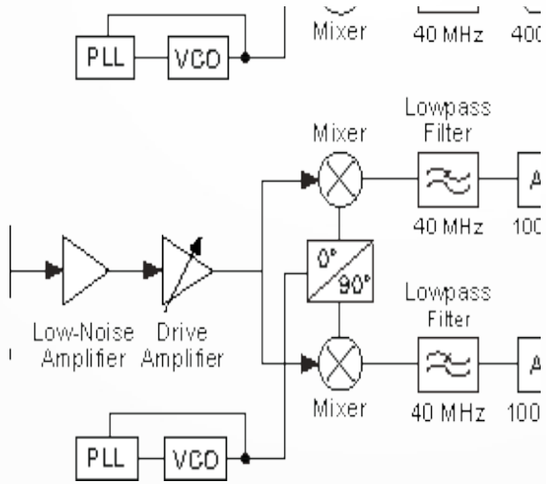
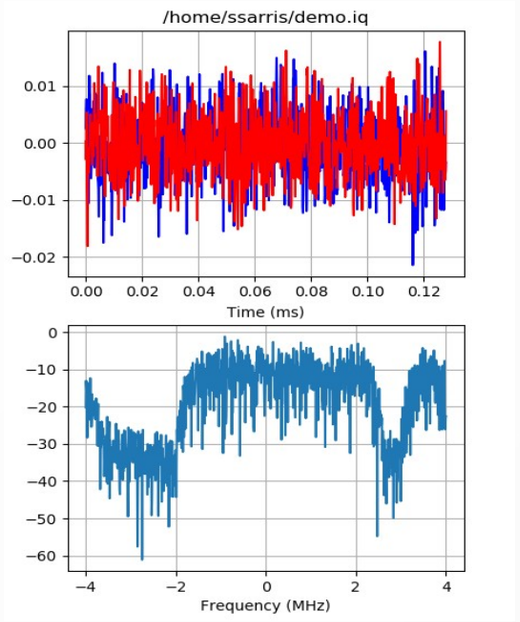
RF Quadrature (I/Q) Signals

Math Model
Implementation in Hardware
Structure of Digital Data
Implementation in Software

Spiro Sarris. 01 /2019

$$A \cdot \cos\left(\omega \cdot t + \phi\right)$$





Links to Download

- This Presentation

https://github.com/spiro-sarris/pres/blob/master/rf_iq.pdf

- Python code - generate signals and display

https://github.com/spiro-sarris/utils/blob/master/iqdemo.py

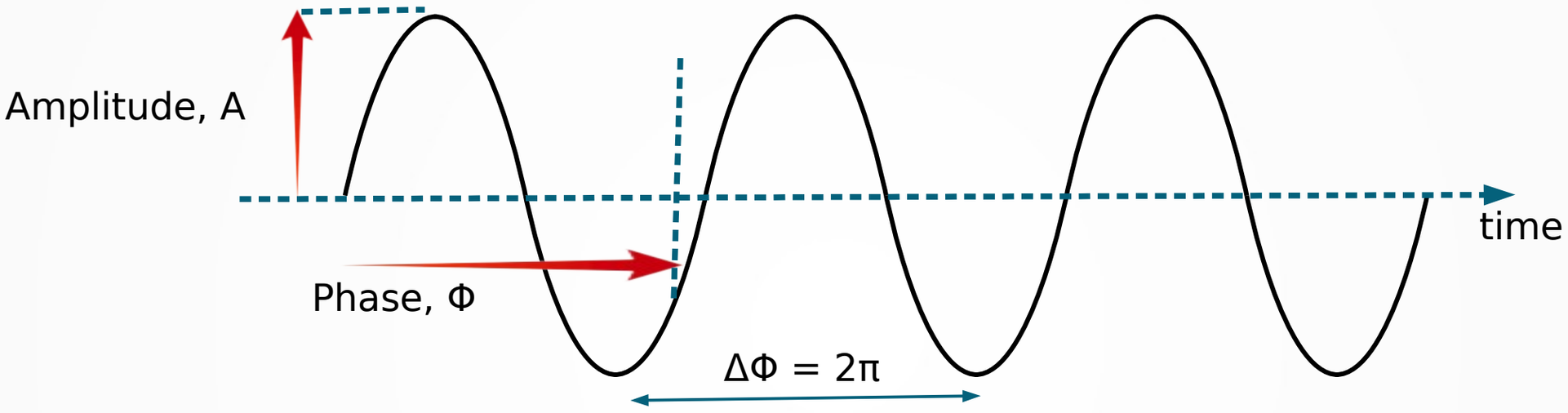
- Python code – read raw .iq file from GNUradio

https://github.com/spiro-sarris/utils/blob/master/iqstep.py

Outline

1. Amplitude, phase. Why is it interesting?
2. Euler’s formula
3. Math model of signal
4. Implementation of model in receiver hardware
5. Analog to digital conversion (ADC)
6. Convenience in software programming languages
7. Implementation of model in software
8. Compare I/Q sampling and real-value sampling

1. Amplitude, Phase

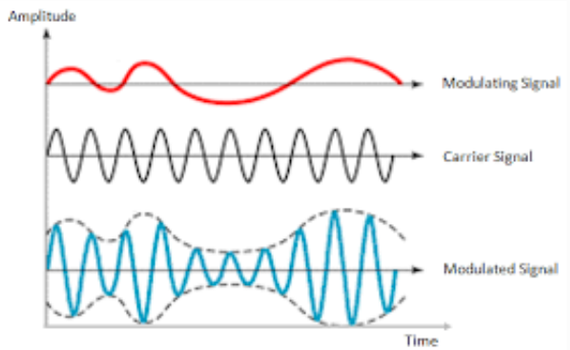


- ΔΦ = 2π radians for 1 cycle in time and for 1 wavelength (λ) [m] in space
- Angular frequency (ω) [radians/sec] is the rate of change of phase vs. time $\omega = \frac{d\phi}{dt}$
- Temporal frequency (f) [Hz] is the number of cycles per second $f = \frac{\omega}{2\pi}$

