A 3rd Generation Frequency Disturbance Recorder:

A Secure, Low Cost Synchrophasor Measurement Device

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Jerel Alan Culliss
August 2015

Copyright © 2015 by Jerel A. Culliss
All rights reserved.

Dedication

I dedicate this work to my parents, Clarence and Maria, who have always stood by me with love and support. They have provided me with tremendous opportunities and have always encouraged me to pursue any path that will bring me happiness. Throughout all of my journeys and life experiences, they have been a constant source of motivation. They are both intelligent, kind, selfless people whom I strive to emulate every day. One could not ask for better parents, and I love them dearly.

Acknowledgements

This work was made possible thanks to the support, assistance, and encouragement received from a number of different people. First, I will be forever grateful to my advisor, Dr. Yilu Liu, for guiding me through much of the last decade of my life. Over all of those years, she has consistently provided excellent advice in academic, professional, and personal arenas. I could not have asked for nor found a better mentor. I would also like to thank the members of my committee, Dr. Leon Tolbert, Dr. Wei Gao, and Dr. Lee Riedinger for all of their comments, suggestions, and excellent advice in regards to my research and my continued career development.

The research that I have performed would fundamentally not have been possible without the prior research and efforts of the Power Information Technology Lab. The groundbreaking work that went into creating the Frequency Monitoring Network in the first place has been catalogued in dozens, if not hundreds, of research papers. This prior work served as excellent reference material for understanding how my own work could contribute to the state of the art. In addition, the direct assistance from many of my current colleagues in the PowerIT group will forever be appreciated. There isn't enough space here to list everyone or all of their contributions, but I must in particular thank Lingwei "Eric" Zhan. His encyclopedic knowledge of the current generation Frequency Disturbance Recorder was immensely valuable in determining how the device could be improved.

During my time here, I have been able to rely on a group of staff members in particular, and they deserve particular mention. Ms. Dana Bryson always kept me on task and went above and beyond any time that I needed her. She expertly handled a number of policy and procedure related issues that cropped up, and I am thankful for her assistance and friendship. Mr. Markus Iturriaga and Mr. Bryan Burke have always been around to assist with any IT-related issues and have also acted as excellent sounding boards for my thoughts on creating a secure measurement platform. I'll miss our weekly lunches, even the 10 minutes up front spent deciding where to go.

Special thanks go to my former colleagues at the Idaho National Laboratory, who were supportive of my endeavors to continue my education. I would also like to thank my current colleagues at the Oak Ridge National Laboratory for allowing me the scheduling flexibility necessary to complete this work.

Finally, I would like to thank my friends and family for their love and support.

Abstract

The Frequency Monitoring Network (FNET) is a wide-area phasor measurement system developed in 2003. It collects power system data using embedded devices known as Frequency Disturbance Recorders (FDRs) which are installed at distribution level voltages. These devices are single-phase synchrophasor measurement units which share a number of common attributes with their commercial counterparts.

Phasor measurements from FDRs across North America and other power grids around the world are transmitted over the Internet back to the FNET servers at the University of Tennessee. By analyzing the fluctuations in the grid's frequency, FNET can identify disruptive events relating to the operation of the system as a whole. Fundamentally, the effectiveness of FNET correlates to the number of units and their placement within the power system. Therefore, an effective method of improving the FNET system is to increase the rate of FDR deployment. The Generation-2 FDRs are limited in this sense by their cost, configurability, and maintainability. This generation of Frequency Disturbance Recorders was developed in 2006 and there have only been incremental improvements to them over their lifetime.

With this in mind, this purpose of this research is to develop a new FDR design which addresses the limitations of the current device – cost, maintainability, and communication. These improvements will ultimately increase the capabilities of FNET as a whole. This will be accomplished by a design which utilizes a modern microcontroller and a full-featured operating system combined with a modular hardware scheme. The combination of these factors will reduce costs and increase device capability while also allowing for improved future expansion and maintainability of the system. Research and development for both the hardware and software of this FDR is presented, along with a finalized prototype working unit which interfaces with the existing FNET servers.

Table of Contents

1. Introduction 1
   1.1. FNET Background 1
   1.2. Motivation and Objective 2
   1.3. Organization of Study 4
2. Review of Existing Technology and Standards 6
   2.1. Definition of Synchrophasor 6
   2.2. Phasor Measurement Unit Overview 7
      2.2.1. Applicable Standards 9
      2.2.2. Synchrophasor Applications 11
   2.3. Current FDR Architecture 13
      2.3.1. Timing/GPS Subsystem Limitations 15
      2.3.2. Computational Limitations 16
      2.3.3. Logistical Limitations 17
3. Next Generation FDR Hardware 18
   3.1. Embedded System Hardware Requirements and Considerations 18
      3.1.1. Pandaboard ES 19
      3.1.2. BeagleBoard-xM 20
      3.1.3. Raspberry Pi (Model B) 22
   3.2. Overview of the BeagleBone Black Development Platform 23
      3.2.1. Programmable Real-Time Unit Subsystem 25
      3.2.2. Touch Screen Controller Subsystem 26
      3.2.3. General Purpose I/O Capability 28
   3.3. External Subsystems 29
      3.3.1. Timing/GPS Module 29
      3.3.2. Power Supply 31
      3.3.3. Signal Acquisition 32
   3.4. Prototype Implementations 37
4. Next Generation FDR Software 47
   4.1. Non-Real Time Operating System 47
   4.2. Operating System Setup and Configuration 52
      4.2.1. Initial Installation and Hardware Support 52
      4.2.2. Network Time Protocol and GPS Daemon 53

4.3. Sampling Considerations 54
4.4. System Software Architecture 59
4.4.1. PRU Layer: FDR_Read.p 61
4.4.2. C Layer: FDR_Calc.c 64
4.4.3. Python Layer: FDR_Main.py 68
4.4.4. Python Layer: FDR_Comm.py 71
4.5. Encrypted Communication 77
5. Unit Testing and Validation 81
5.1. Signal Quality Testing 81
5.2. FNET Integration 109
5.2.1. Super Bowl XLIX 112
5.3. TLS-Enabled Communication Tests 119
6. Conclusions and Future Research 122
6.1. Conclusions 122
6.2. Contributions 123
6.3. Future Work 124
List of References 126
Appendices 134
Appendix A Source Code 135
Appendix B Additional Files 141
Vita 145

List of Tables

Table 2-1 Synchrophasor Message Frames in IEEE 1344-1995 10
Table 3-1 Development Board Feature Comparison 19
Table 3-2 Generation-3 FDR Through-Hole PCB Minimum Design Characteristics 41
Table 4-1 Cyclictest Benchmark Results 49
Table 4-2 Default FDR_Read Parameters 63
Table 4-3 FDR Frame Description - Format 73
Table 4-4 FDR Frame Description - Sizing 73
Table 4-5 C37.118 Data Frame Description 75
Table 5-1 Generation-2 and Generation-3 FDR Frequency Comparison Summary 84
Table 5-2 Signal Analysis – Expected Operational Range 107
Table 5-3 Signal Analysis – Frequency Extrema 108
Table 5-4 Selected Events during Super Bowl XLIX 112
Table 5-5 OpenSSL Benchmark Results 120

List of Figures

Figure 1-1 FDRs Installed in North America 1
Figure 1-2 Overall FNET System Architecture 2
Figure 2-1 High-Level Diagram of Synchrophasor Operation 9
Figure 2-2 Example Alert (Load Shed) Frequency Plot 12
Figure 2-3 Example Alert (Load Shed) Location Map 13
Figure 2-4 C8051F Functional Diagram 14
Figure 3-1 PandaBoard ES 20
Figure 3-2 BeagleBoard-xM 21
Figure 3-3 Raspberry Pi (Model B) 22
Figure 3-4 BeagleBone Black (Rev A5) 23
Figure 3-5 AM335x Sitara Functional Diagram 24
Figure 3-6 PRU-ICSS Functional Block Diagram 25
Figure 3-7 Successive Approximation Register Block Diagram 26
Figure 3-8 TSC-ADC Subsystem Functional Block Diagram 27
Figure 3-9 GPIO Module Integration 28
Figure 3-10 GlobalTop PA6H GPS Module (Top) 30
Figure 3-11 GlobalTop PA6H GPS Module (Bottom) 30
Figure 3-12 Adafruit Ultimate GPS Module Breakout Board 31
Figure 3-13 Non-Inverting Amplifier Circuit 33
Figure 3-14 LM6142 Operational Amplifier CMRR Characteristic 34
Figure 3-15 1st Order Low Pass Filter (RC); 160 Hz 35
Figure 3-16 Bode Plot for 1st Order Low Pass Filter 36
Figure 3-17 Bode Plot for 2nd Order (Butterworth) Low Pass Filter 37
Figure 3-18 PPS Functionality Test Setup 38
Figure 3-19 Protoboard/Breadboard Test Setup 39
Figure 3-20 Generation-3 FDR Through-Hole PCB InstantDFM Result (Top) 41
Figure 3-21 Generation-3 FDR Through-Hole PCB InstantDFM Result (Bottom) 42
Figure 3-22 Generation-3 FDR Through-Hole PCB Minimum Line Width 42
Figure 3-23 Generation-3 FDR Through-Hole PCB Minimum Ring 43
Figure 3-24 Generation-3 FDR Through-Hole PCB Minimum Clearance to Copper 43
Figure 3-25 Generation-3 FDR Through-Hole PCB Minimum Clearance to Plated 44
Figure 3-26 Generation-3 FDR Through-Hole PCB Minimum Clearance to Board Outline 44
Figure 3-27 Assembled Generation-3 FDR Prototype 45
Figure 4-1 Unpatched Linux Installation – Overhead Latency Histogram (µs) 49
Figure 4-2 PREEMPT_RT Patches – Overhead Latency Histogram (µs) 50
Figure 4-3 Xenomai - Overhead Latency Histogram (µs) 50
Figure 4-4 Combined Overhead Latency Histogram (µs) 51
Figure 4-5 NTP Query 54
Figure 4-6 Signal Aliasing 55
Figure 4-7 Digitized Sine Wave (4-Bit Precision) 57
Figure 4-8 Digitized Sine Wave (6-Bit Precision) 57
Figure 4-9 Digitized Sine Waves (12, 14, 16 Bit Precision) 58

Figure 4-10 Overall Data Flow Model for Generation-3 FDR 60
Figure 4-11 FDR_Read Program Flowchart 62
Figure 4-12 FDR_Calc Program Flowchart 65
Figure 4-13 Representation of Sliding DFT 67
Figure 4-14 FDR_Main Program Flowchart 70
Figure 4-15 FDR_Comm Program Flowchart 72
Figure 4-16 C37.118 Frame Transmission Order 74
Figure 4-17 OSI (Open Systems Interconnection) 7 Layer Model 77
Figure 4-18 Asymmetric Cryptography Shared Secret Example 78
Figure 4-19 OpenSSL Key and Certificate Generation 80
Figure 5-1 Doble F6150 Power System Simulator 81
Figure 5-2 ADC Input Capture, 6 Cycles, 60 Hz 82
Figure 5-3 Power Spectral Density of 60 Hz ADC Capture 83
Figure 5-4 Generation-2 and Generation-3 FDR Frequency Comparison 84
Figure 5-5 Calculated Frequency; 59.5 Hz 86
Figure 5-6 Voltage Magnitude; 59.5 Hz 86
Figure 5-7 Phase Angle; 59.5 Hz 86
Figure 5-8 Calculated Frequency; 59.55 Hz 87
Figure 5-9 Voltage Magnitude; 59.55 Hz 87
Figure 5-10 Phase Angle; 59.55 Hz 87
Figure 5-11 Calculated Frequency; 59.6 Hz 88
Figure 5-12 Voltage Magnitude; 59.6 Hz 88
Figure 5-13 Phase Angle; 59.6 Hz 88
Figure 5-14 Calculated Frequency; 59.65 Hz 89
Figure 5-15 Voltage Magnitude; 59.65 Hz 89
Figure 5-16 Phase Angle; 59.65 Hz 89
Figure 5-17 Calculated Frequency; 59.7 Hz 90
Figure 5-18 Voltage Magnitude; 59.7 Hz 90
Figure 5-19 Phase Angle; 59.7 Hz 90
Figure 5-20 Calculated Frequency; 59.75 Hz 91
Figure 5-21 Voltage Magnitude; 59.75 Hz 91
Figure 5-22 Phase Angle; 59.75 Hz 91
Figure 5-23 Calculated Frequency; 59.8 Hz 92
Figure 5-24 Voltage Magnitude; 59.8 Hz 92
Figure 5-25 Phase Angle; 59.8 Hz 92
Figure 5-26 Calculated Frequency; 59.85 Hz 93
Figure 5-27 Voltage Magnitude; 59.85 Hz 93
Figure 5-28 Phase Angle; 59.85 Hz 93
Figure 5-29 Calculated Frequency; 59.9 Hz 94
Figure 5-30 Voltage Magnitude; 59.9 Hz 94
Figure 5-31 Phase Angle; 59.9 Hz 94
Figure 5-32 Calculated Frequency; 59.95 Hz 95
Figure 5-33 Voltage Magnitude; 59.95 Hz 95
Figure 5-34 Phase Angle; 59.95 Hz 95

Figure 5-35 Calculated Frequency; 60.0 Hz 96
Figure 5-36 Voltage Magnitude; 60.0 Hz 96
Figure 5-37 Phase Angle; 60.0 Hz 96
Figure 5-38 Calculated Frequency; 60.05 Hz 97
Figure 5-39 Voltage Magnitude; 60.05 Hz 97
Figure 5-40 Phase Angle; 60.05 Hz 97
Figure 5-41 Calculated Frequency; 60.1 Hz 98
Figure 5-42 Voltage Magnitude; 60.1 Hz 98
Figure 5-43 Phase Angle; 60.1 Hz 98
Figure 5-44 Calculated Frequency; 60.15 Hz 99
Figure 5-45 Voltage Magnitude; 60.15 Hz 99
Figure 5-46 Phase Angle; 60.15 Hz 99
Figure 5-47 Calculated Frequency; 60.2 Hz 100
Figure 5-48 Voltage Magnitude; 60.2 Hz 100
Figure 5-49 Phase Angle; 60.2 Hz 100
Figure 5-50 Calculated Frequency; 60.25 Hz 101
Figure 5-51 Voltage Magnitude; 60.25 Hz 101
Figure 5-52 Phase Angle; 60.25 Hz 101
Figure 5-53 Calculated Frequency; 60.3 Hz 102
Figure 5-54 Voltage Magnitude; 60.3 Hz 102
Figure 5-55 Phase Angle; 60.3 Hz 102
Figure 5-56 Calculated Frequency; 60.35 Hz 103
Figure 5-57 Voltage Magnitude; 60.35 Hz 103
Figure 5-58 Phase Angle; 60.35 Hz 103
Figure 5-59 Calculated Frequency; 60.4 Hz 104
Figure 5-60 Voltage Magnitude; 60.4 Hz 104
Figure 5-61 Phase Angle; 60.4 Hz 104
Figure 5-62 Calculated Frequency; 60.45 Hz 105
Figure 5-63 Voltage Magnitude; 60.45 Hz 105
Figure 5-64 Phase Angle; 60.45 Hz 105
Figure 5-65 Calculated Frequency; 60.5 Hz 106
Figure 5-66 Voltage Magnitude; 60.5 Hz 106
Figure 5-67 Phase Angle; 60.5 Hz 106
Figure 5-68 FNET Server Application – Incoming Data 109
Figure 5-69 FNET Integration – 560 MW Generation Trip 110
Figure 5-70 FNET Integration – 920 MW Generation Trip 110
Figure 5-71 FNET Integration – 500 MW Load Shedding 111
Figure 5-72 FNET Integration – 430 MW Load Shedding 111
Figure 5-73 Event 1: Super Bowl 690 MW Generation Trip – FNET Report 114
Figure 5-74 Event 1: Super Bowl 690 MW Generation Trip – Prototype Measurement 114
Figure 5-75 Event 2: Super Bowl Oscillation – FNET Report 115
Figure 5-76 Event 2: Super Bowl Oscillation – Prototype Measurement 115
Figure 5-77 Event 3: Super Bowl 790 MW Load Shedding – FNET Report 116
Figure 5-78 Event 3: Super Bowl 790 MW Load Shedding – Prototype Measurement 116

Figure 5-79 Event 4: Super Bowl 710 MW Generation Trip – FNET Report 117
Figure 5-80 Event 4: Super Bowl 710 MW Generation Trip – Prototype Measurement 117
Figure 5-81 Event 5: Super Bowl 950 MW Generation Trip – FNET Report 118
Figure 5-82 Event 5: Super Bowl 950 MW Generation Trip – Prototype Measurement 118
Figure 5-83 OpenSSL Benchmark – FNET Server 120

List of Attachments

PEM Key File for TLS Server PEM_Key_Server.txt

PEM Key File for Generation-3 FDR Prototype PEM_Key_FDR.txt

1. Introduction

1.1. FNET Background

The Frequency Monitoring Network, or FNET, is an Internet-based, wide area synchrophasor measurement system developed at Virginia Tech in 2003, and is currently operated by the PowerIT group at the University of Tennessee. It has been providing synchrophasor data since 2004 through the use of single-phase phasor measurement units (PMUs) known as Frequency Disturbance Recorders (FDRs) [1]. These are low-cost, high-accuracy devices which operate at nominal wall voltage – 120V in the United States.

Just as a PMU would, each FDR measures a voltage signal and determines the phase angle, amplitude, and frequency of the waveform. Precise timing across a wide area is made possible through the use of the Global Positioning System (GPS), which allows for synchronized measurements. FDRs send 10 time-stamped messages per second – one every 100 milliseconds – to servers, which act as phasor data concentrators (PDCs) at the University of Tennessee, Knoxville. At present, there are approximately 200 FDR units in North America and another 70 in other countries worldwide. Figure 1-1 shows the FDRs currently deployed throughout the North American power system.

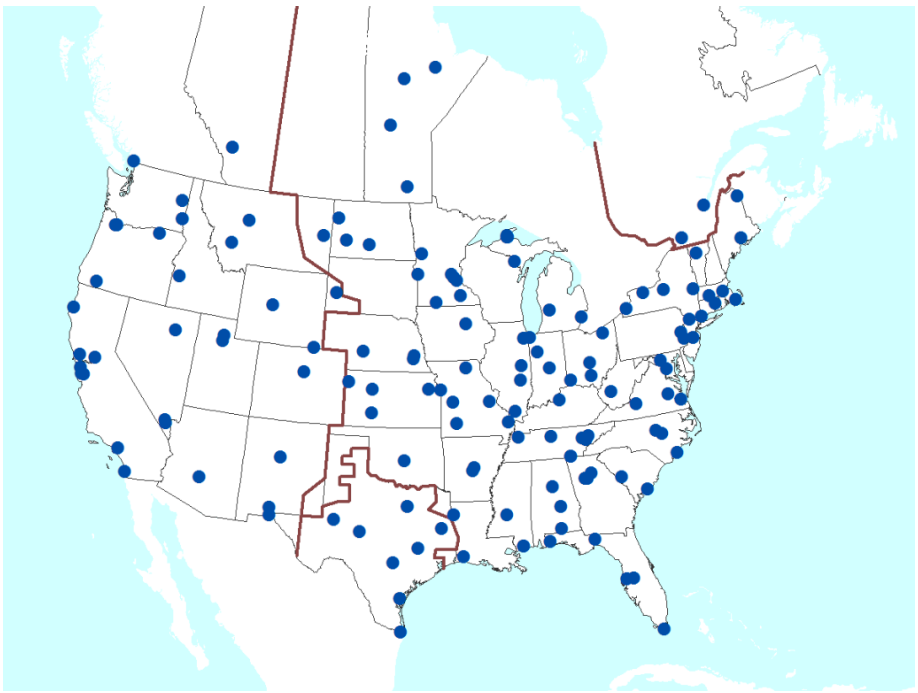


Figure 1-1 FDRs Installed in North America

The units themselves simply compute and report the characteristics of the waveform – they do not perform advanced analysis of any kind. On the FNET servers, applications run on both the streaming and archived data – "online" and "offline" applications, respectively. These applications allow the PowerIT group at UTK to observe and characterize power system problems, visualize power system events, and perform background signal analysis [2]. A high level diagram of the overall FNET system architecture is shown below in Figure 1-2.

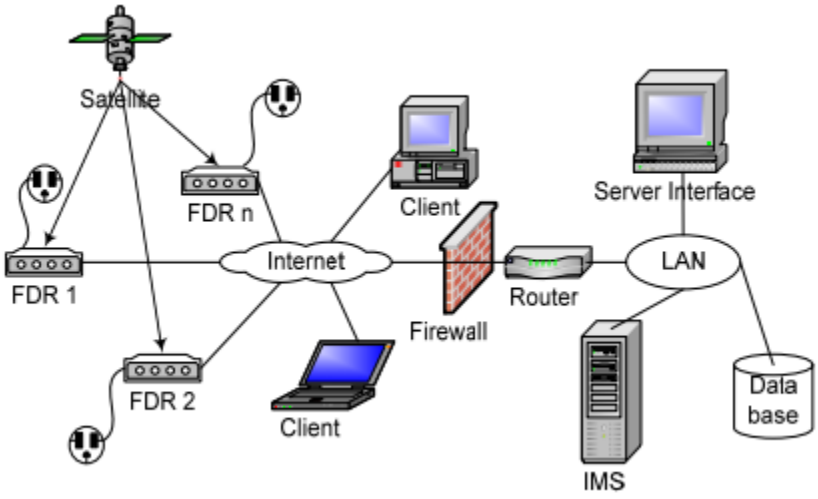


Figure 1-2 Overall FNET System Architecture

Although a major part of the FNET system, the current generation of Frequency Disturbance Recorders was developed in 2006 and there have only been incremental improvements to them over their lifetime. This research presents a new FDR design which utilizes more modern, off-the-shelf hardware combined with a modular design in order to reduce costs and increase capability while also allowing for improved future expansion and maintainability.

1.2. Motivation and Objective

Fundamentally, the measurement capability of FNET is determined by three major factors – the quality of the measurement data, the total number of measurement points, and the physical location of those measurement points. These factors do not operate in isolation – FDRs which send faulty data are of no benefit regardless of how many exist or where they are located. Likewise, a very small quantity of

FDRs which send high quality data is certainly useful; however, once a baseline data quality is achieved, the optimal way to increase overall monitoring capability is to have more FDRs placed in proper locations. Although an FDR is inexpensive compared to a commercial PMU [3], they are still specialized pieces of hardware costing upwards of $1500 USD. This cost is not the only obstacle to increased deployment; as the system grows, there are additional logistics concerns, as increased deployment brings with it higher maintenance and configuration requirements.

The currently deployed Generation-2 FDRs perform their function accurately and correctly, and allow FNET to continue to monitor the power grid and analyze disturbances. However, deploying an FDR requires the unit to be calibrated and configured beforehand. Specifically, the network chip in each device must be flashed with the proper configuration and then physically inserted into the unit. This unit is mailed to the host with all settings preconfigured at UTK. If the network topology changes on either the host's end or UTK's, or the unit simply requires an update, it must be mailed back and reconfigured completely. While this ensures that the units can be hosted easily, it means that troubleshooting a particular unit or configuration can take a considerable amount of time.

These factors are the main reasons for the development of a new FDR – in short, the fundamental motivation is to increase the capability, quantity, reliability, and maintainability of the FDR itself in order to expand the monitoring abilities of FNET. The relatively high cost combined with the configuration and maintenance requirements on the current units limits how quickly new units can be deployed. Additionally, any reconfiguration requires mailing a unit multiple times between UTK and the host. In the event of changes to network topology, this should be unnecessary. Therefore, to substantially increase the monitoring capability of FNET, a cheaper and more maintainable unit must be developed. The purpose of this research is exactly that – by utilizing off-the-shelf components and taking advantage of the increase in capability offered by modern ARM-compatible microprocessor hardware, a new until will be cheaper to produce while simultaneously being easier to configure and maintain. As a part of this, the new units will allow for remote management, reducing the need for transportation of the unit itself.

Simply using new hardware by itself does not achieve these goals. The change in platform to a modernized RISC microcontroller and a General Purpose Operating System (GPOS) is a significant change to the state of the art. Rather than utilizing multiple chips with their own firmware, this unit utilizes a unified operating system and is essentially a fully-capable personal computer. Remote management requires that the new units are capable of secure communication, in order to prevent external

tampering. Compared with the existing Generation-2 FDR, the advances of the new ARM and Linux-based unit are summed up as follows:

- Higher and configurable sampling rate (3600+ vs 1440).
- Lower hardware cost (< $200).
- Support for remote management and reconfiguration.
- Capability for two-way encrypted communications.

In order to develop this proposed Generation-3 FDR, a number of questions must be answered to both determine feasibility of a new unit and then establish proper functionality. Careful selection of the components is necessary from a design point of view, but the move to an ARM microprocessor and a GPOS requires that some fundamental obstacles be overcome in order for development to be feasible. For example, mass-produced hardware has variations in the silicon which must be compensated for in order to establish proper software timings. Execution of a real-time process on a Non-Real Time Operating System (non-RTOS) is another concern – determining the feasibility of this and then developing algorithms to make this possible is required. With regards to measurements, the limit of oversampling is based on both hardware and software limitations, so these limits and processes are evaluated accordingly. Additionally, while the hardware has the capability for secure communications, an evaluation of the extra bandwidth and computational resource usage for such communications is performed to see if the scheme could be practically realized in the near future. As part of this effort, a prototype printed circuit board (PCB) version of the device is created and tested alongside the existing FNET system.

1.3. Organization of Study

This dissertation is divided into six Chapters, each of which discusses a specific aspect of the next generation FDR. Chapter 1 has presented background information on the FNET system and has explained the motivation behind the work presented in this document. The advances to the state of the art have been outlined, as well as an overview of how the work accomplishes these objectives.

Chapter 2 is a review of existing technology and standards. This includes the development history and technology of synchrophasor measurement, and the standards which are applied to modern phasor measurement units. The history and impetus behind FNET and the current generation FDR are

discussed. The capabilities that are afforded with the existing technology as well as some of the limitations and areas for potential improvement are explained.

Chapter 3 presents the hardware research and design for the next generation FDR – hardware requirements moving forward, challenges and tradeoffs which must be addressed, and the various subsystems are discussed. This includes the microcontroller itself, the GPS and ADC subsystems, power conversion hardware, filtering, and the finished board schematics with potential variations.

Chapter 4 is a discussion of the software research design for the new FDR. The operating system itself, challenges related to non-real time operation, system architecture, driver software, and algorithm code are presented in detail. The requirements for secure communication are determined for both the unit itself and the FNET servers.

Chapter 5 is dedicated to unit testing. Test procedures and results are presented, which include the current generation FDR and FNET system as benchmarks, where applicable. The secure communication capabilities of the unit are tested using software to simulate multiple simultaneous connections. Both real and simulated systems are used in these testing and validation procedures.

Chapter 6 presents conclusions based on the development and testing. Contributions to the state of the art are revisited. Lastly, future work and potential capability expansion for the new Generation-3 FDR is discussed.

2. Review of Existing Technology and Standards

This chapter presents an overview of the development of synchrophasor technology. This includes the mathematical formulation and how it relates to power system dynamics, as well as a history of phasor measurement devices and their applications. The key components common to all synchrophasor devices and the relevant standards are discussed. Lastly, the architecture of the Generation-2 FDR is presented to establish their current capabilities and what their limitations are with respect to FNET.

2.1. Definition of Synchrophasor

Phasors as a mathematical concept have existed since Euler, and are a way of representing a sinusoidal, time-varying quantity. As given by Euler's Rule:

$$Ae^{j(\omega t+\theta)} = Acos(\omega t + \theta) + jAsin(\omega t + \theta)$$

where $j = \sqrt{-1}$, ω is angular frequency, and θ is the phase angle. Given an equation in this form, a phasor is a shorthand method of writing the magnitude and phase angle of the wave, as the frequency of the wave is approximately constant. Under most circumstances in a power grid, this holds true – the grid operates at a nominal frequency, 50 or 60 Hz in most countries. Therefore, when dealing with the voltage and current waveforms found in a modern power system, the following convention is used instead:

$$Ae^{j(\omega t+\theta)} = A\angle\theta$$

In a strict mathematical sense, this is a loss of information, as the original equation is frequency dependent. However, given the highly consistent nature of the electric power system, this form allows for frequency to be calculated based on the change in phase angle. This notation and technique was initially standardized by Charles Proteus Steinmetz and is commonly used in AC circuit analysis [4]. The term "synchrophasor" arises from the use of a synchronized timing source used in conjunction with repeated phasor measurements. The term itself does not refer to a separate mathematical construct, simply a technological application of phasor measurements. Modern standards which apply to synchrophasor devices are explored later in this chapter, as the precise definition has shifted in order to accommodate and integrate new technological developments.

The reason that repeated phasor measurements can be used to monitor the power system is that fundamentally, the frequency of the waveform (ω) is the rate of change of the phase angle (θ), or

$\omega = \frac{d\theta}{dt}$. At 50 or 60 Hz, this means a total change in angle of 18,000 or 21,600 degrees each second, respectively. If the measured quantity does not match these nominal values, then the average frequency is known to be off-nominal for that second and can be determined. While this calculation is straightforward, performing this across a wide area at high speeds while maintaining a synchronous time source is the challenge in creating a phasor measurement unit [5].

2.2. Phasor Measurement Unit Overview

To explain the purpose behind developing a new FDR, it is important to first understand the background of the devices themselves, as phasor measurement units are a relatively new technology compared to many other devices used in modern power systems. While the FDR is not intended to be a commercial-class device and will not be installed on high voltage segments, there are standards for communication and signal quality that have been created for PMUs which should be considered. Therefore, this section will provide an overview of PMU technology, including development history as well as current standards and practices.

The concept of a phasor measurement unit as a power system device originates in the 1960's, and the PMU can be considered a descendant of the symmetrical component distance relay (SCDR). At this time, fully computerized relays were infeasible, but the theoretical value of using wide area measurements and state estimation to analyze a power grid had already been established. Although computational power increased steadily throughout the 1970's, a key obstacle remained for wide-area measurement technology – clock synchronization to a common time source for separate areas. In the early 1980's, researchers in Canada and Switzerland developed techniques for using synchronized clocks to accurately measure voltage phase angle [6, 7].

The full realization of the phasor measurement unit as a device originated at Virginia Tech in the mid 1980's. These original devices directly monitored high voltage segments of the power system. Given the high cost of the device – in terms of hardware and installation – this approach was very sensible; installing the phasor measurement device at substations results in a clean measurement signal while typically allowing for indirect measurement of a greater number of buses [8]. Following the creation of these prototype PMUs in 1988, Macrodyne Co. commercialized the device and added multiple innovations, such as one analog-to-digital converter (ADC) per input channel and multiple modem interfaces for remote access [9, 10].

The high installation cost and dedicated communication infrastructure requirement of PMUs meant that having a sufficient number of them to perform event analysis would be very costly. It is these factors that led to the PowerIT group's concept and development of the FDR at Virginia Tech. As mentioned, an FDR unit connects to the distribution level – wall sockets at 120V, nominally. Compared to a substation installation, the voltage signal contains a higher level of noise. However, the device does not have the high installation costs as no specialized tools or knowledge are necessary, and no dedicated communication infrastructure must be in place [11].

As an FDR is essentially a single-phase PMU, there are several core subsystems which are common to both devices, as they are essential for wide area monitoring. These include:

- A timing subsystem; responsible for maintaining a synchronous clock.
- A measurement subsystem which acquires the raw signal information.
- A computation subsystem; necessary to compute the phasor data from the raw measurements.
- A communication subsystem; necessary to transmit the data for real-time monitoring.

Also of high importance is the quality of each of these subsystems – GPS is utilized in FDRs and many commercial PMUs because of the highly accurate Pulse-Per-Second capability. The PPS is the critical component of the timing – establishing exactly when a second begins across the entire system ensures that all measurements are meaningful with respect to each other. The quality of the measurement system directly correlates with the quality of the data, so noise must be filtered as much as possible to ensure a clean, accurate signal. The computation subsystem much be of sufficient power to compute phasor data in real time. The constant progress in the field of processor development has made this a much easier requirement to meet, although the underlying algorithm still matters as well. Lastly, the amount of data which can be acted upon directly depends on how much data can be communicated in the first place and how quickly it can be received. For PMUs installed at a substation, this communication may be local over a serial bus, but ultimately the data stream is transmitted over a higher bandwidth channel to the operating center. For FDRs, the data stream is transmitted directly over the internet instead. A high-level system diagram of this operation is provided in Figure 2-1. Note that this figure shows necessary systems and interactions, but not any specific implementation.

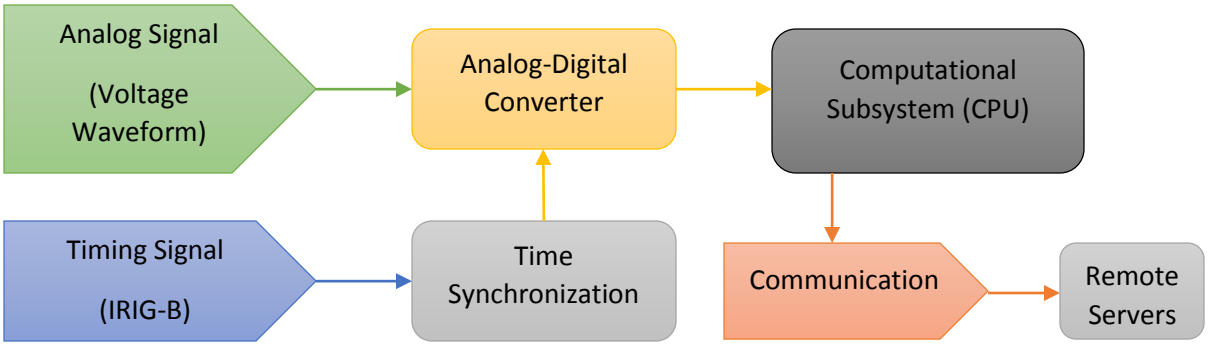


Figure 2-1 High-Level Diagram of Synchrophasor Operation

2.2.1. Applicable Standards

The purpose of this subsection is to provide background on the standards which have been developed specifically for synchrophasor devices. As computation and telecommunication technology have proliferated rapidly within the last two decades, these standards have been revised multiple times. This background is important to provide context as to FNET's adherence to standards when possible, and elucidate why certain standards are either inapplicable or not presently followed.

Once commercial PMUs were being developed by multiple manufacturers, the need for standardization on multiple fronts became apparent. The original synchrophasor standard [12] was developed in 1995 to ensure that consistent definitions and practices were applied across devices developed by all manufacturers. These practices included the digital representation of a phasor quantity, requirements for a synchronizing source, time-tagging of data, configuration schema for phasor measurement units, and a message format. Of particular interest is that a standardized set of message frames for synchrophasor devices is defined, as shown in Table 2-1.

Of particular historical note is the allowance for "user defined" frames. This flexibility within the standard allowed synchrophasor device manufacturers to include additional features into their units to differentiate them on the market. As a result, as PMU technology has matured, synchrophasor measurement capability has been integrated into other intelligent electronic devices, such as relays and reclosers [13]. From a power system perspective, the critical frame, and the one used most often, is the data frame. By unifying this frame, the standard ensured that wide-area measurement on a high level would be vendor-agnostic for the most common applications.

Table 2-1 Synchrophasor Message Frames in IEEE 1344-1995

Bits 15-14-13	Frame Type	Definitions
000	Data Frame	Bits 0-12: sample count, 0-8191
001	Header Frame	Bit 12: first frame flag (0: first frame of header file, 1: continuing frame) Bit 11: last frame flag (0: not last frame, 1: last frame of tile) Bits (0-10: frame count for file, 0-2047 (first frame is count 0)
010	Configuration Frame	Bit 12: first frame flag (0: first frame of header file, 1: continuing frame) Bit 11: last frame flag (0: not last frame, 1: last frame of tile) Bits (0-10: frame count for file, 0-2047 (first frame is count 0)
011-100	Reserved for Expansion	Bits 0-12: to be assigned
101-111	User Defined	Bits 0-12: user defined

This initial standard for synchrophasors properly laid down the groundwork for all of the devices at the time of its writing. However, the timing of the standard's release and the data formats referenced must be investigated to understand the nature of the standards which succeeded IEEE 1344. In the mid 1990's, personal computer and telecommunication technology underwent a surge in relative accessibility. Strictly speaking, transistor density has colloquially improved at a consistent rate for most of a century [14]; this rapid adoption of technology and increased deployment of information technology was essentially a "step change" in the norms of computing. It is during this period of time that 32-bit desktop computing became standard rather than the exception. This is important because the data formats used in the synchrophasor standard are geared at interoperability with these systems. Specifically, the 8-byte floating point format [15] allowed for higher precision data streams. Therefore, the updated standard, IEEE C37.118 was created in part to address the new standards of communication (while keeping backwards compatibility) and also to further standardize allowable error and compliance testing [16]. At the time of the Generation-1 FDR's development, this revised standard was not yet in place. However, the units were designed to communicate with 32-bit architecture. As a result, FNET itself utilizes a different frame than the versions set forth in the synchrophasor standard. An in depth exploration of this frame structure can be found in Section 4.4.3.

In the late-2000's, the continued improvements to microprocessor and telecommunication technology meant that once again the synchrophasor standard would be revised. The integration of broadband connectivity meant that synchrophasor analysis could reliably be applied to systems under

dynamic conditions. The 2005 standard only applied to power systems in a steady state. Additionally, the revision split the standard into two distinct yet related pieces – a standard for synchrophasor measurements [17], and a standard for synchrophasor data transfer [18]. These standards were officially released in 2011.

At present, the FNET system – and Generation-2 FDRs – adhere to most of the synchrophasor measurement standard as it applies to measurement quality and calibration. Specifically, the system attempts to follow the guidelines for Total Vector Error (TVE) and measurement data reporting rate. As FNET is not operated by a transmission owner, and the units act in a different capacity from substation installed PMUs, the operational error standards are not used. This in part relates to the data and communication framework, of which FNET uses a custom frame solution. This is in part due to the communication path being strictly one-way – no command signals can be sent to an FDR, and there is no other inbound remote communication allowed. The configuration flexibility offered by a GPOS and the associated file-system will allow Generation-3 FDRs to send IEEE C37.118-compliant packets as an option. This is explained in more detail in Section 4.4.3.

2.2.2. Synchrophasor Applications

Having knowledge of the voltage magnitude and phase angle is very powerful in steady-state analysis of a power system, and is the basis behind the practice of state estimation [19]. Initially, when synchrophasor technology was first being deployed, this was the most well understood use for the technology, as evidenced by the emphasis on steady-state operation in the initial standards. However, much research has been performed [20, 21, 22, 23] which shows the value of phasor measurements in analyzing a grid in a dynamic state. The high rate of status updates from synchrophasor devices allows for critical portions of the system to be monitored at a rate much faster than traditional SCADA has previously allowed. Knowledge and control of these critical points can ensure system stability, and the need for this is highest during a power system disturbance [24]. Once synchrophasor measurement technology started becoming more prevalent, one of the first wide-area applications was real time monitoring. An excellent example of this, and one of the FNET system's major features is event monitoring and e-mail alerts. These alerts provide information on the type of disturbance (load shed, generation trip, or oscillation) and location of that disturbance through point triangulation. An example of the information provided by alerts is shown in Figure 2-2 and Figure 2-3.

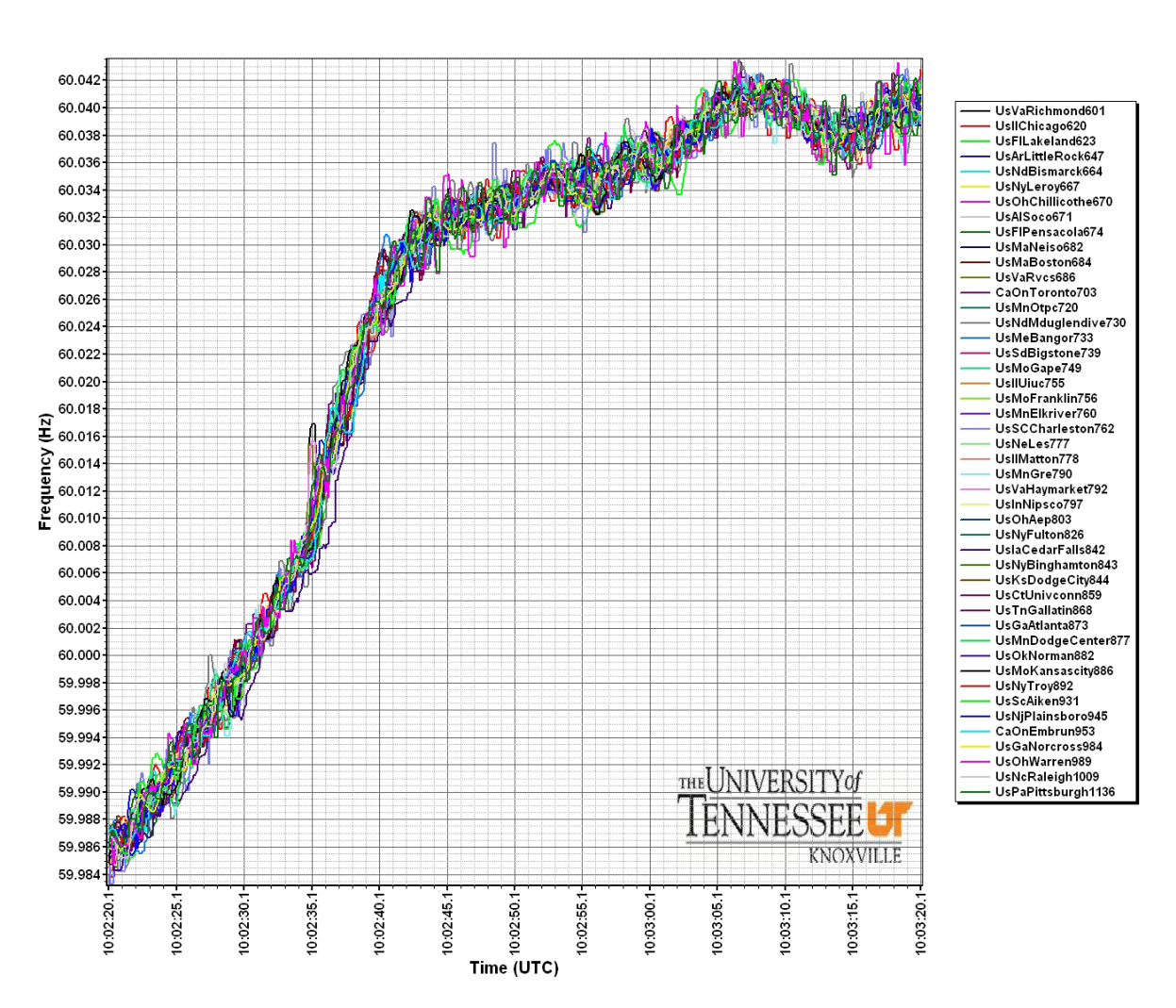


Figure 2-2 Example Alert (Load Shed) Frequency Plot

Even when not utilized in a real time manner, the precision of synchrophasor data means that there are a number of applications which benefit from an archival data set. These include the characterization of power system swings, inter-area oscillations [25], predictive load forecasting, anticipation of societal-influenced disruptions [26], and authentication of audio recordings [27].

Additionally, archived synchrophasor data streams have a number of uses, such as direct and indirect power grid model development, automatic classification of power system events, and impact studies for policy decisions [28]. These applications are not time critical, although the understanding of system dynamics provided by synchrophasors allows for such studies to be conducted. It should be

noted that the effectiveness of many of these applications scales well with the number of data points which can be utilized, reinforcing the stated motivation behind developing a new FDR.

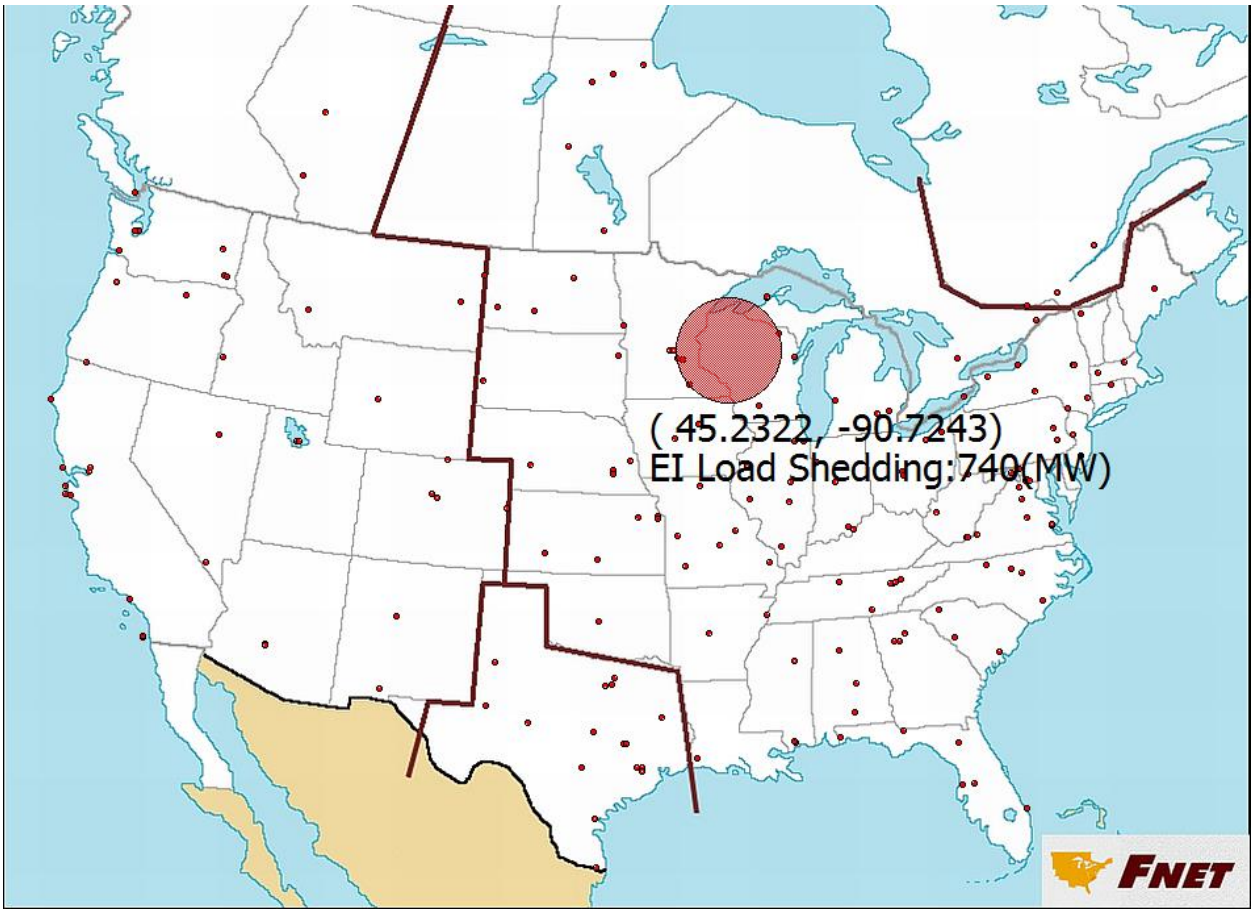


Figure 2-3 Example Alert (Load Shed) Location Map

2.3. Current FDR Architecture

The current Generation-2 FDR is most easily understood as a modular collection of individual subsystems, each of which has a specialized chip and instructions which are flashed onto the memory. This modular architecture is made clear by the fact that the device consists of multiple separate PCBs – a main board with power and processing, and daughter boards for GPS, networking, and ADC/DSP functionality. As one would expect, this makes the unit a specialized embedded device with no underlying operating system. The microcontroller installed in the unit is a Silicon Labs C8051 [29], with a core clock speed of 25 MHz. A reference diagram is provided in Figure 2-4. It includes multiple ADC

subsystems, although they are unused in the current FDR. Instead, an external ADC with higher precision is used and then interfaces with the processor via the digital I/O. The microcontroller is responsible for interfacing with the other subsystems – the daughterboard ADC and communication, and constructing the packets which are sent. This follows the high level general device framework shown in Figure 2-1.

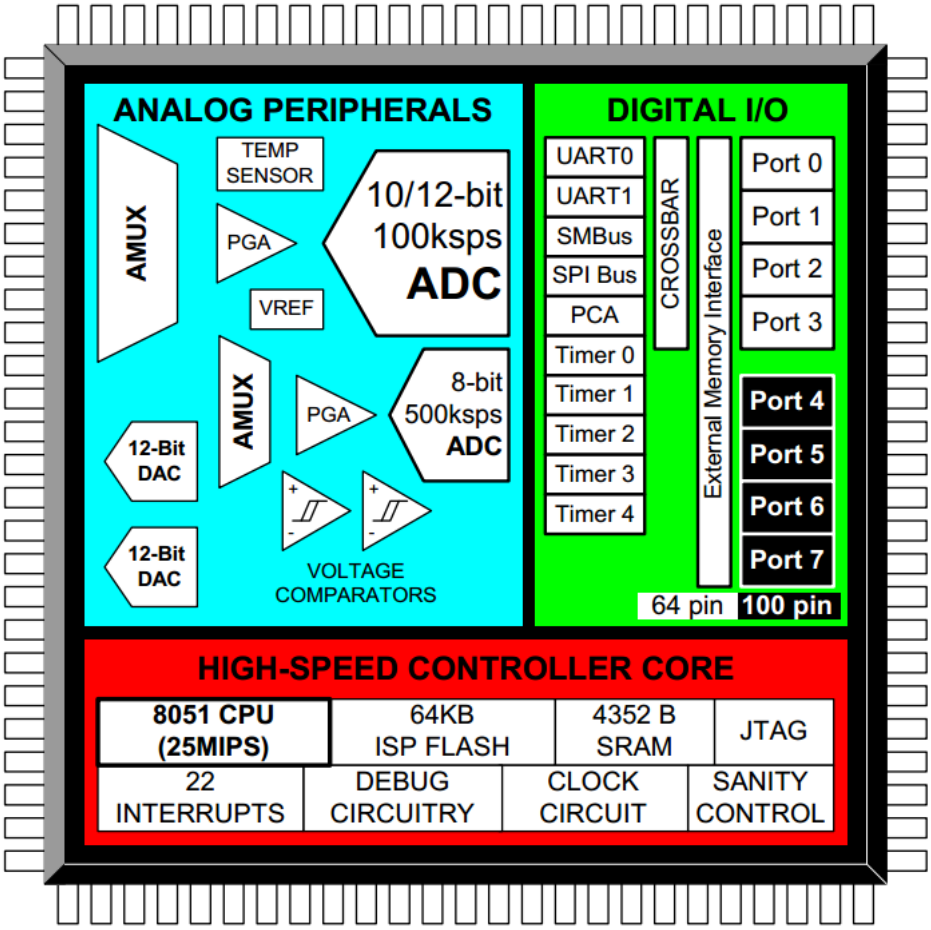


Figure 2-4 C8051F Functional Diagram

The ADC/DSP daughter board contains an Analog Devices AD7865 ADC [30], which is responsible for converting the raw analog input into a digital signal. As it is a 14-bit ADC, the resulting internal numbers range between 0 and 16383. This chip requires +/- 5V logic to operate, so onboard power conversion is used to reduce the levels from the +/- 12V supply present on the main board. Also present on this board is the DSP chip used to perform the frequency. Referring to Figure 2-1, this component is actually part of the computational subsystem, despite being separate from the major processor. This

chip is a Texas Instruments TMS320 LF2407 DSP [31] – a commonly used DSP chip which operates at the same logic level as the ADC. The clock speed of this chip is 30 MHz – faster than the FDR's main processor. This is not the reason for its inclusion; the chip is specially designed for DSP computations and is as such more efficient in this capacity.

The packaged data stream output from the main processor is in the form of serialized bits. To send this data across the internet and back to the central servers, a mediator is required which converts serial to Ethernet communication. In the Generation-2 FDR, this function is performed by the NE-4100T – a standalone module designed for such communication [32]. It is a self-contained daughter board and inserts directly into the FDR's main board. The specific module used does not contain an RJ45 Ethernet port – the port is built onto the main board and the signal is simply converted in the module itself.

The GPS module for the Generation-2 FDR is also contained on a daughter board. The onboard chip for this is the Motorola M12+ [33]. Specifically, the variant used is the "timing receiver," which allows for precise 1PPS output. This daughter board is powered from the FDR's 3 VDC rail. The receiver supports both binary and NMEA 0183 [34] data formats, although the PPS is the primary functionality. The board does not possess an on-chip ceramic microstrip (patch) antenna, and an external antenna is required for use. The antenna's required voltage can vary from 2.5V to 5.5V, so the board also contains necessary power conversion circuitry. Total power draw for the board is just under 750 mW, with antenna. Backup power is possible, although to handle the comparatively large (up to 80 mA) draw from the antenna, a comparatively large energy storage must be used.

As mentioned, the current Generation-2 FDR has some limitations, in terms of both the hardware itself and the logistics involved with deploying and maintaining them. These two categories are not entirely separate. Although the current method of configuration and deployment of FDRs can result in a difficult networking setup, the reliability of the hardware itself also adversely affects the maintainability of the system.

2.3.1. Timing/GPS Subsystem Limitations

The current unit has some minor limitations with regards to signal timing and GPS. The sampling rate – 1440 Hz is optimized for the North American power grid, as this sampling rate is evenly divisible by 60 Hz. However, when a unit is in Europe, the sampling rate becomes 28.8 Hz rather than 24 Hz. Since packets are sent every $1/10^{th}$ of a second, this is acceptable, as an integer number of cycles is used in the analysis window. However, the resample algorithm, which uses an additional cycle of data, must be

changed to accommodate this disparity. This means that the firmware of the unit is specific to the grid frequency and cannot be moved to a grid with a different nominal frequency. Utilizing a higher sample rate – such as 1800 Hz – would allow grids of either frequency to be analyzed without any special casing logic in the algorithm. The nominal voltage level, in comparison, can trivially be solved with a hardware switch and extra input resistance, as is the case on existing computer power supplies which support different voltage levels.

Presently, the major limitation with the M12+ GPS chip is that it is no longer in production. This means that the design will have to migrate to a new chipset, which may require a redesign of the unit to accommodate the new connections. The unit also sporadically has difficulty initially acquiring signal from a sufficient number of satellites, which means that new units can take up to 48 hours to connect and begin streaming data. Aside from this, the relatively high power requirement to power the GPS daughter board prevents reliable backup functionality in the event of a power outage. The combination of these two factors means that it can take a long time to initially set up a new unit, and if the power to the unit is lost, there is no on board memory of satellites to assist in the reboot process.

