Design of an Inexpensive Residential Phasor Measurement Unit

Jeremy Murray Vick

ECE 499, Electrical Engineering Capstone
Department of Electrical and Computer Engineering
Union College
Schenectady, NY

Advisor: Professor Luke Dosiek

March 17, 2015

Abstract

Phasor measurement units, (PMUs) are widely used by power companied to measure the state of transmission lines and the quality of transmitted power. The goal of this project is to design a low cost PMU that takes measurements at the residential level of the power grid. This device must be easy to manufacture and highly reliable. It will communicate results back to a central database via the internet. Compliance with IEEE Standard C37.118.1 and C37.118.2 is required. The widespread introduction of an inexpensive PMU will increase the data resolution available in Wide Area Monitoring Systems (WAMS), providing control room operators with a more accurate picture of the state of the power grid.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

On August 14, 2003, North America suffered its largest blackout. Major 345 kV transmission lines dropped out of service, unbeknownst to operators, causing a cascading outage that extended across the Midwest, Northeast, and into Canada [1]. An investigation launched by the North American Electric Reliability Corporation (NERC) found that the blackout could have been confined to a small region had operators known the status of overstressed and failing lines [2].

Since this catastrophe, steps have been taken to improve real-time, networked monitoring of America's electrical transmission and distribution network, in order to enable system operators to predict and counteract or confine disturbances. Increased situational awareness can also allow the dynamic calculation of maximum load ratings based on environmental conditions. Overall, improved monitoring allows utilities to provide power to customers in a more efficient, more reliable, and safer way.

The installation of phasor measurement units (PMUs) provides a real time image of operating conditions. Increasing the number of PMUs improves the resolution of data available to control room operators. It also creates the possibility for implementation of automatic control systems to correct disturbances or failures. However, these devices are costly, approximately $43,400 per installation, and are hard to install [3]. They also require dedicated communication networks to feed data back to centralized processors, known as phasor data concentrators (PDCs).

PMUs can offer a new insight when installed at the distribution level of the power grid. The prevalence of distributed generation, smaller power plants that supply communities rather than regions, is increasing due to the fact that renewable power generation is better suited for communities. This increase causes an increase in dynamic events at the distribution level, as wind turbines and solar farms increase and decrease their output in step with the weather. Having PMUs measuring at the distribution level will give a more accurate

picture of how the increase in distributed generation affects the power grid on a day-to-day and long term basis.

1.2 Objective

The goal of this project is to design a low cost PMU that takes measurements at the residential level of the power grid. This device should be easy to manufacture and highly reliable. It should communicate results back to a central database using the protocol described in the IEEE Standard for Syncrophasor Data Transmission.

Chapter 2

Background

2.1 Synchrophasor Definition

Alternating Current (AC) is mathematically represented by a cosine wave,

$$x = A\cos(2\pi f_{AC}t + \phi) \tag{2.1}$$

where $f_{AC} = 60Hz$ in North America. Using a technique proposed by Charles Proteus Steinmetz in [4], AC can be represented as a simplified quantity called a phasor. When representing a cosine as a phasor, it is assumed that the frequency of the signal remains the same. Therefore, the variable quantities are magnitude and phase. For AC, magnitude is commonly defined as the root mean square of voltage. Equation (2.1) becomes

$$X = \frac{A}{\sqrt{2}}\angle\phi \tag{2.2}$$

Establishing phase requires either a signal or time reference. Synchrophasors calculate phase using an absolute time reference, commonly Coordinated Universal Time (UTC). Figure 2.1 shows a cosine superimposed on a UTC time pulse. The synchrophasor is defined to be 0° if the cosine has a maximum during the pulse and 90° if the cosine has a zero crossing at the pulse. Values between 0° and 90° are calculated according to the selected phasor estimation algorithm [5]. Previously, phasor measurement at generators and nodes in the transmission network was impractical due to geographic separation between the two. Implementation of synchrophasors allows for easy calculation of magnitude and phase differences between nodes based off a shared standard time.

2.2 Previous Work

The concept of a synchrophasor was first introduced in the 1980s and has since generated a large body of commercial and academic research. It is impossible to address all work

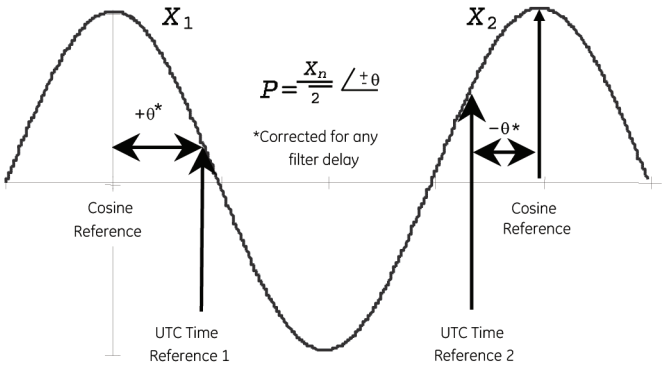

Figure 2.1: Phase calculation based on UTC reference. Source: [6]

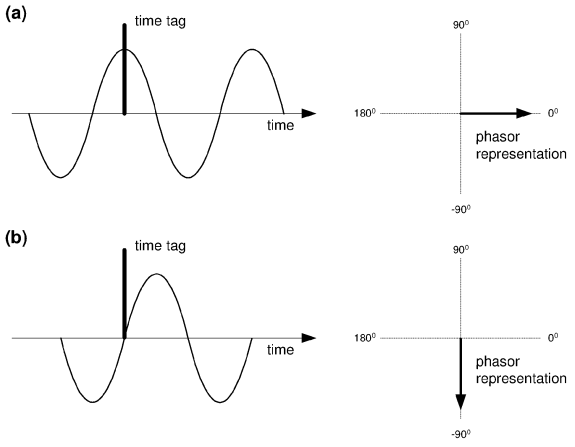

Figure 2.2: Angle convention for synchrophasors. Source: [7]

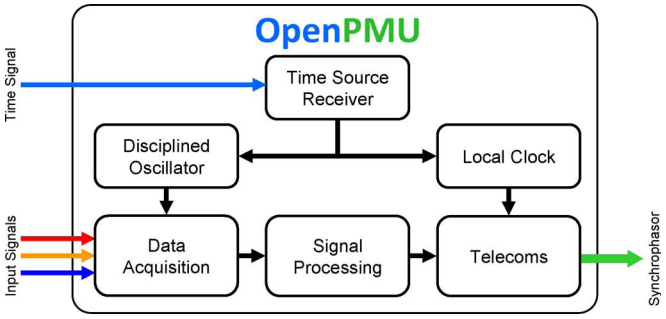

Figure 2.3: OpenPMU block diagram. Source: [10]

on synchrophasors and their applications in the scope of this project so emphasis will be placed on development of *inexpensive* PMUs.

K. Kirihara, B. Pinte, and A. Yoon designed and tested a relatively low cost (approximately $1050) PMU as part of an undergraduate senior project described in [8]. Their project utilized a National Instruments sbRIO for digital filtering and calculation of synchrophasors. Global Positioning System (GPS) was used to generate the time reference. The project was able to successfully measure phasors, but utilized only the National Electrical Code residential voltage standards to test the PMU, ignoring IEEE C37.118.1. The group also did not address the transmission of synchrophasors to a centralized server or phasor data concentrator (PDC).

In Brian Miller's Masters thesis [9], alternatives for conventional current transducers are considered. Miller also examines the use of wireless networks for time synchronization under the IEEE 1588 standard. Use of wireless networks is found to provide a viable alternative to GPS synchronization, useful in areas where signal strength is diminished. It would also provide a cost reduction due to the elimination of the GPS module. These proposed changes were found to be viable improvements while remaining compliant with the IEEE C37.118.1 standard.

In order to lessen the restrictions of proprietary hardware and algorithms on the progress of PMU development, the OpenPMU [10] group was formed, dedicated to designing "open source platform for synchrophasor applications and research." The group utilizes a standard data acquisition device (DAQ) from National Instruments and a GPS receiver from Garmin as the basis for the OpenPMU. A PIC from Microchip synchronizes the DAQ to the GPS timecode. The OpenPMU uses the Python scripting language running on Microsoft Windows. It is currently able to measure synchrophasors, but has yet to achieve full compliance with IEEE C37.118.1.

Chapter 3

Design Requirements

3.1 Performance

A vast majority of the performance requirements for this project are drawn from the IEEE Standard for Synchrophasor Measurements for Power Systems [11], its 2014 amendment [12] and the IEEE Standard for Synchrophasor Data Transfer for Power Systems [13]. Two classes of performance are laid out in the standards: P, for fast response with no explicit filtering and M, for analytic measurements sttausceptible to aliasing. Adherence to these standards will ensure that the device is compatible with existing phasor data concentrators (PDCs) and visualization software.

The device is broken down into seven different component parts as shown in Figure 3.1. The measurement source is a 120v residential outlet. A step down circuit lowers the voltage of the measurement source into the range of the A/D converter. This circuit will also provide DC power for the device itself. An analog anti-aliasing filter will be used to limit the signal bandwidth before sampling. The signal passes through an A/D converter that samples in synchronicity with the time source. The time source provides an absolute time reference to the A/D converter and the Synchrophasor Estimator. The Synchrophasor Estimator will calculate the magnitude of digital signal and run it through a phase estimation algorithm (PEA). The resulting magnitude and phase estimation will be given a time tag and sent to a PDC via the internet interface. The device must also accept commands transmitted by the PDC.

