energies

MDPI

Article

Design and Implementation of Low-Cost Phasor Measurement Unit: PhasorsCatcher

David Schofield [1], Debashish Mohapatra [2], Harold R. Chamorro [3], Juan Manuel Roldan-Fernandez [4], Kouzou Abdellah [5,6,7] and Francisco Gonzalez-Longatt [8,*]

1 Centre for Renewable Energy Systems Technology, Loughborough University, Loughborough LE11 3TU, UK; dave.schofield@open.ac.uk
2 Department of Electrical Engineering, National Institute of Technology Rourkela, Rourkela 769008, Odisha, India; mailto.lipun@gmail.com
3 Department of Electrical Engineering, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden; hr.chamo@ieee.org
4 Department of Electrical Engineering, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, 41092 Sevilla, Spain; jmroldan@us.es
5 Institute for Electrical Drive Systems and Power Electronics, Technical University of Munich (TUM), 80333 Munich, Germany; kouzouabdellah@ieee.org
6 Applied Automation and Industrial Diagnosis Laboratory (LAADI), Faculty of Science and Technology, Ziane Achour University of Djelfa, Djelfa 17000, Algeria
7 Electrical and Electronics Engineering Department, Nisantasi University, Istanbul 34398, Turkey
8 Department of Electrical Engineering, Information Technology and Cybernetics, University of South-Eastern Norway, 3918 Porsgrunn, Norway
* Correspondence: fglongatt@fglongatt.org

Abstract: The need for Phasor Measurement Units (PMUs) is rising as renewable energy sources become more prevalent in power networks since the rate of change of frequency is being deteriorated. Appropriate and accurate network measurements are a requirement for the precise monitoring and control of the system. This paper presents a low-cost PMU development, the so-called PhasorsCatcher, for the frequency and rate of change of frequency measurements in power networks, using sufficient but straightforward modular and reconfigurable friendly technology for its implementation. The entire hardware design, schematics, and instrumentation components are shown. Moreover, the visualisation has been calibrated and verified through an experimentation set-up and the existing electrical and communication standards.

Keywords: phasor measurement unit; wide area measurement systems; rate of change of frequency; frequency measurements; instrumentation; hardware module

Citation: Schofield, D.; Mohapatra, D.; Chamorro, H.R.; Roldan-Fernandez, J.M.; Abdellah, K.; Gonzalez-Longatt, F. Design and Implementation of Low-Cost Phasor Measurement Unit: PhasorsCatcher. *Energies* **2022**, *15*, 3172. https://doi.org/10.3390/en15093172

Academic Editor: Ali Mehrizi-Sani

Received: 28 March 2022
Accepted: 25 April 2022
Published: 26 April 2022

1. Introduction

Today, power systems are massive, multifaceted devices in continuous evolution, requiring the development of innovative monitoring systems that capture measurements over wide areas and subsequently use such measurements to provide grid stability analysis and improved operation [1]. The role of phasor measurement units (PMUs) in the power system stability maintenance and power quality is crucial for avoiding interruption in the service [2]. With the rise of this technology and the applications in development, it has been necessary to propose low-cost platforms and testing units for the further continuation and opportunity to explore and incorporate innovative techniques and methods that help the power systems to continue progressing [2,3]. The motivation for low-cost PMU development has been triggered by the new generation of custom-made programmable devices and their relatively easy scripting, the open-source platforms incorporated, and the increasing community of researchers and manufacturers on the topic [4].

Figure 1 shows a generic architecture of a phasor measurement network architecture, where PMUs process measurements (voltages and currents) and provide a data stream of

time-synchronised phasor data [5]. The *Phasor Data Concentrator* (PDC) is a specific-use computer that receives the phasor data streams from the PMU and produces a real-time, time-aligned output data stream; finally, a super-data concentrator (SPDC) is merely a central PDC that receives, processes, and sends out data/information to applications and/or storage and or signals for control/protection.

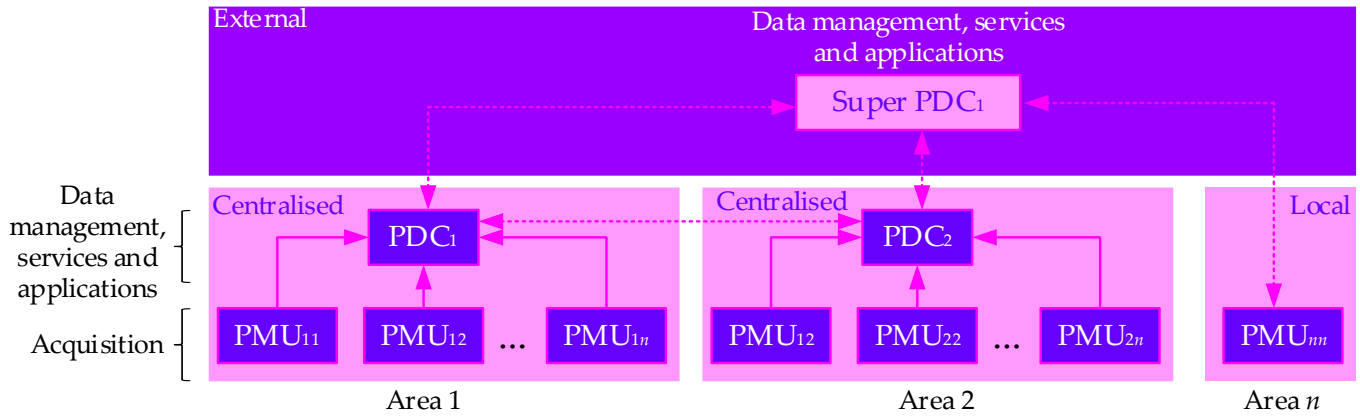

Figure 1. The generic architecture of phasor network architecture, suitable for WAMPAC.

The PMU technology takes advantage of the GNSS (Global Navigation Satellite System) to provide time-stamps for all the measurements. It ensures that all phase angle measurements are synchronised to the same time. These time-correlated measurements are referred to as synchrophasors. The cost of microcontrollers and single-board computers has been reduced in recent time. Following the recent developments on low-cost PMUs, this work proposes a modular friendly hardware implementation of PMUs for the incorporation in power networks and appropriate tracking of frequency and rate of change of frequency (RoCoF) measurements.

This project's starting point and motivation was the previous work developed by Debashish Mohapatra in this MSc project titled "*Development and Hardware Implementation of a Phasor Measurement Unit using Microcontroller*" [6], 2015.

In 2018, David Schofield was finishing his MSc at CREST (Centre for Renewable Energy Systems Technology) at Loughborough University, and he developed the MSc project titled "*Development and Implementation of a Reduced Cost Phasor Measurement Unit*". The project was financially supported by Professor Francisco Gonzalez-Longatt and named "*PhasorsCatcher*". D. Schofield and F. Gonzalez-Longatt published a paper titled "Design and Implementation of a Low-Cost Phasor Measurement Unit: A Comprehensive Review" (2018) [7].

"*PhasorsCather*" is a project dedicated to providing the general public with all the details needed to build its own low-cost PMU. The files and details required to build the PhasorCatcher are located on thesorrsCatcher [8] website and in Prof Gonzalez-Longatt Github's work [9]. This research paper comprehensively explains the design and implementation of a low-cost phasor measurement unit named the PhasorsCatcher.

2. A Review of the Low-Cost Phasor Measurement Unit

There are several factors to consider while implementing PMU: cost, application, location, communication infrastructure, and so on. This section begins with a brief overview of the commercial implementation before moving on to open implementations (open architecture hardware and software). Note that the notion of a virtual PMU is not explored in this work since we focus on hardware-based implementations.

Although there are several manufacturers and plenty of PMU products on the power market, as in the case of ABB, SEL, GE, or Arbiter Systems, the cost of installation and commissioning is relatively high. Additionally, the versatility of exploring new filtering methods, interchangeable sensors, calibration, implementing different algorithms, or just

simply adapting to a specific application is missing. Thus, it requires more flexible and research-affordable solutions for the further inherent development and benefit of the PMU technology itself.

There are many initiatives to develop low-cost PMU at the transmission and distribution levels, all of which originate from academics in an effort to mass use the benefits of phasor measurements for operation, control, and protection. For instance, authors in [10] proposed a microcontroller PMU architecture calibrated and tested under different phase and frequency measurements. This architecture is compared to a commercial PMU showing adequate performance monitoring of the Brazilian grid. An experimental platform developed for educational purposes is presented in [11], including PMU functionalities. Another low-cost PMU prototype proposal was developed in [12], based on a Field Programmable Gate Array (FPGA) exploring latency requirements of PMU applications with a considerably commercially reduced cost. A PMU based on a low-cost microcontroller that integrates data acquisition, data processing, and data communication is presented in [13], analysing stationary conditions with frequency offsets and harmonics. By using a Digital Signal Processor (DSP), a low-cost PMU prototype was developed in [14] that follows the IEEE C37.118 standards [15], being evaluated for frequency estimation during several scenarios.

A LOw COst (LOCO) PMU is developed in [16]; it considers multiple modular components implemented in Raspberry Pi or BeagleBone Black tested on frequency and rate of change of frequency (RoCoF) and steady-state compliance. (Available online: http://ide4l.eu/, accessed on 20 March 2022)

Other more individualised efforts to create a low-cost PMU include "Design of an Inexpensive Residential Phasor Measurement Unit" presented in [17], the MSc Thesis of Mr Onisokonikumen Valiant Sampson [18], and the MSc thesis of Mr Debashish Mohapatra [6].

As explained above, several scientific publications describe the practical implementations of PMUs that have been designed and built using open architecture hardware and software, such as Mohapatra [6], and the *OpenPMU project* [19] describes the design and build of a functional PMU, using open-source hardware and software in some detail and also includes the code for the software implementation. The OpenPMU project describes the implementation of an OpenPMU design (OpenPMU V1) using *National Instruments* (NI) hardware and *LabView* software [20]. There are comprehensive instructions available on how to build the device and implement the software to run on it, making this a useful development tool [21]. However, the use of proprietary hardware makes this an expensive module, especially if the appropriate software licenses are not already held.

Although pictures of an existing module appear in [22], no instructions for implementing the hardware or software are available. A later version (OpenPMU V2) based on truly open-source hardware and software is described, but there are no instructions for implementing the hardware or software available.

In this paper, the authors compared three open access and low-cost PMU implementations: Mr Debashish Mohapatra's MSc thesis [17], OpenPMU V1 (see Figure 2) [1], and OpenPMU V2 (see Figure 3). These three projects have enough data to provide a reasonable comparison in numerous areas, and a summary of that comparison may be seen in Table 1.

Figure 2. OpenPMU V1. Photo: Credits to National Instrument [19].

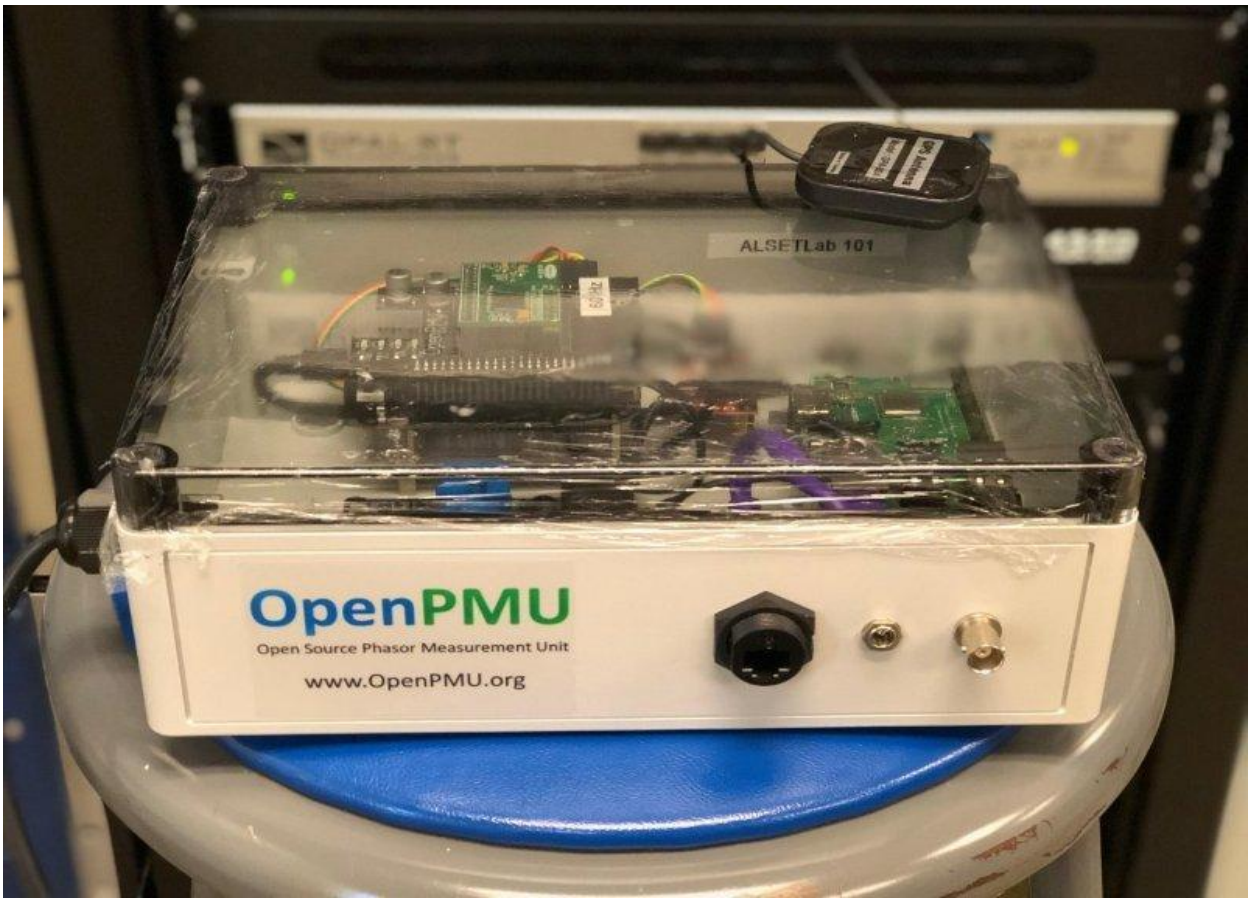

Figure 3. OpenPMU V2. Photo: Credits to Dr Luigi Vanfretti [23].

