FluxPMU — A Maker's Guide of a DIY Synchronized Phasor Measurement Unit

Emmett Williamson, Luigi Vanfretti, Prottay M. Adhikari, Jerry W. Dziuba
ECSE Department
Rensselaer Polytechnic Institute
Troy, NY, USA
email: emmettact@gmail.com, vanfrl@rpi.edu, mondap2@rpi.edu, dziubj@rpi.edu

David Laverty
Energy, Power and Intelligent Control
Queen's University Belfast
Belfast, UK
email: david.laverty@qub.ac.uk

***Abstract*—Synchrophasor technology is transforming the way the power grid is being monitored thanks to the benefits provided by synchronized Phasor Measurement Units (PMUs). To exploit these benefits, future power engineers need better understanding of how PMUs work at all levels, from hardware components to phasor estimation methods implemented in software. However, hands-on experiences with these technologies allowing students to "tinker" with a PMU are limited. The FluxPMU project aims to enable students to explore and build an open source, low cost PMU. Building from the legacy of OpenPMU V1, FluxPMU provides a *Do-it-Yourself (DIY)* guide with all documentation and software sources required to build the FluxPMU, which can be found in the following Github repository: `https://github.com/alsetlab/fluxpmu`. This paper summarizes the authors' efforts in creating this *Maker's Guide*.**

I. Introduction

A. Motivation

Major power system outages, such as 2003 Northeast Blackout [1], 2012 India Blackout [2], 2014 Bangladesh Blackout [3], 2019 London Blackout [4], have highlighted the advantages of high update rate and time synchronized measurements for system awareness. One of the main ways this is achieved is through Phasor Measurement Units (PMUs), which produce time sycnhronized synchrophasor measurements. It has been suggested that, implementing more Phasor Measurement Units (PMUs) could have helped control the power loss by providing grid operators with more information in all those cases. It is to be noted that, although nine PMUs were in use during the blackout reported in [2], two of them (in crucial locations) were not connected to communicate with control centers. The reports associated with these blackouts, suggest that these systems lacked situational awareness and system visibility .

As the number of installed PMUs increases in the United States [7], modern PMUs are considered to be a *black box* of digital signal processing. For instance, industry grade protective relays with PMU functionalities or standard PMUs provide the necessary functionalities, but are difficult to adopt for teaching applications as it is challenging to tinker with their internal hardware and functions beyond configuration changes without risking potential damage. Another constraint is that the cost of industry-standard PMUs are over $7000 USD [8], which is too expensive for educational purposes. Additionally, an industrial grade PMU requires special setup to be safe. Using main power line voltage, current and power, poses safety requirements difficult to fulfill in a classroom setting.

The main requirements to consider for an educational PMU are: (i) low cost, (ii) ability to expose students to hands-on learning on the inner workings of the devices, (iii) safety, and (iv) adequate documentation. While the industry grade devices offer plentiful documentation, they do not fulfill requirements (i)-(iii). On the other hand, several open source PMUs that have been created could offer a solution to the cost, but lack the documentation for students use. FluxPMU aims to solve these issues.

FluxPMU is an open-source, low-cost PMU suitable for educational purposes. Being an open source hardware-software solution, FluxPMU would enable the students to explore and modify the entire signal path. Students will have the opportunity to configure and readjust hardware components like CTs and PTs, burden resistors, etc. It is also possible to have direct access to the ADC stage, where analog signals are converted into digital measurements. Apart from hardware and software, the FluxPMU project consists of a documentation dedicated to the build process. The documentation was written specifically to be used by students to ameliorate the build process, and help improve the understanding of each component, and how a PMU works. However, it must be noted that this implementation requires the students to have electrical safety training, as it involves working with high voltages.

