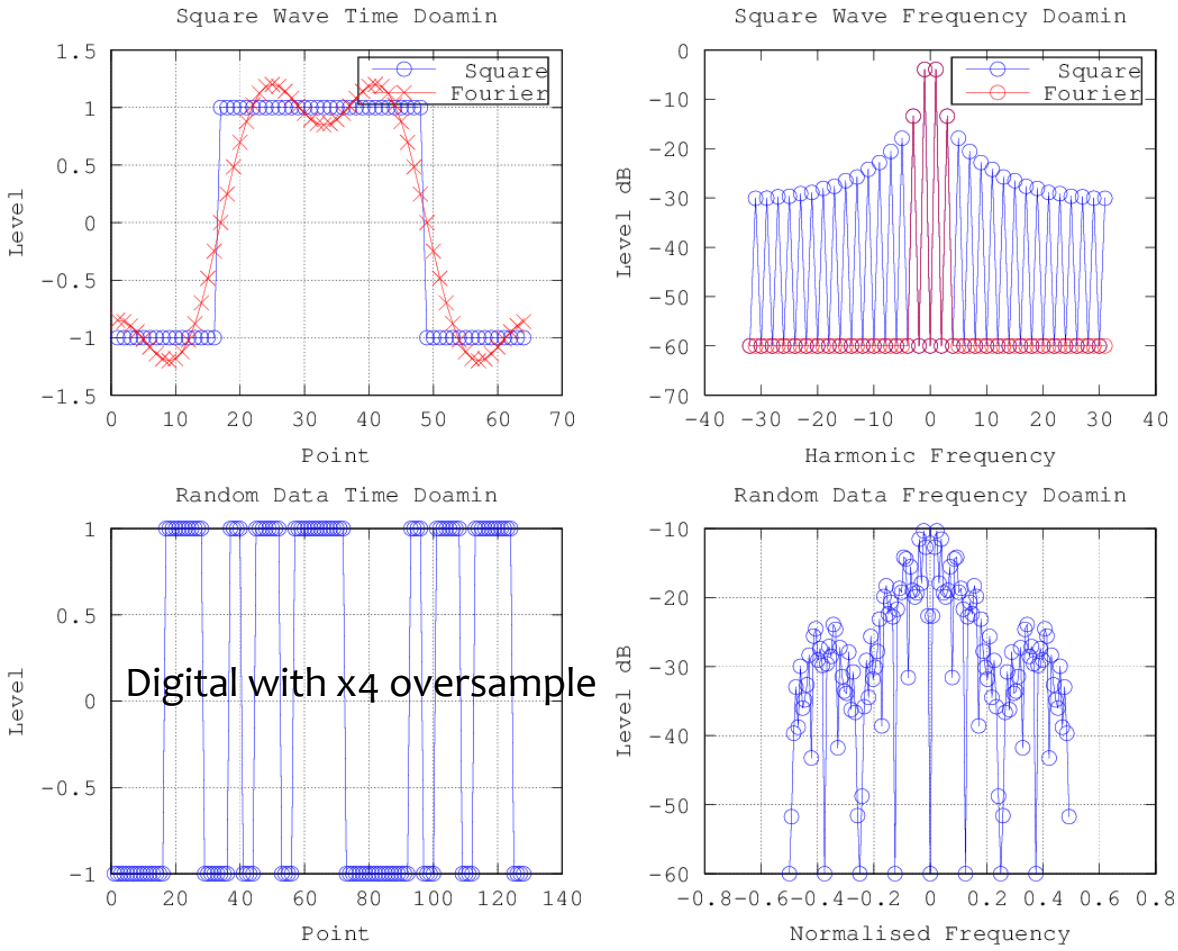






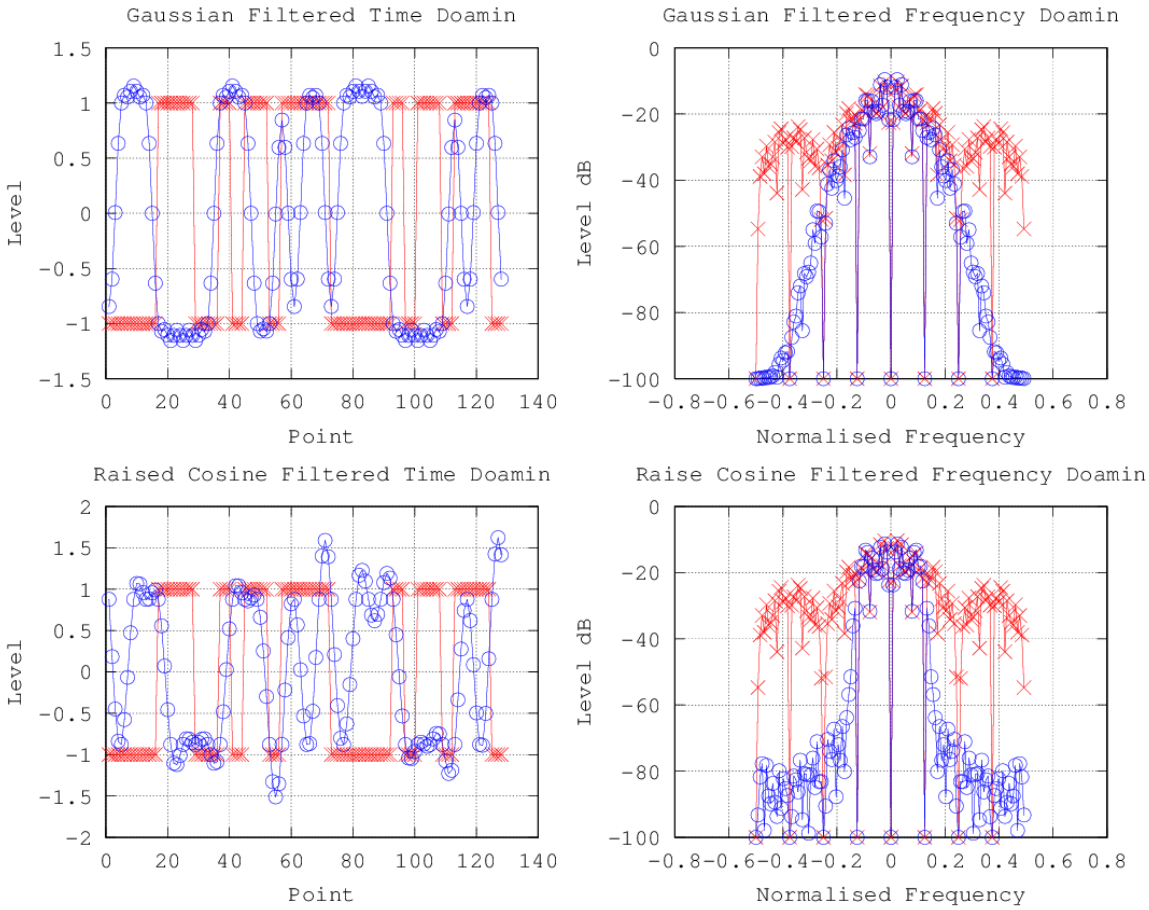
Digital Modulation and Bandwidth

- **Data changes with time.**
  - Looks like sequence of square waves.
- **Fourier Analysis**
  - Complex signals are made of harmonics.
  - Reducing harmonics lead to more gentle rise and fall behaviour.
- **Harmonics of random data**
  - Adjacent channel interference.
  - Must be filtered.