Table 1. Comparative results among low-cost PMUs.

PMU Name	OpenPMU V1 [24]	OpenPMU V2 [25]	'Mohapatra' [6]
Approximate hardware cost	GBP 770 (2013)	GBP 80 without power interface	GBP 400
Open-source hardware?	No (uses NI DAQ module)	Yes (BeagleBone Black, custom DAQ cape)	Yes (Arduino Due, custom interfaces)
Open-source software?	No (uses LabView)	Yes (Linux)	Yes (Linux(?), Python
Implementation difficulty	Medium (software routines available)	Hard (no details of exact configuration or software routines available)	Medium (software routines available, hardware description not complete)
Comments	Not practical due to proprietary hardware and software platforms.	This is the most promising long-term option.	More complex and expensive that OpenPMU2 (uses 3 x processors) and needs an external PC for processing and display. However, this is the most expedient option.
Score (1 = low, 3 = high)	Cost: 1 Open-source: 1 Availability: 2	Cost: 3 Open-source: 3 Availability: 1	Cost: 2 Open source: 3 Availability: 3

3. Proposed Low-Cost Phasor Measurement Unit (PMU)

The proposed PMU follows the typical structure of any commercially available PMU; it consists of a signal acquisition module where the instantaneous line voltage (and possibly current) information is captured from the lines to be measured and a GPS receiver module where the *Pulse Per Second* (PPS) signal, time-stamp, and location information is received. The proposed PMU also incorporates a processing module where the acquired signal is transformed into time-stamped information such as time *synchronised Root-Mean-Square* (RMS) voltage, phase, frequency, and rate of *change of frequency* (ROCOF) information. Finally, a communication module is also where this synchrophasor information is put into the correct format and transmitted to a remote *Phasor Data Concentrator* (PDC) and any local display, as shown in Figure 4. The following subsections briefly discuss the objective of each module, and then the following sections in this paper discuss details of the design, implementation, and testing of them.

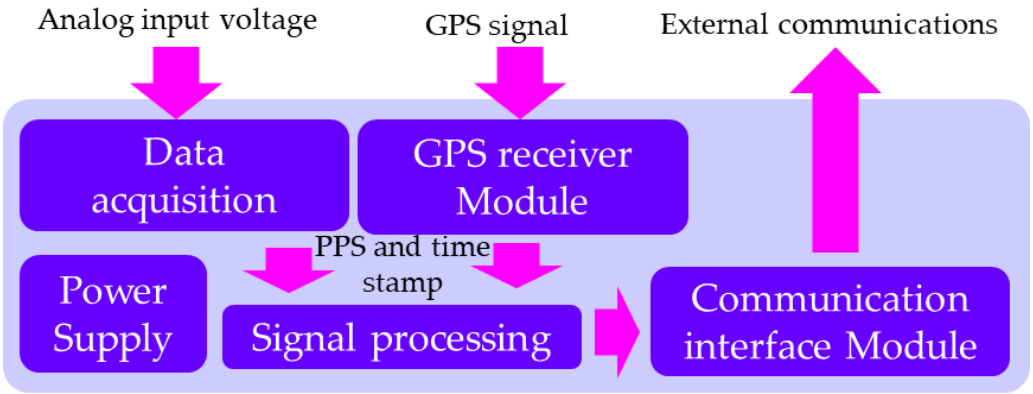

Figure 4. Conceptual block diagram showing the typical structure of a PMU, similar to the one developed in this scientific paper (see Figure 2).

The conceptual block diagram shown in Figure 4 is deconstructed and expanded in more detail showing the block implementation used in the proposed PMU; see details in Figure 5.

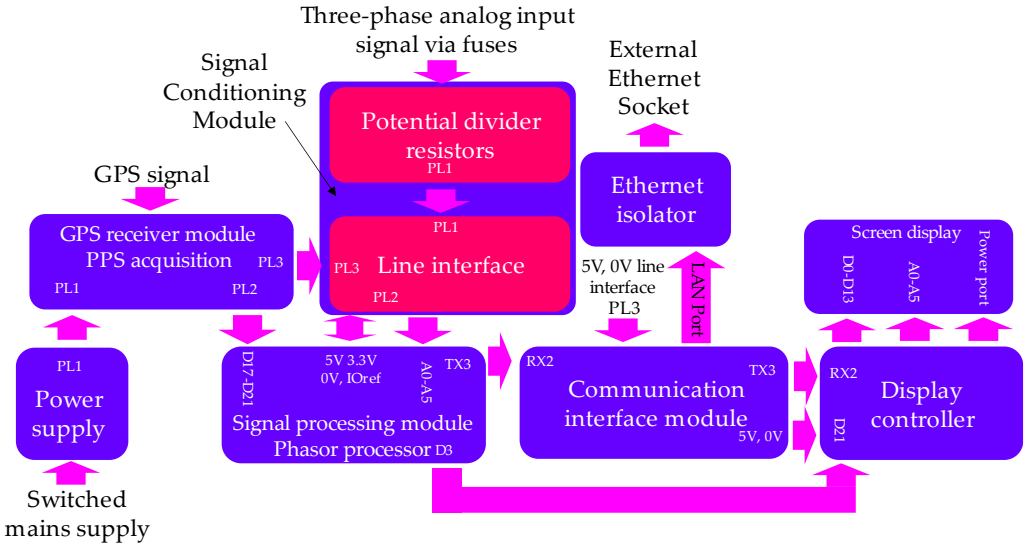

Figure 5. Block diagram showing the proposed PMU implementation, identical to the one developed in this scientific paper. Arrows show the direction flows of signals. Ports are indicated at relevant blocks.

3.1. Signal Acquisition and Conditioning Module

The input data of the proposed PMU are a module receiving the electrical signals coming from the power systems, mainly three-phase voltages and currents. In the context of transmission systems, voltages v_{abc}(t) and currents $i_{abc}(t)$ signals are coming from the secondary of the *potential transformer* (PT) and a *current transformer* (CT) in the substations. However, the input signals coming from the measurement transformers require further step-down and possible isolation from the low-power processing communication components.

In this paper, the practical implementation of the proposed PMU considers three-phase voltages coming from the secondary side of the potential transformers.

Inside the signal acquisition module, a *data acquisition module* (DAM) is responsible for acquiring the raw sample data representing the voltage or current waveforms under test. There are two diverse ways to implement the DAM, using real hardware and using the software.

- Hardware implementation of the DAM: This consists of an *analogue-to-digital converter* (DAC), strictly disciplined to an external time source; using full hardware implementation offers the crucial advantages of higher speeds compared with the software implementation. The authors decided to use a real hardware implementation approach in the actual implementation of the proposed PMU and explained this in detail in later sections of this paper.
- Software implementation of the DAM: In this implementation, a simulated sampling process in a numerical environment is used, e.g., MATLAB, PSSE/Python, or LabVIEW.

3.2. Signal Processing Module

The signal processing module is the core of the PMU implementation, is also called the *phase estimation module* (PEM), and is responsible for operating the phase estimation algorithm employed in the implementation. There are many algorithms available in the literature to estimate the phase angle. However, keep in mind that some algorithms require more computational power and are more hardware demanding than others.

At present, a reference module is available in Python using a least-squares method, which obtains excellent results under IEEE C37.118.1 compliance requirements. Furthermore, the phase estimation stage is agnostic to how the sample data are derived and is essentially *'hot-swappable'*.

3.3. GPS Receiver Module

Precise time synchronisation is required to provide the time-stamp of the signals captured and processed by the proposed PMU. The so-called Global Navigation Satellite Systems (GNSSs) is a cloud of satellites to provide autonomous geospatial positioning. It allows small electronic receivers to determine their location to high precision using time signals transmitted along a line of sight by radio from satellites. In the proposed PMU, the United States' Global Positioning System, also known as GPS, is used. The PMU is equipped with a GPS module that provides an accurate time measurement that can be used to synchronise the measurements.

3.4. Communication Interface Module

The communication interface modules are tasked with providing the external communication functions of the device. When the proposed PMU was implemented, a reference implementation of IEEE C37.118.2 data comms was under test, but the OpenPMU project prefers a more open communication module with security built into its design.

3.5. Power Supply Module

The proposed PMU incorporates an internal power supply unit (PSU), and this power supply is equipped with components to provide safety isolation between the incoming supplies and the user-accessible data interfaces.

4. Signal Acquisition and Conditioning Module

The proposed PMU uses three-phase analogue signals of voltages as inputs; the voltage signals are assumed to come from the secondary of a PT. For the real implementation of the proposed PMU, three voltages in the form of v_{an}, v_{bn}, and v_{cn} are used as input as the voltages provided by the secondary of a star (or wye) connected PT. However, the proposed PMU implementation is able to work with two and single-phase applied, that is, the case of signals taken directly from the power plug available to residential customers. However, the proposed PMU implantation uses sensible electronic circuitry that requires the input signals to step down further and be isolated for safety reasons.

The input analogue signals must be conditioned and prepared for the phasor calculation and production of the synchronised phasor data (i.e., time-stamped voltage signal).

The signal conditioning circuitry has two essential functions: reducing the signals levels to values used by classical electronic circuitry and isolating the input signals and the internal circuitry. The voltage acquisition interface for the incoming three-phase signals offers the potential saving for reducing the cost of the proposed PMU. In this paper, the three-phase signal acquisition and conditioning module is divided into two sections: (1) the line voltage measurement interface and (2) the line frequency measurement interface. Details of those components are presented in the following subsections.

4.1. Line Interface Module

The proposed PMU implementation is based on low-cost microcontrollers, specifically using the power and flexibility of Arduino DUE and Arduino Mega. Those microcontrollers offer the advantages of being equipped with embedded analogue-to-digital converters. However, the ADC is only capable of receiving 3.3-volt DC as input safely. A conditioning circuit is implemented to convert the input three-phase signals 240 V RMS line to neutral (679.21 volts peak-to-peak) to the needed 3.3-volt DC.

An appropriate voltage sensor for this purpose is the use of a hall effect sensor. The use of the hall effect sensor allows the production of an output voltage signal independent of the rate of the detected field, and also, the hall effect sensor can measure zero speed. In addition, the hall effect sensors work over a wide temperature range, provide highly repeatable operation, and are capable of measuring a large current.

The line interface circuitry is designed using an LEM LV25-P [26] hall effect voltage transducer to provide line voltage measurement and reinforced isolation to 800 V to IEC61010-1. An equivalent primary circuit is shown in Figure 6.

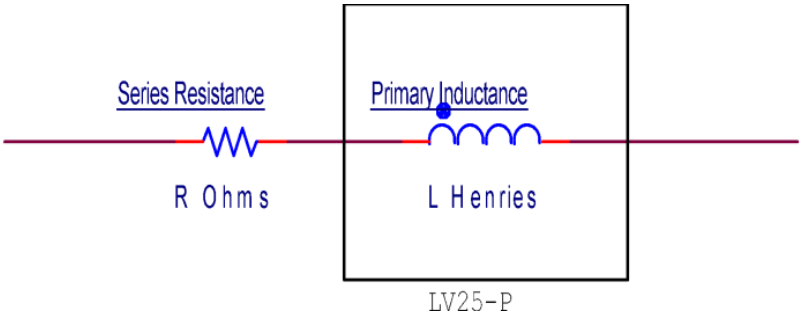

Figure 6. LEM25-P-equivalent circuit.

The voltage transducer operates by measuring a current that is proportional to the instantaneous line voltage and generates a voltage output that is used to drive the analogic-to-digital converter in the proposed PMU, after signal conditioning.

It requires a large current in the primary (10 mA), which leads to dissipation of 2.4 W per phase at 240-volt rms using a 24 kΩ series resistor R and also requires dual voltage supplies to operate (+12 V and –12 V). While it has a linear response to slowly changing input conditions, it also has a transient response time of 40 µs to reach 90% of the final value. The primary inductance of the transducer is not stated, but it can be derived as follows: For a capacitive C_R or inductive L_R circuit with a steady-state output voltage $V_{o,}$ the time constant τ of a circuit is related to the fraction of the final voltage reached $V_c(t)$ after an elapsed time t by:

$$\frac{V_c(t)}{V_o} = 1 - e^{\frac{-t}{\tau}} \quad (1)$$

where $\tau = R/L$ is the time constant. The impedance X_L of an inductor L at frequency f is given by $X_L = 2\pi fL$. For 90% as given in the datasheet [26], Equation (1) gives a value of t/τ of 2.30. Using a primary resistance of 25 kΩ for R as used in [26] and the value of t of 40 µs, Equation (2) gives the primary inductance L = 2.3 H. The bandwidth of a circuit such as an RL filter is defined as the frequency at which the output amplitude is equal to half the input amplitude; i.e., X_L and R are equal. Using (3) gives a bandwidth of 1661 Hz. Whilst adequate for a steady-state 50 Hz or 60 Hz signal, it means that rapid subcycle variations and harmonics cannot be accurately captured.

One alternative is the use of standard iron-cored voltage transformers, which also have an equivalent circuit, as shown in Figure 6. Whilst these have the advantage of providing isolation and being cheap, in order to give an appropriate core size to power rating ratio, the number of primary turns is such that the core is driven at a reasonably high peak flux density. The practical outcome of this is that at the peak of the sine wave input, the core inductance falls, and as it does, the current for a given *di/dt* increases. As this happens, more of the supply voltage is dropped across the internal winding resistance R, and the voltage output becomes distorted. In addition, the transformer requires a significant primary inductance L to keep the magnetising current low, which restricts the bandwidth in the same way as the LEM transducer.

A third alternative is to use *Rogowski coils* [15] as a voltage transducer in the same way as the LEM transducer, using a resistor to provide a primary current that is then converted into a voltage. These have the advantage of providing isolation and can be constructed cheaply in quantity using PCB technology [3]. They also have good high-frequency responses allowing accurate reporting of fast fluctuations and harmonics. However, the output amplitude for a fixed amplitude AC signal is frequency-dependent, requiring com-

plex signal processing either in hardware or after the signal acquisition in software, both of which would be difficult to implement at this stage without introducing further distortions.