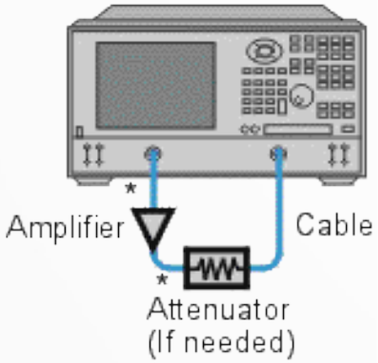
1. Amplitude

RSSI – Received Signal Strength Indicator

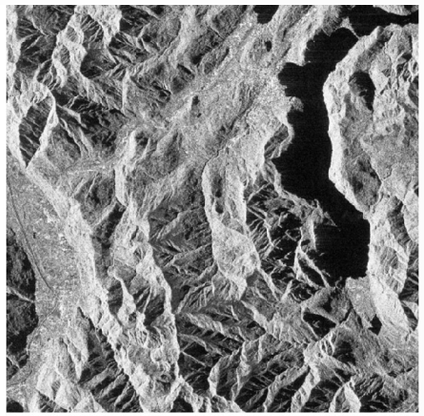
Communication Modulation



Component Gain / Loss

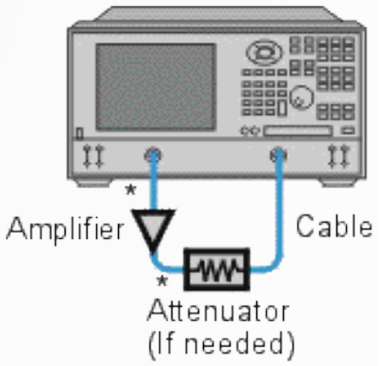


Synthetic Aperture Radar Image

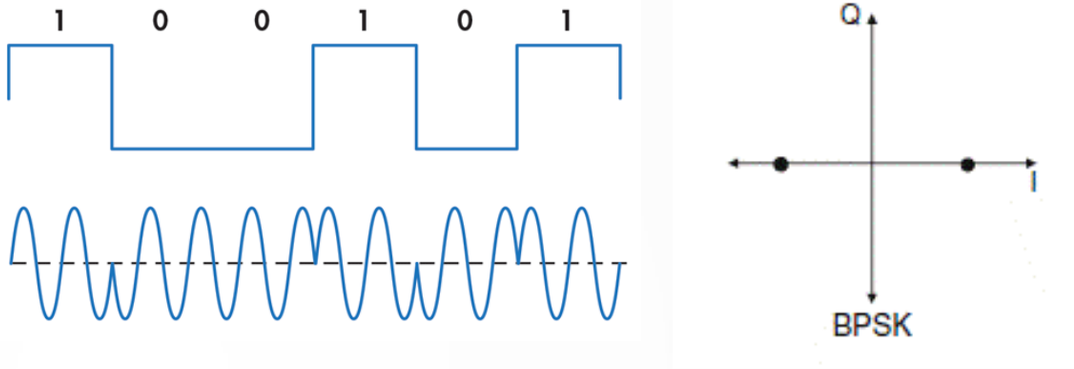


1. Phase

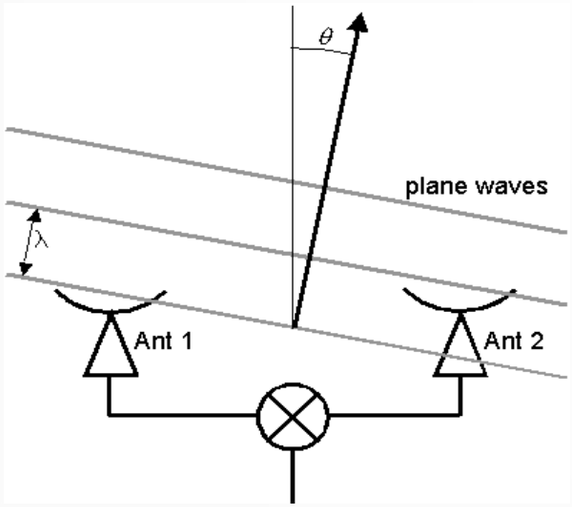
Length / Distance Measurement



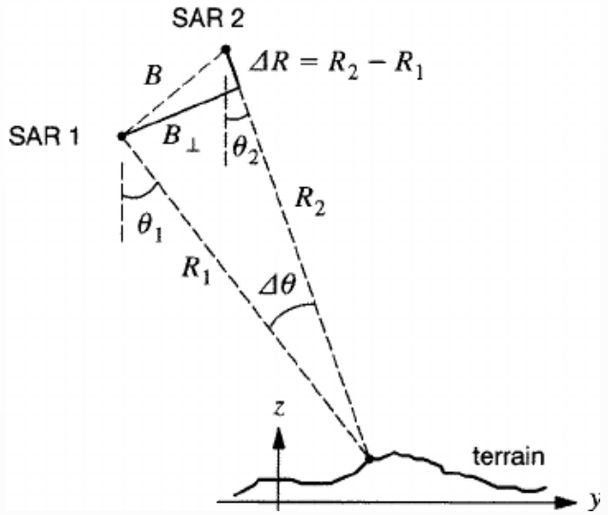
Data Communication Modulation



Radio Interferometer



Synthetic Aperture Radar Interferometer



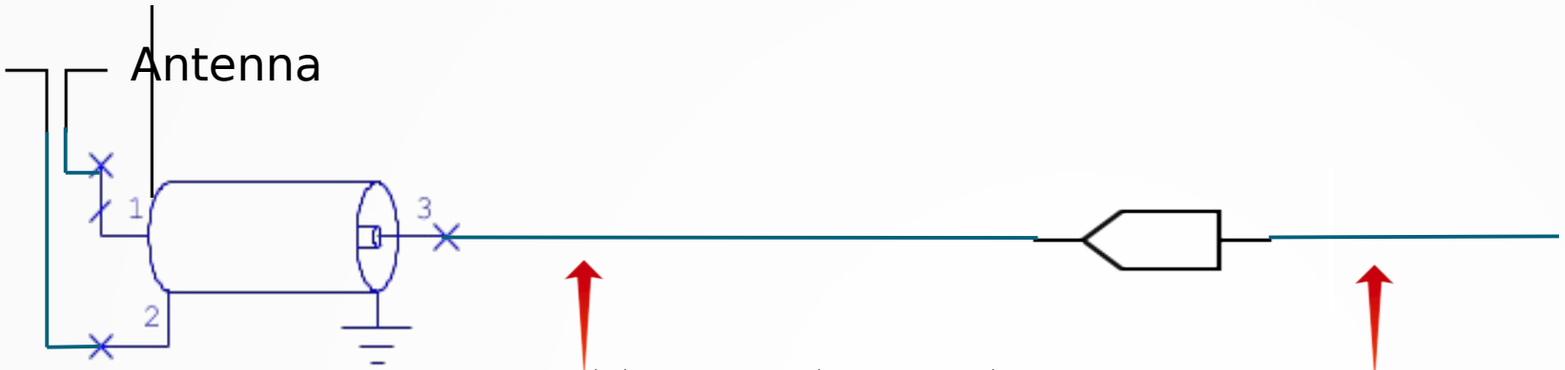
2. Euler’s Formula

$$j = \sqrt{-1}$$

$$A \cdot e^{j \cdot \phi} = A \cdot [\cos(\phi) + j \cdot \sin(\phi)]$$