2.3.2. Computational Limitations

Currently, the Generation-2 FDR requires multiple processors, none of which is independently powerful enough to handle the communication and signal processing at the same time. As mentioned, the current sampling rate is 24 samples per cycle (at 60 Hz), or 1440 Hz. This limit is in part due to the throughput limitations on the individual chipsets in the unit. This also places a limit on how much data can reasonably be transmitted from one chip to another. While the clock rates on the chips are 25 and 30 MHz for the central processor and DSP, respectively, the serial communication between them is far more limited. This is the effective bottleneck on communication – it limits how much data the DSP can send the CPU, and how much the CPU can send the network module. This serial communication is an order of magnitude slower than that of the processor clocks, and synchronization to a common bus means that a serial clock speed compatible with both chips must be utilized. While not an insurmountable obstacle, this limits which DSP modules may be compatible with the devices main board. This restriction on throughput also means that computations which must be split over both the DSP and CPU are restricted to the speed of the serial bus. At present, there is no such computation performed on the device, but this architecture may prevent more advanced algorithms from being applied which would require logic not present on the DSP alone. Additionally, the DSP does not have

direct access to the memory of the main processor, so is limited to the small buffer of data that it can contain within itself.

2.3.3. Logistical Limitations

When referring to logistical limitations with regards to the Generation-2 FDR, it must be noted that these limitations are defined as issues related to the process of deploying and redeploying units. An obvious factor and key motivator for development of a new device is the cost of the hardware itself. Directly reducing this cost allows for the manufacture and purchase of more units, so the benefit correlates well. However, the reliability of the hardware and the configuration process sometimes result in units which have been deployed and are unable to communicate with FNET servers. These units then must be mailed back and require one or more redeployments to correctly configure the network settings or replace an internal component which has ceased functioning. Notably, the DSP chipsets have an unusually high failure rate – approximately 1 in 30 – and the daughter board must be returned. This failure prevents firmware from being downloaded to the chip, and happens in both existing and newly constructed devices.

One particular issue which the PowerIT lab currently experiences has to do with the timing subsystem during one of these maintenance operations. As mentioned, the GPS module sometimes will take upwards of 48 hours to locate satellites. Without synchronization and a PPS signal, the unit cannot be tested. This introduces a comparatively large delay in diagnosing and troubleshooting units which have been sent back to the lab. Additionally, if a component in the unit has failed, such as the DSP board, the unit must be powered down and the component replaced. In a worst-case scenario, the GPS will again require a long time to synchronize for the device to be tested.

3. Next Generation FDR Hardware

Although Chapters 3 and 4 discuss the hardware and software research and development, respectively, this delineation is for clarity of organization. The research process involved both aspects simultaneously, as capabilities of the hardware influenced what then became feasible in the software, and vice-versa. The end result is a merger of the two, as various low-level functions are required to develop a device that can operate in a time-critical manner while lacking an underlying operating system which natively supports it.

3.1. Embedded System Hardware Requirements and Considerations

In designing the next generation FDR, specifications must be determined according to what is absolutely critical for the functionality of the device and also what capabilities can be provided through additional hardware. In the case of a Frequency Disturbance Recorder, a high-accuracy time signal is required as well as a means of analyzing the frequency and then returning that measurement to the FNET servers. Strictly speaking, a built-in GPS receiver is not required – simply an accurate PPS signal. Additionally, while communication over the Internet is necessary, the interface does not necessarily have to be an RJ-45 (Ethernet) port. Therefore, in addition to the functional requirements, flexibility is also necessary in a new FDR. The hardware must be somewhat modular and reconfigurable to allow for a design which can be incrementally improved or changed as needed.

To ensure maximum hardware compatibility and capability, the ARM microcontroller architecture is used in the Generation-3 FDR. Presently, the ARM architecture is the most commonly used embedded framework found in mobile and low-power devices, accounting for nearly 60% of the world's embedded devices [35, 36, 37]. This level of adoption ensures that significant hardware and software support is available for ARM microcontrollers, and that the supply chain is robust. Among performance considerations, availability and longevity of parts is also a key factor in determining next generation hardware. Due to ARM's popularity, a number of development boards have been made available which combine an ARM microcontroller and some peripheral hardware onto a single PCB. These self-contained computers typically have much of the required hardware and interconnection capability necessary for embedded system development.

Given these factors, all of the embedded options that are considered in the assessment are ARM-based, although with processor variants. The four boards considered for this project were the PandaBoard ES, BeagleBoard-xM, Raspberry Pi Model B, and BeagleBone Black (Rev B). This list of four is

substantially pared down from a comprehensive list of single-board computers [38]. These four boards are all development platforms which have support communities – an absolute necessity for troubleshooting and debugging the hardware and software features. Each of these boards offers different integrated hardware and external hardware connectivity. Table 3-1 provides a high level overview of the basic features and cost of each of these boards. These prices and capabilities are accurate as of the time of this writing, although newer revisions may have slightly different specifications. The table only compares the exact models stated in this section.

Table 3-1 Development Board Feature Comparison

Board	Processor Type	Speed	RAM	ADC	GPIO	Price
PandaBoard ES	Dual Core ARM Cortex-A9	1.2 Ghz	1 GB	Yes	28	$182 USD
BeagleBoard-xM	AM37x (ARM Cortex-A8)	1 Ghz	512 MB	Yes	28	$149 USD
Raspberry Pi (Model B)	ARM1176 (ARM11)	700 Mhz	512 MB	No	15	$35 USD
BeagleBone Black	AM335x (ARM Cortex-A8)	1 Ghz	512 MB	Yes	46	$55 USD

At a high level, all of the boards are capable of functioning as a next generation Frequency Disturbance Recorder. The key differences are the additional capabilities offered by each board – an ARM processor alone is sufficient but not optimal. Common to all of these platforms is Internet connectivity via an Ethernet port, USB support, and open-source schematics. A Bill of Materials and original schematic can be found on each of the home websites for these boards. Ultimately, the BeagleBone Black was selected as the development platform. An in-depth look at the features and capabilities of the BeagleBone Black is presented in Section 3.2.

3.1.1. Pandaboard ES

The Pandaboard ES is the latest iteration of the PandaBoard series – a low-cost general purpose embedded development platform capable of HD 1080p video output and internet connectivity. The processor is a Texas Instruments OMAP4 series (4460 specifically), which is an ARM-instruction set compatible microcontroller with additional features and interfacing. It has an LCD expansion header and multiple ports for external output [39]. It does possess general purpose I/O ports (GPIO), but lacks onboard storage. The specifications for the processor indicate that it is capable of onboard ADC, but documentation from Texas Instruments states that this functionality is not exposed for external use – it

is purely for manipulation of audio signals as a subprocessor. There is no real-time functionality available – an external circuit consisting of the ADC and a real-time clock would be required for real-time sampling. Figure 3-1 shows a reference picture of the Pandaboard ES.

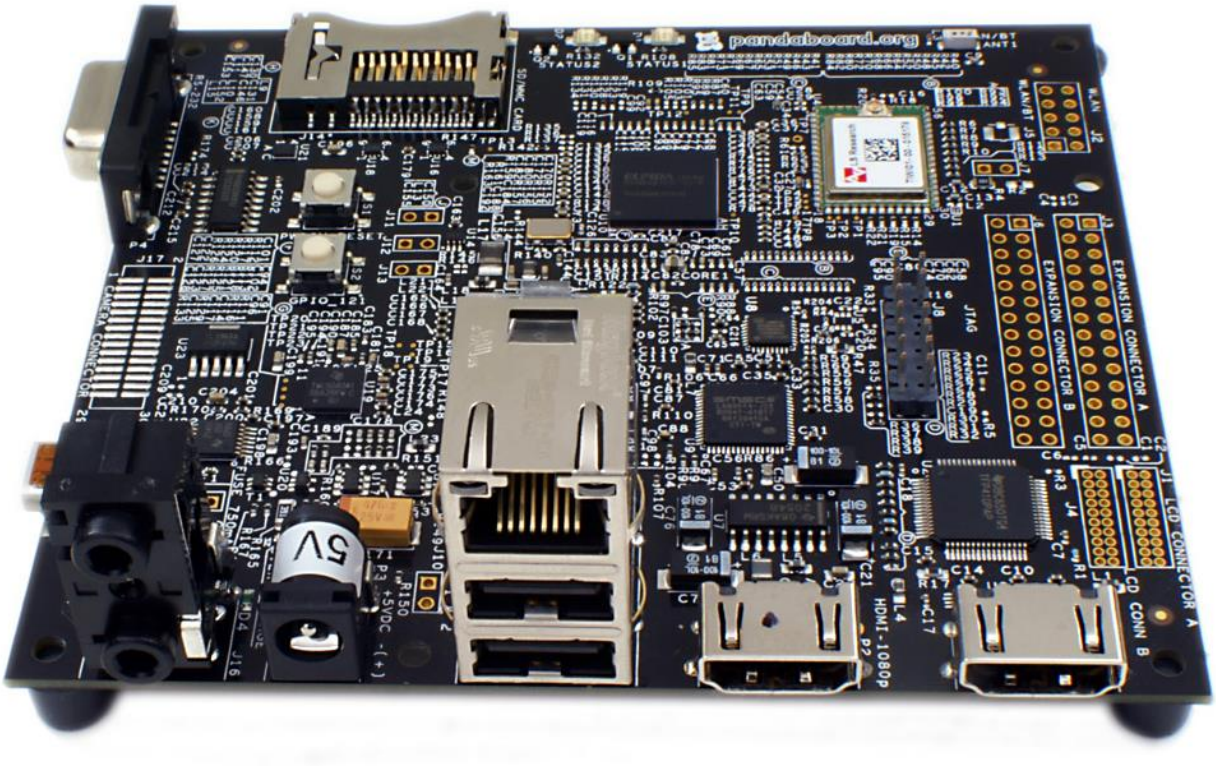


Figure 3-1 PandaBoard ES

The PandaBoard ES is the most powerful of the four platforms, but is not particularly well suited for real-time sensing applications. The platform is intended for mobile or multimedia application development, as the board contains considerable video and audio processing resources. Unfortunately, the lack of accessibility to the processor features combined with the relative lack of control documentation makes the PandaBoard ES a suboptimal choice for a next generation FDR.

3.1.2. BeagleBoard-xM

The BeagleBoard-xM is the latest entry in the BeagleBoard series – the predecessor to the PandaBoard as a general purpose embedded development platform. It is also capable of video and audio playback, although at lower quality, General Purpose I/O, and internet connectivity. As expected,

the specifications and cost are lower, although as it is an older board, there is more documentation available. The BeagleBoard utilizes a Texas Instruments OMAP3 series (3370 specifically) processor; the direct predecessor to the OMAP4 found in the PandaBoard. Therefore, the same basic feature set is present, and there is more than enough RAM to allow for FDR functionality. As with the PandaBoard, there is no onboard ADC, which means that an external ADC chip and real-time clock are required. Figure 3-2 is a reference picture of the BeagleBoard-xM.

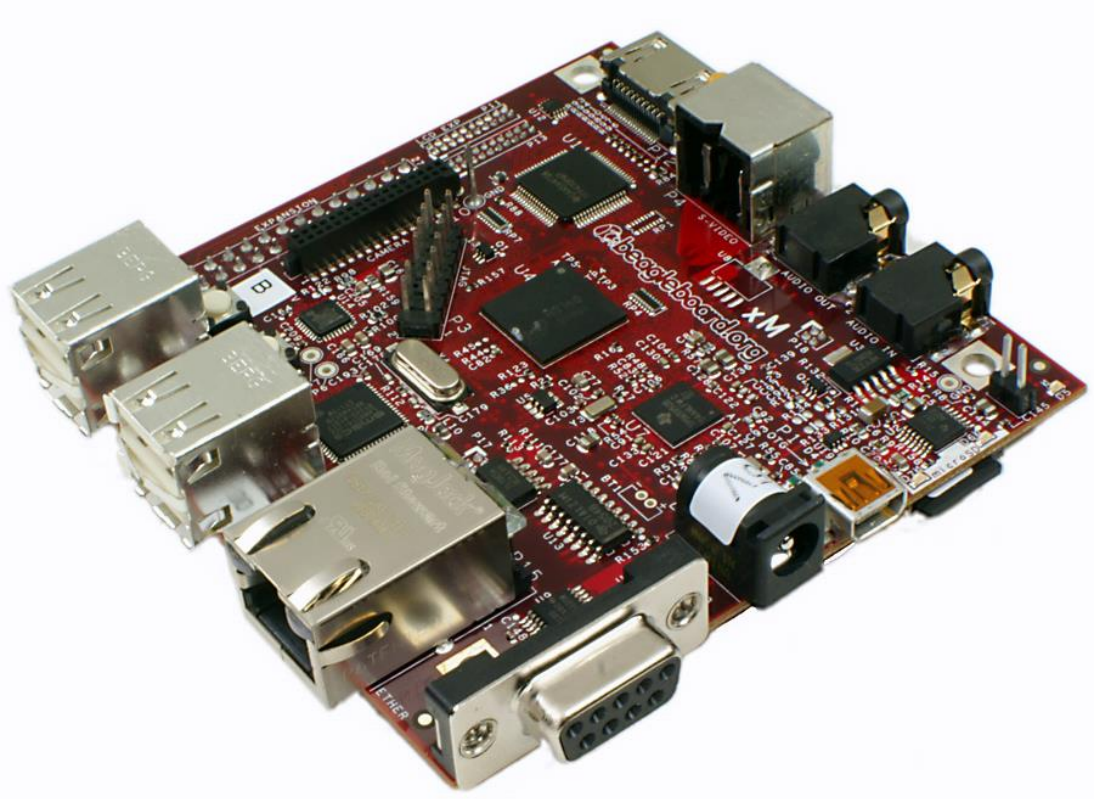


Figure 3-2 BeagleBoard-xM

Although it shares many of the same qualities of its successor, the BeagleBoard offers advantages in terms of the established support community and the relatively straightforward hardware interfacing. The relative power of the BeagleBoard means that most of the community projects have focused on networking or multimedia-based computing applications. However, there are projects which have used the BeagleBone as part of a feedback loop in a control system [40]. Despite this, the platform is primarily designed for software applications.

3.1.3. Raspberry Pi (Model B)

The Raspberry Pi (Model B) is the computationally weakest platform, possessing a 700 MHz Broadcom ARM-compatible processor. Half of the available memory is reserved for video processing, so the Raspberry Pi effectively has 256 MB of RAM for the operating system and computational use. This does, however, allow the Pi to have very sophisticated graphical capabilities for a computer of its size. The Pi does not possess an onboard ADC, but does possess GPIO and other external connectivity options. The Model B is required for Ethernet connectivity and additional RAM – the price point of this version is only $10 higher than the initial Model A. This higher price is already reflected in Table 3-1.

As a platform, the Pi quickly gained notoriety in 2012 as an extremely low cost development platform, given the power offered by the board. It is billed as a "credit card sized computer that plugs into your TV and a keyboard." The primary goal of the Raspberry Pi is to make computers and programming more accessible to children worldwide [41]. As a result, the scale of production is large enough that very capable hardware can offered at a low price, and this price point has allowed for community expansion. Figure 3-3 is a reference picture of the Raspberry Pi (Model B).

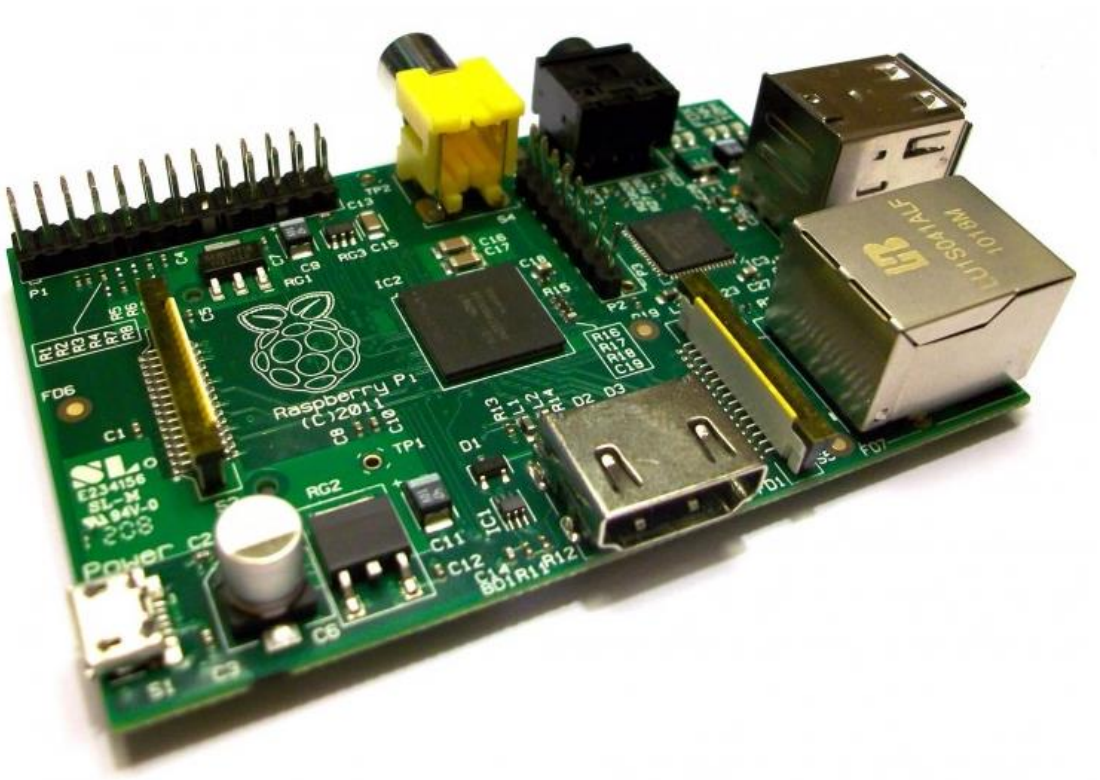


Figure 3-3 Raspberry Pi (Model B)

3.2. Overview of the BeagleBone Black Development Platform

The BeagleBone Black is the second iteration of the "BeagleBone" development platform. It is described as "a low-cost, community supported development platform for developers and hobbyists," and is fully open-source hardware [42]. As detailed in Section 3.2, the BeagleBone Black is most like the Raspberry Pi, with the key differences being the Programmable Real-Time Unit-Industrial Communication Subsystem (PRU-ICSS or simply PRUSS), onboard Touch Screen Controller, which is an Analog to Digital Converter (ADC), and the large quantity of General Purpose Input/Output (GPIO) connections. These three features greatly reduce the amount of external hardware needed and keep the footprint of the device minimized. Figure 3-4 is a reference picture of the BeagleBone Black.

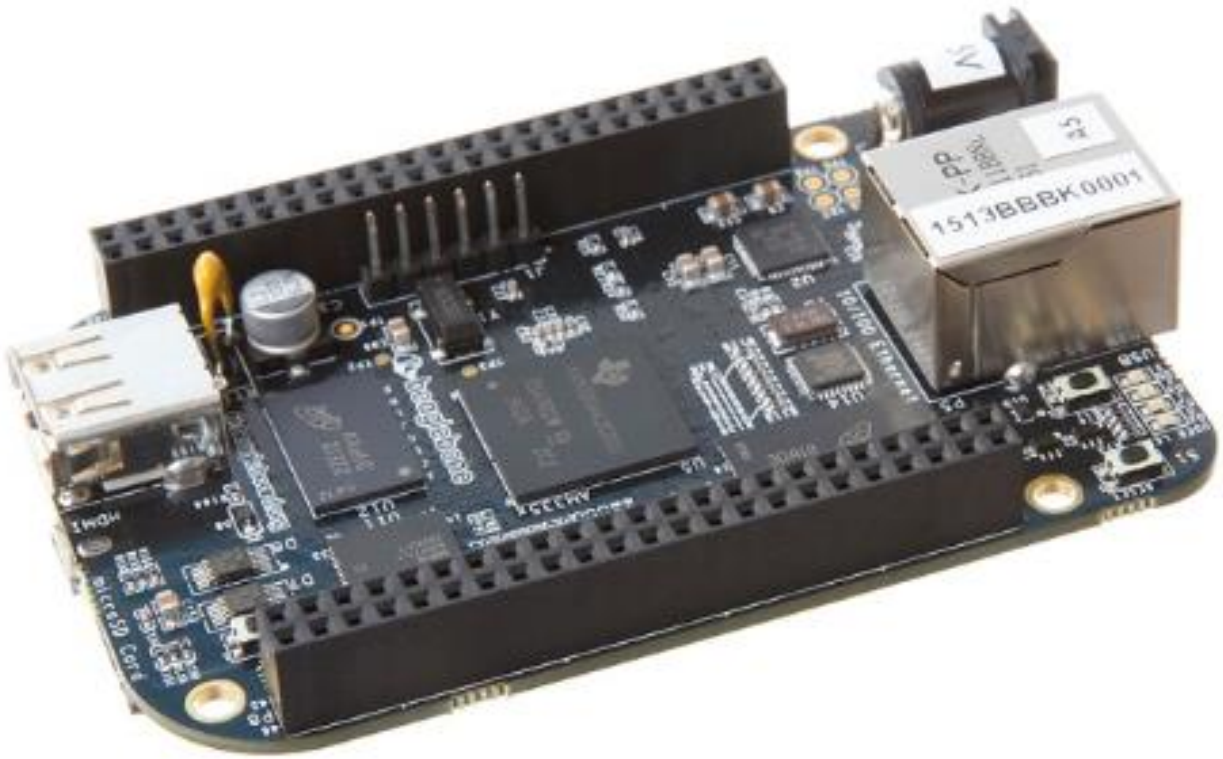


Figure 3-4 BeagleBone Black (Rev A5)

The processor on the BeagleBone Black is the Texas Instruments AM335x Sitara – an ARM Cortex A8 compatible microcontroller. The processor contains multiple subsystems and interface controllers for general purpose peripheral communication. These interface controllers allow the

BeagleBone Black to have substantial GPIO capability while maintaining low latency, as the signals are handled by the processor itself. Figure 3-5 shows the functional subsystems contained within the Sitara processor.

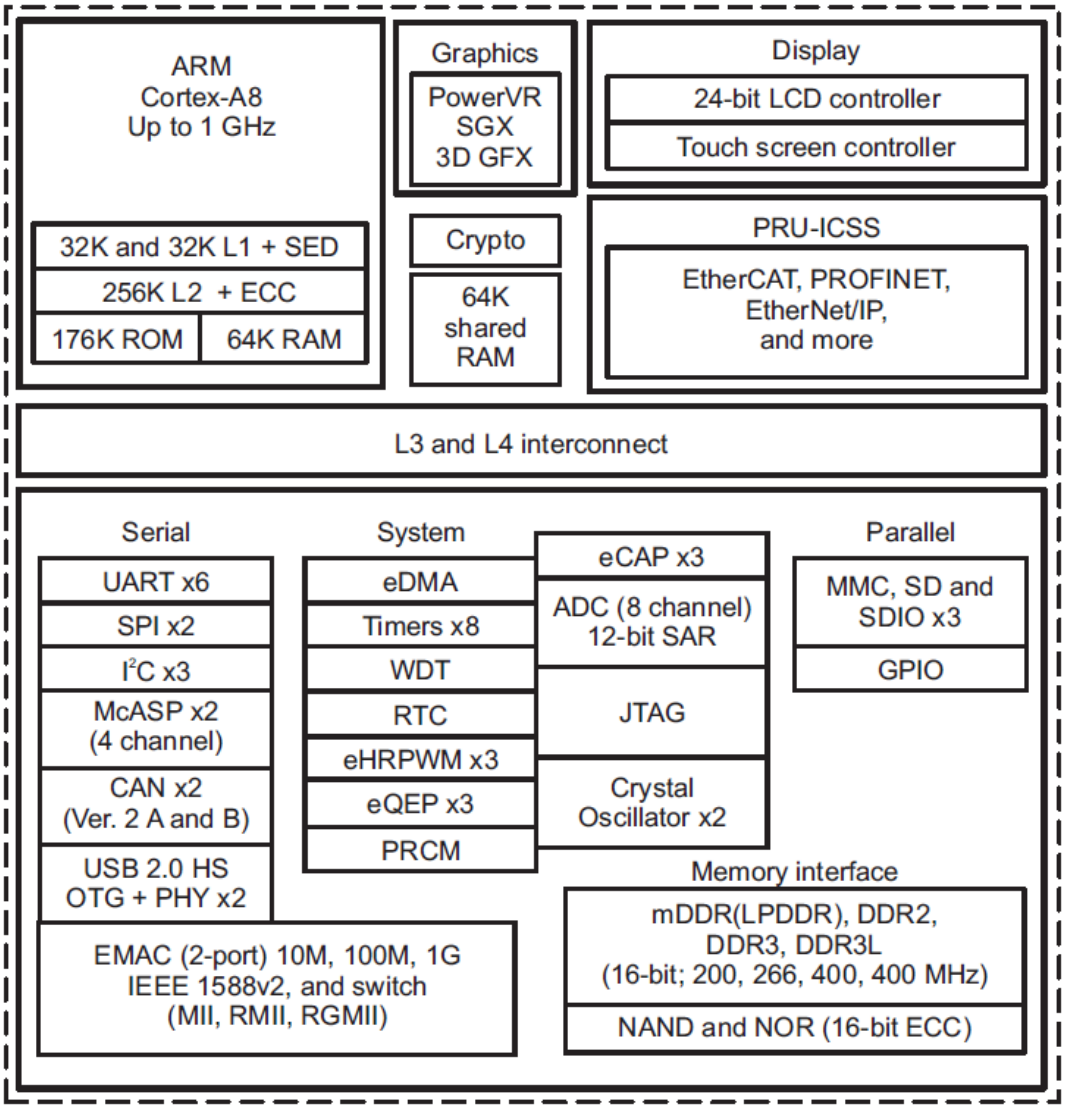


Figure 3-5 AM335x Sitara Functional Diagram

Strictly speaking, the BeagleBone Black has far more capability than is required to function as an FDR. High performance video, embedded cryptography, and the higher level interconnects will remain underutilized. However, their presence does provide possible future expansion opportunities. As

mentioned, the key features are the embedded PRUSS, ADC, and GPIO – all of which provide functionality which is required by an FDR.

3.2.1. Programmable Real-Time Unit Subsystem

The PRU-ICSS module consists of dual 32-bit RISC cores (PRUs) along with shared data and instruction memory pools, internal peripheral modules, and an interrupt controller. This subsystem allows for tasks to be offloaded from the main core of the processor and permits these tasks to be executed independently from the rest of the system. The shared pools allow the PRUs to communicate with each other or the main processor. The exact nature of this communication depends on the firmware loaded onto the cores [43]. As these cores are specialized and limited in function, programs which are executed on them must be written in a small, deterministic instruction set – PRU Assembly. This is a custom assembly language for these cores, only common to the PRU core found in other Texas Instruments embedded SoCs. Figure 3-6 shows the functional block diagram for the PRU-ICSS.

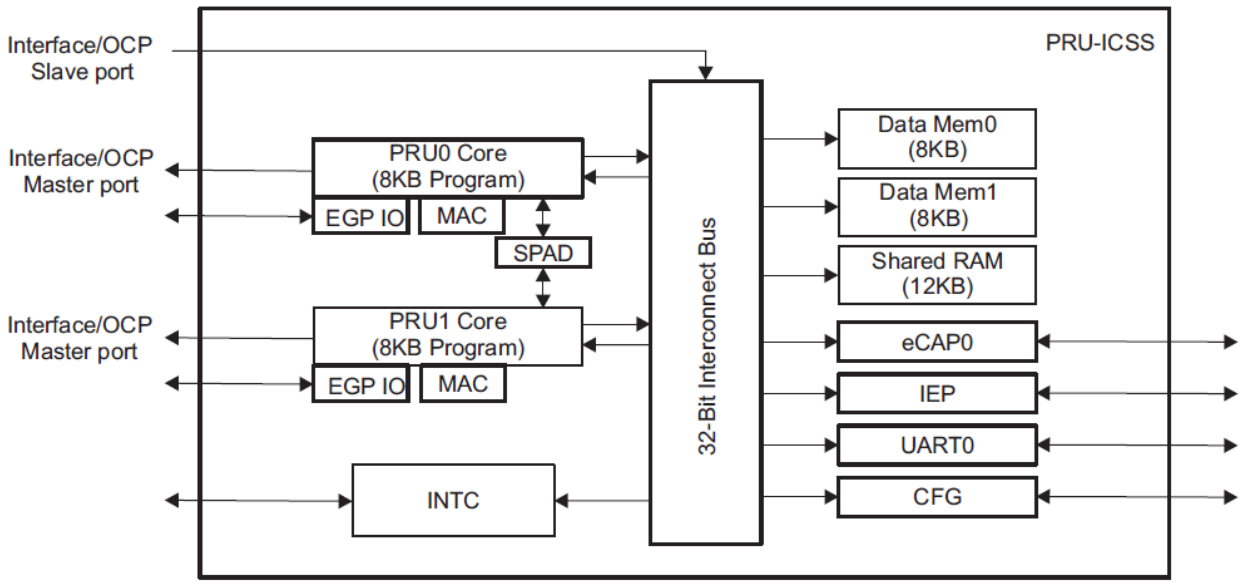


Figure 3-6 PRU-ICSS Functional Block Diagram

The major limitation of the PRU cores is the 8KB of program memory available. This means that any real-time code must be very simple. Thankfully, only the ADC communication and direct measurement needs to take place on these cores. The 32-Bit Interconnect Bus allows for direct peripheral access to the ADC and GPIO, and the shared memory pool (12KB) with the main processor

will allow for communication with another program which can handle the computation and communication. Section 4.4 explains this process in more detail.

3.2.2. Touch Screen Controller Subsystem

The Touch Screen Controller module (TSC-ADC-SS) is an 8-channel general-purpose analog-to-digital converter. The underlying hardware is a successive approximation register (SAR). A control-level block diagram for this register is shown in Figure 3-7.

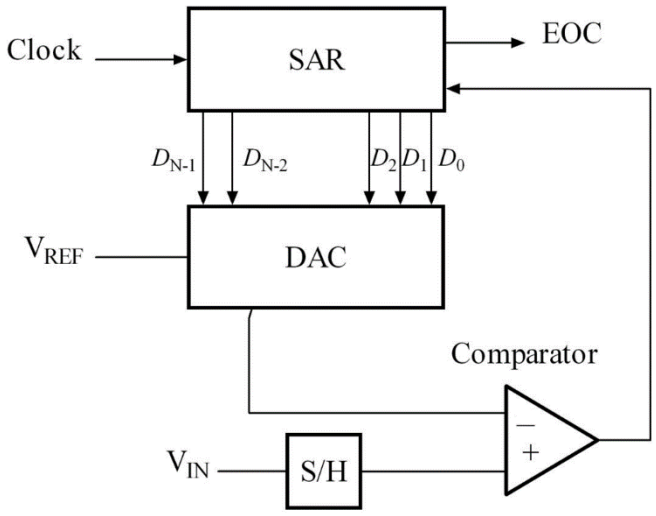


Figure 3-7 Successive Approximation Register Block Diagram

The module has optional support for interleaving touchscreen conversions for different panel configurations [44]. Specifically, the supported modes of operation are:

- 8 general-purpose ADC channels
- 4-wire TSC with 4 general-purpose ADC channels
- 5-wire TSC with 3 general-purpose ADC channels
- 8-wire TSC

For the purposes of the next generation FDR, only a single ADC channel is required, however the flexibility offered here could be used in future iterations to measure additional quantities – such as line current – or a small touchscreen could be added to the FDR to allow for an interactive information display. The ADC subsystem does have two significant limitations – 12 bit precision, and a multiplexed

input selection. The 12-bit sampling precision means that higher precision measurements will require the use of oversampling, as discussed in Section 4.3. The controller has built-in functionality to perform sample averaging, but this functionality is conservatively clocked and does not utilize the full capability of the ADC. Additionally, the multiplexed input lowers the maximum possible sampling rate, as the 24 MHz available for sampling is divided up among 8 separate channels.

As shown in Figure 3-8, the ADC subsystem has direct communication with the PRU, which enables real-time measurement. This low-level interface is necessary to prevent queries from being relayed through the main processor, which would require interaction with the onboard operating system. The timer integration permits the use of external interrupt signaling – such as a PPS signal or alternative control timing signal – to initiate a timing sequence on the ADC. Higher level interconnect buses could be utilized for the touch screen capability, should future iterations include it.

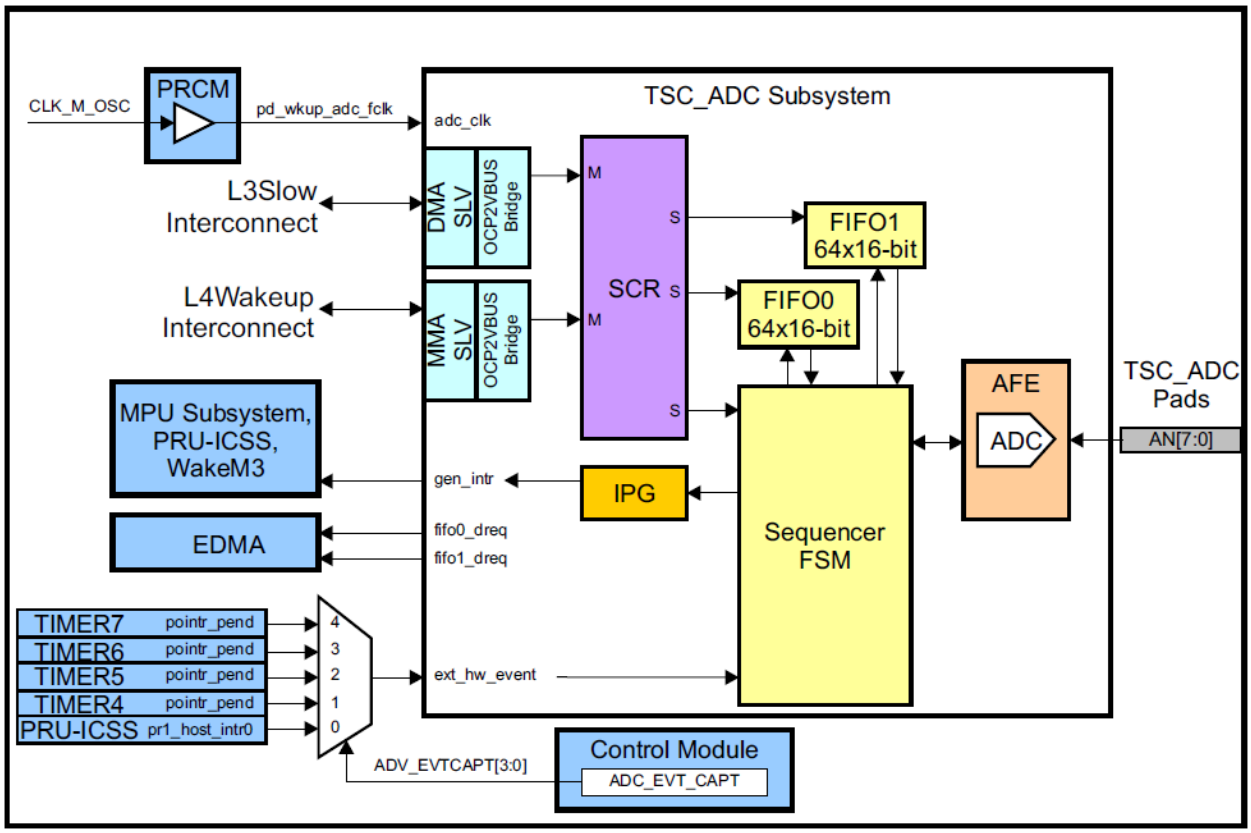


Figure 3-8 TSC-ADC Subsystem Functional Block Diagram

3.2.3. General Purpose I/O Capability

The General Purpose I/O subsystem on the BeagleBone Black is comprised of four general-purpose input/output modules. Each module provides 32 dedicated pins with input and output capabilities, for a total of 128. Each physical pin can be multiplexed via firmware to allow access to all of these functions. This firmware is configured through the Device Tree Overlay functionality of the Linux kernel and is explained in more detail in Section 4.2.1. The primary limitation is that at any given instant, only a subset of these pins is physically available, depending on the mode configuration of the physical pin. The pins may be configured for the following applications:

- Data input/output
- Keyboard interface with debounce cell
- Synchronous interrupt generation in active mode via external signals

Additionally, Pin 0 may be configured as a wake-up signal. With the exception of that extra functionality, all four GPIO modules have identical connections and interfaces – connections to the physical pads, the Microprocessor Unit (MPU) Subsystem, the L4Peripheral Interconnect, and the global system clock.

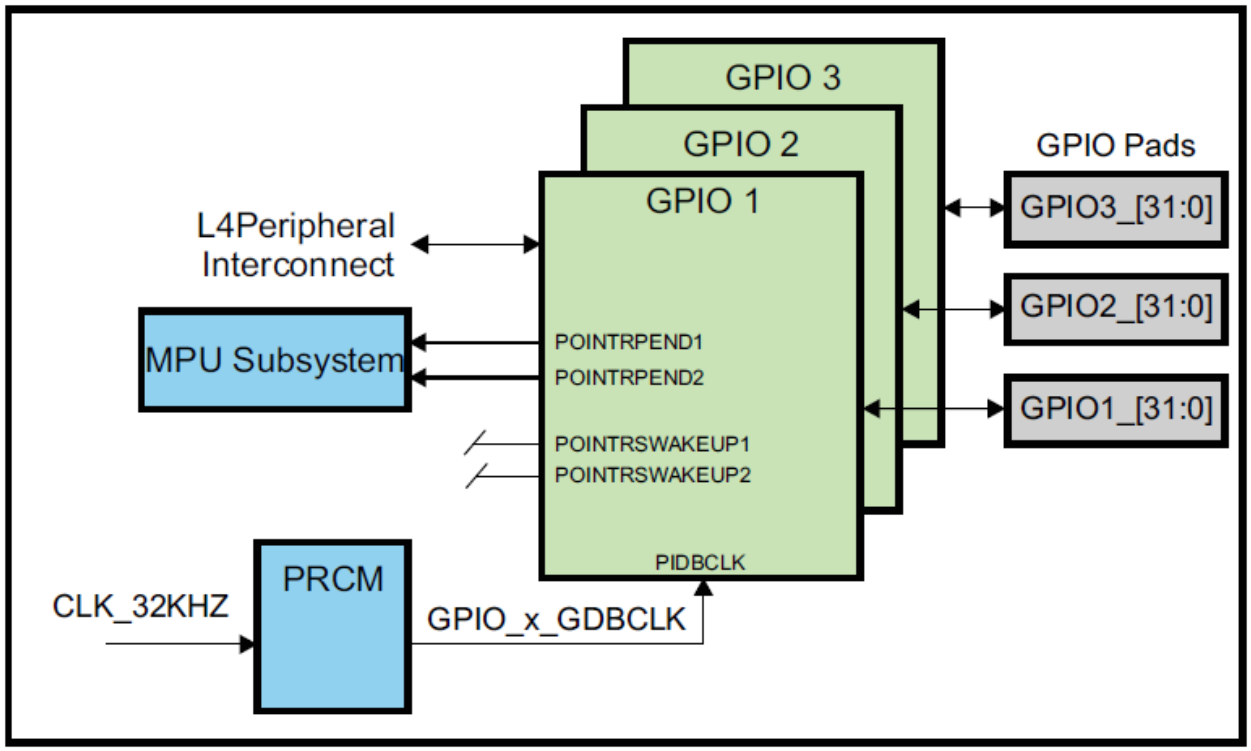


Figure 3-9 GPIO Module Integration

The key attribute of this subsystem is that GPIOs do not require processor time from the ARM core – they act independently and can interface with the PRU-ICSS and ADC through the MPU subsystem. This ensures that any GPIO signals sent – such as a PPS signal or physical user input – will not affect the real-time operation of the unit.

3.3. External Subsystems

In addition to the BeagleBone Black itself, there are three major hardware circuits that must be developed for the next generation FDR. These include the GPS Module, Power Supply, and Signal Input for the device. Each of these circuits is modular, and hardware can be altered or upgraded in future revisions as necessary. Of particular importance is the interface between each of these circuits and the BeagleBone Black – components are selected such that even worst-case tolerance values will not result in voltage or current delivered to the board which is out of specification. This is particularly true in the case of the ADC module, which has a 1.8V maximum input voltage limit.

3.3.1. Timing/GPS Module

The GPS module itself is the GlobalTop PA6H (MTK 3339 chipset) [45]. Specifically, this module has a built-in patch antenna, external antenna connectivity, and provides a PPS signal on a separate channel. The PA6H has internal overcurrent protection for the external antenna and can be mounted to a Printed Circuit Board (PCB) either manually or with a reflow soldering oven. The patch antenna is mounted to the top of the module, and the PCB interfaces are all along the outer footprint of the board. The placement of these connections means that soldering can be performed by hand if necessary. The module optionally supports an external backup power supply. A common implementation of this is essentially a coin cell battery providing 3.0V to the module so that memory is retained in the event of power loss to the device. Figure 3-10 and Figure 3-11 are pictures of the top and bottom of the GlobalTop integrated circuit (IC) module, respectively.



Figure 3-10 GlobalTop PA6H GPS Module (Top)



Figure 3-11 GlobalTop PA6H GPS Module (Bottom)

For prototyping purposes, a breadboard-compatible version of this module is sold by Adafruit [46] – the Adafruit Ultimate GPS Breakout. This is a pre-built breakout board with breadboard-compatible pinouts, an external GPS antenna connector and 3.3V and 5V voltage regulators. This provides a simple way of establishing and testing GPS connectivity with the BeagleBone Black. Additionally, the through-hole compatible design means that the breakout board can be utilized in both the early breadboard tests and as a complete module on the prototype through-hole PCB. Figure 3-12 is a picture of this breakout board.

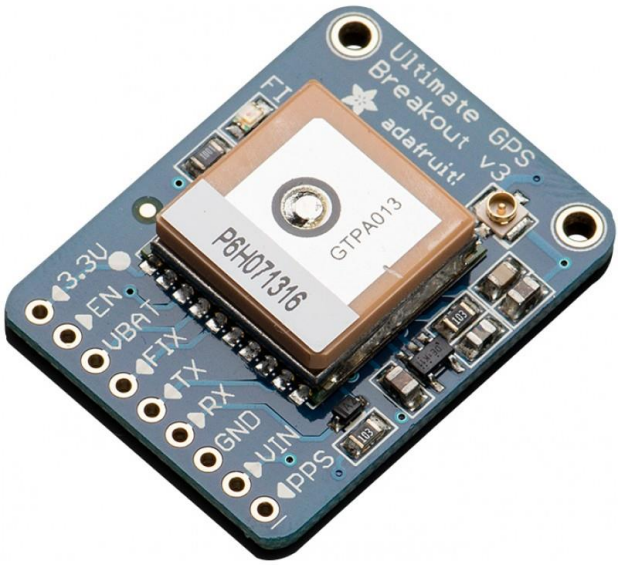


Figure 3-12 Adafruit Ultimate GPS Module Breakout Board

In the prototype breadboard setup, a low power LED is placed in parallel with the PPS signal output for visual feedback to ensure that the signal is being sent correctly. Emphasis should be placed on the fact that this may be considered a reference design; any standard GPS module with PPS capability may be used, so long as the voltage regulation and connectivity requirements are met.

3.3.2. Power Supply

While the BeagleBone Black has onboard circuitry to handle input power, the power input jack is a 5 VDC connection. In order for the FDR to operate from a single wall connection, the input signal must be split into power and measurement components. Electrical noise isolation is critically important here –

the power connection for the board must produce a minimum amount of signal noise for the measurement. This is fundamentally a similar design to that of the current generation FDR. As a reference design, this circuit may be constructed with any off-the-shelf hardware which is within specification. The primary requirements of this circuit are as follows:

- Reduce voltage from distribution level (120V/220V).
- Rectify and smooth voltage waveform (convert to DC).
- Protect power circuitry from shorts and voltage spikes.
- Prevent electrical noise from entering the signal input circuit.
- Provide suitable DC voltages for operational amplifiers.

Constructing a standard analog power supply circuit from a transformer, full wave rectifier, and some capacitors is entirely possible and reasonably cheap. The current Generation-2 FDR uses exactly this method, and as a result, a large portion of the PCB layout is set aside for the power supply of the unit. In the interests of compactness, the prototype Generation-3 FDR uses a Recom Powerline RAC10-05DC/277 switching power supply [47]. This power supply specifically has a +5 VDC and -5 VDC rail, which is the voltage required for both the BeagleBone Black and the transformers. The total current capacity of the Recom is only 2 A, however, this is more than enough for the current hardware specifications. Peak current draw occurs during kernel boot up and is only 460 mA, or 2.3 watts total [48]. This also gives the prototype Generation-3 FDR a significantly lower overall power draw than the current unit. A replaceable fuse is included in this circuit to protect the device from unexpected power surges [49].

3.3.3. Signal Acquisition

The BeagleBone Black contains an onboard ADC which is easily capable of measuring and converting the analog input waveform at grid nominal grid frequencies. To ensure measurement accuracy, input filtering is necessary to block higher-order unwanted frequencies. Additionally, the signal voltage and current must be transformed to acceptable levels for the ADC hardware. The signal input circuit serves both of these purposes.

Primarily, the input voltage must be reduced from nominal wall voltage (120V or 240V) to the 1.8V required by the onboard ADC. However, a simple transformer does not suffice here as it would inadequately protect the device during a voltage surge. Voltage limiting elements in the form of amplifier circuits are required. These ensure that the signal is isolated from the input and that the

bounds of the ADC input are strictly enforced. Specifically, the maximum input current of 2 µA requires that the design remains within specifications for even the worst case component tolerance combinations. The prototype breadboard setup for this consists of two rail-to-rail operational amplifier ICs and a voltage divider. The op-amps act as isolation and also ensure that the voltage level is never out of specification – the amplifier outputs can never provide more voltage than their sources allow for. Additionally, this setup serves as isolation from electrical noise. To perform this conversion, a non-inverting amplifier circuit is used. This allows for a desired gain to be applied to the signal while applying a DC offset utilizing a single op-amp. Figure 3-13 shows a generic non-inverting amplifier circuit.

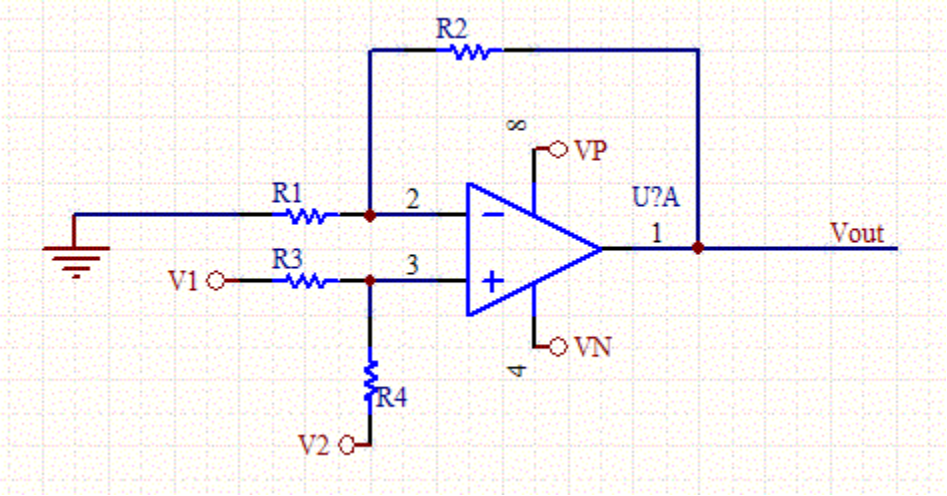


Figure 3-13 Non-Inverting Amplifier Circuit

The equations for the gain (1) and input (2) of this circuit are as follows:

$$1)\ V_{out} = \left(1 + \frac{R_2}{R_1}\right) V_{in}$$

$$2)\ V_{in} = V_2 - \frac{V_2 - V_1}{R_3 + R_4} R_4 = V_1 + \frac{V_2 - V_1}{R_3 + R_4} R_3$$

Substituting (1) into (2), the overall equation is:

$$3)\ V_{out} = \left(1 + \frac{R_2}{R_1}\right)\left(V_2 - \frac{V_2 - V_1}{R_3 + R_4} R_4\right)$$

Where V_1 is the AC input signal and V_2 is the DC bias signal.

Given that V_1 is 0.48V AC (RMS), or 0.68V (Peak-Peak), and V_2 is the +5VDC rail, the input limits for the BeagleBone Black's ADC can be used to determine the required resistors. The acceptable levels for the analog inputs are 0-1.8V. However, this range cannot be fully utilized as the circuit must provide protection against overvoltage conditions. Applying a 10% safety margin to the gain yields an input signal of 0.75V (Peak-Peak) for a 0-1.8V input, with a gain of 1.385. Therefore, setting $R_1 = R_3$, and consulting a list of commonly available 1% tolerance resistors [50], the following values are obtained: $R_1 = R_3 = 10.2k\Omega$, $R_2 = 3.92k\Omega$, and $R_4 = 68.1k\Omega$. The 1% tolerance specification is used to ensure accuracy while keeping the component cost for the circuit cost low. All resistors used have minimal temperature variation parameters and are rated for 0.25W. Deviation from the nominal value at this tolerance does not significantly alter the input signal to the ADC. As this is a non-inverting configuration, the common-mode input signal can negatively affect the quality of the output. This is resolved by using an operational amplifier which has a high common-mode rejection ratio (CMRR) at the target frequency range. Given this criteria and the voltage levels available on the board, Texas Instruments LM6142 Precision Op-Amps are chosen. Figure 3-14 shows the CMRR for the LM6142 operational amplifier [51]. At the operational frequency range of the FDR (< 100Hz), the CMRR is approximately 116 dB, which is excellent for a measurement application.

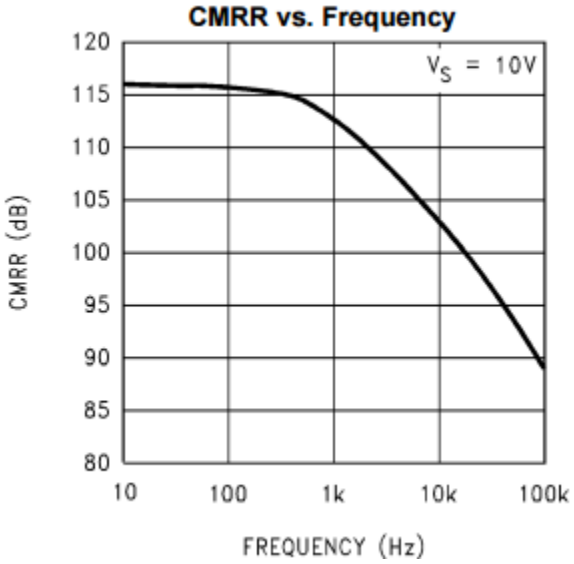


Figure 3-14 LM6142 Operational Amplifier CMRR Characteristic

After the signal is transformed to the proper input levels, a low pass filter is placed in series to ensure that higher order harmonics are reduced or eliminated. While higher order harmonics could potentially indicate irregularities in the operation of a power system, they are undesirable for the analysis performed by the FDR. Although the FDR is not an industrial-grade PMU, the IEEE C37.118 standard can serve as a useful reference for filtering capabilities, if one considers an FDR to be an M (measurement) class device. The attributes of a low pass filter are straightforward:

- Magnitude uniformity throughout the passband.
- Magnitude attenuation above the cutoff frequency.
- Phase shift of the signal in the passband.

In the case of the next generation FDR, the goal is to minimize any distortion in magnitude and phase around nominal system frequency while blocking higher order harmonics. The two options for doing this are a narrow bandpass filter or a low pass filter. Unfortunately, the phase delay for a standard analog bandpass filter shifts by large amounts over the passband region and is undesirable for the operation of the FDR. With a low pass filter, the phase differences are not nearly as pronounced, as the cutoff frequency can be set at a much higher point. The tradeoff then becomes a matter of filter order – a higher order filter has steeper magnitude attenuation, but also a larger total phase shift. Assuming a 1st order filter, shown below in Figure 3-15. Figure 3-16 shows the frequency and phase responses for this filter.

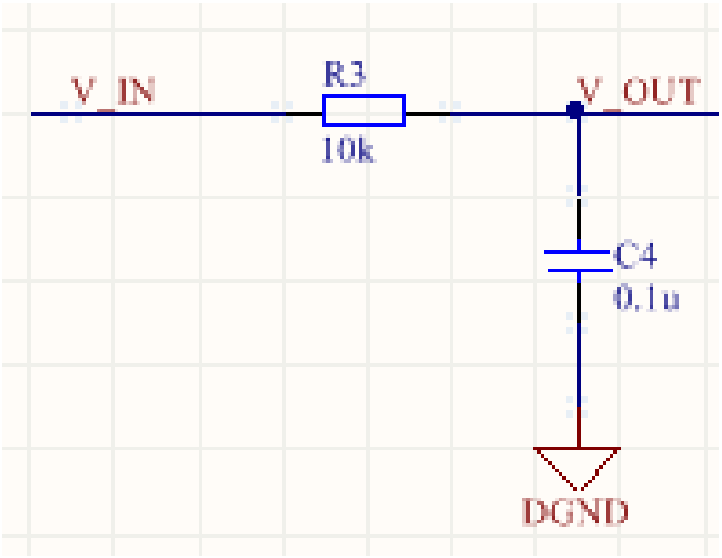


Figure 3-15 1st Order Low Pass Filter (RC); 160 Hz

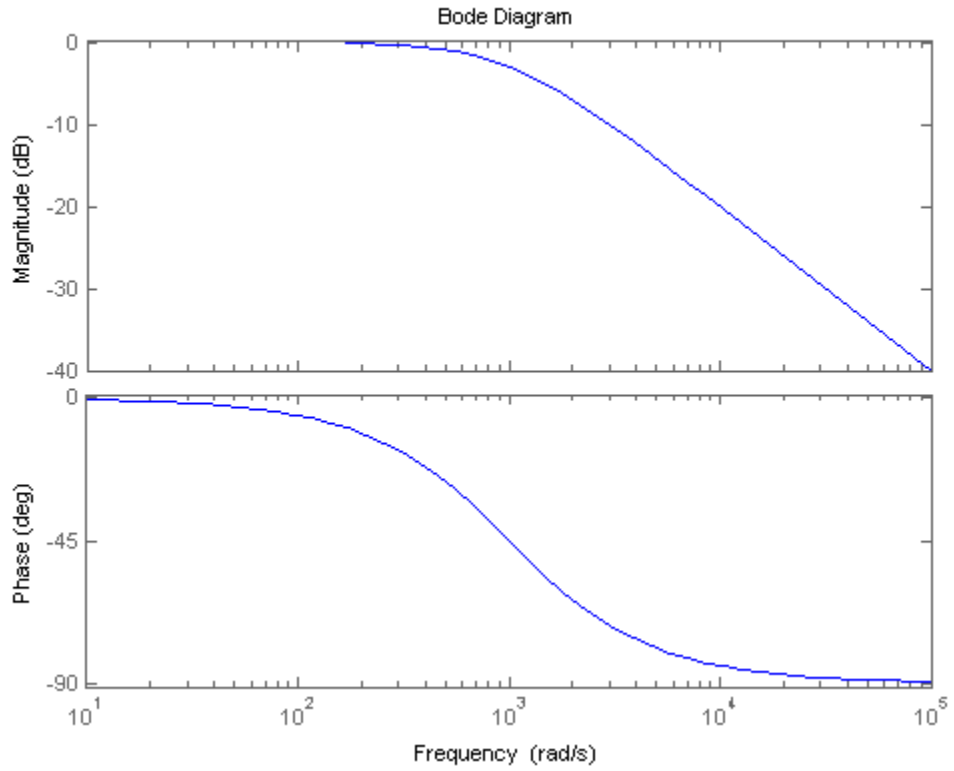


Figure 3-16 Bode Plot for 1st Order Low Pass Filter

The component values are chosen for a cutoff of 160 Hz – this value minimizes higher order harmonics and ensures a high degree of linearity in the nominal frequency range. Additionally, this frequency can be achieved with extremely commonplace components – a 10kΩ resistor and a 0.1 µF capacitor. This is not a cost saving measure so much as an assurance that the board is designed with easily replaceable components, and that there is no foreseeable issue with sourcing replacements. This frequency range also provides a substantial margin for error, as even low grade or faulty components will not appreciably shift the filter's cutoff frequency to unacceptable levels. For comparison, a 2nd order Butterworth Filter [52] would provide a steeper attenuation above the cutoff frequency, with an increased phase delay, as illustrated in Figure 3-17.