4.2. Isolation Barrier

One of the essential aspects of the proposed PMU implementation is safety and, secondarily, cost. From a safety point of view, the isolation barriers are used to provide galvanic isolation at a time that protects electrical signals in potentially explosive or other hazardous areas.

For the baseline PMU design proposed in [6,24], the isolation barrier is placed between the incoming supply to be measured and the data acquisition interface, as shown in Figure 7. Whilst this has the advantage of allowing the PMU circuitry to be earthed and therefore not requiring any further safety considerations, the implementation of the isolation barrier here required expensive hardware; the team found the possibility of reducing the total initial cost of the proposed PMU by taking an alternative approach for the insulation barrier.

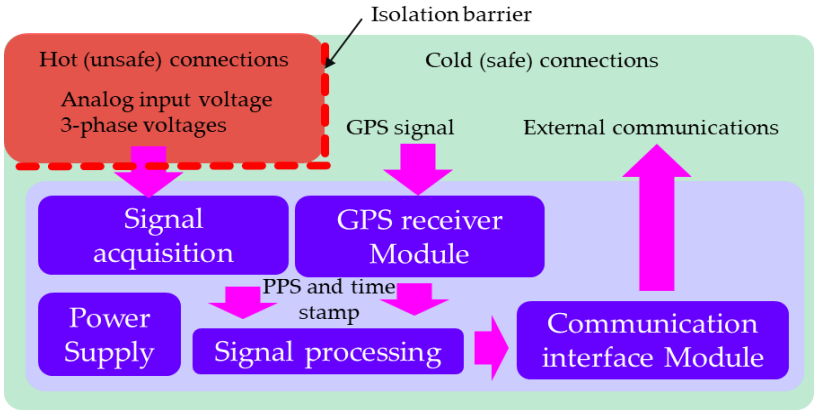

Figure 7. Block diagram of baseline PMU as implemented in [6] indicating the location of the isolation barrier.

The PMU has connections to the three-phase voltage signals to be measured, the main supply to power the PMU circuitry, and an Ethernet port to connect to an external monitoring computer or PDC. As the power supply requires safety isolation, if the isolation barrier is moved to the Ethernet interface and a fully isolated case is used for the module, this simplifies the line voltage acquisition interface considerably. A simple resistive divider (implementation shown in Figure 8) followed by a buffer amplifier can be used to reduce the line voltage to one suitable for signal processing. This approach also provides the advantage of a high bandwidth interface with low harmonic distortion. Consequently, the proposed PMU considers the isolation barrier located as shown in Figure 9 and such a circuit is shown in Figure 10.

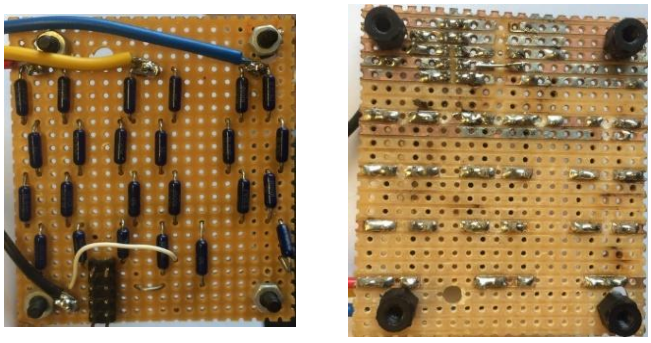

Figure 8. Front (**left**) and back view (**right**) of completed resistor board.

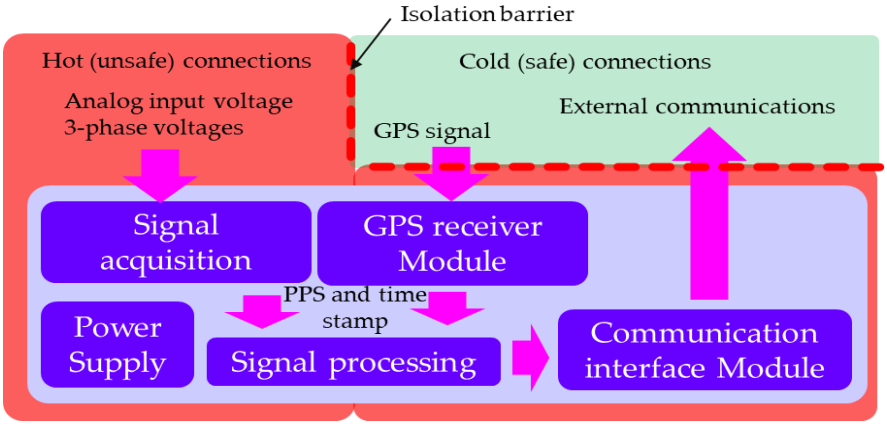

Figure 9. Block diagram of proposed PMU indicating the location of the isolation barrier.

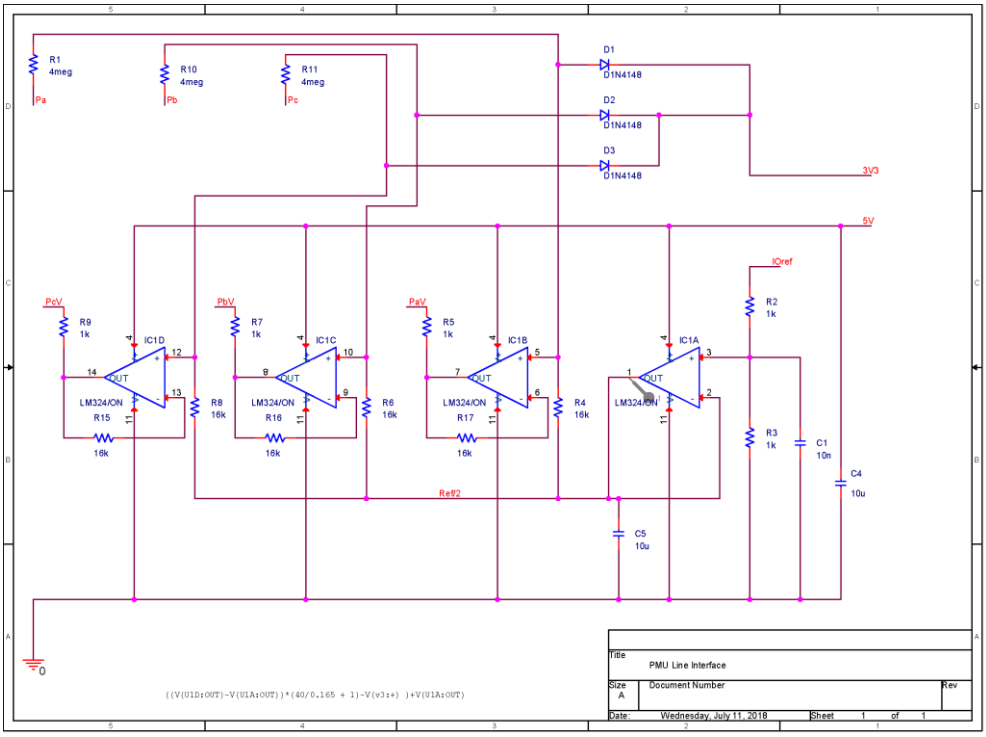

Figure 10. Schematic diagram of the proposed PMU line voltage interface module.

Referring to Figure 10, the three-line phases of the input signals are first connected to the line interface unit by 4 MΩ resistors (not shown in Figure 10) to the connections named PaV_in, PbV_in, and PcV_in (the same nomenclatures used in Figure 10 to make simple to follow) for phases 1, 2, and 3, respectively. In conjunction with R4, R6, and R8 (1 kΩ), these resistors form a potential divider at the input to unity gain buffers IC1B, IC1C, and IC1D (LM324). R10, R11, and R12 (14 kΩ) provide compensation for the input bias current I_{bias} of the operational amplifiers. As the input is a bipolar sine wave signal, but the analogue-to-digital converters in the PMU require a 0–3.3 V positive signal, IC1A in conjunction with R2 and R3 (1 kΩ) provide a voltage at half of the 3.3 V reference voltage (*IOref*) used by the A–D converters for their full scale to provide a 'zero' offset. This gives a transfer function for phase 1:

$$P_aV = \left(V(input) \times \frac{R_4}{(R_4 + R(input)}\right) + \frac{IOref}{2} + \left[I(bias) \times \left(\left(\frac{R_4 \times R(supply)}{R_4 + R(supply)}\right) - R_{12}\right)\right] \quad (2)$$

The same calculation applies to phases 2 and 3. The datasheet for a typical commercial quality op-amp (LM324) gives a value of I_{bias} of 30 nA maximum, making the input bias term in (4) less than 2 nV, and therefore the bias term can be neglected to give:

$$P_aV = \left(V(input) \times \frac{R_4}{(R_4 + R(input)}\right) + \frac{IOref}{2} \quad (3)$$

Using 16 kΩ resistors for R4, R6, and R8, gives a transfer function for the amplifiers P_XV:

$$P_xV = \left[V(input) \times \frac{16}{4016}\right] + 1.65$$

Equation (3) gives a full scale (3.3 V output) of ±414.15 V or 292.8 Vrms. Unlike the baseline implementation, this requires only a 5 V and 3.3 V supply, both of which are available from the microcontroller board that forms the phasor measurement.

The physical implementation of the line interface module is depicted in the photos of the actually implemented boards in Figure 11; constructive details and a list of materials can be found on the PhasorsCatcher [8] website and in Prof Gonzalez-Longatt Github's work [9].

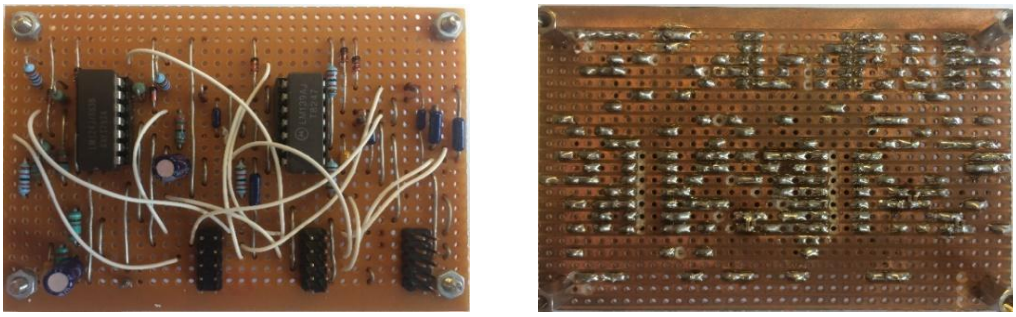

Figure 11. Front (**right**) and back view (**left**) of the completed line interface board.

A processor module simplifies the cost and complexity of the power supply. Simulation of the circuit in Figure 10 using the PSpice [27] simulation platform gives the input and output waveforms shown in Figures 12 and 13 (zoom in to detail the time delay experienced between input and output to the module). A classical frequency response plot is shown in Figure 14.

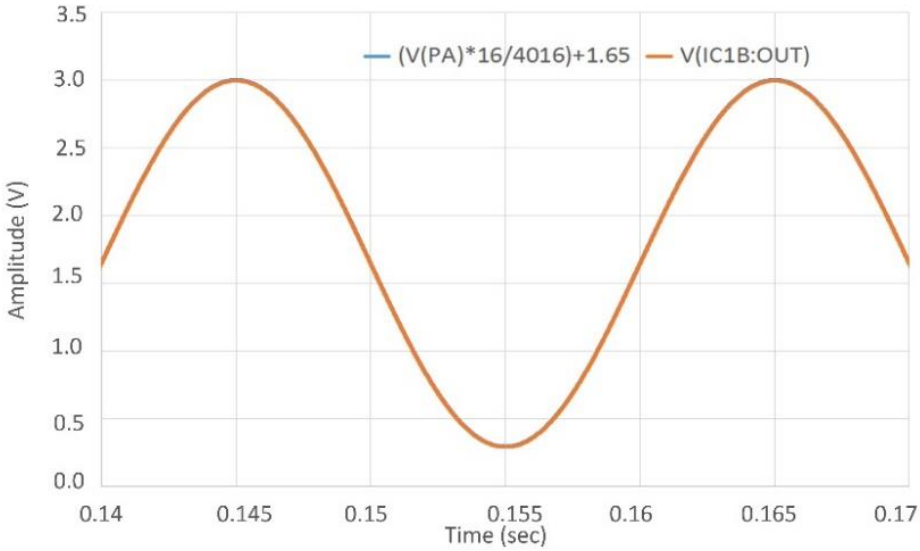

Figure 12. Line voltage circuit simulation response. The blue colour is the input, and the orange colour is the output.

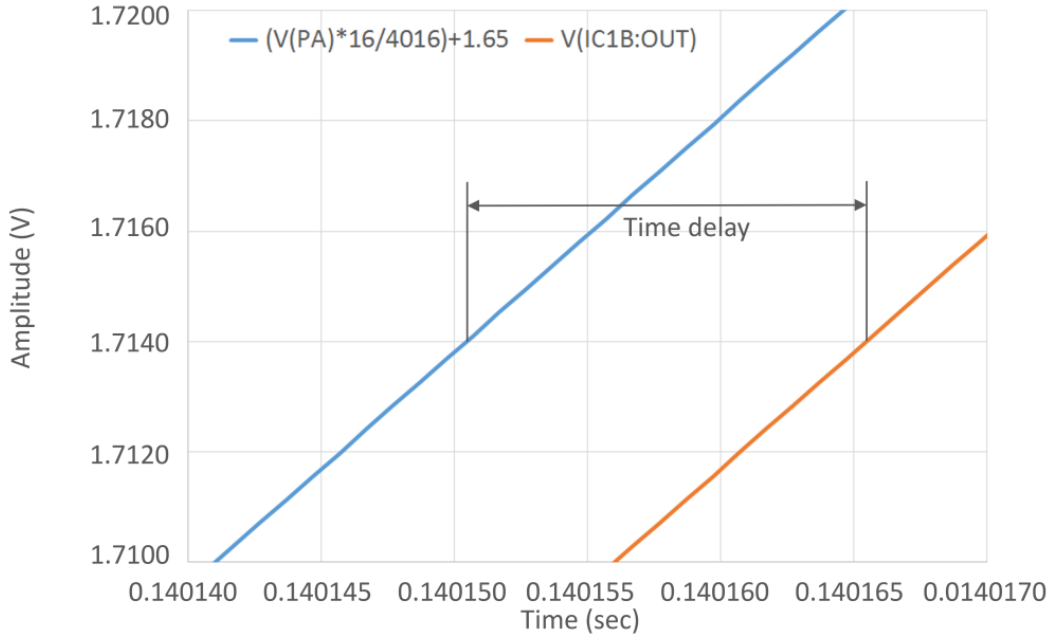

Figure 13. Line voltage circuit simulation delay.

