A Software Test Suite for Data Acquisition and GPU-based Computations on IoT Edge Devices for High Frequency Power Grid Monitoring

Justin Johnson
ECSE Department.
Rensselaer Polytechnic Insitute
Troy, NY, United States
johnsj1821@gmail.com

Luigi Vanfetti
ECSE Department
Rensselaer Polytechnic Insitute
Troy, NY, United States
luigi.vanfretti@gmail.com

***Abstract*—This paper discusses the design and testing results of a highly configurable software suite for data acquisition and GPU-based computation suitable for Internet-of-Things devices supporting both the I2C protocol and equipped with GPUs that support CUDA. The IoT device used is the NVIDIA Jetson TX2, which ingests analog sensor data via the I2C protocol. Thus, to ingest data, a development board was developed, and to test the data acquisition and processing pipeline, a suite of software was developed so that automated tests can be performed repeatedly using the same testing parameters. This suite of software coordinates several devices for testing, communicating over TCP/IP. To take advantage of GPU capabilities in the NVIDIA Jetson TX2 platform used in the experiments in this paper, FFT-based analysis was performed using two different algorithms, one exploiting the on-board GPU capabilities. The results show the great potential of these capabilities for edge computations by comparing several FFT algorithms w.r.t. to their processing speed. The analysis from experiments conducted with both CPU and GPU-based algorithms show how IoT devices with GPU capabilities can meet the real time computational requirements posed by high frequency power grid monitoring applications.**

***Index Terms*—FFT, CUDA, ADC, GPIO, I2C, NVIDIA, Jetson**

I. Introduction

1) Motivation: The ongoing integration of modern solid state power electronics-based devices into the grid has given rise to undesirable high frequency oscillations as evidenced in recent studies from real-world solar PV [1], STATCOM [2] and wind farms [3], [4], installations. Existing substation automation equipment have major difficulties to capture such high frequency oscillations (see [1]), and as oscillatory frequencies continue to grow (e.g., as those from VSC-HVDC [5]), a new generation of high speed monitoring equipment needs to be developed. At the heart of such equipment, the embedded computer can offer unique capabilities that were previously unavailable in typical substation devices based on microprocessors. For example, a new generation of Internet-of-Things (IoT) devices offer advanced on-board capabilities such as Graphical Processing Unit (GPU)-based computations (e.g. machine learning [6]) that can be exploited for high frequency power grid monitoring applications, such as the monitoring of forced oscillations [6].

To facilitate the efficient development of these devices the testing of components needs to be rapid and repeatable [7]. In this work, we present a software test suite developed to allow for consistent and highly customizable testing of analog to digital converters and prototyping of GPU-based on-edge computations, such as FFT-based monitoring [4] using GPUs supporting CUDA [8].

Modularity was a key feature when designing these pieces of software. In order to enable this system to have maximum adaptability the data transfers between modules were handled at a very low level. Cross program instructions are handled by simple text sent over TCP/IP packets and data is stored in plain text files. By doing this, other modules are able to be easily added to the testing process without any proprietary libraries.

The setup used in experiments mentioned in this paper involved playback of waveforms stored as audio files on one computer to an analog to digital convertor(ADC) connected to an embedded development board, the NVIDIA Jetson TX2[1] . This data was then sampled, processed, analyzed, and graphed on the development board.

2) Problem statement: This paper aims to provide a design of a highly adaptable and automated software suite that can be used to test and validate analog to digital converters and their accompanying development boards. This allows for the rapid prototyping of various development boards under identical testing conditions so that an accurate comparison of results can be performed. The software suite was also used to validate and compare different processing algorithms that have recently become available on IoT devices, i.e. CUDA-based GPU computations. While the experiments performed in this paper use a FFT approach other algorithms could be easily swapped in such as others that instead use machine learning [6].