With a second order filter, the phase shift curve is twice as steep, but there is no appreciable magnitude attenuation advantage, as the nominal frequencies are far below the cutoff. Higher order harmonics will be acceptably blocked by either design.

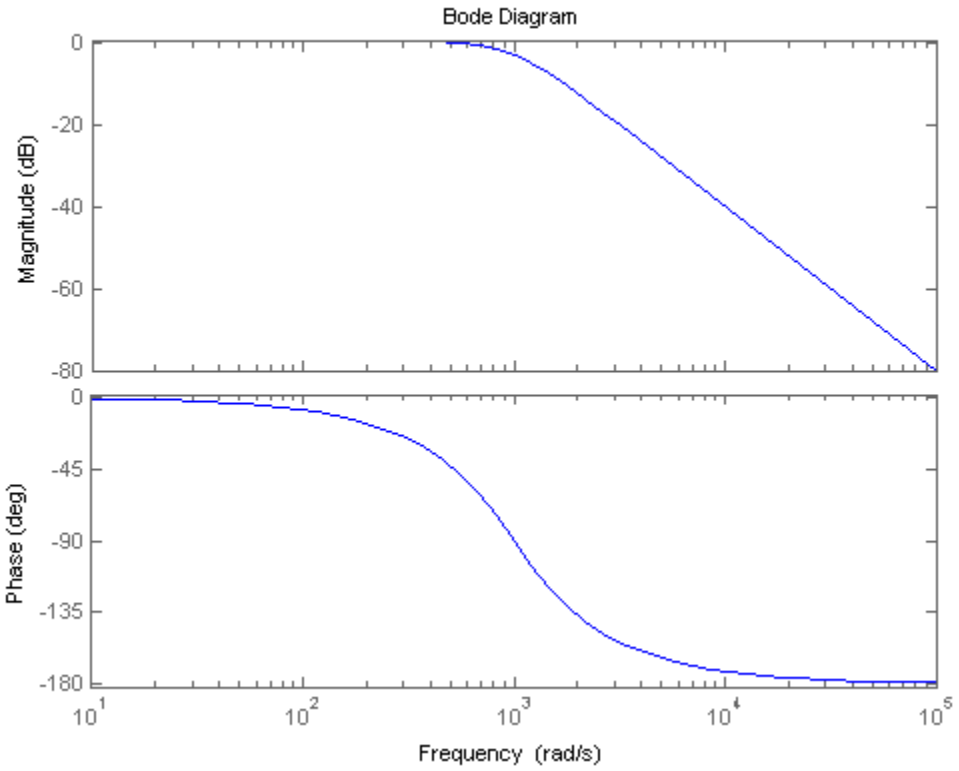


Figure 3-17 Bode Plot for 2nd Order (Butterworth) Low Pass Filter

3.4. Prototype Implementations

The development of the hardware prototype had three main stages – initial component interfacing, a full breadboard/protoboard hardware implementation, and then lastly a printed circuit board (PCB) implementation. This section will explore the progression of those stages and explain the refinement process.

In the early stages, the initial concern and testing procedure was focusing on integrating the GPS receiver with the BeagleBone Black. The software aspect of this process is explained in Sections 4.2.1 and 4.2.2, with source code available in Appendix A. For this test, the GPS module was attached to a breadboard and connected to the BeagleBone Black. Power was provided via the 5VDC barrel jack on the BeagleBone Black. The input signal was provided by a Fluke 271 Function Generator and set to be a 60 Hz sinusoidal waveform with a Peak-to-Peak amplitude of 1.4 volts (0.2 to 1.6 volts). Figure 3-18 shows this early testing setup used to validate the correct operation of the PPS synchronization.

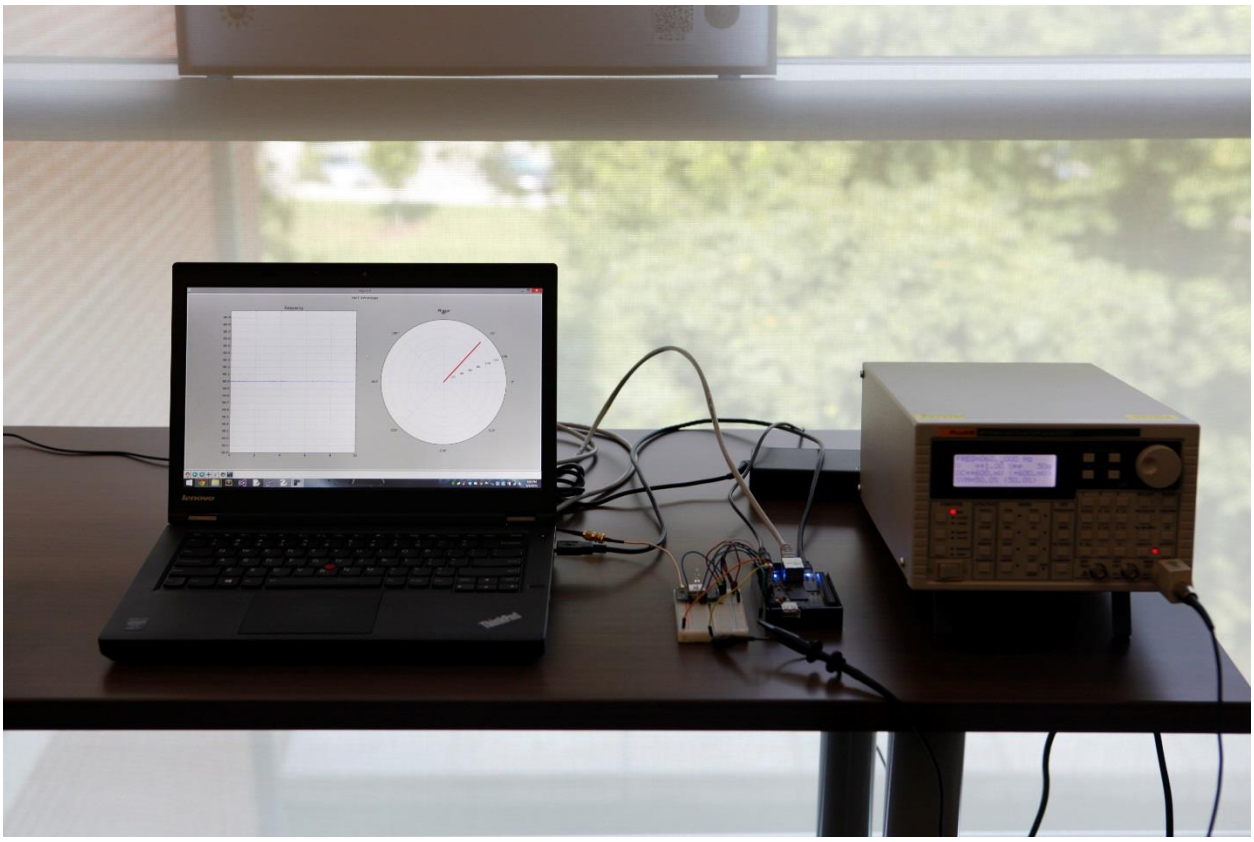


Figure 3-18 PPS Functionality Test Setup

The next phase of development and hardware testing was to integrate the signal acquisition and filter circuitry into the device. In order to do this, a negative voltage rail must be provided for the op-amps to convert an AC signal to the proper voltage range. Therefore, this phase also added the external Recom power supply as well as the current-based voltage transformer, analog filter, and operational amplifiers. This full breadboard setup was used to validate the electrical operation of the device and provided the basis for the through-hole PCB prototype. Figure 3-19 shows the full breadboard setup in testing.

Moving from a protoboard/breadboard configuration to a printed circuit board requires additional considerations beyond the circuitry connections. On the surface, the move to a PCB prototype is entirely beneficial. Breadboards are excellent tools for establishing the basic operation of a device, but the wires which span across the board add undesirable noise to the signals. Additionally, breadboard connections are not as electrically conductive as solder joints, which are used on PCBs.

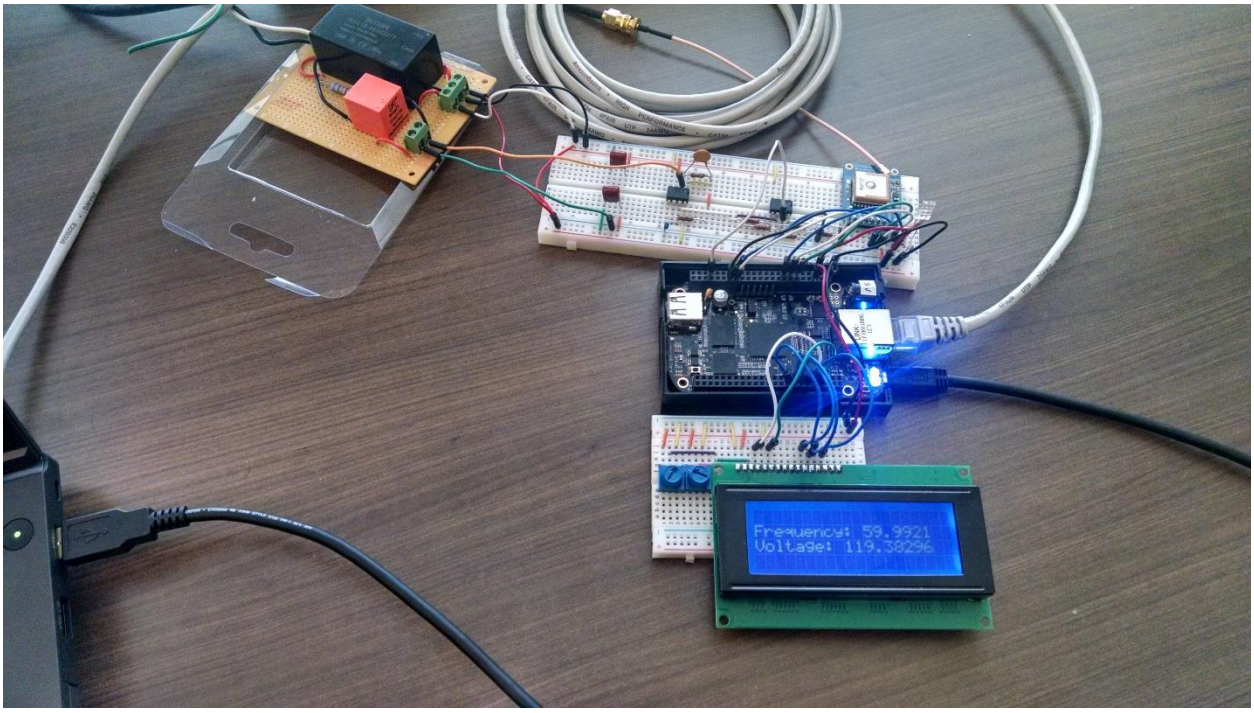


Figure 3-19 Protoboard/Breadboard Test Setup

These considerations mean that a PCB has the potential to be a more reliable device, but only if designed properly. For this prototype, the schematics and PCB drawings were created in Altium Designer 15. This is a CAD software which allows for easy conversion from component schematics to a finalized PCB drawing. The schematic layer establishes electrical connectivity and is essentially representative of the breadboard design. However, when moving to the PCB diagrams, there are further considerations. Put simply, the physical realization of the circuitry comes in the forms of layers, traces, and vias and care must be taken to minimize unintended electrical effects. As the PCB prototype is reasonably simple, a two-layer board suffices. This completely eliminates any chance of a middle-layer (hidden) short circuit, as all traces are on either the top or bottom of the board. The width and length of the electrical paths (traces) is very important, as the voltage over a given path steadily drops due to the resistance of the copper. The formula for determining this resistance is straightforward – $R = \rho \times \frac{L}{A}$, where R is the end-to-end track resistance (Ohms), ρ is the resistivity of the track material (Ohm Meters), L is the track length (Meters), and A is the cross sectional area of the track (Square Meters). Due to the simplicity and small footprint of the PCB, it is not necessary to compute this value for each individual trace, but this equation does serve as a boundary constraint to ensure that sufficient power is delivered to each component on the board. As part of its feature set, Altium Designer automatically enforces given design

rules which ensures consistency and prevents fabrication errors. In developing the board, the following design rules were adhered to:

- Trace width should be maximized in order to reduce path resistance.
- Trace width should never exceed via plate size.
- Traces should be of consistent width for the entire path.
- Perpendicular (90 degree) junctions should not be used.
- Minimum trace width is 20 mils (20/1000 inches)
- Minimum spacing between parallel traces is 10 mil.

By using these design rules, path resistance is minimized and the chance of physical defects in an electrical path is greatly reduced. While maximizing trace width is beneficial up to a point, a trace which is much wider than a connecting via has the potential to induce local field discontinuities. In preventing the use of perpendicular junctions, the board can be manufactured by less advanced equipment, and the path width is more likely to remain consistent over the length of the trace. Utilizing comparatively wide traces and spacing between them also allows for a wider range of potential PCB manufacturers. In addition to the design rules, which are software enforced and prevent mistakes, the following guidelines were also adhered to:

- All components line up on a 5 mil grid structure.
- Routes do not create regions of dead (isolated) copper.
- Routing order for traces is: Power, Signal, and Auxiliary.
- Power and signal traces have integrated vias for test-points.

These guidelines in many ways serve the same purpose as the design rules – to minimize sources of potential electrical errors and to make the board as easy as possible to fabricate. However, these guidelines cannot be automatically enforced by the software. Of particular importance is the trace routing order – power traces are routed before anything else to ensure shortest path length and lowest voltage drop. Signal traces follow, to again ensure shortest path length possible and to group signals whenever possible for ease of testing. Auxiliary traces include the paths from headers to additional hardware – the LCD connector. Inlined with two of the pins on the LCD connector are two 10kΩ trimmer potentiometers [53]. These allow for fine-tuned control of the LCD's backlight brightness and contrast of the displayed text. Once the PCB layout was finalized, the board was manufactured by Bay Area Circuits [54]. To ensure that the board could be manufactured by their fabrication process, their online Design

for Manufacturability Tool [55] was used. The automatically generated board renders from the InstantDFM tool are shown in Figure 3-20 and Figure 3-21.

A summary of the minimum design characteristics is presented in Table 3-2. The specific instances where these tolerances are provided are shown in Figure 3-22 through Figure 3-26.

Table 3-2 Generation-3 FDR Through-Hole PCB Minimum Design Characteristics

Board Parameter	Distance (mil)	
	Top Layer	*Bottom Layer*
Line Width	10.00	10.00
Ring	9.50	9.50
Clearance to Copper	9.07	9.10
Clearance to Pad	> 20.00	> 20.00
Clearance Pad-to-Track	9.55	9.55
Clearance Track-to-Track	9.07	9.10
Self-Spacing	2.00	3.00
Clearance to Plated Through-hole	19.48	19.48
Clearance to Outline	50.00	50.00

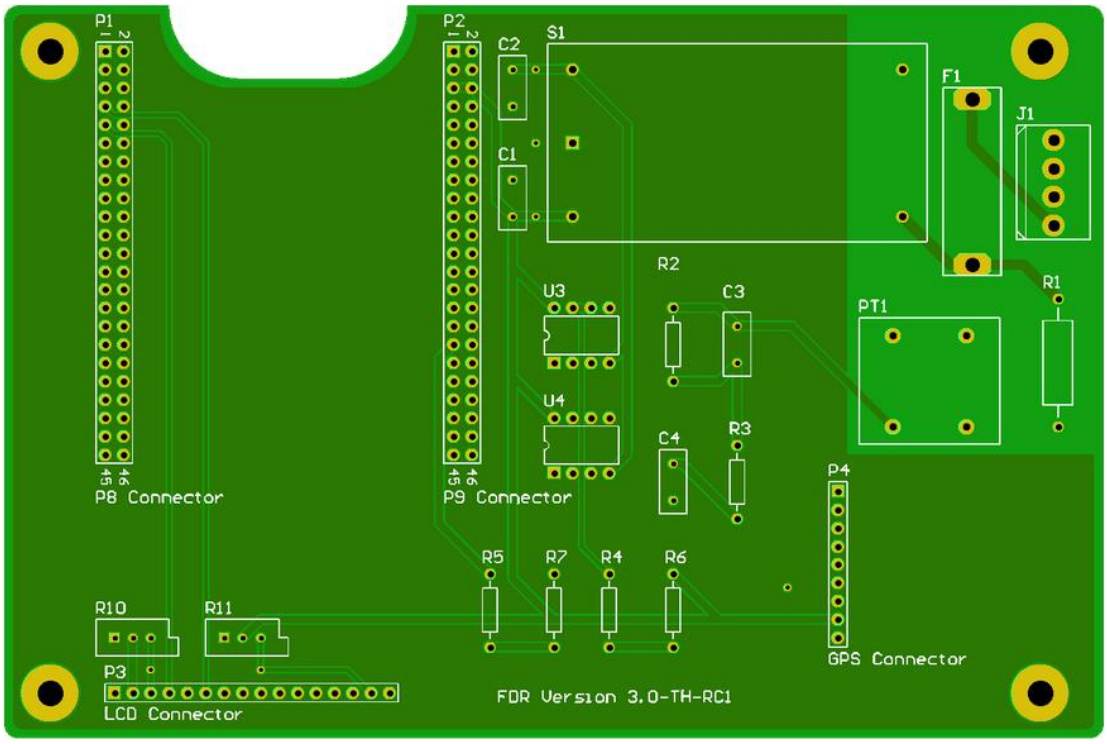


Figure 3-20 Generation-3 FDR Through-Hole PCB InstantDFM Result (Top)

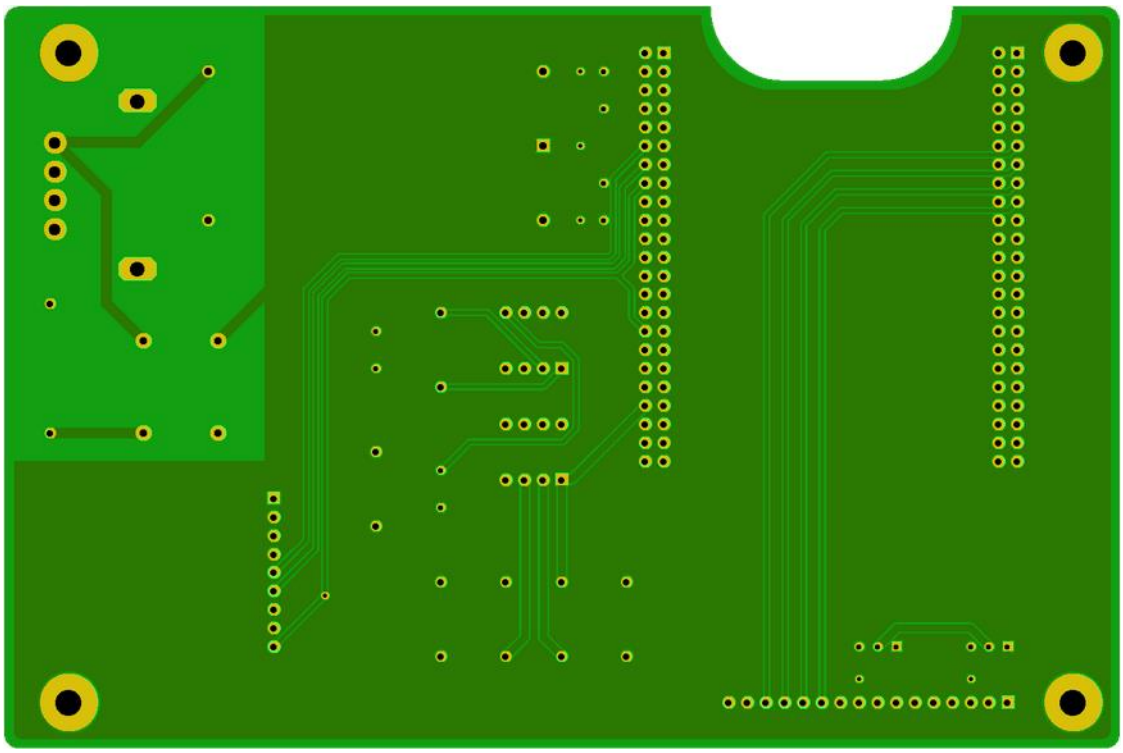


Figure 3-21 Generation-3 FDR Through-Hole PCB InstantDFM Result (Bottom)

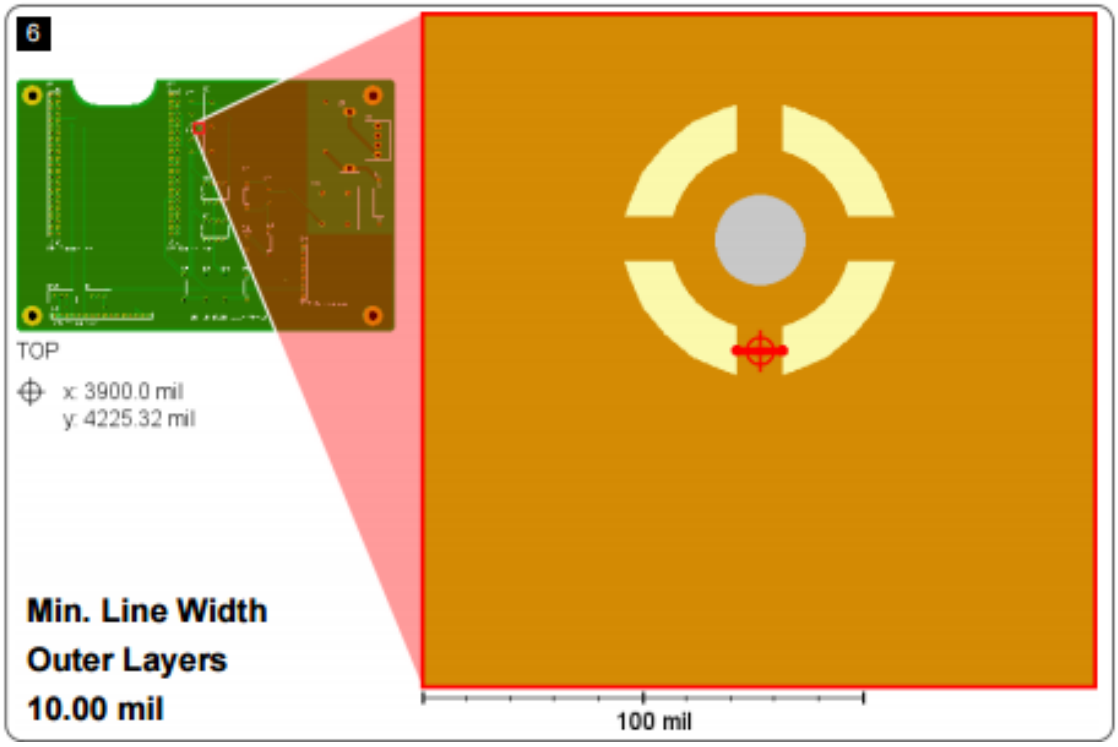


Figure 3-22 Generation-3 FDR Through-Hole PCB Minimum Line Width

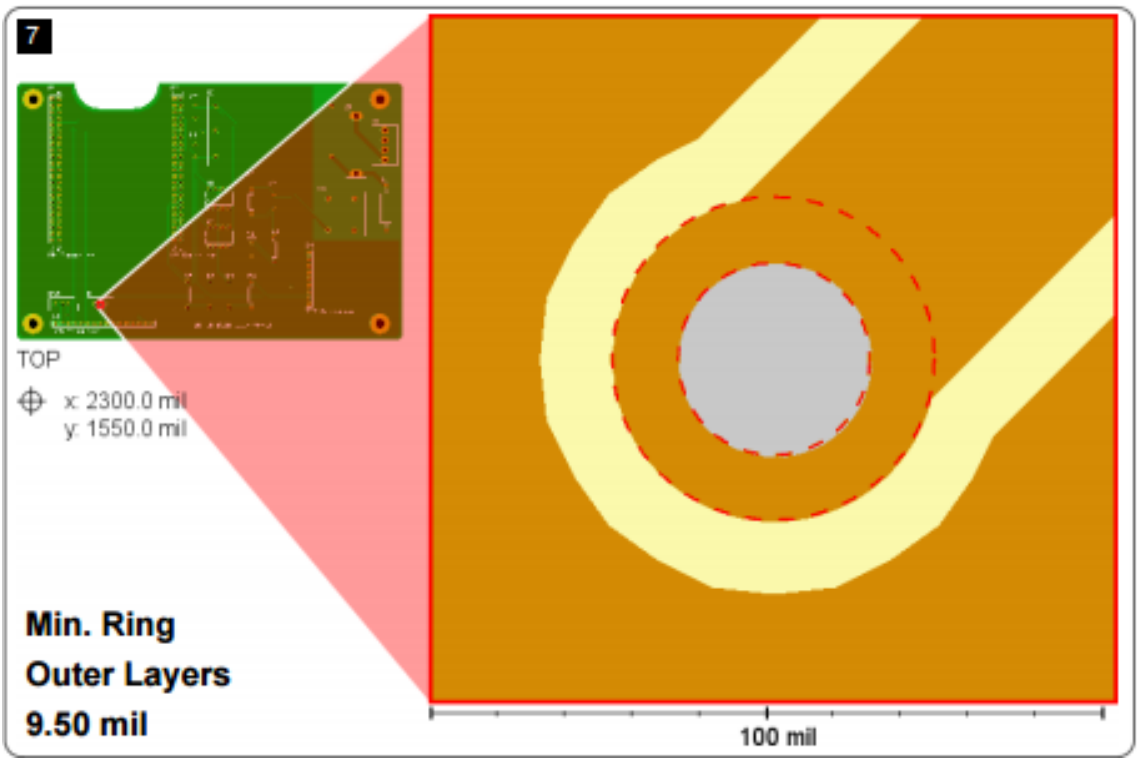


Figure 3-23 Generation-3 FDR Through-Hole PCB Minimum Ring

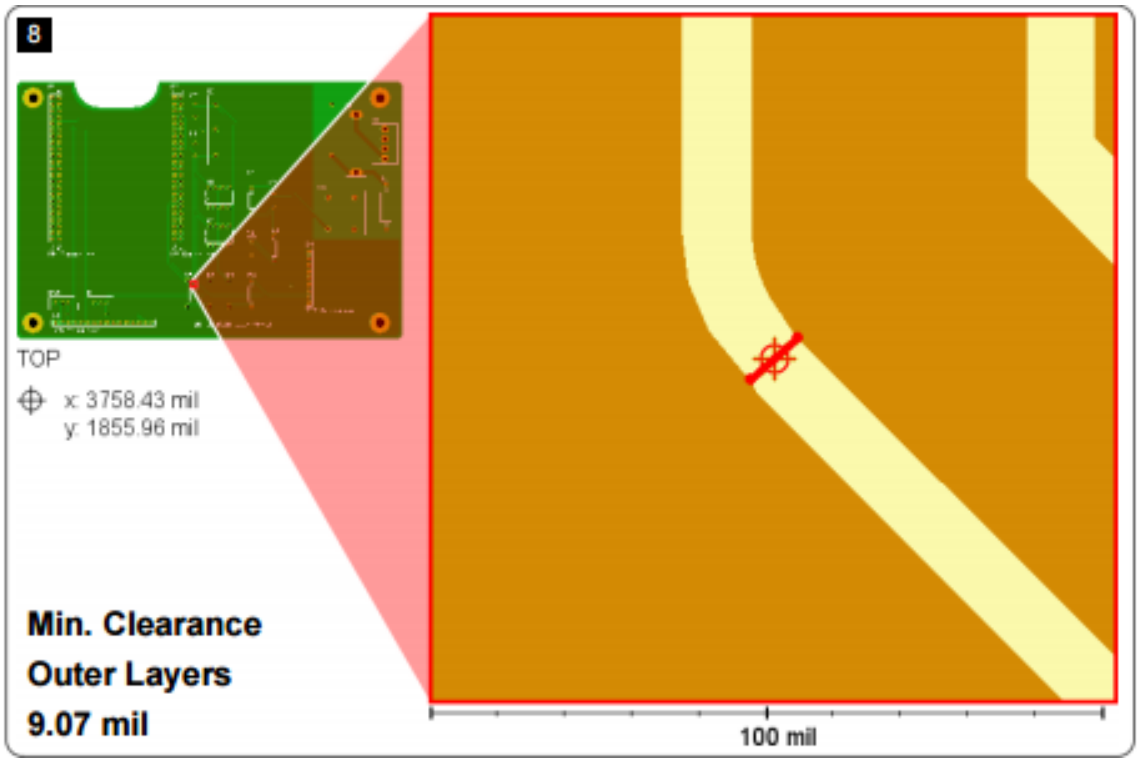


Figure 3-24 Generation-3 FDR Through-Hole PCB Minimum Clearance to Copper

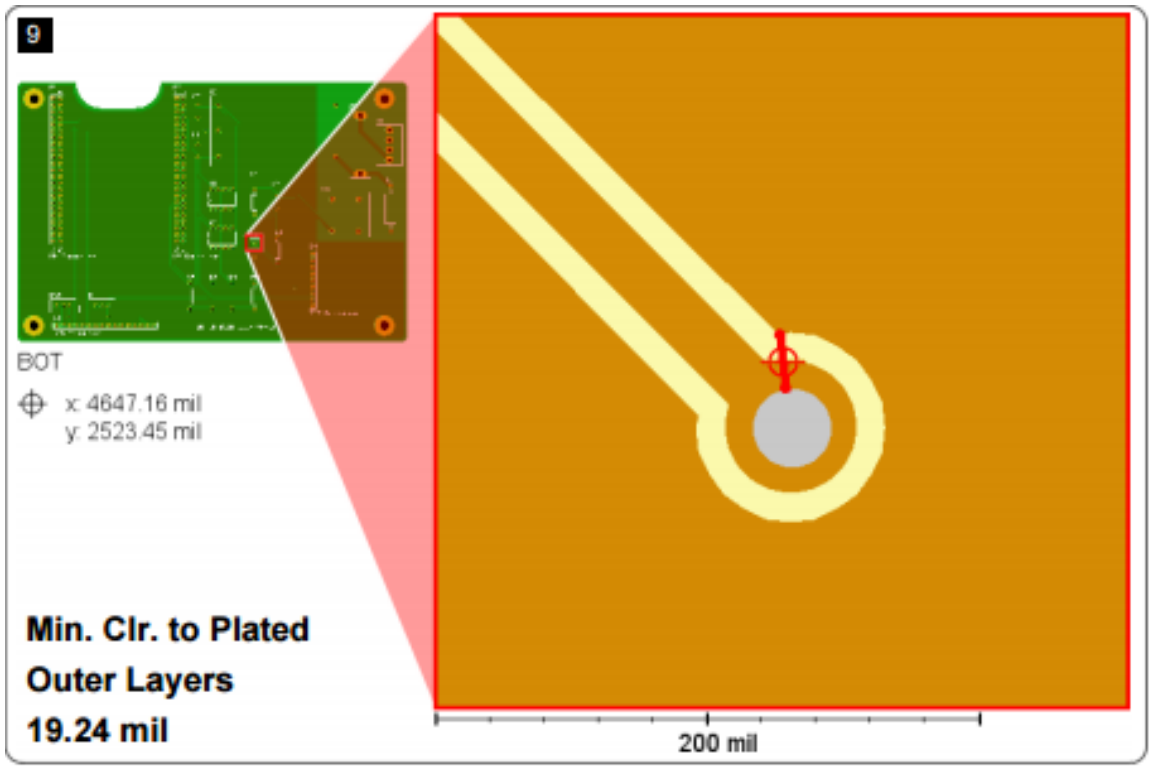


Figure 3-25 Generation-3 FDR Through-Hole PCB Minimum Clearance to Plated

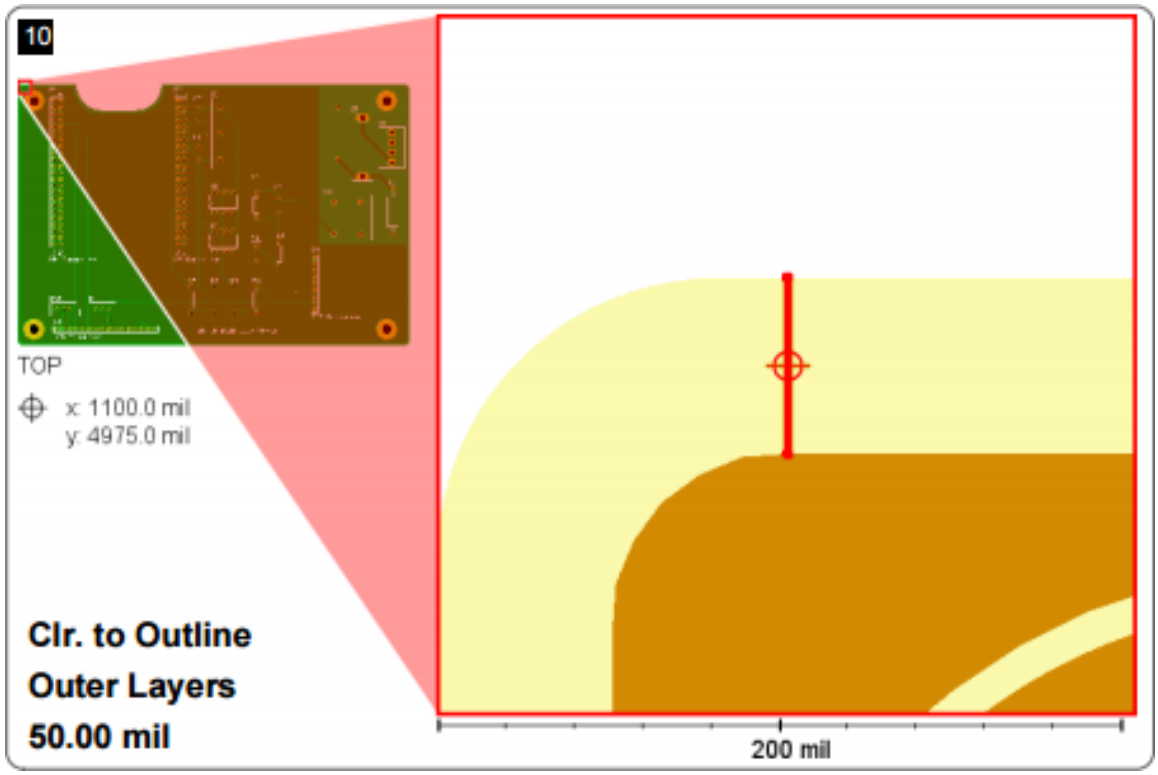


Figure 3-26 Generation-3 FDR Through-Hole PCB Minimum Clearance to Board Outline

Upon receiving the board, the electrical connections were first verified using a signal generator and multimeter, and then the components were soldered onto the board. This includes the subsystems as detailed above in Sections 3.3.1-3.3.3 as well as ancillary components such as headers for the LCD. The GPS module discussed in Section 3.3.1 interfaces with the main board through a series of female header pins. A test via is used to verify the status of the PPS signal. The AC portion of the circuits for both power and signal acquisition are attached to a corner of the board which does not have a solid copper grounding pour. Components with ground connections are instead routed through traces. The LCD used is a generic 20x4 character display which uses a Hitachi 44780 controller – a commonly used alphanumeric dot matrix LCD controller. This particular version [56] of the LCD has backlighting and contrast, but any 16-pin LCD could be used as long as the ribbon cable connections are properly accounted for. The fully assembled prototype PCB (with attached BeagleBone Black) is shown in Figure 3-27. Power, GPS, and LCD external connections are not present in this figure. Resistors [57], capacitors [58], and the signal transformer are soldered directly onto the board, while the op-amps are mounted in IC sockets. This allows for a simple exchange should an op-amp fail or a different IC is determined to be more optimal.

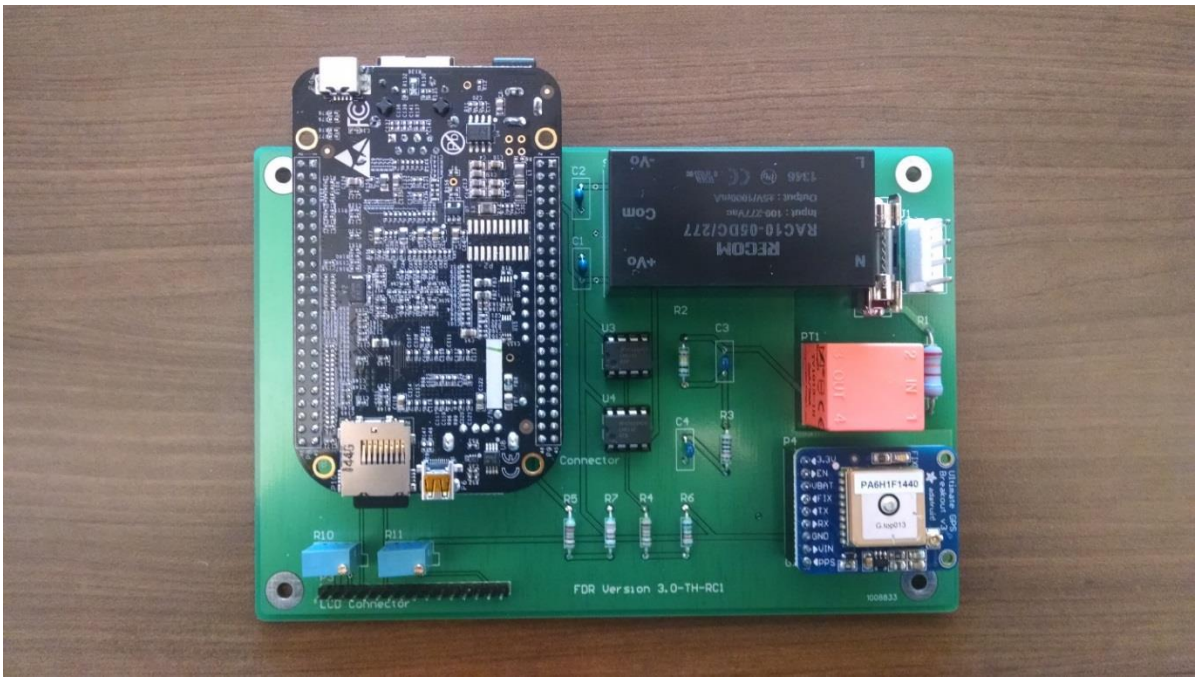


Figure 3-27 Assembled Generation-3 FDR Prototype

The prototype shown here is a through-hole variation, as designated by the TH in the version string. Testing procedure of this prototype is detailed in Chapter 5, Unit Testing and Validation.

4. Next Generation FDR Software

As mentioned in Chapter 3, much of the research and development process for the software directly interfacing with the hardware was performed in concert with that of the hardware itself. Despite the leap in computational power that the AM335x processor offers over similar devices even a few years older, the software must interface with the hardware at a very low level. This is due to the fact that General Purpose Operating Systems (GPOS) do not include specialized driver software to interface with the specific configuration possibilities of embedded platforms.

4.1. Non-Real Time Operating System

Embedded devices with time-critical functionality utilize Real-time Operating Systems (RTOS) in order to ensure that data, such as measurements, are processed in a predictable manner. A RTOS does not by definition mean that the delay of the operating system is completely negligible, only that the system can consistently meet a performance goal – or deadline – in real time. Also of importance is that the randomness in the amount of time to accept and complete a task – termed "jitter," is minimized [59]. The proposed next generation FDR is an embedded device, but it is a fully functional computer with a general purpose processor – the AM335x. This gives it the capability of running more advanced code compared to more primitive embedded devices and various operating systems. While purely real time operating systems, such as QNX, are partially supported on the BeagleBone Black, the level of support at the time of this writing is not sufficient for their use in the FDR. Therefore, the operating system running on the BeagleBone Black must be a variant of Linux, and at its core, Linux is not designed as an RTOS, but rather a General Purpose Operating System (GPOS). Specifically, the Debian Linux distribution was chosen for its wide adoption, stable software packages, and ease of configuration. Both the operating system and distribution ("distro") have implications for embedded development which must be considered in the design of the proposed next generation FDR.

Despite Linux not being a RTOS, it can very often be found in embedded devices which do perform a task repeatedly and consistently in real time. The applicability of Linux in this manner depends on the nature of the task itself. Of particular importance is the timing resolution – a system which needs to monitor the ambient temperature in a room with hourly updates can safely be on a non RTOS, as a reasonably designed system can effectively be guaranteed to take a temperature measurement at some point within that one hour period. On the other end of the spectrum, an industrial control device which must take multiple measurements every second for use in a feedback

control loop would not be an acceptable application. In the case of the next generation FDR, the latter category is more applicable as 10 data points are produced every second.

In general, there are two options for ensuring the suitability of a GPOS in a time-critical task. These options are:

- Ensure that task execution time is deterministic given the "time constant" of the process.
- Apply a software patch to the GPOS designed to convert the underlying scheduler framework into a framework similar to that of a RTOS.

For Linux specifically, series of patches exist which modify the task scheduling portion of the operating system in order to guarantee CPU bandwidth to a process within a specific timing window. This method turns Linux into what is known as a "soft real-time operating system," as there is still no absolute mechanism for guaranteeing task completion within that timeframe. Two of the more prominent patch series are the Linux real-time Pre-emption patches (PREEMPT_RT) [60] and Xenomai [61]. Each of these patch series has a different method for realizing real-time operation in Linux. The Real-time Pre-emption patch integrates directly into the kernel code and minimizes the amount of low-level OS calls which can be pre-empted by the process scheduler. This approach effectively changes kernel-level processes into a machine-language level framework – individual commands attempt to execute with a given overhead. Xenomai, in contrast, originated as an abstract real-time OS core. This process was intended to act as a software RTOS implementation which maintains optimized low level execution on the kernel level, similar to the RTAI (Real Time Application Interface) project [62]. Currently, Xenomai is a kernel-module development framework which is closer in nature to PREEMPT_RT's low-level philosophy, although the core/extension model is still in place.

To determine the relative performance of each of these options, a benchmark was performed using the *cyclictest* utility, a high resolution benchmarking program which determines operating system scheduling latency [63]. For this process, three separate installations of Debian Linux 7.3 "Wheezy" were used – a control installation with no modifications, one with the PREEMPT_RT patch set applied, and one with the Xenomai framework. The underlying Linux kernel for this release of Debian is version 3.8.13. In all cases, the benchmark is assuming minimal background load. This method ensures that the results obtained clearly show the jitter in the scheduling process with no additional processes interfering with the test. The command was executed as follows: `cyclictest –t1 –h 1000 –p 1 –i 500 – l 1000000 > results.log`. This set of parameters specifies a single processor with a priority

level of 1, 500 microseconds between each thread execution, and 1000000 total loops. A histogram is generated and the results are redirected to a logfile. The results are summarized in Table 4-1 and the individual histograms are provided in Figure 4-1, Figure 4-2, and Figure 4-3.

Table 4-1 Cyclictest Benchmark Results

Test	Min Latency	Max Latency	Average Latency	Standard Deviation (σ)
Unpatched	17	407	67.17	11.04
PREEMPT_RT	17	145	53.43	10.63
Xenomai	74	116	85.35	7.44

Initially, these results suggest that the real time patch variants do have an effect on the overall operation of the system, but the major effect is in reducing the maximum latency below a certain threshold. For applications where something must happen within a certain time window and not at precise intervals, this would be an excellent solution. The histograms provide an excellent high level profile of each test result.

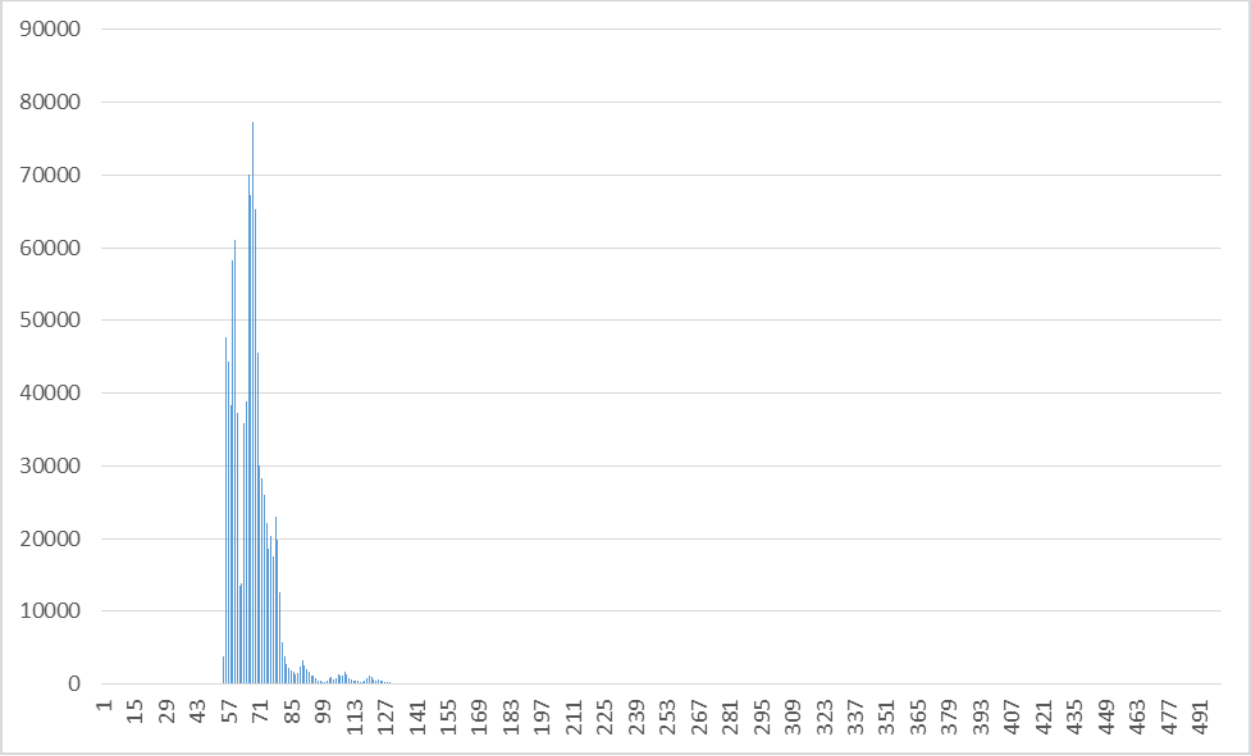


Figure 4-1 Unpatched Linux Installation – Overhead Latency Histogram (µs)

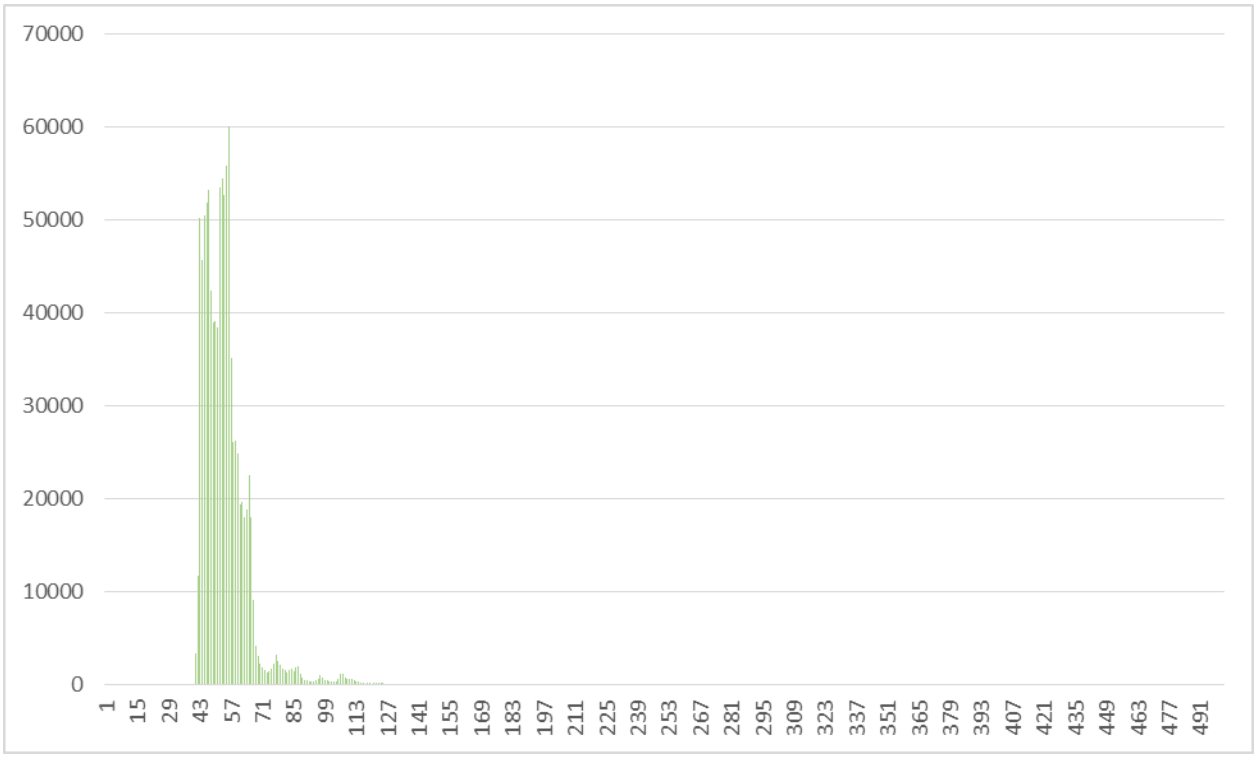


Figure 4-2 PREEMPT_RT Patches – Overhead Latency Histogram (µs)

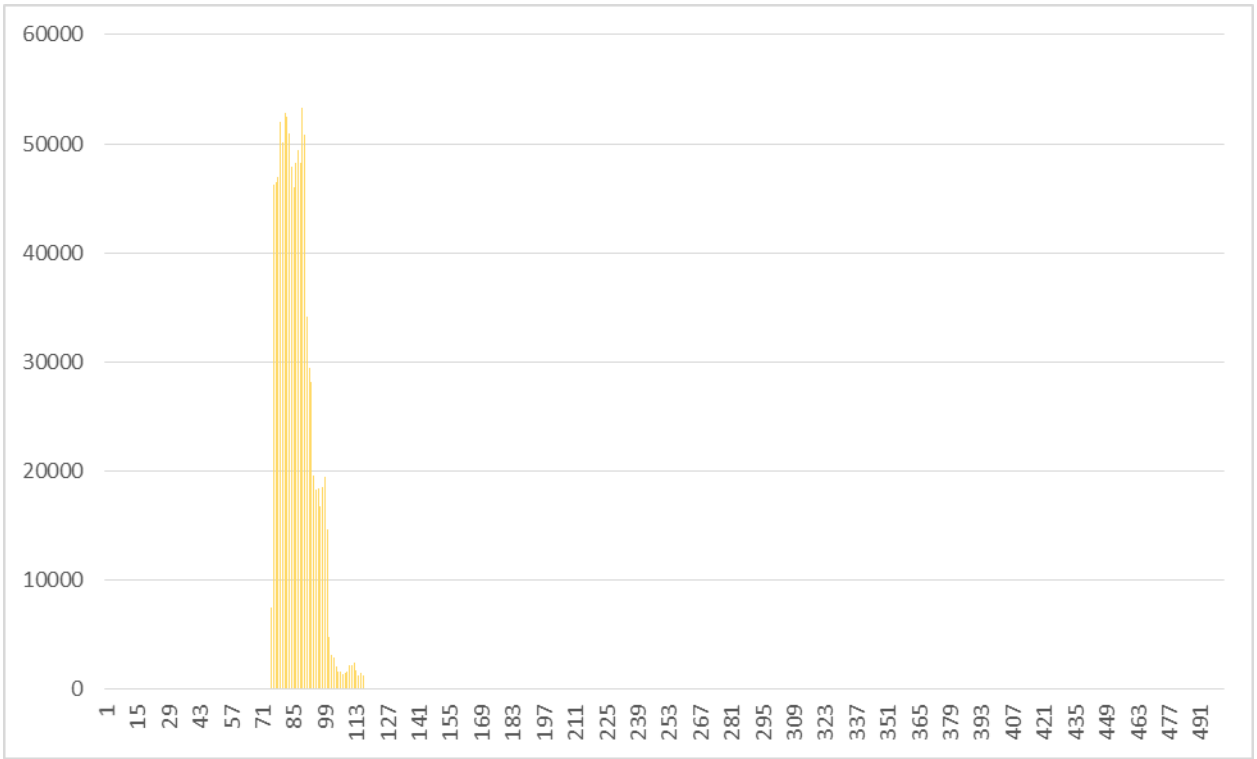


Figure 4-3 Xenomai - Overhead Latency Histogram (µs)

Additionally, Figure 4-4 is a combination of the previous figures and provides a simple visual comparison of the timing effects of each patch.

