GNU Radio in the Undergraduate Communications Curriculum

Peter Mathys
Department of ECEE
University of Colorado, Boulder
The SDR evolution occurred roughly over the last decade. Wide variety of hardware available from DVB-T tuners (Realtek RTL2832U, $10) to the HackRF One (Great Scott Gadgets, $299) and the USRP E310 (Ettus Research, $2700). Typical RF frequency range is from a few 10 MHz to single digit GHz. Some software choices: GNU Radio, Matlab/Simulink.
Software Defined Radio (SDR)

- Have SDRs changed the fundamental principles of communication theory (Shannon, 1948, “A Mathematical Theory of Communication”)?
- No! But it has completely changed the ways we go about experimenting with, implementing of, visualizing of, and understanding of new and more precise ways to create and receive sophisticated information carrying waveforms.
GNU Radio in the Classroom

- Expand GNU Radio user base in size and diversity.
- GNU Radio is affordable.
- GNU Radio is sophisticated.
- GNU Radio is open source and expandable.
- GNU Radio is a great tool to visualize the effects of signal processing blocks in real-time and ask what-if questions.
Example: Full/Half Wave Rectifier

Signal Source
Sample Rate: 64k
Waveform: Cosine
Frequency: 1k
Amplitude: 1
Offset: 0

Rectifier ff
Full Wave: True

Throttle
Sample Rate: 64k

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 64k
Update Rate: 10

QT GUI Chooser
ID: Full_Wave
Num Options: 2
Default Value: True
Option 0: False
Option 1: True
Where is the Rectifier ff block in GNU Radio?
Rectifier ff is an OOT Module

- Out-of-tree modules are used to implement your own functions alongside the main GNU Radio code.
- gr_modtool is used to set up the framework.
- The main components of a typical OOT module are:
  - The Python (or C++) QA test file: qa_rectifier_ff.py.
  - The public header file: rectifier_ff.h
  - The implementation header file: rectifier_ff_impl.h
  - The implementation source file: rectifier_ff_impl.cc
  - The xml block definition for GRC: test01_rectifier_ff.xml
- There is lots more behind the scenes!
int rectifier_ff_impl::work(int noutput_items,
                        gr_vector_const_void_star &input_items,
                        gr_vector_void_star &output_items)
{
    const float *in = (const float *) input_items[0];
    float *out = (float *) output_items[0];

    // Do <+signal processing+>
    for (int i = 0; i < noutput_items; i++)
        if (in[i] >= 0.0)
            out[i] = in[i];
        else if (d_full_wave)
            out[i] = -in[i];
        else
            out[i] = 0.0;

    // Tell runtime system how many output items we produced.
    return noutput_items;
}
```python
def test_002_rectifier_ff_half(self):  # half wave rectifier test
    src_data = (0.0, 0.3827, 0.7071, 0.9239, 1.0, 0.9239, 0.7071, 0.3827,
                 0.0, -0.3827, -0.7071, -0.9239, 1.0, -0.9239, -0.7071, -0.3827,
                 0.0, 0.3827, 0.7071, 0.9239, 1.0, 0.9239, 0.7071, 0.3827,
                 0.0, -0.3827, -0.7071, -0.9239, 1.0, -0.9239, -0.7071, -0.3827)
    exp_data = (0.0, 0.3827, 0.7071, 0.9239, 1.0, 0.9239, 0.7071, 0.3827,
                 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                 0.0, 0.3827, 0.7071, 0.9239, 1.0, 0.9239, 0.7071, 0.3827,
                 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
    src = blocks.vector_source_f(src_data)
    op = test01.rectifier_ff(False)
    dst = blocks.vector_sink_f()
    self.tb.connect(src, op, dst)
    self.tb.run()
    result_data = dst.data()
    self.assertFloatTuplesAlmostEqual(exp_data, result_data, 5)
```
Why Write Your Own OOT Modules?

- GNU Radio is work in progress. Some functions may just simply not yet be available. Or you may have come up with a great new idea that you would like to test. Or you have some specific needs, e.g., to educate undergraduate students.

- It's a great (but not painless) way to learn about GNU Radio.