3) Contributions: Succinctly, the contributions of this paper are:

[1]https://developer.nvidia.com/blog/jetson-tx2-delivers-twice-intelligence-edge/

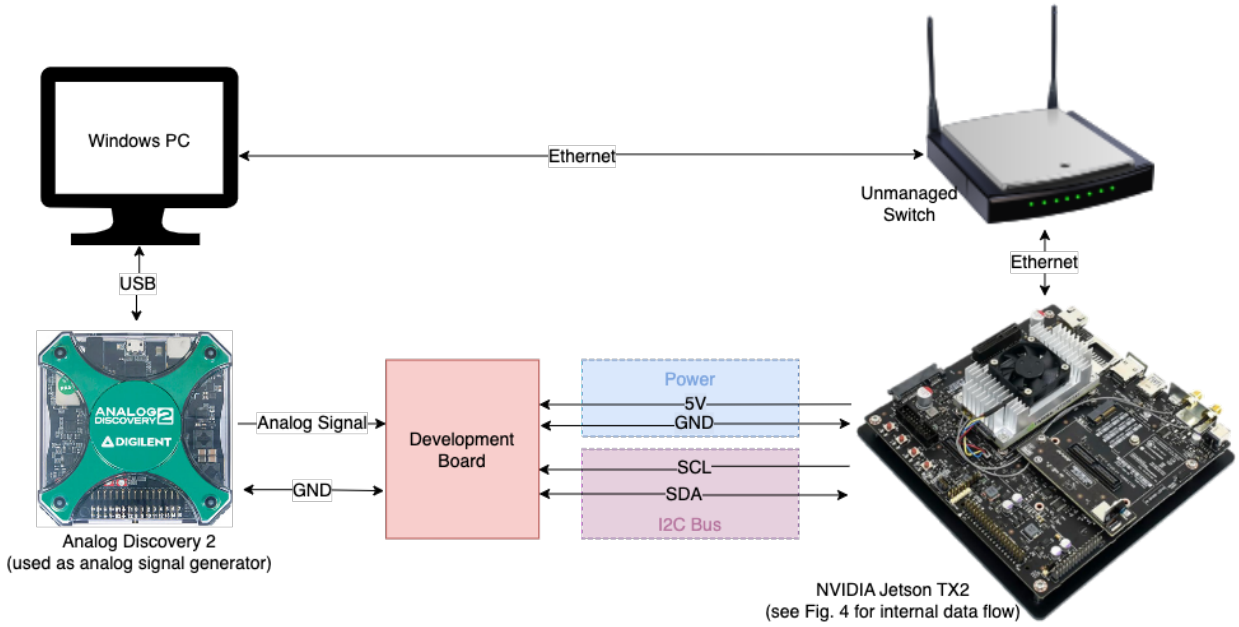

Fig. 1. Low-cost hardware platform for testing the data acquisition board and NVIDIA Jetson TX2. Acronyms — I2C: Inter-Integrated Circuit, SCL: Serial Clock, SDA: Serial Data.

(i) Evaluate FFT algorithms on both CPU and GPU and assess their viability for real time processing.
(ii) Provide a system to create repeatable tests for ADC development boards
(iii) Create a modular system to allow for validation of various processing algorithms on data collected from sampling
(iv) Create a dataflow that allows for on edge processing and real time data streaming

The remainder of this paper is organized as follows. Section II describes the experimental setup for data acquisition and test automation, while Section III explains how GPU-based computations can be performed using ultra high-speed data acquisition on the NVIDIA Jetson TX2. Our experimental results are summarized in Section IV while Section V concludes this paper.

II. Data Acquisition and Test Automation

A. Data Flow

The experimental setup for the experiments performed on the software suite in this paper had three main components. The first was the generation of the waveforms, this was achieved using an Analog Discovery Board 2[2] and the WaveForms SDK[3] with Python. The next step is the sampling of data. This was done using a custom analog to digital converter development board connected to the Inter-Integrated Circuit (I2C) bus of a NVIDIA Jetson TX2. The third and final step is the processing (FFT computations) and graphing of the collected data. This was also done on the TX2 using a combination of C++ and Python and is described in the next section. See Fig.1 for the full data flow.