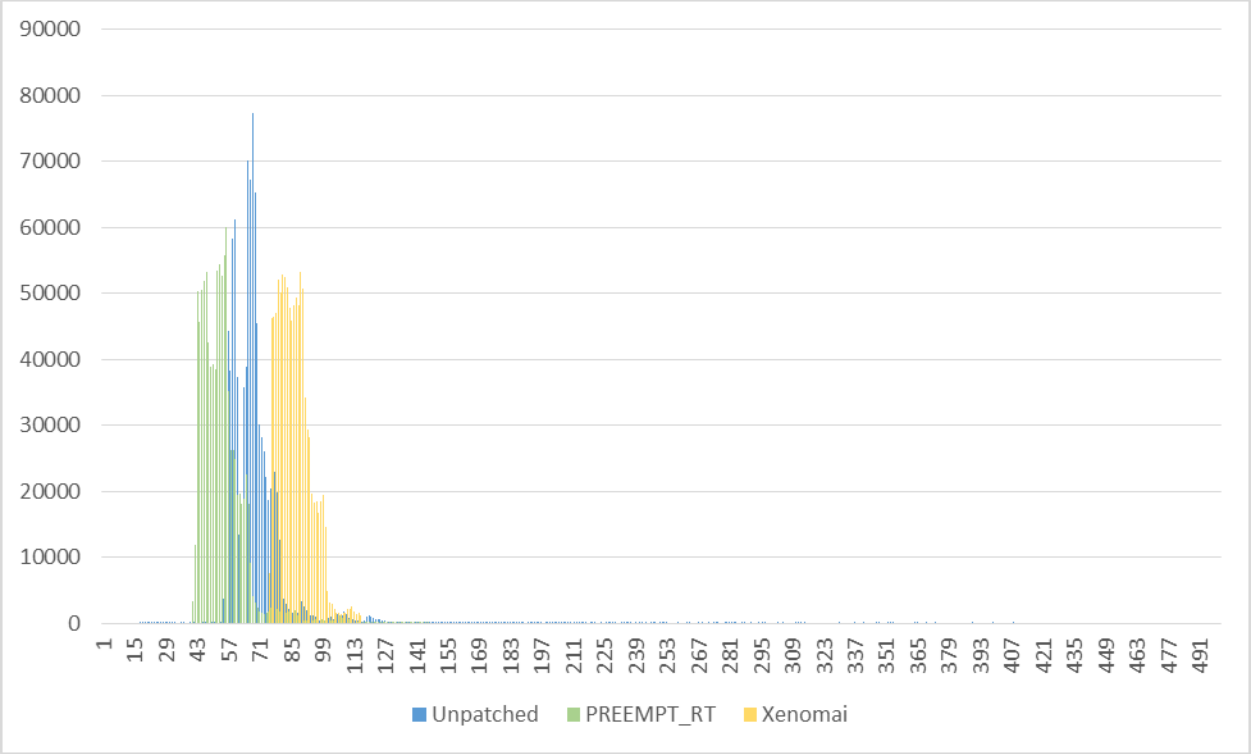


Figure 4-4 Combined Overhead Latency Histogram (µs)

This combination histogram, along with Table 4-1 presents the following conclusions:

- Both patch sets do reduce the standard deviation of the jitter.
- Both patch sets significantly lower the worst-case scheduling latency.
- The patched schedulers smooth out the distribution curve for the latency values, producing a shape with fewer spikes in the data.
- Xenomai actually increases the average latency, but has the smallest total range of values.
- PREEMPT_RT is fastest in the average case, but has a long upper-bound tail.

Unfortunately, while these tests show that task completion can be guaranteed within a given time frame, it does not reduce jitter to sub-microsecond levels. This means that the operating system cannot be in direct control of the measurement process, as it cannot ensure uniform spacing of the signal sampling. This means that although a real-time patch would improve system responsiveness, a subsystem consisting of a dedicated ADC and timing clock is required for real-time operation. Other such

studies have been performed on similar embedded devices and reached the same conclusion [64, 65, 66]. These studies are focused on process scheduling and are not as such directly applicable, but do serve to reinforce the expected performance difference seen with PREEMPT_RT and Xenomai.

As mentioned in Section 3.2.2, the BeagleBone Black does contain an onboard ADC, but the operating system cannot reliably access the samples with low jitter. However, use of the Programmable Real-Time Unit Subsystem (PRUSS) built into the BBB will allow the sampling process to run independently of the calculation and communication process [67]. This will maximize accuracy and consistency of the samples, and any delay in the communication layer is not meaningful, as communication over the Internet already involves significant (relatively speaking) delay. For the purposes of the FDR, this delay is completely acceptable, as the measurements are not used in control actions resolving in sub-millisecond time frames.

4.2. Operating System Setup and Configuration

Using Linux as the FDR's operating system requires some initial setup in order for the BeagleBone Black to function correctly. Notably, the operating system must be made aware of the hardware utilized, which includes chips on the BBB as well as external components connected via the GPIO ports. The system software must be updated to include the basic supporting libraries for Linux as well as the embedded framework used by the PRUSS. This section will explain the basic steps in what is required in the initial operating system setup. Future FDRs can be cloned from an image file and readied with minimal modification. Specific syntax and illustrations for these operations are provided in Appendix B.

4.2.1. Initial Installation and Hardware Support

The initial Debian Linux distribution for the BBB is downloaded from the main support site at elinux.org. There are actually two variants which must be downloaded – the eMMC flasher as well as the raw microSD image. Both of these files are written, or flashed, to microSD cards. First, the eMMC flasher card is used to install Linux onto the BeagleBone Black's onboard SD storage [68]. This operating system is not utilized at any point during the normal operation of the device, but it enables a toggle which then forces the board to attempt to load from the external microSD card for all future starts. Without this operating system, a button on the board must be pressed and held during startup to enable this functionality. Once the image has been successfully written to the BBB's onboard storage, the device is unplugged, and the microSD card containing the raw image file is inserted. Due to the forced-boot behavior provided by the onboard system, the BeagleBone Black will now operate on the microSD card –

the operating system, active programs, and storage are entirely on the removable card. This provides the advantages of speed, modularity and data redundancy. The microSD card supports much faster throughput than the onboard memory, resulting in higher raw performance. Additionally, should the microSD card fail, it can be replaced without affecting the rest of the FDR at a very reasonable price compared to the shipping costs incurred with replacing a current generation failing unit.

The Linux kernel interacts with hardware through a framework known as the "Device Tree." This is essentially a data structure for describing hardware and is loaded into the operating system during the boot process. This means that the operating system does not need to have every detail about all compatible hardware hard coded into the OS itself [69]. Since the goal is to interact with an external GPS unit, a file describing the hardware must be created for the operating system to read in at startup. This requires downloading and installing the Device Tree Compiler (DTC) through the standard Linux package management means – apt-get and then using this program to compile a Device Tree Source (.dts) file into a Device Tree Binary Object (.dtbo). In this case, the Device Tree compiler itself had to be recompiled manually in order to support a necessary feature. This was required due to an outdated version of the software in the upstream repository. Instructions for doing this can be found in Appendix B and the .dts source files are included in Appendix A. Once the .dtbo file is compiled, it is loaded onto the board and then set to run at startup. This is accomplished through the /etc/rc.local file which contains commands to be executed once the operating system has initialized. This file is included in Appendix B and is the configuration used to initialize the Device Tree Overlay files as well as the initialization script for the main FDR loop.

4.2.2. Network Time Protocol and GPS Daemon

The Network Time Protocol (NTP) is a networking protocol used to synchronize a computer system's clock to another reference source. It is one of the oldest Internet protocols still in use and maintains local system time to within tens of milliseconds of the reference clock, despite having up to hundreds of milliseconds of network delay [70]. This is achieved by the use of multiple external references, combined with an intelligent re-polling method, depending on the level of trust (stratum) of the main timing signals [71]. While it is useful functionality to have for many systems, this level of delay is not acceptable for the FDR's time synchronization requirement. However, a critical part of the NTP software package is the Network Time Protocol Daemon (NTPD). In computing terms, a daemon is a software process that runs in the "background," generally invisible to the user, which performs some specific task [72]. This daemon allows for automatic synchronization of time at intervals determined

according to the signal quality, and also provides shared memory access so that other daemons can assist in timing functionality. In this case, the other daemon utilized is the Global Positioning System Daemon (GPSD). This daemon receives data from a GPS receiver and then is able to send that data to any number of applications running on the system, including the NTP service, which can set the host platform's time via the GPS clock [73]. Therefore, inclusion of this daemon allows the next generation FDR to have a local stratum 0 timing signal, as well as a pulse-per-second (PPS) signal for jitter reduction. At the time of this writing, the NTP software package had to be recompiled manually in order to support PPS functionality. As with the DTC, this was required due to an outdated package in the upstream repository. Once property configured, the local time can be queried to ensure that the GPS is functioning properly. The exact sequence and syntax of steps is included in Appendix B. Figure 4-5 shows a screen capture of the query command once the NTP service and GPSD have been properly configured.

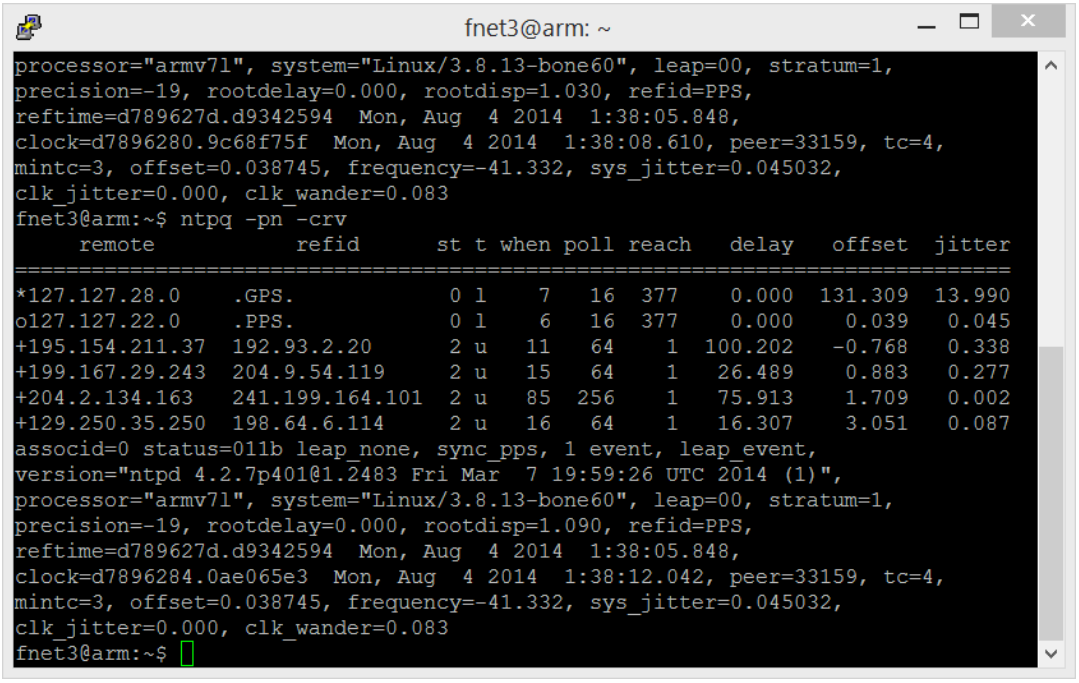


Figure 4-5 NTP Query

4.3. Sampling Considerations

As the primary function of the FDR is to measure the voltage waveform, the methods and tools utilized in acquiring and quantifying the signal are of high importance. When an analog signal is converted to a digital form, information is inevitably lost for two main reasons – discretization and

quantization. Both of these effects are inevitable due to the mathematically complex nature of an analog signal. As an analog signal exists in continuous time, there are effectively an infinite number of samples in any given window of time and likewise infinitesimally small changes in the signal's amplitude between those points. While constructs such as a Fourier series can elegantly represent such complex signals, the underlying computer architecture requires the sampled signal to be in discrete time (**discretization**) with the amplitude having a discrete number of possible levels (**quantization**).

This loss of information thus places an upper limit on the frequency and accuracy of a digitally sampled signal. These limits are established by the Nyquist-Shannon sampling theorem, which states that for a given sample rate f_s, the bandlimit for perfect reconstruction is $B \leq \frac{f_s}{2}$. Alternatively, this means that for a given frequency *B*, there must be *2B* or more samples per second in order to properly sample and reconstruct the waveform [74]. In practice, this lower limit is only theoretical, as noise in the signal will necessitate the use of a higher sampling rate than the bare minimum of *2B* samples per second. If this sampling rate is not achieved, then aliasing can occur – signals which are otherwise different will sample as identical, and the reconstructed signal will be incorrect. Figure 4-6 shows an example of this effect. While not a major concern for the FDR at the nominal frequency, it is an important consideration should future revisions of the device include support for higher-order harmonic analysis.

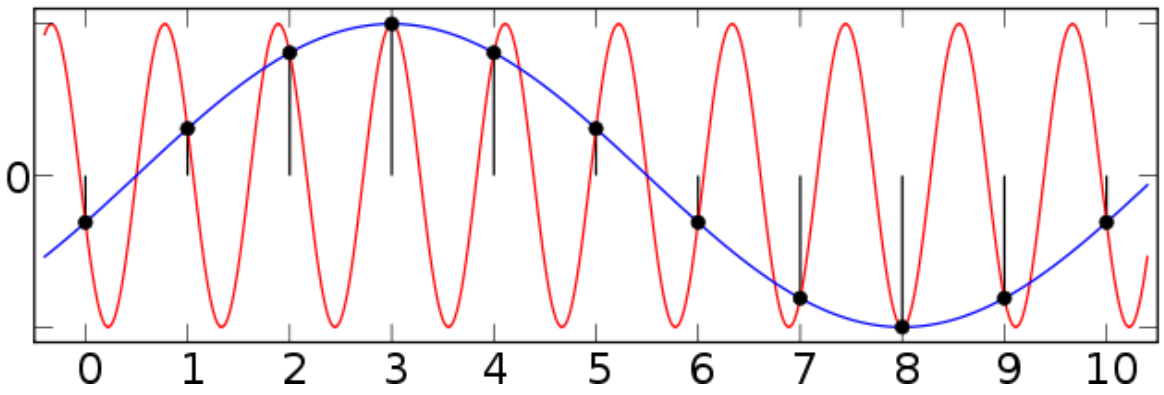


Figure 4-6 Signal Aliasing

The low-pass filter described in Section 3.3.3 has a cutoff frequency of 160 Hz. Naively taking this to be the Nyquist Frequency – the highest frequency that the FDR should be capable of sampling and reconstructing with no artifacts – the resultant minimum required sampling rate is therefore 320 Hz.

The Beaglebone Black's onboard ADC has a sample rate of 2 MHz, so this rate is well within the capabilities of the hardware. In order to fully utilize the hardware and the extra bandwidth available, oversampling can be utilized to increase the resolution of the samples. As mentioned in Section 3.2.2, the onboard ADC has 12 bits of resolution. This means that an analog signal is sampled as one of 2^{12}, or 4096, possible values. Therefore, each quantized level represents approximately 30 millivolts (mV) if the nominal wall voltage is 120V and 54 mV for a 220V signal. After consideration of the 10% safety margin on the signal acquisition circuit, the actual quantization is 33 mV and 60 mV for 120V and 220V, respectively. By using an oversampling algorithm and creating meta-samples which are of a higher resolution, the signal noise can be reduced and aliasing and phase distortion of the signal can be avoided [75]. Specifically, the relationship between an ADC's resolution and signal-to-noise ratio (SNR) is illustrated by the following equation:

$$SNR = (6.02 \times N + 1.76)\ dB$$

Alternatively, the relationship can be written as:

$$ENOB = \frac{(SNR - 1.76)}{6.02\ dB}$$

Where dB is the signal strength in decibels, SNR is the *signal-to-noise ratio*, and ENOB is the *effective number of bits*. The motivation for oversampling in the context of an embedded device is to increase the signal quality as much as the hardware can allow for, as there is no appreciable disadvantage while working within the limits of the chip. Typically the hardware selection is part of the tradeoff process between sampling rate and signal resolution, but as the ADC is already determined in this case, the goal is to maximize the effectiveness of the hardware available. Given the limitations of the hardware itself and the algorithm used to acquire samples, as described in Section 4.4, the new FDR can effectively be oversampled by 1 to 4 bits. The initial implementation as described is oversampled by the maximum of 4 bits, which provides 16 times the effective resolution. To illustrate the effects of oversampling on a signal, show 4-bit and 6-bit signals, respectively. These values are chosen as they are able to convey the effects of oversampling in the context of this document, as higher resolution waveforms would not appear to be visually distinct from one another. The same relative gain is made whenever 2 bits of precision are added – there are 4 times as many possible digital values, regardless of the initial precision.

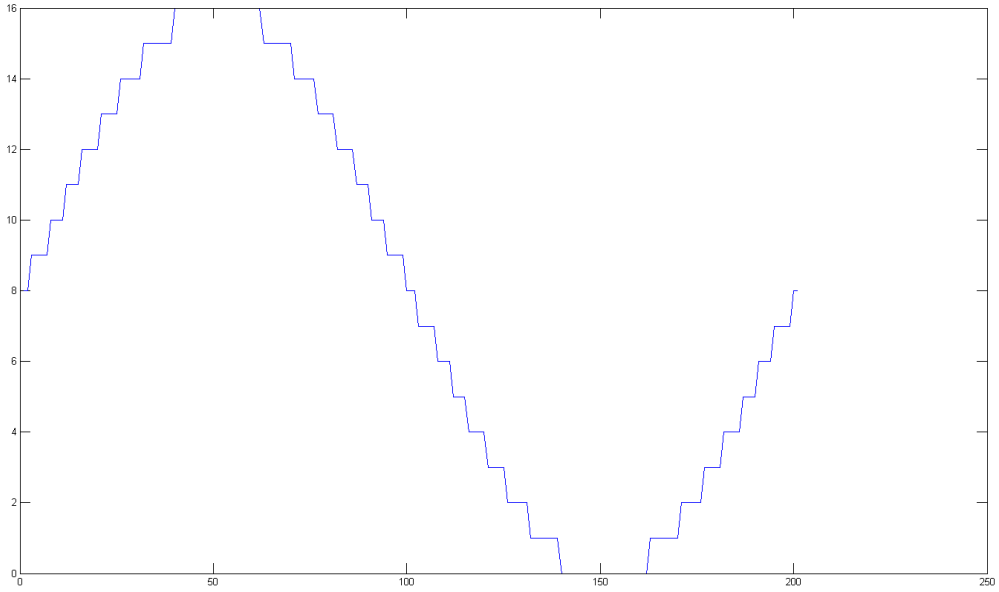


Figure 4-7 Digitized Sine Wave (4-Bit Precision)

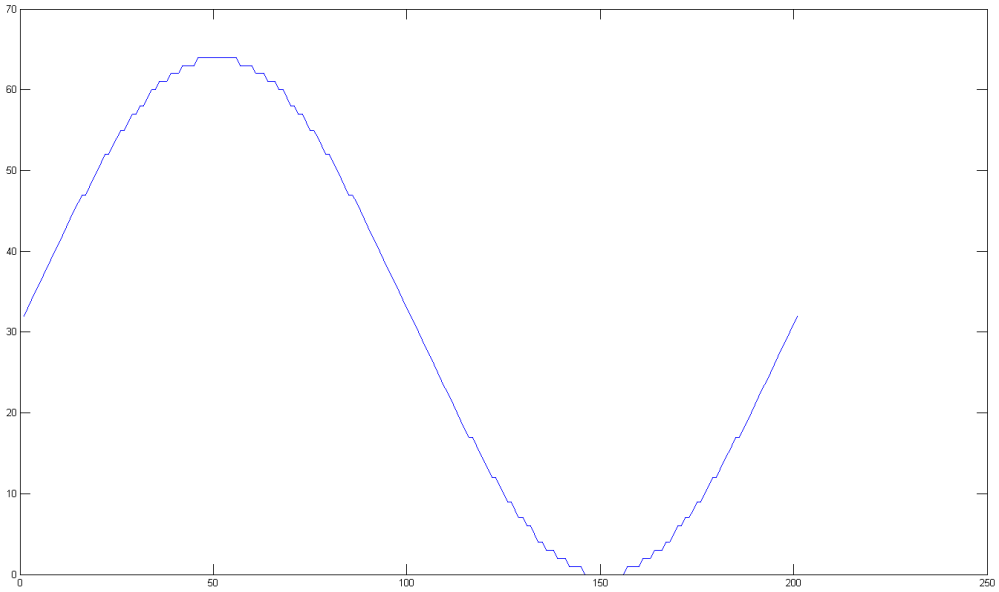


Figure 4-8 Digitized Sine Wave (6-Bit Precision)

In the case of the FDR, oversampling by 4 bits results in 2^{16}, or 65536 distinct levels of quantization. This is a much higher relative gain than the example presented here, but 12 and 16-bit waveforms appear nearly identical with the exception of the y-axis scale. Figure 4-9 shows examples of 12, 14, and 16-bit sinusoidal waveforms. The respective maximum values for each waveform are 4096, 16384, and 65536. These numbers must be taken into consideration as the summing procedure in the algorithm is limited by the size of the individual registers. Fortunately, the 32-bit capability afforded by the PRU allows for a large quantity of 16-bit values to be safely combined with no risk of overflow.

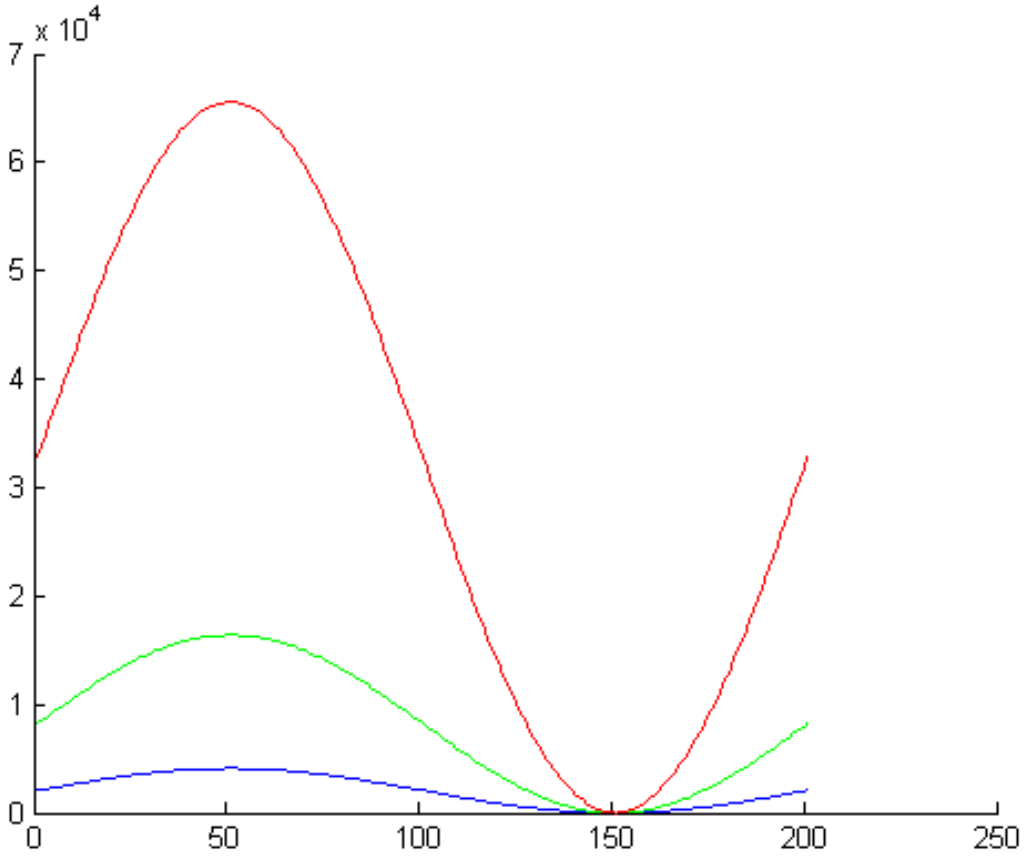


Figure 4-9 Digitized Sine Waves (12, 14, 16 Bit Precision)

4.4. System Software Architecture

The hardware capabilities of the BeagleBone Black allow for a split-program algorithm, with the time-critical computations executing on one of the Programmable Real Time Unit Subsystem (PRUSS) cores. On boot, the hardware runs through an initialization sequence which establishes network and peripheral connectivity. Following this startup sequence, a main program – *FDR_Main* – is executed. This program runs indefinitely while the board is powered on. Despite the name, it is not actually the main functional program on the unit, but rather a local server program used to facilitate communications among the device peripherals and sub-processes, such as external communication and the underlying calculation algorithm. *FDR_Main* communicates with these calculation and communication sub-processes – *FDR_Calc* and *FDR_Comm* – respectively, through the use of UNIX sockets [76]. These are constructs similar to that of a TCP/IP socket, except that there is no routing information contained. This construct predates the modern internet protocols and is used for generalized interprocess communication at a very rapid pace when shared memory is not necessary for the task. *FDR_Calc* handles the sliding Discrete Fourier Transform (DFT) used to calculate the frequency from the raw data. Like *FDR_Main*, it is also run directly from the command line and loops indefinitely while the unit has power. As *FDR_Calc* is still running within the context of the operating system, it does not directly perform the ADC measurement. Instead, it initializes a subtask – *FDR_Read*, which is an assembly program operating on the first Core of the PRUSS. This time critical program communicates with the adjoining C program through the use of a shared memory pool. This method of communication not only ensures maximum communication speed, but is necessary as other constructs such as UNIX sockets are outside the capabilities of the embedded code.

The overall communication architecture is one way – *FDR_Read* and *FDR_Calc* communicate through a double-buffer loop. Using two buffers for data means that the assembly program is writing a new set of data as the calculation program is reading from the previous set. However, in all cases, *FDR_Read* writes all of the data, including status flags, and *FDR_Calc* reads the data and performs the calculations. As the calculation algorithm runs in system time, it cannot be used to reliably change status flags which would relate to real time operation. Likewise, all communication between *FDR_Calc* and the adjoining *FDR_Main* program is also one way – *FDR_Calc* sends information to *FDR_Main* but at present does not receive any information from it. Additionally, the link between *FDR_Main* and *FDR_Comm* is one way as well, as this prevents the subprocess from blocking local operation. Figure 4-10 shows this communication hierarchy as well as the hardware interfacing performed by each layer.

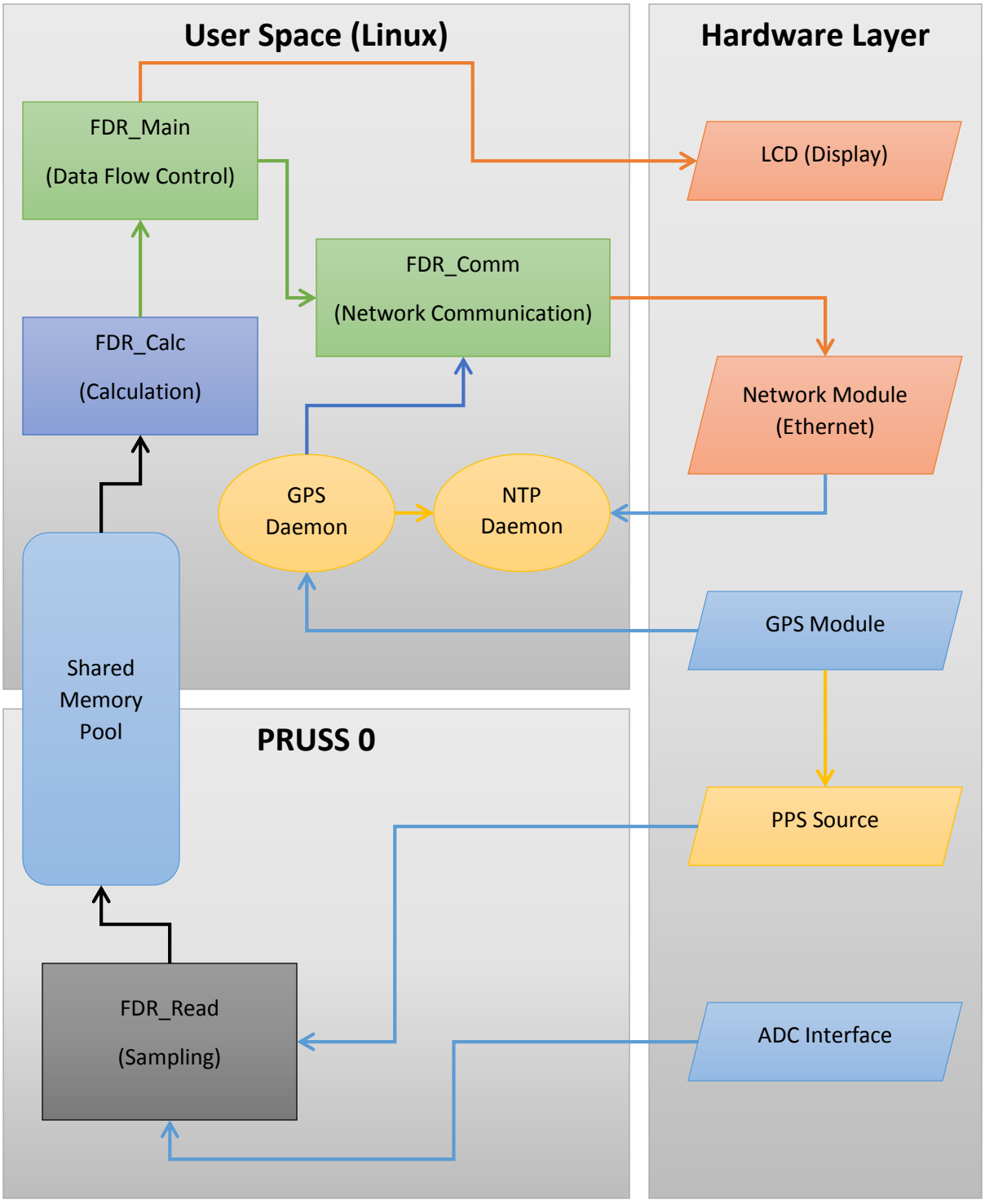


Figure 4-10 Overall Data Flow Model for Generation-3 FDR

This model is used to promote software abstraction – each layer should only be concerned with its directly assigned task. This modularizes the design and promotes information stability. Additionally, the main layer – written in Python, can be used to easily redirect data without modifying the underlying calculation algorithm. One such example of this is the unit's external Liquid Crystal Display (LCD) which frequency and voltage information is periodically written to. This allows for the capabilities of the device to be expanded by the PowerIT lab while minimizing the chance that a bug or mistake will be inadvertently introduced into the critical functions of the device.

4.4.1. PRU Layer: FDR_Read.p

The low-level sampling algorithm runs entirely on one of the Beaglebone Black's PRUSS modules, allowing it to operate continuously with no interruption or delays from the Operating System. This program is executed from within the *FDR_Calc* program, which is detailed in Section 4.4.2. As there are 10 measurements computed each second, the program must save batches of measurement data into the shared memory pool for analysis every 100ms. Therefore, timing for this program is determined by the following factors:

- Clock speed of the PRU – nominally 200 MHz.
- Frequency of the power grid – nominally 50 or 60 Hz.
- Number of samples per signal cycle.
- Number of subsamples per sample.
- CPU clock cycles used for an instruction.

The two-buffer design means that with proper timings, the program will run through five buffer loops every second. Therefore, the algorithm keeps track of the number of samples as well as the number of total buffer cycles which have passed. After five two-buffer loops, the machine waits for a PPS signal to begin again. A flowchart illustrating this process is shown in Figure 4-11. There is no break condition included in this process, although if the PRU itself is reset, then the program will halt.

Timing control in an embedded program is typically handled through one of two methods – interrupts and delay methods. The program here uses both of these – the PPS signal is essentially a hardware interrupt, and the system idles while waiting for a bit to be set. During the subsample and sample acquisition loops, a delay routine is used to ensure even spacing between measurement points.

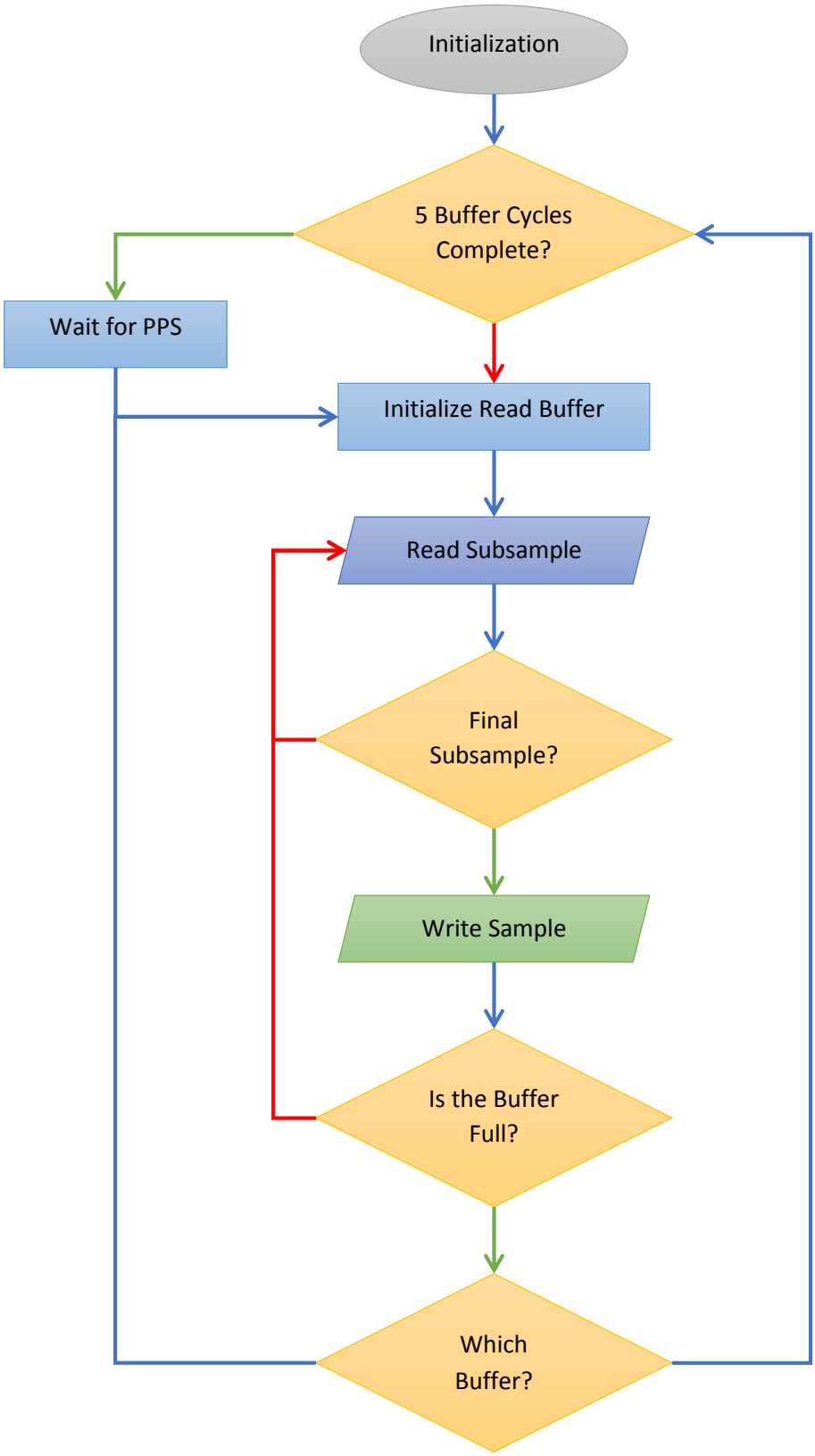


Figure 4-11 FDR_Read Program Flowchart

As there are multiple decision points and paths for the low-level program, the delay subroutine must be altered to accommodate the differences in execution time for each particular path. The path length for each loop is determined in a straightforward manner by summing the instruction counts over the course of the path. To simplify matters, the PRU Assembly language has a limited instruction set and all instructions except for memory access require exactly one cycle. The exceptions to this are Load Byte Burst (LBBO), Load Byte Burst with Constant Offset (LBCO), Store Byte Burst (SBBO), and Store Byte Burst with Constant Offset (SBCO). Load instructions require 2 + <word count> cycles and Store instructions require 1 + <word count> cycles [77]. The *FDR_Read* and *FDR_Calc* programs contain constants which may be adjusted to alter the sampling rate or sample count of the unit. The default parameters are presented in Table 4-2.

Table 4-2 Default FDR_Read Parameters

Name	Default Value	Assignment	Notes
BUFF_SIZE	0x0000B40	User Defined	
HALF_SIZE	0x0000A20	Formula	No User Input
SAMPLING_RATE	3600	User Defined	
SUBSAMPLES	256	User Defined	
SHIFT_AMT	4	User Defined	Square Root of SUBSAMPLES
RAW_SAMPLING_RATE	921600	Formula	No User Input
CLOCKS_PER_LOOP	2	Constant	Never Changes
DELAY_COUNT1	166	User Defined	Divided by 2
DELAY_COUNT2	418	User Defined	Divided by 2
DELAY_COUNT3	256	User Defined	Divided by 2
DELAY_COUNT4	254	User Defined	Divided by 2

To calculate these parameters, the number of total subsamples must be determined, which is simply **SAMPLING_RATE** × **SUBSAMPLES**. The **BUFF_SIZE** field is the memory pool for both buffers, and must be changed along with **SAMPLING_RATE**. The formula do to this easily is **SAMPLING_RATE** ÷ 10 × 8. This ensures that each buffer holds all of the required samples for the 100ms time frame. The

SUBSAMPLES field is determined by how much oversampling is being performed. This number is essentially $\mathbf{4^{SHIFT_AMT}}$ where **SHIFT_AMT** is the number of bits being added to the precision of the measurement. With the default parameters, the 12-bit resolution is being increased to 16 bits. The **CLOCKS_PER_LOOP** field is simply a reference for clarity in delay loop calculations. The **DELAYCOUNT** fields should be determined by number of operations in a given code path, and then dividing that by 2 – the number of operations per delay loop – ensures that no additional math is required beyond of the operation count. Essentially, the numbers are determined via the following guidelines:

- [(48 + **DELAY_COUNT1**) × (**SUBSAMPLES** - 1) + (60 + **DELAY_COUNT<2,3,4>**)] × **SAMPLING_RATE** should be approximately 200×10^6, or 200 MHz.
- **DELAY_COUNT4** = **DELAY_COUNT3** – 4
- **DELAY_COUNT3** = **DELAY_COUNT2** – 2
- **DELAY_COUNT1** and **DELAY_COUNT<2,3,4>** should be as close to equal as possible.
- The result of the calculation should never be higher than 200×10^6.

The total cycle count should not sum to exactly 200 MHz, as there are variations in processor speed which could result in improper operation of the unit. These variations are both environmental (temperature) and internal to the system (silicon variations). In all cases, the measurement for a given loop is taken before the delay routine. It is important to note that in the event of a full buffer, the status flag is also set before the delay timer, which ensures that the memory can be read by the main program as soon as possible.

4.4.2. C Layer: FDR_Calc.c

The calculation algorithm is run within the context of the operating system. It is initially executed after *FDR_Main* from the command line – the local server program. *FDR_Calc* is effectively the wrapper for the sample measurement algorithm and communicates with the PRU Subsystem via a Shared Memory Pool. As previously mentioned, after initialization, the communication is one way – the PRU program writes buffer status flags to the pool, and the main program reads them to control when data processing and interprocess communication with *FDR_Main* occurs. The overall program flow is outlined in Figure 4-12.

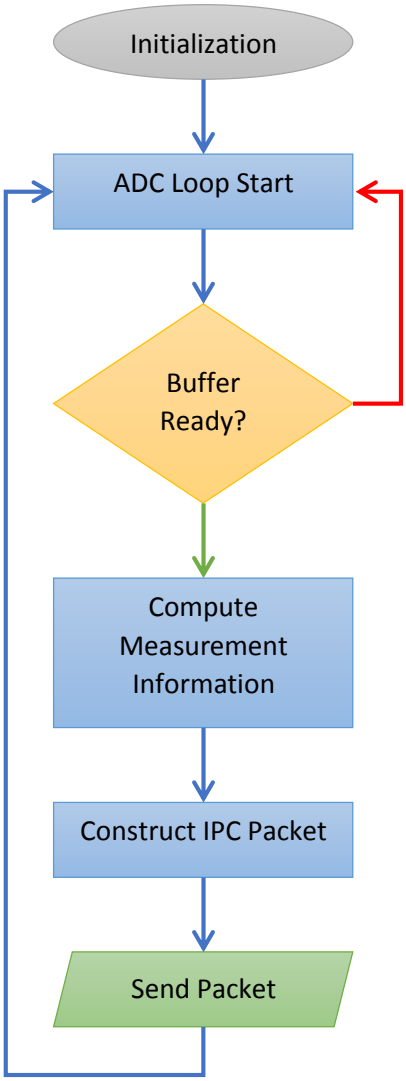


Figure 4-12 FDR_Calc Program Flowchart

As with the PRU module, the initialization step of *FDR_Calc* prepares the program for constant iteration. This includes opening the socket connection to the *FDR_Main* process, generating the trigonometry tables for use in the frequency calculation subroutine, and initializing the PRU Subsystem and Shared Memory Pool. The PRU program is then executed and the program enters the ADC loop. This main loop waits for the buffer flag to be set, and then computes the Sliding Discrete Fourier Transform for the data set, and subsequently the frequency measurement. Each of these is written as a separate function, operating on a statically held data set – this allows for program modularity so that alternate functions can be utilized or additional functionality added to the overall loop.

The frequency measurement computation is an integral part of the overall algorithm, and is contained in a separate function so as to be easily altered or upgraded. Given that the FDR must compute and send 10 measurements per second, computational efficiency in the algorithm is of high importance. As already mentioned, the algorithm utilized is the Sliding Discrete Fourier Transform, or Sliding DFT. This algorithm is a computationally efficient variation on the standard DFT, with the caveat that it is only applicable with a continuous stream of data.

Traditionally, the algorithm for the Fourier transform involves O(N) operations for every individual point. This means that the overall computational complexity of the standard algorithm is $O(N^2)$. In contrast, the Fast Fourier Transform, or FFT, has a computational complexity of O(N log N). Mathematically, the equation for the Discrete Fourier Transform is as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\left(\frac{2\pi}{N}\right)nk} \text{ (k = 0, 1, …, N-1).}$$

The term "Fast Fourier Transform" is applied to any algorithm which computes the DFT in O(N log N) time. This is often accomplished by taking advantage of the inherent symmetry of the equation and utilizing a "divide and conquer" strategy. In contrast, the Sliding DFT utilizes a principle known as the DFT shifting theorem, or the circular shift property [78]. Given a finite-length windowed time-domain sequence and its DFT $X(k)$, the DFT of that sequence circularly shifted by one sample, is $X(k)e^{j2\pi k/N}$. Therefore, the spectral components of the shifted time sequence are the original spectral components multiplied by $e^{j2\pi k/N}$, where *k* is the DFT bin of interest. Mathematically, this process is expressed as:

$$S_k(n) = S_k(n-1)e^{\frac{j2\pi k}{N}} - x(n-N) + x(n)$$

Where $S_k(n)$ is the new spectral component and $S_k(n-1)$ is the previous spectral component [79]. The computational algorithm for this equation is handled differently, as the processor does not deal directly with numbers in the complex plane. Instead, the components of the sample are split into real and imaginary components. This is accomplished through the use of pre-calculated trigonometry tables. As the sampling rate is chosen in advance, the tables can be generated once during initialization and used for all subsequent calculations. The initial real and imaginary sample quantities – **init_real** and **init_imag** are computed through summation over one full cycle of data. Once these initial quantities have been determined, the first sample within the sliding window is calculated and the components

stored in the **sample_real[0]**, **sample_imag[0]**, and **sample_angle[0]** variables. This is achieved by removing the contribution of the first sample from the init_real and init_imag values and then adding the contribution of the first sample within the target window, effectively shifting the window. Once the first sample is calculated, all subsequent samples (n) are determined using the sample immediately preceding itself (n-1). The pseudocode for this process is as follows:

```
sample_real[n] = sample_real[n - 1] + 2 / SAMPLES_PER_CYCLE *
(sample[SAMPLES_PER_CYCLE + n] - sample[n]) * cos_table[n];

sample_imag[n] = sample_imag[n - 1] - 2 / SAMPLES_PER_CYCLE *
(sample[SAMPLES_PER_CYCLE + n] - sample[n]) * sin_table[n];

sample_angle[n] = arctan(sample_imag[n], sample_real[n]);
```

The source code used differs from the above in that it includes memory offset values and the sample constant of 2 / SAMPLES_PER_CYCLE is pre-computed for the purposes of efficiency. A visual representation of this process is shown below in Figure 4-13.

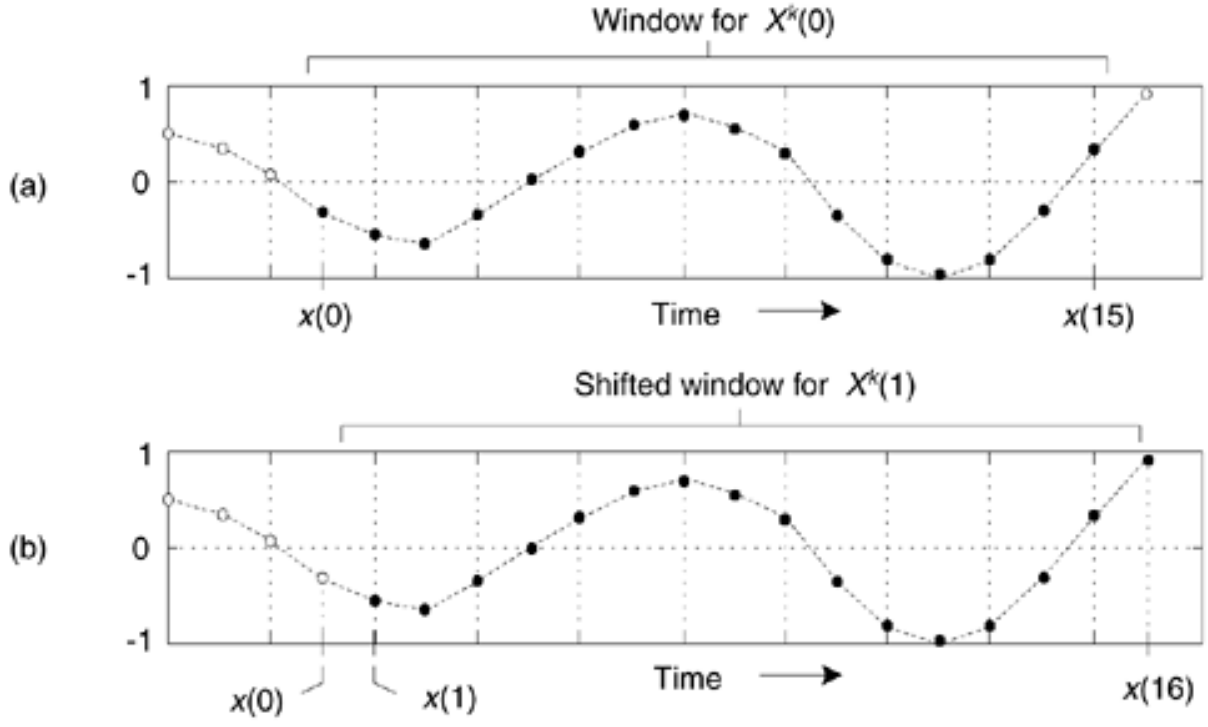


Figure 4-13 Representation of Sliding DFT

Once the spectral values have been determined using the sliding DFT, the frequency computation is performed. Nominally, given a sinusoidal signal, frequency is the rate of change of the angle. Mathematically, this can be represented as:

$$\varphi(t) = a_0 + a_1 t + a_2 t^2$$

$$f(t) = \varphi'(t) = a_1 + 2a_2 t$$

As the algorithm deals with discrete sample quantities, this process can be discretized as follows:

$$f(k) = A_1 s_k + 2A_2 s_k$$

Where A_1 and A_2 are pre-calculated constant values, and s_k is a discrete sample angle. As the sampling rate is predetermined, the time value can be used in the initial calculation, removing it from the formulation. It is instead included in A_2. Frequency computation is performed by iterating over the sample window and calculating the frequency at points evenly spaced over the window. The final frequency is determined by averaging the samples taken over the window. The resulting computation is then normalized and added to the nominal frequency. In the case of a resample, the initial frequency measurement is used rather than the nominal frequency. In most cases, there is no appreciable difference, however if the system greatly deviates from the nominal frequency, a resampling operation can provide a more accurate measurement. A downsampling factor is utilized in the algorithm to minimize the total time spent in this operation. These tables are generated and stored at the start of the program in the **a1_table** and **a2_table** variables.

4.4.3. Python Layer: FDR_Main.py

The main FDR process is run in the context of the operating system and is a non-real time process that is handled by the process scheduler. It is executed as a background process from the command line handles the overall coordination of the data flow within the device. This functionality includes initialization of the unit's peripheral hardware, such as the LCD. Two UNIXs socket are opened for interprocess communication with the calculation module (FDR_Calc) and the communication module

(FDR_Comm). These sockets allow for interprocess communication and real-time data flow. As previously mentioned in Section 4.4, the communication module is a separate process to ensure that any errors in the data transmission process does not adversely affect the local functionality of the device. Within the main loop, a UNIX socket receives data from the calculation program *FDR_Calc*. From this point, the data values are stored in local variables and then used for the LCD and network communication.

Internally, each measurement is stored in a temporary instance of *MeasurementEntry*. This class allows for object-based serialization of data. The purpose of this is for data integrity as well as future extensibility. Time-tagging individual measurements prior to packet construction ensures that any communication failure does not incorrectly shift the measurement. Additionally, the serialization method offered by this class allows for data to be retransmitted easily as well as archived locally for potential calculation uses. The prototype unit does not maintain local archive for measurement data, but the size of the microSD card potentially allows for such functionality. In the event of failed transmission, the archive could be used to re-send data when a connection with the remote server can be reestablished.

If the sub-second packet counter is equal to 0, which happens with the first measurement every second, then the frequency and voltage are formatted as Strings and then sent to the LCD for display. This is handled using a generic Python library for the HD44780 LCD controller [80]. While this is a commonly used controller, and is the hardware utilized in the prototype Generation-3 device, the driver library functionality can be changed to suit alternate LCD controllers. For readability reasons, the external LCD is only updated every second rather than updating for each new data packet. This matches the existing Generation-2 FDRs, which implemented the 1-second timing for the same reason. Once data values have been read and sent to the LCD, the same data packet is retransmitted to the external communication module (FDR_Comm) for external transmission. Program flow for this main module is shown in Figure 4-14. Compared to the calculation algorithm, this is an extremely simplistic program, but in many ways acts as an analog to the C8051 CPU present in the current generation FDRs. In serving as a router for the data streams, consistent operation can be ensured, and a modular software design can be used to add further peripherals or communication channels to the device.

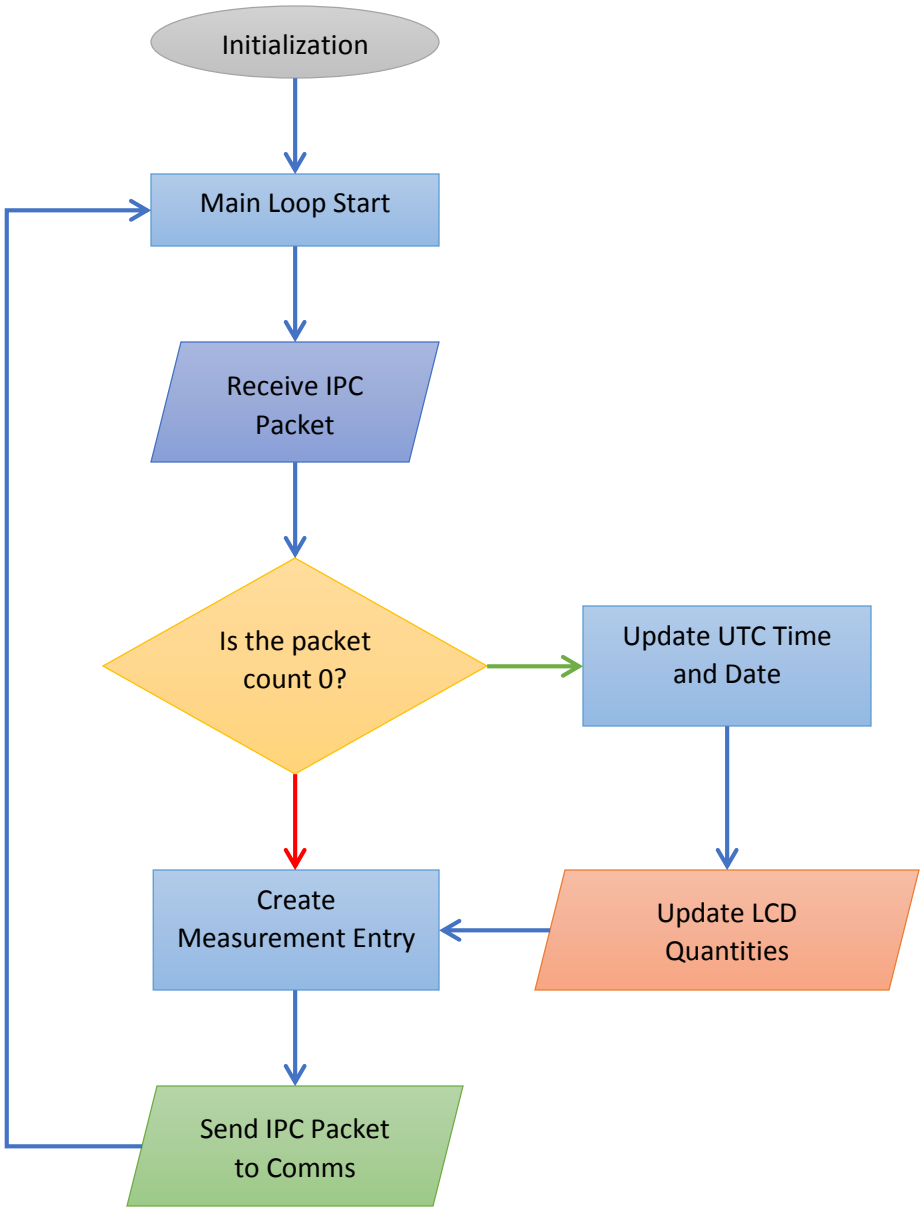


Figure 4-14 FDR_Main Program Flowchart

The overall purpose of the *FDR_Main* program is to act as a mediator between the device, its sub-processes, and the outside world. For the prototype unit, this means providing local information via the LCD controller and passing on information to be communicated via Ethernet. The structure of this program is left open for modification in the event that local interaction with a unit – potentially via a keyboard or touchscreen is desired. By limiting external interactions to this process, the risks of new

peripheral code adversely affecting the calculation algorithm are negated. Additionally, as a background process, the *FDR_Main* process can operate at a much slower speed, and consequently, a lower priority, than the calculation algorithm. The “time constant” for human-computer interaction is much higher than that used by the frequency computations..