B. Previous and Related Work

FluxPMU is based on OpenPMU V1 [9], [15], [16]. OpenPMU V1 has the main benefit from educational purposes to expose the student to all software and hardware aspects that need to be considered for a functional PMU. *National Instruments* has archived the OpenPMU V1 project [24] in their library. The authors have collaborated to build on the legacy of OpenPMU V1 for educational purposes, and thus, the hardware and software is nearly identical between FluxPMU and OpenPMU. The main difference is that the documentation provided by FluxPMU will offer a more comprehensive guide

This work was supported in part by Dominion Energy Virginia and in part by the Center of Excellence for NEOM Research at the King Abdullah University of Science and Technology under grant OSR-2019-CoE-NEOM-4178.12.

specifically tailored for students and educational purposes. The main criticism of OpenPMU V1 [11] and thereby of FluxPMU is that of the use of a National Instruments (NI) 6009 Data Acquisition (DAQ) and the NI LabVIEW [10] software being proprietary. While the dependency on the NI 6009 still remains an issue, the dependency on LabVIEW is now less important as NI has released *LabVIEW Community Edition* in 2020 which provides full featured access for open source projects, however, using this design in a classroom setting would require the University to have licenses available. This last aspect is not of concern for the authors' institutions, which have LabVIEW licenses, and similarly for many other institutions that also subscribe to NI's educational licensing programs.

OpenPMU V2 [17] reports a next generation design replacing the paradigm used in OpenPMU V1, which addresses the previous limitations by using low cost Single Board Computers (SBCs), mainly the *Raspberry Pi* and the *BeagleBone Black*. It is a truly open source and open hardware PMU project, and estimated to cost only around $200 USD [7]. However, the technology stack used for this project is too complex for its use in an educational setting at the undergraduate level.

Other than OpenPMU, it is difficult to find open source, low cost PMU designs suited specifically for students. Several other low-cost PMU designs have been created [11]. Designs, such as FluxPMU, emerge from the pre-SBC era, for example GridTrak PMU from Arnold Stadlin [18] is a classic micro controller-based device that performs all computations in an integrated board, while the DTU-PMU developed at Technical University of Denmark [19], [20] uses a combination of a DAQ, a DOS computer and a Windows 2000 computer. While FluxPMU and GridTrak designs are still possible to build and further develop, the rapid advances of SBC make their integrated features more attractive for new designs. This is evident in the new OpenPMU V2, PhasorsCatcher [11] and the recent efforts from RWTH Aachen University LOCO. A similar implementation was carried out in [23]. However, most of these projects have focus on producing a low cost PMU, while they are unsuitable for educational purposes as they offer limited or no documentation on how to build the PMUs by oneself, with exceptions (GridTrak and OpenPMU V2).

OpenPMU V1 utilizes NI LabVIEW, which uses its "graphical programming environment" for easy visualisation of data within the LabVIEW environment. But, the main drawback of V1 is that the sampling clock of the NI-DAQ can't be synced directly to the GPS time source. This limitation is solved in the OpenPMU V2 hardware with the introduction of a GPS disciplined ADC unit inside the Data Acquisition (DAQ) System [17]. However, this hardware is predominantly driven through command-line and lacks an easy 'graphical interface'. Thus, OpenPMU V2 somewhat requires skills beyond those expected from undergraduate students.

Another option to implement a PMU for educational purposes is through the use of a National Instrument (NI) Compact Reconfigurable Input/Output (CRIO). By selecting input modules that collect voltages and a timing GPS signal, the software run on the FPGA used by a CRIO could model a

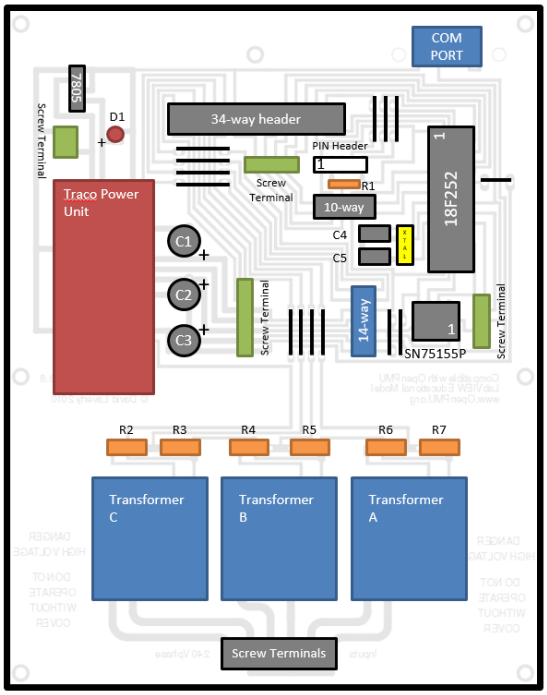

Fig. 1. Hardware layout for the circuit board components of FluxPMU

PMU [14]. However, there are several problems in using a CRIO. First, is the cost of both a CRIO and the modules themselves would easily cost more than other open source hardware examples. Secondly, the CRIO PMU would be mostly software driven and not offer exposure to any other sort of applicable hardware involved in a PMU other than configuring and programming the modules. This can result in a misunderstanding of the importance of both software and hardware aspects in PMUs. Thirdly, CRIO hardware has a steeper learning curve, specially for undergraduate students.