3.1.1 Step Down and Device Power

Analog to Digital (A/D) converters are not typically capable of measuring signals at 120v, meaning a voltage step down circuit must be designed to reduce the magnitude of the AC signal to match the specified range of the A/D. The device may only have one connection to the power source, meaning the step down circuit must also include a tap and rectification circuit to provide power for the chosen processor. The supply circuit should have over-

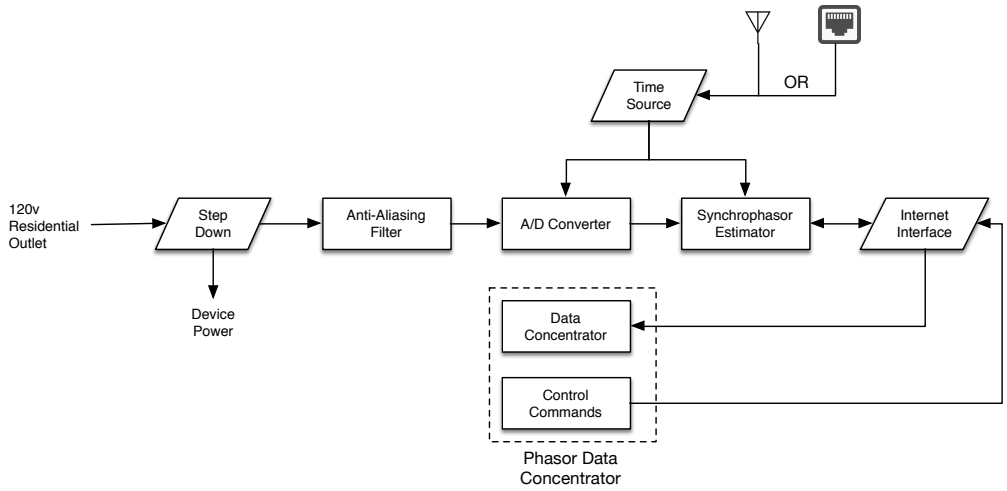

Figure 3.1: rPMU block diagram.

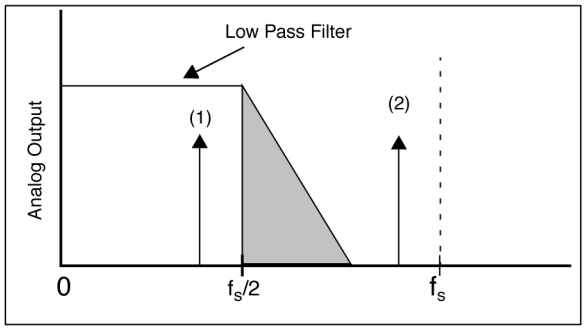

Figure 3.2: Anti-aliasing filter frequency response. Source: [14]

voltage protection to prevent damage to the device and have an output voltage ripple that meets the constraints of the chosen processor.

3.1.2 Analog Filtering

Since an A/D conversion is being performed, it necessary to have an analog low-pass filter to reduce the bandwidth of the input signal and eliminate aliasing. The cutoff frequency for the low-pass filter should be just above $f_s/2$, the chosen sampling frequency. The desired frequency response, defined in terms of the sampling frequency, is shown in Figure 3.2.

System Frequency	50 Hz		60 Hz				
Reporting Rates (frames per second)	10	25	10	12	15	20	30

Table 3.1: Required synchrophasor reporting rates. Source: [13]

3.1.3 Timing

Synchrophasors must, by definition, be recorded with respect to an absolute time reference. The absolute reference used by IEEE C37.118.1 is Coordinated Universal Time (UTC). UTC can be obtained from either a GPS receiver or through the internet based Precision Time Protocol (PTP) [15]. The time must be accurate within $\pm 26\mu$s according to the standard. Receiving UTC via the internet is more practical for the scope of this project, as GPS signals can be very weak indoors. However, testing is needed to confirm that synchronization with internet time servers can be achieved with sufficient accuracy.

Each synchrophasor must be given a time tag according to Coordinated Universal Time (UTC). The time tag consists of three numbers: a System On a Chip (SOC) count, a fraction-of-second count, and a time status value. SOC is specified as a 4-byte binary count of the number of seconds since the Unix epoch, 00:00 January 1, 1970. Occasionally, a leap second must be inserted to keep SOC synchronized with UTC, which is specified using a special case of the fraction-of-second as specified in section 4.3 of [11]. Time status indicates the reliability of the clock, which can become unsynchronized due to loss of signal. Values for time status are specified in Table 6 of [11].

3.1.4 Measurement

Synchrophasor measurements must be synchronized with the time code source so they can be time-aligned with measurements from other PMUs by a PDC. Reporting rates are also defined in IEEE C37.118.1 to ensure that multiple PMUs will take measurements at the same rate.

Reporting Rate

The reporting rate, measured in phasors per second, must be a factor of the nominal system frequency. Required rated are listed in Table 3.1. The reporting rate must be selectable by the user via the device's internet interface according to the protocol defined in [13].

Phase Estimation

There are two categories of phase estimation algorithms (PEAs): time domain and frequency domain. An example of a time domain PEA is the Weighted Least Squares method. WLS uses a Taylor series expansion of the signal to determine the phase. In [16], variation

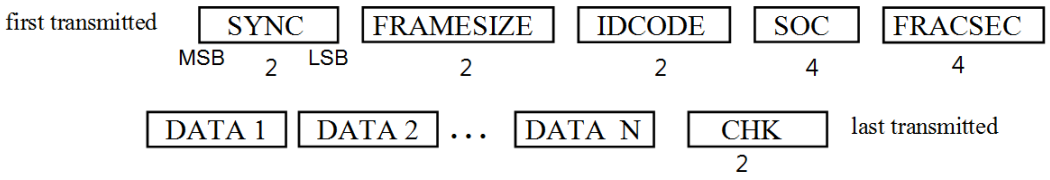

Figure 3.3: Data frame transmission order. Source: [13]

of the number of terms in the series is studied in an attempt to reduce error. The Interpolated Discrete Fourier Transform (IpDFT) is an example of a frequency domain algorithm. The IpDFT is significantly faster than WLS, but does not perform as well when disturbances occur. A thorough comparison between WLS and IpDFT is carried out in [16]. The selection of the PEA will provide the constraints for selection of a processor.

Total Vector Error

Total Vector Error (TVE) is a measurement of the difference between a perfect theoretical phasor and the actual phasor measured by the PMU. The IEEE Std. C37.118.1 defines TVE as:

$$TVE(n) = \sqrt{\frac{(\hat{X}_r(n) - X_r(n))^2 + (\hat{X}_i(n) - X_i(n))^2}{X_r(n)^2 + X_i(n)^2}} \tag{3.1}$$

Where $\hat{X}_r(n)$ and $\hat{X}_i(n)$ are the real and imaginary components, respectively, of the measured phasor and $X_r(n)$ and $X_i(n)$ are the components of the theoretical phasor. The standard specified that TVE must be less than 1%. Sources of TVE include timing inaccuracy, off-nominal signal frequency, and low frequency oscillations.

3.1.5 Communication

Communication between the PMU and PDC will take place via the internet. Data packets will be sent using Transmission Control Protocol (TCP). Data packets are subdivided into frames, each containing a specific piece of data. The frames required for sending phasor data, as defined in IEEE C37.118.2, to a PDC are listed in Table 3.2. The phasor itself is transmitted in frame 7. The DIGITAL frame can be used to transmit extra device status indicators not included in the STAT frame, relay statuses, breaker statuses or other information. The generic order of frame transmission is shown in Figure 3.3, where DATA1, DATA2, etc. are frames 7-11 from Table 3.2

No.	Field	Size (bytes)	Description
1	SYNC	2	Sync byte followed by frame type and version number.
2	FRAMESIZE	2	Number of bytes in frame.
3	IDCODE	2	PMU ID number.
4	SOC	4	Second Of Century time stamp.
5	FRACSEC	4	Fraction of Second and Time Quality.
6	STAT	2	PMU status flags.
7	PHASOR	4	Phasor estimate. May be single phase or 3-phase positive, negative, or zero sequence.
8	FREQ	2/4	Frequency.
9	DFREQ	2/4	Rate Of Change Of Frequency.
10	ANALOG	2+	Analog data, available for extra features.
11	DIGITAL	2+	Digital data, available for extra features.
12	CHK	2	Cyclic redundancy check (CRC-CCITT)

Table 3.2: Data frame organization. Source: [13]

3.2 Safety

The device must comply with the National Electric Code regulations for connection spacing and insulation for 120v connections [17]. The connection to the wall outlet should be made with a NEMA 5-15 compliant connector as it is the most common outlet found in residences. The connector is rated for a maximum voltage of 125v, sufficient for the requirements of this project.

3.3 Cost

Commercial PMUs cost an average of $43,400 per installation [3]. This device will be installed en masse in residences and should have a cost commensurate with mass production. The target cost for this project is under $1,000.

Section	**Comments**
Step down	Step down 120v measurement source to acceptable range for A/D converter
Device power	Determined by the choice of processor
Analog filtering	Low pass filter with f_c just above $f_s/2$
Timing	Either internet or GPS based
Phase estimation	Either WLS or IpDFT
Safety	Must follow all wiring and spacing regulations for 120v
Cost	Target cost is under $1000

Table 3.3: Summary of design requirements

Chapter 4

Design Alternatives

4.1 Component Selection

4.1.1 Computing Platform

The computing platform is the core of the phasor measurement unit. It is responsible for acquiring raw AC voltage waveform data from an Analog to Digital Converter (ADC) in synchronicity with the GPS Pulse Per Second (PPS) time code, computing the magnitude and phase of the signal, packaging the measured data into the IEEE C37.118.2 transmission format and sending the resulting data packet over the internet to a PDC. Many options were considered in the choice of the computing platform for this project, including the well-known Raspberry PI, the Arduino, BeagleBone Black, and Intel Edison. The Raspberry PI, while it is a relatively powerful platform with thorough documentation and an active user base, was dismissed due to the lack of an onboard ADC. Choosing a platform with an onboard ADC is important because it simplifies the circuitry and reduces the cost of the device. An Arduino, while it has an onboard ADC, lacks the computing power of the other SOC based alternatives, requires additional components to connect to the internet, and does not have the ability to be reprogrammed remotely, an important consideration when deploying a device in the homes of laymen residents.