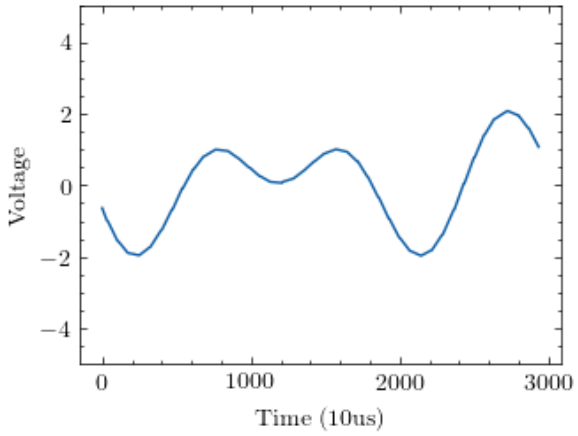

Fig. 2. 30 milliseconds of data sampled from a generated waveform using the ADC.

B. Waveform Storage and Playback

The waveforms that were used for testing are stored in `.wav` files. The reason for this is that the Digilent WaveForms SDK only supports playback of custom waveforms through `.wav` files. A Python script was used to generate test waveforms of various voltage amplitudes and frequencies. The purpose of these files was to validate the sampling abilities of ADC's being tested.

For the experiments in this paper the files were generated as follows, all files had a sixty hertz base sine wave with a secondary sine wave of equal amplitude added to represent an

[2]https://digilent.com/reference/test-and-measurement/analog-discovery-2/
[3]https://digilent.com/reference/software/waveforms/waveforms-sdk/

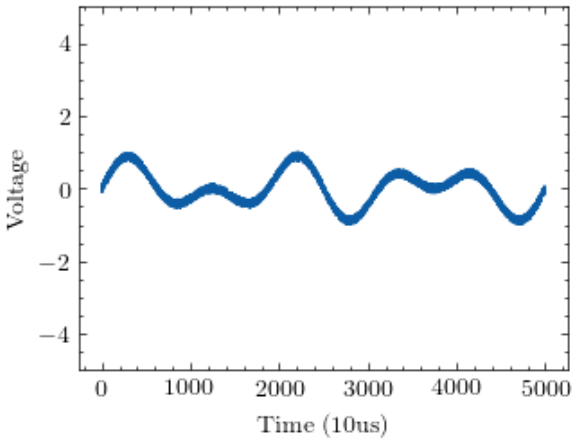

Fig. 3. 50 milliseconds of data sampled from a generated waveform with noise using the ADC.

oscillation see Fig.2. Waveforms were generated for several frequencies of the secondary wave, ranging from 10 Hz to 20,000 Hz. The frequency of the secondary wave was incremented by 10 Hz from 10 Hz to 100 Hz, then was incremented by 100 Hz from 100 Hz to 1000 Hz, and finally they were then incremented by 500 Hz from 1,000 Hz to 20,000 Hz. All of these combinations were then recreated with and without white noise to represent a non-ideal sampling condition, see Fig.3. Finally all of the frequency combinations both with and without noise were generated for 20%, 40%, 60%, and 80% of the dynamic range of the ADC, [0.5 1.0 1.5 2.0] V, noting that in the experiments, 100% was not used to avoid clipping.

C. Analog to Digital Conversion

The ADC that was chosen was a Texas Instruments ADS7823E[4]. It was chosen for the several sampling modes it could operate in. The ADC had three modes: a standard 2 kHz continuous sampling mode, a pulsed 8 kHz mode where 4 samples were taken at a time on a 2 kHz frequency, and the final mode being a 50 kHz continuous mode. Only the 2 kHz and 8 kHz modes were used for the reasons explained below.