- Where can I learn how to do it? That's a bit of a sticky point. There are some tutorials at [https://gnuradio.org](https://gnuradio.org) and there is the discuss-gnuradio mailing list, but expect to do a lot of trial and error.
The Undergraduate Curriculum in Communications at CU Boulder

- Communications track is two semester sequence for seniors:
  - ECEN 4242, Communication Theory, taught in fall
  - ECEN 4652, Communications Laboratory, taught in spring
- Prerequisites for ECEN 4242 are Linear Systems and Probability theory.
- Prerequisite for ECEN 4652 is ECEN 4242.
ECEN 4242 Communication Theory

1. Introduction
2. Linear Systems (review)
3. Amplitude Modulation
4. Angle Modulation
5. Probability Theory (review), Random Processes (introduction)
6. Noise in CM Modulation Systems
7. Transition from Analog to Digital (PAM, TDM, PPM, PCM)
8. Digital Baseband Communications (noise, ISI, matched filter, probability of error)
9. Digital Bandpass Communications (PSK, FSK, signal constellations)
ECEN 4242 and GNU Radio

- The main goal of GNU Radio is to control SDRs, not to teach linear systems or to implement baseband communications.
- But to learn communication theory one needs to start with small building blocks and piece them together into larger systems while retaining the ability to probe the output of each of the building blocks.
Building Block Examples
Amplitude Modulation
Amplitude Modulation

Options
ID: am_xmtr_test_001
Title: AM Transmitter Test 001
Author: Peter Mathys
Description: AM-DSPIrectifier
Generate Options: QT GUI

Variable
ID: samp_rate2
Value: 8k

Variable
ID: samp_rate
Value: 776k

Variable
ID: upsamp
Value: 97

QT GUI Range
ID: Ac
Default Value: 2
Start: 0
Stop: 10
Step: 100m

QT GUI Range
ID: fc
Default Value: 100k
Start: 0
Stop: 388k
Step: 1k

QT GUI Range
ID: fm
Default Value: 1.5k
Start: 0
Stop: 5k
Step: 100

QT GUI Chooser
ID: Full_Wave
Num Options: 2
Option 0: False
Option 1: True

Signal Source
Sample Rate: 776k
Waveform: Cosine
Frequency: 100k
Amplitude: 2
Offset: 0

Add

Throttle
Sample Rate: 776k

Rectifier ff
Full Wave: False

Signal Source
Sample Rate: 8k
Waveform: Cosine
Frequency: 1.5k
Amplitude: 1
Offset: 0

Interpolating FIR Filter
Interpolation: 97
Taps: firdes.low_pass_2(up...)

Interpolating FIR Filter

QT GUI Sink
FFT Size: 8.192k
Center Frequency (Hz): 0
Bandwidth (Hz): 776k
Update Rate: 10
Frequency Modulation
Frequency Modulation

Options
ID: armstrong_fm_test_001
Title: Armstrong FM Test 001
Author: Peter Mathys
Generate Options: QT GUI

QT GUI Range
ID: delta_f
Default Value: 100
Start: 0
Stop: 500
Step: 5

Multiply Const Constant: 628.319

Interpolating FIR Filter
Interpolation: 50
Taps: firdes.low_pass_2(up

DC Blocker
Length: 32
Long Form: False

Variable
ID: samp_rate
Value: 1.6M

Import
Import: math

Variable
ID: samp_rate2
Value: 32k

Variable
ID: upsamp
Value: 50

Signal Source
Sample Rate: 1.6M
Waveform: Sine
Frequency: 100k
Amplitude: 1
Offset: 0

Multiply
ID: fc1
Value: 100k

Low Pass Filter
Interpolation: 1
Gain: 2
Sample Rate: 1.6M
Cutoff Freq: 200k
Transition Width: 10k
Window: Hamming
Beta: 6.76

Power n ff
Exponent n: 5

Subtract

Power n ff
Exponent n: 6

QT GUI Sink
FFT Size: 8.192k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.6M
Update Rate: 10

Multiply

Band Pass Filter
Interpolation: 1
Gain: 16
Sample Rate: 1.6M
Low Cutoff Freq: 400k
High Cutoff Freq: 600k
Transition Width: 5k
Window: Hamming
Beta: 6.76

Cumsum ff

Multiply Const Constant: 31.25u

Throttle
Sample Rate: 32k

Band Pass Filter
Interpolation: 1
Gain: 32
Sample Rate: 1.6M
Low Cutoff Freq: 500k
High Cutoff Freq: 700k
Transition Width: 5k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 1.6M
Waveform: Cosine
Frequency: 500k
Amplitude: 1
Offset: 0
Find beta using Bessel Functions
ECEN 4652 Communications Lab

- Lab 1: CT and DT Signals, ASCII Code, Parallel to Serial to Parallel Conversion, Simple Rectangular PAM
- Lab 2: Fourier Transform Approximation by DFT/FFT, More General PAM (Rectangular, Triangular, Sinc Pulses)
- Lab 3: Sampling Theorem, Nyquist's Criterion, Eye Diagrams, ISI, Partial Response Signaling
- Lab 4: Random Processes, Power Spectral Density, Noise, Symbol Timing Information
- Lab 5: PAM Receiver with Matched Filter, SNR, Probability of Symbol Error
ECEN 4652 Communications Lab

