gr-pdw: An OOT Module for Pulse Descriptor Word (PDW) Generation

James 'Trip' Humphries TRIP.HUMPHRIES@GTRI.GATECH.EDU
Stan Sutphin STAN.SUTPHIN@GTRI.GATECH.EDU
Brian Mulvaney BRIAN.MULVANEY@GTRI.GATECH.EDU
James Landreth JAMES.LANDRETH@GTRI.GATECH.EDU
Georgia Tech Research Institute (GTRI), Sensors and Electromagnetic Applications Laboratory (SEAL)
7220 Richardson Road, Smyrna, GA 30080, USA

Abstract

Characterization of pulsed waveforms results in measurements commonly referred to as pulse descriptor words (PDW). PDWs are often utilized for radar sensor characterization, identification, and emulation. Pulse measurements are typically provided as an option on high-SWaP, high-cost laboratory test equipment or as custom FPGA firmware which is not easily scalable to multiple SDR/RF platforms. gr-pdw is an out-of-tree (OOT) module for GNU Radio developed by GTRI that performs pulse detection and PDW generation. The intention is that gr-pdw provides the ability for any commercial-off-the-shelf (COTS) SDR to perform PDW measurements. Currently it is capable of measuring pulse width, power, frequency, and time-of-arrival (ToA), and signal-to-noise ratio (SNR). gr-pdw also provides blocks for writing PDWs to an HDF5 formatted log file and visualizing measurements in GNU Radio Companion. Testing has been performed with a USRP B210 in both laboratory and field settings with measurements comparable to currently used PDW generators.

1. Introduction

Pulse descriptor words (PDW) are commonly employed to study, characterize, and identify radar systems (Wiley, 2006). PDWs measure various properties of detected RF pulses including pulse width, pulse power, time of arrival (TOA), frequency, and modulation.

This paper presents a GNU Radio approach to generating PDWs. gr-pdw is a utility OOT meant to facilitate radar characterization and support other GNU Radio radar modules such as gr-radar, gr-ofdmradar, and gr-plasma (Wunsch; Analog Devices, Inc.; Flandermeyer et al., 2022). This OOT includes the necessary functions to measure PDW data including pulse detection, pulse characterization measurements, data visualization, and data logging. gr-pdw is agnostic to the source of complex data which enables SDR, I/Q recordings, or virtual pulse generators as inputs.

gr-pdw is free and open-source and available on GitHub:

https://github.com/gtri/gr-pdw

1.1. Motivation

GTRI often participates in tests and exercises in rugged environments. These environments include desert (extreme heat, dust), marine (moisture, corrosion), and mountainous (rough terrain, snow) areas. COTS laboratory grade test equipment is available that can generate PDWs, but they are not suitable for these environments and are very cost prohibitive (Hardware and software license can easily exceed $250k US in 2024). GTRI also has custom equipment capable of the same but these are often mission critical and cannot be appropriated for other tasks beyond its main objective.

Typical PDW generators require an FPGA based approach, especially where sensor's waveforms must be accurately characterized at the PRI and dwell level (Pelan, 2013; Jing Wen et al., 2015). GNU Radio and SDR offers an attractive platform for this application. These offer a lower barrier to entry for generating PDWs due to the wide variety of SDR platforms available (for both cost and performance) and the utilization of GNU Radio as a free and open source signal processing framework. GTRI developed gr-pdw to take advantage of the various COTS SDR platforms already available at GTRI and enable additional methods for collecting PDW data.

2. gr-pdw

gr-pdw is an OOT that contains blocks to perform the steps of PDW generation. Figure 1 is a block diagram depicting the operation of gr-pdw in GNU Radio. An input stream of

Proceedings of the 14^{th} GNU Radio Conference, Copyright 2024 by the author(s).

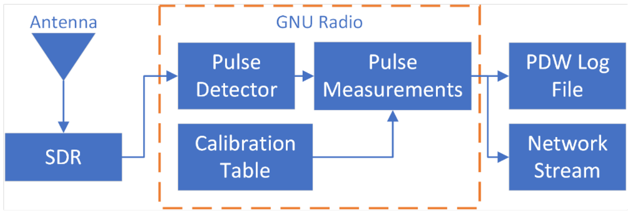


Figure 1. Block diagram of generating PDWs with GNU Radio. Pulses are detected from an input data stream and passed to a seperate block to perform measurements on the pulse.

