A Modular Phasor Measurement Unit Design featuring Open Data Exchange Methods

A Modular Phasor Measurement Unit Design featuring Open Data Exchange Methods

David M. Laverty, *Member, IEEE*, John Hastings, *Student Member, IEEE,*
D. John Morrow, *Member, IEEE,* R.Khan, K. Mclaughlin, *Member, IEEE,* S. Sezer, *Member, IEEE*

***Abstract*—Synchrophasor measurements continue to show great potential for innovative protection, control and analytical functions in the emerging Smart Grid. Commercially available Phasor Measurement Units (PMU) are generally produced and sold in a manner that makes their internal workings unavailable, or closed, to the end user. With this in mind, the authors have pursued the goal of producing an open source PMU, the OpenPMU.**

This paper describes how the functions of the OpenPMU have been separated into distinct modules. This is a significant change in design, and has been done so with the intention of reducing the effort required for persons with interest in PMU technology to adopt OpenPMU components in their designs.

Using the design described in this paper, in particular the open data exchange schema, it is possible to interchange modules with ease. For example, one may change the phasor estimation algorithm, without affecting signal acquisition or communication. The modular design also simplifies the use of OpenPMU components in popular simulation environments.

***Index Terms*—Synchrophasor, Open Source, Smart Grid, PMU**

I. Introduction

Phasor Measurement Unit (PMU) technology has become a popular tool across the power systems community, particularly in recent years when lowering costs and the availability of telecoms bandwidth has made the widespread use of PMU feasible. PMU are considered in many novel technologies, as well as for use in enhancing current practices, where measurement of amplitude, phase angle and frequency of electrical quantities (i.e. a phasor) is required. It is common to see PMUs proposed in real-time monitoring, protection and control.

The problem arises that by strict definition a phasor is only valid when a system is stationary. That is, the frequency of the system is unchanging, and the values which the phasor represent could be said to continue indefinitely. In the event that the system frequency is changing, the voltages and currents will be non-cyclical, and the definition of a phasor becomes more challenging. This is well understood [1], [2] and has been addressed in IEEE C37.118.1-2014 [3] which discusses permissible error limits for phasor estimation under dynamic conditions.

For novel real-time protection and control systems, it is understandable that the phasors that are of most importance are whenever the system is non-stationary, as this is when the protection or control function will be expected to take action. Consequently there is a need to understand how the phasors are estimated from the waveforms that they represent.

Many commercial phasor measurement units are developed to a 'black box' philosophy in that the process of signal acquisition, numerical processing and phase estimation are left unknown to the end user. Sometimes this is due to the vendor wishing to protect their intellectual property. Whatever the reason, this leaves researchers using such PMU devices without the necessary information to be confident in their data. Such vendors will usually cite compliance with the IEEE standards. Although the IEEE standard provides a good benchmark for dynamic compliance, it cannot be expected to standardize the response of a PMU under every dynamic or transient event. Thus the authors have been motivated to create an open source phasor measurement platform, named OpenPMU.

The original OpenPMU was released in 2009 [4] and was focused on producing a low-cost PMU with accessible hardware and a flexible phase estimation system that could be optimized to particular real-time control requirements [2]. Since this time it has become apparent that a broad range of expertise is required to fully develop PMU technology, and thus intentionally separating the functions of the PMU out into discrete modules will allow domain experts to focus their skills on a productive area. That is, the module approach described in this paper will allow an expert in phase estimation algorithms to readily apply their work in a practical PMU, without requiring knowledge of either ADC or telecoms technology. Likewise an expert in signal acquisition does not require detailed knowledge of advanced numerical analysis techniques. Each person can make contributions in their area and benefit from the overall community. Such is the spirit of many open source projects.