4.4.4. Python Layer: FDR_Comm.py

The communication module is in many ways similar to the main process – it is run from within the context of the operating system, allowing for the process scheduler to control the execution. As such, it is also a non-real time process and the time from initial measurement and calculation to packet transmission is variable. In the context of the entire FNET system, this is of no concern as the measurements are time-stamped and transmission of data over the Internet already involves variable and unpredictable communication delays.

Upon initialization, the process reads two external files – unit configuration (unit_config) and server configuration (server_config). These two files tell the process which unit to identify as when transmitting data and where that data should be transmitted to. As previously mentioned, the *unit_config* file is also utilized by the *FDR_Calc* program to read the nominal frequency, voltage, and the voltage calibration constant. The communication layer only needs to know the Unit’s ID number, but the file is a shared resource to allow for easier configuration. The *server_config* file is a resource unique to *FDR_Comm*. Data entries in this file are stored in 4 columns: the type of packet, IP address, port number, and whether or not encrypted communication should be used. Examples of both the *unit_config* and *server_config* files are provided in Appendix B.

Once the configuration options have been read by the program, an array of *ServerEntry* objects is created. This object is a structure which encapsulates all necessary information about a given connection – the packet type, IP address, port number, and security setting. It is essentially the in-memory representation of a single line of the *server_config* file. By enumerating over this array, the required sockets are first created and remote connections are established. This array is also enumerated when IPC communication is received. The basic overall operation of the process is to wait for the IPC packet from *FDR_Main* and then construct and send appropriate packets to all of the specified servers using the desired protocols. A flowchart detailing the operation of the program is presented in Figure 4-15.

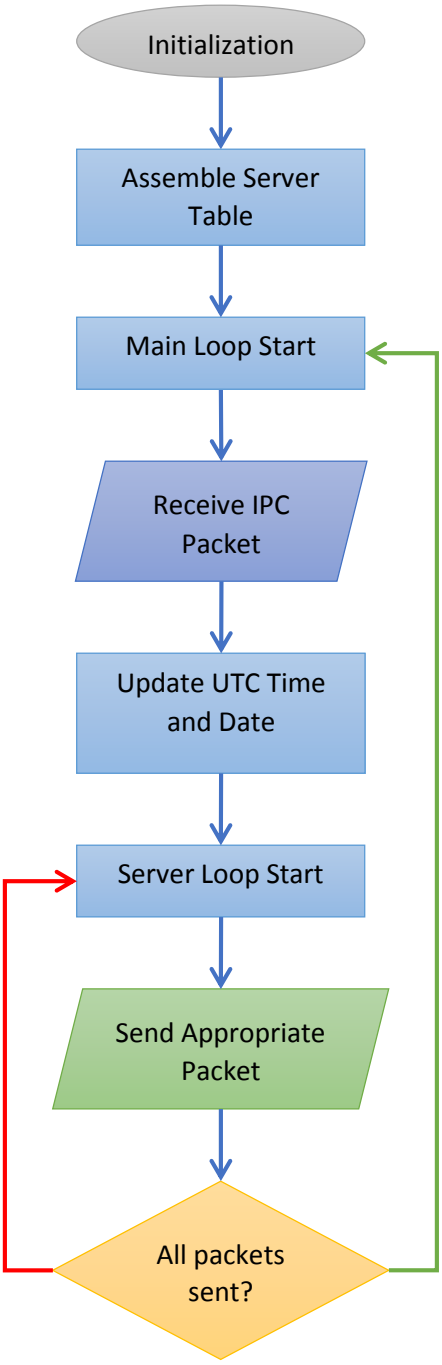


Figure 4-15 FDR_Comm Program Flowchart

For external communications, the program re-encapsulates the data into a format suitable for transmission over Ethernet. Currently, both FNET packets and C37.118 packets are supported. These packet constructions are handled within their own functions for the purposes of modularity and flexibility should the FNET or C37.118 packet structure be redesigned at a future point. Each packet is a data frame which is later wrapped in a TCP/IP header. For FNET packets, the data values are written to a buffer as ASCII text and the encapsulated data values are essentially a single string of characters. The FNET frame sizing information is shown in Table 4-3.

Table 4-3 FDR Frame Description - Format

Start Byte	UnitID	Date	Time	ConvNum	Variable	Frequency	Voltage	Angle	Stop Byte
1	3	6	6	2	*	7	8	6	1

Additionally, there is a 1-byte buffer between each field, with the exception of **Angle** and Stop byte. There are four different frame descriptions, each one utilizing the **Variable** field in a different manner: Latitude, Longitude, Satellite Lock, and Standard. The Latitude packet is sent as the first message of every 10 second cycle, Longitude as the second, and Satellite Lock as the third. The 4th through 10th packets are sent with Standard frames and contain an initial frequency measurement which can be used for debug information. The total frame size, considering usage of the **Variable** field is provided in Table 4-4.

Table 4-4 FDR Frame Description - Sizing

Packet Type	Packet Message	Variable Size	Total Size
Longitude	1	8	56
Latitude	2	9	57
Satellite Lock	3	2	50
Normal	4+	7	55

For C37.118 communications, data values are packaged according to the standards set forth in the most recent revision of the standard – C37.118.2-2011. Per this standard, all messages begin with a 2-byte SYNC word followed by a 2-byte FRAMESIZE word, a 2-byte IDCODE, and 8 bytes of timing

information. Within the timing information, the word is subdivided into the “second of century,” or SOC, the fraction of the second, or FRACSEC, which uses 24 bits, and the remaining 8 bits, which are used for Time Quality Flags. These 14 bytes of header information are present in all C37 messages. An amount of data up to approximately 64 kilobytes may follow this header information. Completing the frame, there is a 2-byte checksum value. Figure 4-16 shows the generalized frame transmission order for all C37 frames.

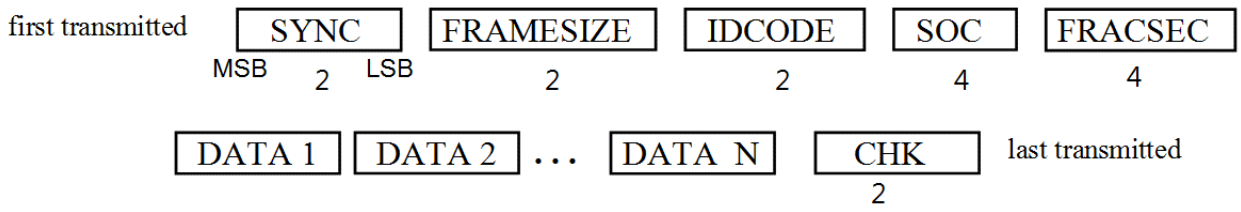


Figure 4-16 C37.118 Frame Transmission Order

As mentioned in Section 2.2.1, the C37 framework is highly generalized and is meant to operate on all types of PMU regardless of manufacturer or installation. Therefore, the individual fields may be of varying length, depending on fixed or floating-point measurements, and a varying number of measurements may be sent. Table 4-5 shows the generalized structure of a C37.118 data frame.

In the case of the next-generation FDR, much of the functionality specified by the standard is not applicable. The units have a specific functionality and no external measurements are relayed through the device. With this in mind, there is no need for the prototype unit to support configuration frames. Specifically, the data frames are pre-configured to send the same information that is present in the FNET data frame – a frequency measurement, a voltage magnitude, and a phase angle. Unlike the FNET unit, the data values are written as specified by the standard and shown in Table 4-5, not as ASCII text. By adhering to the standard, the same data can effectively be transmitted while using a smaller amount of total bandwidth. As constructed, the C37.118 compliant frames are 32 bytes, compared with the 50-55 bytes used for each FNET frame. While this is illustrated directly by the respective tables, it is not quite as apparent that the 4-byte floating point values actually have higher precision. When represented as ASCII text, the values are truncated to 4 decimal places, although the binary representation is actually more precise. This is because floating-point values are encoded with 23 bits of precision, with 8 bits being used for the exponent and the final bit used to represent the sign.

Table 4-5 C37.118 Data Frame Description

Order	Field	Size (bytes)	Comment
1	SYNC	2	Sync byte followed by frame type and version number.
2	FRAMESIZE	2	Number of bytes in frame.
3	IDCODE	2	Stream source ID number, 16-bit integer.
4	SOC	4	SOC time stamp, for all measurements in frame.
5	FRACSEC	4	Fraction of Second and Time Quality, for all measurements in frame.
6	STAT	2	Bit-mapped flags.
7	PHASORS	4 x PHNMR or 8 x PHNMR	Phasor estimates. May be single phase or 3-phase positive, negative, or zero sequence. Four or 8 bytes each depending on the fixed 16-bit or floating-point format used, as indicated by the FORMAT field in the configuration frame. The number of values is determined by the PHNMR field in configuration 1, 2, and 3 frames.
8	FREQ	2 or 4	Frequency (fixed or floating point).
9	DFREQ	2 or 4	Rate of Change of Frequency (fixed or floating point)
10	ANALOG	2 x ANNMR or 4 x ANNMR	Analog data, 2 or 4 bytes per value depending on fixed or floating-point format used, as indicated by the FORMAT field in configuration 1, 2, and3 frames. The number of values is determined by the ANNMR field in configuration 1, 2, and 3 frames.
11	DIGITAL	2 x DGNMR	Digital data, usually representing 16 digital status points (channels). The number of values is determined by the DGNMR field in configuration 1, 2, and 3 frames.
	Repeat 6-11	Variable	Fields 6–11 are repeated for as many PMUs as in NUM_PMU field in configuration frame.
12	CHK	2	CRC-CCITT

A binary (hexadecimal) comparison of the two packets emphasizes the size difference. In this comparison, it must be noted that the C37 frame does not contain latitude and longitude values, as no provision for their inclusion exists within the standard. Given a standard frame with:

Count = 4, UTC_Date = 071715, UTC_Time = 213451, V (magnitude) = 122.712268 V, θ (angle) = 0.929178 radians, and f (frequency) = 59.9939336, the hexadecimal equivalents are as follows:

FNET Frame:

0120393030310d2030373137313520323133343531202034203539 2e393933392039 2e393933392031322 2e3731323320302e3932393200

C37.118 Frame:

aa011c002923fc74a95504000000ae6cf5429dde6d3fcaf96f42155b

In order to accommodate the C37 data frame specification, the packet counter is used as the FRACSEC field. This is a valid conversion as the C37 TIME_BASE for the FDR is specified to be 10. Also unlike the FDR frames, the nature of the data does not change based on the packet counter. Latitude and Longitude measurements are never sent, nor is the amount of GPS satellites that the unit is connected to. The final major difference is the addition of the checksum for the packet, or cyclic redundancy check. The C37.118 standard specifies a 16-bit checksum utilizing the CRC-CCITT algorithm with a polynomial of 0x1021 in hexadecimal notation; $x^{16} + x^{15} + x^2 + 1$ written mathematically. To facilitate computation of this value, a Python library – CRC_C37.py – was developed to calculate this value for an arbitrary series of bytes. The constructC37Frame function inside *FDR_Comm.py* packs the frame data into a byte array and then uses the library to compute the checksum. This checksum is then appended onto the frame. Source code for this library is provided in Appendix A.

Due to the nature of the device, communication is one-way – the FDR only ever sends data to the servers specified in the *server_config* file. This file can potentially be changed remotely and the process can be restarted, but none of the programs in the measurement sequence receive packets from remote sources. The prototype device uses standard TCP/IP communication for the purposes of compatibility with the existing FNET server framework, as shown in Section 5.2. However, the design allows for encrypted communications to be used as well. If encryption is utilized, then the socket used is initially constructed as an encrypted connection. In the enumeration process, there is no substantial difference between a standard TCP/IP socket and an encrypted one. While the program can be altered to use any form of Transport Layer Security (TLS) [81], the prototype unit supports TLS 1.2 and SSL

encryption. This is accomplished with the use of standard libraries and protocols and is detailed in Section 4.5.

4.5. Encrypted Communication

While the current Generation-2 FDRs as well as the overall FNET system do not support any form of encrypted communication, providing the prototype Generation-3 FDR with a method of sending encrypted data allows for potential future increases in capability. The enumerated server communication made possible by the communication subprocess allows for standard FNET packets to be sent back to the main FNET servers and for encrypted packets to be sent to other locations. The purpose of this section is to explain the TLS communication methods supported by the unit and the resulting requirements on the backend if encrypted communication is to be employed.

The standard foundational model of network communications is the OSI (Open Systems Interconnect) Model. It characterizes and standardizes the communication functions of a system by dividing the overall structure into abstraction layers. Each layer interacts with the layers above and below itself. Figure 4-17 shows the standard OSI Model, with 7 abstraction layers.

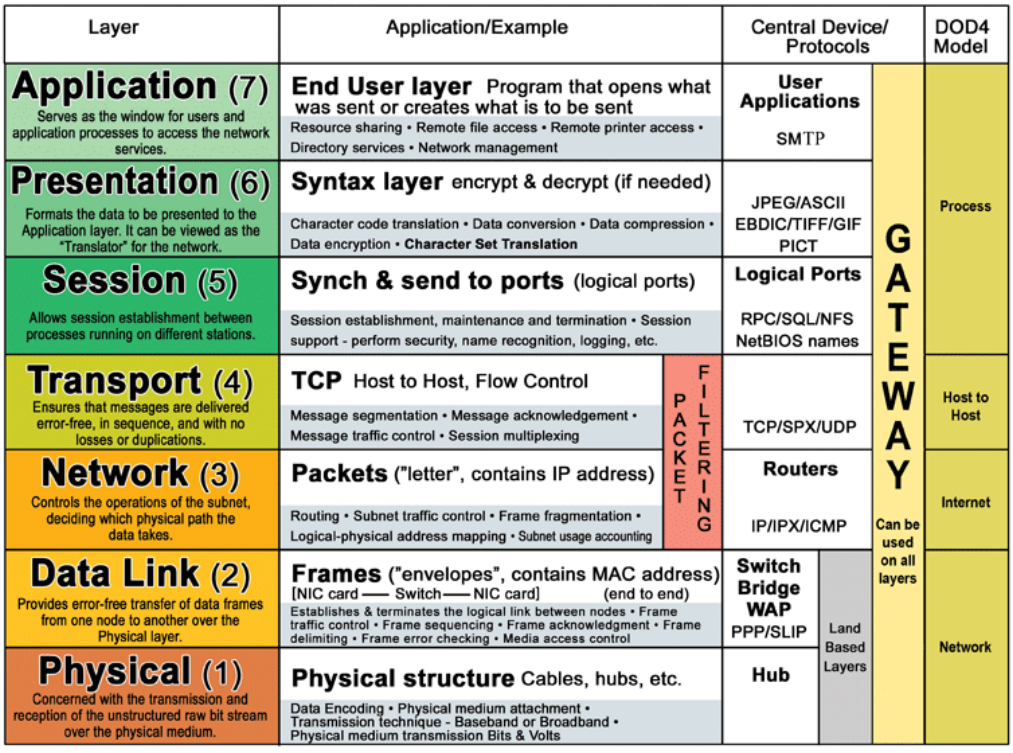


Figure 4-17 OSI (Open Systems Interconnection) 7 Layer Model

As this model is an abstraction, encrypted communication does not fit neatly into a single layer. Rather, Transport Layer Security is initialized at layer 5, such that the session is secure, but operates mainly in layer 6, the Presentation Layer. The reason for this multi-layer distinction is that TLS utilizes X.509 certificates and hence asymmetric (or public-key) cryptography for authentication. These certificates allow a third party to verify that each party is who they are claiming to be [82].While the underlying mathematics of asymmetric encryption differ with various implementations, the fundamental concepts rely on mathematical problems which currently admit no efficient solution [83]. The strength lies in the fact that key generation is computationally simple, and determining the validity of the key is also simple. Determining the key itself, however, is computationally infeasible. This initial authentication occurs at the Session layer, and the asymmetric cryptography is used to share a symmetric key used inside the session. This symmetric encryption occurs in the Presentation Layer. Figure 4-18 below is a classic illustration of how this shared information reaches each party without the symmetric key ever being sent across the open network.

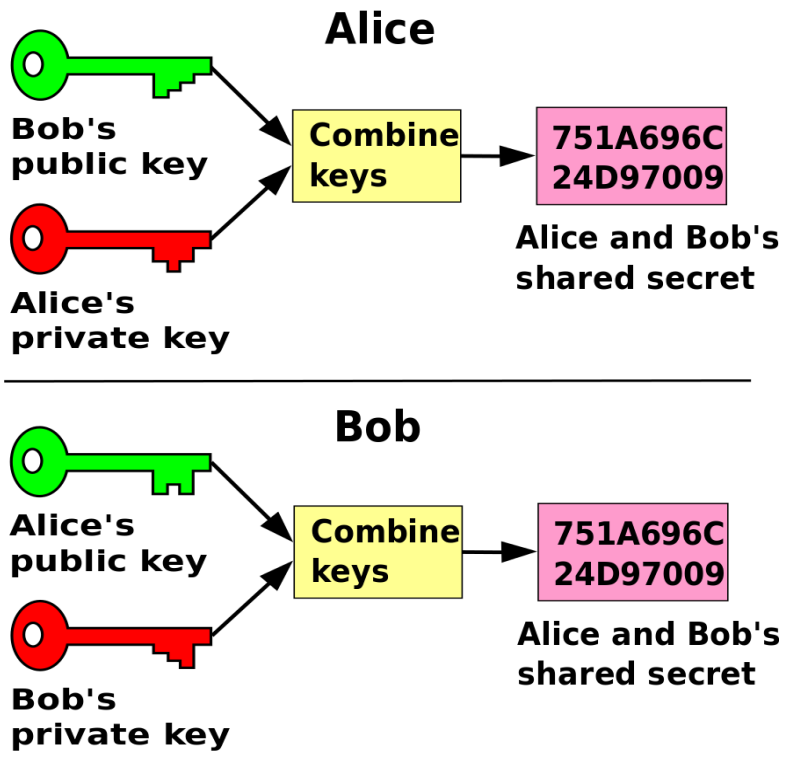


Figure 4-18 Asymmetric Cryptography Shared Secret Example

Once the symmetric encryption is established, data can flow freely between the participants with no further need of certificate authentication. The actual encryption used depends on the cipher settings negotiated during the asymmetric handshake. In the case of the Generation-3 FDR, the Beaglebone Black's processor has dedicated hardware which supports AES, SHA, and MD5. This allows the FDR to use ciphers which at the time of this writing, are considered computationally infeasible to defeat.

Making use of this security layer within the program is straightforward. As mentioned in Section 4.4.4, communication sockets are established based on the entries in the *server_config* file. In Python, a socket object contains all of the information about the connection and also serves as the interface for that connection. In Python, the "ssl" library exists to handle secure communications. The underlying framework used by Python is OpenSSL, a widely used TLS implementation [84]. By utilizing Python's "ssl" library, a standard socket (socket.socket) can be encapsulated as a secure socket (ssl.SSLSocket) [85], which is a subtype of the standard. The parent-child relationship between the classes allows *FDR_Comm* to use a single array to manage both types of connections. The function that encapsulates a socket object is ssl.wrap_socket, and its definition is as follows:

```
ssl.wrap_socket(sock, keyfile=None, certfile=None, server_side=False,
cert_reqs=CERT_NONE, ssl_version={see docs}, ca_certs=None,
do_handshake_on_connect=True, suppress_ragged_eofs=True, ciphers=None)
```

In the function definition above, the only required parameter is a socket object. All other parameters have default values. However, authentication parameters – certificates – are necessary for the asymmetric handshake to be secure. For many secure transactions over the Internet, these certificates are issued by trusted third parties known as Certificate Authorities (CAs) [86].

While TLS does not have a specific requirement for how a certificate is stored, the Python "ssl" library utilizes PEM files – a base 64 encoded file with a header and footer line. These files are part of the key management framework by the Internet Engineering Task Force (IETF) and go beyond the basic requirements specified by X.509 [87]. The benefit of doing this with the next generation FDRs is that the units can validate that the server is officially sanctioned and allowed to receive measurements, while the server can validate that only officially deployed units are sending measurements. Additionally, a certificate may be given an expiration date, at which point a new signature may be generated for the unit. While this is often considered a security "best practice," it is not strictly required and the prototype unit does not make use of a time-limited certificate or private key. A single PEM file can be used to store a certificate chain, which is a list of signatures which can be traced back to a root certificate. This chain

can be used to assist in determining if a given certificate is authentic. As the FDR is a specialized device and is only intended to communicate with specific servers, purchasing certificates from a CA is not necessary, and the devices can use self-signed certificates instead. This is no less secure cryptographically; the only resulting limitation is that the certificate will not be in the cache of a trusted authority. OpenSSL provides a method for generating both a private key and self-signed certificate. Figure 4-19 OpenSSL Key and Certificate Generation is an example of self-signed certificate generation. The private key and certificate are written to the same file, which is allowed. The cipher used is RSA, and the certificate is valid for 1 year.

```
% openssl req -new -x509 -days 365 -nodes -out cert.pem -keyout cert.pem
Generating a 1024 bit RSA private key
.......++++++
..............................++++++
writing new private key to 'cert.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:MyState
Locality Name (eg, city) []:Some City
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Organization, Inc.
Organizational Unit Name (eg, section) []:My Group
Common Name (eg, YOUR name) []:myserver.mygroup.myorganization.com
Email Address []:ops@myserver.mygroup.myorganization.com
```

Figure 4-19 OpenSSL Key and Certificate Generation

Using this generation feature, the prototype unit possess a public and private keypair, as well as a unique certificate identifying it as a Generation-3 FDR. The current FNET system does not support encrypted communication, so a test server program was created in order to verify proper functionality. The testing procedure and results are elaborated upon in Section 5.3. The PEM files for both the prototype Generation-3 FDR and the TLS test server are provided as external attached files.

5. Unit Testing and Validation

This chapter deals with the testing and validation of the prototype Generation-3 FDR unit. Each section deals with a specific facet of the unit. Of primary interest is the signal quality – determination of how much noise is in the measured data and whether or not the calculated frequency and voltage levels agree with that of existing units. Following those tests, full integration of the prototype unit into the FNET system is performed. Lastly, a simple test case using TLS encryption capabilities and a custom server program determines the feasibility of using encrypted transmission for the FNET servers. For each of these tests, the through-hole PCB prototype version as described in Section 3.4 is used.

5.1. Signal Quality Testing

For signal quality, there are two major benchmarks – comparison with existing FDR units, and compliance with the IEEE C37.118 standard. As the units are not 3-phase devices, full compliance is not entirely possible nor applicable. However, the metrics described in the standard are sound methods of determining the performance of the device.

In order to establish a baseline for signal quality, a highly accurate signal source must be used to serve as the reference. For the procedure described, the source used is the Doble F6150 – a power system simulator, often utilized for the testing of protection relays and schemes. This source allows for precise control of voltage magnitude and frequency. On-board GPS synchronization is used to ensure a high degree of accuracy in the output signal. Figure 5-1 is a reference picture of a Doble F6150.

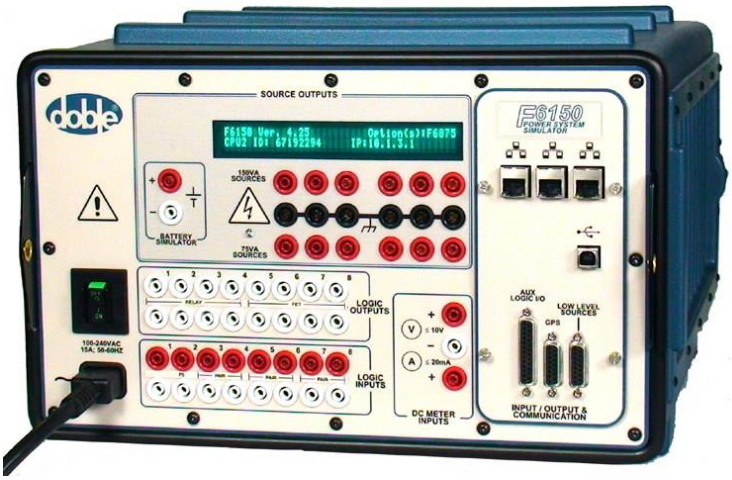


Figure 5-1 Doble F6150 Power System Simulator

The primary concern for device performance at the hardware level is ensuring that the signal is being captured accurately with minimal distortion introduced by the filtering circuits or ADC. Electrical and thermal noises are the major concerns with regards to physical distortion. While these effects can be mitigated through proper board design and electrical isolation, they cannot be entirely removed. A small amount of thermal noise is always present across any resistance, as given by the equation $v_n = \sqrt{4k_B TR\Delta f}$ where k_B is Boltzmann's constant (Joules per Kelvin), T is the resistor's absolute temperature, R is the resistance in ohms, and Δf is the frequency bandwidth. As a general rule, this means that cooler temperatures are better, but the amount of uncertainty present in the 1% tolerance resistors used in the device far exceeds this value. For testing purposes, the best practice is simply to maintain a consistent temperature in the room and allow the device a warmup period before measurements are taken. As the sample rate is 3600 Hz, any frequency in the expected operational range should be accurately reproduced. Figure 5-2 shows a 6-cycle raw ADC waveform capture at 60 Hz and 120 VAC.

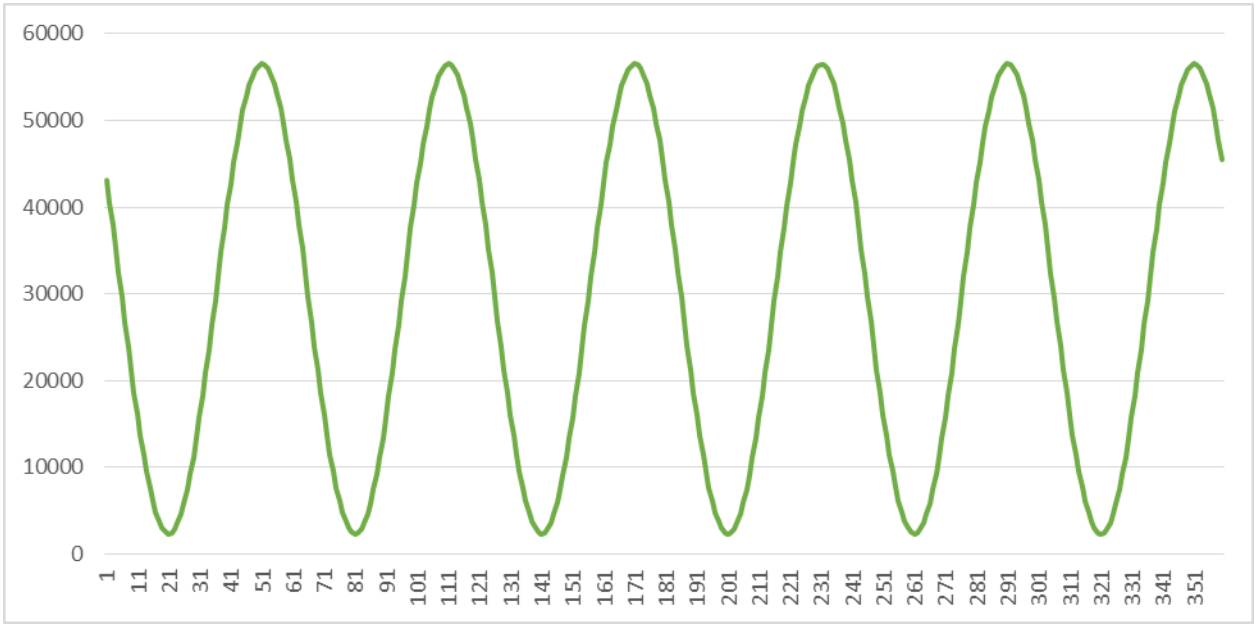


Figure 5-2 ADC Input Capture, 6 Cycles, 60 Hz

Upon visual inspection, the wave appears to be approximately 60 Hz and there is no clipping present in the capture. A periodogram of the Power Spectral Density (PSD) provides more insight into

which harmonics are present and in what quantities. Figure 5-3 is such a periodogram, using 5 seconds, or 300 cycles, of raw ADC samples.

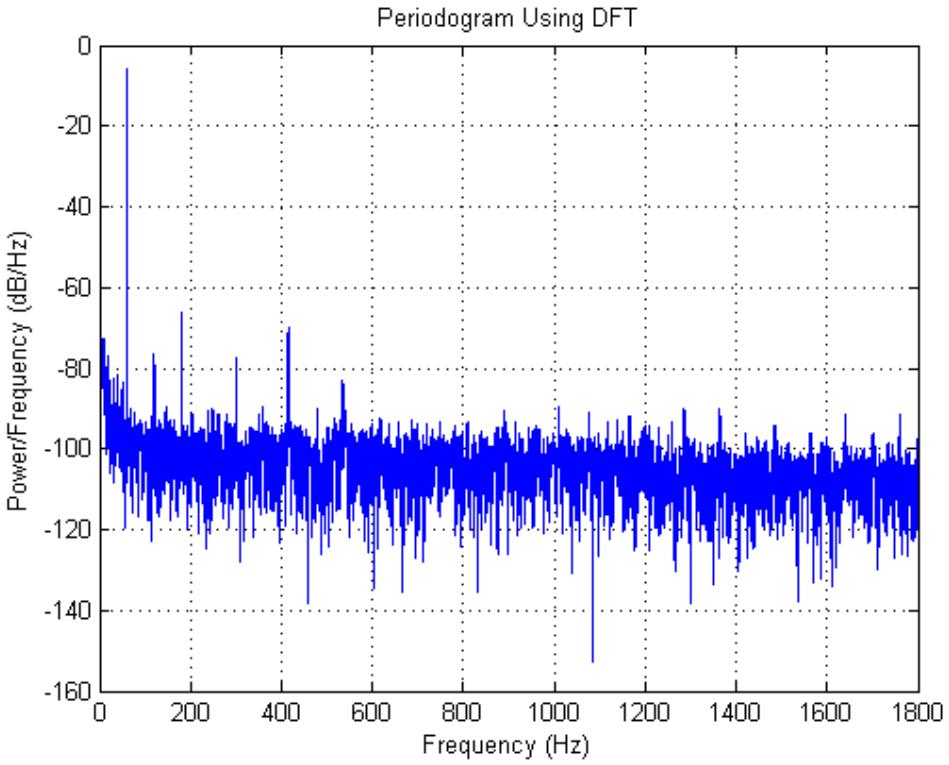


Figure 5-3 Power Spectral Density of 60 Hz ADC Capture

As expected, spikes are present at the harmonic values of the nominal frequency (60 Hz) – the 2nd through 9th harmonics can all be observed. Above that, the higher order harmonics are indistinguishable from the noise floor. Even the most prominent off-nominal harmonic is at -67 dB signal strength, so the overall noise level is quite minimal. The slow decreasing dB trend at the noise floor is due to the low pass filter's 20 dB/decade attenuation.

While the raw ADC characteristics can provide insight into the approximate performance of the prototype unit, the synchrophasor quantities are ultimately what must be calculated by the device. As such, a direct comparison of the frequency measurements can be used as a benchmark [88]. For this test, a Generation-2 FDR and the prototype unit were connected with the Doble F6150 and a 60 Hz, 120 VAC signal was generated. Each unit acquired a GPS signal and then began streaming data back to openPDC. A plot of the frequency data from each unit is presented in Figure 5-4 and a summary of the test results is provided by Table 5-1.

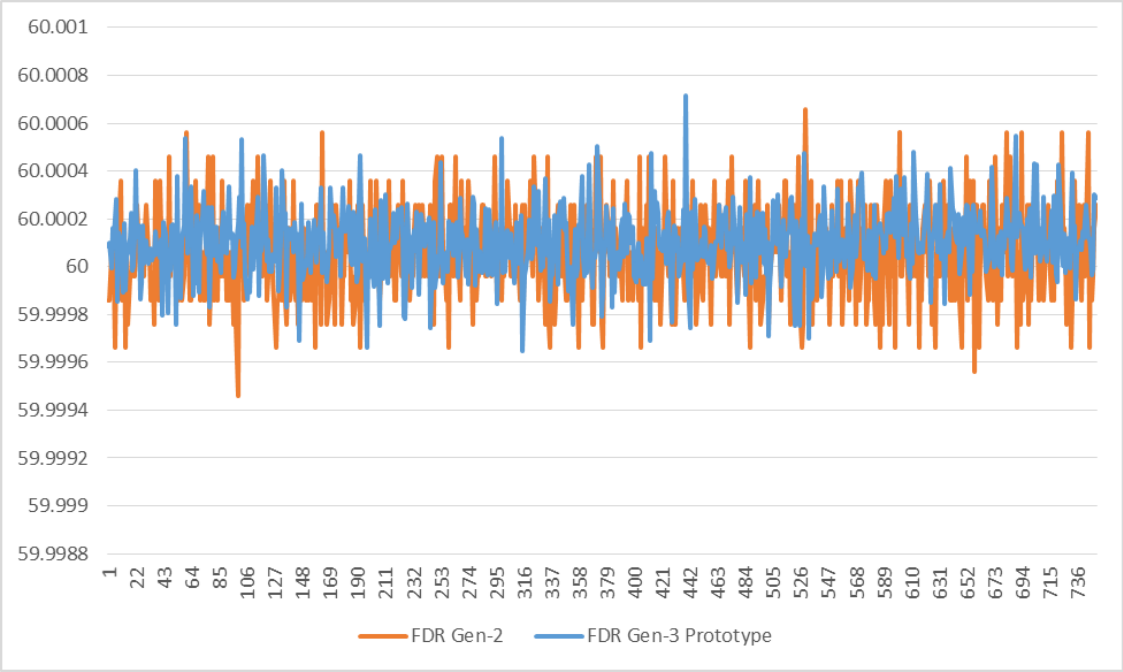


Figure 5-4 Generation-2 and Generation-3 FDR Frequency Comparison

On a high level, the two units seem to perform approximately equally well. The Doble's precision is 0.001 Hz, with an uncertainty of up to 10 ppm. Therefore, the only conclusion upon inspection of the measurement window is that both units measure the frequency to be within a very small range at 60 Hz and that some amount of signal noise is present in each device. While this is a simple comparison, it does show that the measurements provided by the prototype unit are not drastically different from those provided by the existing Generation-2 FDRs.

Table 5-1 Generation-2 and Generation-3 FDR Frequency Comparison Summary

Unit	Mean	Median	Variance	Standard Deviation	Total Range
Gen-3 Prototype	60.00010026	60.0000985	1.94618E-08	0.000139506	0.001069
Gen-2 FDR	60.00005708	60.00006082	3.64426E-08	0.000190899	0.00120163

Statistically, the units perform approximately equally in a nominal frequency test. The variance, standard deviation, and total range of the prototype unit are slightly lower than that of the existing Generation-2 FDR, but the differences are too small as to be meaningful. Mean and median values are

also approximately equal, with both units calculating the average frequency to be within 0.0002 Hz of the nominal value and 0.00005 Hz of one another.

While the nominal frequency test shows that the two units have similar signal characteristics, the C37.118 standard has more rigorous testing procedures to determine the capability of a unit. The three metrics as specified by the most recent version of the standard are Total Vector Error (TVE), Frequency Error (FE), and Rate of Change of Frequency Error (RFE). The equation for Total Vector Error is as follows:

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

Where X'_r and X'_I are the estimated real and imaginary samples from the unit under test, and X_r and X_i are the ideal values at the instant of time *n* from the unit's frame of reference. Put simply, the TVE is the magnitude of the difference of the measured and expected values. The Frequency and Rate of Change of Frequency Errors are calculated in similar fashion:

$$FE = |f_{true} - f_{measured}|$$

$$RFE = |(df/dt)_{true} - (df/dt)_{measured}|$$

Additionally, acceptable input ranges are specified for voltage, current, and frequency. The FDR differs from traditional units under test in that the input also serves as the power source, so a voltage far outside the normal range – 10% of nominal, for example, is infeasible to test. With this in mind, the test procedure must be established so as to be reasonable with respect to the limitations of the prototype FDR. As such, the derivation of C37.118 compliance testing is as follows:

- Frequencies within +/- 0.5 Hz from nominal (60 Hz) are tested.
- Frequency increment is 0.05 Hz, for a total of 21 tests.
- Each test is performed in the C37.118 compliant ambient conditions (23 +/- 3 °C).
- Individual tests last 2 minutes (120 seconds) once the unit has acquired GPS lock.
- Frequency, Voltage Magnitude, and Phase Angle results are saved from each test.
- TVE and FE are calculated for each test; RFE is not applicable for steady-state tests.

Figure 5-5 through Figure 5-67 on the following pages present the data from this testing, and Table 5-2 summarizes the results.