The Github repository provides a Bill of Materials (BOM) with costs for all parts. The total cost depends on market prices and the type of transformers used. Based on recent purchases, costs when using the Hall effect-based PCB will vary from $897.52 to $937.01 and $726.29 and $765.78 for the one using AVB isolation transformers.

C. Contribution

- The entire build process, and all unexpected issues associated with have been documented, along with solutions and work around that were found. The documentation for the build process has been consolidated in a *Maker's Guide* in an attempt to improve ease-of-use and facilitate a student's experience building FluxPMU.
- All of the software associated with FluxPMU was documented to give students a better understanding of what happens behind the scenes. Additional software was also added to give students more resources to understand how FluxPMU works.
- The FluxPMU project adds documentation and work through guides to all of the hardware associated with building the project, along with examples of how

FluxPMU was built in the ALSET Lab of Rensselaer Polytechnic Institute.

II. Background

A Phasor Measurement Unit is a device that measures the voltage and/or current in power lines based on phasor representation of those periodically varying quantities. In measuring the phasor, a PMU produces a synchrophasor measurement by combining the phasor measurement with a time stamp. In most cases, PMUs combine results from several measurement-windows in order to publish a single output value. Time stamps typically come from Global Positioning System (GPS) satellites. In doing so, power grid operators can use the time synchronous measurements to perform more accurate data collection and analysis when multiple PMUs are used. A complete treatment of PMU technology is outside the scope of this work. A more comprehensive background information on PMUs is provided in [22], [9].

III. Hardware

The implementation of FluxPMU can be broadly separated into two parts,

- An electronic board for the acquisition of electrical signals. NI 6009 DAQ is used for acquiring the signals and converting them from analog to digital quantities.
- Algorithms implemented in LabVIEW that acquire the data for phasor estimation executed within a PC.

A simplified system diagram of FluxPMU is shown in Fig.3. The hardware for FluxPMU consists of an assortment of components for the circuit board itself, along with several external devices. The bulk of the hardware comes from the components that are soldered on to the FluxPMU board, as shown in Fig. 1. This board is a standard HV-transducer board. A comprehensive list of these components is shown in the bill of materials (within the *Maker's list*). The basic layout for the circuit board hardware is shown in Fig. 1. It is important to note that, this part of the work requires the usage of high-voltage equipment, which should be used only under the supervision of a qualified individual.

The external components consist of the following:

A. PIC 18F2525

The PIC micro controller is used to work with the GPS NMEA signals received via the RS-232 standard, and output a code used by FluxPMU. Additionally, the PIC micro controller generates the 60 Hz (or 50 Hz) pulse train that triggers the NI-6009. Once the PIC records the NMEA data, it parses through that data to retrieve the time information. The PIC micro controller essentially enables the time transfer to the PC. The way this is achieved is that the NI-DAQ toggles a pin momentarily high, and records the CPU counter value (of the order of nanoseconds) at the moment this happens. The PIC sends a message by the RS232 port to LabVIEW to tell LabVIEW the time of day that it saw the rising edge. LabVIEW then extrapolates the time from the CPU counter. The major hardware constraints in this approach are, the latency of the LabVIEW code, the latency of the USB bus, and the NI DAQ.

A programmer along with some additional software from *Microchip Technology* are required to program the PIC micro controller. The programming hardware is a *PICkit 3*, which connects to 5 of the *PIC 18F2525*'s pins. To run the programmer, Microchip's *MPLAB IDE* is used. Although, it must be noted, that the 1PPS of the GPS connects directly to the PIC, which does the time-transfer, and enables the PMU for time synchronous measurements.

B. GPS Reciever

A GPS receiver is required to implement this hardware. In this particular implementation a *Germic 18x* integrated GPS receiver and antenna was used. The location data is transmitted via *National Marines Electronics Association's (NMEA) 0183 ASCII interface specification* [12] and uses the RS-232 standard to send the location information to the PIC micro controller. The *Garmin* hardware sends navigational information in *NMEA* format, which contains the crucial timing-information required for PMU operation [12]. It needs to be noted, that the NI-6009 receives the timing information for debugging purposes, which would be useful specially when the substation is in a remote location.