For the experiments in this work, there were two technical roadblocks encountered that shaped the process. The first being issues increasing the NVIDIA Jetson TX2's I2C bus's clock speed to use the 50kHz mode. The second was being unable to get the 4-th sample read from the ADC in the pulsed mode, the remaining three samples were left intact and were used for the experiments, but this resulted in an effective 6kHz sampling rate as opposed to the ideal 8kHz.

D. NVIDIA Jetson TX2 Data Connection

The NVIDIA Jetson TX2 had two data connections in the experimental setup. The first being a basic I2C bus connection to the ADC. This connection facilitated the communication to the ADC to read the sample data and to control the ADC. The second connection was a Gigabit Ethernet LAN (Local Area Network) connection. This connection was used to control the waveform generator as well as receive the ideal signal files from the computer attached to the signal generator.

The I2C bus was sufficient at transmitting data from our chosen ADC. The actual limiting factor for the ADCs that could be tested is the 100 Kbit/s limit on the NVIDIA Jetson TX2. Higher speed ADC's either required the I2C clock to be increased or to use a Serial Peripherial Interface (SPI) connection, which is known to break the network interface on the NVIDIA Jetson TX2[5], which was required for test automation and other purposes, as described next.

The use of the Gigabit Ethernet link allows for the potential to network together many of these devices physically distanced from each other. In a real world application where points on the power grid are being sampled these devices could be deployed and perform all necessary processing on board and then send the results to a central location anywhere in the world. For this potential to be realized, time discipline would have to be added to the sampling process so samples could be be syncronized across all devices. Protocols such as IEEE 1588 [9] or IRIG timecode could be used [10]. However, time-synchronization is out of the scope of this work because the focus was first to implement and test the edge data acquisition and GPU-based processing capabilities of a single device, as described in the next section.

III. GPU-based Edge Computing

A. Edge Pre-Processing

With the two modes of sampling the data had to be normalized to be in the proper form before attempting any edge computations (i.e. the FFT). For the single sample mode where there was a regular and continuous sampling period there was no pre-processing that needed to be done other than cropping the data to a uniform number of samples. For the pulsed mode where three samples were taken at a regular interval the data needed to be padded with interpolated data points so it could be sent through a traditional FFT algorithm.

To achieve a continuous data set the original data was scaled to a theoretical 100 kHz sampling rate. A 100 kHz rate was chosen because the individual samples in each pulse were taken between 20-30 μs apart from each other and 100 kHz allows for each sample to have a guaranteed unique time stamp. A frequency greater than 100 kHz could have also been used but to minimize data size and processing time it was kept as low as possible. The new data points were calculated with a linear interpolation using two neighboring data points, as follows

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1$$

where x_1 is the x coordinate of earlier data point, y_1 is the y coordinate of earlier data point, x_2 is the x coordinate of later data point, y_2 is the y coordinate of later data point, x is the desired x coordinate of the interpolated data point, and y is

[4]https://www.ti.com/product/ADS7823

[5]https://forums.developer.nvidia.com/t/how-to-enable-spi-spidev-on-28-1-on-target/53999/17

