OpenPMU: Open Source Platform for Synchrophasor Applications and Research

David M. Laverty, *Member, IEEE*, D. John Morrow, *Member, IEEE,*
Alastair McKinley, *Member, IEEE*, and Michael Cregan, *Member, IEEE*

***Abstract*—Synchrophasor measurement is increasingly becoming a standard tool of the electrical power systems engineer and is attracting much interest in the research community due to the novel new methods and techniques it offers for observing and controlling the power system. A common difficulty in the development of Synchrophasor based technology is the closed environment in which the measurement algorithms and processes governing the operation of Phasor Measurement Units, PMUs, are developed. While most PMUs will support the relevant IEEE standards for Synchrophasor measurement, where transient or dynamic operation is concerned it would be useful to have detailed knowledge of the mathematical transforms and indeed any limitations of the measurement hardware. This can be especially important in developing new protection systems utilizing Synchrophasors, when confidence in the system rests on consistent performance between devices over a long period of service, including device upgrades. The "OpenPMU" described in this paper intends to settle some of these concerns by defining and developing a research orientated platform for Synchrophasor algorithm and application development.**

***Index Terms*—Synchrophasor, Telecommunications, Python, PMU, Smart Grid**

I. Introduction

Synchrophasor measurement is increasingly becoming a standard tool of the electrical power systems engineer. Access to time synchronized measurements, especially phase angle, opens up a suite of new applications and is attracting interest amongst both the research and utility communities alike. It affords the means to improve understanding and modeling of power system phenomena, and can be directly applied in power system protection schemes. Vendors of various measurement devices are beginning to offer Synchrophasor measurement as a secondary function on their devices as a marketing incentive, indicating that the technology is now incredibly cost effective and will become increasingly ubiquitous.

The authors have interests in anti-islanding technologies, otherwise known as Loss-of-Mains detection [1], and in the phase control of islanded generators, dubbed "Synchronous Islanding" [2]. In the development of these technologies, the authors were often frustrated by lack of information regarding the formatting of Phasor Measurement Unit (PMU) data output and the transient performance of PMU devices [3].

In order to address this problem, the authors embarked on an effort to build their own PMU equipment from first principles and generally in accordance to the method proposed by Phadke in [4]. The authors have subsequently found that they are far from an isolated case, indeed many research clusters have pursued similar courses in parallel over the last number of years, e.g. the "DTU PMU" [5]. This lead to the realization that much resource is being spent duplicating efforts, and an online community is needed for the sharing of experience and technology.

One of the barriers of entry to the use of Synchrophasor technology is that most vendors used closed source algorithms that are considered commercially sensitive intellectual property. There is nothing intrinsically wrong with this model, but it is an obstacle for those working outside of the vendor's organization. Without detailed knowledge of the algorithms by which Phasor Measurement Units (PMUs) determine phase angle, it is difficult to stand over research that applies these devices in applications were variations in measurement algorithms can determine the success or failure of a study. For example, real-time control or transient analysis.

The OpenPMU project intends to address this problem by the creation of an open source software (OSS) platform on which the research community, and business, may contribute their expertise and create a unified library of PMU algorithms and functions.

One of the primary considerations is that different research outfits will have different views and objectives as to how to approach the measurement process, especially in terms of hardware configurations and phase estimation algorithms. While not wishing to be Draconian in imposing a standard model for the OpenPMU, the authors present in this paper a "Recommended Minimum" standard for the equipment that should prove affordable and agreeable to most institutions. This can be used as a reference from which the OpenPMU project can evolve.

The authors take a fairly liberal attitude to the development of the software, preferring that new techniques are tried and explored rather than forcing the adoption of prescribed methods. It is hoped that in this way the expert community will naturally tend in the direction of optimal solutions based

Invited Paper submitted for the IEEE PES GM Panel Session on "Open Source Software enabling the Smart Grid"

D. M. Laverty, D. J. Morrow and M. Cregan are with the Electrical Power and Energy Research Cluster, Queen's University Belfast, Ashby Building, Belfast, BT9 5AH, Northern Ireland (e-mail: david@laverty.org.uk; dj.morrow@ee.qub.ac.uk; r.best@qub.ac.uk).