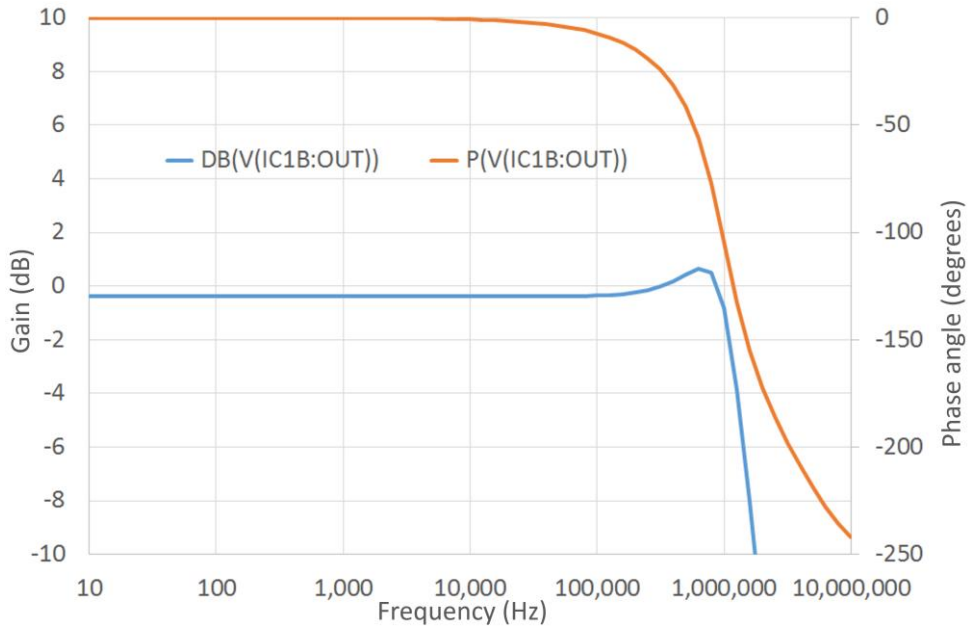

Figure 14. Simulation results of the line voltage circuit simulation frequency response: bandwidth (V_{out}/V_{in}) 10 Hz–10 MHz.

Figure 12 shows the ideal output waveform (orange) and the simulated output voltage waveform. It can be seen that there is no amplitude distortion, and in Figure 13, it can be seen that the output waveform (orange) is delayed by 15 µs, which corresponds to an error of 0.27°. The IEEE C37.118 [15] gives the contribution to the *Total Vector Error* (TVE) of this as 0.47% and gives steady-state limits for both Class M and Class P PMUs of 1% TVE, showing that the interface contributes about half of the allowed TVE. If this needs to be reduced, a correction term can be added to the phase estimator used. In Figure 14, it can be seen that the bandwidth as given by the −3 dB point is more than 1 MHz and that at 10 kHz, the phase error is less than 1°.

4.3. Line Frequency Measurement Interface

The baseline PMU implemented by Debashish Mohapatra [6] used the iron-cored transformers to provide isolation and provide drive to optoisolators that provide a square-wave input to the data acquisition module using a circuit as shown in Figure 15 (see implementation in Figure 16).

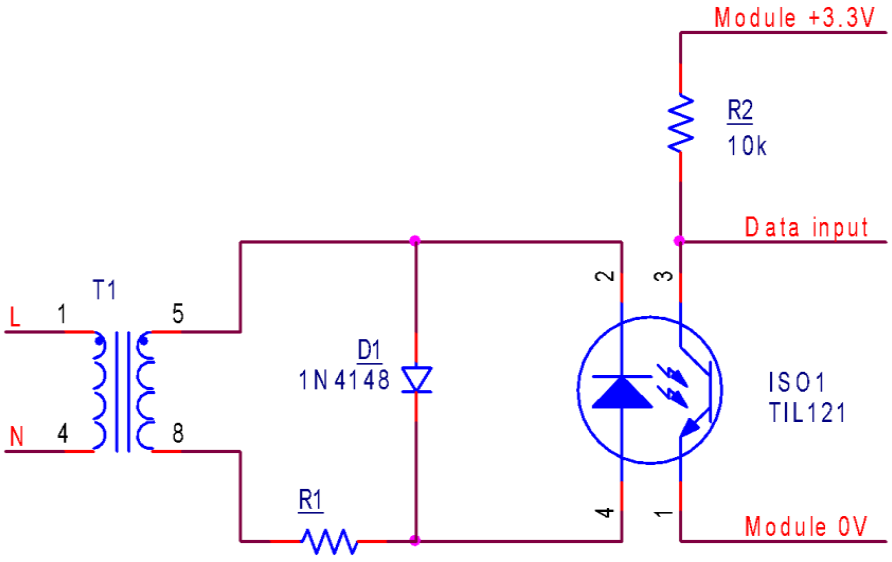

Figure 15. Baseline frequency measurement circuit as developed by D. Mohapatra in [6].

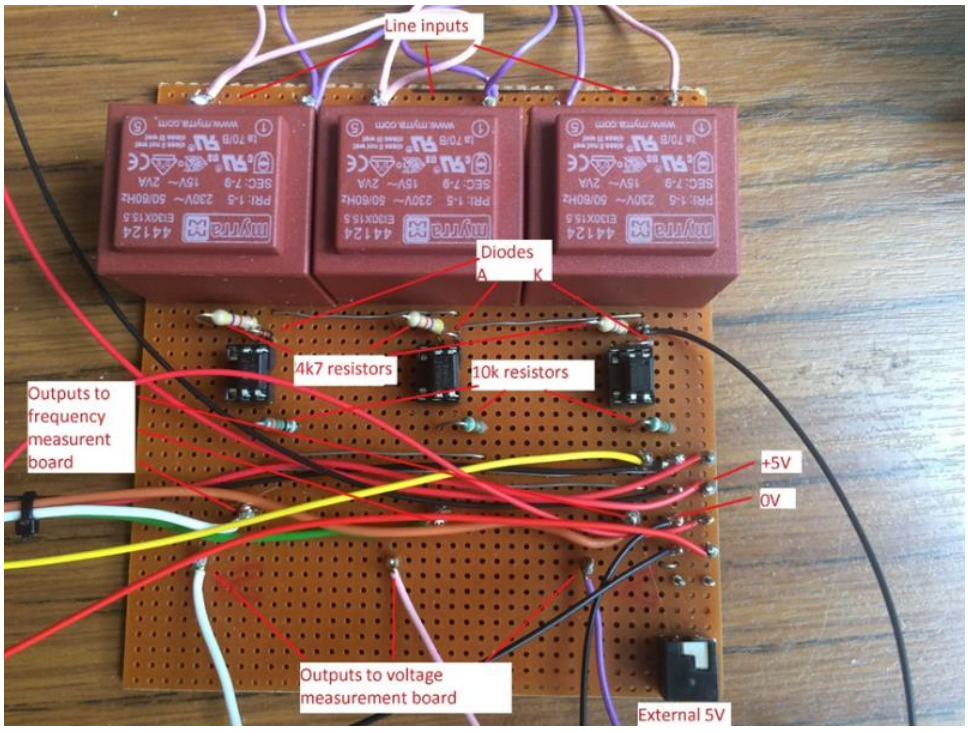

Figure 16. Baseline frequency measurement board implementation as developed by D. Mohapatra in [6].

Unlike the line voltage measurement interface, which requires good linearity and high-frequency response, this interface produces a rising edge produced by the zero crossings of the input waveform, repeating every complete cycle. Whilst not very expensive items, the transformers add significant volume, and as isolation is not required, it can be replaced with a circuit similar to that in Figure 10 but using comparators, as shown in Figure 17.

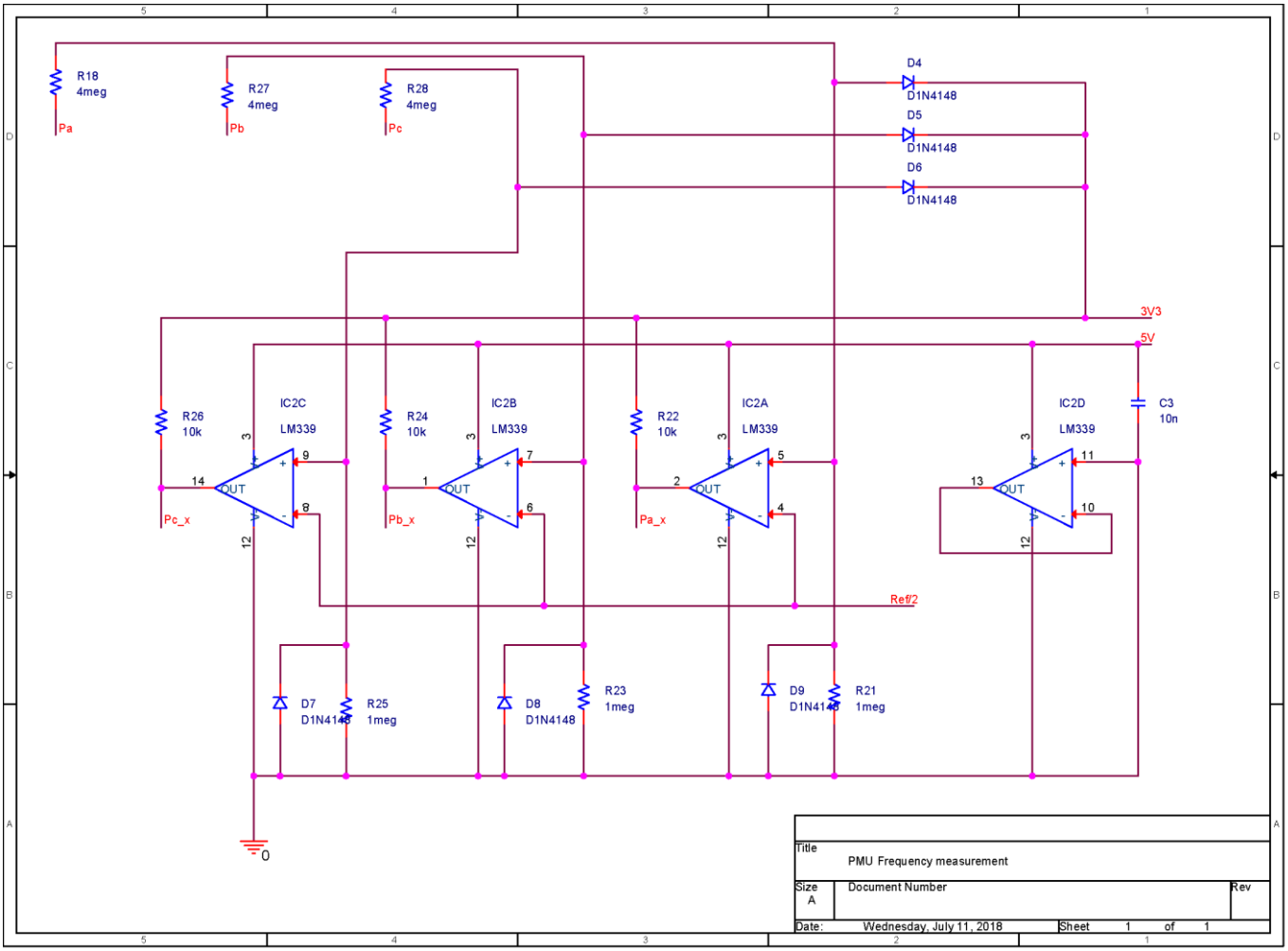

Figure 17. Schematic diagram of the proposed PMU line frequency interface circuit.

The proposed frequency interface circuit uses the same 5 V and 3.3 V supplies as in Figure 10 and uses the *IOref*/2 (1.65 V) reference that Figure 10 generates. It also uses 4 MΩ resistors (not shown) to interface with the three-phase lines. The inputs are clamped to a diode drop below *IOref* /2 and above the 3.3 V line to avoid the inputs exceeding the common-mode range. In addition, the choice of the lower divider resistor (R_1) and hysteresis resistor (R_{42}) gives a rising edge when the input voltage ($\|\|$ represents electrical parallel circuit).

$$V(input) = \frac{R_{42} \parallel R_1 + R(input)}{R_{42} \parallel R_1} \frac{IOref}{2} - [I(bias)R(input) \parallel (R_{42} \parallel R_1)] - V(offset) \quad (4)$$

The LM339 datasheet [26] gives values of *V(offset)* of 9 mV max and *I(bias)* of −400 nA max, giving a maximum offset term of 0.187 V. Whilst this would lead to a slight change in the time from the zero crossings of the rising output edge, it is the time between edges that is measured, and as the values of offset voltage and bias current do not change over short periods, the offset and bias terms can be neglected to give

$$V(input) = \frac{(R_{42} \parallel R_1) + R(input)}{R_{42} \parallel R_1} \frac{IOref}{2} \quad (5)$$

Equation (5) gives a rising edge when *V(input)* = 14.85 V. These component values give a hysteresis of 14.85 V, meaning that the output will not fall until *V(supply)* approaches 0 V. Note that this original design did not incorporate C5, C6, and C7, which were added during testing. As the signal processing module only takes note of rising edges, the timing of the falling edge is not critical. Both the line voltage measurement and line frequency measurement circuits can conveniently be placed together on the same circuit board. The circuit was simulated, and the operation around the positive-going zero crossings is shown in Figure 18. It can be seen that the positive-going output

transition occurs at a line voltage of 14.78 V. As this simulation incorporates terms for *V(offset)* and *I(bias)*, this is in agreement with Equation (5).