II. Overview of PMU Modules

A PMU can be broken down into distinct functions, with the successful operation of the device contingent on the correct performance of each function. Informed by the

D. M. Laverty, J. Hastings, D. J. Morrow, R. Khan, K. McLaughlin and S. Sezer are with the School of EEECS, Queen's University Belfast, Ashby Building, Belfast, BT9 5AH, Northern Ireland (e-mail: {david.laverty; jhastings02; dj.morrow; rafiullah.khan; kieran.mclaughlin; s.sezer} @qub.ac.uk). This work has been supported by EPSRC CAPRICA and SuppleHeat projects.

generic PMU described in [1], the authors describe the three core functions of a PMU as "Signal Acquisition", "Phasor Estimation" and "Telecommunications", with the data flows between each stage depicted in Fig. 1.

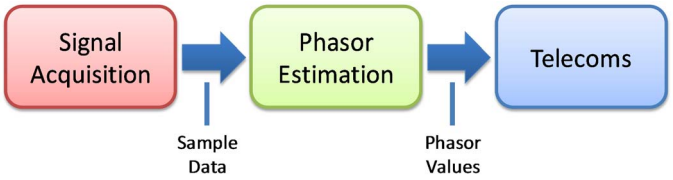

Fig. 1. Overview of the modular design concept.

The signal acquisition block contains within it the analogue electronics for conditioning and filtering of signals prior to analogue-to-digital conversion, the ADC itself, and the time source receiver for synchronizing the PMU to a global clock. Typically a GPS receiver is employed.

The phasor estimation block is a software function which receives time-coded sample data from the signal acquisition block and performs the numerical functions to arrive at a phasor representation of the acquired waveforms. Many different approaches are available to achieve this end. Common techniques are variants on curve fitting using a least-squares type algorithm.

The telecommunications block receives from the phasor estimation block the values which represent the phasors estimated from the acquired waveforms, including the amplitude, phase angle, frequency and rate-of-change-of-frequency. The telecoms block will format this data to an appropriate schema, such as IEEE C37.118.2 [5] or IEC 61850-9-50 [6], and handle the functions required to start/stop streaming and send configuration data. The telecoms block will also contain the functions required to deliver the security mechanisms described for IEC 61850 environments [7]. It will also allow for communication protocols to enable reliable and resilient communication over wide area networks [8].

Note that the descriptions of the modules above are high level discussions of the functions contained within each block and are not intended to be definitive or prescriptive. The objective is to provide a reference model for PMU similar to that which the OSI 7-layer Model [9] provides for telecommunications.

III. Data Exchange Between Modules

It is desirable that the modules described above are able to exchange data in an interoperable way, such that one module may be substituted for another module of the same type. For example, a signal acquisition module that employs Vendor A's ADC and microcontroller should be interchangeable with Vendor B's ADC and microcontroller. To achieve this will require an open data exchange model.

The authors propose that UDP/IP is used as the transport mechanism between modules, with XML markup [10] employed to format the data. The rationale for this approach is that the signal acquisition stage will employ an ADC and microcontroller combination. Many modern low cost microcontrollers and development boards now feature an Ethernet interface as standard, for example Beaglebone [11], Raspberry Pi [12], Arduino [13], amongst others. This interface provides sufficient bandwidth for streaming sample data, and enables easy connection of the signal acquisition stage to desktop/laptop computers or other embedded environments. Whilst other interface methods, for example SPI or I2C, may be preferable for a production device, Ethernet offers great flexibility for research and development activities and would still present a viable route to implementation for a commercial product. The use of XML markup allows data to be exchanged in messages containing markup that is human readable.

It is not necessary to be concerned with bandwidth efficiency as the XML messages are intended to operate by direct connection (cross-over cable), over a local area network (LAN), or by local-loopback address within an operating system. The datagrams described in the following sections are not intended to traverse wide area networks or networks were bandwidth is restricted. Interfaces of 100 Mbps (100BaseT) will usually provide ample overhead.

A. *Signal Acquisition Output*

The signal acquisition module will output the sample data acquired by the ADC, along with a time code indicating the time at which the data was taken (that is, the time the first sample was taken), the sampling rate and other metadata associated related to the origin of the data. This will include the waveform type (voltage/current), transducer information (ratio, optionally type), location and name. The data is arranged hierarchically by channel on the ADC and the raw sample data is contained in a 'payload' field.