Intel's Edison platform was considered for its high computing power density (dual core 500 MHz processor), but rejected due to the scarcity of public documentation. Ultimately, the BeagleBone Black was chosen as the computing platform. It has a 1 GHz processor, which outperforms the Raspberry PI's 700 MHz, a built in ethernet port for internet connection, and an onboard ADC with eight input channels. The Black also can run the Debian or Ubuntu Linux distributions. Using these Linux distributions provides built in support for remotely accessing the device and a large package database to pull from when implementing components of the project. Of the most consequence in choosing this board was the NEON and Programmable Realtime Unit (PRU) subsystems. The NEON subsystem provides hardware acceleration for floating point calculations and a implementation of

the Fast Fourier Transform that utilizes this capability has already been developed. Utilizing this library will allow for the reduction of the computational load on the processor, which in turn should enable the device to achieve a higher reporting rate. The PRU, essentially an onboard microcontroller in which the execution of each instruction is fixe is significant because all of the instructions available in this subsystem have a fixed execution time of 5ns. The PRU interfaces directly with the ADC subsystem and the Black's onboard memory, meaning it can acquire data from both the GPS and AC voltage inputs to the ADC and store it for processing in a fixed, known amount of time that can be easily compensated for in the final calibration of the device.

4.1.2 GPS Module

As stated in the Timing section of the Design Requirements, the time source must be accurate within $\pm 26\mu$s in order to achieve the accuracy specified in [11]. There are many timing-specific GPS modules on the market, but their average price is $450, [18] which is prohibitive for the budget of this project. The Adafruit Ultimate GPS Module is offered at a reasonable $40 and achieves $\pm 9\mu$s accuracy [19] on its PPS output. Though this is not as accurate as timing-specific sources, some boast sub $\pm 5\mu$s accuracy, [20] it is well within the specifications and budget for this device.

4.2 Software Design

4.2.1 Signal Processing

From the initial stages of the project, Python was the desired programming language for processing raw data into synchrophasor measurements. Availability of Python packages for signal processing and ethernet packet transmission, cross platform compatibility, and the ease with which the language can be interpreted by a lay person were the driving factors in this choice. By choosing such a widely known and supported language the code generated in this project can be of greater utility to others researching PMUs. However, Python is a high-level programming language which presents a few challenges when interacting directly with hardware. Python code has to be parsed by the Python interpreter before it is executed, exacting a performance penalty. In addition, low-level programming languages like C are more suited to direct memory interaction than Python.

It was possible to use Python, despite its deficiencies, in this project because the heavy lifting is handled by the PRU. Samples generated by the ADC are stored in shared memory accessible from the CPU by the PRU, which ensures that latency between sample acquisition and storage in memory is a fixed, known value. Texas Instruments, the designer of the AM335x processor onboard the BeagleBone Black, provides a library for sending assembly code to and monitoring interrupts from the PRU written in C, another obstacle. Fortunately, PyPRUSS, a community project focused on 3D printing with the Black [21],

incorporates a Python wrapper for the C library, enabling its functions to be called within Python code.

4.2.2 Signal Acquisition

Acquisition of data with the BeagleBone Black's onboard ADC can either be handled by the CPU or by the PRU. The host CPU runs Debian, a variant of linux, which is not a realtime operating system. Any processes interacting with the ADC are scheduled at the mercy of the operating system, which is not a desirable characteristic in an application where precise timing is of the utmost importance. Therefore, the PRU was used to communicate with the ADC subsystem. The PRU is controlled with assembly code loaded by the CPU. Coding in assembly presents a few challenges; operations involve direct manipulation of registers and memory locations which is time consuming and potentially catastrophic as the PRU has access to the memory and storage used by the operating system. Ultimately, the tight timing constraints imposed by the design requirements necessitate the use of the PRU despite the faults of coding in assembly.

Chapter 5

Preliminary Proposed Design

5.1 Hardware Design

5.1.1 Voltage Step Down

Connection to the wall will be made with a NEMA 5-15 compliant connector, the common standard for residential outlets [17]. The ADC input range of the BeagleBone Black is 0 to 1.8v [22], but going directly to this range would require a transformer ratio of 66.7:1 or greater, something that is not commonly found in 120v transformer offerings. A 14:1 transformer was chosen instead because of its availability and price. This transformer yields a 8.57v peak-to-peak output when connected to the 120v wall outlet. The voltage is further reduced into the ADC range by attenuation in the low pass filtering circuit.

Input Protection

Although 120v is not as dangerous as the high voltage that commercial PMUs measure, it is important that this device have safety features to protect both the low voltage electronics and the end user. A 0.5A fast-acting fuse was placed between the hot wire of the plug and the device as the power supply for chosen for the BeagleBone Black has a maximum current draw of 0.3A. The digital (DGND), analog (GND_ADC), and earth grounds as well as the neutral wire are all tied together to ensure that there are no ground loops that might affect measurement, but also to ensure that any shorts or loose wires will not generate unexpected voltages throughout the circuit.

DC Bias

It is also necessary to add a DC bias to the AC signal in order to fall within the 0 to 1.8v range. This is accomplished by a simple DC bias circuit shown in Figure 5.1. The resistor connected from the input terminal to ground is important because it ensures that the input is at 0v before a connection is made, protecting the measurement source from any

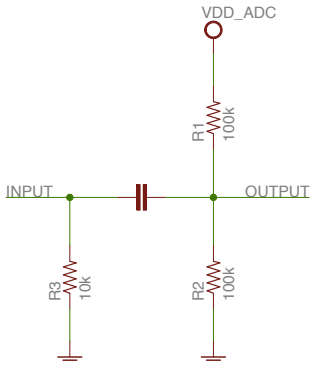

Figure 5.1: Schematic of DC bias circuit.

unexpected charge on the capacitor. The DC bias voltage will be generated by a voltage divider between VDD_ADC and ADC_GND with equal value resistors to ensure the mean of AC signal falls exactly in the middle of the input range.

5.1.2 Anti-Aliasing Filter

PMUs typically sample at relatively low rates, 3kHz or less, as the nominal frequency of the power grid is only 60Hz. It is essential to low pass filter the signal before it is sampled as sampling at such a low rate means there is a much higher chance of aliasing. As gain reduction is desired, a non-inverting active low-pass filter was chosen. The BeagleBone Black is a single supply board necessitating a single supply operational amplifier be used in the filter, specifically the LM358 which both met the specifications and was already available in the lab. A DC coupled low-pass RC active filter design presented in the LM358 datasheet [23] was used as the basis for the design of this circuit. Resistor and capacitor values, listed in the circuit schematic in Figure 5.2 were calculated using the equations provided in the datasheet with a cutoff frequency of 100Hz, gain of 0.2 and quality factor of 1 as the design parameters. Filters inherently generate a phase difference between the input and output signals; it is important that this phase shift is measured and accounted for in the final calibration of the device.

5.1.3 Analog to Digital Conversion

Sampling is handled by the touchscreen controller and analog-to-digital converter subsystem (TS_ADC_SS). The systems was designed to be used as a digitizer for touch screen input, but can also operate as a regular ADC when set to general purpose mode. Analog input can range from 0 to 1.8v and there are 8 channels available on the BeagleBone Black. However, the 8 channel count is somewhat misleading as there is only one ADC. Channels

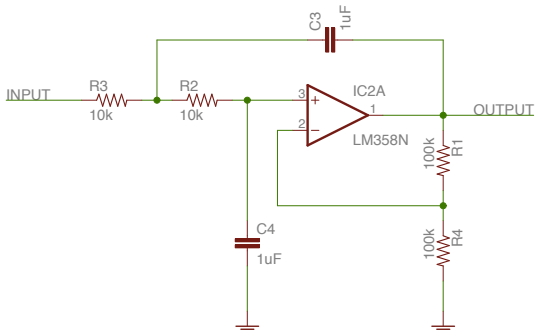

Figure 5.2: Schematic of anti-aliasing filter. Design adapted from the LM358 datasheet [23].

are multiplexed to the ADC, as shown in Figure 5.3, meaning that channel capture can only happen sequentially. Multiplexing of channels is controlled by a finite state machine with sixteen sequence steps. Each step corresponds to the acquisition and storage of data from one channel as outlined in Figure **??**. Two steps will be used to acquire both the AC voltage waveform and the pulse per second signal from the GPS. The minimum delay between sampling steps is 15 ADC clock cycles, a value that must be taken into account in the calibration of the device.

5.1.4 GPS Timing

The Ultimate GPS module from Adafruit is largely self-contained, necessitating only a few external connections to enable functionality of the chip. An onboard voltage regulator enables the use of sources from 3.3 to 5v, both of which are available on the pinout of the BeagleBone Black. 3.3v was chosen for the supply voltage simply because the 5v pin was already connected to the low pass-filter circuit. Communication between the GPS module and the Black is handled by the Universal Asynchronous Receiver Transmitter (UART). National Marine Electronics Association (NMEA) sentences, a standard format for GPS data, are received from the GPS via UART and used to assign UTC time tags to phase measurements.

The PPS signal is acquired via the onboard ADC in conjunction with the AC voltage waveform. This signal serves as the marker for the start of a Discrete Fourier Transform (DFT) window. The output of VFix, which is pulled low when a strong signal is found, is connected to a general purpose input output, indicating when the GPS is ready for measurement to begin.