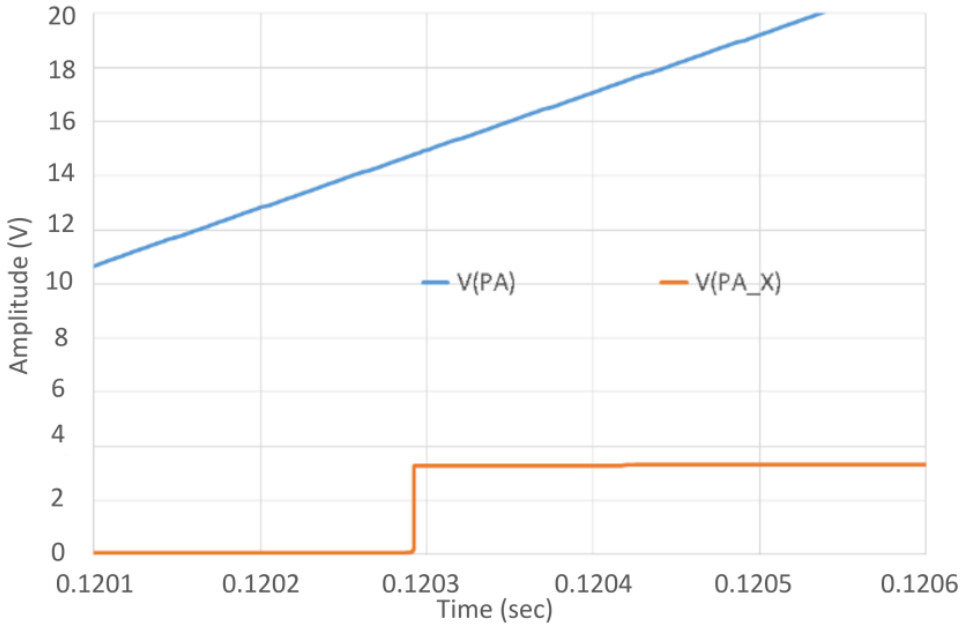

Figure 18. Simulation results of the line frequency circuit.

5. Signal Processing Module

After the three-phase signals are acquired and conditioned, the next step is in charge of the core activity of the proposed PMU, which is the signal processing module. Two essential tasks are done inside this module. One is to process signals and extract information obtained from the raw data, i.e., frequency and ROCOF calculation, and then the processed signals are time-stamped. In this section, the signal processing module is explained.

PMU Reference Generator

The implemented PMU considers all the calculation methods for signal processing, and additional parameter generations are performed over a synchronised reference at the rated frequency of the system (implemented PMU considers 50 Hz). A programmable three-phase signal generator, specifically sine wave, is required. The literature considers two main approaches for this: (i) the use of predefined samples and (ii) a signal generator with programmable capabilities.

The use of predefined samples (pairs of values) for the three-phase signals is a fast implementation, and time series of the three-phase signals are generated using third-party software, e.g., MATLAB, SCiLaB, Mathematica, GeoGebra, etc., or it can be the case of using another software application to store the values in the memory of a microcontroller; the latest implementation can be done by using ProgMEM, a programme that enables storing the time series in a flash (programme) memory instead of SRAM of Arduino microcontrollers.

The predefined samples (pairs of values) approach offers a massive advantage as implementation could be extremely single (no need for signal generation hardware, acquisition hardware, and development of calculation algorithms). However, one of the drawbacks of this approach is that by neglecting the data acquisition process, the computational time will differ; it occurs as the analogue-to-digital conversion time is ignored in the process.

An implementation of a programmable sine wave generator hardware is another option. In this case, the hardware includes a fully controllable signal (three-phase sine wave) generator: amplitude, frequency, and phase shifts can be fully controlled in order to obtain the appropriate waveforms for analysis and estimation of phasors and any other related parameter.

This approach uses a precalculated lookup table that is stored in the flash memory of a microcontroller that it is used as the programmable sine wave generator.

The microcontroller fetches a value from memory and writes it to one of its ports. The digital signal is received by a digital-to-analogue converter (DAC), and it converts this digital value into an analogue voltage. A similar approach is then used by subsequent conversions of the values stored in

the lookup table; a full controllable sine wave is generated when the lockup table is completed. As the PMU used three-phase signals, three lookup tables are used to write data to three ports of the microcontroller, making the three-phase signal generator.

The proposed PMU implementation used an identical approach as D. Mohapatra described in [6,26], Arduino Mega 2560 microcontrollers are used to generate the three-phase sinusoidal signals, and a suitable conditioning system is enabled from an LM358 operational amplifier (op-amp).

The IEEE std C37.118 requires a PMU to be able to provide data streams of the frequency calculated from the input signals and of the frequency to each one of the phases. In addition, it requires providing the rate of change of frequency.

When considering methods to calculate the frequency of a signal, there are many methods reported in the scientific literature; methods range from a straightforward calculation of the time required by the signal to cross zero to very complicated mathematical calculations; in general, they require a signal processing method. A very ingenious but straightforward method is to convert the input signal from a sine waveform into a square wave, where the signal's frequency can be elementarily calculated by measuring the time required between two consecutive pulses. In this proposed PMU implementation, the ingenious sine to square method is used. This method provides a straightforward but efficient way to calculate the signal frequency without requiring optimisation to accurately calculate the signal frequency when there is a considerable deviation from the rated frequency. The authors of this paper explored some other algorithms to extract the frequency from the signals, but the computational power and memory requirements prove to increase the total cost of the PMU. The method implemented in the PMU has been demonstrated to work, but further refinement may be needed in harmonic perturbation signals.

A circuitry was designed to produce a square wave from the pulse stream (adjusted from 0 V to 3.3 V), and the steam is used as input at the Arduino Due microcontroller, working as a phasor processor module.

The modern PMU takes advantage of the Global Navigation Satellite Systems (GNSSs) to accurately track the time and uses them to create the synchronisation between a measurement with the support of a time-stamp. The proposed PMU is equipped with a GPS receiver module (also referred as PPS acquisition) to cope with three essential tasks:

- Provide a pulse per second (PPS) signal: The PPS is used to generate a stream signal that triggers the GPS-disciplined oscillator (GPSDO) to produce the pulses used for signal sampling. The GPSDO works by disciplining (or steering) the oscillator output to the GPS signal via a tracking loop. The idea behind the GPSDO is to provide an accurate time signal up to nanoseconds capable of generating frequency accuracies and stabilities up to parts per trillion
- Provide universal coordinated time (UTC) reference: the GPS navigation system is able to provide reference time using the UTC reference system. This very accurate time is used by the microcontroller in the signal processing module to accurately calculate the phasors at the time that provide the time-stamp. The time-stamp is extraordinarily important in the PMU, as it allows synchronised signals irrespective of their origin and the time delays involved between the transmission/reception of the data stream from the MPU and the centralised phasor measurement data concentrator (PDC).
- Provide geographical coordinates: One of the advantages of the GPS signals is to obtain a very accurate geographic coordination (longitude and latitude); the geolocation tag is addressed to the phasor data, and it can be transmitted together to the PDC where the geographical location can be accurately used for representation in any application including mapping. Modern wide area application takes a step forward using geodata for oscillation source identification.

The GPS module requires the GPS sensor to successfully fix tracking of at least three satellites simultaneously to provide the PPS signal. The GPSDO requires the 1 PPS signal from the GPS module in order to generate the sampling initialisation clock.

In this PMU implementation, the GNSS used is the classical Global Positioning System (GPS), which evolved from the originally Navstar GPS. The GPS is a satellite-based radio navigation system that is owned by the United States government and operated by the United States Space Force. The implementation of the proposed PMU uses a GPS module with a supply of 3.3 V DC, and it has a transmitter and receiving pins with serial communication that are used to transmit the information to the phasor processor microcontroller where the UTC time and geolocation information (longitude and latitude) are derived.

The IEEE C37.118 establishes that for a rated frequency of 50 Hz, it is desirable that up to 50 phasors may be calculated per second. In this case, the job of the GPSDO is to produce a predefined set of equidistant pulses per second based on the 1 PPS signal generated by the GPS module. The

set of equidistant pulses per second generated by the GPSDO is the core of the sampling used to generate the time-stamped phasor information.

The proposed PMU implementation uses an Arduino Mega microcontroller to generate the required number of equidistant pulses per second from the 1 PPS signal provided by the GPS module. Figure 19 shows the implementation of the GPS module using an ATMEGA328p as the core of the processing unit and a NEO 6 M as a GPS receiver with a built-in 25 × 25 × 4 mm ceramic antenna, which provides a strong satellite search capability.

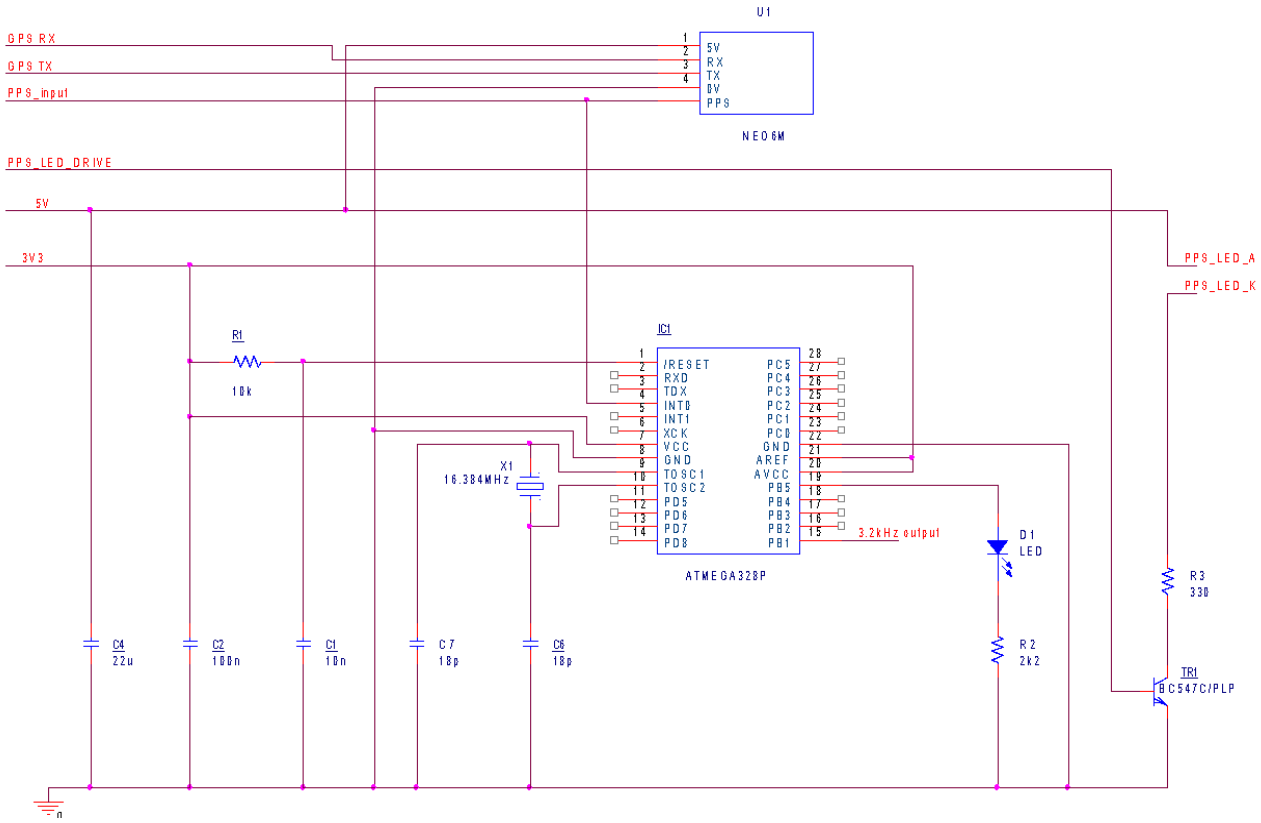

Figure 19. GPS receiver module and PPS acquisition circuit.

The implementation is able to generate 3200 pulses per second (50 µsec width and 400 µsec interval). A simplified flowchart of the programmed GPSDO is shown in Figure 20.

A local clock is programmed in the Arduino Mega microcontroller to provide a synchronised time attached to the UTC time that the GPS module reads over the UART. The local clock provides time data with a resolution of milliseconds, and it is attached in the form of a time-stamp to the phasor calculation and then is transmitted to the PDC. This approach is widely used in many PMU applications and is reported as a time-stamping phasor.

The core of the phasor calculation is implemented by using an Arduino Due micro-controller. The Arduino Due is equipped with an ARM relatively powered Cortex M3 core; moreover, the microcontroller board is equipped with 54 digital I/O pins, 12 analogue inputs, 4 UARTs, and an 84 MHz clock [6]. In addition, the microcontroller board has 96 KB SRAM, 512 KB FLASH memory, and direct memory access (DMA) controller. A 12-bit analogue-to-digital converter is inbuilt with the microcontroller, which can very easily operate at 3200 samples per second. The conversion time of the ADC is 4 µsec.

The proposed PMU implementation uses part of the 96 KB SRAM of the microcontroller to store the three-phase voltage samples as a 64-element buffer (a full cycle of signal); it keeps updating every time a new sample arrives at the microcontroller from the GPSDO. A counter is implemented in the microcontroller board; it keeps a count of the number of samples in the buffer, and when the counter reaches 64, the calculation of the phasor is initiated.

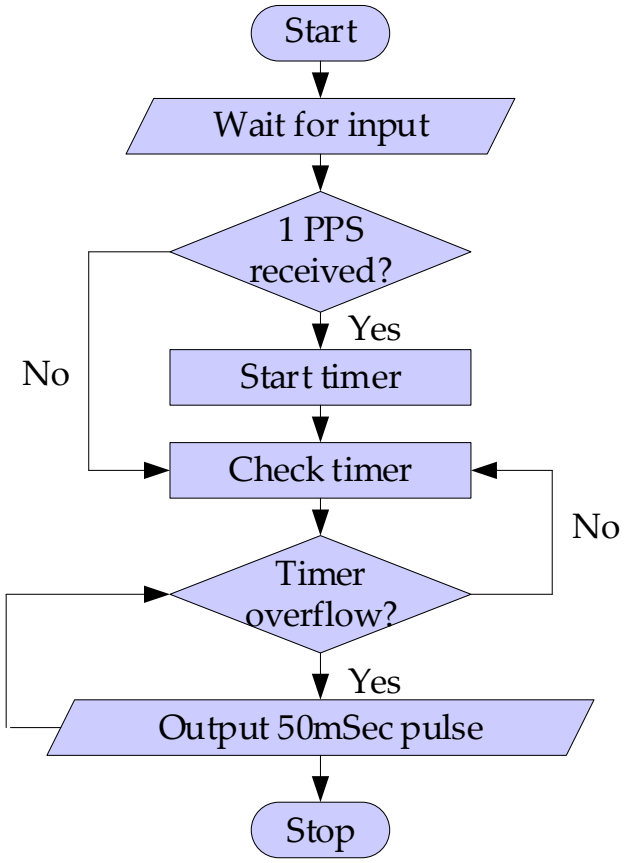

Figure 20. Simplified flowchart of the programmed GPSDO in the proposed PMU.