The time transfer comes in the form of a 1 Pulse-Per-Second (1PPS) signal. This signal is sent directly to the PIC micro-controller, which is supposed to generate the 60 Hz pulse train-that will trigger the NI-6009. If the NI-6009 fails to receive the trigger, LabVIEW will get stuck. It is worth noting that, this part of the setup is responsible for enabling the time-transfer, which is crucial in terms of the operation of the PMU. Without the functional time-transfer the PMU will not be able to work.

C. NI USB-6009 OEM

The NI 6009 is a multifunction Input/Output (IO) device that provides DAQ functionalities which are easily accessed through a host computer running different NI software, including LabVIEW. In other words, it acts as a data acquisition board which records the input voltage signals through three analog input pins. It is able to produce 14 bit long digital outputs at a rate of 48 kS/second.The digital I/O pins can be accessed in order to develop a real-time control system development such as the one reported in [25]. Additionally, the *NI 6009*, receives the 1PPS timing signal that can be used for debugging purposes only. The *NI 6009 DAQ* is interfaced with the host computer via a standard USB cable.

D. Power Supplies and Other Hardware

An external power supply is required to complete the construction of the FluxPMU hardware. The power supply has an input of 100-240 V AC supply with a maximum current rating of 1.2 A and a frequency rating of 50/60 Hz. The output is rated at 12 V, 4.5 A. In this particular implementation *MeanWell HDR 60-12 power-supply unit* was used. The wiring connected to it used standard gauge 22 wire.

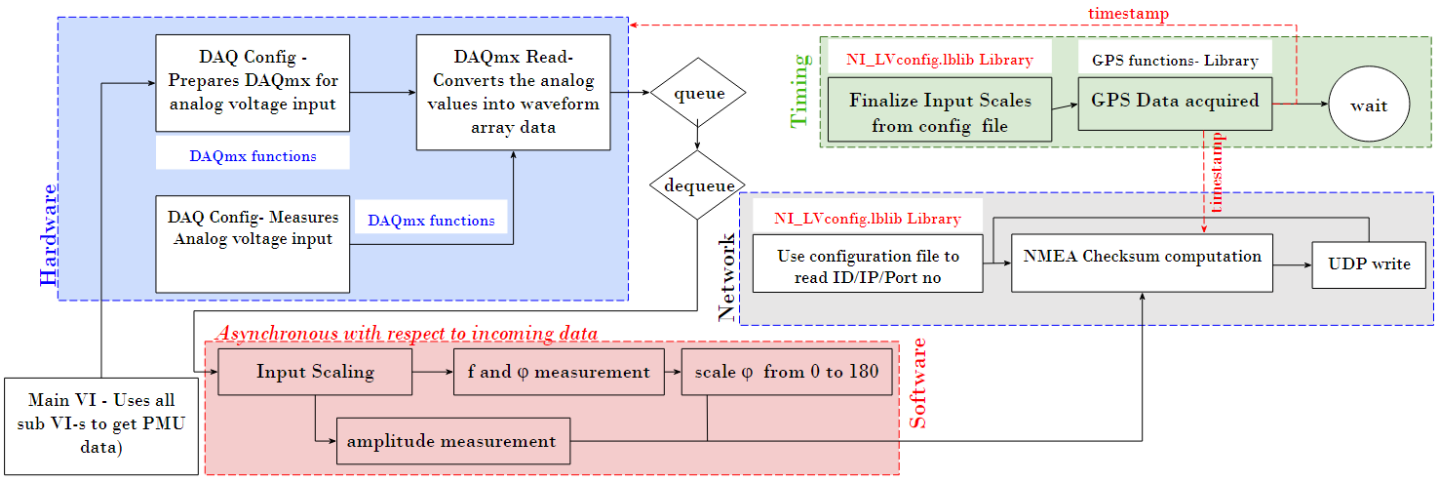

Fig. 2. Algorithmic Flowchart for the FluxPMU

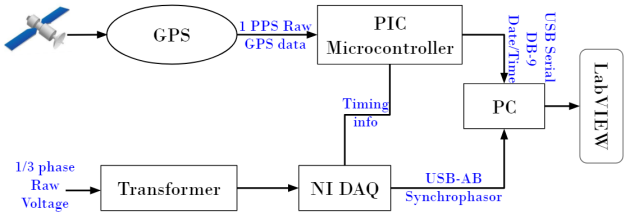

Fig. 3. Simplified system diagram of FluxPMU

Overall Assembly

The steps required to assemble the hardware have been documented in the *Maker's Guide* and are summarized in section V. The simplified flowchart of FluxPMU in Fig. 2 summarizes how different VIs communicate with each other and complete the overall algorithm to compute the PMU data. The operations performed by these VIs can be classified into four different categories. In Fig. 2