1) Base64 Encoding

The main challenge to this approach is that waveform sample data from the signal acquisition stage would be acquired in binary form and would need to be translated to a character set that would allow for a valid XML datagram. This is achieved by translating the binary sample data to an ASCII character set through the Base64 encoding scheme [14]. Consider the following example, in which an arbitrary sinusoid has been digitized using 8-bits of resolution, Fig. 2. The sample values are represented in decimal, binary and hexadecimal in Table 1.

The complete sinusoidal waveform can be represented in hexadecimal form as **00 5A 7F 5A 00 A6 81 A6 00**. This is done by converting groups of 4-bits (nibbles) into the ASCII characters 0-9 and A-F. This is a relatively inefficient use of the ASCII character space available in an XML datagram. The Base64 encoding scheme instead converts groups of 6-bits into a 64 character set of ASCII characters, and yields the expression **AFp/WgCmgaYA** for the sinusoidal waveform of Fig. 2.

In practice waveforms are likely to be sampled at rates of up to 256 samples per cycle, at resolutions up to 16-bits. For one cycle, this yields a data rate of 4096 bits per cycle (512 bytes/cycle), or 683 Base64 encoded characters. This method

is not designed for efficiency; rather it maintains the human readability of the overall datagram, and the raw data is easily recovered by application of a Base64 decode function. An 8-channel ADC employing this method will produce datagrams at a datarate of ~3 Mbps.

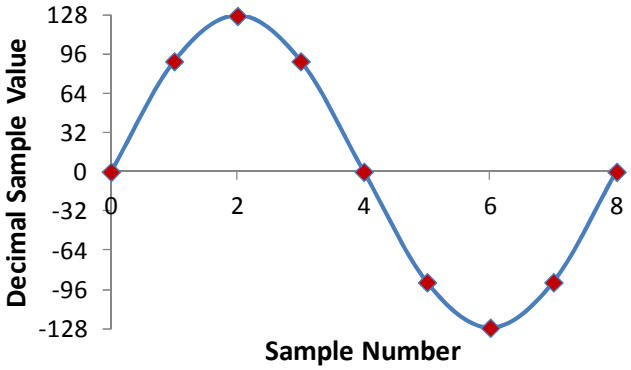

Fig. 2. Digitization of an arbitrary sinusoid.

TABLE I
SUMMARY OF BANDWIDTH TEST (SUBCLAUSE 5.5.6)

Sample No.	Sample Value Dec. (Base10)	Binary (Base2)	Hex. (Base16)
0	0	0000 0000	0x00
1	90	0101 1010	0x5A
2	127	0111 1111	0x7F
3	90	0111 1111	0x5A
4	0	0000 0000	0x00
5	-90	1010 0110	0xA6
6	-127	1000 0001	0x81
7	-90	1010 0110	0xA6
8	0	0000 0000	0x00

2) Required Fields

Each datagram from the signal acquisition module will be constructed in a hierarchical structure. The datagram will feature at the top level the date and time at which the sampled values were taken, the sampling rate, number of samples, and bits per sample. A frame number will allow for sequence checking and identification of missing datagrams.

The top level will identify the number of channels in the datagram. Within this field each channel will be individually identified, and be given fields for channel name, type, phase, and a scalar value indicating the nominal value. The payload of samples is included at this level.

The channel field repeats until all channels have been communicated. The datagram will then close. An example datagram is depicted in Fig. 3. Additional fields can be added where the end user desires, provided the specification described is followed.

B. Phasor Estimation Output

The phasor estimation module accepts as input the time coded sample data from the signal acquisition block, and processes this data to arrive at a phasor representation of the applied waveforms. By applying the time code provided along with the sample data to the resulting phasor data, the phasor estimation module is capable of producing a synchrophasor. Note that a real-time environment is not required for this function, so commodity computing components can be employed. The authors typically employ Raspberry Pi 2/3 development boards for this function. The phase estimation block will echo all of the metadata from the signal acquisition block so that the origin of the synchrophasor data is understood.

1) Required Fields

The output of the phase estimation block will contain the data and time for which the estimated phasor values are valid, thus providing a time synchronized phasor, or synchrophasor. It will provide a frame number so that the sequence of datagrams can be determined, and missing datagrams identified. At the top level, the nominal frequency used in the phasor estimation algorithm will be identified, along with the algorithm name, type or version number.