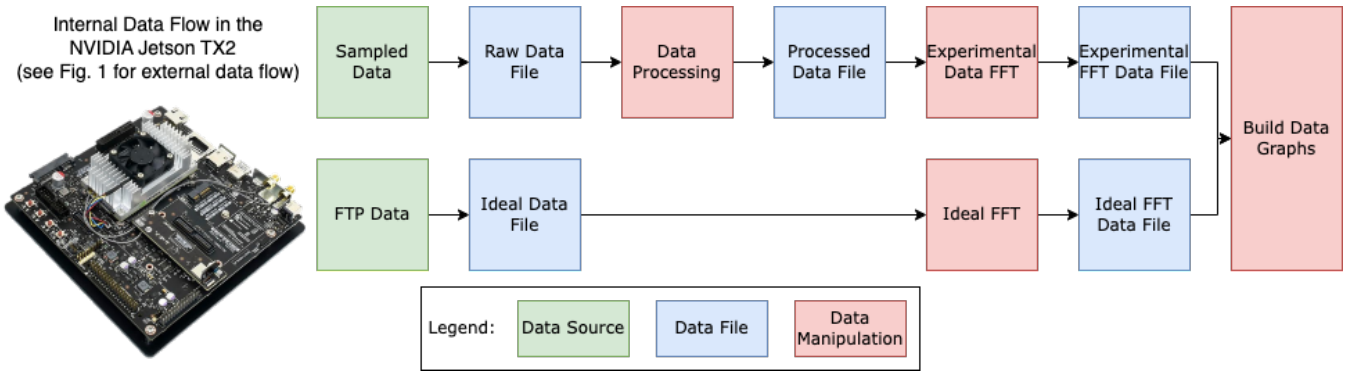

Fig. 4. Data flow within the NVIDIA Jetson TX2.

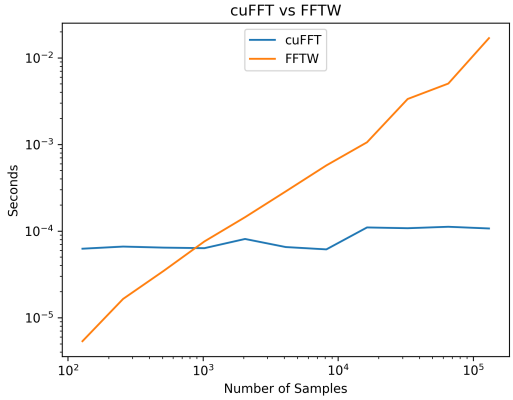

Fig. 5. Computation times for cuFFT and FFTW libraries

TABLE I
FFT EXECUTION TIME FOR FFTW AND CUFFT IN MILLISECONDS

# of Samples	cuFFT	FFTW
128	0.062495	0.005344
256	0.066143	0.016448
512	0.064319	0.034720
1,024	0.063391	0.075807
2,048	0.080960	0.144447
4,096	0.065311	0.286462
8,192	0.061375	0.572155
16,384	0.109983	1.057940
32,768	0.107743	3.343330
65,536	0.111967	5.039600
131,072	0.107007	16.866400

the y coordinate of the interpolated data point given x having the x axis be time and the y axis be voltage

After pre-processing there are two categories of data. The data from the single sample mode is stored without any interpolated data at a sampling rate of 2 kHz, and the data from the pulsed mode which is stored with linearly interpolated data at a sampling rate of 100 kHz. This pre-process normalization and data point interpolation was performed entirely in Python.

B. Edge Fast Fourier Transform

Once the data had been collected and pre-processed on the NVIDIA Jetson TX2 frequency analysis could be performed. To determine the frequencies present in the recorded waveform an FFT was used. Two C++ libraries were used, one that used the Central Processing Unit (CPU) and one that used the GPU. A comparison for the runtimes for both libraries can be seen in Fig.5 and Table I. The CPU of the NVIDIA Jetson TX2 is a (Dual-core NVIDIA Denver 2 64-bit CPU and quad-core Arm Cortex-A57 MPCore processor, and the GPU is a 256-core NVIDIA Pascal architecture GPU. A non-continuous FFT library was also considered but do to a lack of well documented options for both CPU and GPU it was deemed not viable.

For the experiments herein the cuFFT library from NVIDIA [6] was used to perform GPU-computations due to its faster computational speed and wide compatibility with NVIDIA hardware. For the experimental setup the FFT's were run in batches containing all of the collected samples. These batches had up to 500,000 data points in these experiments. Note that these could have been larger if a longer test was run.