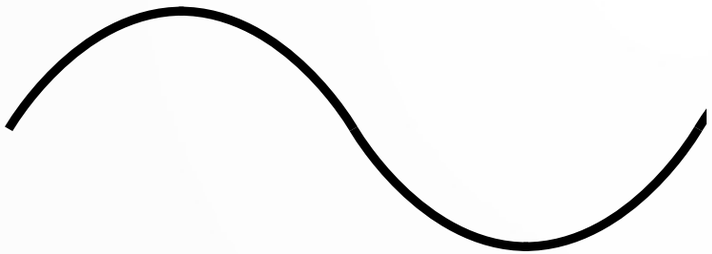
- Year 1748 - Mathematician Leonhard Euler published that periodic functions such as cosine and sine can be represented as complex exponentials of instantaneous phase (phasors)
- This model simplifies math related to amplitude and phase

3. Math Model of Signal



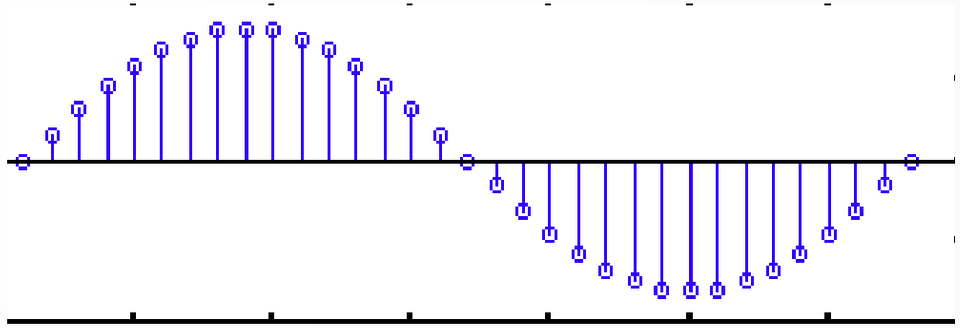
$V(t) = A \cdot \cos(2\pi ft + \phi)$

t in continuous time



$S[n] = A \cdot \cos(2\pi nT_s + \phi)$

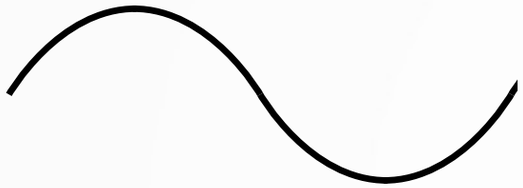
n is integer index for discrete time



At any location, we can measure real-valued voltage and record with ADC. Recorded data includes only instantaneous amplitude. How do we find the phase information?