6. Communication Module

The communication module in the PMU is dedicated to communicating the synchrophasor data to the phasor data concentrator (PDC).

In the proposed PMU, the communications module receives data transmitted by the phasor controller module and then shares the data with the display module and sends the information to the Ethernet interface. The communication module receives the data from the phasor controller module at a rate of 3200 complete data strings per second and then decodes the individual data values from the string. The communication module transmits data for two critical applications: (1) data for the LCD screen to the display controller module and (ii) via the Ethernet interface to an external data monitor or PDC.

The baseline PMU implementation created by D. Mohapatra [6] does not format the data into the format required as required by the IEEE Std. C37-118.2-2011 [15] to be compliant. The implementation of this module in [6] gave no details of the Ethernet interface and did not contain any driver code for the Ethernet module.

For the proposed PMU, the process of creating the communication module evolved during the implementation process. Initially, the authors decided to use an Ethernet interface based on the ENC28J60 chipset. Unfortunately, the drivers were not compatible with the Arduino Due used for the communications controller, and the code for the communications controller would not compile to an Arduino Mega2560 target.

Another interface was tested, this time based on the WJ5100 chipset (see Figure 21), which was more successful, allowing a functional Ethernet interface to be implemented. However, it sent out the data strings one character at a time. The cause turned out to be an artefact of the way that the data string was assembled, which worked for data sent out on a serial line but not as packets on an Ethernet connection—rewriting this section of the code to assemble the whole string before sending corrected this.

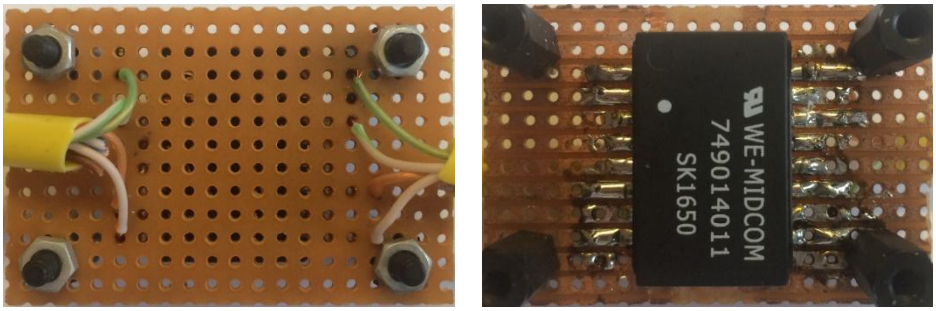

Figure 21. Front (**right**) and back view (**left**) of completed Ethernet isolator board.

In order to create the isolation barrier, as shown in Figure 10, an isolation transformer was added to the Ethernet connection to the external interface. During testing, it was noted that intermittently the voltage data for phase 1 'lost' the first character, i.e., 237.24 was received as 37.24 from the phasor controller module. Observation of the data stream with oscilloscopes and data analysers showed no errors, and as this was very intermittent, a trap was put in place to ignore up to two such errors on this channel only. The code was also added to pass the average frequency and average ROCOF data to the display module and remote monitor.

7. Local Display Terminal Module

A Human-Machine Interface (HMI) is a feature enabled in many commercial PMUs, that enables humans to interact with it. A commercial PMU has a very potent HMI allowing bidirectional communication (read and write). However, typically an HMI can increase the production cost as it requires display screens, input buttons, etc., and depending on the implementation of the HMI, it can substantially increase the cost. In the proposed implementation, the authors decided to follow an implementation similar to the baseline PMU implemented by D. Mohapatra [6]. A local small-size display screen was added to show the main parameters measured by the proposed PMU.

However, data requirements for display are much less demanding than the phasor data used for power system operation and control; as a consequence, the reporting data rate at the display screen uses a prolonged refresh data rate of one phasor per second, making this representation simple and easy to catch by the human eye.

The local display module consists of an Arduino microcontroller board and a thin-film-transistor (TFT) liquid-crystal display (LCD). The TFT LCD screen offers several advantages, including less energy consumption, superb quality visibility, particularly good response time, and less eye strain, etc. A TFT display offer is brighter and faster than a regular LCD display; for this reason, the authors decided to use it for the PMU implementation.

D. Mohapatra's [6] original display module drove a 7″ 800 × 480-pixel UTFT display using an Arduino Due module. The phasor calculation method sends the data to the display module using a serial communication channel; it has a speed of 921,600 bauds. The local display module uses an Arduino Due to receive the data stream (phasor data) and then decodes the data and extracts the variables to be displayed on the TFT screen. The proposed PMU displays the magnitude in voltage (*a*, *b*, and *c*), angle in electrical degrees (*a*, *b*, and *c*), frequency in Hz, rate of change of frequency (ROCOF) in Hz/sec, time (hh:mm:ss), date (dd/mm/yyyy), latitude, and longitude. Then the Arduino Due microcontroller, used in the display module, displays the destructured data on the TFT LCD display using the TFT display upon receiving the same 1 PPS signal from the GPS module. It is important to notice that displaying the data is triggered upon arrival of the 1 PPS signal from the GPS module. One crucial issue is the process of updating the data presented in the display takes around 300 msec to update the display completely.

The proposed PMU reduced cost by using a smaller screen display, only a 3.5″ 400 × 240-pixel module was available, but it imposes several technical challenges as it had an incompatible driver and hardware interface. This meant that significant modification was required to the code to drive the display and format the output correctly, and also it became necessary to change the processor from an Arduino Due to an Arduino Mega2560 as the display drivers caused compiler errors for the Arduino Due target. When this was initially done, the display processor could not correctly receive the communications from the communications controller. Initially, this was due to the different I/O voltages used—the Arduino Due uses 3.3 V logic, and the Arduino Mega2560 uses 5 V logic. A level translator circuit was designed and built, but before it was tested, it was realised that due to the way the UART controller on the Arduino Due works, setting the data rate to 230,400 baud actually sets it

to approximately 250,000 baud. When the display processor was set to 250,000 baud, communications were re-established. As communications between the communications controller and the display controller are only one way, there was no need to connect the 'return' data path, which would need a level shifter to avoid applying an overvoltage to the communications controllers receiving input.

8. Power Supply

The proposed PMU requires a power supply; it must comply with the power supply requirements of the different modules inside the PMU implementation. The baseline PMU implemented by Debashish Mohapatra [5] requires a power supply able to provide five voltage rails (+3.3 V, +5 V, +12 V, +15 V, and −15 V) using two large iron-cored transformers and linear regulators and utilises a temperature monitoring system and fan cooling. The full details of its implementation are found in [6].

The proposed PMU considers a different isolation barrier; therefore, the dual 15 V supplies are no longer required, and no significant current is drawn from the 3.3 V supplies. The power supply can be significantly simplified, providing a single 5 V supply to the Arduino modules, which then generate the 3.3 V supply internally. Although the power supply can be implemented from scratch, the modern implementation commercially available and ready to use at the specialised electronic stores offers an extremely attractive option in terms of cost, size, efficiency, and time of implementation. Consequently, the proposed PMU uses a low-cost off-the-shelf 5 V power supply, such as that used for charging mobile phones.

9. Results and Discussion

The proposed PMU design was built up module by module, starting with the PPS acquisition module, the circuit of which is shown in Figure 19. The data acquisition interface incorporating the line voltage and line frequency interfaces was added next, followed by the communications interface incorporating the isolated Ethernet output and the local display.

9.1. PPS Acquisition Circuit Testing

When the GPS modules were tested, it was found that the output frequency was 3.205 kHz, not the expected 3.200 kHz. In addition, the oscillator was not reset by the PPS signal. An examination of the baseline code showed that a noninteger period (312.5 µs) was being programmed into a timer that could only accept integer settings. This was solved by changing the crystal frequency from 16.000 MHz to 16.384 MHz and adjusting the period setting.

When measured using a Keithley 2000 Digital MultiMeter (DMM), the frequency was found to be 3.20095 kHz. This design uses an 'entrained' oscillator, which is reset by the PPS signal, not a *Phase Locked Loop* (PLL) that uses the PPS signal to align a set of 3200 pulses. Therefore, any error in the frequency generated will result in a shortened final pulse period and possibly less than 3200 pulses (frequency too low) or a shortened final pulse period and 3201 pulses or more (frequency too high). In these circumstances, there will be a small error in the phase estimated during this period.

The resetting error was due to an incorrect port assignment in the baseline PMU schematic [6], and when corrected, the oscillator was correctly entrained by the PPS signal, as shown in Figure 22.

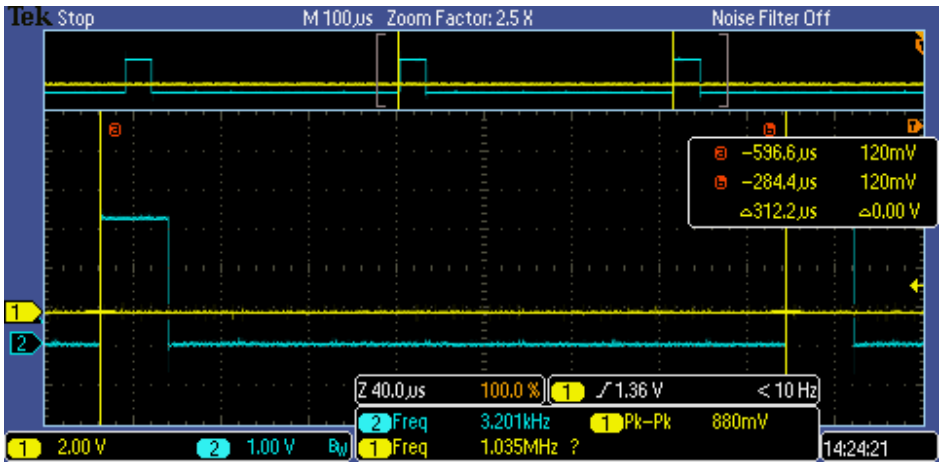

Figure 22. Oscilloscope capture of the 3.2 kHz pulse timing before PPS signal.

Note that as implemented, this circuit does not support the requirement of [15] to insert time quality information. It can be seen from Figure 22 that before the PPS pulse, the period of the 3.2 kHz oscillator is correct at 312.5 µs (subject to cursor measurement accuracy). When the PPS pulse arrives, the period of the pulse is shortened, as shown in Figure 23, due to the slight error in the frequency; this shortened pulse is an additional one. After the pulse, the period between rising edges is shortened but not between falling edges, as shown in Figures 23 and 24.

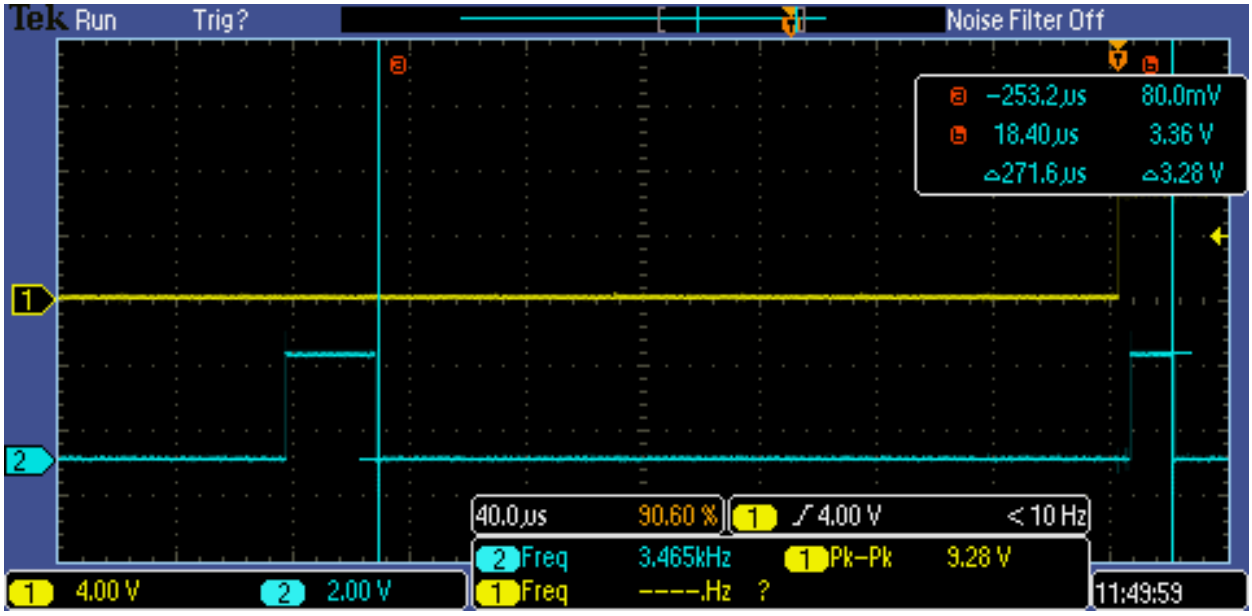

Figure 23. Oscilloscope capture of 3.2 kHz falling edge pulse timing at PPS signal.