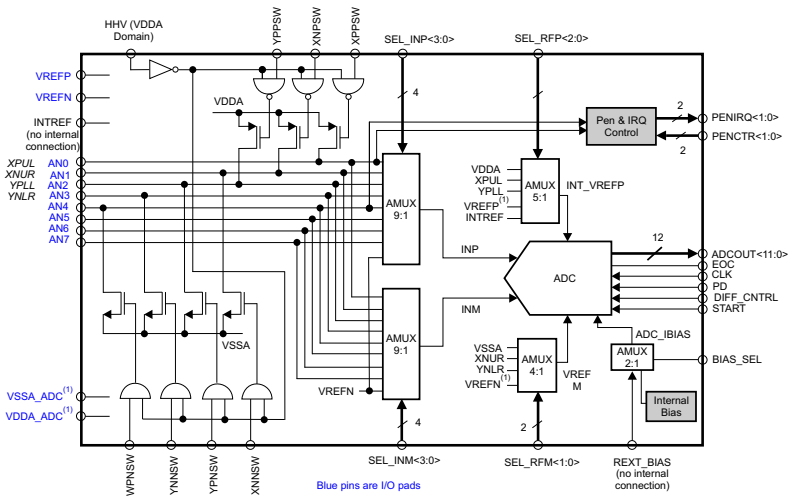

Figure 5.3: Block diagram of ADC analog front end operation. Source: [22]

5.2 Signal Processing

5.2.1 Sampling Rate

The sampling rate of ADC acquisitions is controlled both by dividing the ADC clock and by increasing the delay between measurement steps. Both methods will be implemented to achieve sampling at an integer multiple of 60Hz, desired to ease the processing requirements once sampled.

5.2.2 Raw Data Processing

Data from the ADC is stored as a 32 bit hex value. Both the four bit channel ID and the twelve bit converted value are contained within these 32 bits. Two bitwise shift operations are used to extract each component and then the values are converted to integers for further processing.

5.2.3 Synchrophasor Estimation

Synchrophasor estimation will be handled by a basic DFT at the beginning of this project. Other methods such as IpDFT and Weighted Least Squares are shown to provide higher accuracy [24], but DFT will allow a proof of concept. The choice of Python as the programming language allows for easy substitution of the phase estimation algorithm.

Each second of raw data was divided into 30 windows and a DFT performed on each to achieve the desired reporting rate of 30 phasors/second. Once the DFT for each window is

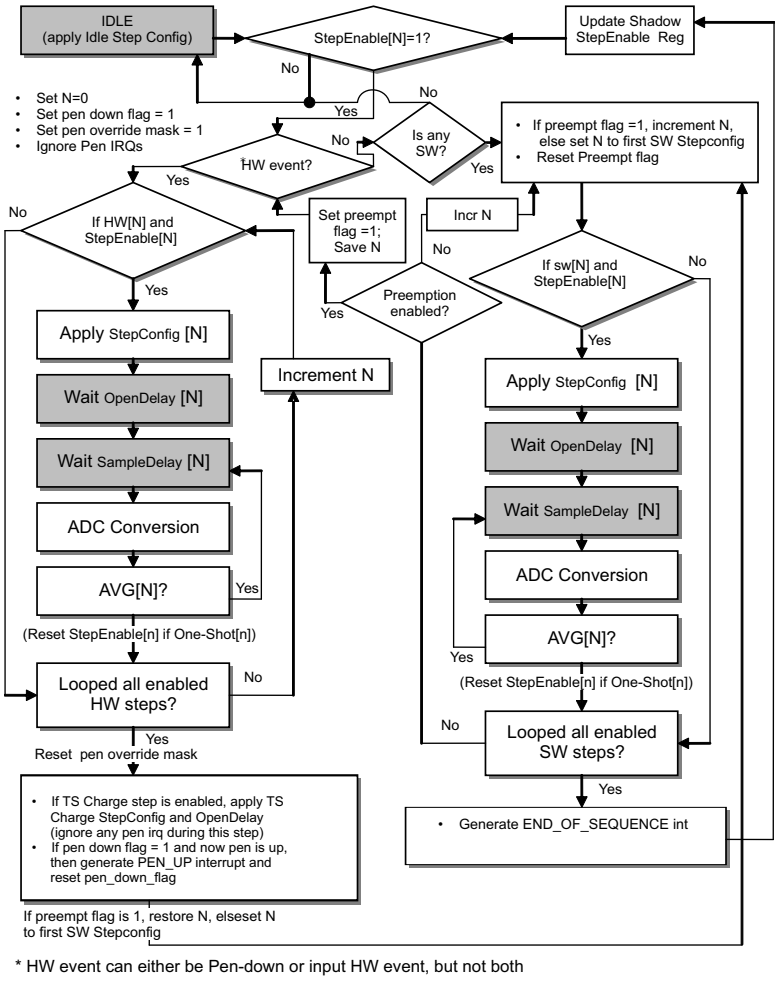

Figure 5.4: Flowchart for ADC subsystem step sequencer. Source: [22]

completed, it is paired with its corresponding GPS time tag and transmitted via ethernet.

Component	**Source**	**Mfr. Part Num.**	**Cost**
BeagleBone Black	Adafruit	1876	$55.00
Ultimate GPS	Adafruit	746	$39.95
GPS Antenna	Adafruit	960	$12.95
5v 2A DC Power Supply	Adafruit	276	$7.95
14:1 Power Transformer	DigiKey	HM510-ND	$19.82
LM358 Single Supply Op Amp	DigiKey	LM358NFS-ND	$0.49

Table 5.1: Bill of Materials

Chapter 6

Final Design

6.1 Hardware Design

6.1.1 Voltage Step Down

Connection to the wall will be made with a NEMA 5-15 compliant connector, the common standard for residential outlets [17]. The ADC input range of the BeagleBone Black is 0 to 1.8v [22], but going directly to this range would require a transformer ratio of 66.7:1 or greater, something that is not commonly found in 120v transformer offerings. A 14:1 transformer was chosen instead because of its availability and price. This transformer yields a 8.57v peak-to-peak output when connected to the 120v wall outlet. The voltage is further reduced into the ADC range by attenuation in the low pass filtering circuit.

Input Protection

Although 120v is not as dangerous as the high voltage that commercial PMUs measure, it is important that this device have safety features to protect both the low voltage electronics and the end user. A 0.5A fast-acting fuse was placed between the hot wire of the plug and the device as the power supply for chosen for the BeagleBone Black has a maximum current draw of 0.3A. The digital (DGND), analog (GND_ADC), and earth grounds as well as the neutral wire are all tied together to ensure that there are no ground loops that might affect measurement, but also to ensure that any shorts or loose wires will not generate unexpected voltages throughout the circuit.

DC Bias

It is also necessary to add a DC bias to the AC signal in order to fall within the 0 to 1.8v range. This is accomplished by a simple DC bias circuit shown in Figure 5.1. The resistor connected from the input terminal to ground is important because it ensures that the input is at 0v before a connection is made, protecting the measurement source from any

unexpected charge on the capacitor. The DC bias voltage will be generated by a voltage divider between VDD_ADC and ADC_GND with equal value resistors to ensure the mean of AC signal falls exactly in the middle of the input range.

6.1.2 Anti-Aliasing Filter

PMUs typically sample at relatively low rates, 3kHz or less, as the nominal frequency of the power grid is only 60Hz. It is essential to low pass filter the signal before it is sampled as sampling at such a low rate means there is a much higher chance of aliasing. As gain reduction is desired, a non-inverting active low-pass filter was chosen. The BeagleBone Black is a single supply board necessitating a single supply operational amplifier be used in the filter, specifically the LM358 which both met the specifications and was already available in the lab. A DC coupled low-pass RC active filter design presented in the LM358 datasheet [23] was used as the basis for the design of this circuit. Resistor and capacitor values, listed in the circuit schematic in Figure 5.2 were calculated using the equations provided in the datasheet with a cutoff frequency of 100Hz, gain of 0.2 and quality factor of 1 as the design parameters. Filters inherently generate a phase difference between the input and output signals; it is important that this phase shift is measured and accounted for in the final calibration of the device.

6.1.3 Analog to Digital Conversion

Sampling is handled by the touchscreen controller and analog-to-digital converter subsystem (TS_ADC_SS). The systems was designed to be used as a digitizer for touch screen input, but can also operate as a regular ADC when set to general purpose mode. Analog input can range from 0 to 1.8v and there are 8 channels available on the BeagleBone Black. However, the 8 channel count is somewhat misleading as there is only one ADC. Channels are multiplexed to the ADC, as shown in Figure 5.3, meaning that channel capture can only happen sequentially. Multiplexing of channels is controlled by a finite state machine with sixteen sequence steps. Each step corresponds to the acquisition and storage of data from one channel as outlined in Figure **??**. Two steps will be used to acquire both the AC voltage waveform and the pulse per second signal from the GPS. The minimum delay between sampling steps is 15 ADC clock cycles, a value that must be taken into account in the calibration of the device.

Values from the ADC are stored in a FIFO buffer as shown in Figure 6.1. Samples are retrieved from this buffer by the PRU via the code shown in Appendix D. They are moved into shared memory, accessible from the CPU via a two part linear buffer. When the first segment of this buffer is full, an interrupt is generated, signaling the CPU to read the data from memory and store it within a Python array.

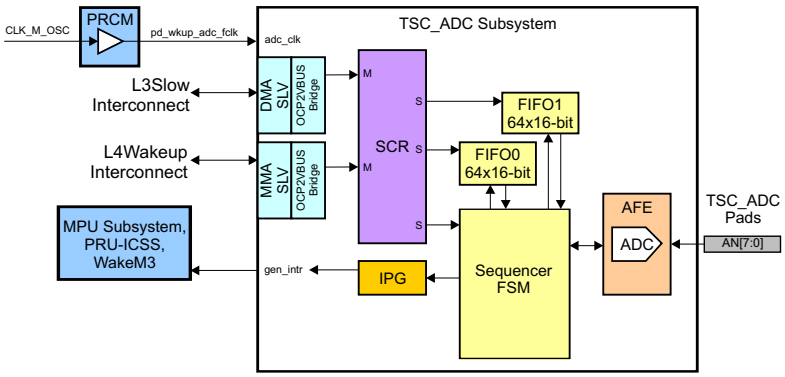

Figure 6.1: Block diagram for ADC sample acquisition and storage. Source: [22]

6.1.4 GPS Timing

The Ultimate GPS module from Adafruit is largely self-contained, necessitating only a few external connections to enable functionality of the chip. An onboard voltage regulator enables the use of sources from 3.3 to 5v, both of which are available on the pinout of the BeagleBone Black. 3.3v was chosen for the supply voltage simply because the 5v pin was already connected to the low pass-filter circuit. Communication between the GPS module and the Black is handled by the Universal Asynchronous Receiver Transmitter (UART). National Marine Electronics Association (NMEA) sentences, a standard format for GPS data, are received from the GPS via UART and used to assign UTC time tags to phase measurements.