- The blue box contains all the data acquisition functionalities,
- The red box includes the software based signal conditioning and scaling functionalities,
- The grey box contains the network functionalities,
- The green box contains the timing functions.

In summary, the PMU system starts up by receiving the raw voltage and GPS time signals simultaneously. The 1PPS received from the GPS is utilized by the PIC micro-controller to generate time-synchronous measurements for the PMU. The data acquisition and analog-to-digital conversion is done by a DAQ board from NI. Once all of the data processing has been complete, the GPS time stamp signal and voltage measurements are formatted as synchrophasors in LabVIEW.

IV. Software

The software to drive FluxPMU is based on the NI *G* programming language of the LabVIEW software. *G* is a graphical programming language, designed for systems engineering and real time applications [10]. Within LabVIEW, a Virtual Instrument (VI) is the basic building block of programs written in the G language. A VI consist of a *front panel* which acts as a graphical user interface, and a *block diagram* that consists of the graphical programming blocks. An example of the front panel for FluxPMU is shown in Fig.5.

Before running the main VI which utilizes all the other subordinate VIs, a few LabVIEW specific drivers including NI DAQmx Driver, NI VISA Driver, and LabVIEW Real Time need to be installed on the system. Once these are resolved, FluxPMU software can be opened in LabVIEW. Once opened, there are several configuration settings that need to be taken care of, as shown through the example from the Maker's Guide.

There are are 18 additional VIs that are used to set up configuration files, performing computations, and displaying, storing and if desired, reporting of PMU data via UDP with a simplified data frame which is based on NMEA. Since, the

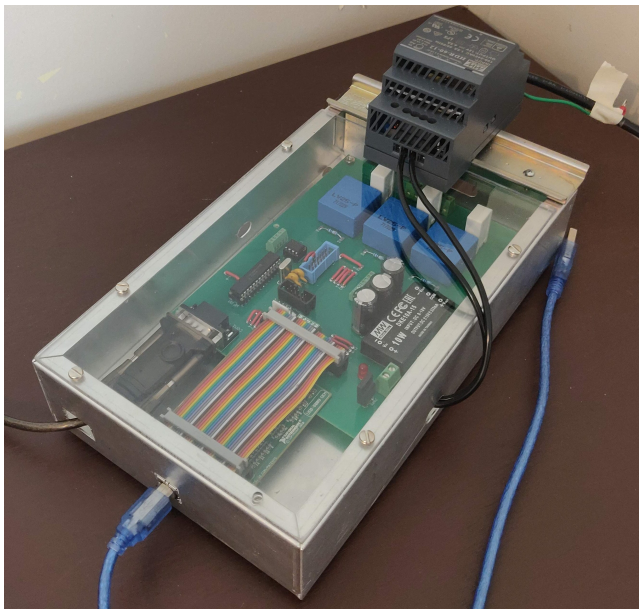

Fig. 4. Hardware prototype of a fully operational FluxPMU.

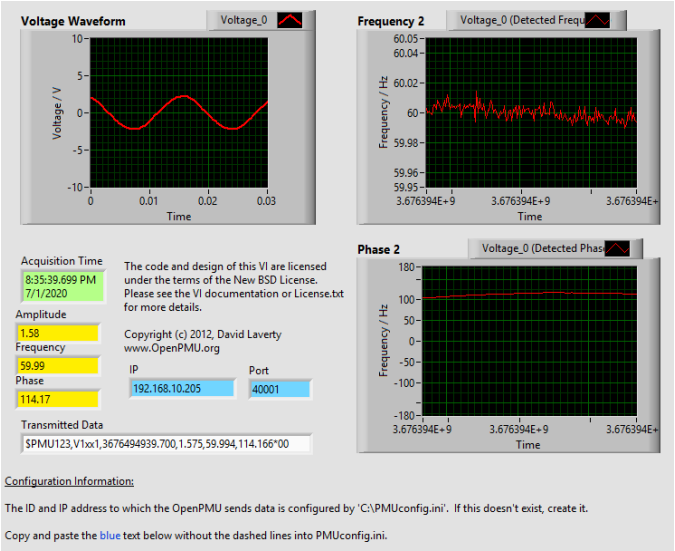

Fig. 5. A fully operational FluxPMU with the front panel for the single phase software with a single phase input

GPS stage uses the same NMEA specifications, it would be easier for the students to work with this data as they are already familiarized with it.