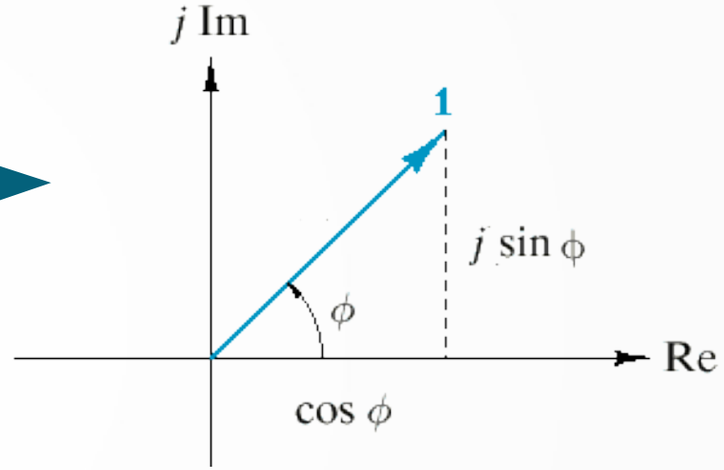
3. Math Model of Signal

Real-value cosine



Receiver

Euler's model of complex phasor

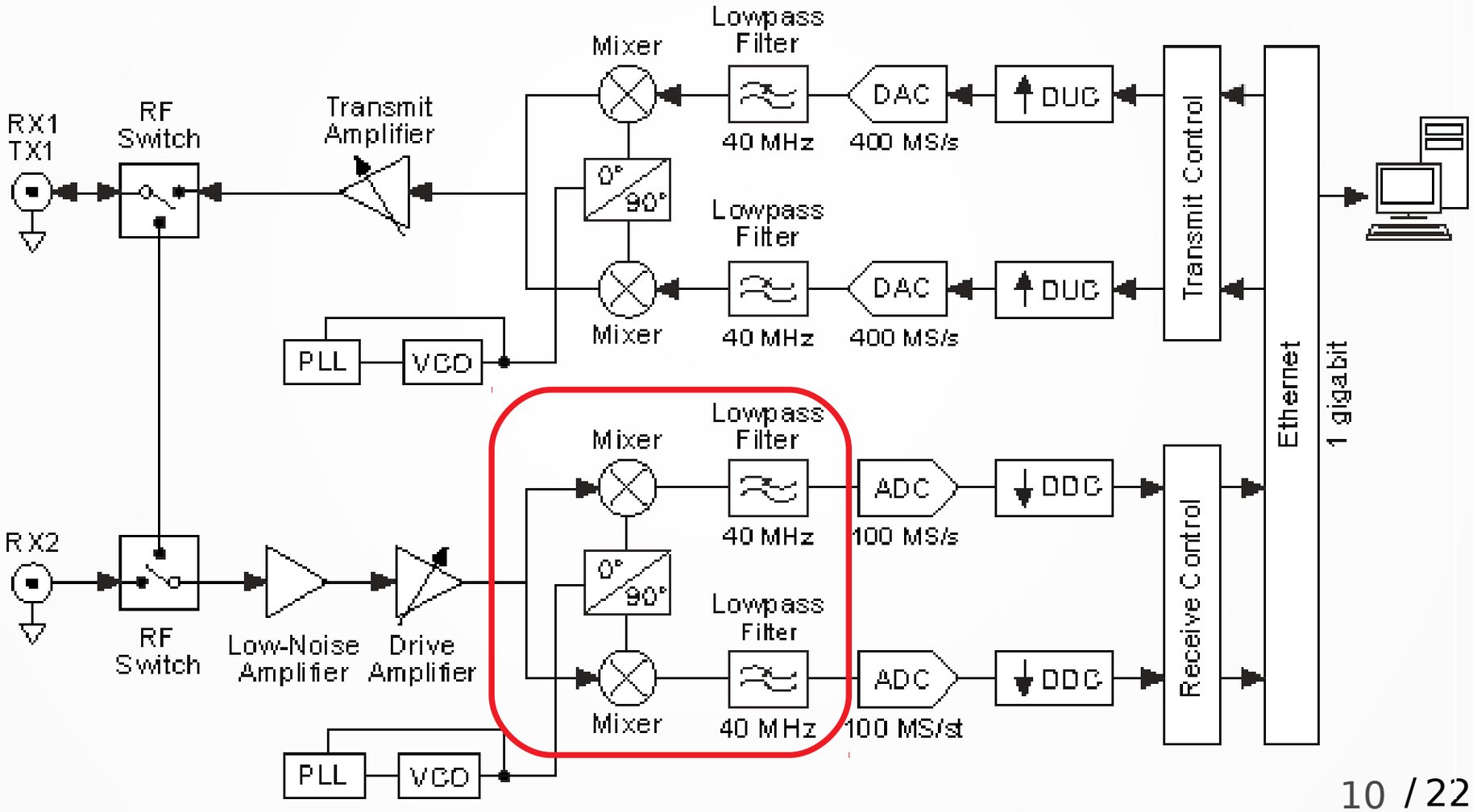


$$V(t) = A \cdot \cos(2\pi ft + \phi) \longrightarrow A \cdot e^{j \cdot \phi} = A \cdot [\cos(\phi) + j \cdot \sin(\phi)]$$

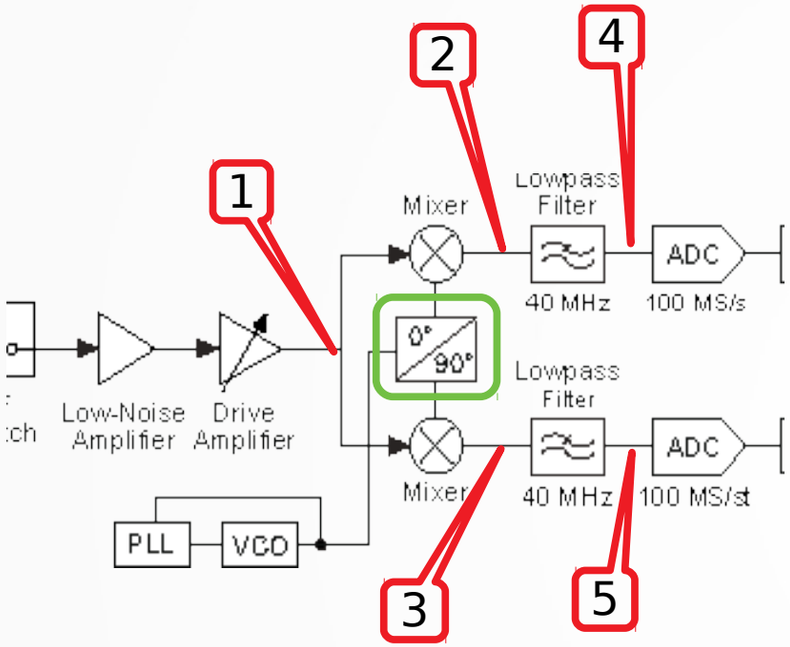
How can we translate real-value voltage signal model (easy to understand in the physical world) to quadrature I/Q signal model (easy to process using DSP software)?

4. Implementation of Model in Hardware

- I/Q Downconverter Receiver (example Ettus USRP N210+WBX)



4. Implementation of Model in Hardware



$$V_1(t) = A \cdot \cos(\phi(t)) = A \cdot \cos(2\pi f_{RF} t + \phi_0)$$

Use trig identity – product to sum

$$V_2(t) = A \cdot \cos(2\pi f_{RF} t + \phi_0) \cdot \cos(2\pi f_{LO} t + \phi_{LO}) \ldots$$
$$A \cdot \cos[2\pi (f_{RF} + f_{LO}) t + \phi] + A \cdot \cos[2\pi (f_{RF} - f_{LO}) t + \phi]$$

Lowpass filter removes

$$V_3(t) = A \cdot \cos(2\pi f_{RF} t + \phi_0) \cdot \cos(2\pi f_{LO} t + \phi_{LO} - \frac{\pi}{2}) \ldots$$

90 degree phase delay

$$A \cdot \cos(2\pi f_{RF} t + \phi) \cdot \sin(2\pi f_{LO} t) \ldots$$
$$A \cdot \cos[2\pi (f_{RF} + f_{LO}) t + \phi] + A \cdot \sin[2\pi (f_{RF} - f_{LO}) t + \phi]$$