The PPS signal was acquired via the onboard ADC in conjunction with the AC voltage waveform. The signal is a pulse width modulation signal with a frequency of 1Hz and a magnitude of 3.3v. A simple voltage divider was used to divide this signal in half to fit the 1.8v maximum on the ADC inputs. This signal serves as the marker for the start of a Discrete Fourier Transform (DFT) window. The output of "VFix", which is pulled low when a strong signal is found [19], is connected to a general purpose input output, indicating when the GPS is ready for measurement to begin.

6.2 Signal Processing

6.2.1 Sampling Rate

The sampling rate of ADC acquisitions is controlled both by dividing the ADC clock and by increasing the delay between measurement steps. Both methods will be implemented to achieve sampling at an integer multiple of 60Hz, desired to ease the processing requirements once sampled.

6.2.2 Raw Data Processing

Data from the ADC is stored as a 32 bit hex value. Both the four bit channel ID and the twelve bit converted value are contained within these 32 bits. Two bitwise shift operations are used to extract each component and then the values are converted to integers for further processing.

6.2.3 Synchrophasor Estimation

Synchrophasor estimation will be handled by a basic DFT at the beginning of this project. Other methods such as IpDFT and Weighted Least Squares are shown to provide higher accuracy [24], but DFT will allow a proof of concept. The choice of Python as the programming language allows for easy substitution of the phase estimation algorithm.

Each second of raw data was divided into 30 windows and a DFT performed on each to achieve the desired reporting rate of 30 phasors/second.

Chapter 7

Results

There currently exists an issue in the code with the way interrupts are generated by the PRU and handled by the CPU when the two part buffer reaches its capacity. As such, only one buffer's worth of data can be captured and processed at a time. Repeatedly executing the capture process allows the user to overcome this by repeatedly executing the capture operation.

Figure 7.1 and 7.2 show the change in phase and frequency over one second of time for the measured data.

7.1 Evaluation Plan

Though the device is not fully functional yet, the following evaluation plan has been devised for the device once the coding issues have been resolved.

7.1.1 Timing Accuracy

Acquiring a highly accurate time source, such as an atomic clock, to test the accuracy of the timing circuit is impractical, due to the cost of such devices. However an indirect test can be performed to analyze the stability of the timing circuit. Measuring the period of the pulse per second signal provided by the timing source over sixty minutes will give a good indication of the short term stability of the time source and local oscillator.

7.1.2 Measurement

The reference conditions for each testing parameter, as specified in section 5.5.4 of IEEE C37.118.1, are listed below. Each parameter should remain constant, unless it is currently being tested.

- Voltage at nominal

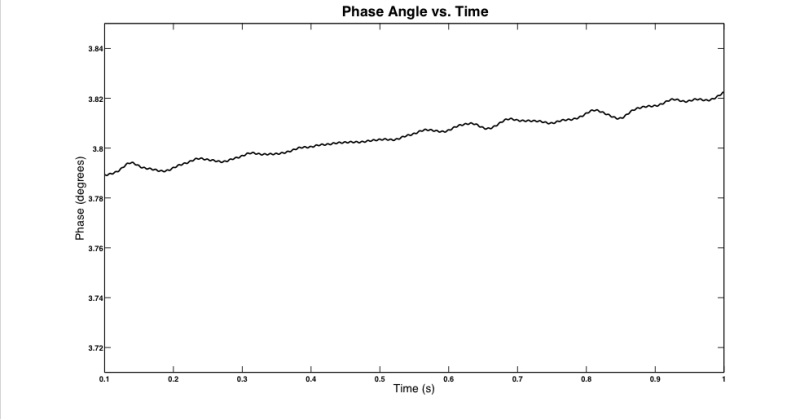

Figure 7.1: Plot of phase vs. time for one second of measured data.

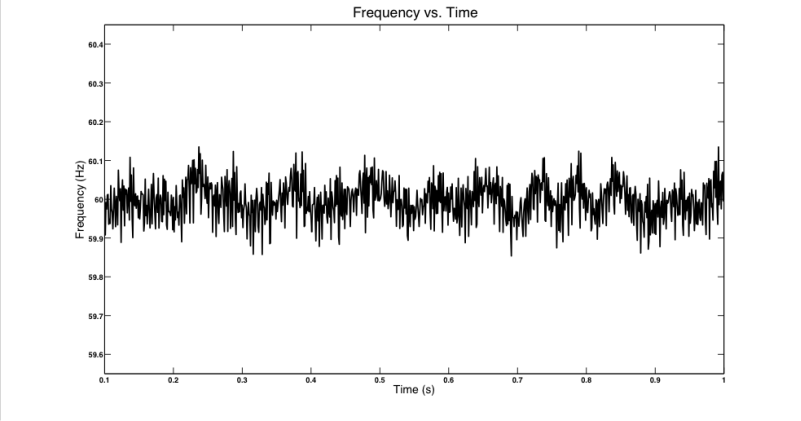

Figure 7.2: Plot of frequency vs. time for one second of measured data.

- Current at nominal
- Frequency at nominal
- Voltage, current, phase, and frequency constant
- All interfering signals $< 0.2\%$ of the nominal frequency (60Hz)
- Temperature $23° \pm 3°C$
- Humidity $> 90\%$

Synchrophasor Estimation

Synchrophasor estimation should be tested by calculating the TVE as defined in (3.1) for each testing condition listed in Table 3 of [11]. The TVE should remain below 1% in all testing conditions. The signal generator and oscilloscope must have a testing uncertainty ratio of 4, i.e. for desired TVE of 1%, the devices should be able to measure TVE within $\pm 0.25\%$.

7.1.3 Communication

Confirmation of the proper communication protocol specified in IEEE C37.118.2 will be verified using the PMU Connection Tester software package provided by the Grid Protection Alliance[1]. If necessary, the Wireshark filter for IEEE C37.118 communication may be used to examine the raw data frames[2].

[1] The software can be found at http://pmuconnectiontester.codeplex.com

[2] A more detailed summary of the filter can be found at http://www.wireshark.org/docs/dfref/s/synphasor.html

Chapter 8

Production Schedule

For the most part, time estimates proposed at the beginning of the project and reproduced in 8.1 were accurate. The order of the weeks was shuffled around in order to prioritize the selection of more important components over the design of voltage step down and filtering circuits.

Week	Objective
1	Design voltage step down and device power circuit
2	Design anti-aliasing filter, select sampling frequency and A/D converter
3	Research on and selection of time code source
4	Design timing and synchronization circuit
5	Select phase estimation algorithm
6-7	Design synchrophasor estimator
8	Design internet interface
9	Construction and testing of voltage step down and device power circuit
10	Construction of anti-aliasing filter

Table 8.1: Proposed weekly schedule for Fall 2014

Chapter 9

Cost analysis

The final cost for this project came in well below the initial proposed budget, reproduced in Table 9.1. A number of factors contributed to this, most notably the identification of a sufficiently accurate GPS module available for 1/10 the cost of timing specific offerings. Replacement of the proposed FPGA development board with a single-board computer also enabled a large cost reduction. The change in computing platform was based on the need for remote management access, something that would have been impossible to reliably implement on an FPGA.

A transformer and the LM358 chip were obtained from previous projects within the department. The transformer also included a NEMA 15-5 plug. Components obtained for free are indicated in parentheses in the cost summary shown in Table 9.2.

Construction of this residential phasor measurement unit came in well below the $14,000 estimated cost of commercial offerings [3]. Achieving such a substantial cost reduction is made possible by the fact that this device takes measurements at low voltage levels, removing a large number of insulation and protection requirements for high voltage PMUs. Commercial devices are more accurate than this residential PMU, but the low cost of this device will enable installation in more locations, offsetting accuracy shortfalls with an increased volume of data.

Component	Quantity	Cost	Subtotal
Project Enclosure	1	$20	$20
NEMA 5-15P Connector	1	$16	$16
Transformer - 12:1	1	$20	$20
AC/DC Converter	1	$21	$21
GPS Time Code Receiver	1	$400	$400
FPGA - Altera DE2	1	$269	$269
Total Projected Cost			$746

Table 9.1: Proposed budget for PMU components

Component	Quantity	Cost
BeagleBone Black	1	$55.00
Ultimate GPS	1	$39.95
GPS Antenna	1	$12.95
5v 2A DC Power Supply	1	$7.95
14:1 Power Transformer	1	$(19.82)
NEMA 5-15P Connector	1	$(15.98)
LM358 Single Supply Op Amp	1	$(0.49)
Total Cost		$142.16

Table 9.2: Cost of PMU components

Chapter 10

User Manual

10.1 Setup

Setup of the residential PMU is designed to be quick and simple. Device power and measurement input are both drawn from the same connector, a NEMA 15-5 plug that is compatible with residential outlets. The plug is simply inserted into the wall outlet to provide power and the AC voltage to be measured. Transmission of measured data and device management is handled via ethernet; the device must be connected to the same network as the server receiving data. The GPS antenna is connected to the external antenna connector mounted on the device and should be placed as close to a window or outside wall as possible. It may be necessary to experiment with antenna positioning; the LED labelled "Fix" on the GPS module will cease flashing when a strong enough signal is obtained to begin measurements.