V. Assembly

The process to assemble all of the components in order to run the FluxPMU are detailed throughout the *Maker's Guide*. The following summarizes the main steps of the process:

1) Decide on a version of FluxPMU and order the required parts and make/order the printed circuit boards, i.e. 1ϕ or 3ϕ voltages.
2) Download and install all of the necessary software and become familiar with the code.
3) When parts arrive, complete each individual external hardware component, and attach circuit components to the board.

Once FluxPMU has been completely assembled it should resemble the prototype in Fig. 4. To acquire measurements, the GPS antenna should be placed following the recommendations in [21], the voltage signals should be connected to the board's input, and the USB cable to the computer. Once this is done, after executing the main program, the front panel will look like that of in 5, which shows the input voltage waveform and all computed quantities.

Performance analysis on the base designs of FluxPMU have been conducted under both nominal and dynamic conditions in [26] and [27], respectively. Testing results show that this PMU fulfills the steady state requirements set out in the IEEE Standard C37.118.1-2005 standard, but not all the dynamic tests stipulated in the 2011 standard and the amendments made in 2014 [27]. This can be improved based on the results in [17] and will be subject to future work.

VI. Conclusion

The purpose of the FluxPMU project is to enable students to explore and build an open source, low cost PMU. While there are several other open source, low cost PMUs available for further research, FluxPMU aims to tackle issues students may find when trying to build open source projects. By providing thorough documentation on the build and operation process, FluxPMU offers more to the student experience than other similar open source PMUs. All documentation and software sources required to build the FluxPMU can be found in the following Github repository: `https://github.com/alsetlab/fluxpmu` .

References

[1] "Report on August 2003 Blackout", Department of Electricity [Online] Available: https://www.energy.gov/oe/services/electricity-policy-coordination-and-implementation/august-2003-blackout

[2] Loi Lei Lai, Hao Tian Zhang, Chun Sing Lai, Fang Yuan Xu and S. Mishra, "Investigation on July 2012 Indian blackout," 2013 International Conference on Machine Learning and Cybernetics, Tianjin, 2013, pp. 92-97, doi: 10.1109/ICMLC.2013.6890450.

[3] M. A. Kabir, M. M. H. Sajeeb, M. N. Islam and A. H. Chowdhury, "Frequency transient analysis of countrywide blackout of Bangladesh Power System on 1st November, 2014," 2015 International Conference on Advances in Electrical Engineering (ICAEE), Dhaka, 2015, pp. 267-270, doi: 10.1109/ICAEE.2015.7506847.

[4] W. Mathis, M. Carr, "London Blackout Blamed on Drop in Wind and Natural-Gas Power"[Online] Available: https://www.bloomberg.com/news/articles/2019-08-09/london-blackout-occurred-amid-drop-in-wind-and-natural-gas-power

[5] R. F. Nuqui and A. G. Phadke, "Phasor measurement unit placement techniques for complete and incomplete observability," IEEE Transactions on Power Delivery, vol. 20, no. 4, pp. 2381–2388, 2005.

[6] Wald, M. L. (2013, November 11). The Blackout That Exposed the Flaws in the Grid. Retrieved from https://www.nytimes.com/2013/11/11/booming/the-blackout-that-exposed-the-flaws-in-the-grid.html

[7] "Advancement of Synchrophasor Technology," U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability , Mar-2016. [Accessed: 20-Jul-2020]

[8] Schweitzer Engineering Laboratories, "SEL 421-4, -5 Instruction Manual." 2016.

[9] D. M. Laverty, R. J. Best, P. Brogan, I. Al Khatib, L. Vanfretti and D. J. Morrow, "The OpenPMU Platform for Open-Source Phasor Measurements," in IEEE Transactions on Instrumentation and Measurement, vol. 62, no. 4, pp. 701-709, April 2013, doi: 10.1109/TIM.2013.2240920.