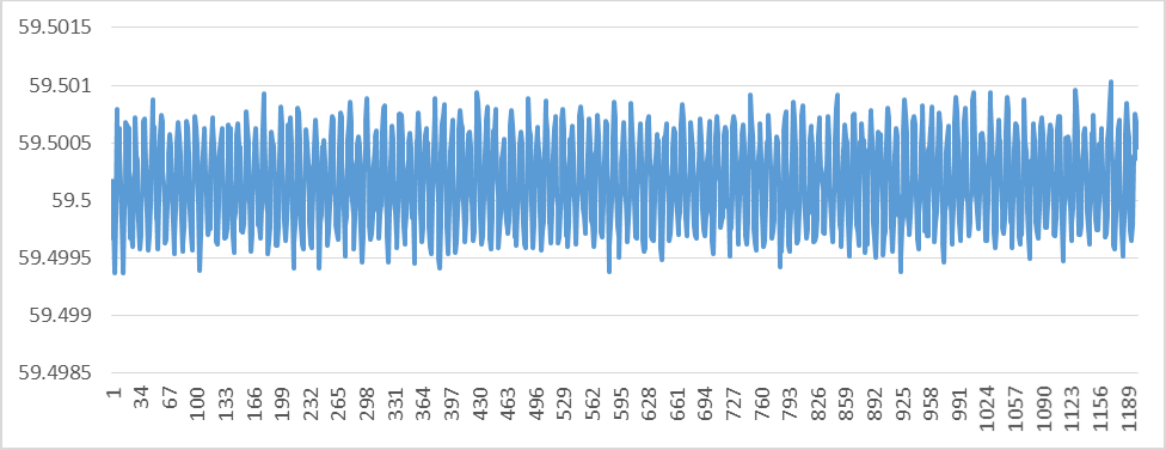


Figure 5-5 Calculated Frequency; 59.5 Hz

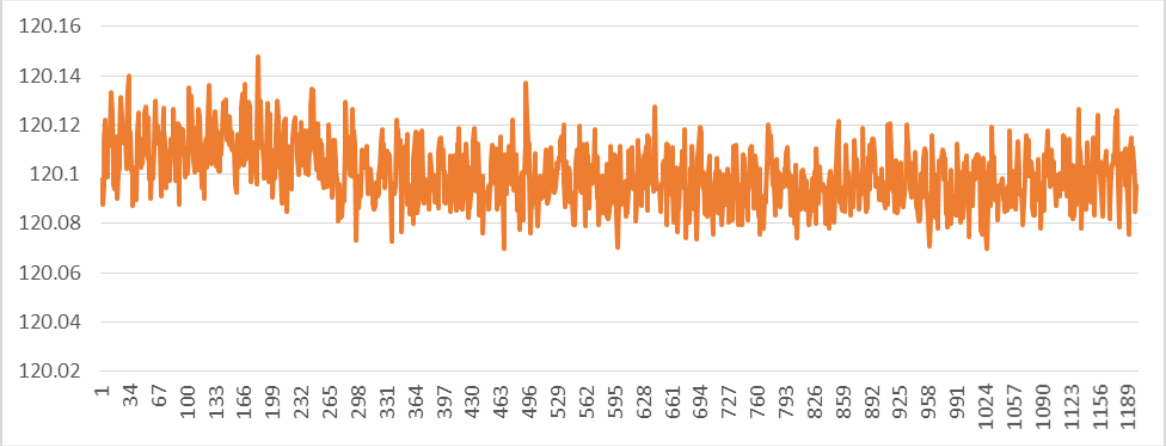


Figure 5-6 Voltage Magnitude; 59.5 Hz

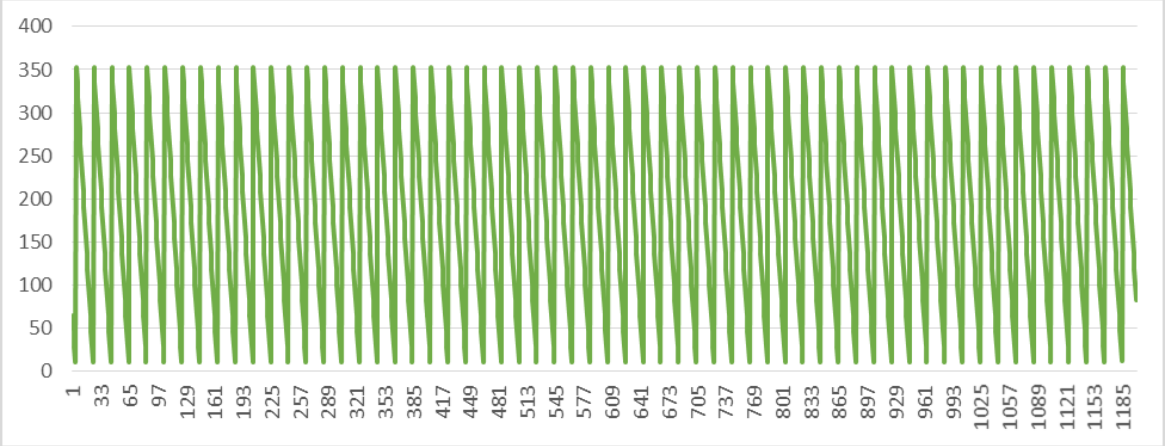


Figure 5-7 Phase Angle; 59.5 Hz

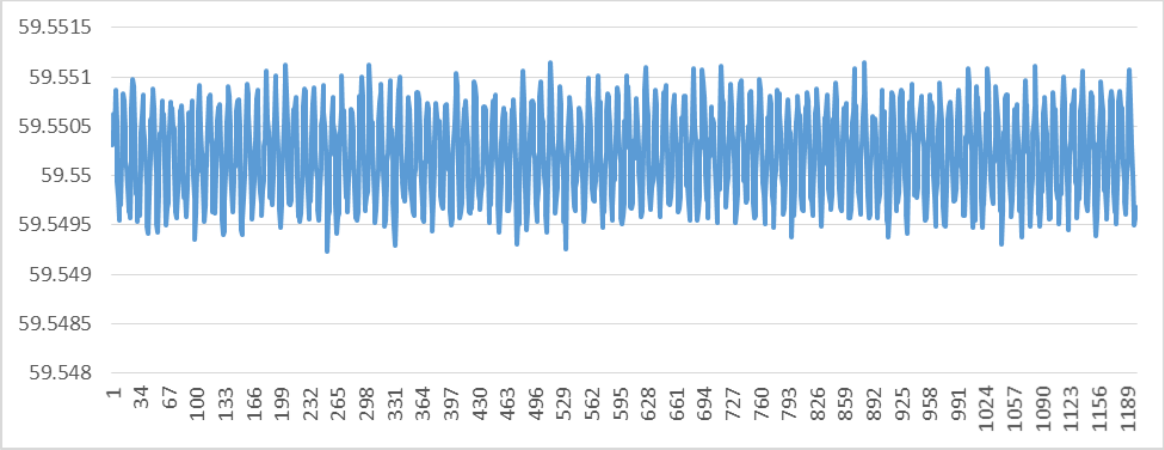


Figure 5-8 Calculated Frequency; 59.55 Hz

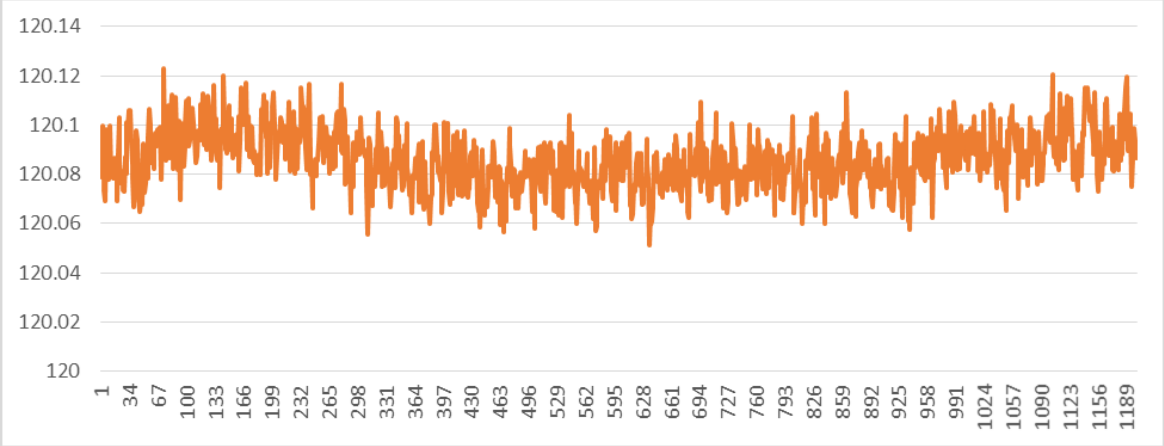


Figure 5-9 Voltage Magnitude; 59.55 Hz



Figure 5-10 Phase Angle; 59.55 Hz

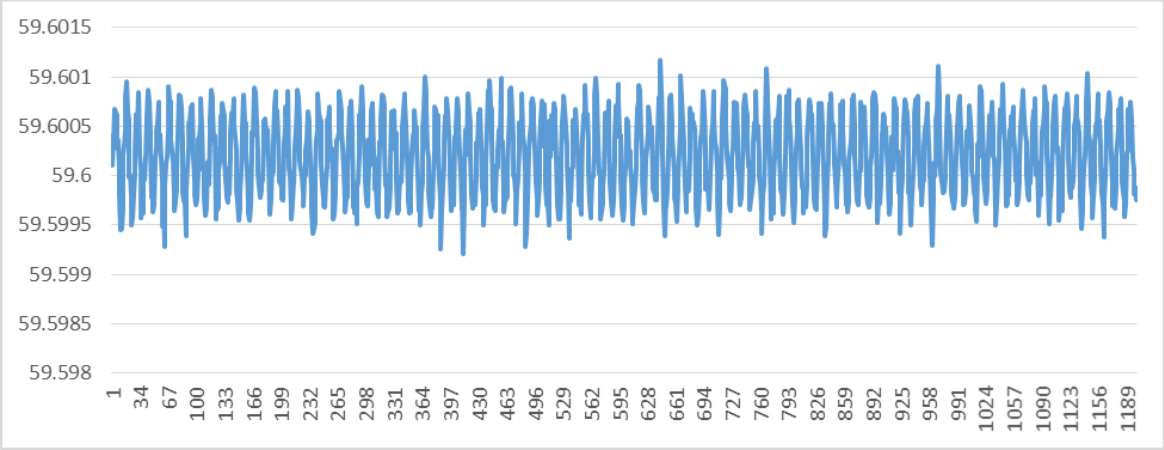


Figure 5-11 Calculated Frequency; 59.6 Hz

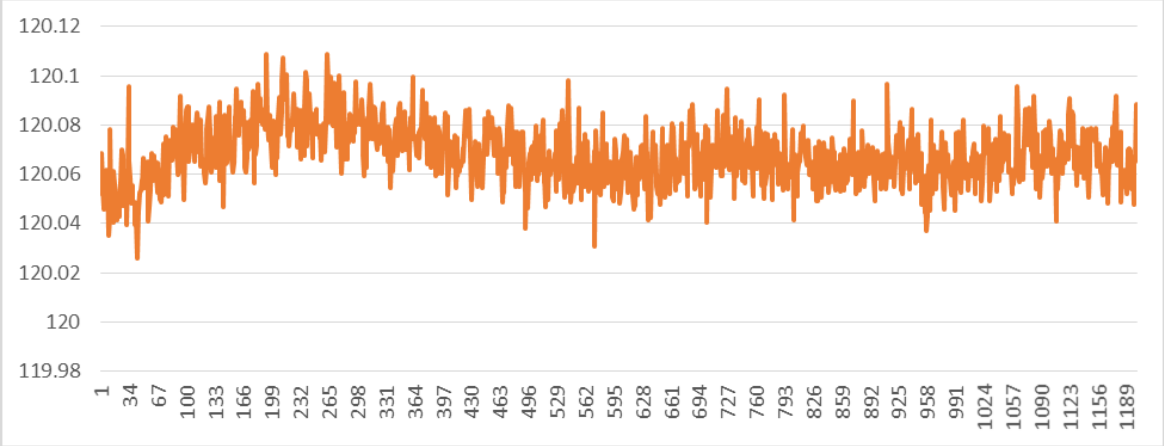


Figure 5-12 Voltage Magnitude; 59.6 Hz

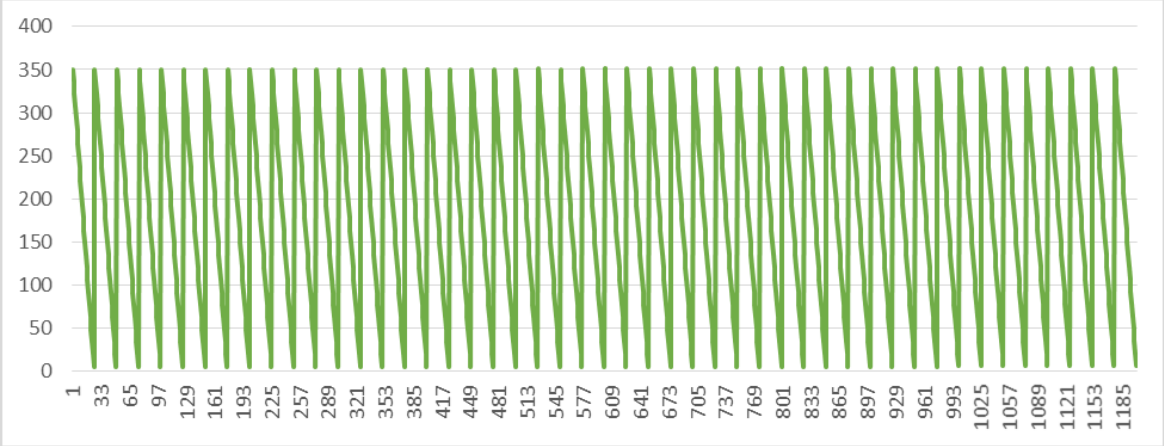


Figure 5-13 Phase Angle; 59.6 Hz

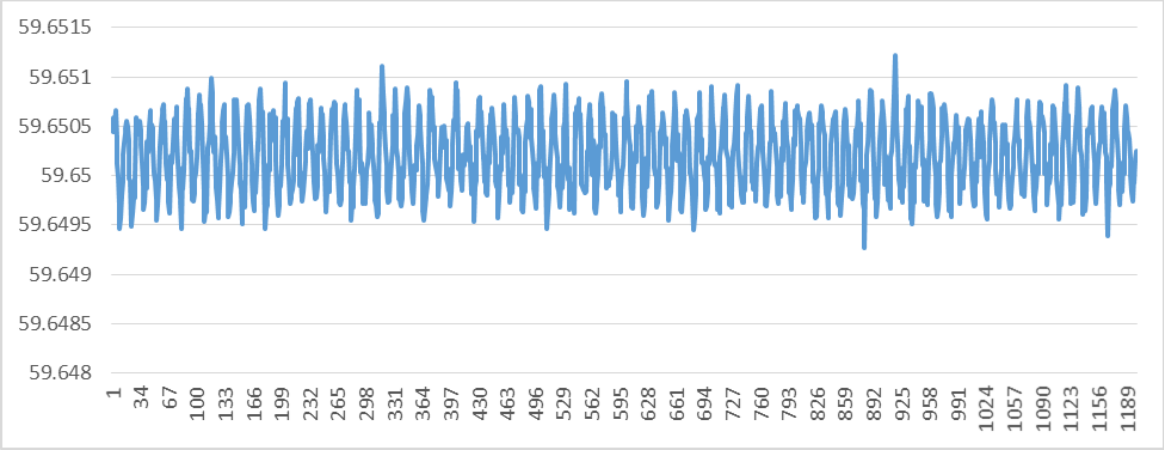


Figure 5-14 Calculated Frequency; 59.65 Hz

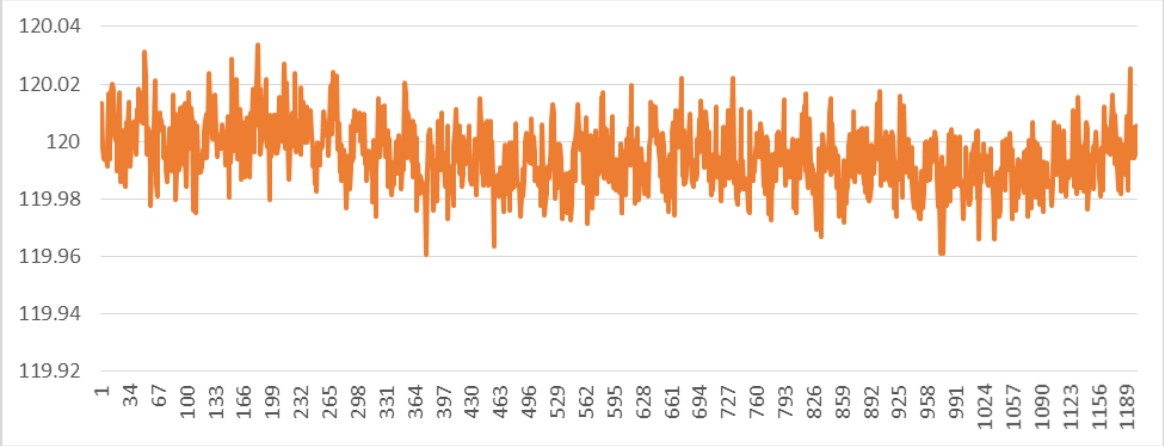


Figure 5-15 Voltage Magnitude; 59.65 Hz

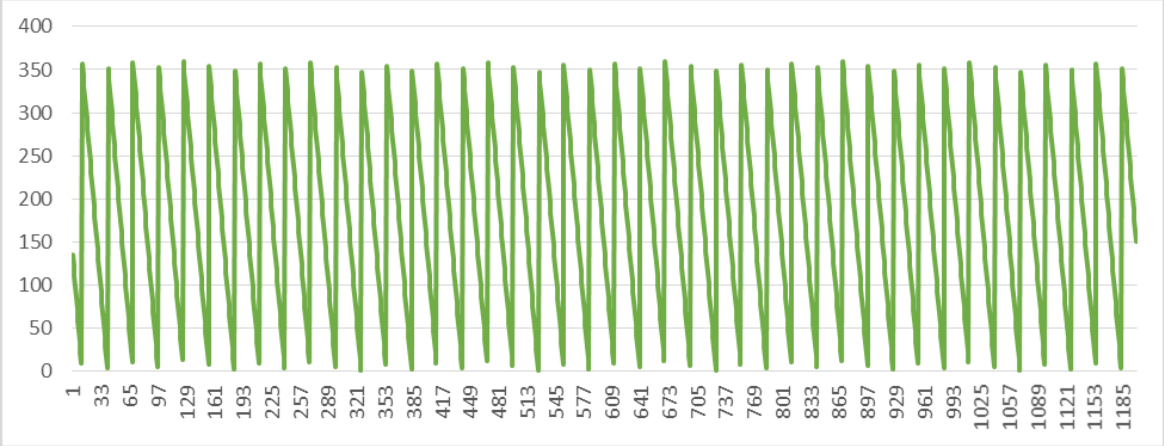


Figure 5-16 Phase Angle; 59.65 Hz

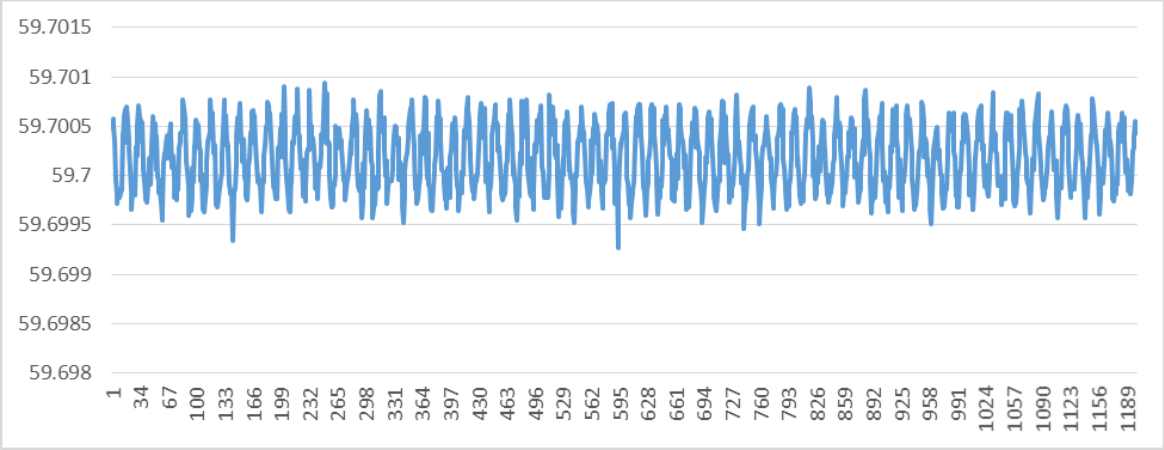


Figure 5-17 Calculated Frequency; 59.7 Hz

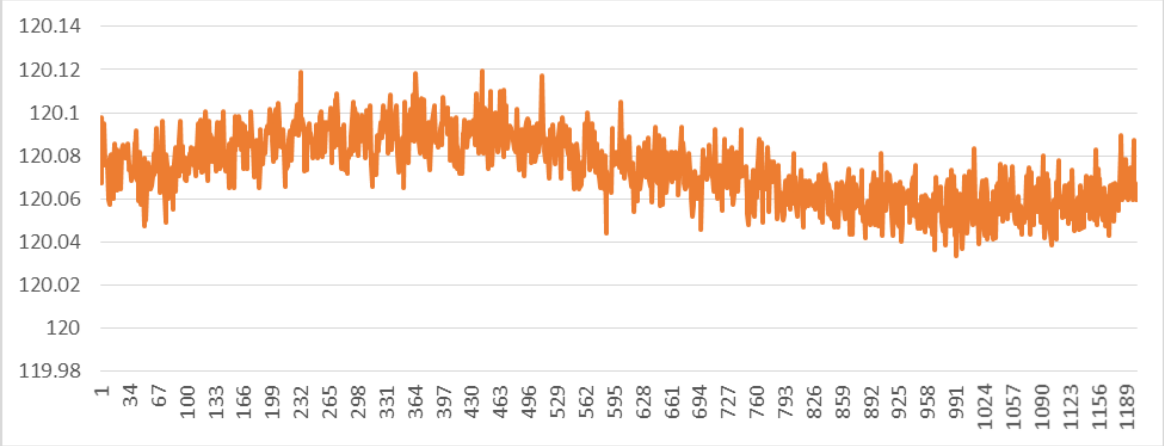


Figure 5-18 Voltage Magnitude; 59.7 Hz

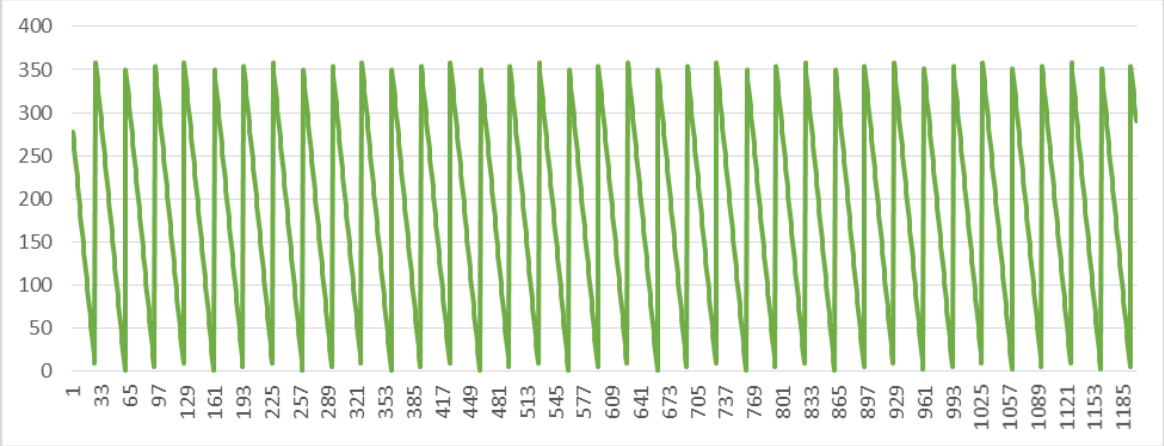


Figure 5-19 Phase Angle; 59.7 Hz

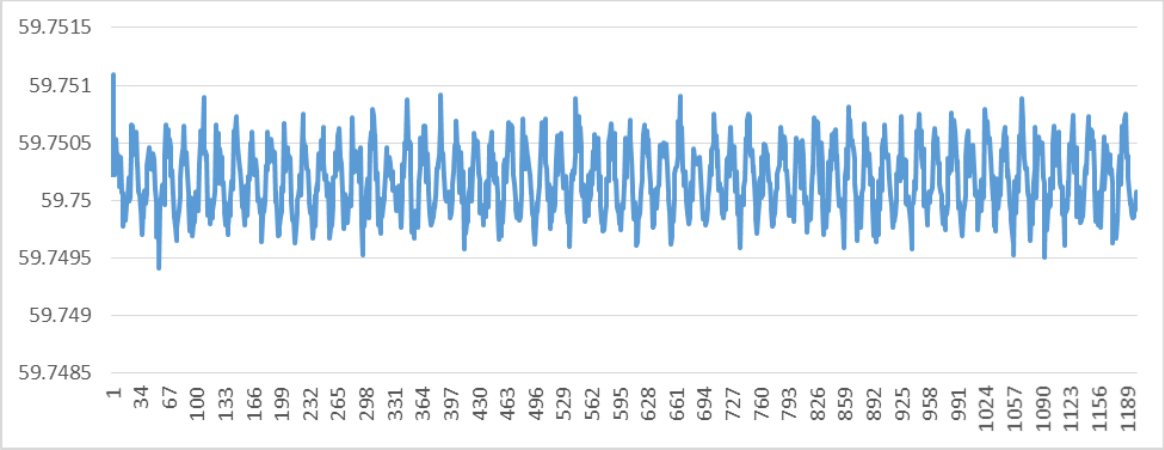


Figure 5-20 Calculated Frequency; 59.75 Hz

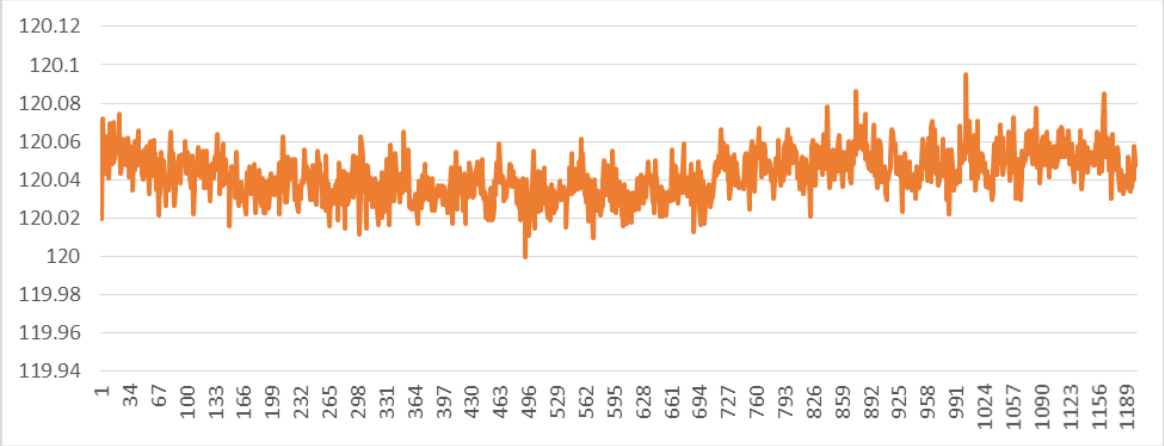


Figure 5-21 Voltage Magnitude; 59.75 Hz

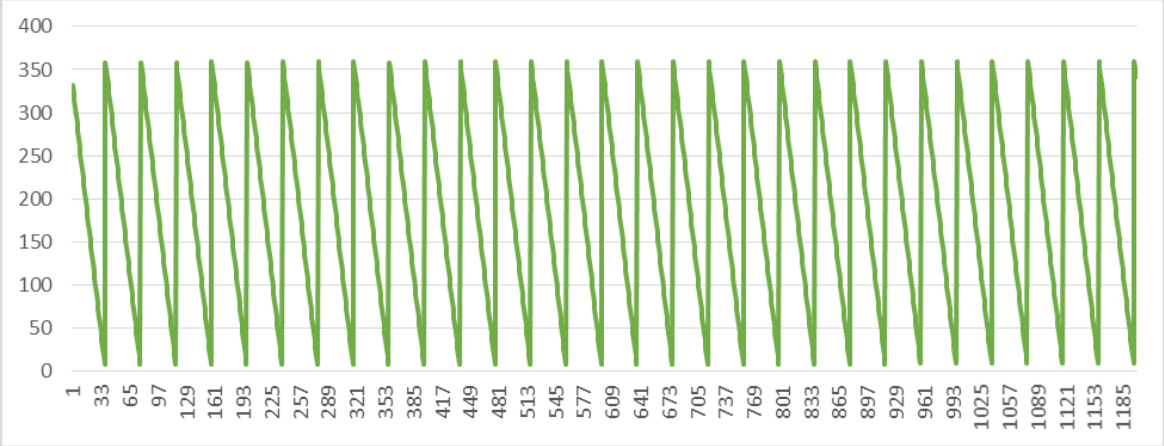


Figure 5-22 Phase Angle; 59.75 Hz

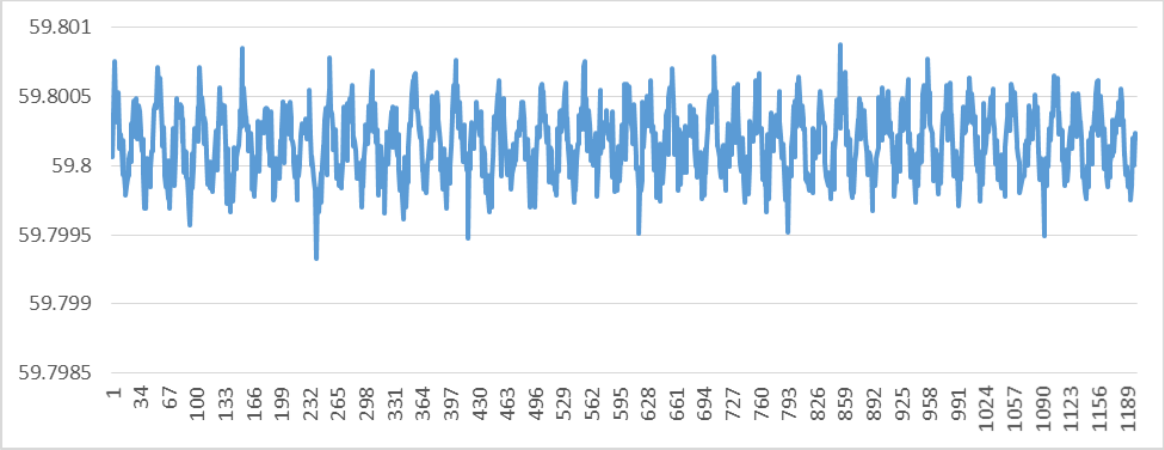


Figure 5-23 Calculated Frequency; 59.8 Hz

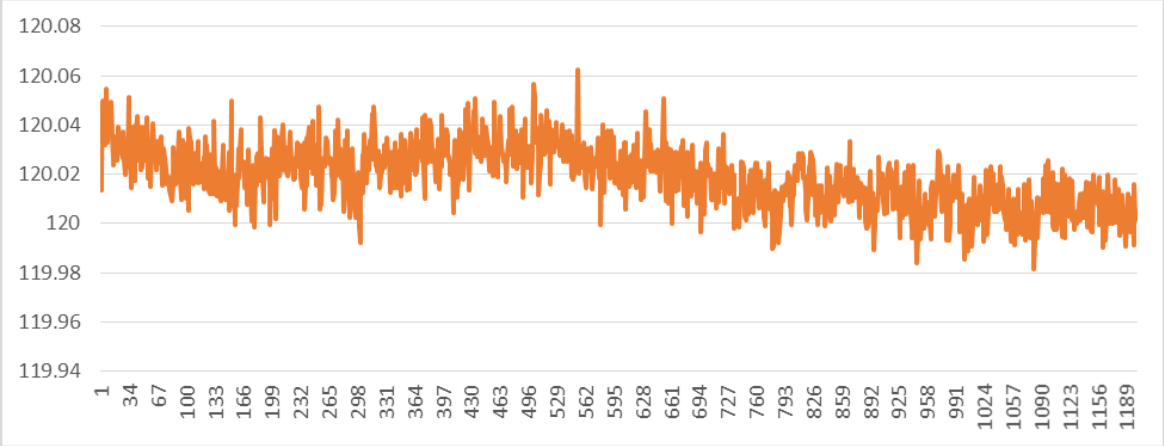


Figure 5-24 Voltage Magnitude; 59.8 Hz

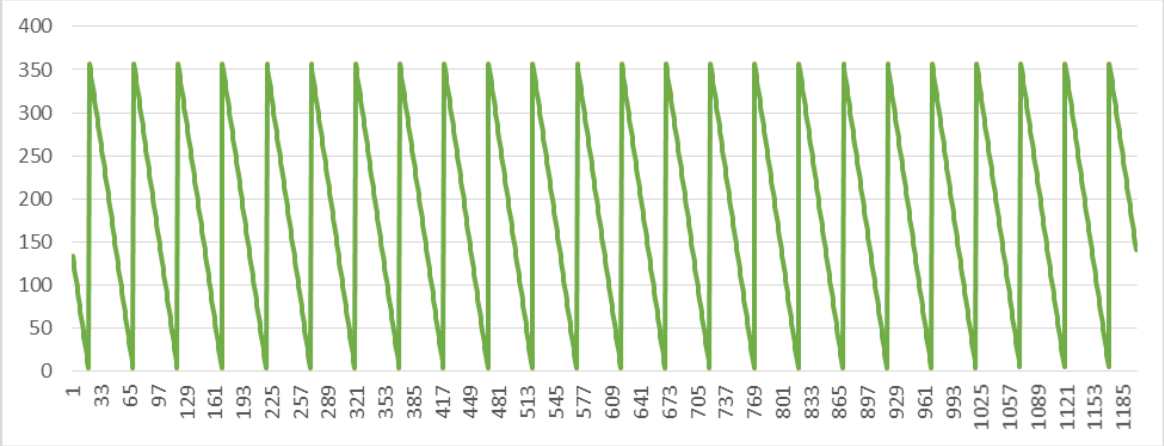


Figure 5-25 Phase Angle; 59.8 Hz

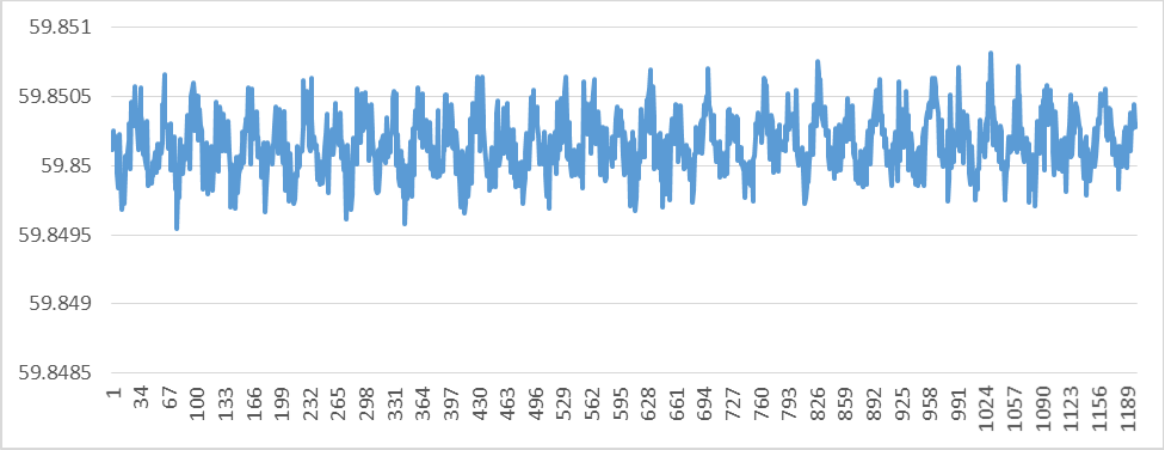


Figure 5-26 Calculated Frequency; 59.85 Hz

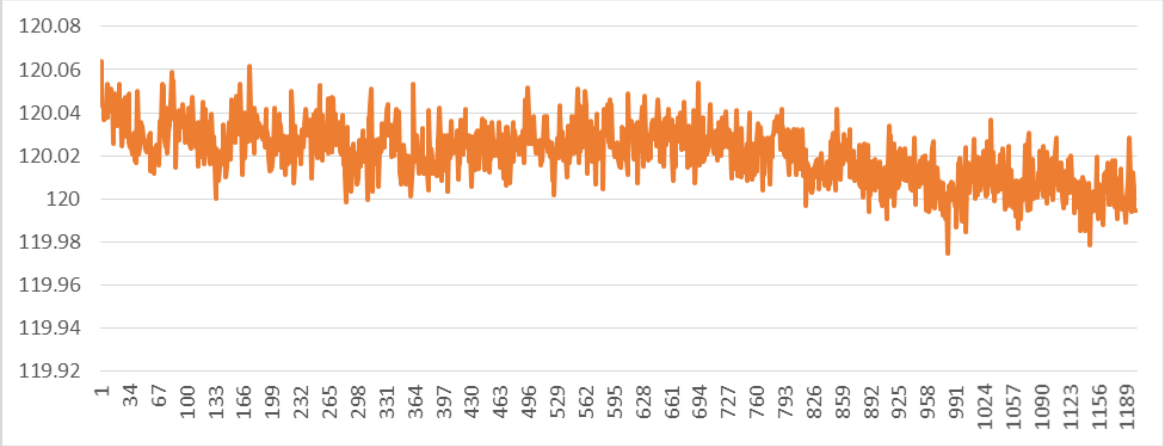


Figure 5-27 Voltage Magnitude; 59.85 Hz

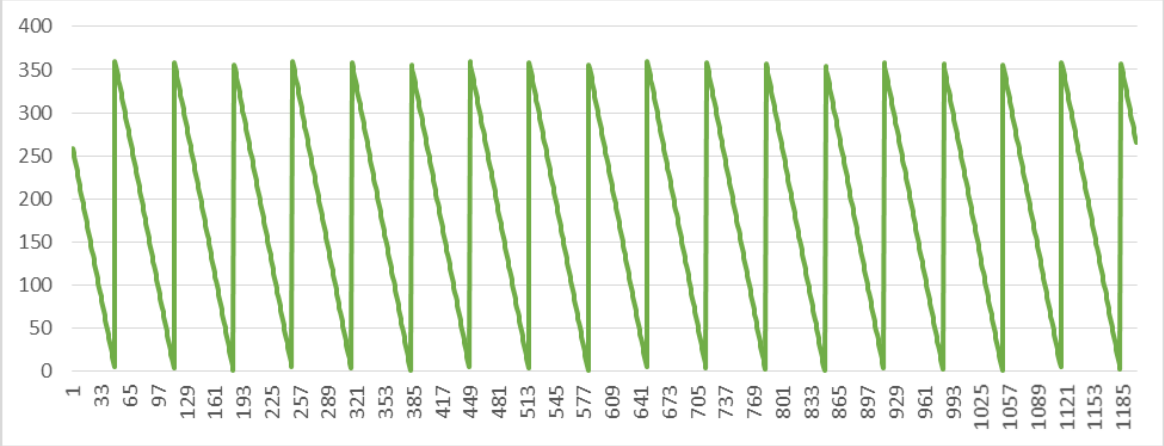


Figure 5-28 Phase Angle; 59.85 Hz

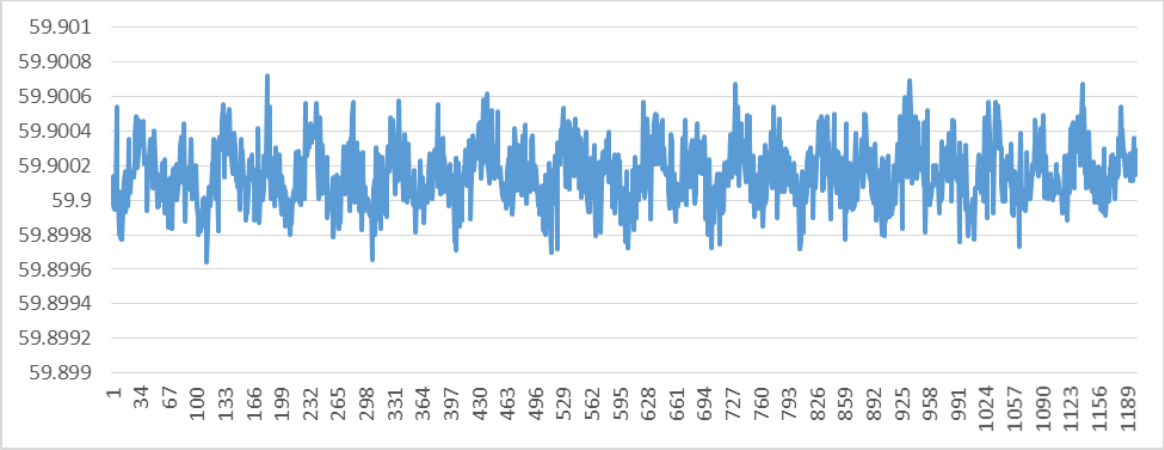


Figure 5-29 Calculated Frequency; 59.9 Hz

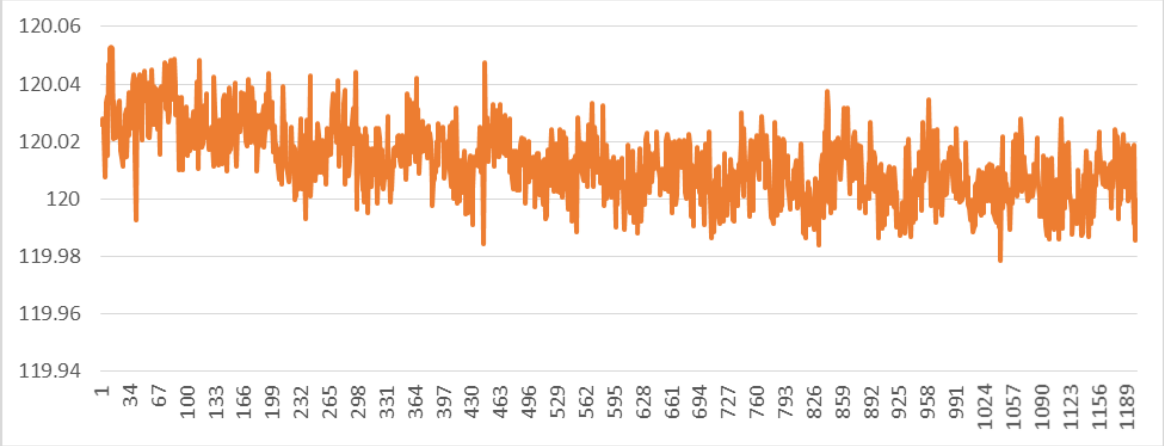


Figure 5-30 Voltage Magnitude; 59.9 Hz

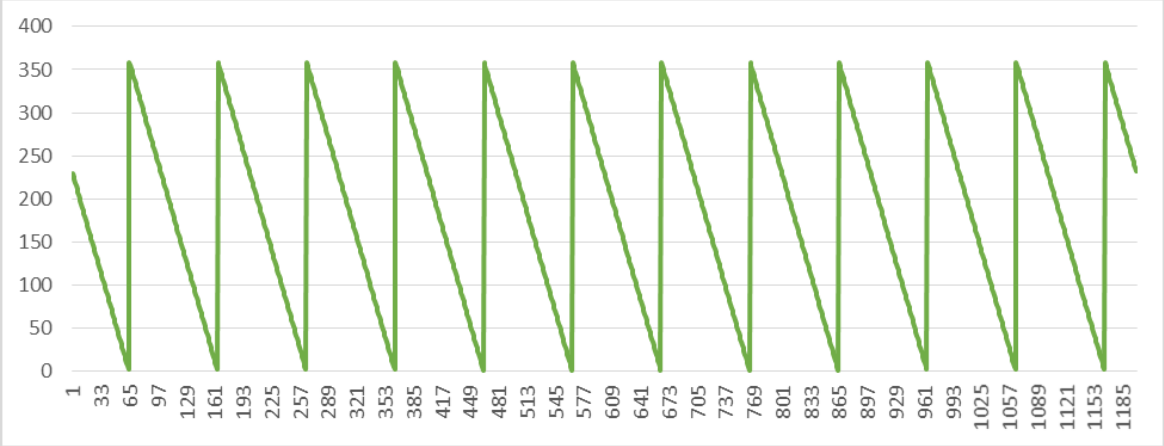


Figure 5-31 Phase Angle; 59.9 Hz

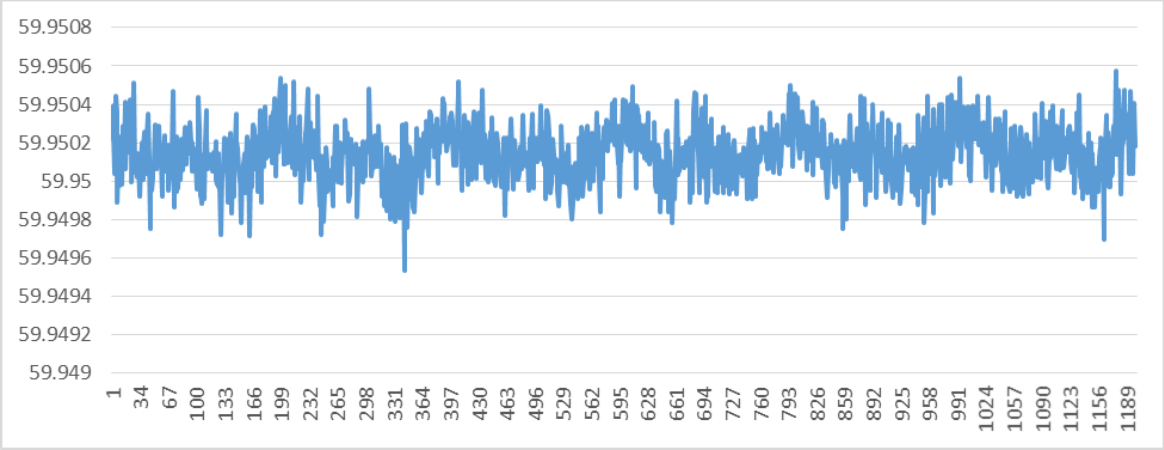


Figure 5-32 Calculated Frequency; 59.95 Hz

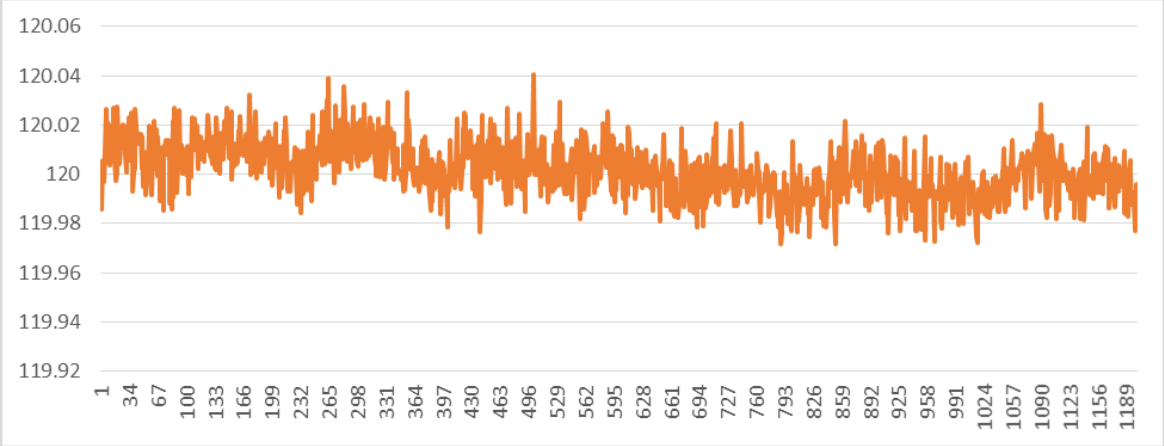


Figure 5-33 Voltage Magnitude; 59.95 Hz

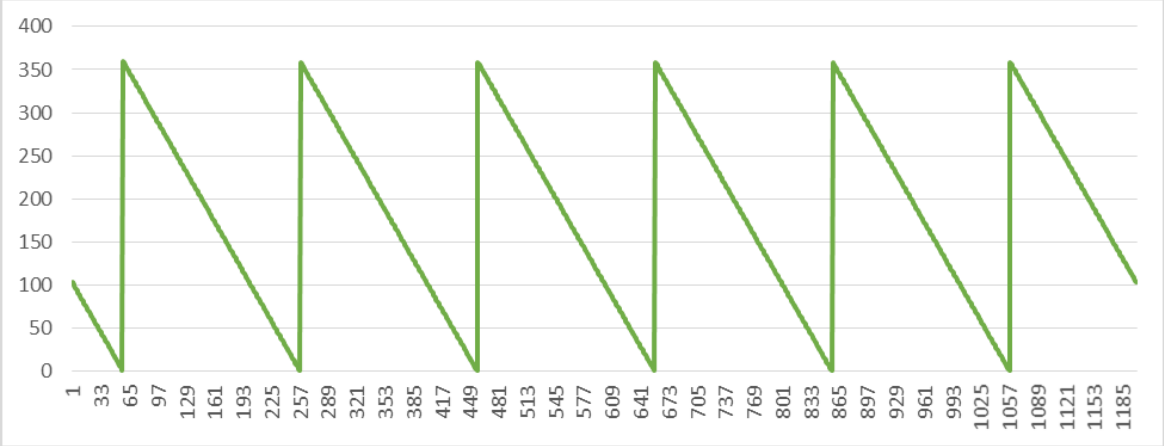


Figure 5-34 Phase Angle; 59.95 Hz

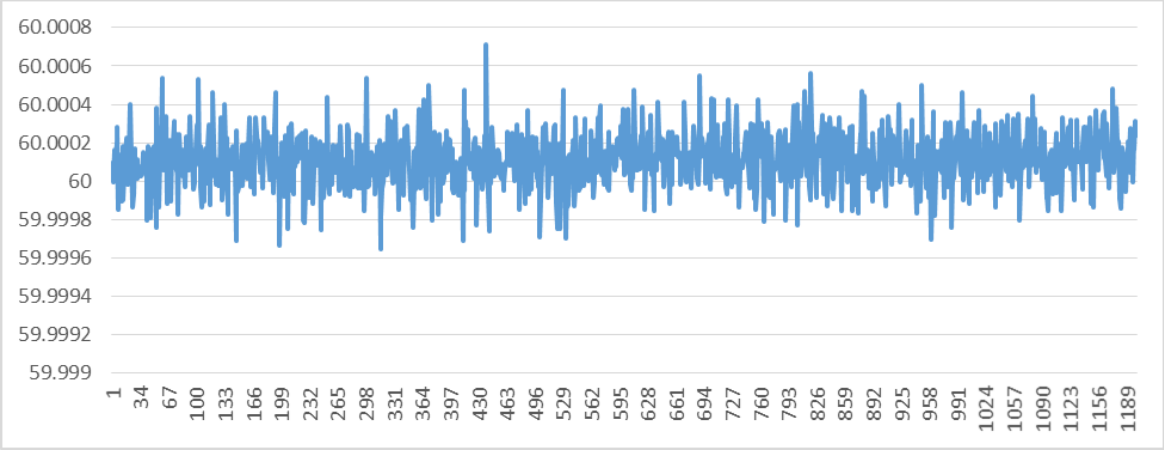


Figure 5-35 Calculated Frequency; 60.0 Hz

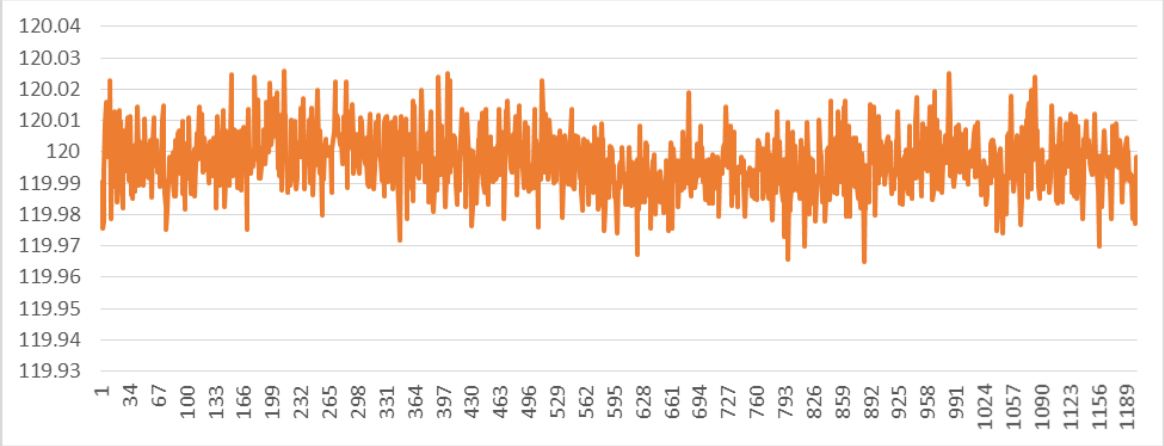


Figure 5-36 Voltage Magnitude; 60.0 Hz

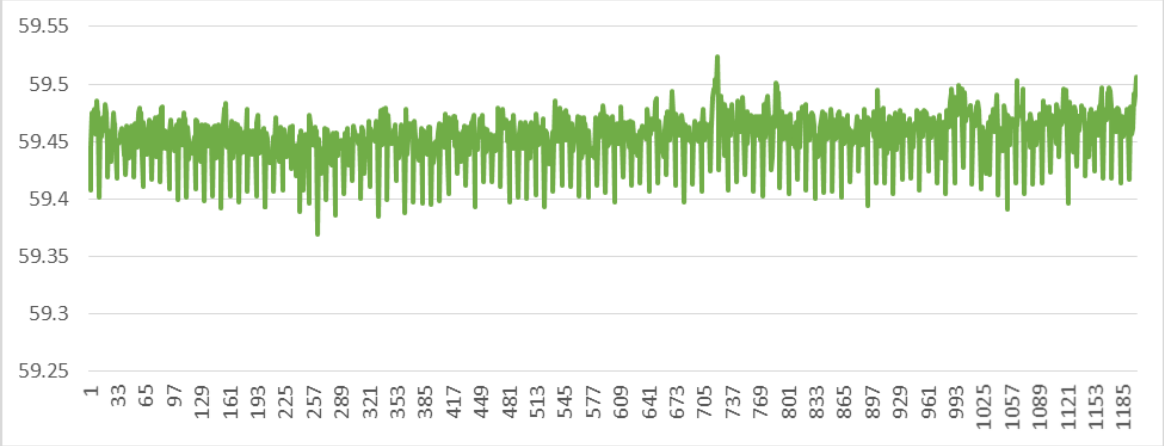


Figure 5-37 Phase Angle; 60.0 Hz

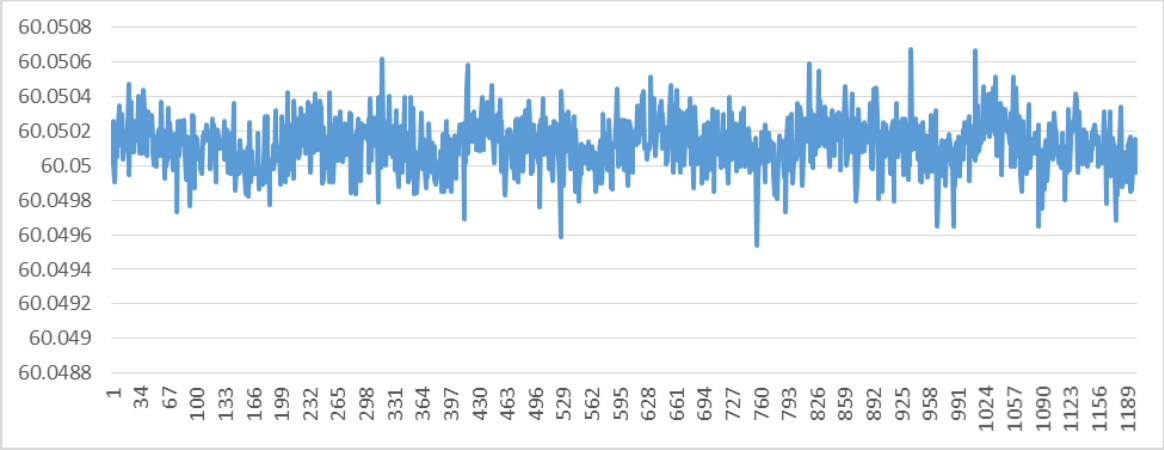


Figure 5-38 Calculated Frequency; 60.05 Hz

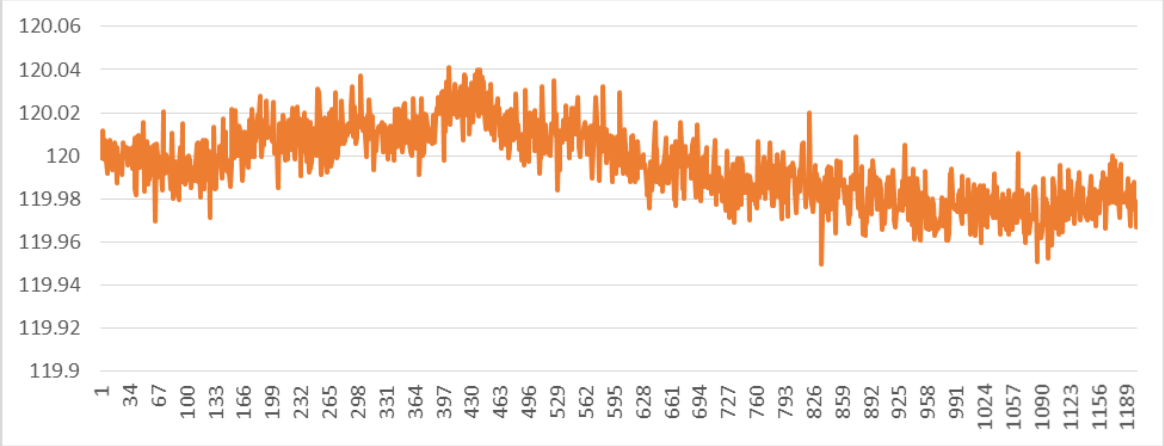


Figure 5-39 Voltage Magnitude; 60.05 Hz

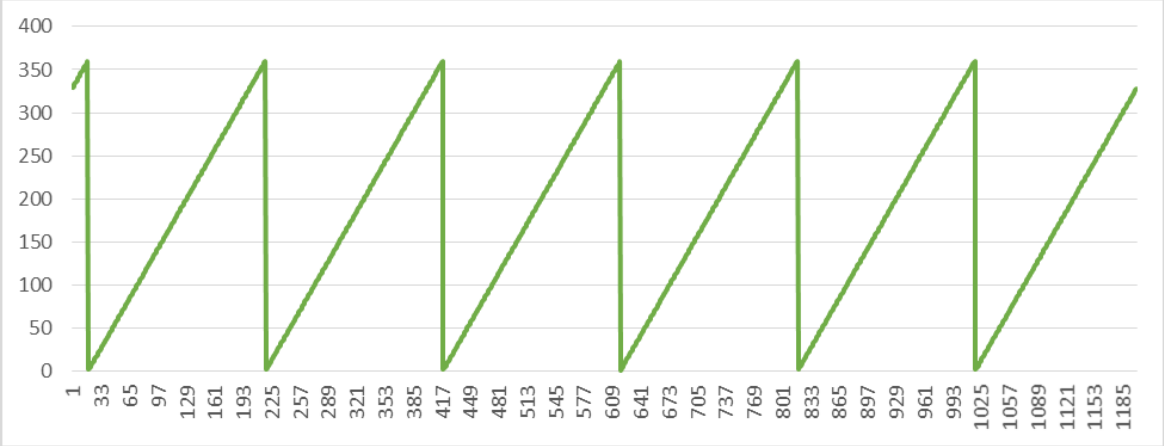


Figure 5-40 Phase Angle; 60.05 Hz

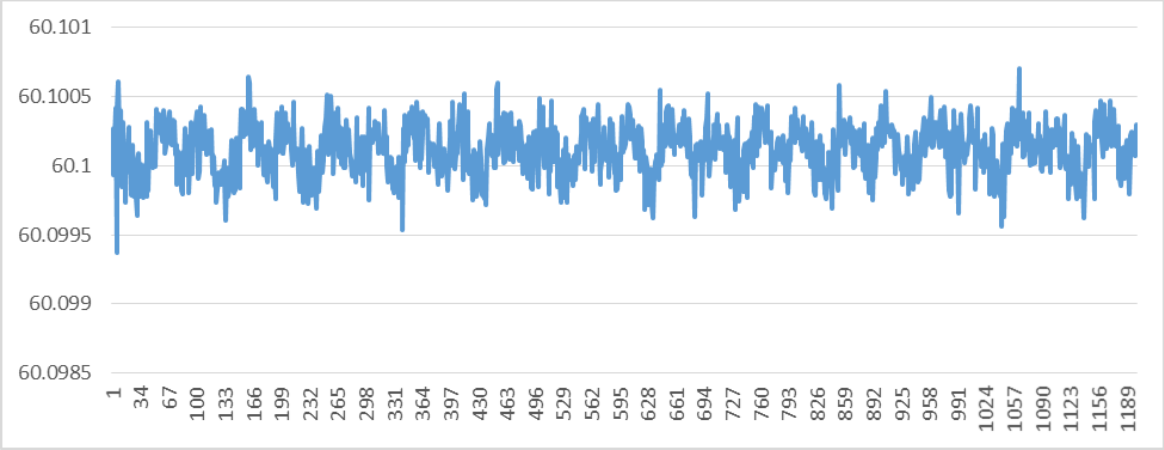


Figure 5-41 Calculated Frequency; 60.1 Hz

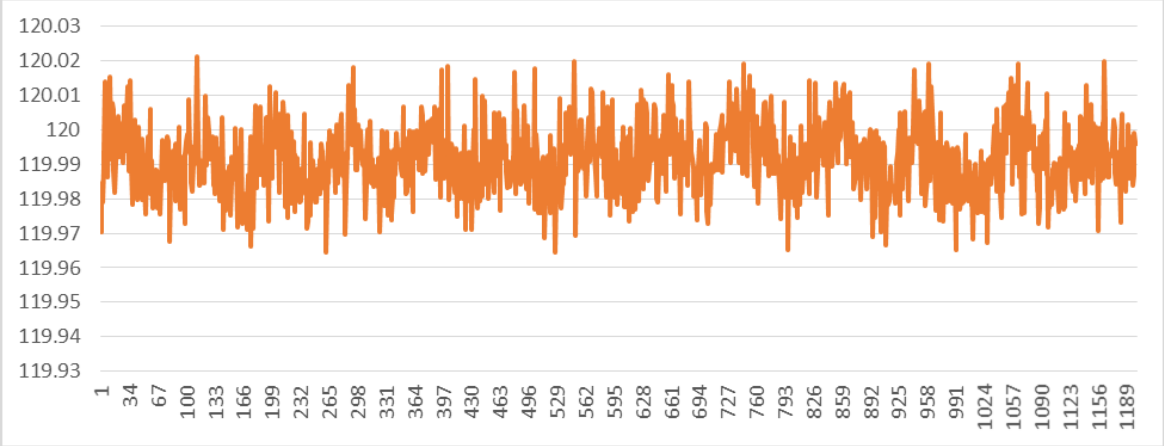


Figure 5-42 Voltage Magnitude; 60.1 Hz

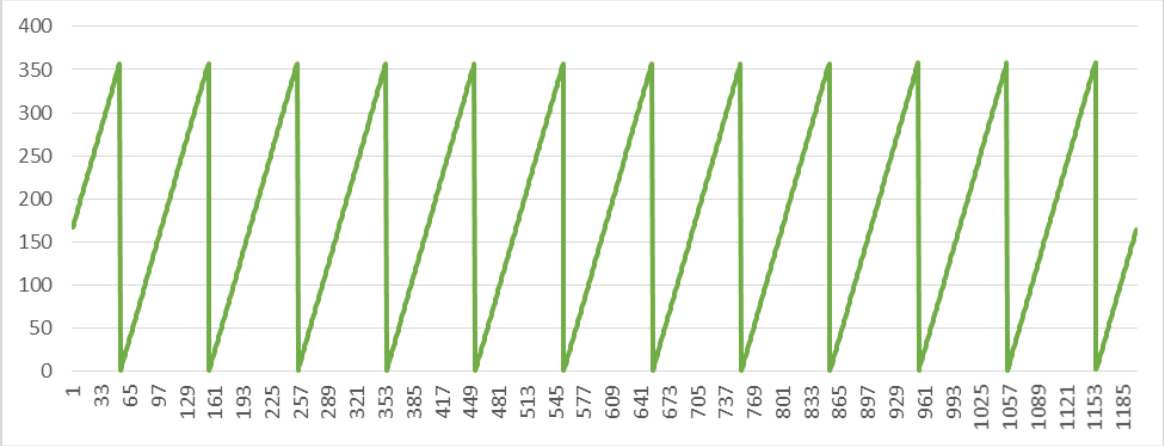


Figure 5-43 Phase Angle; 60.1 Hz

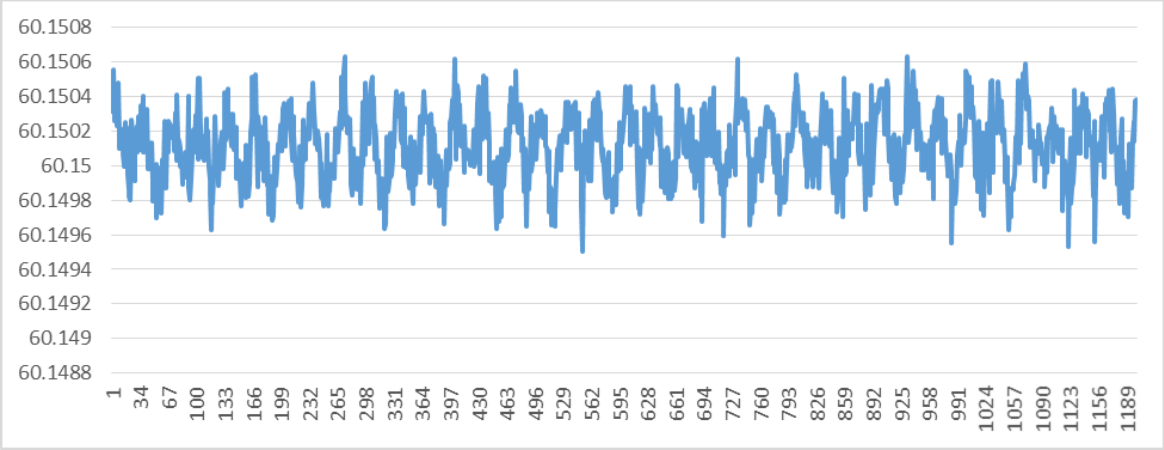


Figure 5-44 Calculated Frequency; 60.15 Hz

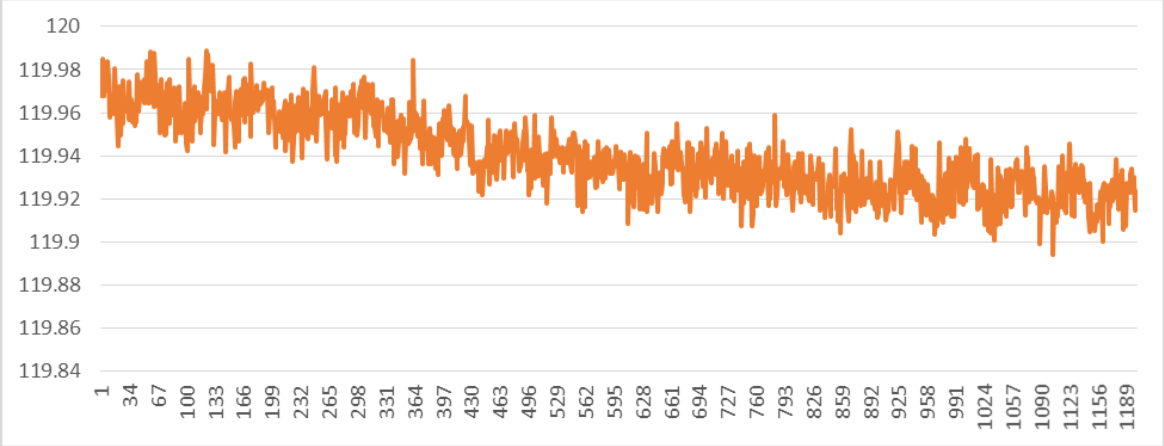


Figure 5-45 Voltage Magnitude; 60.15 Hz

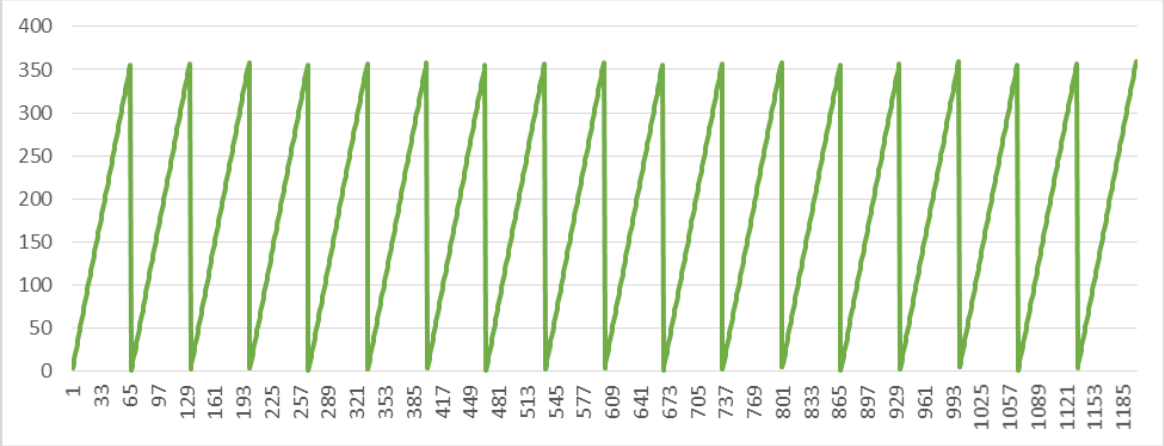


Figure 5-46 Phase Angle; 60.15 Hz

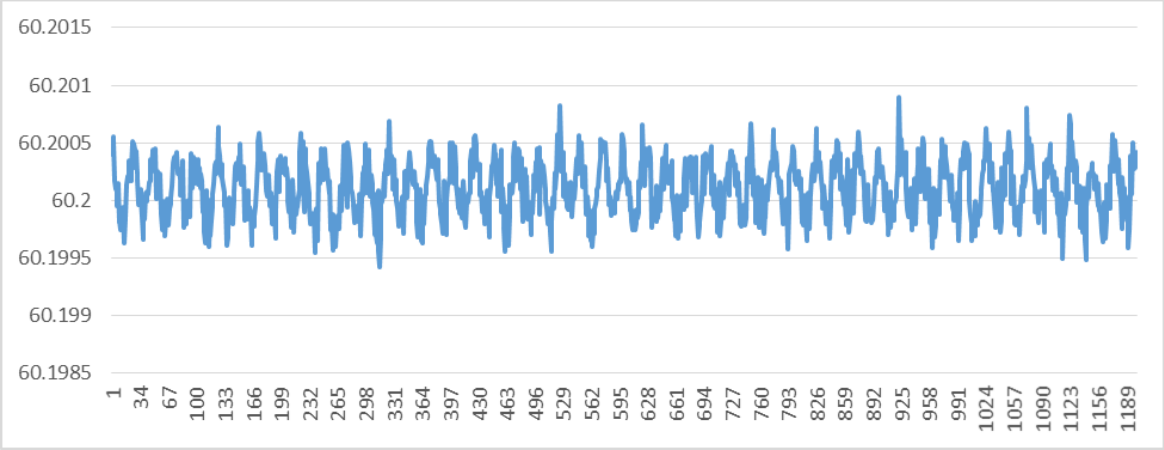


Figure 5-47 Calculated Frequency; 60.2 Hz

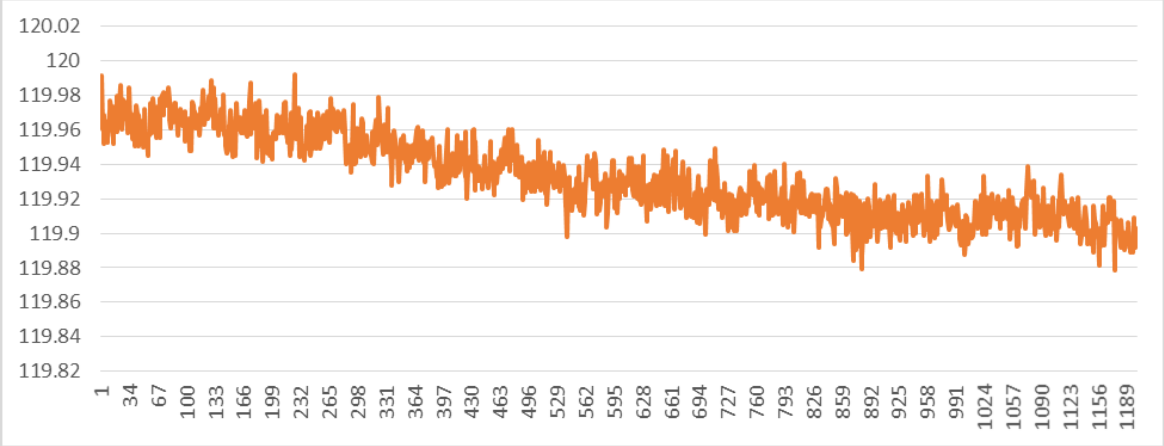


Figure 5-48 Voltage Magnitude; 60.2 Hz

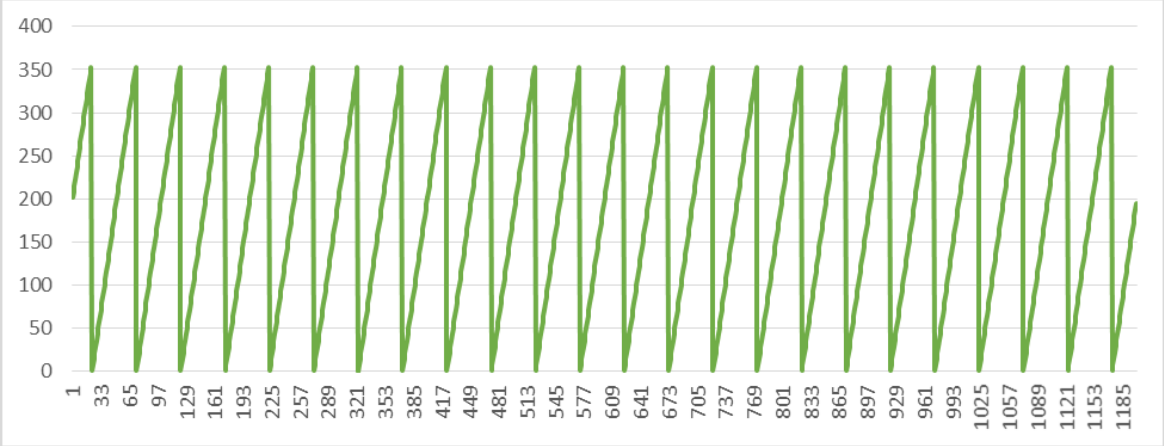


Figure 5-49 Phase Angle; 60.2 Hz

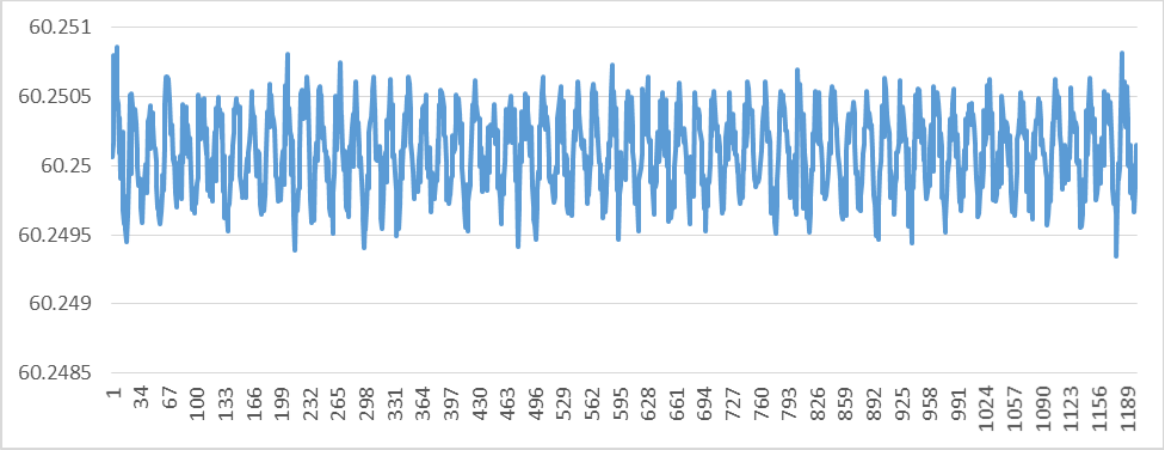


Figure 5-50 Calculated Frequency; 60.25 Hz

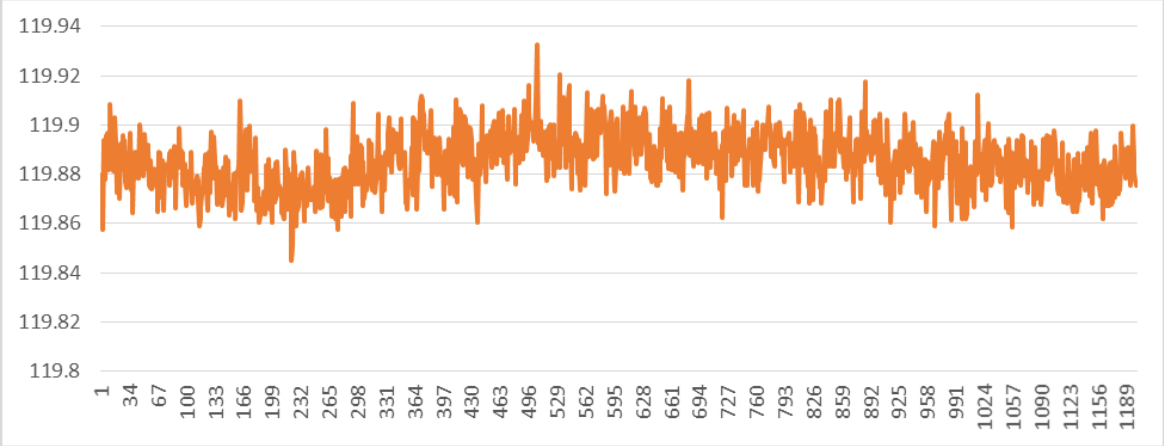


Figure 5-51 Voltage Magnitude; 60.25 Hz

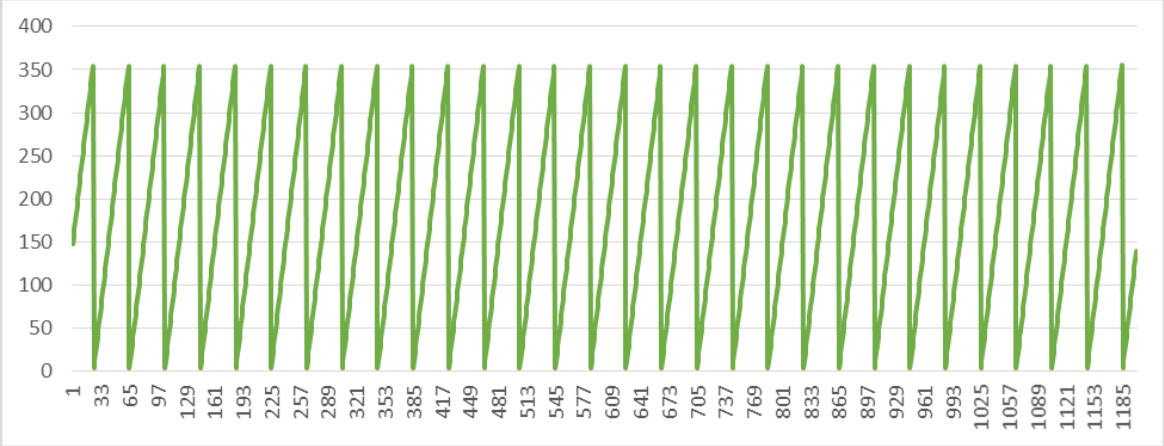


Figure 5-52 Phase Angle; 60.25 Hz

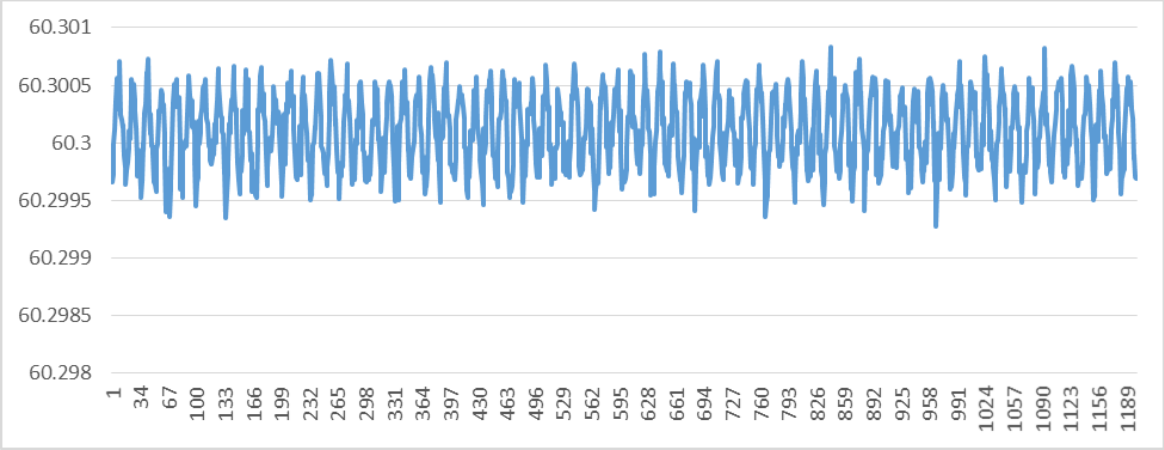


Figure 5-53 Calculated Frequency; 60.3 Hz

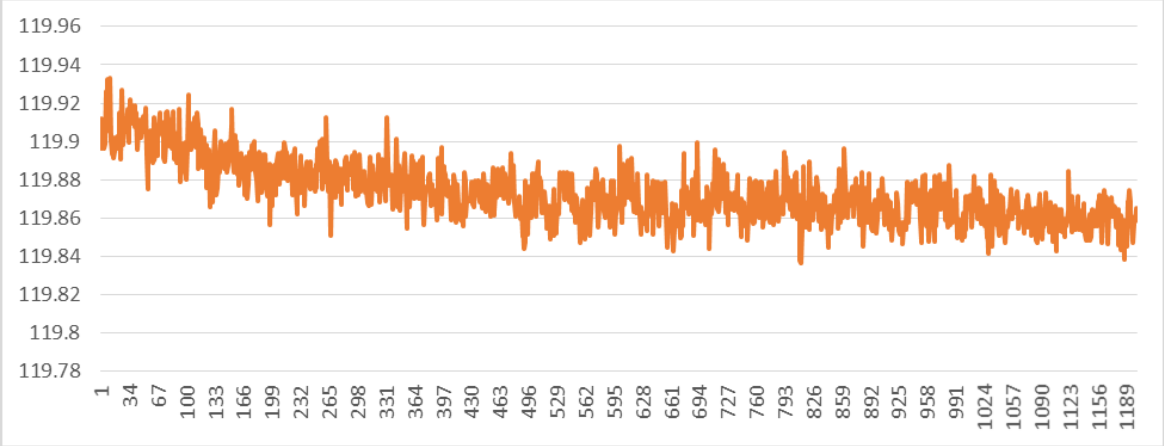


Figure 5-54 Voltage Magnitude; 60.3 Hz

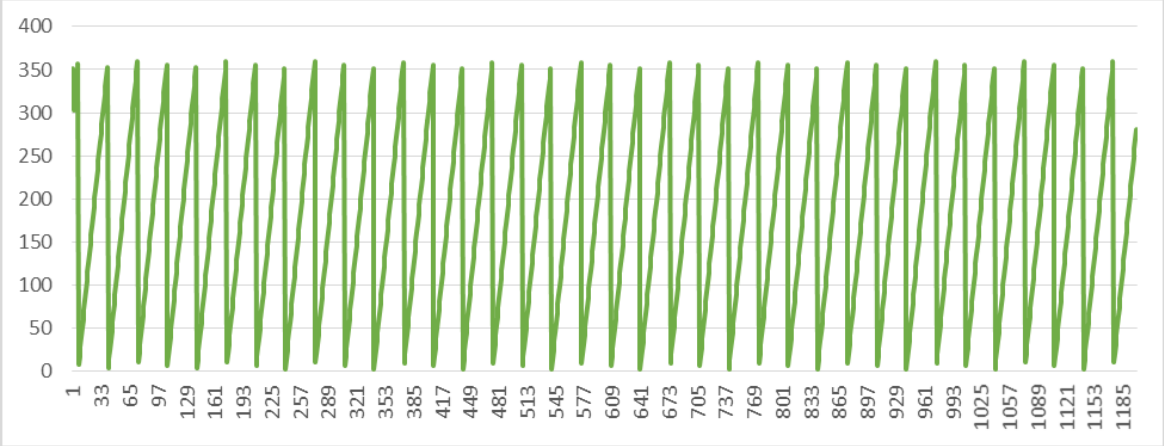


Figure 5-55 Phase Angle; 60.3 Hz

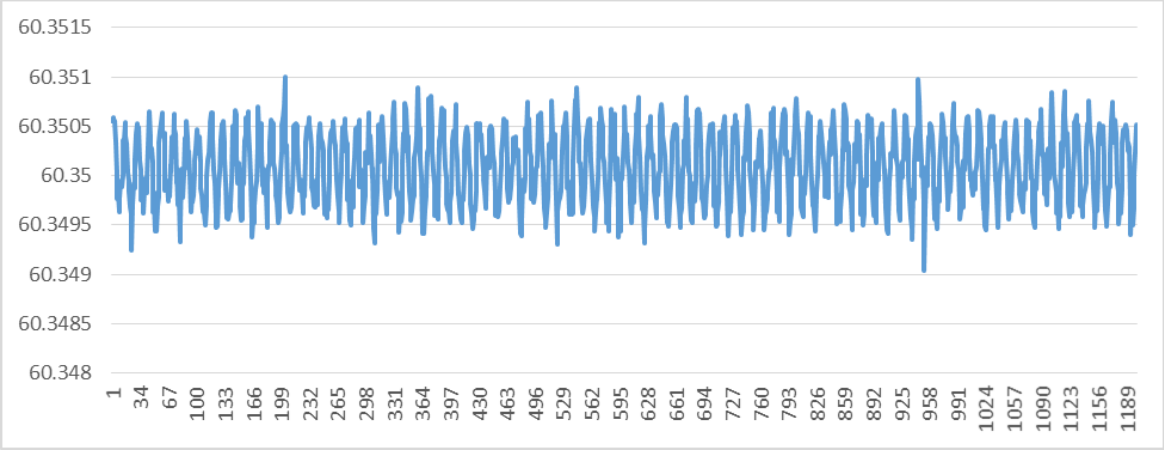


Figure 5-56 Calculated Frequency; 60.35 Hz

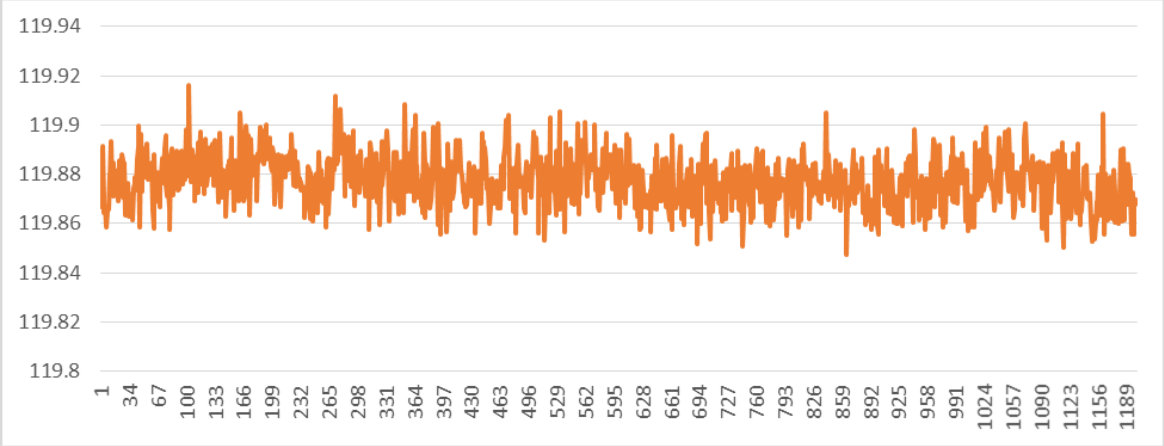


Figure 5-57 Voltage Magnitude; 60.35 Hz

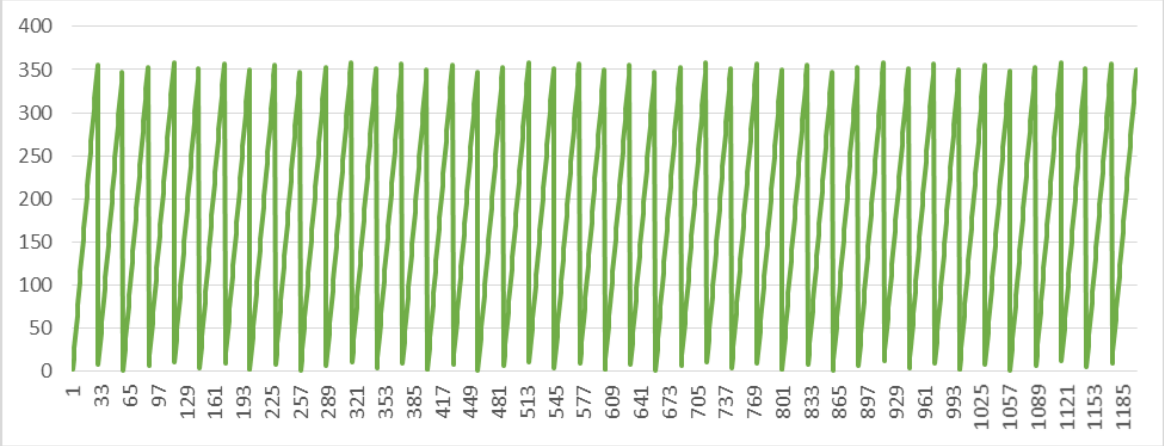


Figure 5-58 Phase Angle; 60.35 Hz

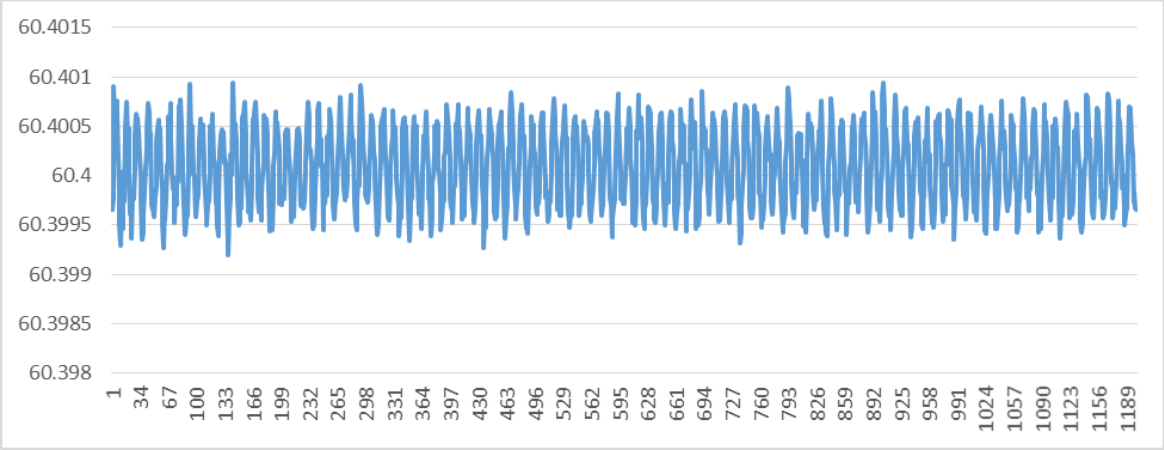


Figure 5-59 Calculated Frequency; 60.4 Hz

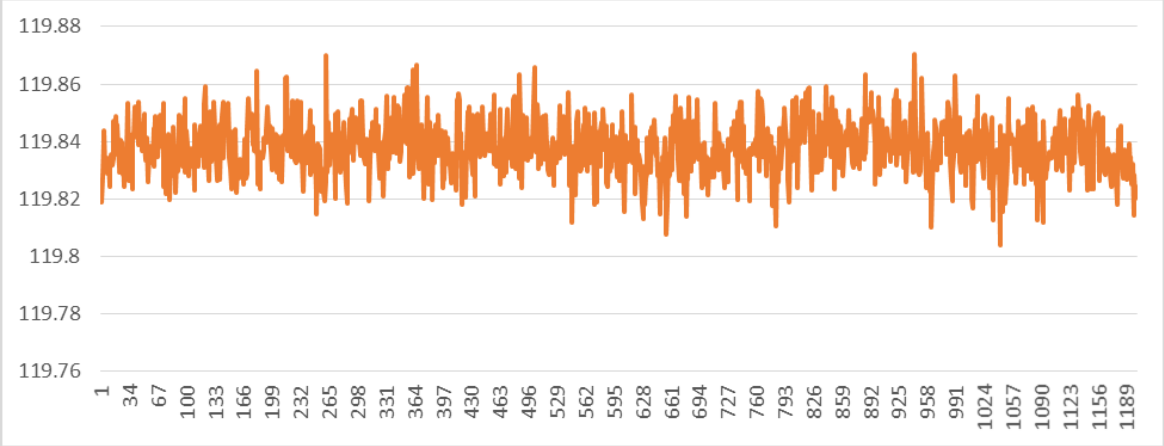


Figure 5-60 Voltage Magnitude; 60.4 Hz

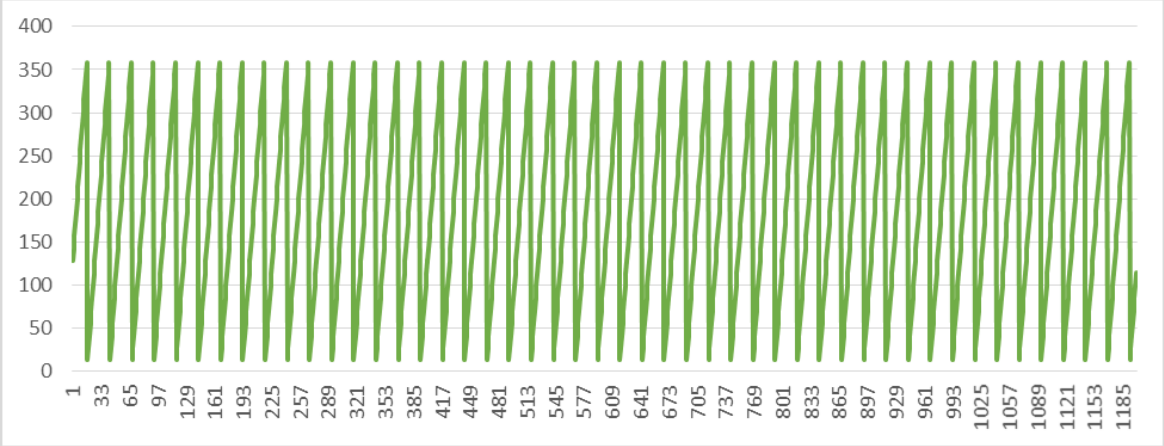


Figure 5-61 Phase Angle; 60.4 Hz

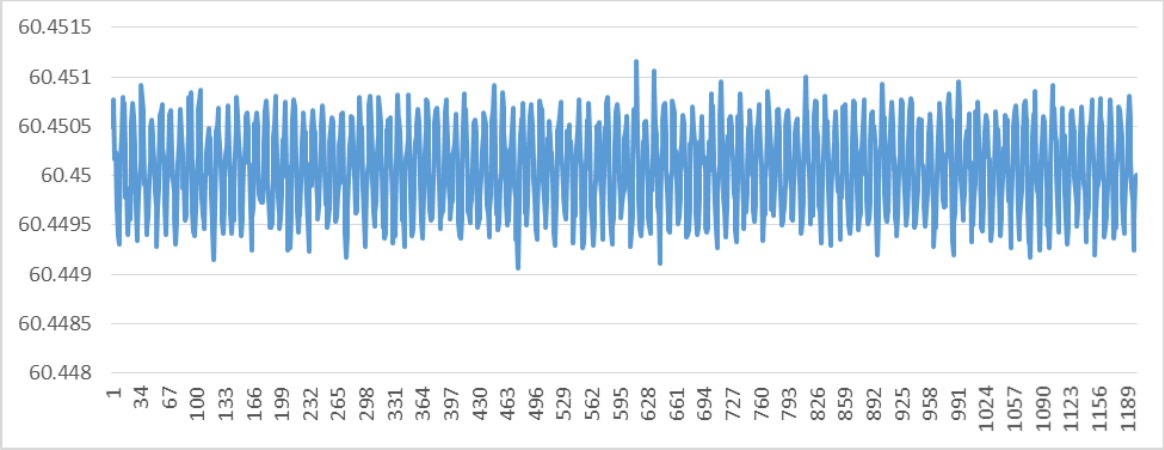


Figure 5-62 Calculated Frequency; 60.45 Hz

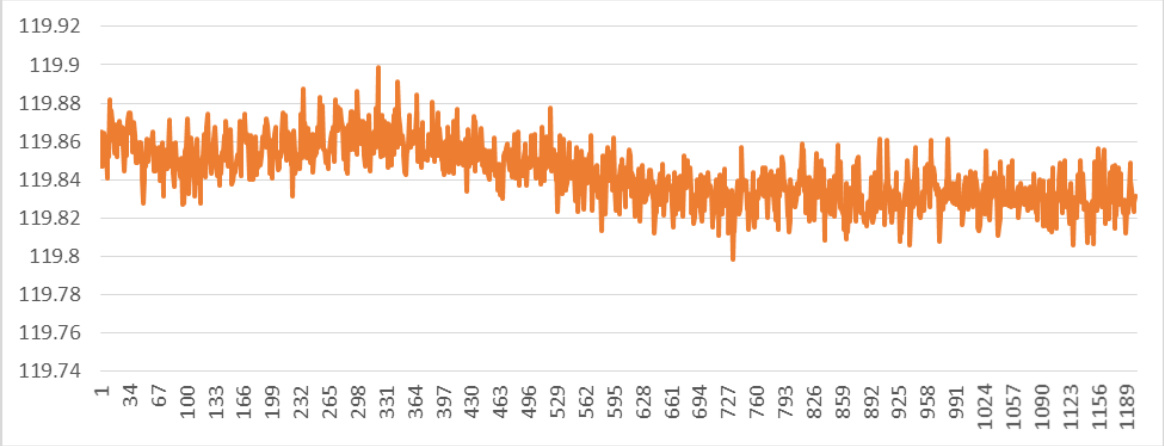


Figure 5-63 Voltage Magnitude; 60.45 Hz

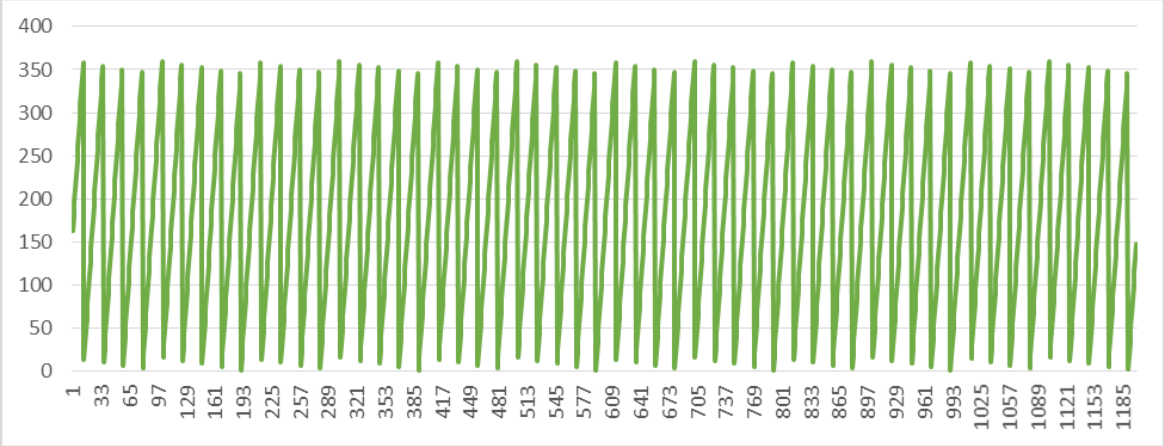


Figure 5-64 Phase Angle; 60.45 Hz

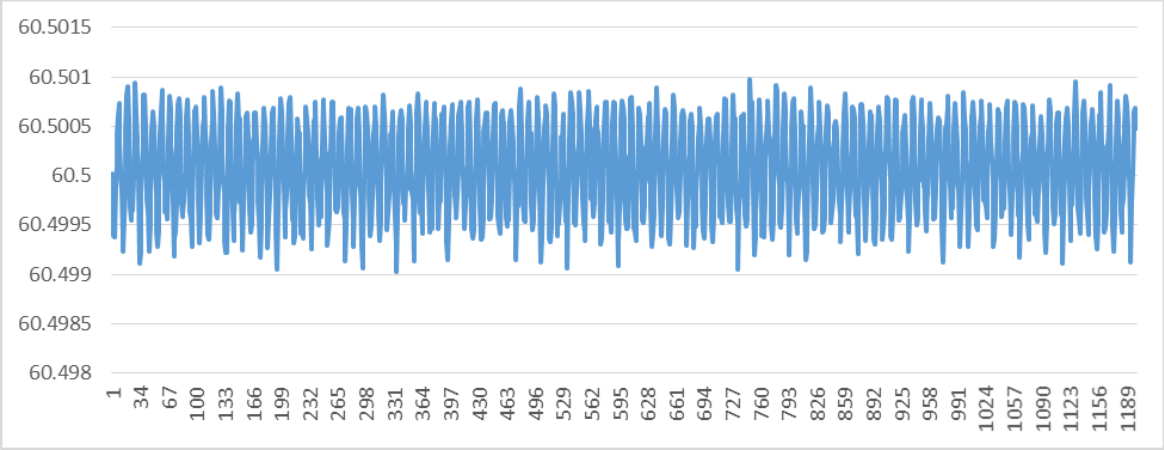


Figure 5-65 Calculated Frequency; 60.5 Hz

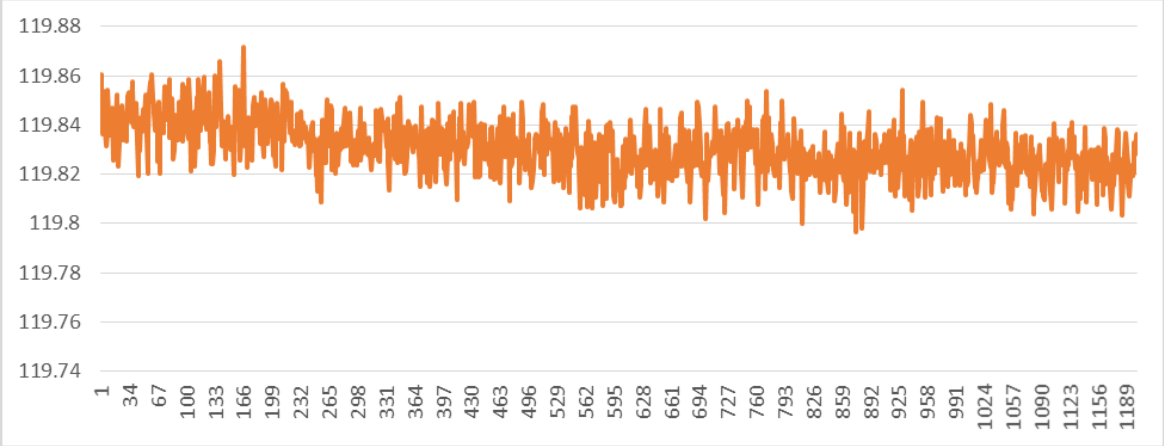


Figure 5-66 Voltage Magnitude; 60.5 Hz

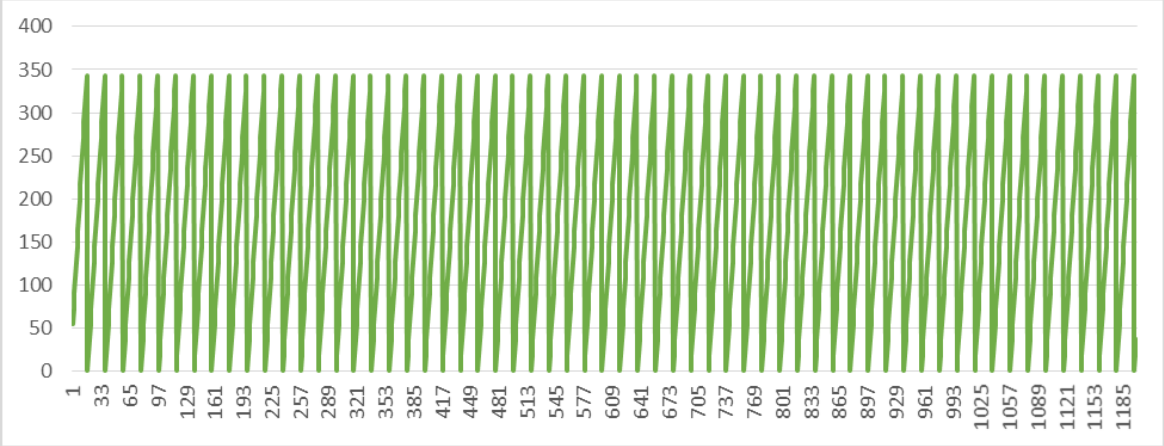


Figure 5-67 Phase Angle; 60.5 Hz

Prior to any statistical analysis, a cursory investigation of the graphs shows the following patterns in the data:

- Calculated frequency values have higher variance when off nominal-frequency.
- The variance in frequency values appears to correlate to the absolute deviation from the nominal frequency.
- Voltage magnitude tends to appear slightly higher below nominal frequency (60 Hz) and slightly lower above it.
- The precession of the phase angle is decreasing below nominal frequency (60 Hz) and increasing above it (as expected).

Using the equations for TVE and FE, and performing basic statistical analysis on the data, the following results are obtained:

Table 5-2 Signal Analysis – Expected Operational Range

Test	Mean (Hz)	Standard Deviation (Hz)	Mean (V)	Standard Deviation (V)	TVE (Average)	FE (Average)
59.5	59.50016854	0.000395283	120.1000743	0.012230439	0.000833952	0.00016854
59.55	59.55021118	0.000452733	120.0860703	0.011636725	0.000717252	0.00021117
59.6	59.60020072	0.000407913	120.0679095	0.01162286	0.000565912	0.00020071
59.65	59.65020125	0.000361902	119.9949369	0.010994785	4.21925E-05	0.00020125
59.7	59.70018077	0.000324594	120.0736664	0.015731308	0.000613886	0.00018076
59.75	59.75017793	0.000282898	120.0423311	0.012041044	0.000352759	0.00017792
59.8	59.80016719	0.000237412	120.018924	0.012454758	0.0001577	0.00016718
59.85	59.85016036	0.000207628	120.0209947	0.013559077	0.000174956	0.00016035
59.9	59.90015402	0.000178464	120.0126658	0.012608837	0.000105548	0.00015401
59.95	59.95014621	0.000145905	120.0014847	0.010943171	1.23728E-05	0.00014620
60	60.00010897	0.000139614	119.9967723	0.012230439	2.68976E-05	0.00010896
60.05	60.0501261	0.000146812	119.9956094	0.017093915	3.65881E-05	0.00012609
60.1	60.10011824	0.000181291	119.9916552	0.00983259	6.95401E-05	0.00011824
60.15	60.15011553	0.00019533	119.9401177	0.018354602	0.000499019	0.00011552
60.2	60.20010538	0.000240122	119.9325759	0.023428692	0.000561868	0.00010538
60.25	60.25010006	0.000289528	119.885657	0.011267091	0.000952859	0.00010005
60.3	60.30010114	0.000315385	119.8735631	0.015807422	0.001053641	0.00010114
60.35	60.35008384	0.000366033	119.8773899	0.010220987	0.001021751	8.38417E-05
60.4	60.40009163	0.00040197	119.8378473	0.009416325	0.001351272	9.16292E-05
60.45	60.45007199	0.00045057	119.8431813	0.015682237	0.001306822	7.19867E-05
60.5	60.50007513	0.000479315	119.8305022	0.010934599	0.001412482	7.51308E-05

The statistical analysis confirms what the initial inspection of the data suggests – the further away from the nominal frequency, the more pronounced the error in the frequency. A less apparent trend is that the average frequency error decreases even as the frequency moves away from the nominal in the positive direction. In all cases, the deviations from the expected values are minimal, as reflected in the TVE and FE calculations. The frequency range of this test encapsulates almost all expected operating points of the device, as severe grid stabilization measures take place starting at +/- 0.15 Hz off-nominal. Within this expected operating range, the maximum TVE occurs at 60.5 Hz, and is 0.14%. Maximum FE occurs at 59.55 Hz and is 0.02%.