10.1.1 Calibration

In the current iteration of the device, a 10kΩ potentiometer configured as a voltage divider is connected between the secondary side of the transformer and the input of the low pass filter. This allows the magnitude of the incoming signal to be adjusted manually to span the entire 0 to 1.8v range of the ADC, thus maximizing measurement resolution. Once the potentiometer is set, the magnitude and phase difference between the 120v wall outlet and signal at the ADC input pin on the BeagleBone Black should be measured with an oscilloscope or other suitable tool. These values need to be entered into the Calibration section of the Python code, shown in Appendix E. Calibration should be performed on an annual basis according to the standards set forth in IEEE C37.119.1 [25].

10.2 Operation

Once connected to power, the device will boot up and automatically begin to execute the Python code. After a strong GPS signal is obtained, indicated when the Fix light on the GPS is extinguished, the device will begin collecting and processing data. The incoming data stream can be viewed using OpenPDC, an open source PDC provided by the Grid Protection Alliance [26]. As of March 6, 2015 the software is now compatible with Mac OS X and Linux operating systems in addition to Windows Server.

When powering down the device, either the command "sudo shutdown -h now" should be issued via a SSH login or the power button on the BeagleBone Black should be pressed. These shutdown methods are mandated by the manufacturer in order to prevent damage to the device or loss of data.

10.3 Maintenance

SSH access is enabled on the BeagleBone Black, allowing remote login to update or modify code on the device. IP address configuration is currently handled by DHCP so any network changes may alter the IP address. Though an inconvenience, this choice was a necessity as the device may be connected to a variety of different networks with different address ranges. Connecting a monitor and keyboard to the device and issuing the command "ifconfig" from the command line is the most reliable way to determine the IP address once the device is installed.

The revisions of all hardware and software components used by the device are listed in Table 10.1. Monthly software updates should be issued using the command "sudo apt-get update" to update package lists from the Debian repository and "sudo apt-get upgrade" to perform the update. Proper backup practices should be followed before these updates to prevent data or functionality loss. It is not recommended to update the Linux kernel, a process that requires the internal memory of the BeagleBone Black to be flashed with a new image, without thorough testing as subsequent versions make major changes to the way in which the PRU is initialized and accessed by the CPU.

Component	Revision
Hardware	
BeagleBone Black	C
Ultimate GPS Module	3
Software	
Linux Kernel	3.8.13-bone47
Debian	7.8
Python	2.7.3
AM335x Driver	Commit e4d44bd on GitHub
PyPRUSS	Commit 6feef2b on Bitbucket

Table 10.1: Component and software versions used in this project.

Chapter 11

Conclusion

The goal of this project is to design an inexpensive residential phasor measurement unit. Designing an inexpensive PMU will allow the proliferation of real-time, networked monitoring devices across the grid. Doing so increases the resolution of data available to control room operators and regional administrators. This will allow for the design of automatic control systems that can react faster than control room operators to counteract and confine disturbances on the grid.

The device will be entirely self contained, with the only external connections to the internet and the residential wall outlet. There should be no interaction between the resident of the house and the device, initial installation will be done by the power company. The device will be designed in compliance with the IEEE Std. C37.118, in order to ensure compatibility with existing phasor data concentrators and visualization software.

Having an inexpensive PMU on the market opens up many possibilities for future development and supports some of the objectives of the smart grid. In particular, it will support increased use of distributed generation. Distributed generation allows for the use of small, sustainable sources to supplement or be the sole power source of small areas. This capability is key to the spread of sustainable power.

Appendix A

Preliminary Circuit Diagram

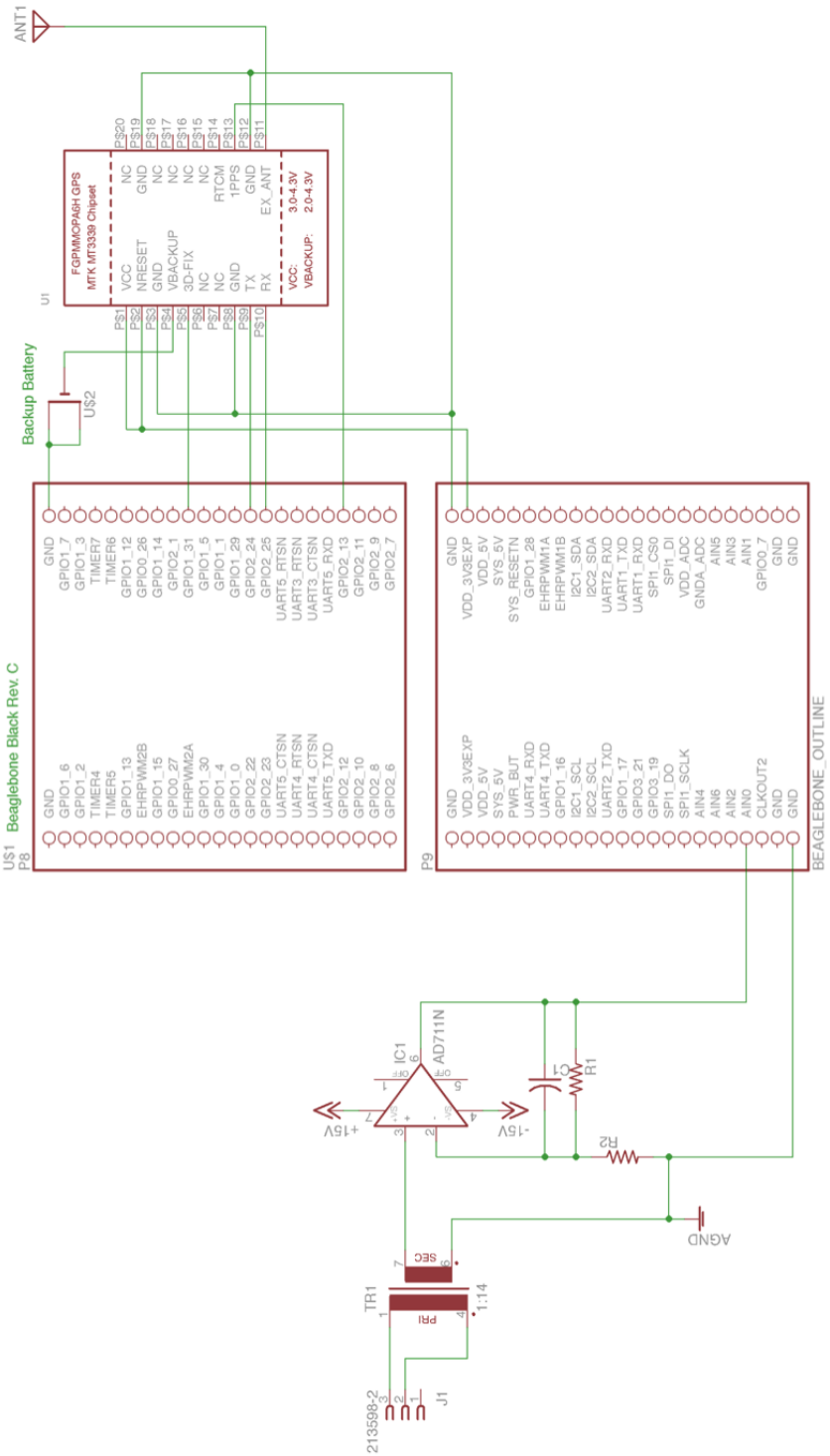

Figure A.1: Circuit schematic of preliminary proposed design.

Appendix B

Final Circuit Diagram

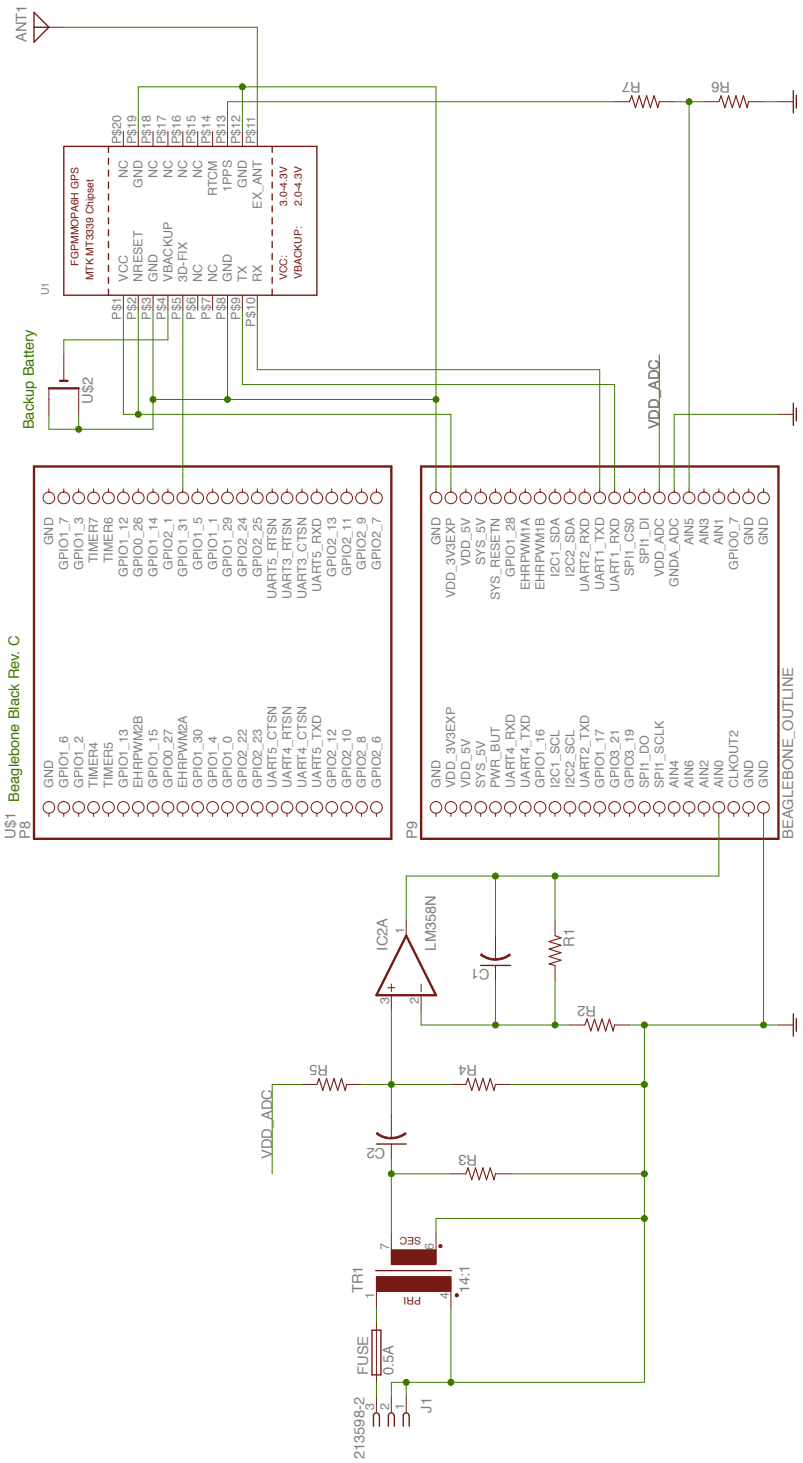

Figure B.1: Circuit schematic of finally design.