A. McKinley is with the wireless network security startup TOM Ltd. and is an independent software consultant.

on peer feedback and discussion.

This paper will begin by outlining the OpenPMU hardware platform, which is presently centered on standard laboratory hardware from established vendors. It is hoped that this will allow easy adoption of the project at minimal capital outlay. In the subsequent section, the software framework will be introduced. Then follows a timeline for the introduction of functionality to that framework, with an invitation for submissions to the project from the Synchrophasor community.

II. OpenPMU Architecture

The OpenPMU system is split into two distinct parts, the measurement system (analogue data acquisition and time synchronization), and the analysis system (an embedded PC running the analysis algorithms and communications structure).

The separation between the measurement system and the analysis system has been made such that the two components can develop independent from each other. The subsequent sections of this paper will discuss the present day implementation of both components and their future development.

The intention of OpenPMU is to work towards the creation of a fully featured, IEEE C37.118 [6] compliant Phasor Measurement Unit. While this is the ultimate goal, initially development will concentrate on creating an open measurement system and an open platform for development of phase estimation algorithms. Parallel to algorithm development on this platform will be components addressing some of the requirements of IEE C37.118, such as the messaging format described in the standard, and other additional functionality such as a Web Administration interface.

III. Measurement System

At present, the OpenPMU measurement system uses standard laboratory equipment from established vendors. Much consideration has been given to the creation of a custom hardware solution to address the specific needs of Synchrophasor estimation, however at this stage in the project the authors are of the view that it is preferable to maintain accessibility to the project by using equipment that is readily available. A future architecture for the measurement system will be discussed later.

A. Present (Proprietary) Measurement System

The present day measurement system uses a standard Data Acquisition device (DAQ) from National Instruments and a GPS time source receiver from Garmin. National Instruments and Garmin are perhaps some of the more expensive manufacturers in their fields, but both manufacturers provide excellent support resources and documentation for their equipment. The major downside in using proprietary hardware such as this in an open source project is that the open source components of the hardware become dependent on the availability of proprietary drives. The next section of the paper will discuss the development of an open measurement system.

The GPS time source receiver is connected to the DAQ via a microcontroller, a PIC from Microchip, which runs firmware of the authors' design. The connection of these devices is described in Fig. 1. The Garmin GPS Engine provides the time synchronization required for Synchrophasor estimation. As is typical with most GPS engines the outputs of the Garmin unit are a TTL level "One-Pulse-Per-Second" (1PPS) and RS232 level NMEA sentences containing the UTC date and time.

The 1PPS signal, a 1 Hz square wave which rises on the transition of the UTC second, is input directly to a PIC microcontroller which is running an Embedded C program written by the authors. The microcontroller firmware describes the function of a Phase-Locked-Loop (PLL) with a 50x multiplier which produces a 50 Hz square wave in phase with the UTC second transition (60x multiplier in 60 Hz countries) from the GPS receiver. The 50 Hz square wave is used as a windowing trigger to chop the signals under analysis into 20 ms segments.

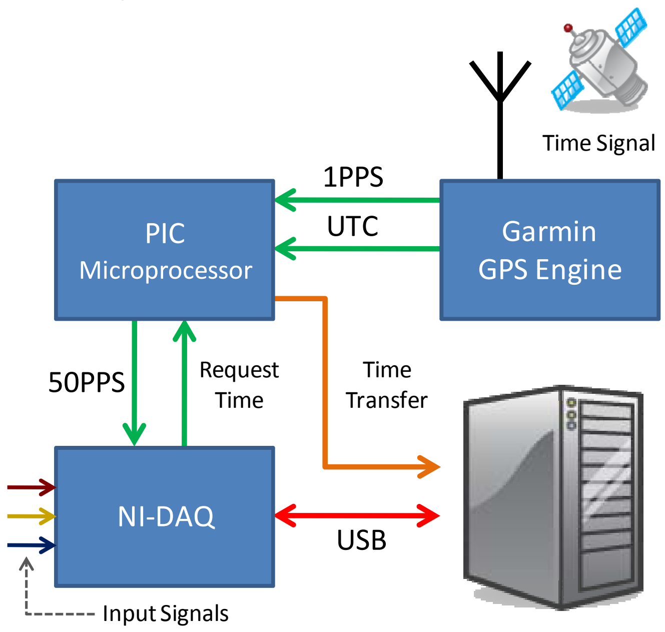

Fig. 1. OpenPMU measurement system schematic overview.