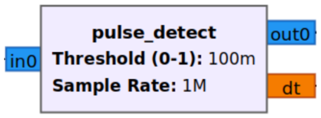


Figure 2. pulse_detect: Tags the start and stop of pulses that exceed a constant threshold value.

samples (from an SDR or digital source such as the signal generator in GNU Radio) is passed into a block that performs pulse detection which adds a tag to identify the start and stop of the pulse in the sample stream. The next block extracts the I/Q data of the pulse and performs the PDW measurements. After the PDW is generated it can be written to a log file (HDF5) or plotted with an included block for PDW visualization. This section describes each block's operation and input parameters.

2.1. Block: Pulse Detection

The pulse detection block (pulse_detect) tags the start and stop of a pulse based on a constant threshold value. Stream tags are added in pairs with "pdw_sob" indicating the start of the pulse and "pdw_eob" indicating the end of the pulse. The output of this block is identical to the input samples with stream tags added. An option output, dt, is the time derivative of the magnitude of the input samples to assist in debugging pulse detection. Figure 2 is the block as shown in GNU Radio companion. An example output of the tagged stream and the derivative debug port is illustrated in Figure 3.

Pulse start and stop are detected applying a constant threshold. The start of the pulse is marked with the "pdw_sob" tag when the magnitude of the signal exceeds the threshold value. Once the signal drops back below the threshold, the "pdw_eob" tag is added. Each tag also has a value associated with it. The value is the received time of the sample where the first sample received is assumed to be zero (0).

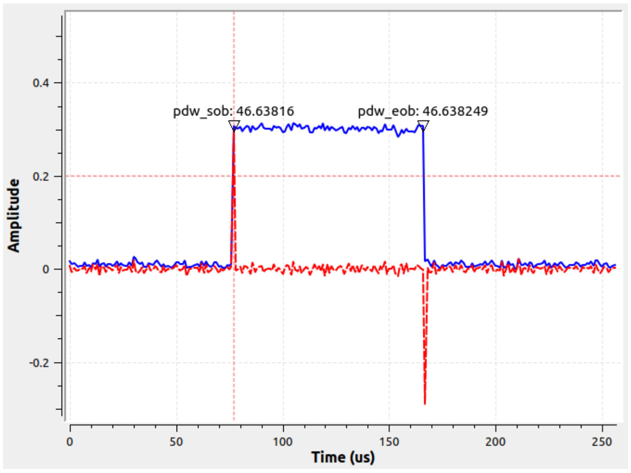


Figure 3. Example output of the pulse detection block. The start / stop of the observed pulse is seen with the "pdw_sob" and "pdw_eob" tags (solid blue). The derivative output (dashed red) helps indicate the beginning and end of the pulse.

Parameter	Description
Threshold	Linear pulse threshold (0-1)
Sample Rate	Flowgraph sample rate

Table 1. Block parameters for pulse_detect.

2.2. Block: Pulse Measurement Extraction

The pulse measurement block (pulse_extract) extracts the I/Q between pairs of "pdw_sob" and "pdw_eob" tags. This section documents the series of measurement steps to calculate pulse width, pulse power, and pulse frequency.

Calculation of pulse width utilizes the time value associate with the SOB and EOB tags for each pulse. Pulse width in seconds is given by:

$$Pulse\,Width = Time_{EOB} - Time_{SOB}$$

The rest of the PDW calculations use the I/Q samples of the pulse, referred to as IQ in the following equations and assumed to be a numpy array.

Power is calculated from the magnitude of the sample in the middle of the IQ window. The sample is selected by:

Parameter	Description
Sample Rate	Flowgraph sample rate

Table 2. Block parameters for pulse_extract.

$$sample = IQ\left[int\left(\frac{len(IQ)}{2}\right)\right]$$

Decibels relative to full scale ($dbfs$) is calculated as:

$$dbfs = 20 * np.log10(np.abs(sample))$$

and subsequently the power is determined by:

$$Pulse\ Power = dbfs + Ref_{dBm}$$

where Ref_{dBm} is the power value in dBm that corresponds to a signal magnitude of 1 as specified by the calibration table. The reference level is specified by the user prior to runtime.