Appendix C

Texas Instruments Header

N.B. Additions to the example header provided by Texas Instruments are marked with the title "Addition."

```
1  // ***************************************************************************/
2  // file:    PRU_memAccess_DDR_PRUsharedRAM.hp
3  //
4  // brief:  PRU_memAccess_DDR_PRUsharedRAM assembly constants.
5  //
6  //
7  //  (C) Copyright 2012, Texas Instruments, Inc
8  //
9  //  author     M. Watkins
10 // ***************************************************************************/
11
12 #ifndef _PRU_memAccess_DDR_PRUsharedRAM_HP_
13 #define _PRU_memAccess_DDR_PRUsharedRAM_HP_
14
15 // *****************************************
16 // *      Global Macro definitions         *
17 // *****************************************
18
19 // Refer to this mapping in the file - \prussdrv\include\pruss_intc_mapping.h
20 #define PRU0_PRU1_INTERRUPT     17
21 #define PRU1_PRU0_INTERRUPT     18
22 #define PRU0_ARM_INTERRUPT      19
23 #define PRU1_ARM_INTERRUPT      20
24 #define ARM_PRU0_INTERRUPT      21
25 #define ARM_PRU1_INTERRUPT      22
26
27 #define CONST_PRUCFG        C4
28 #define CONST_PRUDRAM       C24
29 #define CONST_PRUSHAREDRAM  C28
30 #define CONST_DDR           C31
31
```

```
32 // ********************
33 //  Addition - J. Vick
34 // ********************
35 #define PRUSHAREDRAM 0x00010000 //12kB
36 #define PRUDRAM1 0x00000000
37 #define PRUDRAM2 0x00002000
38 #define PRUDRAMSIZE 8192
39 #define PRUSHAREDRAMSIZE 12288
40 // **************
41 //  End addition
42 // **************
43
44 // Address for the Constant table Block Index Register (CTBIR)
45 #define CTBIR           0x22020
46
47 // Address for the Constant table Programmable Pointer Register 0(CTPPR_0)
48 #define CTPPR_0         0x22028
49
50 // Address for the Constant table Programmable Pointer Register 1(CTPPR_1)
51 #define CTPPR_1         0x2202C
52
53 .macro  LD32
54 .mparam dst,src
55     LBBO    dst,src,#0x00,4
56 .endm
57
58
59 .macro  LD16
60 .mparam dst,src
61     LBBO    dst,src,#0x00,2
62 .endm
63
64
65 .macro  LD8
66 .mparam dst,src
67     LBBO    dst,src,#0x00,1
68 .endm
69
70
71 .macro ST32
72 .mparam src,dst
73     SBBO    src,dst,#0x00,4
74 .endm
75
76
77 .macro ST16
78 .mparam src,dst
79     SBBO    src,dst,#0x00,2
80 .endm
81
```

```
82
83 .macro ST8
84 .mparam src,dst
85     SBBO     src,dst,#0x00,1
86 .endm
87
88
89 // ********************
90 //  Additions - J. Vick
91 // ********************
92 //fill the entire DRAM with val (2-byte-value)
93 .macro  FILLSHAREDRAM
94 .mparam val
95     MOV r3,PRUSHAREDRAMSIZE
96 FILLSHAREDRAMREPEAT2:
97     SUB r3,r3,2
98     SBCO val,CONST_PRUSHAREDRAM,r3,2
99   QBNE FILLSHAREDRAMREPEAT2,r3,0
100 .endm
101 // **************
102 //  End addition
103 // **************
104
105 // ************************************
106 // *    Global Structure Definitions    *
107 // ************************************
108
109 .struct Global
110     .u32 regPointer
111     .u32 regVal
112 .ends
113
114
115 // ************************************
116 // *     Global Register Assignments    *
117 // ************************************
118
119 .assign Global, r2, *, global
120
121
122 #endif //_PRU_memAccess_DDR_PRUsharedRAM_
```

Appendix D

PRU Assembly Code

```
1  .origin 0 // offset of the start of the code in PRU memory
2  .entrypoint START // program entry point, used by debugger only
3
4  #include "ADCCollector.hp"
5
6  // REGISTER ADDRESS DEFINITIONS
7  #define ADC_CTRL          0x44E0D040
8  #define SYSCONFIG         0x44E0D010
9  #define ADCSTAT           0x44E0D044
10 #define ADC_CLKDIV        0x44E0D04C
11 #define IRQENABLE_SET     0x44E0D02C
12 #define IRQENABLE_CLR     0x44E0D030
13 #define IRQSTATUS         0x44E0D028
14 #define FIFO0COUNT        0x44E0D0E4
15 #define FIFO0THRESHOLD    0x44E0D0E8
16
17 #define STEPENABLE        0x44E0D054
18 #define STEPCONFIG1       0x44E0D064
19 #define STEPDELAY1        0x44E0D068
20 #define STEPCONFIG2       0x44E0D06C
21 #define STEPDELAY2        0x44E0D070
22
23 // Data locations
24 #define FIFO0DATA 0x44E0D100
25
26
27 // Variable definitions
28 #define BUFF_SIZE 0x0000FA0 //Total buff size: 4kbyte(Each buffer has
       2kbyte: 500 piece of data)
29 #define HALF_SIZE BUFF_SIZE / 2
30 #define FIFOTHRESHOLD 0x00000031 //value -1
31 #define FIFOTHRESHOLDNUM FIFOTHRESHOLD+1
32
33 // ##########################
```

```
34 //     MACRO DEFINITIONS
35 // ########################
36
37 .macro FIFOWAIT
38     FIFO:
39   LBBO r3, r13, 0, 4
40   QBBC FIFO, r3.t2
41 .endm
42
43 .macro READADC
44     //Initialize buffer status (0: empty, 1: first buffer is ready, 2:
           second buffer is ready)
45     MOV r2, 0x0
46     SBCO r2, CONST_PRUSHAREDRAM, 0, 4
47     MOV r7, HALF_SIZE
48
49     MOV r12, FIFO0DATA
50     MOV r10, FIFO0COUNT
51     MOV r13, IRQSTATUS
52
53     MOV r5, 0 //Shared RAM address of ADC Saving position
54     MOV r6, BUFF_SIZE  //Counting variable
55     QBA READ
56
57     INITV:
58         MOV r5, 0 //Shared RAM address of ADC Saving position
59         MOV r6, BUFF_SIZE  //Counting variable
60   //QBNE EMPTYFIFO, r14, 0
61   QBA READ
62
63     READ:
64   MOV r3, 0x0000000F
65         SBBO r3, r13, 0, 1
66
67   FIFOWAIT
68
69   MOV r14, FIFOTHRESHOLDNUM
70         EMPTYFIFO:
71       LBBO r3, r12, 0, 3
72             ADD r5, r5, 4
73             SBCO r3, CONST_PRUSHAREDRAM, r5, 3
74
75       SUB r14, r14, 1
76             SUB r6, r6, 4
77             QBEQ CHBUFFSTATUS1, r6, r7 //If first buffer is ready
78             QBEQ CHBUFFSTATUS2, r6, 0 //If second buffer is ready
79
80       QBNE EMPTYFIFO, r14, 0
81
82   QBA READ
```

```
83
84          //Change buffer status to 1
85          CHBUFFSTATUS1:
86        MOV r3, 0x00000001
87              SBCO r3, CONST_PRUSHAREDRAM, 0, 4
88        //QBNE EMPTYFIFO, r14, 0
89        MOV r31.b0, PRU0_ARM_INTERRUPT+16
90        QBA READ
91
92          //Change buffer status to 2
93          CHBUFFSTATUS2:
94              MOV r3, 0x00000002
95              SBCO r3, CONST_PRUSHAREDRAM, 0, 4
96        MOV r31.b0, PRU0_ARM_INTERRUPT+16
97              QBA INITV
98
99      //Send event to host program
100     END:
101       MOV r31.b0, PRU0_ARM_INTERRUPT+16
102       HALT
103 .endm
104
105 // #####################
106 //     Initialize ADC
107 // #####################
108 START:
109     // Enable OCP master port
110     LBCO r0, CONST_PRUCFG, 4, 4
111     CLR r0, r0, 4
112     SBCO r0, CONST_PRUCFG, 4, 4
113
114     //C28 will point to 0x00012000 (PRU shared RAM)
115     MOV r0, 0x00000120
116     MOV r1, CTPPR_0
117     ST32 r0, r1
118
119     //Reset SYSConfig Register to 0
120     MOV  r2, 0x44E0D010 // load register address
121     MOV  r3, 0x00000000
122     SBBO r3, r2, 0, 4   // set register
123
124     //Wait for ADC to be idle
125     AdcIdle:
126       MOV  r2, 0x44E0D044 // load register address
127       QBBS AdcIdle, r2, 5
128
129     //Write enable steps, disable ADC
130     MOV r2, ADC_CTRL
131     MOV r3, 0x00000004
132     SBBO r3, r2, 0, 4
```

```
133
134     //Set ADC_CLKDIV
135     MOV r2, ADC_CLKDIV
136     MOV r3, 0x0000031F //value-1 399 0x18F
137     SBBO r3, r2, 0, 4
138
139     MOV r0, 0
140     FILLSHAREDRAM r0
141
142     CLEARFIFO:
143         MOV r2, FIFO0DATA
144         LBBO r3, r2, 0, 4
145         MOV r2, FIFO0COUNT
146         LBBO r3, r2, 0, 4
147         QBNE CLEARFIFO, r3, 0
148
149     //Clear any residual interrupts
150     MOV r2, IRQSTATUS
151     MOV r3, 0x000007FF
152     SBBO r3, r2, 0, 4
153
154     //Disable all interrupts
155     MOV r2, IRQENABLE_CLR
156     MOV r3, 0x000007FF
157     SBBO r3, r2, 0, 4
158
159     //Enable FIFO0 interrupt in INTENABLE_SET
160     MOV r2, IRQENABLE_SET
161     MOV r3, 0x00000004
162     SBBO r3, r2, 0, 4
163
164     //Set FIFO0THRESHOLD
165     MOV r2, FIFO0THRESHOLD
166     MOV r3, FIFOTHRESHOLD //value-1
167     SBBO r3, r2, 0, 4
168
169     //STEPCONFIG1
170     MOV r2, STEPCONFIG1
171     MOV r3, 0x00000001 //continuous mode
172     SBBO r3, r2, 0, 4
173
174     //STEPDELAY1
175     MOV r2, STEPDELAY1
176     MOV r3, 0x0000000E //value-1
177     SBBO r3, r2, 0, 4
178
179     //STEPCONFIG2
180     MOV r2, STEPCONFIG2
181     MOV r3, 0x00280001 //continuous mode
182     SBBO r3, r2, 0, 4
```

```
183
184     //STEPDELAY2
185     MOV r2, STEPDELAY2
186     MOV r3, 0x0000000F //sample is value-1 open is not
187     SBBO r3, r2, 0, 4
188
189     //Set ADC STEPENABLE
190     MOV r2, STEPENABLE
191     MOV r3, 0x00000004
192     SBBO r3, r2, 0, 4
193
194     //Write protect steps, enable ADC
195     MOV r2, ADC_CTRL
196     MOV r3, 0x00000003
197     SBBO r3, r2, 0, 4
198
199
200     READADC
```