Lowpass filter removes

$$f_{IF} = f_{RF} - f_{LO}$$

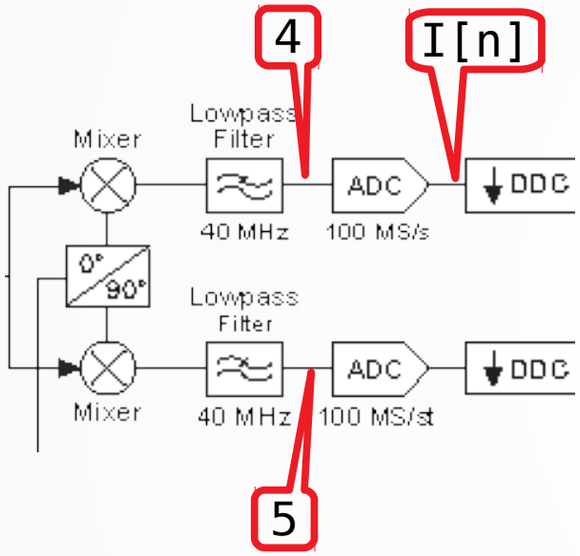
$$V_4(t) = A \cdot \cos[2\pi (f_{RF} - f_{LO}) t + \phi]$$
$$V_5(t) = A \cdot \sin[2\pi (f_{RF} - f_{LO}) t + \phi]$$

$$\Rightarrow$$

$$V_4(t) = A \cdot \cos[2\pi f_{IF} t + \phi]$$
$$V_5(t) = A \cdot \sin[2\pi f_{IF} t + \phi]$$

5. Analog to Digital Conversion (ADC)

- 2 ADCs with same sample clock and same length of cable / PCB trace



Example 8-bit = 1 byte per sample

Channel I “in phase”

byte 0	byte 1	byte 2	...
I[0]	I[1]	I[2]	...

byte 0	byte 1	byte 2	...
Q[0]	Q[1]	Q[2]	...

Channel Q “quadrature phase”
(delay by $\pi/2$ radians = λ / 4)

$$I[n] = A \cdot \cos(2\pi f_{IF} n T_s + \phi)$$

$$Q[n] = A \cdot \sin(2\pi f_{IF} n T_s + \phi)$$

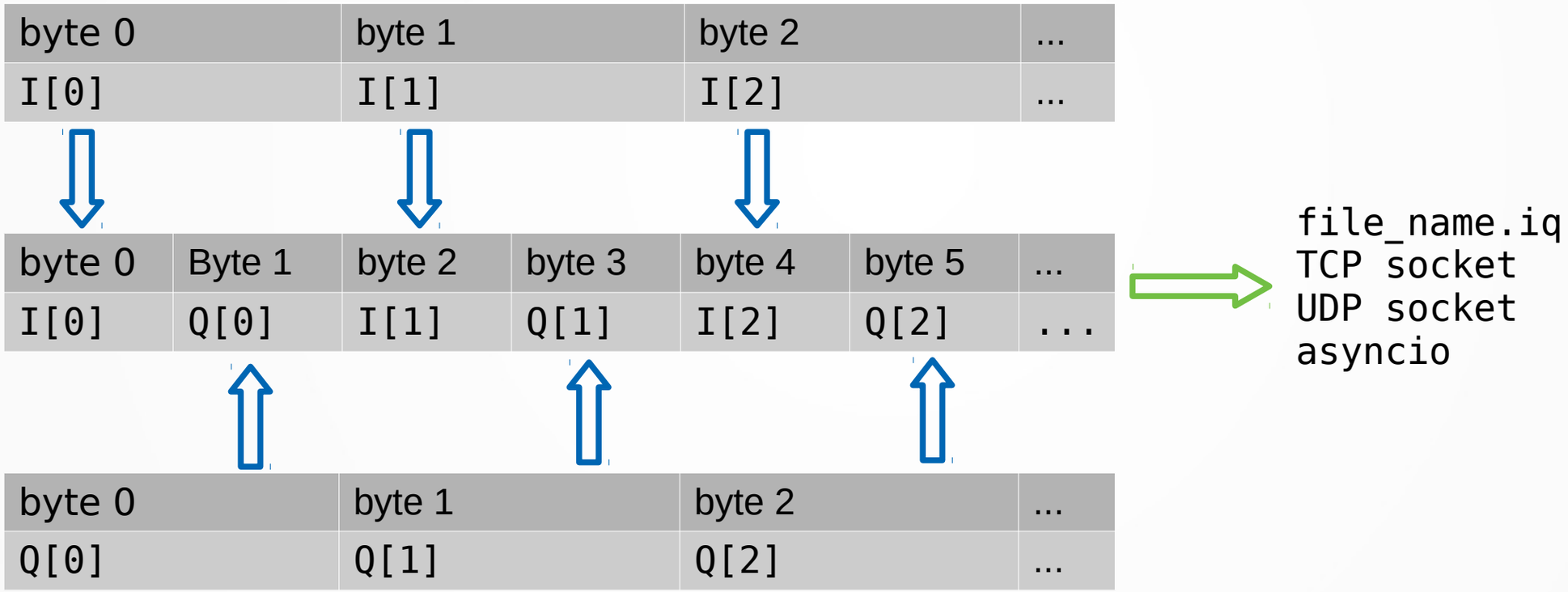
How to combine … ?

$$S_{IQ}[n] = I[n] + j \cdot Q[n] = A \cdot e^{j(2\pi f_{IF} n T_s + \phi)}$$

NOTE: ‘I’ and ‘Q’ in this context are for 1 polarization. Not related to Stokes parameters {I, Q, U, V}

5. Analog to Digital Conversion (ADC)

- Interleave the I and Q data buffers into 1 on synchronous system before sending out to file or asynchronous system (non-real-time Ethernet, etc.)