Pulse frequency measurement is performed via FFT of a zero padded signal. The IQ samples of the extracted pulse are zero padded prior to the FFT to interpolate between points in the frequency domain. Again, IQ is assumed to be a complex numpy array and Z_N refers to the number of zeros to pad at the beginning and end of the signal:

$$IQ_{PAD} = np.pad(IQ, (Z_N, Z_N), mode = `constant')$$

IQ data is assumed to be baseband and thus the FFT is calculated with:

$$X = np.fft.fftshift\left(np.fft.fft(IQ_{PAD})\right)$$

where X is the Fourier transform of IQ_{PAD}. FFT frequencies are:

$$freqs = np.arange\left(-\frac{F_S}{2}, \frac{F_S}{2}, \frac{F_S}{N}\right)$$

Finally, the frequency of the pulse, $Freq_{pulse}$ is found by:

$$Freq_{pulse} = freqs\left[np.argmax(np.abs(X))\right]$$

If the measurement is with an SDR, the RF tune frequency is added to this measurement. Each of these measurements are stored into a Python dict and emitted from this block as a message.

2.3. Block: Power Calibration

Pulse power measurements require calibration of the receiver to map signal magnitude to power across frequency and gain settings. gr-pdw includes a power calibration table for the USRP B210 that was generated using the power calibration routine included in UHD (Ettus Research Power Calibration). An R&S SM100A signal generator was utilized as the calibration source. Figure 5 is a photograph of the test setup with a USRP B210, signal generator, and laptop to control the calibration process. Calibration was performed in gain steps of 10dB across the entire tuning range of the B210. An excerpt of the calibration table is plotted in Figure 6 for reference levels of -10 dBm, -20 dBm, -30 dBm, and -40 dBm.

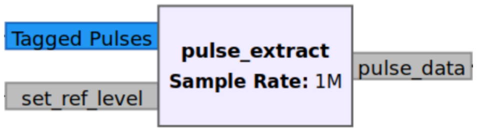


Figure 4. pulse_extract: Generates pulse measurements (PDW) from detected pulses.

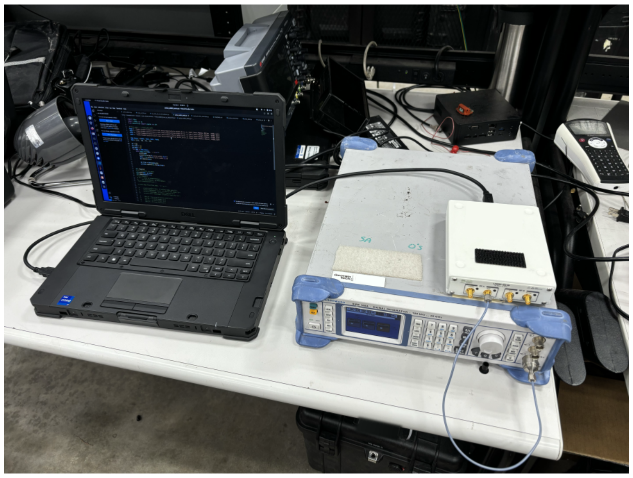


Figure 5. Power calibration and test setup. The laptop performs the automated calibration by commanding the signal generator (R&S SM100A) to output signals are the specified frequencies and power levels. UHD power calibration utilities generate the calibration table.

This calibration file is included with gr-pdw and has been tested with multiple B210 with comparable accuracy in power measurements. If better accuracy is required, each SDR would need a unique calibration table. A future update to gr-pdw will allow the selection of the calibration table. The USRP calibration block is shown in Figure 7 with the input parameters described in Table 3.

A virtual calibration block is also included. This block simply passes the reference level to the pulse extraction block to map the reference level to the signal magnitude of 1. The virtual calibration block is shown in Figure 8 with the input parameters described in Table 4.

