Intro to DSP: Sampling

with GNU Radio

Jeff Long
Analog Signals
- Radio frequency (over the air)
- Baseband (input to ADC)
  - Continuous
  - Real

Digital Signals
- Represent analog
- Discrete
- Evenly spaced in time
- Travel in groups (DMA, buffers)

Front End
- SDR Hardware
- Reconfigurable Logic

Typical SDR System (RX)
- May include multiple Phases (I/Q or Quadrature)
- Antennas (MIMO)

Front End
- SDR Hardware
- Reconfigurable Logic

Bus USB2 USB3 GBE PCI

Digital
- Data
- Control

(no two are alike)
Start with something simple ...
Linear motion
Constant speed
Known start time
→ sawtooth wave
→ periodic signal
→ clock sync
Continuous Analog Signal

Discreet Values

Measure (1, 0, -1, 0, 1)

\[ T_{\text{signal}} = \frac{1}{F_{\text{signal}}} \]

Constant Rate Clock \( (F_{\text{sample}}) \)
Nyquist Rate

\[ F_{\text{sample}} > 2 F_{\text{signal}} \]

Sampling Ratio = \( \frac{F_{\text{sample}}}{F_{\text{signal}}^{\text{MAX}}} = \frac{T_{\text{signal}}^{\text{MIN}}}{T_{\text{sample}}} \)
Time (Phase) shift cannot represent phase

Time is information!
Can we measure 2 signals with one value? Take advantage of phase ... different start times.
I Signal

Q Quadrature

F_{signal} > 4 F_{sample}

\[ S = I + jQ \]

'j' means 90° out of phase
RF communication is based on sine waves
Sine and Cosine are the same function, shifted by 90° ($\pi/2$)

Addition?

Same function, shifted!

$$\text{sig1} = \cos(x)$$

$$\text{sig2} = \sin(x)$$

$$\left(\text{sig1} + \text{sig2}\right) \cdot \frac{1}{\sqrt{2}}$$

Why?

Same frequency
Different phases
Adding Sines in the time domain is the same as adding impulses in the frequency domain.

Display using GNU Radio frequency sink.

A-major Chord

440 Hz
554 Hz
659 Hz
880 Hz

$F_{\text{sample}} = 4000 \text{ S/s}$
Sample rate = 4000 S/s can represent signal freq < 2000 Hz

- **F_sample** = 4000 S/s

- Frequencies: 440 Hz, 554 Hz, 659 Hz, 1000 Hz

**Hurts the ears** frequencies are indicated above.
Sample rate = 4000 S/s can represent signal freq < 2000 Hz

3120 Hz tone aliases down to 880 Hz from Second Nyquist Zone
4000-3120 = 880
Sine Wave
Swept 0-500 kHz
1MS/s

Keep 1 in 2
(Decimation)
500kS/s
Both signal AND noise will be aliased ... another reason to filter.
Downsampling = Filtering + Decimation
Upsampling = Interpolation + Filtering
Clocking Complications

GNU Radio does not have its own clock (* exception)

Device sources and sinks have clocks

Usually different clocks ... creating clock domains
Frequency tolerances
Not synced (out of phase)

Must pick a clock
Usually only one good choice
Audio drivers will often resample (ALSA plughw:1,0)
Clocking Domains

BAD (with ALSA hw:)
Processor and Audio both working off their own clocks
Over/underruns, stutter

GOOD (with ALSA plughw:)
Audio driver does software resampling to match clocks
Clocking Complications

BAD
No good way to sync clocks
Will see either

OOOOOOOOOOO – RTL buffer overrun
UUUUUUUUUUU – USRP buffer underrun
Clocking Domains

**RTL-SDR Source**
- Sample Rate (sps): 48k
- Ch0: Frequency (Hz): 100M
- Ch0: Freq. Corr. (ppm): 0
- Ch0: DC Offset Mode: Off
- Ch0: IQ Balance Mode: Off
- Ch0: Gain Mode: Manual
- Ch0: RF Gain (dB): 10
- Ch0: IF Gain (dB): 20
- Ch0: BB Gain (dB): 20

**File Sink**
- File: Unbuffered: Off
- Append file: Overwrite

**Signal Source**
- Sample Rate: 48k
- Waveform: Cosine
- Frequency: 1k
- Amplitude: 1
- Offset: 0

**Audio Sink**
- Sample Rate: 48k

**BOTH GOOD**

Only one clock domain

Rest of flowgraph will maintain pace
Buffers
Handling one sample at a time is slow.

Many blocks require multiple samples:
- Filter history
- FFT and other block operations
- SIMD optimization

Add buffering, operate on batches of samples:
- Improves performance
- Increases latency
- Hides rate mismatches for a short time
Buffers hold the output of each block.

GR passes “items” between blocks, the most common being samples.

Multiple blocks can read from the same buffer.
The Vertical
Clipping

Voltage limits
Analog circuitry
ADC settings

Representation limits
Maximum value
Minimum resolution
Headroom

Quantization Noise

Signal Level Tradeoff
Lower resolution or lower signal  
= More quantization noise

Can trade off oversampling (averaging) vs resolution  
1 bit of resolution ~ 2x oversampling  
(*) depends on noise characteristics
Sample Types

Common GR Types

Float32 (float): 23 bits (Complex is 2 floats)
Integer8 (byte): 8 bits
Integer16 (short): 16 bits
Integer32 (int): 32 bits

One bit for +/-
Sample Type Conversion

Float/Complex
Can scale to any useful range
Often scaled to +/- 1.0

Signed Byte/Short/Int
Have fixed range
+/- 127 / 32K / 2G

**Short → Float**
Sink expects -1.0 ~ 1.0
Signal will be clipped
Scale 1.0 / 32767 or lower

**Float → Short**
Source produces -1.0 ~ 1.0
File will contain values near zero
Scale 32767 or lower
Sampling Window (Integration Time)

Due to capacitance in ADC
Acts as a low-pass filter
What is the “right” rate?

- **Real**: $> 2 \times \text{max signal frequency}$
- **Quadrature**: $> 4 \times \text{max signal frequency}$
- Account for filter rolloff
- Oversample to reduce noise
- Oversample to relax filter quality requirements
- If aliasing is desired, undersampling is OK
- **Hardware**
  - May not support arbitrary rates
  - May have “better quality” rates
  - Performance limits (sampler, bus, computer)
- **Prescribed rates** (LTE 10 MHz channel → 15.36 MS/s)
Isolate a signal from noise and other signals
Perform anti-aliasing or image-rejection (rate change)
GR filters can interpolate or decimate
FAQ: Why do I get 'OOOOOOOOO' with a simple flowgraph?
Efficiency in filtering
Use the cheapest filter that does the job right
Downsample as early as possible, decimate late
Use multiple stages for large rate changes

Some blocks contain filters, perform rate changes

1MS/s, 100 Hz transition → 24091 taps → ~24 GFLOPS (using FIR)