The inputs to the DAQ are the signals under analysis (3-phase voltages and currents) and the sampling window trigger. Depending on the nature of the DAQ device used, the sampling trigger may be connected to the start sampling trigger, but for a more general solution it is preferred to capture this signal as an analogue waveform and adjust the windowing by digital processing.

An example of the type of data captured by the DAQ is shown in Fig. 2. The green square wave is the output of the GPS disciplined PLL, the windowing function trigger. It is this edge which the phase estimation algorithm uses as a reference for zero degrees. By the cosine convention for representing Synchrophasors, Fig. 3, the red phase is just approaching 0º while the yellow phase has just passed 90º.

The waveform shown in Fig. 2 is just a section of a continuous data stream which is stored in a buffer and digitally re-windowed and time stamped using the time transferred from the GPS receiver. This new time-aligned data stream can be processed by the phase estimation algorithms.

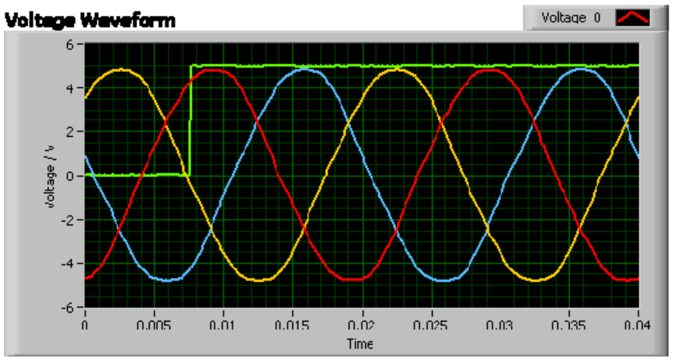

Fig. 2. Typical waveform captured by NI-DAQ (green = GPS reference).

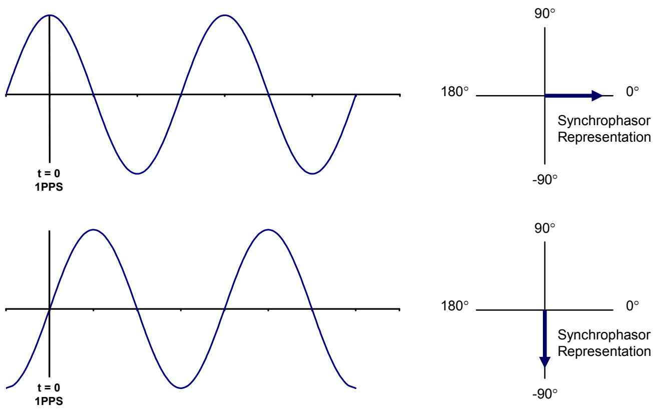

Fig. 3. Synchrophasor representation (cosine convention) [6].

The OpenPMU analysis algorithms, discussed later, run on a standard x86 / x64 personal computer. The PIC microprocessor also provides a mechanism for time transfer from the GPS receiver to the analysis PC. Via the DAQ card, the PC can actuate a TTL trigger named "Request Time". The PC records the value of its internal clock (typically the CPU counter) at the same moment that the "Request Time" signal is actuated. The PIC will asynchronously communicate to the PC the UTC time at which it recognized the time transfer request. The PC can then correct the time indicated by its internal clock to UTC and extrapolate UTC from the internal clock until the next time transfer event is required. The drift in the PC clock is sufficiently low to allow a period of several minutes between time transfer events.

By defining a specification for the representation of the time aligned data it is possible at this point to decouple the phase estimation algorithms from the specific hardware of the measurement system. This allows an easy transition to an open source measurement system which shall be described next.

B. Open Source Measurement System

As alluded to in the previous section, one of the major limitations of using proprietary hardware in an open source project such as this is that the project becomes dependent on the availability of proprietary drivers from the hardware vendors. While National Instruments presently has excellent support for the Microsoft Windows family of operating systems, support on the Linux platform is less mature. This is not a criticism of National Instruments, rather it is just a common scenario that arises when a vendor's business plan seems not to meet the needs of a specific long term project such as this.