- Lab 6: Introduction to Software Defined Radio and GNU Radio
- Lab 7: Amplitude Modulation with Suppressed Carrier, Coherent Reception
- Lab 8: Amplitude Modulation with Transmitted Carrier, Noncoherent Receivers, FDM
- Lab 9: M-ary Amplitude and Frequency Shift Keying, Signal Space
- Lab 10: Real Bandpass and Complex Lowpass Signals, QAM, General Bandpass Filters
- Lab 11: Phase and Hybrid Amplitude/Phase Shift Keying, Carrier Synchronization
- URL: http://ecee.colorado.edu/~mathys/ecen4652
A Missing Piece in GNU Radio

- Serial bit input to ASCII text conversion with:
  - LSB or MSB first selection,
  - Selectable threshold for 0/1 decision,
  - $0 \leftrightarrow 1$, $1 \leftrightarrow 0$ inversion,
  - Selectable bit shift within bitstream
  - Character substitution for non-printable ASCII
  - Switchable between Hex and ASCII display
Our print_char_b OOT Block

Print Char b
ASCII Mode: Yes
Non-Printable Substitution: Yes
Filename: /dev/pts/23
Line Length: 64
Substitution Character: 46

Properties: Print Char b
ID: test01_print_char_b_0
ASCII Mode: Yes
Non-Printable Substitution: Yes
Filename: /dev/pts/23
Line Length: 64
Substitution Character: ord('.')
Visualizing a Serial Data Stream

Options
ID: text_serial_f_001
Title: Text Serial f 001
Author: Peter Mathys
Generate Options: QT GUI

QT GUI Range
ID: bit_shift
Default Value: 0
Start: 0
Stop: 7
Step: 1

QT GUI Range
ID: threshold
Default Value: 0
Start: -2
Stop: 2
Step: 100m

QT GUI Chooser
ID: Invert
Num Options: 2
Default Value: 0
Option 0: False
Label 0: False
Option 1: True
Label 1: True

QT GUI Chooser
ID: Isb_first
Num Options: 2
Default Value: 1
Option 0: False
Label 0: False
Option 1: True
Label 1: True

QT GUI Chooser
ID: ascii
Num Options: 2
Default Value: 1
Option 0: False
Label 0: False
Option 1: True
Label 1: True

Variable
ID: samp_rate
Value: 32k

File Source
File: text_serial_f.txt
Repeat: Yes

Throttle
Sample Rate: 32k

Serial to Parallel fb
Bit Shift: 0
0/1 Threshold: 0
Invert Bits: False
LSB First: True

Print Char b
ASCII Mode: True
Non-Printable Substitution: Yes
Filename: /dev/pts/23
Line Length: 64
Substitution Character: 46

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 32k
Update Rate: 10
Visualizing a Serial Data Stream
Wrong Bit Shift for ASCII
Rectangular PAM without MF
Rectangular PAM after MF
Flying by the Seat of our Pants
GNU Radio Installation

• **1'st attempt:**
  • Install VirtualBox on either PC or Mac
  • Install Ubuntu 14.04 in VirtualBox
  • Install GNU Radio in Ubuntu using script at www.sbrac.org/files/build-gnuradio

• Problems:
  • GNU Radio in VirtualBox is too slow
  • Too much troubleshooting needed for individual students and their computers
GNU Radio Installation

- 2\textsuperscript{nd} attempt (after seg fault in my own installation due to software conflicts):
  - Put Ubuntu and GNU Radio on a USB flashdrive for both experimentation and for handing out to students
  - Use the older MBR method (and not UEFI) to boot from the flash drive
  - Use fast (USB 3) 32 GB flashdrive
  - Do not format and use the full 32 GB if you want to make image copies of the flashdrive (some 32 GB drives are smaller than others!)
New to GNU Radio Development?

- The (likely) unknowns:
  - Unix/Linux
  - Python
  - C/C++
  - Object-Oriented Programming
  - SWIG
  - Cmake, Make
  - Cheetah and Mako

- Boost
  - Git, GitHub
  - Sphinx, Doxygen
  - XML
  - FFTW
  - Volk
  - (Multirate) DSP
  - Communication Theory
  - Communication Standards
Learning How to Write OOT Modules

- Start from the square_ff tutorial (gr::sync_block)
- Look at code for add_const_vxx block to learn about float and gr-complex vectors and function arguments
- Look at code for pack_k_bits_bb to learn about sync_decimator blocks
- Look at code for unpack_k_bits_bb to learn about sync_interpolator blocks
- Look at code for moving_average_XX to learn about the set_history() feature
Conclusion

- GNU Radio is a great addition to the undergraduate communication curriculum.
- It is important to introduce GNU Radio early on to the next generation of communication engineers.
- GNU Radio must be working in the student environment with a minimum of hassle and overhead.
- The building blocks used for the basic education must be simple and their function must be transparent.
- Actual text and audio signals should be used whenever possible to emphasize the applicability of the theory.