[10] "What is LabVIEW?," NI. [Online]. Available: https://www.ni.com/en-us/shop/labview.html. [Accessed: 31-Jul-2020].

[11] D. Schofield, F. Gonzalez-Longatt and D. Bogdanov, "Design and Implementation of a Low-Cost Phasor Measurement Unit: A Comprehensive Review," 2018 Seventh Balkan Conference on Lighting (BalkanLight), Varna, 2018, pp. 1-6.

[12] Garmin International Inc, "GPS 18x Technical Specifications." Oct-2011.

[13] C. G. C. Carducci, et al, "A Versatile Low-Cost OS-based Phasor Measurement Unit," 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Auckland, New Zealand, 2019, pp. 1-6.

[14] P. Romano and M. Paolone, "Enhanced Interpolated-DFT for Synchrophasor Estimation in FPGAs: Theory, Implementation, and Validation of a PMU Prototype," in IEEE Transactions on Instrumentation and Measurement, vol. 63, no. 12, pp. 2824-2836, Dec. 2014.

[15] D. M. Laverty, et al "OpenPMU: Open source platform for Synchrophasor applications and research," 2011 IEEE Power and Energy Society General Meeting, Detroit, MI, USA, 2011, pp. 1-6.

[16] D. M. Laverty, L. Vanfretti, I. Al Khatib, V. K. Applegreen, R. J. Best and D. J. Morrow, "The OpenPMU Project: Challenges and perspectives," 2013 IEEE Power Energy Society General Meeting, Vancouver, BC, 2013, pp. 1-5, doi: 10.1109/PESMG.2013.6672390.

[17] X. Zhao, D. M. Laverty, A. McKernan, D. J. Morrow, K. McLaughlin and S. Sezer, "GPS-Disciplined Analog-to-Digital Converter for Phasor Measurement Applications," in IEEE Transactions on Instrumentation and Measurement, vol. 66, no. 9, pp. 2349-2357, Sept. 2017.

[18] A. Stadlin, "GridTrak OpenSource PMU (GTosPMU) Sensor." [Online]. Available: https://github.com/ajstadlin/GridTrak. [Accessed: 7-August-2020].

[19] R. Garcia-Valle et al. "DTU PMU laboratory development — Testing and validation," 2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe), Gothenberg, 2010, pp. 1-6,

[20] L. Vanfretti et al., "Estimation of Eastern Denmark's electromechanical modes from ambient phasor measurement data," IEEE PES General Meeting, Providence, RI, 2010, pp. 1-8.

[21] North American SynchoPhasor Initiative, "A Guide for PMU Installation, Commissioning and Maintenance. Part II. PMU INstallation Procedures." June 5, 2007. [Online] Available: https://www.naspi.org/sites/default/files/reference_documents/81.pdf. [Accessed: 7-August-2020.]

[22] North American SynchroPhasor Initiative, Available: https://naspi.org/reference-documents

[23] Colin Chapman et al "Wide Area Measurement System Utilizing Open Source Tools", Proceedings of The National Conference On Undergraduate Research (NCUR) 2013 University Wisconsin La Crosse, La Crosse, WI

[24] "Why choose NI for PMUs and Wide Area Monitoring", [Online], Available: https://www.ni.com/en-us/innovations/white-papers/16/why-choose-ni-for-pmus-and-wide-area-monitoring.html

[25] R. J. Best, et al, "Synchrophasor Broadcast Over Internet Protocol for Distributed Generator Synchronization," in IEEE Transactions on Power Delivery, vol. 25, no. 4, pp. 2835-2841, Oct. 2010, doi: 10.1109/TPWRD.2010.2044666.

[26] D.M. Laverty, R.J. Best, P. Brogan, I. Al Khatib, L. Vanfretti, and D.J. Morrow, "OpenPMU Platform for Open Source Phasor Measurements", IEEE Transactions on Instrumentation and Measurements, vol. 62, no. 4, April 2013, pp. 701 – 709

[27] P. Brogan, D.M. Laverty, Xiaodong Zhao, John Hastings, D.J. Morrow, and L. Vanfretti, "Technique for Pre-Compliance Testing of Phasor Measurement Units" International Journal of Electrical Power Energy Systems, Vol. 99, July 2018, pp. 323 – 330.