An open source measurement system should utilize hardware that is commonly available, and firmware that is in a language that is easy to programme in and easy to obtain development tools for. There are a huge range of options to choose from, however to ensure there is sufficient processing power and memory I/O we should limit our selection to 32 bit microcontrollers, these including PIC, PSoC, Freescale, ST Micro and many more besides. More recently there has been a significant development in 32 bit microcontrollers that utilize the .NET Micro Framework for example, Netduino, ChipworkX, EMX, USBizi, FEZ. These offer significant lower initial costs and free development software which makes them particularly attractive to educational establishments and hobbyists. It is difficult to make a choice when the options are so similar, however the authors propose that "Netduino plus" [7] be adopted as the platform for the open source measurement system.

The advantages of Netduino plus are:

- Low cost of hardware
- Free development tools
- Familiar .NET programming language
- Powerful library of functions available
- Simpler than PIC / Embedded C
- Ethernet interface
- Flash SD memory card reader

Full technical specifications are available online. The Netduino hardware designs are available under an open source license and can be purchased from starting prices of circa $50.

C. Hardware Licensing

Although at first it seems a little unusual, hardware can be licensed in a similar manner to open source software. The problem arises that open hardware licenses protect the design documentation and instructions, but not the actual hardware itself. Hardware rights are traditionally protected by patents, and as yet the Open Source community has not found an effective way of replacing patents [8]. The author of [8] even goes as far as to say there are "no good open hardware licenses to date". Of the Open Hardware licenses available, the most complete would seem to be the TAPR Open Hardware License (OHL) [9]. This offers protection on the design rights, allowing products to be made from the designs so long as the designs are distributed with the product. This would seem to fit with the philosophy of the OpenPMU project.

Fortunately the firmware running in the proposed open source measurement system is an entirely different matter, and may be licensed under any of the OSS licenses available. Presently the firmware for the existing PIC microcontroller is available under the BSD license [10], whether the new firmware is release under BSD, GPL [11] or otherwise is a matter for discussion.

IV. Analysis Algorithms and Communications Software Platform

As mentioned in an earlier section, the OpenPMU analysis algorithms run on a standard x86 / x64 personal computer. The authors would rather that the OpenPMU software be operating system neutral, since there are cases for and against both Windows and Linux depending on the application. While Windows is perhaps the platform of choice were a familiar desktop user interface is desired, Linux is by far the preferred choice when considering deploying software in an embedded capacity.

The dependence on National Instruments hardware effectively limits OpenPMU to Microsoft Windows at the present time, but with the plan to migrate to an open measurement system, it is logical to keep the analysis software platform operating system independent. This is largely determined by the choice of programming language.

A. Language

Choosing a programming language for a large project such as OpenPMU is often a difficult decision. There are many factors to consider, not lease the authors' own experience and background in programming, what the project intends to achieve, and the accessibility to the language by other contributors. In this case, the decision has been made somewhat easier since one of the authors has expensive experiences in 'Python' and the remaining authors have experienced that it is an accessible language to understand and programme for.

Python is an open source, high level, general purpose programming language. The design of the language ensures that Python code is highly readable and is thus easily maintainable in collaborative projects. Python contains many high level data structures, such as native data types for lists, maps and sets. It is also dynamically typed, which significantly reduces the amount of code required. These features mean that Python developers can be more productive than traditional languages (e.g. C/C++/Java) as they are able focus more on algorithms and problem solving instead of writing large amounts of 'boiler plate' code.

The popularity and uptake of Python has increased rapidly over the last decade and boasts some very high profile commercial users, such as Google, who make heavy use of Python in their web based products, and employ the creator of Python. The recent increase in Python usage is also due to the increasingly large array of high quality third party libraries. Python is a truly general purpose language, with applications in scripting, web programming, desktop applications and scientific computing.

In the scientific computing domain, Python is especially useful due to the availability of the powerful NumPy and SciPy libraries, which provide users with a high level interface for dealing with scientific data and algorithms, without major sacrifices in execution speed. The combination of Python, NumPy and SciPy provides a fully open source, cross-platform and highly productive development environment for OpenPMU.