As mentioned, while not all conditions set forth in the C37.118 standard are applicable, a key criterion for a device meeting "M-class" requirements is no more than 1% TVE at +/- 2.0 Hz from nominal [89]. While these values are well outside of the typical operating range, these extreme off-nominal conditions should show how much error could possibly be expected in an absolute worst case. Table 5-3 shows summary results of additional 58 Hz and 62 Hz tests with the nominal frequency results included for comparison. As with the tests in the 59.5-60.5 Hz range, the Doble F6150 Power System Simulator was utilized and measurements were recorded for 2 minutes.

Table 5-3 Signal Analysis – Frequency Extrema

Test	Mean (Hz)	Standard Deviation (Hz)	Mean (V)	Standard Deviation (V)	TVE (Average)	FE (Average)
58	58.00169566	0.005418403	120.3592322	0.011636725	0.002993602	0.001695661
60	60.00010897	0.000139614	119.9967723	0.012230439	2.68976E-05	0.000108965
62	61.99932781	0.005078481	119.1691446	0.01162286	0.006923795	0.000672191

The test results at +/- 2.0 Hz off-nominal do show increased TVE and FE when compared with the results from tests in the operational range. The maximum TVE occurs with the 62 Hz test – 0.69%, which is still within the 1% TVE requirement set forth by C37.118. The Frequency Error is likewise increased at these frequency extrema, with the maximum value of 0.17% corresponding with the 58 Hz test. These results concur with the pattern established in prior testing. Also as expected from the initial results, the voltage measurement is higher below nominal frequency, and lower when above nominal frequency. While the full range of C37.118 tests cannot be applied to the FDR, the steady-state frequency tests performed on the prototype unit indicate that the measurements are within acceptable tolerances.

5.2. FNET Integration

This section deals with field testing – ensuring that the Generation-3 FDR is able to reliably communicate with the FNET server and that the measurement data sent back agrees with that provided by the existing Generation-2 devices. The procedure for this is straightforward – the unit was connected to a wall outlet and provided with Internet connectivity through a laptop computer. This allows for local logging of results as well as serving as a multi-server communication test – the laptop is acting as a local FNET frame-receiving server. The unit shown sends data to both the local server as well as the main FNET server. Figure 5-68 shows the FNET Server application receiving data from the prototype Generation-3 unit, labeled as "UsNextGen9001."

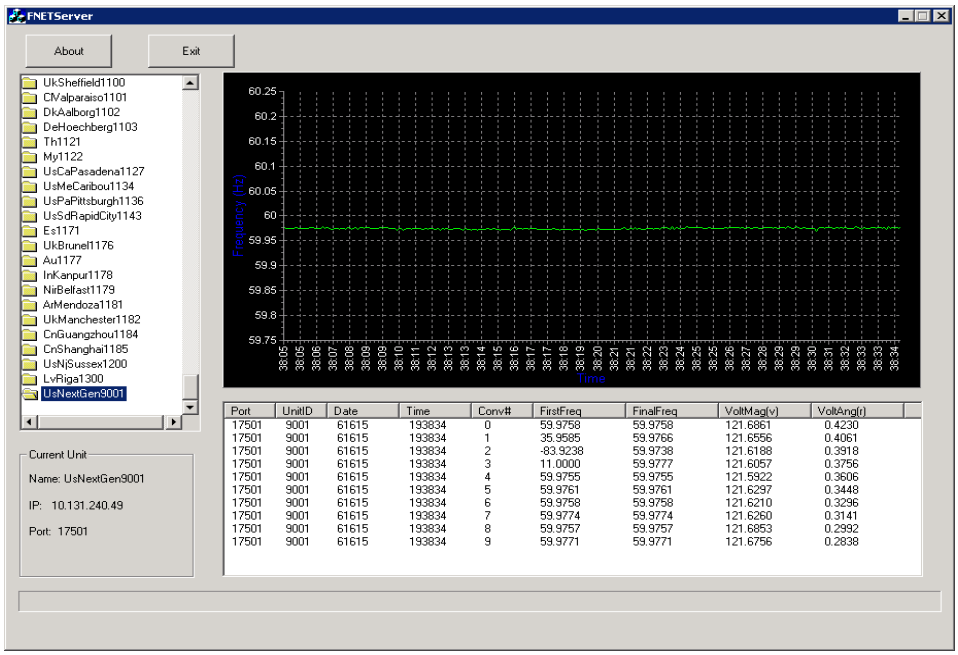


Figure 5-68 FNET Server Application – Incoming Data

As shown in Figure 5-68, the FNET frame data stream is being properly received by the server. The first 3 frames of each second transmit the latitude, longitude, and number of connected satellites, respectively. To verify that the unit produces useful data, event plots were generated which include the prototype device, "UsNextGen9001" as one of the selected units. While network conditions are unpredictable, the unit maintained higher than a 99.9% transmission rate during the field test. In three specific instances, 10 to 20 packets of data were lost. Example events captured by the unit as part of the field test are shown in Figure 5-69 through Figure 5-72. Each plot shows 10 units – the Generation-3 FDR and 9 others from across the Eastern Interconnect.

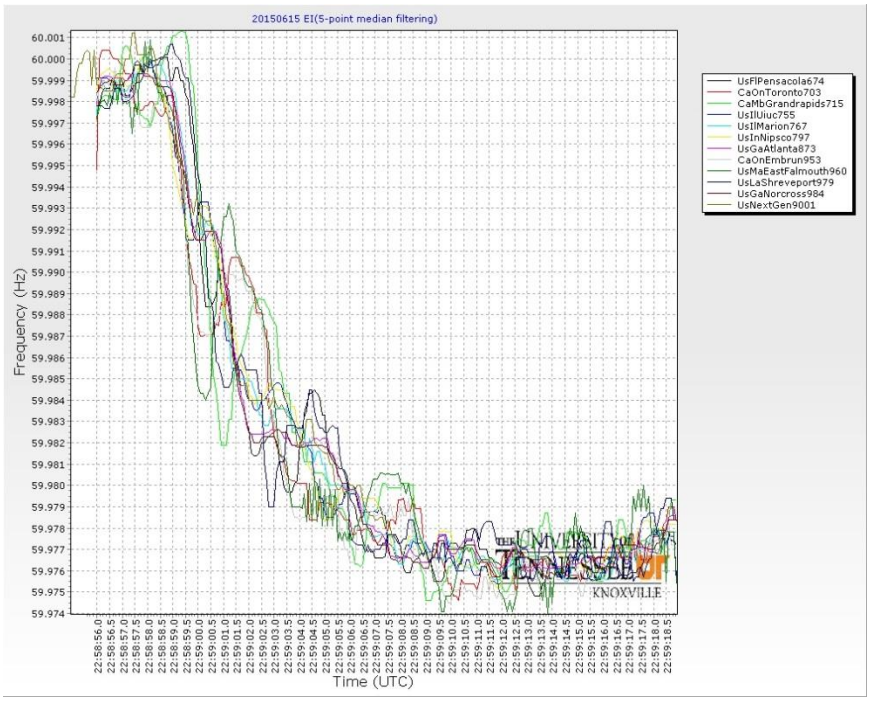


Figure 5-69 FNET Integration – 560 MW Generation Trip



Figure 5-70 FNET Integration – 920 MW Generation Trip

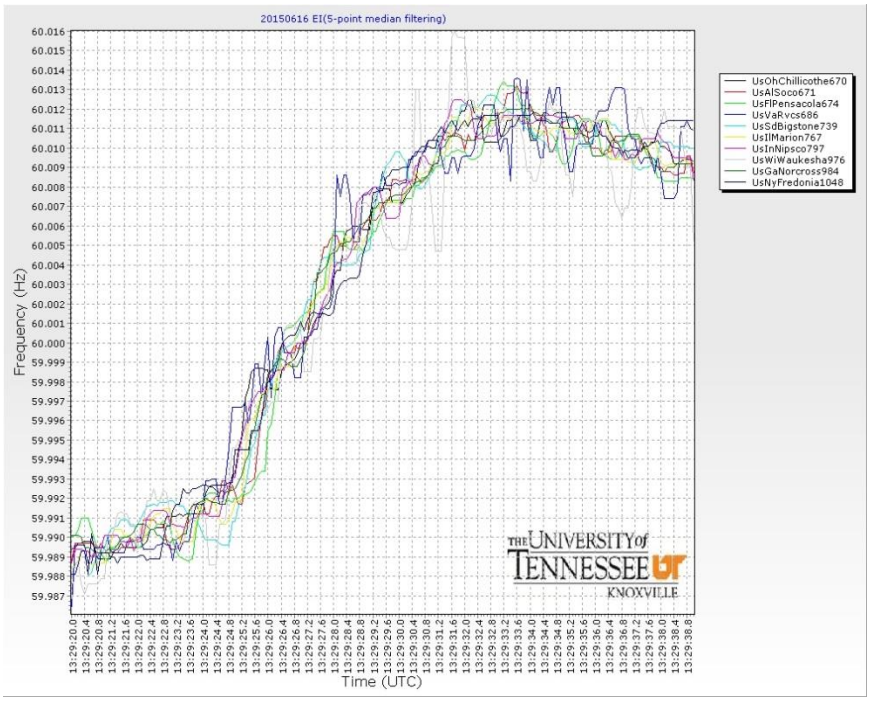


Figure 5-71 FNET Integration – 500 MW Load Shedding

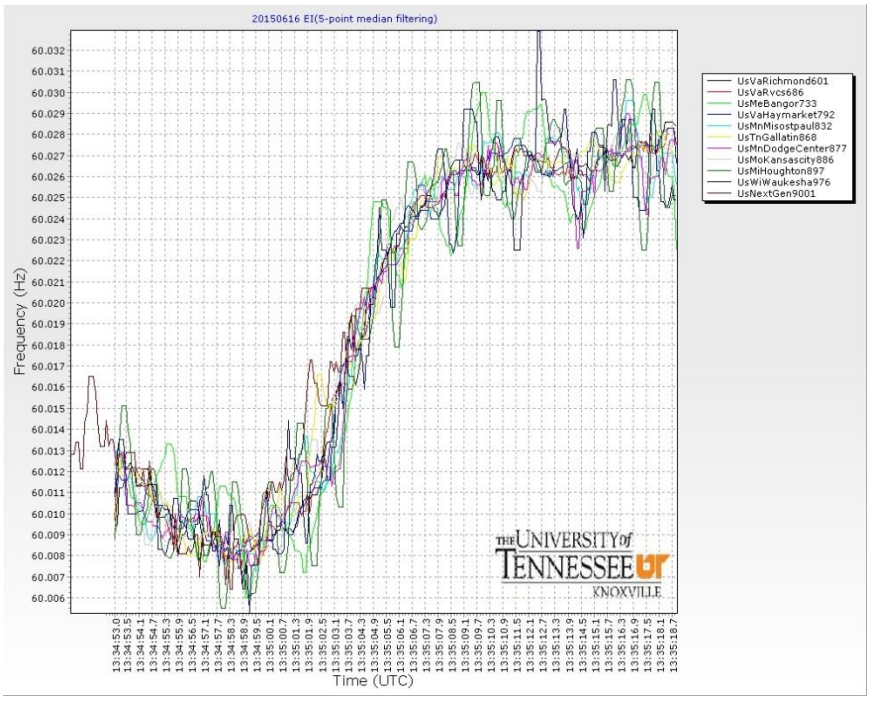


Figure 5-72 FNET Integration – 430 MW Load Shedding

In each of these events, the Generation-3 FDR's measurements agreed with the general trend shown by the FNET system. Typical event reports include far more units – the low number used in these examples is so that the line corresponding to the test unit can be easily seen.

5.2.1. Super Bowl XLIX

Each year, the NFL Super Bowl provides an excellent case study for the resiliency of the North American power grid and serves as a sort of mass use case for FNET as a whole. The phenomenon of mass social behavior having impacts on the power grid has been studied and analyzed in depth [90]. As mentioned, the goal of the work presented in Section 5.2 is to ensure that the Generation-3 FDR can communicate with the existing FNET system. However, barring unusual events, the power grid is not unusually stressed and there is no guarantee of a large number of example events to validate the correct operation of the device. While the signal testing presented in Section 5.1 suggests that the device is capable of monitoring a grid event, field testing during such events is ultimately necessary. On February 1, 2015, Super Bowl XLIX (49) – played between the Seattle Seahawks and New England Patriots – provided exactly that opportunity. During the game, there were 14 grid events in the Eastern Interconnect. A selection of five events from the game is presented which show the overall response of the FNET system and then the local response of the Generation-3 prototype unit. A summary of these events is provided in Table 5-4.At the time of the Super Bowl, the device was not connected to the overall FNET system, so the measurements from the prototype unit were sent directly to a connected laptop.

Table 5-4 Selected Events during Super Bowl XLIX

Time (UTC)	Event	Circumstances
1:13:26	690 MW Generation Trip	Halftime Show - Roar / Hot N Cold
1:39:31	Oscillation	Commercial
1:41:50	790 MW Load Shedding	Commercial
2:30:37	710 MW Generation Trip	Commercial following 3rd New England Touchdown
2:52:10	950 MW Generation Trip	Commercial following 4th New England Touchdown

This group of 5 events is selected so that an example of each major disturbance is presented – a Generation Trip, a Load Shedding, and an Oscillation. There is no common underlying cause for these 5 events, but there is an established correlation between light-dark and dark-light transitions in the television signal, as modern televisions draw very little power to display a dark screen and considerably higher amounts to display bright colors. These abrupt transitions, when enough televisions are viewing

the same source, can cause a swing in the power grid. Anecdotally, Super Bowl 48 had fewer total power grid events, and no events at all in the second half of the game, which can be attributed to a mass loss of interest when the game ceased to be competitive.

Event 1 is a typical generation trip, as the system frequency decreases rapidly within a short time frame. The timing of this event corresponds to a song transition in Katy Perry's set list during the Super Bowl halftime show, specifically a stadium lighting shift between "Roar" and "Hot N Cold."

Event 2 is an oscillation which is potentially associated with another event – a 560 MW Load Shedding which happened within 5 seconds of the window. As expected, the oscillation is simply one or more exaggerated deviations in the frequency.

Event 3 is a typical load shedding, as system frequency quickly rises and then is damped back down to nominal frequency. This event took place during a commercial, which is typically when power draw can fluctuate due to aforementioned social and technical factors.

Event 4 occurred following resumption of the game after New England's 3rd touchdown. It is emblematic of a generation trip, as the system frequency decreases rapidly, but settles around the nominal frequency.

Event 5 occurred during the commercial immediately following New England's 4th touchdown, which occurred at the end of the game. A massive swing in viewership most likely precipitated this, as noted by a rapid decrease in frequency, with a small overcorrection.

In each test, the prototype unit proved able to track the "big-picture" frequency changes in time with the established FNET system. Differences in the values can be attributed to the natural operation of the system – event triangulation relies on each unit's observation being unique. Calibration and unit position both influence these offsets from the average.