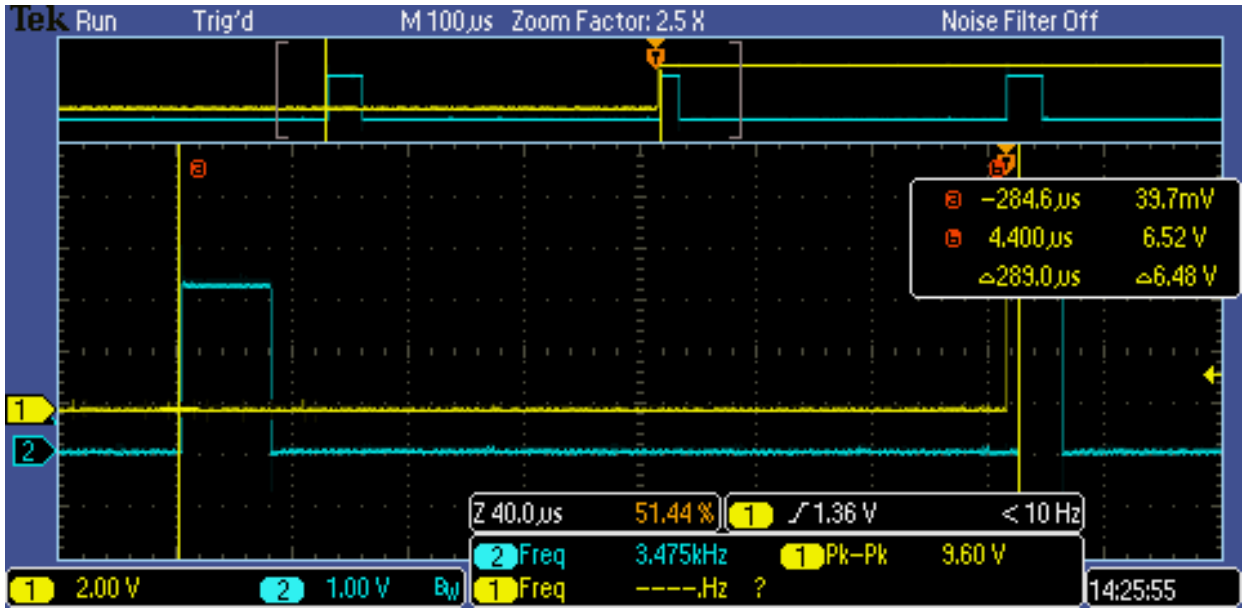

Figure 24. Oscilloscope capture of 3.2 kHz rising edge timing after PPS signal.

In light of this, the active edge of the 3.200 kHz acquisition pulse was changed from rising to falling. This still, however, leaves the oscillator frequency at 3201 Hz, not 3200 Hz (see Figure 25).

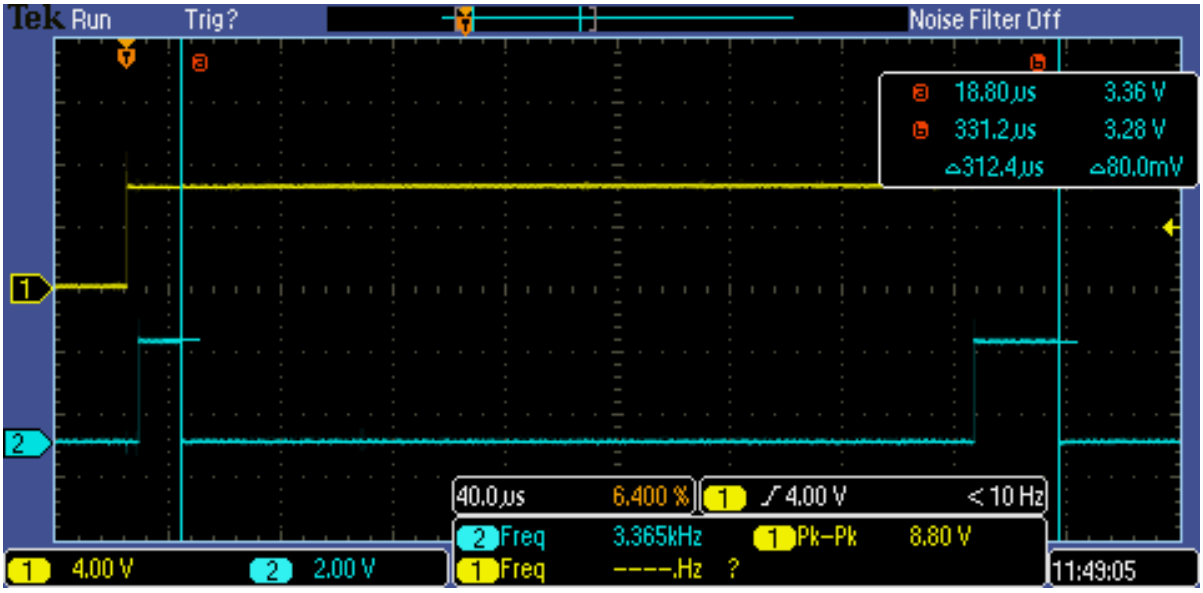

Figure 25. Oscilloscope capture of 3.2 kHz falling edge timing after PPS signal.

9.2. Line Voltage Interface Circuit

In the interests of safety, the initial testing of the line voltage interface circuit (Figure 7) was carried out using a 50 Vrms transformer isolated supply and only connected directly to a live voltage feed once correct operation was confirmed. After correcting a small number of build errors, the circuit worked correctly and gave the appropriate output waveforms. The input and output waveforms are shown in Figure 26.

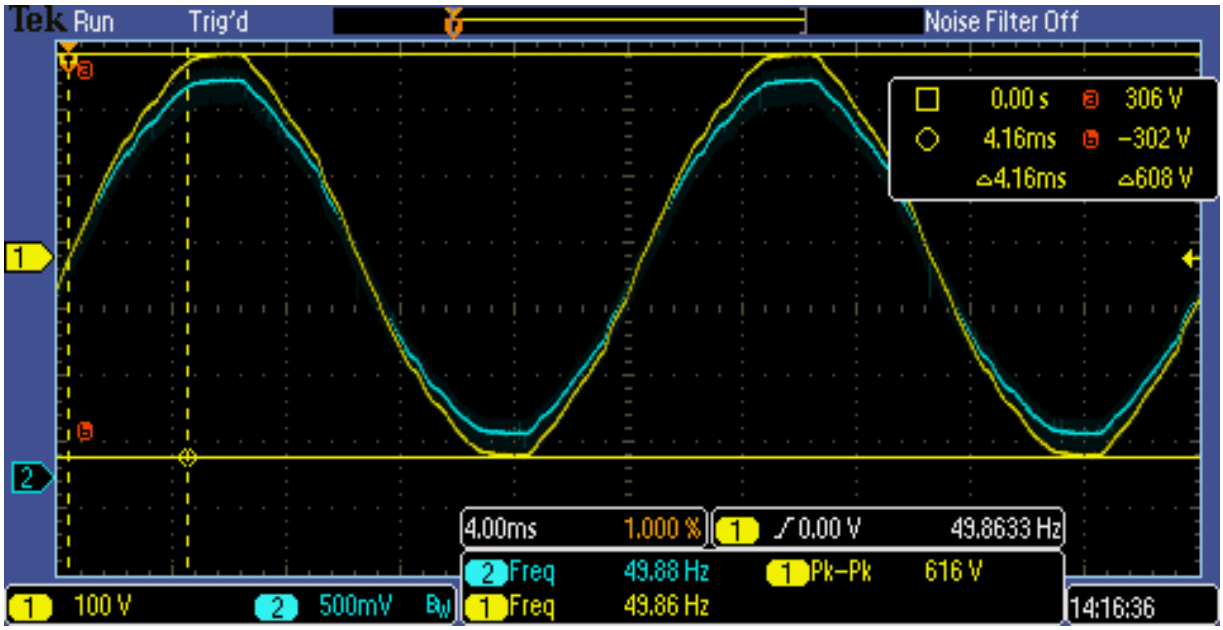

Figure 26. Oscilloscope capture of line input signal waveforms.

Note that the 'real' mains waveform shows significant peak flattening and evidence of the presence of higher frequency harmonics, 1 kHz being one. The ideal calibration coefficients to be used give a voltage at the A–D input of 3779 counts at +350 V and 317 for −350 V. However, it was found that this produced a calculated magnitude of approximately 1.7% too high. Examination showed that the A–D reference was 3.25 V, not 3.3 V, which accounts for 1.5% of the error. As the tolerance of the components used in the divider circuits is of 1% tolerance, this can easily account for the rest. When all three-phase lines are fed from a single supply, the measured amplitudes are

within 1 Vrms of each other. It was also noted that there was approximately 2 V of noise on all three amplitude measurement channels. This was traced to the ribbon cable connecting the board that takes the three phases in and contains the 4 MΩ resistors to the line interface board, picking up noise from the adjacent microcontroller board. Repositioning of the resistor board and the cable eliminated this noise giving stable readings. The *TVE* is given by [4] as

$$TVE = \sqrt{\frac{(\hat{X}_r(n) - X_r(n))^2 + (\hat{X}_i(n) - X_r(n))^2}{(X_r(n))^2 + (X_i(n))^2}} \quad (6)$$

where $\hat{X}_r(n)$ and $\hat{X}_i(n)$ are the real and imaginary vectors as measured by the PMU, and $X_r(n)$ and $X_i(n)$ are the ideal vectors. For the *TVE* to be less than 1%, given a 0.5% error as calculated for the interface phase offset, the in-phase amplitude error needs to be less than 0.5%, i.e., 1.2 Vrms for a 240 Vrms supply. After adjusting the calibration constants, the voltage measured by the PMU was within 1.0 V of that measured by a Keithly 2000 DMM.

9.3. Line Frequency Interface Circuit

The circuit shown in Figure 9 provides the correct drive to the three-line frequency interrupt pins, as simulated in Figure 11. When initially tested, there was significant jitter on the rising edge, which sometimes led to multiple interrupt triggers and significant frequency reading errors (100 Hz instead of 50 Hz). This was traced to interference being picked up by the ribbon cable connecting the resistor board to the line interface board from the phasor processing module and the 5 V power supply. Rerouting this ribbon cable as was required for the line voltage interface circuit was also eliminated. When tested in a domestic environment, 'clean' frequency readings were obtained with little noise. However, when tested in an industrial environment, where a three-phase supply is available, bursts of noise were intermittently visible on the frequency display. On close examination of the position of the interrupt rising edge about the input waveform, it could be seen that the high-frequency harmonics present in the input waveform shown in Figure 27 were varying cycle by cycle leading to jitter on the output of the zero-crossing detector. At this point, the implementation of a frequency detector using the rate of change of phase from the output of the phase detector already designed was investigated. The additional code was written and worked correctly, but the phase disturbance in the 3.200 kHz pulse noted during testing of the PPS acquisition module led to a disturbance in the frequency measurement once per second. On close examination, a small phase fluctuation could be seen at that point, but normally this is not noticeable, but the effect on the time derivative is much more significant. Although the use of the rate of change of phase would have allowed the removal of the line frequency measurement circuitry, it was decided to revert to the zero-crossing-based frequency measurement circuit and add capacitors C5, C6, and C7 as shown in Figure 9. This has the effect of delaying the zero-crossing interrupt until the input waveform is at 177 V rising, as shown in Figure 27. However, as only a zero-crossing period is required, this does not affect the operation of the frequency detector, apart from reducing the effect of higher harmonics.

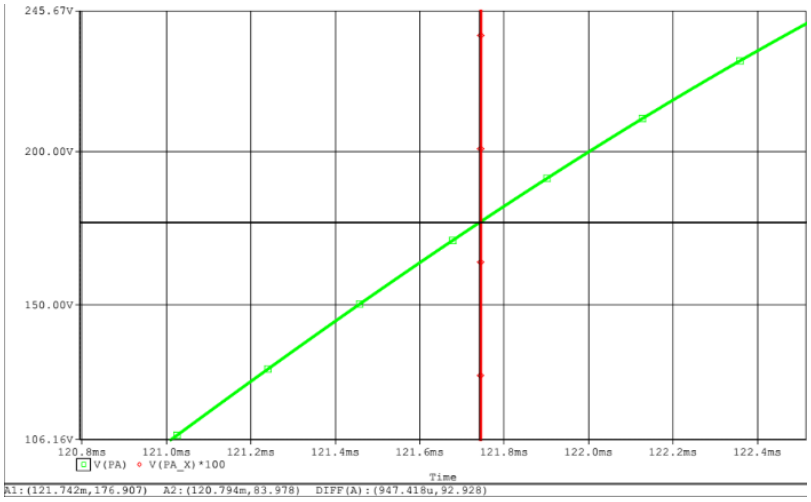

Figure 27. Modified line frequency circuit simulation.

This section may be divided into subheadings. It should provide a concise and precise description of the experimental results, their interpretation, and the experimental conclusions that can be drawn.

9.4. Display Module

The original display module drove a 7" 800 × 480-pixel UTFT display using an Arduino Due module. Due to procurement and delivery issues, only a 3.5" 400 × 240-pixel module was available, which used an incompatible driver and hardware interface. This meant that significant modification was required to the code to drive the display and format the output correctly, and also it became necessary to change the processor from an Arduino Due to an Arduino Mega2560 as the display drivers caused compiler errors for the Due target. When this was initially done, the display processor could not correctly receive the communications from the communications controller. It was initially thought that this was due to the different I/O voltages used—the Arduino Due uses 3.3 V logic, and the Arduino Mega2560 uses 5 V logic, and a level translator circuit was designed and built, but before it was tested, it was realised that due to the way the UART controller on the Arduino Due works, setting the data rate to 230,400 baud actually sets it to approximately 250,000 baud. When the display processor was set to 250,000 baud, communications were re-established. As communications between the communications controller and the display controller are only one way, there was no need to connect the 'return' data path, which would need a level shifter to avoid applying an overvoltage to the communications controllers receiving input.

9.5. External Display Unit

The baseline PMU also interfaced with an external PC running a Python script that decoded the data stream from the communications processor and provided a real-time graphical display of the phase data. The original implementation displayed only phase and amplitude. This was also modified to display frequency and ROCOF. A screenshot of a typical display is shown in Figure 28.