Once the data was processed and the FFT was run the data was then ready to be graphed so observations could be made, as described next. For the full data flow inside the Jetson see Fig.4

IV. End-to-End Testing

This section gives an example of how to analyze end-to-end testing results. By using the graphed data (see Figs. 2 and 3) it is possible to see the capabilities and potential errors of the ADC being tested. By observing the sampled data it was possible to see if the ADC was clipping or if it was sampling irregularly. The charts for the various voltages show that as the voltage got closer to the full dynamic range of the ADC the frequency detection was more reliable. Additionally by using the charts for the FFT the maximum reliably detectable frequency can be observed. Figure 6 shows an example of an FFT graph from the experiments performed in this work.

[6]https://docs.nvidia.com/cuda/

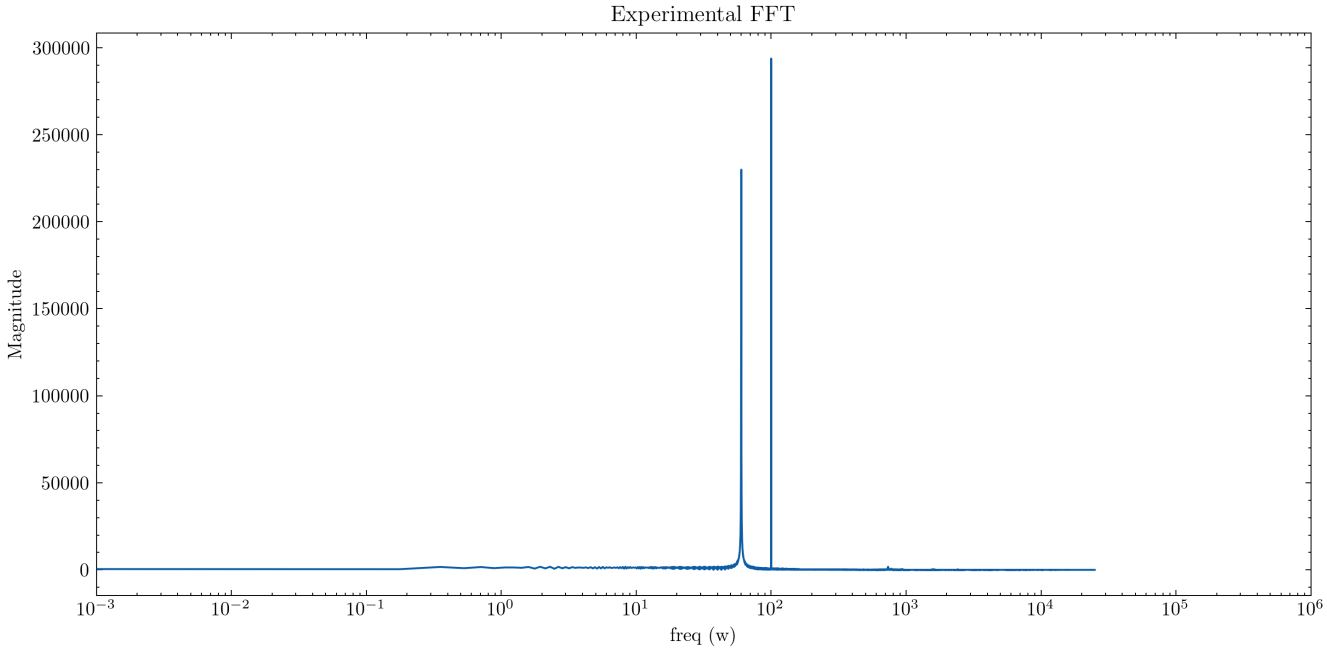

Fig. 6. FFT graph for experimental data with a 60Hz and 100Hz wave.

V. Conclusion

As more power electronic-based devices are integrated into the power grid, the needs for high frequency power grid monitoring rises. Calling for new monitoring equipment capable of performing high-speed data acquisition and advanced computation capabilities at the edge. With the advent of IoT technologies with onboard GPUs and modern networking features, there is tremendous potential to develop new monitoring technologies for high frequency power grid monitoring.