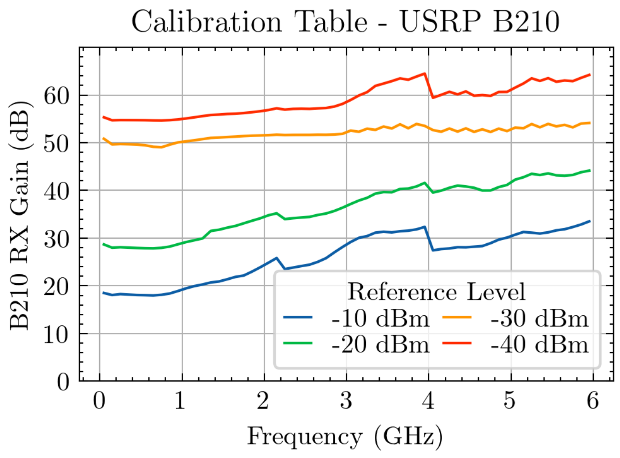


Figure 6. Excerpt of the generated power calibration table for a USRP B210. The usrp_power_cal_table uses this table to load the appropriate gain setting for a selected reference level. Gain values are interpolated if a specific reference level or frequency is not in the table.

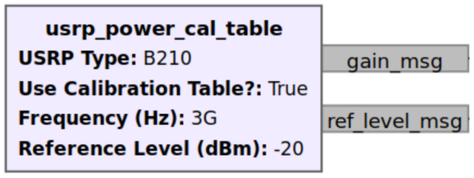

usrp_power_cal_table
USRP Type: B210
Use Calibration Table?: True
Frequency (Hz): 3G
Reference Level (dBm): -20

Figure 7. usrp_power_cal_table: Selects USRP gain value from a calibration table based on frequency and reference level.

2.4. Block: PDW to File (pdw_to_file)

The PDW to file block (pdw_to_file) logs PDW messages to a file while the flowgraph is running. This enables PDW data to be stored and analyzed later. PDW measurements are stored in the HDF5 format (The HDF Group) which allows the PDW to be stored in a portable format containing the PDWs and other meta-data about the data log. The block as seen in GNU Radio companion is shown in Figure 9. A table of block parameters is also given in Table 5.

The PDW log file is comprised of a header, which contains general information about the PDW generation, and the datasets, which contain the actual PDW measurements in their respective fields. The header and dataset fields are described in Table 6. gr-pdw utilizes h5py (Andrew Collette) for PDW logging and data retrieval although any HDF5 compatible tool set should work to read the PDW logs. An example Python script is included within the examples folder of gr-pdw.

Parameter	Description
USRP Type	Type of USRP for calibration table selection
Use Calibration Table?	Enable or disable calibration table use
Frequency	USRP RF frequency for calibration table
Reference Level	Reference level that maps to signal magnitude of 1

Table 3. Block parameters for pulse_detect.

Parameter	Description
Frequency	Frequency for virtual calibration table
Reference Level	Reference level that maps to signal magnitude of 1

Table 4. Block parameters for pulse_detect.

2.5. Block: PDW Plotter

The PDW plotting block (pdw_plot) plots PDW measurements from the PDW messages. The user selects the type of measurement to plot (pulse width, pulse power, or pulse frequency) along with a few other inputs. Figure 11 is the block as shown in GNU Radio companion. A list of the block input parameters is given in Table 7. This block makes use of embedded matplotlib plots for displaying the visualizations. The update rate of the block limits the amount of PDWs that will actually be plotted as high PRF scenarios can generate thousands of PDW measurements each second. For complete PDW measurements the pdw_to_file will log all of the pulse data.

PDW plots may be combined with other GNU Radio companion GUI elements. An example of a GUI in GRC with various elements is shown in Figure 10 which includes the

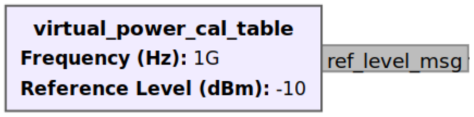

virtual_power_cal_table
Frequency (Hz): 1G
Reference Level (dBm): -10

Figure 8. virtual_power_cal_table: Maps signal magnitude to the selected reference level.