6. Convenience in Software

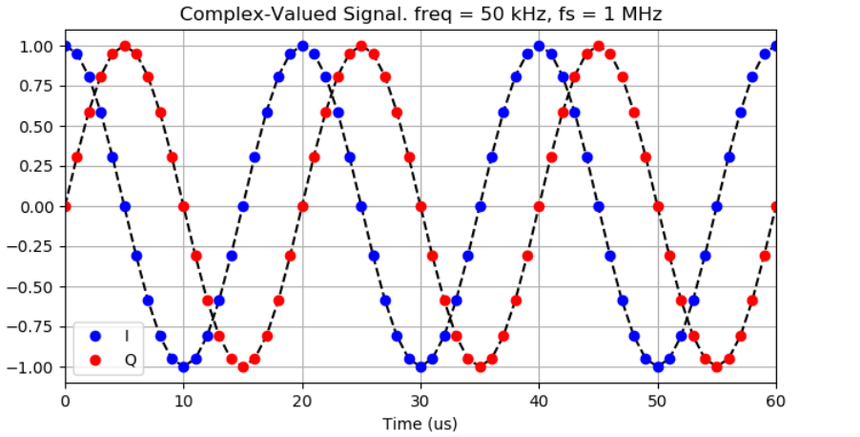
- Programming languages include support for complex/imaginary numbers and exponential function.
- Many libraries of software are available to process data in this format

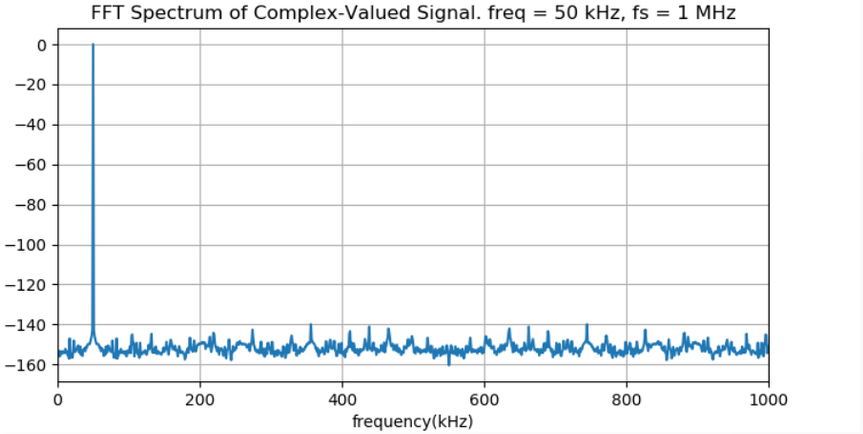
Language	$symbol\ \sqrt{-1}$	$function\ e^{x}$
Python	`1j`	`numpy.exp(x)`
Matlab	`J, j, I, i`	`exp(x)`
C	`<complex.h> I`	`<complex.h> cexp(x)`
Octave	`J, j, I, i`	`exp(x)`

6. Convenience in Software

```
import numpy as np
fp = open('demo.iq','rb')
fp.seek(0,0)
dt = np.dtype([('i',np.float32),('q',np.float32)]
data = np.fromfile(fp,dtype=dt,count=nsamples)
sigt = data['i']+1j*data['q']
```

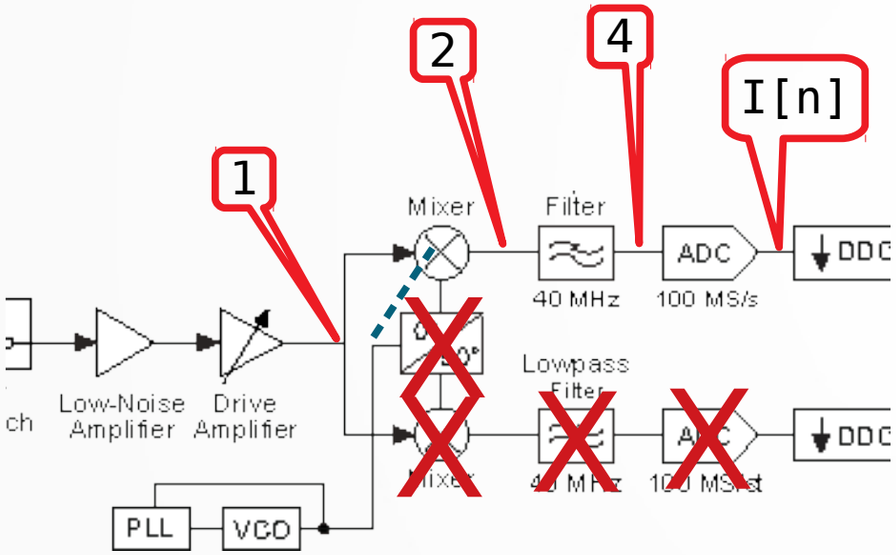
$$S_{IQ}[n] = S_I[n] + j \cdot S_Q[n] = e^{(j 2\pi f_{IF} n T_s + \phi)}$$





7. Implementation in Software

- Single-phase downconverter (not I/Q)
- Same math from page 10-11 but remove hardware and data of Q channel



Channel I "in phase"

byte 0	byte 1	byte 2	byte n
I[0]	I[1]	I[2]	I[n]

$$V_4(t) = A \cdot \cos[2\pi f_{IF} t + \phi]$$

$$I[n] = A \cdot \cos(2\pi f_{IF} n T_s + \phi)$$

How to create the 'Q' component ... ?

$$S_{IQ}[n] = S_I[n] + j \cdot S_Q[n] = A \cdot e^{j(2\pi f_{IF} n T_s + \phi)}$$

7. Implementation in Software

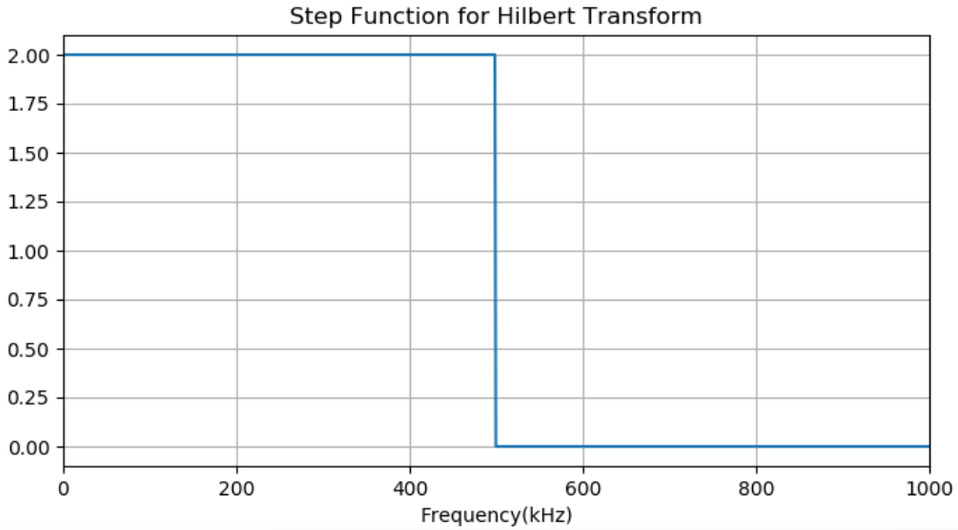
- Create the complex-valued analytic signal by **Hilbert transform** of the real-valued signal