Per channel, the datagram will list the phasor parameters of magnitude, angle, frequency and rate-of-change-of-frequency (ROCOF). Metadata from the signal acquisition module, for example the channel names, should be copied over to the phasor estimation module output.

An example datagram is shown in Fig. 4. As with the signal acquisition module, additional fields may be added as the end user requires, as long as this minimum specification is adhered to.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<DATA>
    <Date>2015-11-16</Date>
    <Time>10:06:08.010</Time>
    <Frame>1</Frame>
    <Fs>12800</Fs>
    <n>128</n>
    <bits>16</bits>
    <Channels>8</Channels>
        <Channel_0>
            <Name>PHASE_Va</Name>
            <Type>V</Type>
            <Phase>a</Phase>
            <Range>275</Range>
            <Payload>
            UWdaVzVqYjJSbE1HSnBibUZ5ZVNCa11YUmhJR0o1SUhSeVpX
            RjBhVzVuSUdsME1HNTFiV1Z5YVdOaGJHeDVJR0Z1WkNCMGNt
            RnVjMnhoZEdsdXhoZEdsdVp5QnBkQ0JwYm5Sdk1HRWdZbUZ6
            WlNBMk5DQnlaWEJ5WlhObGJuUmhkR2x2Ymk0Z1ZHaGxJRUpo
            YzJVMk5DQg==
            </Payload>
        </Channel_0>
        .
        .
        .
</DATA>
```

Fig. 3. Example datagram originating from the signal acquisition module showing required fields.

IV. ADDITIONAL FUNCTION MODULES

The descriptions in Section III describe the core functions required to deliver a PMU. Many PMUs go beyond these core functional requirements and offer additional features such as voltage/power quality estimation and transient disturbance fault recorder (DFR) functionality. It is possible to deliver these functions using the data exchange model described. Essentially these functions will be add-on modules which

acquire the sample data in the same way as the phase estimation module, but will then interface with the outside world by their own means. To make adoption of such add-on modules as seamless as possible, in addition to sending data to the address/port described in Section IV, the signal acquisition module can make its data available to a multicast address. If desired, a phase estimation module with the same functionality can be used to make the synchrophasor data available.

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenPMU>
    <Format>Phasors</Format>
    <Date>2016-11-04</Date>
    <Time>13:14:23.120</Time>
    <Frame>1</Frame>
    <Algorithm>LSE V1.0 by The Author</Algorithm>
    <Channels>3</Channels>
    <Channel_0>
        <Name>Phase_Va</Name>
        <Type>V</Type>
        <Phase>a</Phase>
        <Range>275</Range>
        <Mag>238.95063</Mag>
        <Angle>18.4037490493</Angle>
        <Freq>50.6909118201</Freq>
        <ROCOF>0</ROCOF>
    </Channel_0>
    .
    .
    .
</OpenPMU>
```

Fig. 4. Example datagram originating from the phase estimation module showing required fields.

V. Use with Simulation Environments

The XML schema described makes it possible to use OpenPMU modules with popular power system simulation environments. Sampled data values may be exported from the simulation environment, processed by a phasor estimation algorithm, and the synchrophasors returned to the simulation using the XML schema and UDP/IP datagrams. Similarly, physical signals can be acquired and fed to the simulation environment, enabling Hardware-in-the-Loop simulations.

VI. Multicast Between Modules

The descriptions in Section III describe the core functions required to deliver a PMU. Many PMUs go beyond these core functional requirements and offer additional features such as voltage/power quality estimation and transient disturbance fault recorder (DFR) functionality. It is possible to deliver these functions using the data exchange model described. Essentially these functions will be add-on modules which acquire the sample data in the same way as the phase estimation module, but will then interface with the outside world by their own means. To make adoption of such add-on modules as seamless as possible, in addition to sending data to the address/port described in Section IV, the signal acquisition module can make its data available to a multicast address.