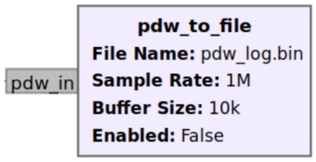
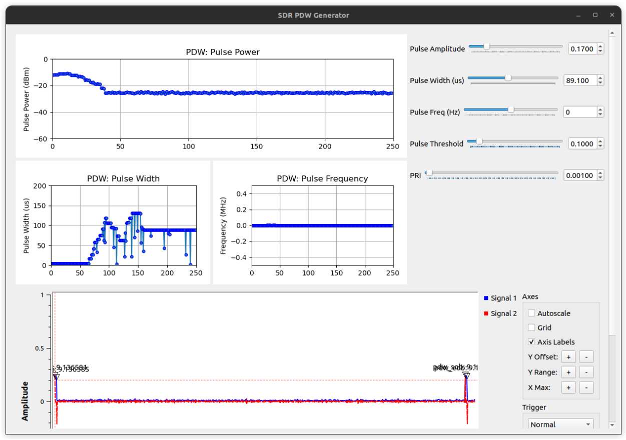

pdw_to_file
File Name: pdw_log.bin
Sample Rate: 1M
Buffer Size: 10k
Enabled: False

Figure 9. pdw_to_file: Writes PDW measurements to a log file in the HDF5 format.

Parameter	Description
File Name	PDW log file name
Sample Rate	Flowgraph sample rate for HDF5 meta-data
Buffer Size	Number of PDW entries to buffer before writing to file
Enabled	Enable / Disable writing PDW messages to file.

Table 5. Block parameters for pdw_to_file.



Pulse Amplitude 0.1700
Pulse Width (us) 89.100
Pulse Freq (Hz) 0
Pulse Threshold 0.1000
PRI 0.00100
Trigger: Normal

Figure 10. Example GUI layout with gr-pdw plots in GNU Radio companion. Three (3) PDW plots (Upper left) are enabled (width, power, and frequency) and combined with QT Time Sink and various QT Range sliders.

PDW plots, QT time sink, and QT sliders for parameter controls.

2.6. Block: Pulse Generator (Hier)

The pulse generator block is a hierarchical block that generates rectangular pulses based on the input parameters amplitude, pulse repetition interval (PRI), and pulse width. The block as seen in GNU Radio companion is shown in Figure 12 and the internal components of the hier block is shown in Figure 13.

Pulse samples are calculated in the heir block based on the input parameters described above. The pulse samples are padded with zeros to the appropriate PRI. These samples are stored in a vector source block which repeat enabled. Amplitude, pulse width, and PRI may all be adjusted while the flowgraph is running allowing changes in PDW measurements to be viewed in real-time.

Header	
Field	**Description**
time_unix	Unix timestamp in integer seconds
time_py	Python datetime timestamp
samp_rate	Sample rate of SDR during PDW measurements
ref_level	Power reference level used during PDW measurements
Datasets	
Dataset	**Description**
pulse_width	Pulse width measurements in seconds
pulse_power	Pulse power measurements in dBm
noise_power	Noise power measurements in dBm
freq_start	Pulse frequency measurements in Hz
toa_course	Course time-of-arrival for each pulse in integer seconds
toa_fine	Fine time-of-arrival for each pulse in fractional seconds

Table 6. Header and dataset fields in the PDW HDF5 log file.

3. Measurements

This section presents an example of pulse measurements using a USRP B210 SDR as the pulse receiver and an R&S SM100A signal generator. A photograph of the test setup is shown in Figure 5. Keep in mind that precision and accuracy of pulse measurements vary greatly with measurement setup. Such factors may include SNR (background noise, interference), antenna pointing (antenna alignment to device being characterized), and environment (rain, foliage, terrain, buildings) as a few examples. Also consider that there are countless combinations of experiments that could be performed (varying frequency, input power, reference level, pulse width, pulse repetition interval, etc). The results shown here are meant to give a general idea of performance.

3.1. Pulse Power

Power measurements were made across frequency and for various input power levels from the signal generator. The first test measured pulse power using the USRP B210 cal-

Parameter	Description
Plot Data	PDW Data to Plot (Pulse width, power, or frequency)
Plot Y-Limits	Min/Max plot limits for the Y-Axis ([Min, Max])
Max PDW Per Update	Max PDW measurements to plot during each update
Max PDW to Display	Max number of PDW measurements in plot history.
Plot Update Rate	Time between plot updates.

Table 7. Input parameters for the pdw_plot block.