In this work, we built a low cost, easily adaptable, and reliable suite of software that was able to evaluate the capabilities of an analog to digital converters, while also demonstrating the potential of GPU accelerated on IoT edge devices. This software was deployed to a Windows workstation and a NVIDIA Jetson TX2 where an Analog Discovery Board 2, connected to the Windows workstation, was used to generate specific waveforms. Two different FFT libraries were tested using both CPU and GPU-based computation to demonstrate the superior speed of GPU based computation, which is a key enabler for high frequency power grid monitoring at the edge. Finally the data was graphed for ease of human analysis, and showed the possibility of using the existing Ethernet connection to stream data from further processing or analysis after the addition of time disciplining [9] and synchronization [10], which will be subject to future work.

Acknowledgment

This work was funded in part by the Dominion Energy, and in part by the Center of Excellence for NEOM Research at King Abdullah University of Science and Technology.

References

[1] C. Wang, et al., "Identifying Oscillations Injected by Inverter-Based Solar Energy Sources," 2022 IEEE Power & Energy Society General Meeting (PESGM), 2022, pp. 1-5, doi: 10.1109/PESGM48719.2022.9916830.

[2] C. Mishra et al., "Analysis of STATCOM Oscillations using Ambient Synchrophasor Data in Dominion Energy," 2022 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT), 2022, pp. 1-5, doi: 10.1109/ISGT50606.2022.9817489.

[3] L. Vanfretti, M. Baudette, J. L. Domínguez-Garcíaz, A. White, M. S. Almas and J. O. Gjerde, "A PMU-based fast real-time sub-synchronous oscillation detection application," 2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC), 2015, pp. 1892-1897, doi: 10.1109/EEEIC.2015.7165461.

[4] L. Vanfretti, M. Baudette, J.L. Dominguez-Garcia, M.S. Almas, A. White, and J.O. Gjerde, "A PMU-Based Real-Time Oscillation Detection Application for Monitoring Wind-Farm Dynamics," Electric Power Components and Systems, Taylor & Francis, Vol. 44, Iss. 2, 2016. doi: 10.1080/15325008.2015.1101727

[5] Yin, C., Xie, X., Xu, S. and Zou, C. (2019), "Review of oscillations in VSC-HVDC systems caused by control interactions," in The Journal of Engineering, 2019: 1204-1207. https://doi.org/10.1049/joe.2018.8634

[6] S. A. Dorado-Rojas, S. Xu, L. Vanfretti, M. I. I. Ayachi and S. Ahmed, "ML-Based Edge Application for Detection of Forced Oscillations in Power Grids," 2022 IEEE Power & Energy Society General Meeting (PESGM), 2022, pp. 1-5, doi: 10.1109/PESGM48719.2022.9917070.

[7] S. A. Dorado-Rojas, S. Xu, L. Vanfretti, G. Olvera, M. I. I. Ayachi and S. Ahmed, "Low-Cost Hardware Platform for Testing ML-Based Edge Power Grid Oscillation Detectors," 2022 10th Workshop on Modelling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2022, pp. 1-6, doi: 10.1109/MSCPES55116.2022.9770146.

[8] L. Kosmidis, et al. "Embedded GPU benchmarking for high-performance on-board data processing." European Workshop on On-Board Data Processing (OBDP2019). Vol. 29. 2019.

[9] H. Guo and P. Crossley, "Design of a Time Synchronization System Based on GPS and IEEE 1588 for Transmission Substations," in IEEE Transactions on Power Delivery, vol. 32, no. 4, pp. 2091-2100, Aug. 2017, doi: 10.1109/TPWRD.2016.2600759.

[10] M. S. Almas, L. Vanfretti, R. S. Singh and G. M. Jonsdottir, "Vulnerability of Synchrophasor-Based WAMPAC Applications' to Time Synchronization Spoofing," in IEEE Transactions on Smart Grid, vol. 9, no. 5, pp. 4601-4612, Sept. 2018, doi: 10.1109/TSG.2017.2665461.