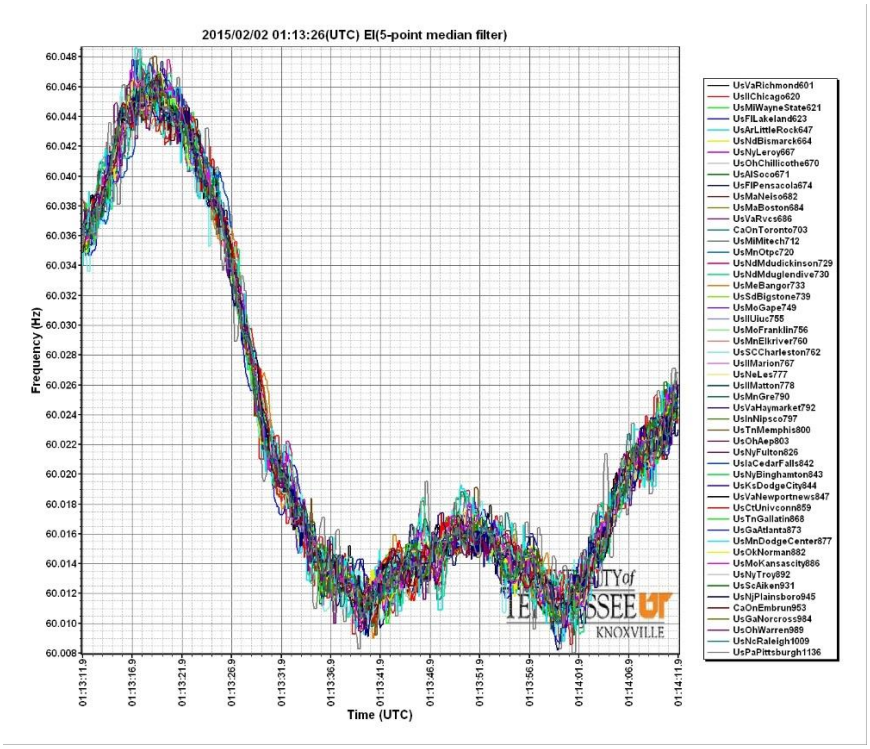


Figure 5-73 Event 1: Super Bowl 690 MW Generation Trip – FNET Report

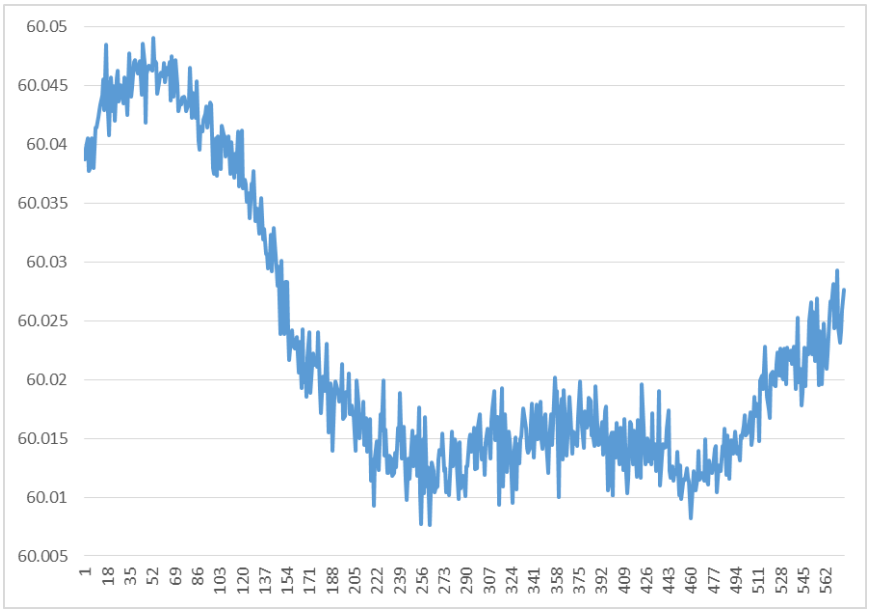


Figure 5-74 Event 1: Super Bowl 690 MW Generation Trip – Prototype Measurement

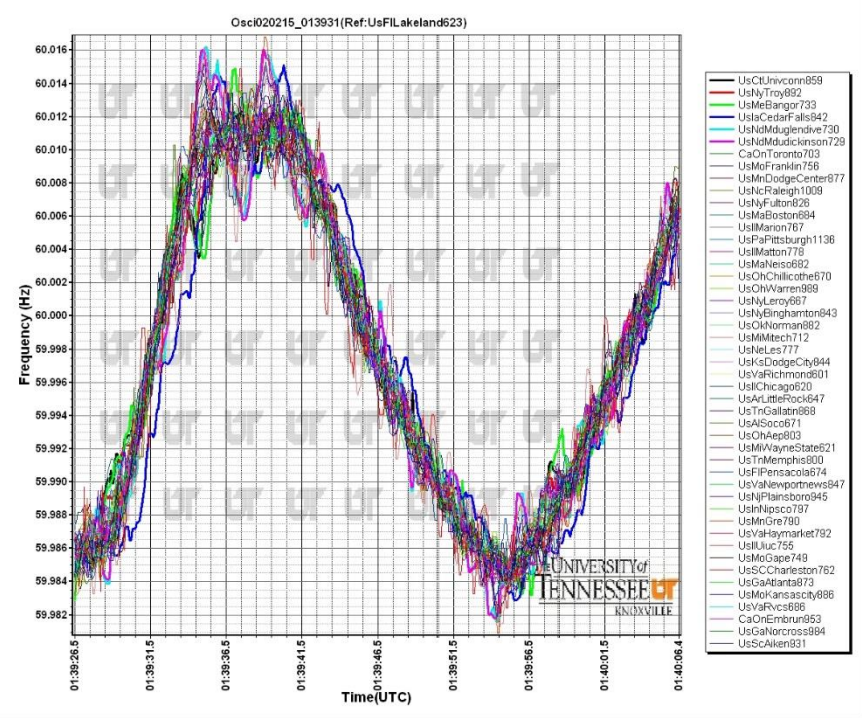


Figure 5-75 Event 2: Super Bowl Oscillation – FNET Report

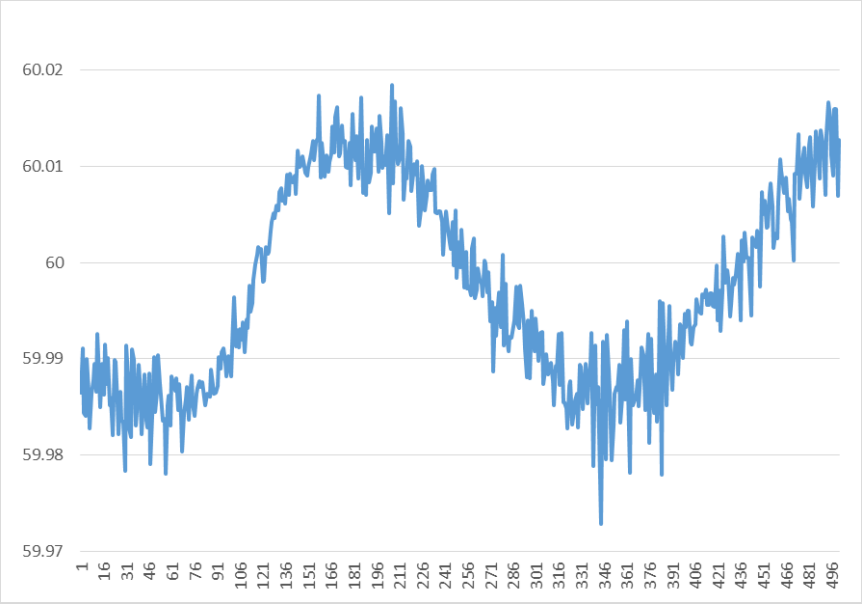


Figure 5-76 Event 2: Super Bowl Oscillation – Prototype Measurement

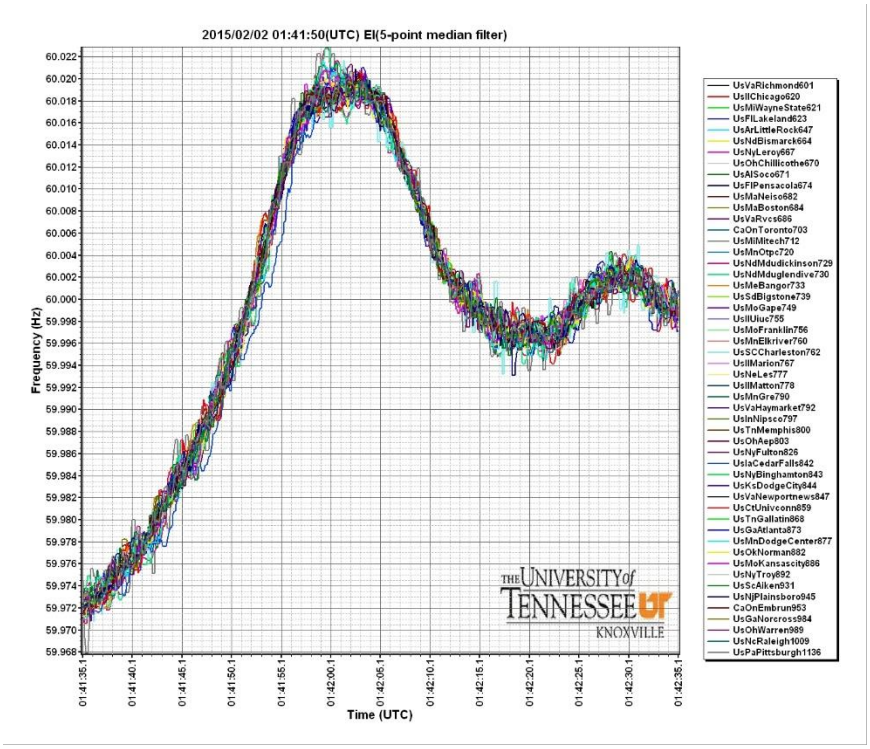


Figure 5-77 Event 3: Super Bowl 790 MW Load Shedding – FNET Report

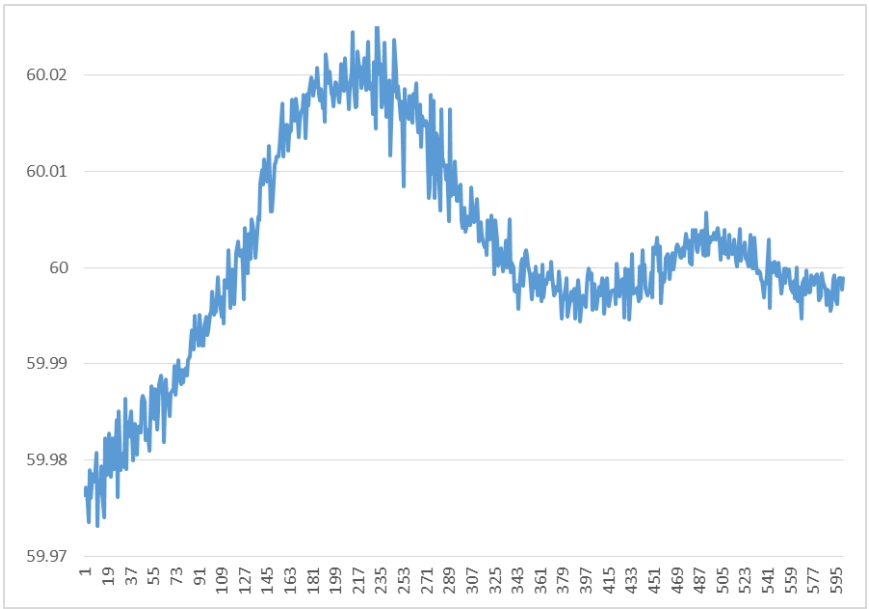


Figure 5-78 Event 3: Super Bowl 790 MW Load Shedding – Prototype Measurement

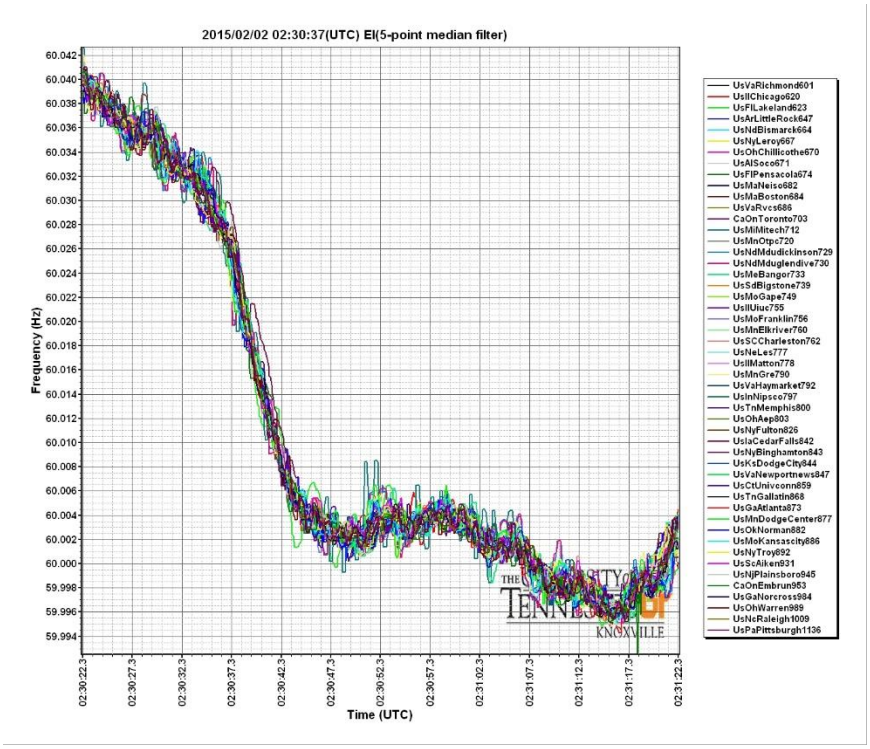


Figure 5-79 Event 4: Super Bowl 710 MW Generation Trip – FNET Report

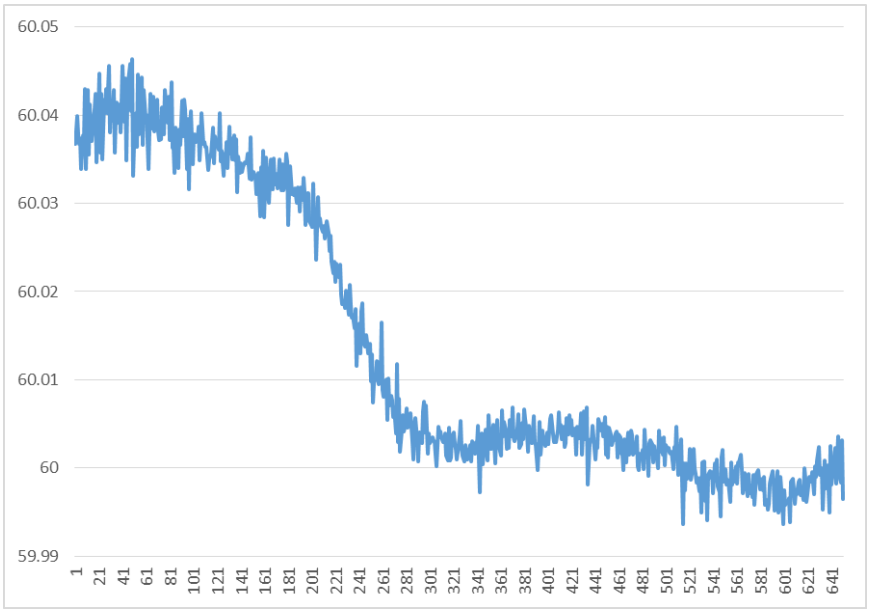


Figure 5-80 Event 4: Super Bowl 710 MW Generation Trip – Prototype Measurement

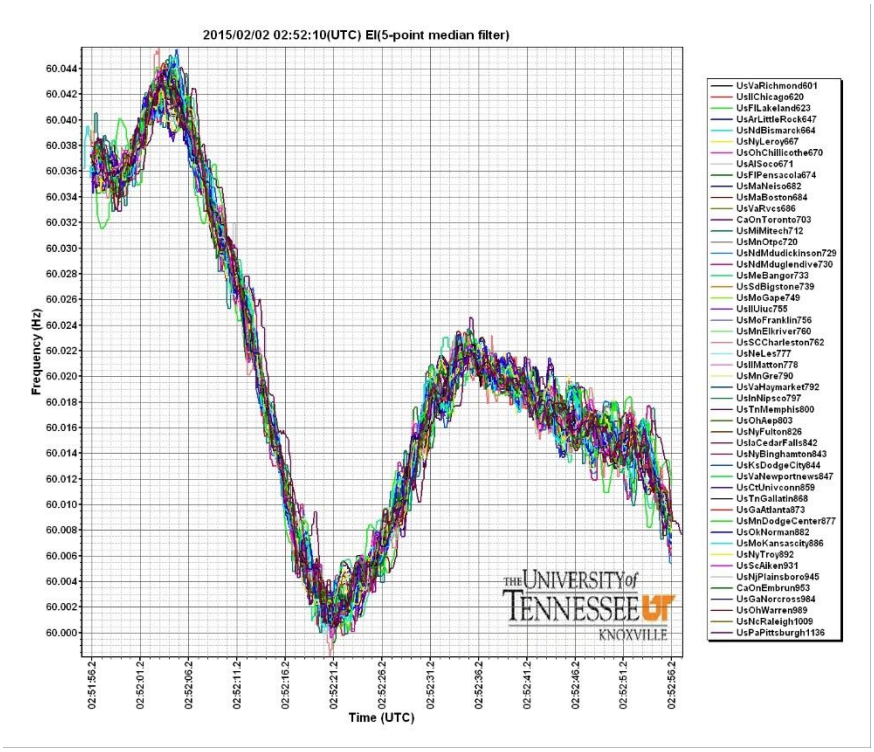


Figure 5-81 Event 5: Super Bowl 950 MW Generation Trip – FNET Report

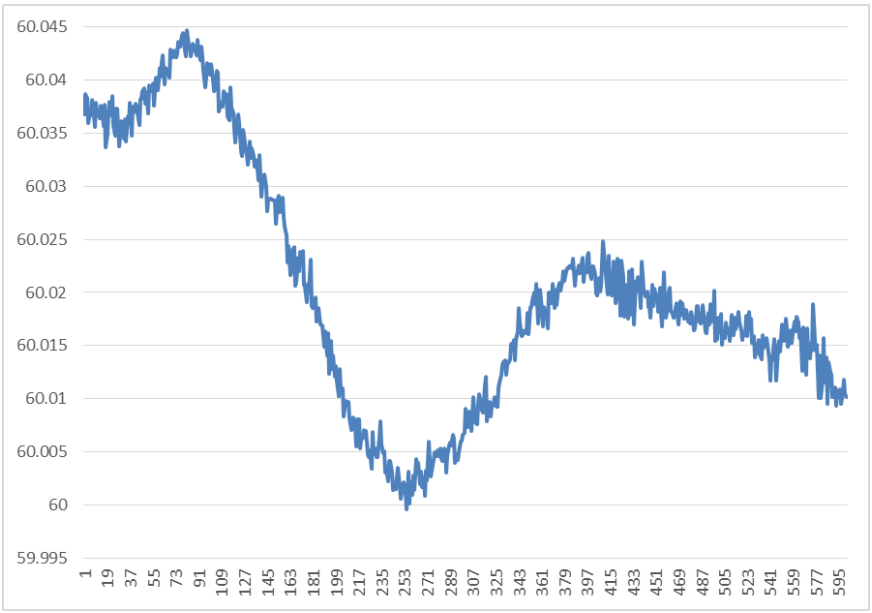


Figure 5-82 Event 5: Super Bowl 950 MW Generation Trip – Prototype Measurement

5.3. TLS-Enabled Communication Tests

The prior sections have focused on the performance of the device and its integration into the existing FNET system. This section focuses on the testing of the added capability unique to the new unit – encryption. Specifically, a basic test of encrypted communications as well as a simulated mass encryption stress test are presented. The former serves to verify that encrypted communication is working as designed, while the purpose of the latter is to determine the feasibility of mass deployment when encryption is utilized. The procedure for verifying encrypted communication is straightforward – a test server program is written in Python using the same *ssl* library utilized on the device. The only purpose of the server is to receive data over an encrypted connection and then echo that data to the screen, confirming reception.

The more difficult test is to determine if large-scale encryption is feasible for the FNET system as a whole. As mentioned in Section 4.5, there is a computational overhead cost incurred when using TLS. While this overhead is minimal from the perspective of an individual unit, a server which must handle multiple incoming connections incurs a proportionally large overhead. For this test, steady-state operation is of most concern – the situation where all units are already connected and transmitting data. At a high level, the asymmetric handshake incurs a much higher overhead penalty than the established connection, which uses symmetric encryption, but this handshake cost only occurs during an initial connection or dropped connection. Even in an event where the server drops all incoming connections and the remote units must all reconnect, the asymmetric handshakes will proceed as a queue and steady-state operation will resume in time.

Therefore, the testing procedure consists of using a large amount of simulated clients, which all stream data to the same server. A client is simulated using a discrete connection which sends 10 64-byte packets every second. The result of this test was conclusive – *there is no noticeable effect on the operation of the server with up to 1024 TLS encrypted connections.* The limit of 1024 is set by the operating system, and could be circumvented in a full server environment. Regardless, the testing shows that at least for the Test Server (a ThinkPad laptop), *the computer can easily process an amount of encrypted connections far larger than the current number of deployed FDRs.*

The simulated client test provides a real-world benchmark for mass encrypted communication, though an important caveat is that the computer acting as the test server is far more capable than the computer which is currently the main FNET server. Given this fact, an alternative benchmark should prove useful in determining the encryption potential of existing hardware. While hardware

specifications such as CPU, RAM, and network throughput are typically indicative of the power of a computer, the benchmark which is more applicable is OpenSSL. The OpenSSL benchmark calculates the total encryption/decryption bandwidth available to a processor, for a given cipher. In this case, 128-bit Advanced Encryption Standard with Cipher Block Chaining is used (AES-128-CBC). Figure 5-83 shows the command and results for the FNET server.

```
C:\OpenSSL-Win64\bin>openssl.exe speed -evp aes128
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Doing aes-128-cbc for 3s on 16 size blocks: 18427367 aes-128-cbc's in 2.95s
Doing aes-128-cbc for 3s on 64 size blocks: 5163435 aes-128-cbc's in 2.89s
Doing aes-128-cbc for 3s on 256 size blocks: 1383461 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 1024 size blocks: 347101 aes-128-cbc's in 2.98s
Doing aes-128-cbc for 3s on 8192 size blocks: 43801 aes-128-cbc's in 2.96s
OpenSSL 1.0.2c 12 Jun 2015
built on: reproducible build, date unspecified
options:bn(64,64) rc4(16x,int) des(idx,cisc,2,long) aes(partial) idea(int) blowf
ish(idx)
compiler: cl -D_USING_V110_SDK71_ -D_WINSOCK_DEPRECATED_NO_WARNINGS  /MD /Ox -DO
PENSSL_THREADS  -DDSO_WIN32 -D_USING_V110_SDK71_ -D_WINSOCK_DEPRECATED_NO_WARNIN
GS -W3 -Gs0 -Gy -nologo -DOPENSSL_SYSNAME_WIN32 -DWIN32_LEAN_AND_MEAN -DL_ENDIAN
 -DUNICODE -D_UNICODE -D_CRT_SECURE_NO_DEPRECATE -DOPENSSL_IA32_SSE2 -DOPENSSL_B
N_ASM_MONT -DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM
-DSHA512_ASM -DMD5_ASM -DAES_ASM -DVPAES_ASM -DBSAES_ASM -DWHIRLPOOL_ASM -DGHASH
_ASM -DECP_NISTZ256_ASM -DOPENSSL_USE_APPLINK -I. -DOPENSSL_NO_RC5 -DOPENSSL_NO_
MD2 -DOPENSSL_NO_KRB5 -DOPENSSL_NO_JPAKE -DOPENSSL_NO_STATIC_ENGINE
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
aes-128-cbc      99998.64k   114503.72k   118243.77k   119287.54k   121057.86k

C:\OpenSSL-Win64\bin>
```

Figure 5-83 OpenSSL Benchmark – FNET Server

Table 5-5 contains the test results from each computer. Values shown are in thousands of bytes per second processed. It must be noted that this benchmark was performed while the server was in operation, so the results are expected to be slightly lower than the hardware's maximum potential.

Table 5-5 OpenSSL Benchmark Results

Computer	16	64	256	1024	8192
Test Server	642005.16k	732721.28k	748184.02k	750961.06k	757699.39k
FNET Server	99998.64k	114503.72k	118243.77k	119287.54k	121057.86k

The benchmark tests for the processing throughput of five different packet sizes – 16 bytes up to 8192 bytes. The encrypted packets sent by the Generation-3 FDR are 64 bytes – the standard FNET frame size, with additional packing used to reach the size necessary for encryption. At this size and transmission rate, a given unit sends 640 bytes of data to the server every second, whereas the server is able to process 114,503,720 bytes of encrypted data every second. On a high level, the benchmark suggests that the server is capable of decrypting data from well over 100,000 units simultaneously. Even considering processing load which may reduce the effective performance, this number is far larger than the bandwidth of the network connection available to the server. A more modern computer possesses upwards of 5 times this capability. Converting this value to a time delay, assuming a linear relationship, the decryption process requires approximately 2-3 microseconds of computational time for each packet. Ultimately, the computational delay introduced by encrypted communications is orders of magnitude lower than the communication latency over the Internet. While the tests presented here are straightforward – the results are very clear: the TLS implementation on the Generation-3 FDR does not incur any noticeable performance penalties on either the server or devices. The server hardware is more than capable, although the FNET server software would require a substantial rewrite in order to support the new framework.

6. Conclusions and Future Research

6.1. Conclusions

As the power grid evolves, phasor measurement units have become a cornerstone technology in the monitoring of wide area events. FNET continues to act as a pioneer in this field, with FDRs being deployed in new locations in new and existing grids. To further develop the capabilities of the FNET system, units must be deployed in larger quantities in more locations. To that end, the goal of this research has been to assist in the development of the FNET system as whole by providing a new unit which addresses limitations present in the current Generation-2 devices. These limitations include the cost of the device, effort required in maintaining them, and their methods of communication.

Chapters 3 and 4 presented the research and development required for the creation of a new Frequency Disturbance Recorder. As noted in Chapter 4, Linux is not appropriate for true real-time operation without specific underlying hardware capabilities. A hardware survey in Chapter 3 showed which SOC technologies were viable for the creation of such a system. In this case, the PRU on the BeagleBone Black provides a method for a real-time process to interact with a non-real-time layer. This structure allows for the advantages that come with a modern processor and operating system – modular software processes, process scheduling, integrated network communications, and encryption.

The signal quality tests performed on the prototype Generation-3 FDR revealed that the measurements are more than adequate to determine the frequency, voltage magnitude, and phase of a given signal. The tests in the operational range (59.5-60.5 Hz) show a minor frequency and voltage skew as the frequency deviates from the nominal. Additionally, using the applicable portions of the C37.118 criteria shows that the Total Vector Error (TVE) and Frequency Error (FE) are within the tolerance levels as required for an “M-class” device. FNET Integration of the Generation-3 prototype was shown to be successful in Section 5.2, and the annual opportunity afforded by the Super Bowl allowed for a “big-picture” look at the event reporting capability of the device.

The new communication features of the device have been shown, with the ability to send both FNET and C37.118 data streams, coupled with the ability to add encryption. This particular step does not currently exist in the power industry – while encryption is utilized, it is typically performed by a Phasor Data Concentrator (PDC) or equivalent device. Low-level encryption directly from the device is a new step, and the tests performed show that encryption is feasible on both a device and system level. The FNET server software will require fundamental rewrites to support encrypted communication, but there

is nothing inherent to the hardware itself which prevents such communication from being used. A new server architecture would simply need to handle multiple units on a singular port (443) and have an additional buffering step to remove excess packing data.

Overall, the work presented has resulted in the creation of a next-generation Frequency Disturbance Recorder which is of lower cost, is more maintainable, and possesses more flexibility in its communication. As the grid evolves and new generation, transmission, protection, and communication are added, phasor measurement units and the FNET system will remain valuable tools in understanding the underlying dynamics of the system. The Generation-3 FDR presented here allows for an expansion of that capability to keep pace with the changing grid.

6.2. Contributions

The major contributions of this research include:

- A survey of existing System-on-a-Chip (SOC) technologies and a determination of their potential for use in synchrophasor measurement devices.
- An analysis of different techniques and patches for emulated real-time operation of the Linux operating system as well as its resulting use in a next-generation FDR.
- A modular hardware scheme which can be applied to new generations of Frequency Disturbance Recorders.
- A modular software scheme to be used in tandem with the hardware design. This software scheme consists of multiple programs and is an example of how multiple processes can allow for continuous real-time operation on a non-RTOS.
- An approximately 10x reduction in the material cost of an FDR.
- Support for multiple communication formats and flexible server assignment including FNET and C37.118 data streams.
- Support for encrypted communications on a synchrophasor measurement device in accordance with modern encryption standards set forth by the IETF.
- An example prototype Generation-3 FDR constructed and integrated with the existing FNET system.

- Analysis of encrypted communication bandwidth overhead and a verification of implementation feasibility on existing FNET hardware.

6.3. Future Work

For future work related to the Generation-3 FDR, there are a number of different research aspects which may be investigated – hardware, software, and applications. While the hardware is very capable on a base level, the device could be modified and further developed to add capability. As mentioned in 3.2.2, the Touch Screen Controller (ADC) has a total of 7 channels and the current design is only utilizing one. The current prototype units could be modified in a fairly straightforward fashion to act as three-phase units, and even allow for current to be measured. However, while the unit presently measures typical voltage harmonics, current signals are noted for having much higher order harmonics – up to 49th order. Therefore, the sampling rate and filtering capability of the device would need adjustment in order to analyze these quantities [91]. This would move the units outside of the current paradigm of FNET, but the device cost would not be significantly increased. Additionally, the use of GPS is not strictly required, and the *ntpd* and *gpsd* programs exist simply to facilitate a PPS signal. The use of alternate timing sources, such as a chip-scale atomic clock [92] could be investigated.

With regards to mobility and convenience, wireless communication as well as wireless signal acquisition are a possibility. The power needs of the current prototype unit allow for a low-power Wi-Fi module to be installed. The difficulty in integrating such a module into the device would be the human-computer interaction required by the host when considering access to secure Wi-Fi hotspots. The standard encryption used in these hotspots would be negotiated by the network device, and in the case of a USB-compliant device, there would be no additional processor overhead. Wireless signal acquisition would entail using an antenna which would couple with an overhead line to directly read the voltage. The PowerIT group has performed research in this area with existing units, and the key advantage offered by the new unit is a much lower overall power draw, which makes this more feasible. For remote locations, a cellular modem could be utilized. Integration of a cellular modem into the existing units is difficult due to the serial communication, but USB support in Linux allows for much easier integration and relatively low processor overhead.

On the software end, the frequency analysis algorithm itself could be modified to calculate additional quantities, especially in conjunction with a multi-phase device, if such hardware research is explored. If harmonic analysis is explored, then the sliding window must be expanded to cover 200ms, or 12 cycles, as specified in IEEE 519 – Recommended Practice and Requirements for Harmonic Control

in Electric Power Systems [91]. The *FDR_Calc* program spends the vast majority of its time idle, and could be using the extra processor cycles for calculation or data archival. New algorithms could be utilized which use-software-based filtering [93]. The current image file for the device is for a 16 GB flash card, and most of this storage capacity remains unused. It would be possible to maintain a local data backup on the device in the event that network connectivity is lost. This oversight could be performed by the local server program – *FDR_Main* – and the calculation algorithm would not have to be modified.

Presently, the new device supports a secure communication structure as well as alternative methods of packet transport, although the FNET servers on the backend do not. As determined in Section 5.3, the overhead required on the part of the server is minimal, and secure communication is possible. The capability of the device to send C37.118 data frames provides a method for some hosts, particularly utilities, to integrate the FDR's readings into their own frameworks in addition to FNET.

Finally, the simple fact that the device is very low costs opens up new avenues of research as more devices may be deployed. This is one of the major reasons for creating the new FDR, but should be mentioned again. Placing an FDR on different circuits of a large building with multi-phase power could provide insight into the load imbalance of that building at different times of the day. This was observed during initial breadboard testing in the FNET lab, but was not further explored. Better understanding and characterization of the load imbalance in a building could lead to improved energy use and lower losses due to unbalanced loading.

It must be mentioned that these concepts are by no means exclusive. As an example, one could create a 3-phase variant of the FDR, connect it to a 480V receptacle in a facility and monitor the load imbalance. This information could then be reported to the owner of the facility as well as the usual frequency measurement data being relayed back to the FNET servers. As this loading information may be considered sensitive data, the communication could be encrypted where necessary and when possible. From a pricing perspective, this additional hardware would add minimal cost while providing much more capability.

List of References

[1] Z. Zhong, C. Xu, B. J. Billian, L. Zhang, S.-J. S. Tsai, R. W. Conners, V. A. Centeno, A. G. Phadke and Y. Liu, "Power System Frequency Monitoring Network (FNET) Implementation," *IEEE Transactions on Power Systems,* vol. 20, no. 4, pp. 1914-1921, 2005.

[2] Y. Zhang, P. Markham, T. Xia, L. Chen, Y. Ye, Z. Wu, Z. Yuan, L. Wang, J. Bank, J. Burgett, R. W. Conners and Y. Liu, "Wide-Area Frequency Monitoring Network (FNET) Architecture and Applications," *IEEE Transactions on Smart Grid,* vol. 1, no. 2, pp. 159-167, 2010.

[3] U.S. Department of Energy, "Factors Affecting PMU Installation Costs," Office of Electricity Delivery & Energy Reliability, Washington, D.C., October 2014.

[4] F. Bedell, "History of A-C Wave Form, Its Determination and Standardization," *Transactions of the American Institute of Electrical Engineers,* vol. 61, no. 12, pp. 864-868, 1942.

[5] C. Xu, Z. Zhong, V. Centeno, R. Conners and Y. Liu, "Practical Issues in Frequency Disturbance Recorder Design for Wide-Area Monitoring," *Electrical power Quality and Utilisation,* vol. XI, no. 1, pp. 69-76, 2005.

[6] G. Missout and P. Girard, "Measurement of Bus Voltage Angle Between Montreal and Sept-Iles," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-99, no. 2, pp. 536-539, 1980.

[7] P. Bonanomi, "Phase Angle Measurements With Synchronized Clocks - Principle and Applications," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-100, no. 12, pp. 5036-5043, 1981.

[8] T. L. Baldwin, L. Mili, M. B. Boisen and R. Adapa, "Power System Observability With Minimal Phasor Measurement Placement," *IEEE Transactions on Power Systems,* vol. 8, no. 2, pp. 707-715, 1993.

[9] A. Phadke, "Synchronized Phasor Measurements - A Historical Overview," in *Transmission and Distribution Conference and Exhibition 2002: Asia Pacific*, 2002.

[10] A. Phadke, "Synchronized Phasor Measurements in Power Systems," *IEEE Computer Applications in Power,* pp. 10-15, 1993.

[11] M. Gardner and Y. Liu, "FNET: A Quickly Deployable and Economic System to Monitor the Electric Grid," in *2007 IEEE Conference on Technologies for Homeland Security*, Woburn, MA, 2007.

[12] "IEEE Standard for Synchrophasors for Power Systems," IEEE 1344-1995(R2001), 1995.

[13] G. Benmouyal, E. O. Schweitzer and A. Guzmán, "Synchronized Phasor Measurement In Protective Relays for Protection, Control, and Analysis of Electric Power Systems," Schewitzer Engineering Laboratories, Inc., Pullman, WA, 2002.

[14] Wikipedia, "Moore's Law," Wikimedia Foundation, [Online]. Available: https://en.wikipedia.org/wiki/Moore%27s_law.

[15] "IEEE Standard for Floating-Point Arithmetic," IEEE 754-2008, 2008.

[16] "IEEE Standard for Synchrophasors for Power Systems," IEEE C37.118-2005, 2005.

[17] "IEEE Standard for Synchrophasor Measurements for Power Systems," IEEE C37.118.1-2011, 2011.

[18] "IEEE Standard for Synchrophasor Data Transfer for Power Systems," IEEE C37.118.2-2011, 2011.

[19] F. C. Schweppe, "Power System Static-State Estimation, Part III: Implementation," *IEEE Transactions on Power Apparatus and Systems,* Vols. PAS-89, no. 1, pp. 130-135, 1970.

[20] R. F. Nuqui, "State Estimation and Voltage Security Monitoring Using Synchronized Phasor Measurements," Blacksburg, VA, 2001.

[21] R. F. Nuqui and A. G. Phadke, "Hybrid Linear State Estimation Utilizing Synchronized Phasor Measurements," in *2007 IEEE Power Tech*, Lausanne, Switzerland, 2007.

[22] J. A. Culliss, "A PMU-Based Method for Reconfigurable Monitoring," Blacksburg, VA, 2009.

[23] F. Aminifar, M. Shahidehpour, M. Fotuhi-Firuzabad and S. Kamalinia, "Power System Dynamic State Estimation With Synchronized Phasor Measurements," *IEEE Tranasactions on Instrumentation and Measurement,* vol. 63, no. 2, pp. 352-363, 2014.

[24] M. Adamiak and R. Hunt, "Application of Phasor Measurement Units for Disturbance Recording," August 2007. [Online]. Available: https://www.gedigitalenergy.com/smartgrid/Aug07/PMU_disturbance_recording.pdf.

[25] R. M. Gardner, "A Wide-Area Perspective on Power System Operation and Dynamics," Blacksburg, VA, 2008.

[26] L. Chen, P. N. Markham, C.-f. Chen and Y. Liu, "Analysis of Societal Event Impacts on the Power System frequency using FNET Measurements," in *2011 IEEE Power and Energy Society General Meeting*, San Diego, CA, 2011.

[27] Y. Liu, Z. Yuan, P. N. Markham, R. W. Conners and Y. Liu, "Application of Power System Frequency for Digital Audio Authentication," *IEEE Transactions on Power Deliverty,* vol. 27, no. 4, pp. 1820-1828, 2012.

[28] P. N. Markham, "Data Mining and Machine Learning Applications of Wide-Area Measurement Data in Electric Power Systems," Knoxville, TN, 2012.

[29] Silicon Labs, "C8051F020/1/2/3," Datasheet, 8K ISP Flash MCU Family, 2003. [Online]. Available: https://www.silabs.com/Support%20Documents/TechnicalDocs/C8051F02x.pdf.

[30] Analog Devices, "Four-Channel, Simultaneous Sampling, Fast, 14-Bit ADC," Datasheet, AD7865, 2000. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD7865.pdf.

[31] Texas Instruments, "TMS320LF2407, TMS320LF2406,TMS320LF2402 DSP Controllers," Datasheet, 2003. [Online]. Available: http://www.ti.com/lit/ds/symlink/tms320lf2407.pdf.

[32] Moxa, "Embedded Device Servers," Datasheet, NE-4100 Series, 2014. [Online]. Available: http://www.moxa.com/doc/specs/NE-4100_Series.pdf.

[33] Motorola, "M12+ GPS Receiver User's Guide," Motorola, 2004. [Online]. Available: http://www.cnssys.com/files/M12+UsersGuide.pdf.

[34] Wikipedia, "NMEA 0183," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/NMEA_0183.

[35] Wikipedia, "ARM Holdings," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/ARM_Holdings.

[36] D. Travlos, "ARM Holdings And Qualcomm: The Winners in Mobile," Forbes, 28 February 2013. [Online]. Available: http://www.forbes.com/sites/darcytravlos/2013/02/28/arm-holdings-and-qualcomm-the-winners-in-mobile/.

[37] K. Bent, "ARM Snags 95 Percept of Smartphone Market, Eyes New Areas For Growth," CRN, 16 July 2012. [Online]. Available: http://www.crn.com/news/components-peripherals/240003811/arm-snags-95-percent-of-smartphone-market-eyes-new-areas-for-growth.htm.

[38] Wikipedia, "Comparison of single-board computers," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_single-board_computers.

[39] pandaboard.org, "PandaBoard Platform," 2014. [Online]. Available: http://www.pandaboard.org/content/platform.

[40] beagleboard.org, "BeagleBoard-xM," 2014. [Online]. Available: http://www.beagleboard.org/beagleboard-xm.

[41] Raspberry Pi Foundation, "What is a Raspberry Pi?," 2014. [Online]. Available: http://www.raspberrypi.org/help/what-is-a-raspberry-pi/.

[42] beagleboard.org, "BeagleBone Black," beagleboard.org, 2014. [Online]. Available: http://www.beagleboard.org/black.

[43] Texas Instruments, "Programmable Real-time Unit Subsystem and Industrial Communication Subsystem (PRU-ICSS)," in *AM335x Sitara™ Processors Technical Reference Manual*, 2014, pp. 197-204.

[44] Texas Instruments, "Touchscreen Controller," in *AM335x Sitara™ Processors Technical Reference manual*, 2014, pp. 1735-1741.

[45] GlobalTop Technology Inc., "FGPMMOPA6H GPS Standalone Module Data Sheet," Datasheet, Revision: V0A, 2011. [Online]. Available: https://www.adafruit.com/datasheets/GlobalTop-FGPMMOPA6H-Datasheet-V0A.pdf.

[46] adafruit, "Home Page," 2014. [Online]. Available: https://www.adafruit.com/.

[47] Recom, "POWERLINE AC/DC-Converter 10 Watt Single / Dual Output," RAC10-xxS_DC/277 Series Datasheet, 2014. [Online]. Available: http://www.recom-power.com/pdf/Powerline-AC-DC/RAC10-S_DC_277.pdf.

[48] elinux.org, "BeagleBone Community," elinux.org, 2014. [Online]. Available: http://elinux.org/BeagleBone_Community.

[49] Littelfuse®, "Cartridge Fuse Blocks," Datasheet, 646 Series Molded Base Fuse Block For 5x20mm Fuses, 2013. [Online]. Available: http://www.littelfuse.com/~/media/electronics/datasheets/fuse_blocks/littelfuse_fuse_block_646_datasheet.pdf.pdf.

[50] "Preferred number series for resistors and capacitors," IEC 60063, 1977.

[51] Texas Instruments, "17 MHz Rail-to-Rail Input-Output Operational Amplifiers," LM6142/6144 Datasheet, 2000-2013. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm6142.pdf.

[52] S. Butterworth, "On the theory of filter amplifiers," *Experimental Wireless and the Wireless Engineer,* vol. 7, pp. 536-541, 1930.

[53] muRata, "Trimmer Potentiometers," Datasheet, Lead Sealed Type Multi-turns PV12/PV37/PV22/PV36 Series, 2011. [Online]. Available: http://media.digikey.com/pdf/Data%20Sheets/Murata%20PDFs/PV12,PV22-23,PV36-37%20Series.pdf.

[54] Bay Area Circuits, "Bay Area Circuits," 2014. [Online]. Available: http://bayareacircuits.com/.

[55] Bay Area Circuits, "Instant DFM," 2014. [Online]. Available: http://instantdfm.bayareacircuits.com/.

[56] Tinsharp Industrial Co., Ltd, "LCM Module TC2004A-01," Datasheet, 2009. [Online]. Available: https://www.adafruit.com/datasheets/TC2004A-01.pdf.

[57] Stackpole Electronics, Inc., "RNF/RNMF Series General Purpose Metal Film Resistor," Datasheet, 2015. [Online]. Available: https://www.seielect.com/Catalog/SEI-RNF_RNMF.pdf.

[58] TDK, "MLCC with Dipped Radial Lead Commercial Grade," Datasheet, General (Up to 50V) FK Series, 2013. [Online]. Available: http://product.tdk.com/capacitor/leadmlcc/en/documents/leadmlcc_commercial_general_en.pdf.

[59] Wikipedia, "Real-time operating system," Wikimedia Foundation, [Online]. Available: https://en.wikipedia.org/wiki/Real-time_operating_system.

[60] kernel.org, "CONFIG PREEMPT RT Patch," [Online]. Available: https://rt.wiki.kernel.org/index.php/CONFIG_PREEMPT_RT_Patch.

[61] Xenomai, "Xenomai," 2014. [Online]. Available: https://xenomai.org/.

[62] Wikipedia, "Xenomai," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/Xenomai.

[63] Real-Time Linux Wiki, "Cyclictest," https://rt.wiki.kernel.org, 29 10 2013. [Online]. Available: https://rt.wiki.kernel.org/index.php/Cyclictest.

[64] K. Koolwal, "Investigating latency effects of the Linux real-time Preemption Patches (PREEMPT_RT) on AMD's GEODE LX Platform," in *Eleventh Real-Time Linux Workshop on September 28 to 30*, Dresden, Germany, 2009.

[65] J. H. Brown and B. Martin, "How fast is fast enough? Choosing between Xenomai and Linux for real-time applications," in *Twelfth Real-Time Linux Workshop on October 25 to 27*, Nairobi, Kenya, 2012.

[66] F. Cerqueira and B. B. Brandenburg, "A Comparison of Scheduling Latency in Linux, PREEMPT_RT, and LITMUS™," Max Planck Institute for Software Systems (MPI-SWS), 2013.

[67] Texas Instruments, AM335x ARM® Cortex™-A8 Microprocessors (MPUs) Technical Reference Manual, 2013.

[68] elinux.org, "BeagleBoardDebian," elinux.org, 2014. [Online]. Available: http://elinux.org/BeagleBoardDebian#BeagleBone.2FBeagleBone_Black.

[69] Device Tree, "Main Page," 2014. [Online]. Available: http://www.devicetree.org/Main_Page.

[70] ntp.org, "NTP: The Network Time Protocol," 2014. [Online]. Available: http://www.ntp.org/.

[71] D. L. Mills, J.Martin, J. Burbank and W. Kasch, "Network Time Protocol Version 4:Protocol and Algorithms Specification," RFC 5905, June 2010. [Online]. Available: https://tools.ietf.org/html/rfc5905.html.

[72] Wikipedia, "Daemon (computing)," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/Daemon_(computing).

[73] E. S. Raymond, "gpsd," 2014. [Online]. Available: http://www.catb.org/gpsd/gpsd.html.

[74] Wikipedia, "Nyquist-Shannon sampling theorem," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem.

[75] Silicon Labs, "Improving ADC Resolution by Oversampling and Averaging," 2013. [Online]. Available: https://www.silabs.com/Support%20Documents/TechnicalDocs/an118.pdf.

[76] Linux Programmer's Manual, "unix(7)," man7.org, 07 05 2015. [Online]. Available: http://man7.org/linux/man-pages/man7/unix.7.html.

[77] Texas Instruments, "PRU Assembly Instructions," Texas Instruments, 2014. [Online]. Available: http://processors.wiki.ti.com/index.php/PRU_Assembly_Instructions.

[78] T. Springer, "Sliding FFT computes frequency spectra in real time," *EDN Mag,* pp. 161-170, 29 Sept 1988.

[79] E. Jacobsen and R. Lyons, "The Sliding DFT," *IEEE Signal Processing Magazine,* vol. 20, no. 2, pp. 74-80, March 2003.

[80] Hitachi, "HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver)," Datasheet, 1999. [Online]. Available: https://www.adafruit.com/datasheets/HD44780.pdf.

[81] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, August 2008. [Online]. Available: https://tools.ietf.org/html/rfc5246.html.

[82] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280.html.

[83] B. Kaliski, "The mathematics of the RSA Public-Key Cryptosystem," RSA Laboratories, 2006. [Online]. Available: http://www.mathaware.org/mam/06/Kaliski.pdf.

[84] OpenSSL Project, "OpenSSL: Cryptography and SSL/TLS Toolkit," [Online]. Available: https://www.openssl.org/.

[85] python.org, "17.3 ssl - TLS/SSL wrapper for socket objects," python.org, Oct 2008. [Online]. Available: https://docs.python.org/2/library/ssl.html.

[86] Wikipedia, "Certificate authority," Wikimedia Foundation, 2014. [Online]. Available: https://en.wikipedia.org/wiki/Certificate_authority.

[87] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management," RFC 1422, February 1993. [Online]. Available: https://tools.ietf.org/html/rfc1422.html.

[88] Z. Huang, B. Kasztenny, V. Madani, K. Martin, S. Meliopoulos, D. Novosel and J. Stenbakken, "Performance Evaluation of Phasor Measurement Systems," in *IEEE Power Engineering Society General Meeting*, Pittsburgh, PA, 2008.

[89] K. E. Martin, "Synchrophasor Standards Development - IEEE C37.118 & IEC 61850," in *Proceedings of the 44th Hawaii International Conference on System Sciences*, Honolulu, HI, 2011.

[90] Y. Lei, Y. Zhang, J. Guo, D. Zhou, J. Culliss, P. Irminger and Y. Liu, "The Impact of Synchronized Human Activities on Power System Frequency," in *2014 IEEE PES General Meeting | Conference and Exposition*, National Harbor, MD, 2014.

[91] "IEEE Recommended Practice and Requirements for Harmonic Control in Electric Power Systems," IEEE 519-2014, 2014.

[92] Microsemi, "Quantum™ SA.45s Chip Scale Atomic Clock," Microsemi Corporation, 2015. [Online]. Available: http://www.microsemi.com/products/timing-synchronization-systems/embedded-timing-solutions/components/sa-45s-chip-scale-atomic-clock#overview.

[93] A. Ouadi, H. Bentarzi and J. C. Maun, "Improvement of Phasor Measurement Unit Performance," in *Recent Researches in Circuits, Systems, Electronics, Control & signal Processing*, Athens, Greece, WSEAS Press, 2006, pp. 40-44.

Appendices

Appendix A Source Code

FNET-GPS-00A0.dts:

```
/*
 * Copyright (C) 2013-2015 PowerIT Lab, University of Tennessee
 *
 * FNET GPS overlay for UART4 on connector pins P9.11 P9.13 P9.15
 *
 * This program is free software; you can redistribute it and/or
modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 */
/dts-v1/;
/plugin/;

/ {
      compatible = "ti,beaglebone", "ti,beaglebone-black";

      /* identification */
      part-number = "FNET-GPS";
      version = "00A0";

      /* state the resources this cape uses */
      exclusive-use =
            /* the pin header uses */
            "P9.11",        /* uart4_rxd */
            "P9.13",        /* uart4_txd */
            "P9.15",        /* gpio1_16  */
            /* the hardware ip uses */
            "uart4",
            "gpio1_16";

      fragment@0 {
            target = <&am33xx_pinmux>;
            __overlay__ {

                  gps_uart4_pins: pinmux_gps_uart4_pins {
                        pinctrl-single,pins = <
                              0x70 0x26 /* P9.11 uart4_rxd.uart4_rxd
MODE6 INPUT (RX) */
                              0x74 0x26 /* P9.13 uart4_txd.uart4_rxd
MODE6 OUTPUT (TX) */
                              >;
                  };

                  gps_pps_pins: pinmux_gps_pps_pins {
                        pinctrl-single,pins = <
                              0x40 0x27 /* P9.15 gpio1_16 MODE7 */
                        >;
```

```
                };
        };
    };

    fragment@1 {
        target = <&uart5>;    /* really uart4 */
        __overlay__ {
            status = "okay";
            pinctrl-names = "default";
            pinctrl-0 = <&gps_uart4_pins>;
        };
    };

    fragment@2 {
        target = <&ocp>;
        __overlay__ {
            pps {
                compatible = "pps-gpio";
                status = "okay";
                pinctrl-names = "default";
                pinctrl-0 = <&gps_pps_pins>;
                gpios = <&gpio2 16 0 >;
                assert-rising-edge;
            };
        };
    };
};
```

FNET-PRU-00A0.dts:

```
/*
* Copyright (C) 2013-2015 PowerIT Lab, University of Tennessee
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation.
*/
/dts-v1/;
/plugin/;

/ {
      compatible = "ti,beaglebone", "ti,beaglebone-black";

      /* identification */
      part-number = "FNET-PRU";
      version = "00A0";

      /* state the resources this cape uses */
      exclusive-use =
            /* the pin header uses */
            "P9.27",   /* pru0: pr1_pru0_pru_r30_5 */
            "P9.31",   /* pru0: pr1_pru0_pru_r30_0 */
            /* the hardware IP uses */
            "pru0";

      fragment@0 {
            target = <&am33xx_pinmux>;
            __overlay__ {

                  pru_gpio_pins: pinmux_pru_gpio_pins {
                        pinctrl-single,pins = <
                              0x1a4 0x0f      /* P9 27 GPIO3_19:
mcasp0_fsr.gpio3[19] | MODE7 | OUTPUT */
                        >;
                  };

                  pru_pru_pins: pinmux_pru_pru_pins {
                        pinctrl-single,pins = <
                              0x1a4 0x25 /*
mcasp0_fsr.pr1_pru0_pru_r30_5, MODE5 | OUTPUT | PRU */
                              0x190 0x26 /*
mcasp0_aclkx.pr1_pru0_pru_r31_0, MODE6 | INPUT | PRU */
                        >;
                  };
            };
      };

      fragment@2 {
            target = <&pruss>;
```

```
            __overlay__ {
                  status = "okay";
                  pinctrl-names = "default";
                  pinctrl-0 = <&pru_pru_pins>;
            };
      };
};
```

CRC_C37.py:

```
#!/usr/bin/python
#
# CRC_C37.py
# This is a CRC-CCITT library for use with C37.118 packets. It is a
Python
# port of the C library available at:
#   http://www.lammertbies.nl/comm/info/crc-calculation.html
#
# 2014
# jculliss
#

# Standard CRC-CCITT Polynomial
POLYNOMIAL = 0x1021

# Preset Value of 0xFFFF or -1; this is C37.118.2-2011 specified.
PRESET = -1

def _initial(c):
     crc = 0
     c = c << 8
     for j in range(8):
          if (crc ^ c) & 0x8000:
               crc = (crc << 1) ^ POLYNOMIAL
          else:
               crc = crc << 1
          c = c << 1
     return crc

_tab = [ _initial(i) for i in range(256) ]

def _update_crc(crc, c):
     cc = 0xff & c

     tmp = (crc >> 8) ^ cc
     crc = (crc << 8) ^ _tab[tmp & 0xff]
     crc = crc & 0xffff

     return crc

def crc(str):
     crc = PRESET
     for c in str:
          crc = _update_crc(crc, c)
     return crc

def crcb(*i):
     crc = PRESET
     for c in i:
```

```
        crc = _update_crc(crc, c)
    return crc
```

rc.local:

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Add device tree overlays to cape manager
/bin/echo FNET-GPS > /sys/devices/bone_capemgr.9/slots
/bin/echo FNET-PRU > /sys/devices/bone_capemgr.9/slots
/bin/echo BB-ADC > /sys/devices/bone_capemgr.9/slots

# Execute FDR Initialization Script
sudo /home/fnet3/runtime/FDR_Initialize.sh

exit 0
```

FDR_Initialize.sh:

```
#!/bin/bash -e
#

# Ensure freshly started ntp and gpsd daemons
/etc/init.d/ntp stop
/etc/init.d/ntp start
/etc/init.d/gpsd stop
/etc/init.d/gpsd start

# Change working directory
cd $(dirname $(readlink -f $0))

# Wait for daemons to start
sleep 1

# Execute communication sub-process
python FDR_Comm.py &

# Execute main process
python FDR_Main.py &

# Wait for main process to start
sleep 5

# Execute calculation sub-process
./FDR_Calc

exit 0
```

unit_config:

```
# Unit ID
9001
# Nominal Frequency
60
# Nominal Voltage
120
# Voltage Scaling Constant
27200
```

server_config:

```
#Type Address               Port  Encryption
fnet  192.xxx.xxx.11   17501 false
fnet  192.128.137.1    9930  false
```

Vita

Jerel Culliss is a native of Clifton Forge, Virginia. He received his B.S. and M.S. in Electrical Engineering (Power Systems) from Virginia Polytechnic Institute and State University in 2006 and 2009, respectively. His research interests include applications of wide-area measurements and information technology as applied to power systems, cyber security of embedded devices, and power systems modeling. He is a Distinguished Graduate Fellow in the Department of Electrical Engineering and Computer Science at the University of Tennessee, and a member of both the IEEE and the Power & Energy Society.