$$S_{IQ}[n] = HILBERT(S_I[n]) = IFFT(FFT(S_I[n]) \cdot 2 \cdot UNITSTEP[n])$$

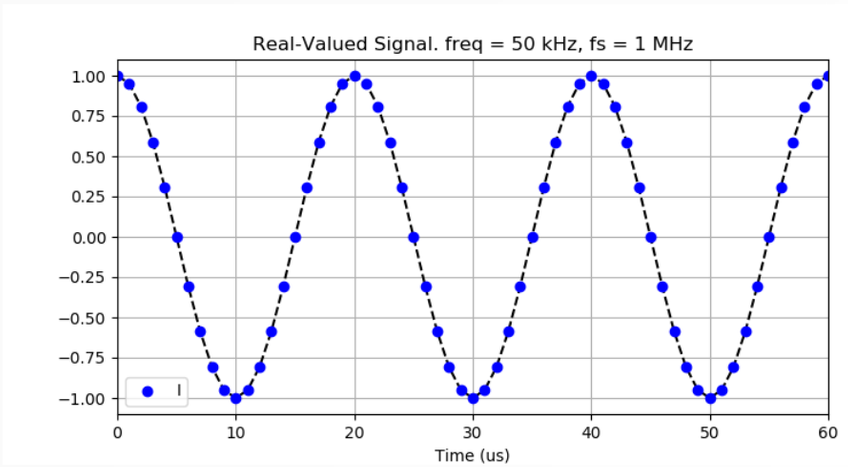
- Function is available in Python : scipy.signal.hilbert()

```
sig_r = np.cos(2*np.pi*freq*tt+phi0)

sig_h = scipy.signal.hilbert(sig_r)
```

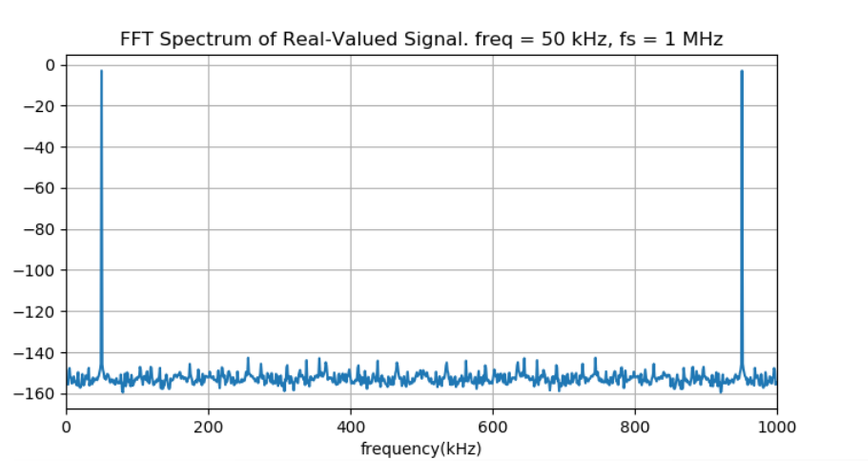


7. Implementation in Software

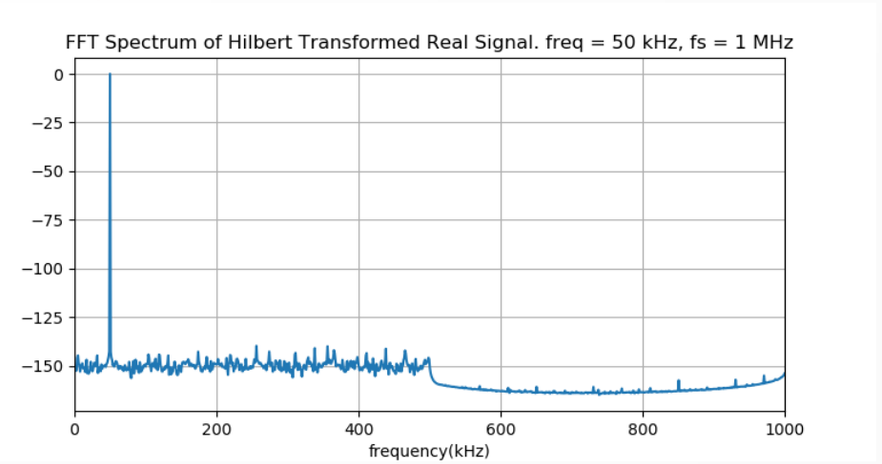


FFT

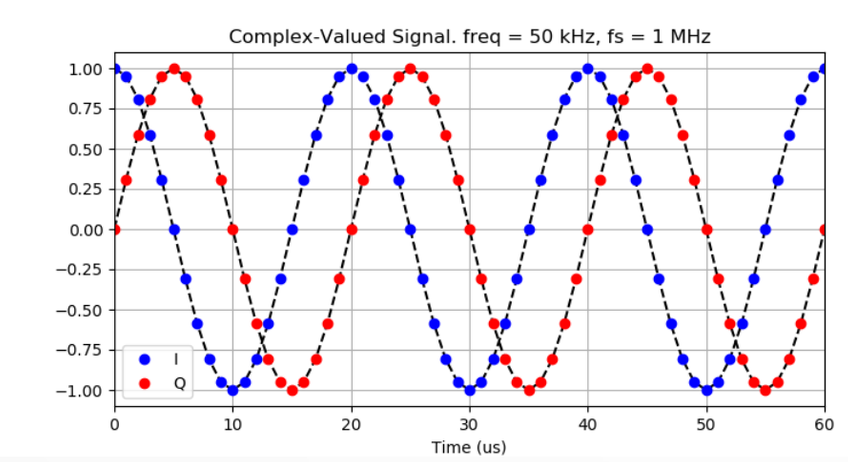
(data bytes x 2)