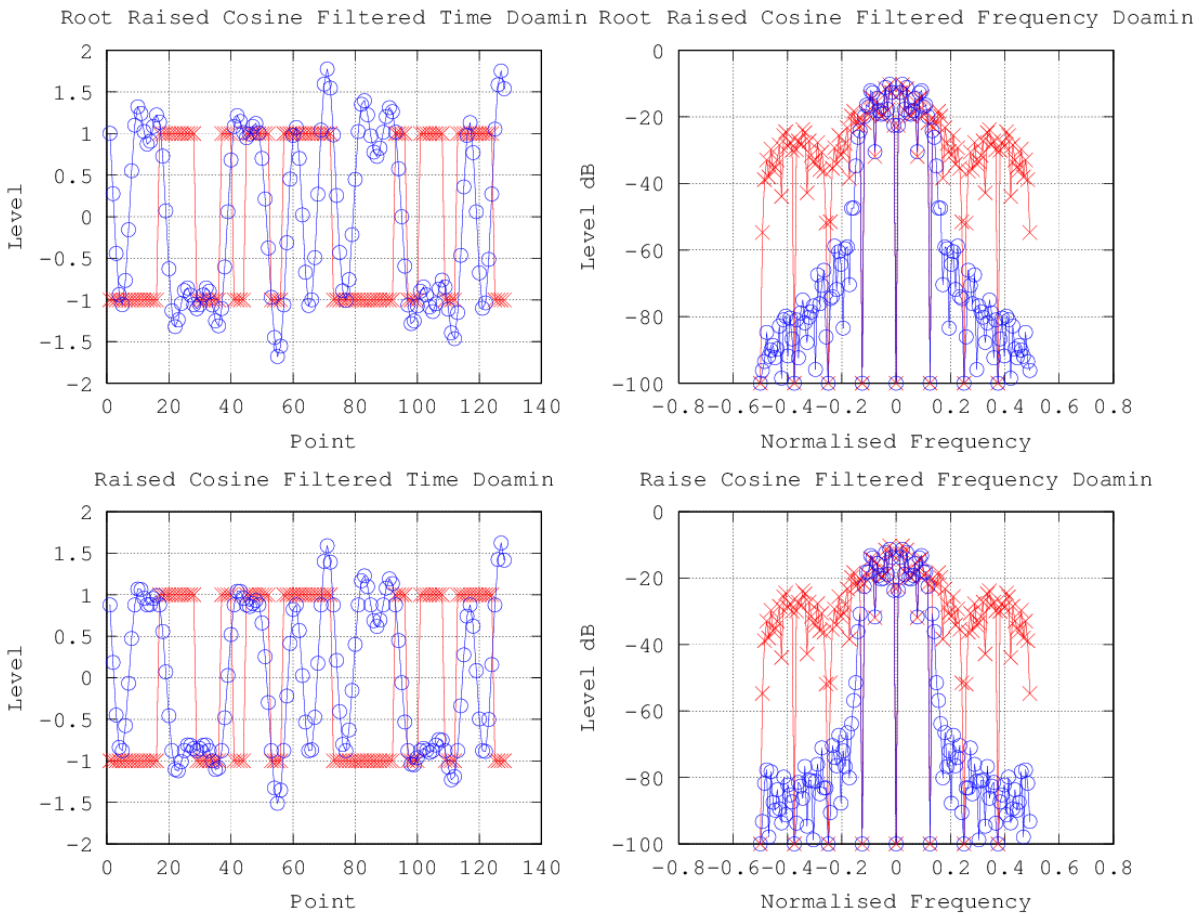
Pulse Shaping Filters and Digital Modulation

- **Pulse shaping has two purposes.**
- **To reduce the bandwidth of a digital signal.**
  - Reduce adjacent channel interference.
  - Increase capacity of available bandwidth.
- **To minimise interference from previous bits.**
  - Intersymbol interference (ISI)
- **Two common types**
  - Gaussian (2G and Bluetooth)
  - Root Raised Cosine (3G)
  - Implemented as FIR Filters