IP multicast is a UDP-based one-to-many function of the standard TCP/IP protocol (as opposed to one-to-one, or one-to-all of broadcast, see Fig. 5). IP multicast allow for publish-subscribe type operations to be achieved, usually over a LAN environment. In this case, this can mean between modules within the standard OpenPMU setup, or to/from additional function modules added on later.

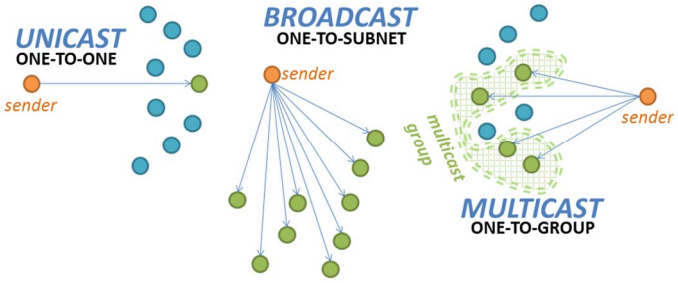

Fig. 5. Visualization of differences between TCP/IP Unicast, Broadcast and Multicast sending mechanisms.

IP multicast utilizes designated section of the IP address space (224.0.0.0/4 for IPv4, and ff00::/8 for IPv6). Essentially these reserved sections of IP address space are used to setup multicast channels.

These channels can be joined by any connected network interface within the scope of the group. The mechanism is simple, and to 'join' a group, a host is configured to listen for datagrams destined for a specific multicast address. Likewise, to publish to the multicast group, a host simply publishes to the group's multicast IP address.

Multicasting is used for services such as routing protocol management, for streaming services, and for real-time data transport, such as sensor data within a substation; IEC 61850 [6] for example, has multicast layers to transport streaming sensor data as well as periodic event data, or fault information to other 'interested' devices within the substation network.

The concept of inter-module common data channels is highly supportive of an extensive modular design, particularly if inter-module message passing follows the designated XML data representation as described in Section III. In addition, with defined multicast channels for certain data set as standard within the OpenPMU implementation, this makes adoption and development by other contributors much more feasible.

However, this communication is not suitable beyond the Local Area Network (LAN) environment, as multicast traffic is not routable without advanced configuration.

VII. Wide Area Communication

Communication between modules takes place over local connections and is thus not limited by bandwidth restrictions. This is not so for wide area communication. Consideration has to be given to the available resources between the data source and the data sink, i.e. all the devices/systems interested in receiving the data in real-time. These data sinks can be databases to store Synchrophasor data for later analysis, they may be near-time monitoring systems that display system data to operators in a control center, or they may be real-time systems that require Wide-Area Synchrophasor data as inputs to Demand-Response or Protection systems. No matter what the case may be, the performance of the system should be such that the nominal PMU output can be handled effectively by the communications and processing infrastructure between

source and sink. Therefore, data rates need to be set so that they fit with available bandwidth. As described, these will nominally be set at 50/60 Hz, or 100/120 Hz for the OpenPMU telecommunication module, so a chosen WAN communications medium should have supporting bandwidth.

Additionally, the multicasting method for inter-module communication is not suitable for WAN-traversal. Setting up IP multicast on a WAN requires either specialized VPN systems or specially configured multicast router infrastructure, which would require all routers to be within utility control. This would be unfeasible.

The current prevailing WAN Synchrophasor protocol is the IEEE C37.118-2011 standard. However, this is highlighted in IEEE C37.244 [15], [16] as requiring significant modification to be a well-rounded communications protocol. The Authors believe C37.118 to be a suitable data representation standard for PMU data. Although, with a lack of extensibility, an XML based structure would be preferred, such as that described in this paper. However, for compatibility the OpenPMU shall be flexible to accommodate various protocols, again highlighting its modular design. The core issue with existing WAN Synchrophasor protocols is that underlying transmission is not well-defined. For example, with C37.118.2, the standard does not specify any specific transport mechanisms, beyond recommending use of TCP or UDP communications.