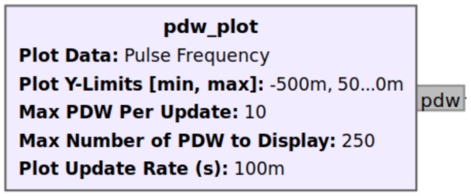

pdw_plot
Plot Data: Pulse Frequency
Plot Y-Limits [min, max]: -500m, 50...0m
Max PDW Per Update: 10
Max Number of PDW to Display: 250
Plot Update Rate (s): 100m

Figure 11. pdw_plot: Visualization of PDW measurements in embedded matplotlib plots.

ibration table across its tuning range. Reference level was set to -20 dBm and the signal generator was configured to output a -30 dBm pulse while varying frequency. Results of this test are shown in Figure 15 where a small measurement error can be seen at some frequencies but overall all measurements are within a few dB.

Next, the signal generator was set to generate a 3 GHz pulse while varying the pulse power. Reference level for the SDR was set to -20 dBm for the calibration table. Pulse power was measured starting at -20 dBm and ending at -70 dBm. Measurement results are given in Figure 16 where power was accurately measured down to -70 dBm. Note that when the pulse power is set to -20 dBm and the reference level is also -20 dBm the SDR will be saturated, leading to erroneous measurements as exhibited at this configuration in

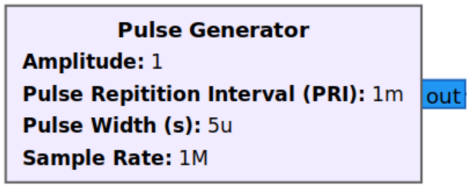
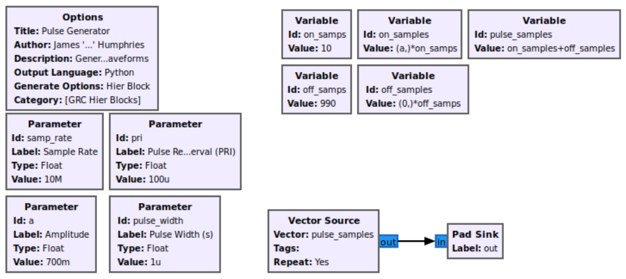

Pulse Generator
Amplitude: 1
Pulse Repitition Interval (PRI): 1m
Pulse Width (s): 5u
Sample Rate: 1M

Figure 12. Pulse Generator: Hierarchal block that creates a rectangular pulse waveform with a user specified amplitude, PRI, and pulse width.


Options
Title: Pulse Generator
Author: James '...' Humphries
Description: Gener...aveforms
Output Language: Python
Generate Options: Hier Block
Category: [GRC Hier Blocks]

Variable
Id: on_samps
Value: 10

Variable
Id: on_samples
Value: (a,)*on_samps

Variable
Id: pulse_samples
Value: on_samples+off_samples

Variable
Id: off_samps
Value: 990

Variable
Id: off_samples
Value: (0,)*off_samps

Parameter
Id: samp_rate
Label: Sample Rate
Type: Float
Value: 10M

Parameter
Id: pri
Label: Pulse Re...erval (PRI)
Type: Float
Value: 100u

Parameter
Id: a
Label: Amplitude
Type: Float
Value: 700m

Parameter
Id: pulse_width
Label: Pulse Width (s)
Type: Float
Value: 1u

Vector Source
Vector: pulse_samples
Tags:
Repeat: Yes

Pad Sink
Label: out

Figure 13. Components of the pulse generator hierarchal block. One PRI worth of samples is generated and stored in a vector source block with repeat enabled.

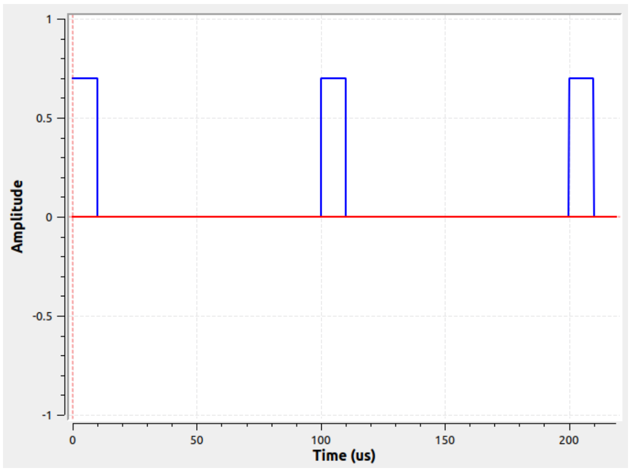


Figure 14. Example output of the pulse generator block. The user selects pulse width, pulse amplitude, and pulse repetition interval (PRI). These parameters can be updated while the flowgraph is running.

the test.