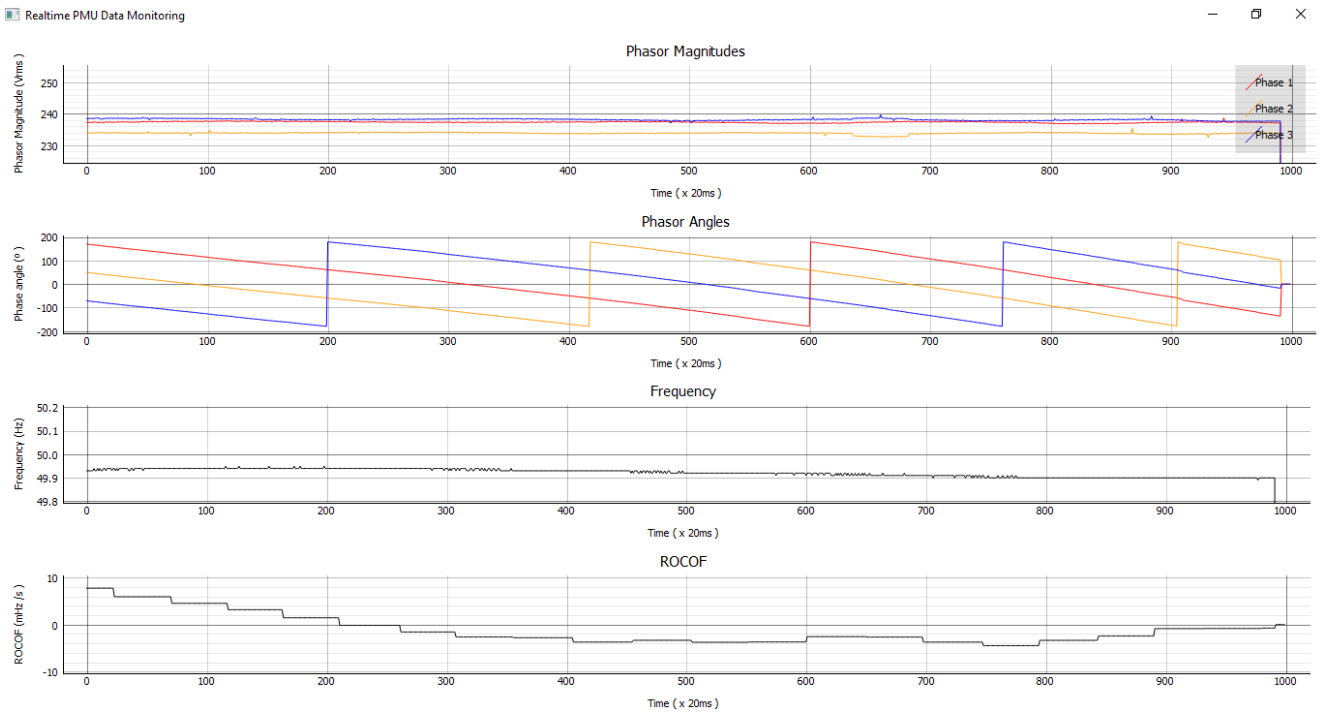

Figure 28. External monitoring display.

9.6. Overall PMU

When tested, the individual PMU modules were mounted in a box using appropriate connectors, indicators, and mounting methods to ensure that all external interface connections were safely isolated from the mains power input and the three-phase measurement input. The completed PMU is shown in Figure 29. The overall power consumption of the PMU was measured as 0.7 A at 5.0 V. The performance of the new interfaces, as previously described, is as expected, and the reported phase–

amplitude readings are within the measurement tolerance of the internal reference and component spreads. Thus, the modified PMU has, at the minimum, the same behaviour and response as the baseline, and due to the increased bandwidth of the line interface, it should be capable of meeting the requirements of IEEE C37.118 if the code running on the processors is modified to implement this.

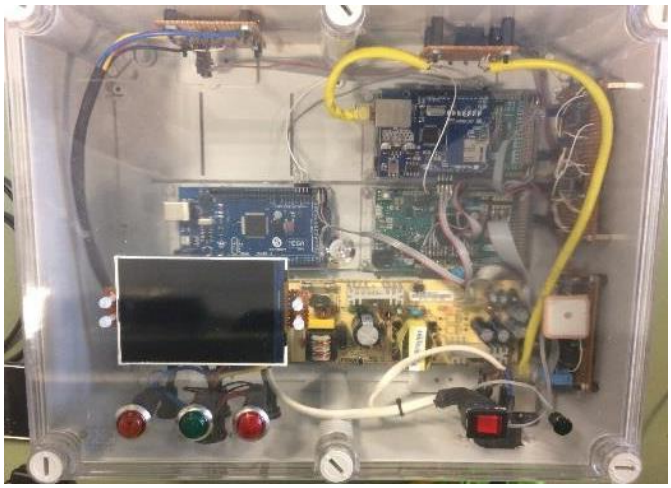

Figure 29. Full implementation of the proposed **PMU**: PhasorsCatchers.

10. Further Work

The baseline PMU used was chosen for ease of implementation, not because it was the most appropriate for a compliant unit. There is considerable expense still in the three Arduino modules that make up the processors. A more appropriate solution would be an implementation using a single-board microcontroller in the same way as the OpenPMU V2 [24], which uses the BeagleBone Black module, which reduces the overall cost. Or the use of more advanced iDFT algorithms using modules, such as the Xilinx Zynq 7020 System-on-Chip, as demonstrated by [12], which may give better performance.

If the current module configuration is used, the 3.2 kHz oscillator requires further work to turn it into a PLL-based source that will eliminate the phase disturbance when the PPS signal is received, allowing the frequency measurement to be derived from the rate of change of phase, reducing cost more and improving phase measurement accuracy. The output to the external data interface should be converted to one that is compliant with [6], and the ROCOF determination code should also be rewritten to be compliant with [4]. The GPS module should be replaced with one that can provide the time quality indicator required by [15].

It may be practical to replace the Arduino Mega2560 module used by the display controller module with an Arduino UNO, which will save approximately an extra GBP 20.

Although there is generally no difficulty in finding satellite signals for the GPS receiver, the addition of an external antenna interface could be helpful. If a passive antenna is used, all that would be required is an interface isolated by suitable capacitors. If, however, an active antenna is used, a source of power (3.3 V or 5 V) on the 'safe' side of the isolation barrier is required, which is not presently available. This could easily be obtained from a simple power supply such as those used to charge mobile phones at little cost and could be contained within the PMU unit itself.

Using a precision external reference voltage for the ADC in the phasor controller module would remove the uncertainty in the internal reference voltage used, removing the need for individual unit calibration and potentially improving long-term stability.

Other refinements include the use of storage within the module (the communications controller module already provides for a memory card interface, but presently there is no code written to implement this) and the provision of battery backup. This could be achieved by using a power supply with a temperature-compensated voltage of 6.8 V and a current limit appropriate for the battery used charging a lead–acid battery, with the output of the battery being fed into a buck converter with an output of 5.0 V and employing undervoltage protection to avoid excessive battery discharge. Even a small-sized battery could give more than two hours of operation if the mains power supply fails.

11. Conclusions

The aim of this project was to design and evaluate lower-cost methods of interfacing a PMU with external voltage lines to be measured and to the external communications interface, and the results from the simulation and testing of the line interface circuit show that it is capable of fulfilling the required function without introducing an unacceptable level of phase or voltage distortion. The moving of the safety isolation barrier to the communications interface from the signal acquisition interface does not change the operation of the unit and removes considerable cost. Whilst tested with the specific A–D converter interface used on the Arduino modules, which require a 0 V to 3.3 V input, this interface can easily be scaled to provide other output voltage ranges by the use of appropriate supply voltage and scaling resistors and could be incorporated into a bespoke interface for any data acquisition system.

Author Contributions: Conceptualisation, D.S., F.G.-L. and D.M.; methodology, D.S.; software, D.S. and D.M.; validation, D.S., D.M. and J.M.R.-F.; investigation, D.S. and J.M.R.-F.; writing—review and editing, D.S., H.R.C., D.M., J.M.R.-F., K.A. and F.G.-L.; supervision, F.G.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chakrabortty, A. Wide-Area Control of Power Systems: Employing Data-Driven, Hierarchical Reinforcement Learning. *IEEE Electrif. Mag.* **2021**, *9*, 45–52. [CrossRef]
2. Liang, X.; Chen, X.; Ding, H.; Zhang, G.; Zhen, N. Research on Stability of PμWAMS Service Performance Index Based on Wide Area Control. In Proceedings of the 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC, Xi'an, China, 25–27 May 2018; pp. 2014–2018. [CrossRef]
3. Qin, C.; Greig-Prine, J.; Nie, Z.; Banerjee, P.; Srivastava, A.K. Remote PMU Testing using Low-cost FPGA Platform and PPA following IEEE TSS. In Proceedings of the 2019 IEEE Industry Applications Society Annual Meeting, IAS, Baltimore, MD, USA, 29 September–3 October 2019. [CrossRef]
4. Sobrinho, A.S.F.; Flauzino, R.A.; Liboni, L.H.B.; Costa, E.C.M.; Spatti, D.H. A Fuzzy Inference System for Disturbance Classification in Power Distribution Networks Using a Low-Cost PMU. In Proceedings of the 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Rio de Janeiro, Brazil, 8–13 July 2018. Available online: https://ieeexplore.ieee.org/document/8491446/ (accessed on 28 March 2022).
5. Vanfretti, L.; Chow, J.H. Synchrophasor Data Applications for Wide-Area Systems. In Proceedings of the 17th Power Systems Computation Conference (PSCC), Stockholm, Sweden, 22–26 August 2011.
6. Mohapatra, D. Development and Hardware Implementation of a Phasor Measurement Unit Using Microcontroller. NIT Rourkela. 2015. Available online: http://ethesis.nitrkl.ac.in/7621/1/2015_MT_Development_Debasis_mohapatra.pdf (accessed on 20 March 2022).
7. Schofield, D.; Gonzalez-Longatt, F.; Bogdanov, D. Design and Implementation of a Low-Cost Phasor Measurement Unit: A Comprehensive Review. In Proceedings of the 2018 Seventh Balkan Conference on Lighting (BalkanLight), Varna, Bulgaria, 20–22 September 2018; pp. 1–6. [CrossRef]
8. PhasorsCatchers: We Are Creating More than a PMU ... Next Generation of ... More to Come. Available online: http://phasorscatcher.fglongatt.org/index.html (accessed on 12 August 2019).
9. Fglongatt (Francisco Gonzalez-Longatt) GitHub. Available online: https://github.com/fglongatt (accessed on 20 August 2021).
10. Seger, P.V.H.; Moreto, M.; Grando, F.L.; Lazzaretti, A.E.; Denardin, G.W.; Pastro, C.R. Monitoring Low Voltage Electrical Networks Using Low Cost PMUs: A Case Study. In Proceedings of the 2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe, EEEIC/I and CPS Europe, Madrid, Spain, 9–12 June 2020. [CrossRef]
11. Cetina, R.Q.; Roscoe, A.J.; Atoche, A.C. Low-cost power systems metrology laboratory based on raspberry Pi. In Proceedings of the 2018 1st International Colloquium on Smart Grid Metrology, SmaGriMet, Split, Croatia, 24–27 April 2018; pp. 1–2. [CrossRef]
12. Romano, P.; Paolone, M.; Chau, T.; Jeppesen, B.; Ahmed, E. A high-performance, low-cost PMU prototype for distribution networks based on FPGA. In Proceedings of the 2017 IEEE Manchester PowerTech, Manchester, UK, 18–22 June 2017. [CrossRef]
13. Femine, A.D.; Gallo, D.; Landi, C.; Luiso, M. A design approach for a low cost phasor measurement unit. In Proceedings of the I2MTC 2019—2019 IEEE International Instrumentation and Measurement Technology Conference, Auckland, New Zealand, 20–23 May 2019. [CrossRef]
14. Du, L.; Huang, J.K.; Liu, Q.Y. A realisation of measurement unit for phasor measurement unit based on DSP. In Proceedings of the Asia-Pacific Power and Energy Engineering Conference, APPEEC, Shanghai, China, 27–29 March 2012. [CrossRef]
15. *IEEE Std C37.118.1-2011*; IEEE Standard for Synchrophasor Measurements for Power Systems. IEEE Power & Energy Society: Piscataway, NJ, USA, 2011. [CrossRef]

16. Angioni, A.; Lipari, G.; Pau, M.; Ponci, F.; Monti, A. A Low Cost PMU to Monitor Distribution Grids. In Proceedings of the AMPS 2017—IEEE International Workshop on Applied Measurements for Power Systems, Liverpool, UK, 20–22 September 2017. [CrossRef]
17. Garcia-Valle, R.; Yang, G.Y.; Martin, K.E.; Nielsen, A.H.; Stergaard, J. DTU PMU laboratory development—Testing and Validation. In Proceedings of the IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT Europe, Gothenburg, Sweden, 11–13 October 2010; pp. 1–6. [CrossRef]
18. Sampson, O.V. Construction of a Phasor Measurement Unit (PMU) for Power System Applications. Ph.D. Thesis, University of Manitoba, Winnipeg, MB, Canada, 2015.
19. OpenPMU: The Open-Source Phasor Measurement Unit-Solutions-National Instruments. Available online: http://sine.ni.com/cs/app/doc/p/id/cs-14787#prettyPhoto[gallery]/0/ (accessed on 4 September 2018).
20. OpenPMU. Available online: https://sites.google.com/site/openpmu/ (accessed on 4 September 2018).
21. Open Source SynchroPhasor PMU-CodePlex Archive. Available online: https://archive.codeplex.com/?p=gridtrak (accessed on 4 September 2018).
22. Highlights-Power Standards Lab. Available online: https://www.powerstandards.com/product/micropmu/highlights/ (accessed on 4 September 2018).
23. Luigi Vanfretti-Associate Professor-Rensselaer Polytechnic Institute | LinkedIn. Available online: https://www.linkedin.com/in/vanfretti (accessed on 4 September 2018).
24. Laverty, D.M.; Vanfretti, L.; Best, R.J.; Morrow, D.J.; Nordstrom, L.; Chenine, M. OpenPMU technology platform for Synchrophasor research applications. In Proceedings of the IEEE Power and Energy Society General Meeting, San Diego, CA, USA, 22–26 July 2012. [CrossRef]
25. Zhao, X.; Laverty, D.M.; McKernan, A.; Morrow, D.J.; McLaughlin, K.; Sezer, S. GPS-Disciplined Analog-to-Digital Converter for Phasor Measurement Applications. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 2349–2357. [CrossRef]
26. Texas Instruments. LMx39x, LM2901xx Quad Differential Comparators. Available online: https://www.ti.com/lit/ds/symlink/lm2901.pdf (accessed on 20 August 2021).
27. Electronic Circuit Optimization & Simulation | Cadence PSpice | PSpice. Available online: https://www.pspice.com/ (accessed on 20 August 2021).