B. Modules

The OpenPMU software will need to handle a number of functions, including accessing inputs from the measurement hardware, analyzing the data by Digital Signal Processing (DSP) and communicating this data to the outside world via some form of telecommunications. An outline specification for each of these three components will now be described.

1) Measurement Hardware Interface

The measurement hardware is presently a USB device from National Instruments and is thus tied to National Instruments device drivers, available for Windows and certain distributions of Linux. It is proposed to migrate the measurement hardware to an open source platform and interface to the analysis computer via IP messaging over Ethernet. The Measurement Hardware Interface component of the analysis platform is required to take care of the hardware in whatever form it comes and provide to the digital signal processing block a consistent representation of the analogue waveforms and time signal acquired by the measurement system.

2) Digital Signal Processing (DSP)

The DSP stage might be considered the main component of the PMU software since this is the component that actually makes the raw analogue data and converts it into a meaningful Synchrophasor, featuring phase, frequency, amplitude and time information about the waveforms under observation.

To achieve this, the DSP must window the analogue data according to the time signal to insure that synchronization with the GPS time signal is maintained. The length of the window will determine the accuracy, responsiveness and delay of the measurement. A short window will provide a responsive measurement with little delay and low accuracy, a longer window will improve accuracy but at the expense that changes in the input take longer to be reflected in the output.

The actual phase estimation algorithm might take one of many forms, for example Newtown-type algorithm [12], Weighted-Least-Squares / Adaptive FIR [13] or dynamic types that vary the measurement algorithm depending on input conditions [14]. For the time being, the authors' have decided that while the 'boiler plate' of OpenPMU is being finalized that a simple FFT / Spectral Leakage type algorithm will suffice. At present, the incoming data is split into nominally two 50 Hz cycles (40 ms) segments, and the "Blackman-Harris" windowing function is applied before processing by an FFT. A spectral leakage function determines the nominal frequency, which is subsequently used to generate a reference from which the phase of the measured signal can be estimated. Individual per-phase estimations can be combined to produce sequence component representations.

This is clearly a very basic phase estimation technique and far from ideal, but is included at this early stage as a useful means of testing the functionality of the other components of the systems.

3) *Communications*

The communications section has two responsibilities. Firstly, the communications need to be formatted such that they are compliant with industry standards, such as IEEE C37.118 or IEC 61850. Secondly, it must manage the communications with the outside world. This latter section includes transmission of the Synchrophasor packets to their destination, and also remote access to the PMU for configuration (web based configuration being preferred). The communications section will also manage the local recording of data, since this is itself a data representation exercise.

C. *Software License*

The OpenPMU analysis and communications software is presently available under the BSD license. Where this is modified to use the GPL license is a matter for discussion. The BSD license allows for greater flexibility in terms of producing products from the software, so may be of greater interest to small business, but GPL promotes community development by requiring modifications to be returned to the community. BSD may be converted to GPL, but it is difficult to covert GPL licensed code to a BSD license. The authors very much welcome feedback and discussion on this point.

V. Community Involvement

To get involved with the OpenPMU project and to communicate with the authors of this paper, a number of options are available. A SourceForge.net project page has been created and maintained for the OpenPMU project, but better communications are available via the project's Facebook and Google Group accounts. These are all accessible via the Project domain, www.OpenPMU.org.

The authors particularly welcome suggestions for improving the project and partners interested in contributing to the project. Please do not hesitate to make contact if you have any questions regarding taking part in OpenPMU or building your own OpenPMU.