3.2. Throughput

Pulse measurement throughput is influenced by many factors. For example, if the PRI of the waveform decreases (PRF is increasing), then the number of detections per unit time is increasing. Other factors may also influence throughput including sample rate, pulse width / amplitude, and other flowgraph functions such as plotting.

This test was performed on the following machine:

- Model: Dell 5430 Rugged
- Processor: Intel Core i7-1185G7 3.0 GHz
- RAM: 64 GB
- OS: Ubuntu 22.04
- GNU Radio: 3.10.10

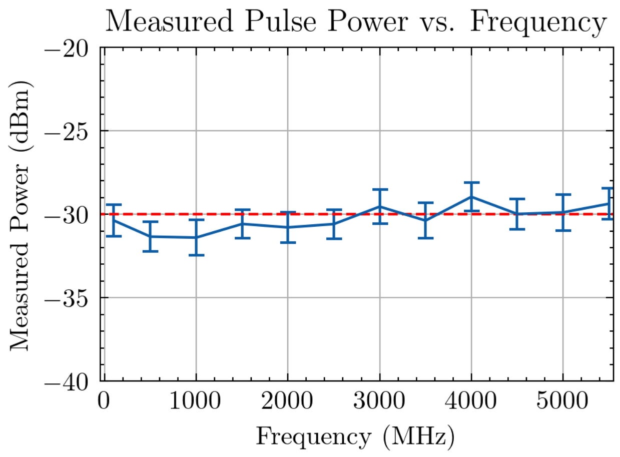


Figure 15. Measured pulse power with an input signal of -30 dBm over the frequency range of the USRP B210. A measurement error of a few dBm is evident at some frequencies. The error bars indicate one standard deviation over approximately 30 seconds of observations at each frequency.

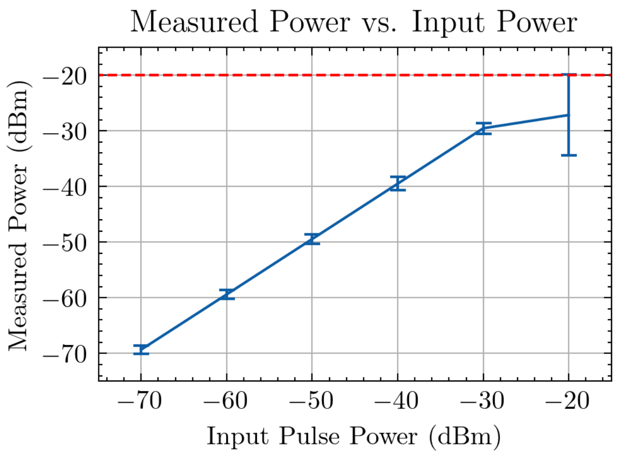


Figure 16. gr-pdw measured pulse power with various input powers from a signal generator. Here the reference level is set to -20 dBm. gr-pdw was able to measure down to -70 dBm accurately with the USRP B210. Note that when the input power is equal to the reference level, the SDR receiver will be saturated and will produce erroneous measurements.

The virtual flowgraph was configured with a $5\mu s$ pulse, a pulse amplitude of 1.0, and a threshold of 0.1. The PRI of the waveform was varied and the throughput of the flowgraph was measured with the “Probe Rate” block. Figure 17 shows the results of these measurements. Here the throughput was normalized to the sample rate such that a normalized throughput of 1 indicates that the flowgraph is keeping up with processing in realtime.

It is apparent that both waveform PRI and sample rate do affect the throughput as expected. Of course this will vary from machine to machine depending on processing resources but this helps illustrate where Python block processing may be a bottleneck. Another interesting observation is that the throughput curves are tending towards 0 throughput. This would correspond to a PRI of 0 ms where each pulse would become indistinguishable and thus no pulse could be detected.