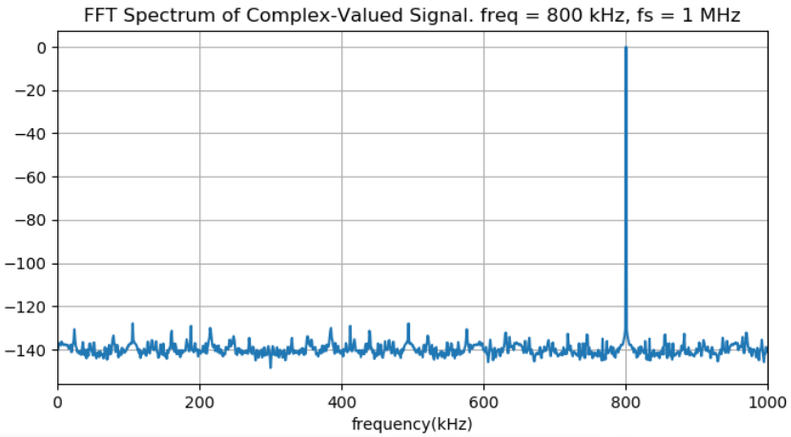
Multiply step function (delete 2nd Nyquist zone)



IFFT



8. Compare: I/Q vs Real-Value Sampling

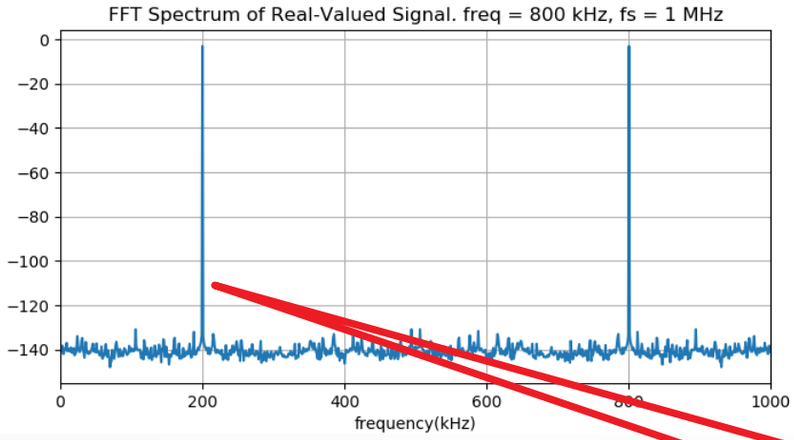


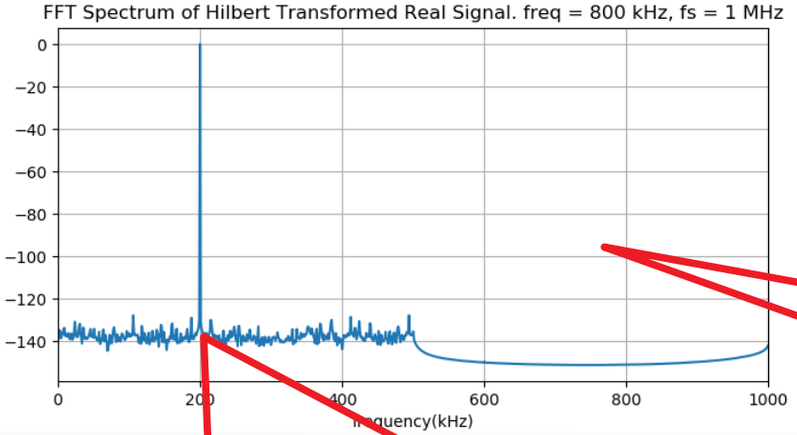
Signal appears at 800 kHz. Good!

Example for I/Q Sampling:

- Sample rate f_s = 1 MHz
- RF signal frequency f_RF = 800 kHz
- Both I and Q channels of receiver are sampled.
- I and Q data arrays are combined in software to be complex samples

8. Compare: I/Q vs Real-Value Sampling





Example for Real-Value Sampling:

- Sample rate f_s = 1 MHz
- RF signal frequency f_RF = 800 kHz
- Only “I” channel is sampled
- “I” data array is converted to complex in software with Hilbert transform

Complex FFT creates a 2nd signal at 200 Khz

Hilbert transform removes the signal in 2nd Nyquist zone

Result signal is aliased at (f_s - f_RF) = 200 kHz. NOT GOOD!

8. Compare: I/Q vs Real-Value Sampling

Conclusions for fixed sample rate:

- I/Q complex sampler can record RF signal not more than 1 x sample rate
- Real-value sampler can record RF signal not more than 0.5 x sample rate

Conclusions for fixed frequency and bandwidth

- I/Q complex sampler must use sample rate >= 1 x RF bandwidth – but on 2 channels at the same time.
- Real-value sampler must use sample rate >= 2 x RF bandwidth – only on 1 channel.
- Final data rate is the same for both methods. But the work load is distributed in different ways.

8. Compare: I/Q vs Real-Value Sampling

	I/Q Conversion in software (real-value samples)	I/Q Conversion in hardware (complex-value samples)
PCB Component Quantity	1x	2x
PCB Design Difficulty (balance I and Q signal path)	easy	difficult
Sample Rate (for same bandwidth of signal)	2x	1x
Operational Bandwidth (for same sample rate)	1x	2x
Compute Complexity	more	less
Electric Power Consumption	more?	less?
Data rate and storage size after some processing	same	same

If the required bandwidth for the system is small (low data rate, easy for ADC, CPU / GPU), we can implement a single-phase, real-value sampling receiver. Reduce design work in RX hardware. Reduce load on software and computer.

If the required bandwidth is big (high data rate, limit of CPU / GPU), we can implement the I/Q receiver design. Increase design work in RX hardware. Reduce load on software and computer.