Appendix E

Python Code

```
1  import pypruss as pru
2  import mmap
3  import numpy as np
4  import struct
5  import time
6
7  ## MEMORY LOCATIONS ##
8
9  PRU_ICSS = 0x4A300000
10 PRU_ICSS_LEN = 512*1024
11
12 RAM0_START = 0x00000000
13 RAM1_START = 0x00002000
14 RAM2_START = 0x00012000
15
16 TOTAL_BUFFER_LEN = 0x00000FA0
17 BUFFER_LEN = TOTAL_BUFFER_LEN/2
18 BUFFER1_START = RAM2_START + 4
19 BUFFER2_START = BUFFER1_START + BUFFER_LEN
20
21 ## FUNCTION DEFINITIONS ##
22 def processRawADC(value):
23     value = 0x00000FFF & value
24     value = int(value)
25     value = (value*1.8)/(2^12)
26       return value
27
28 def channelID(value):
29     value = 0x000F0000 & value
30     value = value >> 16
31     return value
32
33 ## PRU SETUP ##
34 pru.modprobe()     # enable uio_pruss module
```

```
35 pru.init()     # initialize PRU
36 pru.open(0)    # open connection to PRU 0
37 pru.pruintc_init()  # configure interrupt handlers
38 pru.exec_program(0, "./oneshot.bin")  # load assembly file
39
40 counter = 0
41
42 f = open("/dev/mem", "r+b")
43 output = open("./results.txt", "w")
44
45
46 while counter < 10:
47   start = time.time()
48   pru.wait_for_event(0)
49
50   ddr_mem = mmap.mmap(f.fileno(), PRU_ICSS_LEN, offset=PRU_ICSS)
51         shared = struct.unpack('L', ddr_mem[RAM2_START:RAM2_START+4])
52   print shared[0]
53   if shared[0] == 1:
54     print "buffer 1"
55     for i in range(0,500):
56       fifo =  struct.unpack('L',
             ddr_mem[BUFFER2_START+(i*4):BUFFER2_START+4+(i*4)])[0]
57       value = processRawADC(fifo)
58       channelNum = channelID(fifo)
59       output.write(str(channelNum) + "," + str(value) +"\n")
60     counter += 1
61     pru.clear_event(0)
62
63   elif shared[0] == 2:
64           shared = struct.unpack('L', ddr_mem[RAM2_START:RAM2_START+4])
65           print "buffer 2"
66     for i in range(0,500):
67       fifo = struct.unpack('L',
             ddr_mem[BUFFER2_START+(i*4):BUFFER2_START+4+(i*4)])[0]
68       value = processRawADC(fifo)
69       channelNum = channelID(fifo)
70                   output.write(str(channelNum) + "," + str(value) +"\n")
71     counter +=1
72     pru.clear_event(0)
73   end = time.time()
74   #print end-start
75
76 f.close()
77 output.close()
78
79
80 pru.clear_event(0)
81 pru.pru_disable(0)
82 pru.exit()
```

Bibliography

[1] E. Lipton, R. Perez-Pena, and M. Wald, "Overseers missed big picture as failures led to blackout," *New York Times*, September 2003.

[2] NERC, "NERC Final Blackout Reccommendations," North American Energy Reliability Corporation, Tech. Rep., 2004.

[3] U.S. Department of Energy, "Synchrophasor technologies and their deployment in the recovery act smart grid programs," Online, August 2013.

[4] C. P. Steinmetz, "Complex quantities and their use in electrical engineering," in *Proceedings of the International Electrical Congress Held in the City of Chicago, August 21st to 25th, 1893*. American Institute of Electrical Engineers, 1894, pp. 33–74.

[5] S. Das and T. Sidhu, "A simple synchrophasor estimation algorithm considering ieee standard c37.118.1-2011 and protection requirements," *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, no. 10, pp. 2704–2715, Oct 2013.

[6] M. Adamiak, B. Kasztenny, and W. Premerlani, "Synchrophasors: definition, measurement, and application," *Proceedings of the 59th Annual Georgia Tech Protective Relaying, Atlanta, GA*, 2005.

[7] K. Martin, D. Hamai, M. Adamiak, S. Anderson, M. Begovic, G. Benmouyal, G. Brunello, J. Burger, J. Y. Cai, B. Dickerson, V. Gharpure, B. Kennedy, D. Karlsson, A. Phadke, J. Salj, V. Skendzic, J. Sperr, Y. Song, C. Huntley, B. Kasztenny, and E. Price, "Exploring the ieee standard c37.118-2005 synchrophasors for power systems," *Power Delivery, IEEE Transactions on*, vol. 23, no. 4, pp. 1805–1811, Oct 2008.

[8] K. Kirihara, B. Pinte, and A. Yoon, "Phasor measurement unit," Online, February 2013.

[9] B. R. Miller, "Concept for next generation phasor measurement: A low-cost, self-contained, and wireless design," Master's thesis, University of Tennessee, 2010.

[10] D. Laverty, D. J. Morrow, A. McKinley, and M. Cregan, "Openpmu: Open source platform for synchrophasor applications and research," in *Power and Energy Society General Meeting, 2011 IEEE*, July 2011, pp. 1–6.

[11] "Ieee standard for synchrophasor measurements for power systems," *IEEE Std C37.118.1-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–61, Dec 2011.

[12] "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)*, pp. 1–25, April 2014.

[13] "IEEE standard for synchrophasor data transfer for power systems," *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, pp. 1–53, Dec 2011.

[14] B. C. Baker, "Anti-aliasing, analog filters for data acquisition systems," Microchip Technology Inc., Tech. Rep., 1999.

[15] "Ieee standard profile for use of ieee 1588 precision time protocol in power system applications," *IEEE Std C37.238-2011*, pp. 1–66, July 2011.

[16] D. Belega, D. Macii, and D. Petri, "Fast synchrophasor estimation by means of frequency-domain and time-domain algorithms," *Instrumentation and Measurement, IEEE Transactions on*, vol. 63, no. 2, pp. 388–401, Feb 2014.

[17] National Fire Protection Association, *National Electrical Code 2011*, ser. International electrical code series. National Fire Protection Association, 2010.

[18] *WI125 Specifications*, Connor Winfield, 1 2015.

[19] J. Delano, "FGPMMOPA6H gps standalone module data sheet," Globaltop Technology Inc., Tech. Rep., 2012.

[20] *125 Series Wi125 GPS Receiver*, Connor Winfield, 01 2015.

[21] E. Bakken, *PyPRUSS*, Hipster Circuits, March 2014.

[22] Texas Instruments, "Am335x sitara processors technical reference manual," Online, October 2011.

[23] *LMx58-N Dual-Operational Amplifiers*, Rev. i ed., Texas Instruments, January 2000.

[24] G. Barchi, D. Macii, and D. Petri, "Synchrophasor estimators accuracy: A comparative analysis," *Instrumentation and Measurement, IEEE Transactions on*, vol. 62, no. 5, pp. 963–973, May 2013.

[25] "Ieee standard for synchrophasors for power systems," *IEEE Std C37.118-2005 (Revision of IEEE Std 1344-1995)*, pp. $0_1 - -57, 2006$.

[26] Grid Protection Alliance, "Openpdc," March 2015. [Online]. Available: http://openpdc.codeplex.com