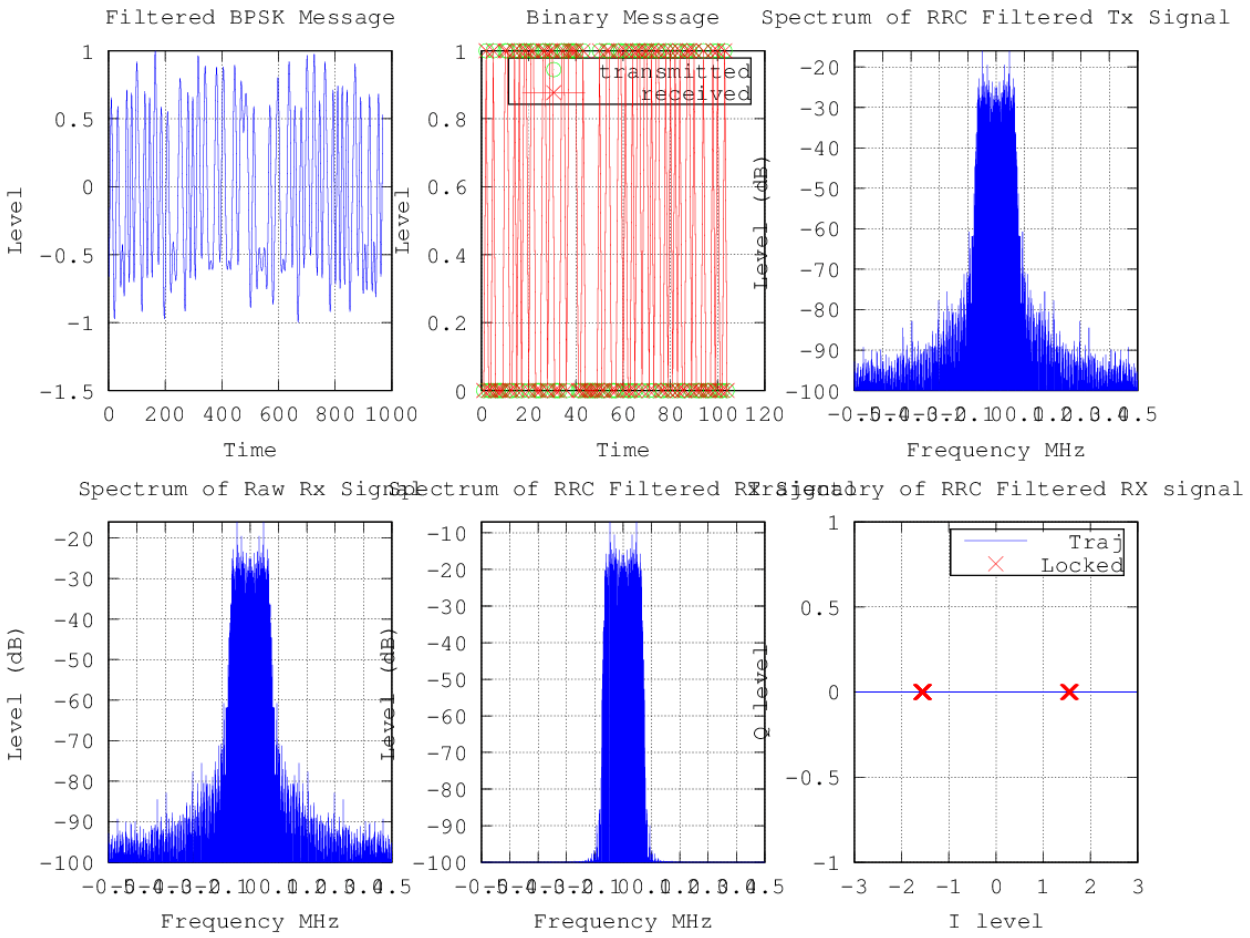
Matched Pulse Shaping Filters

- **Raised Cosine filter split into two Filters**
  - 2x Root Raised Cosine.
  - Combined have same effect as original Raised Cosine filter on the data.
- **RRC used before TX DAC**
  - Reduce noise in adjacent channels when transmitting.
- **RRC used after RX ADC**
  - Prevent receiver adding noise from adjacent channels.



NumptyText1 Demo

- **Demonstrates**
  - Transmit ASCII message from TX to RX
  - ASCII /Binary conversion
  - Auto Correlation Function for sync detection and phase correction
  - BER Calculation
  - Root Raised Cosine Filtering for pulse shaping
- **Variations**
  - Try with SMA Adapter
  - Try with Antenna
  - Change Antenna angle



NumptyText2 Demo

- **Demonstrates**
  - ASCII to Binary conversion
  - Forward Error Correction with BCH(7,4,1) code
    - Works easily with 8 bit ASCII
  - BER Calculation
  - Auto Correlation Function for sync detection and phase correction
  - Root Raised Cosine Filtering for pulse shaping
- **Variations**
  - Try with unfavourable antenna alignment
  - Try with BCH(15,7,2)
    - with 7 bit ASCII
    - 2 bit errors
  - Try with BCH(63,21,2)
    - With padded 7 bit ASCII
    - 2 bit errors
  - Try with BCH(63,16,3)
    - With padded 8 bit ASCII
    - 3 bit errors
  - Try unscrewing one antenna!!!

Lime microsystems

NumptyText3 Demo

- **Demonstrates**
  - ASCII to Binary conversion
  - Forward Error Correction with BCH(7,4,1) code
    - Works easily with 8 bit ASCII
  - BER Calculation
  - CDMA Type Spread Spectrum
    - Spread code unique random
    - Pilot channel orthogonal to data
    - Code spreading and despreading
  - Using ACF for Frame synchronisation.
  - Root Raised Cosine Filtering for pulse shaping
- **Variations**
  - Try unscrewing both antennas!!!!
    - Alter spread factor to restore link!

Lime microsystems