To gain similar flexibility as with the multicasting technique used for inter-module communications, pseudo-multicast mechanisms can be used on WAN systems. MQTT [17] is one such protocol, which was ratified by the IEC in 2016. MQTT is a publish-subscribe protocol which utilizes broker systems to manage message-centric communications within a hierarchical topic-based system. MQTT is built upon TCP so is inherently reliable; it can also natively utilize TLS, which encrypts data transmissions adding much needed security when traversing WAN system. The low two-byte overhead and simple setup makes MQTT an attractive option as a transport mechanism. It is well-suited to the transport of Synchrophasor data to multiple subscribers simultaneously.

VIII. License

OpenPMU components are individually licensed by their respective authors, as identified in the accompanying license files. Typically either GPL license [18] or BSD license [19] are used. Likewise with regards the hardware aspects of the OpenPMU, the design files require protection by copyleft licenses. Examples of such licenses include [20], [21].

IX. Community Involvement

Resources described are available from the www.OpenPMU.org website. The authors welcome collaboration and encourage use of the OpenPMU pages on the Facebook and Twitter social media platforms.

The authors particularly welcome suggestions for future additions and seek collaborators and partners who will help contribute to, test and deliver future releases.

X. Conclusion

This paper has described a modular PMU design which enables the modules of a PMU to be interchanged with ease. This allows experts in the respective areas to focus their work in their module of interest, with certainty that communication is possible with other relevant modules using the schema described. This method has now been adopted by the OpenPMU project, which provides reference designs.

XI. References

[1] Phadke, A.G., Thorp, J.S.; “Synchronized Phasor Measurements and Their Applications.” New York: Springer, 2008.

[2] 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

[3] IEEE Standard for Synchrophasor Measurements for Power Systems -- Amendment 1: Modification of Selected Performance Requirements," IEEE Std C37.118.1a-2014 (Amendment to IEEE Std C37.118.1-2011) , vol., no., pp.1,25, April 30 2014

[4] 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.

[5] "IEEE Standard for Synchrophasor Data Transfer for Power Systems," IEEE Std C37.118.2-2011, 2011, Online: http://standards.ieee.org/findstds/standard/C37.118.2-2011.html [Accessed: November 06 2016]

[6] ISO IEC, “Communication networks and systems for power utility automation – Part 90-5: Use of IEC 61850 to transmit synchrophasor information according to IEEE C37.118.” 2012.

[7] Khan, R., McLaughlin, K., Laverty, D. & Sezer, S.: “Analysis Of IEEE C37.118 And IEC 61850-90-5 Synchrophasor Communication Frameworks,” IEEE PES General Meeting 2016, July 2016

[8] Laverty, D. M., O'Raw, J. B., Li, K. & Morrow, D. J.: “Secure Data Networks For Electrical Distribution Applications,” Journal of Modern Power Systems and Clean Energy . 3, 3, p. 447-455, Sept. 2015

[9] ISO, “ISO/IEC 7498-1 Open Systems Interconnection Model (OSI).”

[10] Extensible Markup Language (XML), Online: https://www.w3.org/XML/ [Accessed: November 05 2016]

[11] Beaglebone, Online: http://beagleboard.org/bone [Accessed: November 05 2016]

[12] Raspberry Pi, Online: https://www.raspberrypi.org/ [Accessed: November 05 2016]

[13] Arduino, Online: https://www.arduino.cc/ [Accessed: November 05 2016]

[14] IETF, RFC 4648, “The Base16,Base32,and Base64 Data Encodings,” October 2006

[15] IEEE Guide for Phasor Data Concentrator Requirements for Power System Protection, Control, andMonitoring," in *IEEE Std C37.244-2013* , vol., no., pp.1-65, May 10 2013

[16] J. Hastings, D. Laverty and D. J. Morrow, "Towards an improved phasor measurement unit data communications framework," *2014 IEEE PES General Meeting | Conference & Exposition*, National Harbor, MD, 2014, pp. 1-5.

[17] ISO/IEC, “ISO/IEC 20922:2016 Message Queuing Telemetry Transport (MQTT) v3.1.1.” ISO/IEC, 2016.

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

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

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

[21] The TAPR Open Hardware License, Online: http://www.tapr.org/ohl.html [Accessed: Nov 05 2016]