VI. References

[1] Laverty, D.M.; Morrow, D.J.; Littler, T.; Crossley, P.A., "Loss-of-Mains Detection by Internet Based ROCOF," *Developments in Power System Protection, 2008. DPSP 2008. IET 9th International Conference on*, pp.263 268, 17-20 March 2008

[2] Best, R.J.; Morrow, D.J.; Laverty, D.M.; Crossley, P.A.; , "Synchrophasor Broadcast Over Internet Protocol for Distributed Generator Synchronization," *Power Delivery, IEEE Transactions on* , vol.25, no.4, pp.2835-2841, Oct. 2010

[3] Laverty, D.M.; Morrow, D.J.; Best, R.; Crossley, P.A.; , "Performance of phasor measurement units for wide area real-time control," *Power & Energy Society General Meeting, 2009. PES '09. IEEE* , vol., no., pp.1-5, 26-30 July 2009

[4] Phadke, A.G.; Thorp, J.S., "History and Applications of Phasor Measurements," *Power Systems Conference and Exposition, 2006. PSCE '06. 2006 IEEE PES*, pp.331-335, 2006

[5] Design, installation and application of Phase Measuring Units (PMUs), DTU, 2009
Internet: http://power.inescporto.pt/EES-UETP/DTU_PMU_Course.pdf

[6] "IEEE Standard for Synchrophasors for Power Systems,"
IEEE Std C37.118 2005 (Revision of IEEE Std 1344-1995), 2006

[7] Netduino Plus Technical Specifications, Secret Labs LLC,
Internet: http://www.netduino.com/netduinoplus/specs.htm

[8] Open Hardware Licenses, The Foundation for P2P Alternatives
Internet: http://p2pfoundation.net/Open_Hardware_Licenses

[9] The TAPR Open Hardware License,
Internet: http://www.tapr.org/ohl.html

[10] The Open Source Initiative OSI – The BSD License
Internet: http://www.opensource.org/licenses/bsd-license.php

[11] GNU General Public License
Internet: http://www.opensource.org/licenses/bsd-license.php

[12] Terzija, V.; Stanojevic, V., "Power quality indicators estimation using robust Newton-type algorithm," *Generation, Transmission and Distribution, IEE Proceedings-* , vol.151, no.4, pp. 477-485, 11 July 2004

[13] Kusljevic, M.D.; Tomic, J.J.; Jovanovic, L.D.; , "Frequency Estimation of Three-Phase Power System Using Weighted-Least-Square Algorithm and Adaptive FIR Filtering," *Instrumentation and Measurement, IEEE Transactions on* , vol.59, no.2, pp.322-329, Feb. 2010

[14] Warichet, J.; Sezi, T.; Maun, J.C.: "A Synchrophasor Measurement Algorithm Suitable for Dynamic Applications," *16th Power Systems Computation Conference,* PSCC 2008

VII. Biographies

David M. Laverty (S'07) was born in Belfast, Northern Ireland, in 1984. He received the M.Eng and Ph.D. degrees from Queen's University Belfast, Belfast, UK, in 2006 and 2010 respectively.

Since graduating he has been with the Electrical Power and Energy Research Cluster at Queen's University Belfast, Belfast, UK. His placements as an undergraduate were with Northern Ireland Electricity and Universität Paderborn, Germany. His current research interests are in power system measurements, anti-islanding detection, phasor measurements, and Smart Grid telecommunications, messaging and security.

Dr. Laverty is a member of the IEEE and a member of and volunteer with the Institution of Engineering and Technology (IET).

D. John Morrow (M’99) was born in Dungannon, Northern Ireland, in 1959. He received the B.Sc and Ph.D. degrees from Queen’s University Belfast, Belfast, U.K., in 1982 and 1987, respectively.
Since 1987, he has been a Lecturer in electrical engineering at Queen’s University Belfast, Belfast, UK with research and consulting interests in electric power systems, power system instrumentation, and gen-set controllers.
Dr. Morrow is a member of the Institute of Engineering and Technology and also a member of the IEEE PES Excitation Systems Subcommittee working group since 1999.

Alastair McKinley was born in Belfast, Northern Ireland in 1981. He received the MEng and PhD degrees from Queens University Belfast in 2005 and 2009 respectively. Since graduating he has worked on University research projects in wireless network security and distributed haptic virtual environments. He is currently working as a director of a local technology startup in wireless network security (TOM Ltd.) and also as an independent software consultant.

Michael Cregan was born in Portadown, Northern Ireland, in 1967. He graduated from The University of Ulster in 1990 with an Honours Degree in Engineering and from Queeen's University Belfast with an M.Sc and a Ph.D.
He is currently working at Queen’s University Belfast. Prior to this he has worked for Northern Ireland Electricity and General Controls and Automation.