4. Conclusion

In this paper, we have presented a new OOT module for GNU Radio, gr-pdw, which is a tool to generate PDW measurements of pulsed waveforms. This OOT provides blocks for pulse detection, PDW generation, power calibration tables, PDW log files, and PDW visualizations. The presented results demonstrate that this tool can accurately measure pulse width, pulse frequency, and pulse power. gr-pdw was designed as a tool to faciliate characterization of radar systems but it can be applied to any waveform that is comprised of pulses.

Additional features are planned to enhance the utility of gr-pdw. Some of the planned additions include:

- Port blocks to C++: Increase speed of pulse detection and PDW generation for higher sample rates and higher PRF scenarios.
- Modulation detection: Determine pulse modulation such as linear frequency modulation (LFM) or phase shift keying (PSK) modulation.
- PDW Network Streaming: Enable external tools the ability to received PDW measurements via network connection.
- Pulse I/Q Export: Include the raw I/Q data from each pulse into the PDW.
- RFNoC Module: FPGA based implementation of gr-pdw for very high bandwidth (GHz range) and PRF.
- SigMF Integration: Write PDW measurements to SigMF recordings as meta-data.
- Adaptive Pulse Detection: Adapt CFAR methods to dynamically adjust threshold for pulse detection.

GTRI plans to continue development of gr-pdw and provide these additional features on GitHub. gr-pdw is available now on GitHub:

https://github.com/gtri/gr-pdw

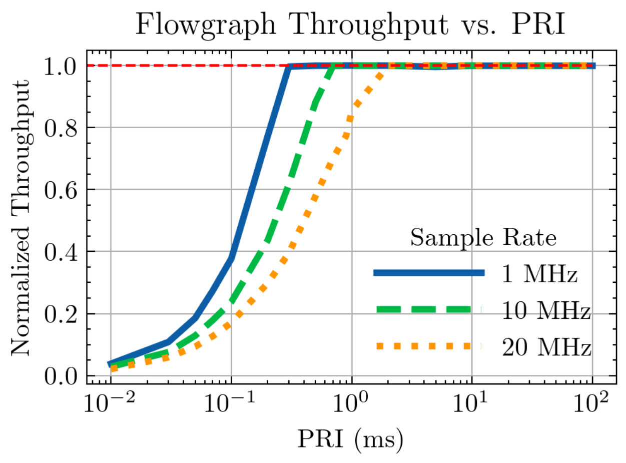


Figure 17. Measured flowgraph input as a function of the waveform PRI. As the time between pulses decreases, the flowgraph throughput also decreases at a certain point.

References

Richard Wiley. ELINT: The Interception and Analysis of Radar Signals, 2006. URL https://ieeexplore.ieee.org/document/9101176.

Stefan Wunsch. gr-radar. URL https://github.com/kit-cel/gr-radar.

Analog Devices, Inc. gr-ofdmradar. URL https://github.com/analogdevicesinc/gr-ofdmradar.

Shane Flandermeyer, Rylee Mattingly, and Justin Metcalf. gr-plasma: A New GNU Radio-based Tool for Software-defined Radar. *Proceedings of the GNU Radio Conference*, 7(1), 2022. URL https://pubs.gnuradio.org/index.php/grcon/article/view/121.

Justin Darrell Pelan. Modular Multi-Signal Tracking Pulse Descriptor Word (PDW) Generator with Field Programmable Gate Array (FPGA) Implementation. 2013.

Jing Wen Jing Wen, Lei Wang Lei Wang, ChunHong Zhang ChunHong Zhang, and SiSi Chen SiSi Chen. FPGA implementation of PWD generator for multi-channel interferometer radar. In *IET International Radar Conference 2015*, pages 4 .–4 ., Hangzhou, China, 2015. Institution of Engineering and Technology. ISBN 978-1-78561-038-7. doi: 10.1049/cp.2015.1254. URL https://digital-library.theiet.org/content/conferences/10.1049/cp.2015.1254.

Ettus Research Power Calibration. UHD Power Level Controls. URL https://files.ettus.com/manual/page_power.html.

The HDF Group. Hierarchical Data Format, version 5. URL https://github.com/HDFGroup/hdf5.

Andrew Collette. h5py. URL https://www.h5py.org/.