Open Source Implementation of PMU Connection Tester in Linux

An Internship Project By

Faizan Bhat

Under the guidance of

Prof. A. M. Kulkarni

DEPARTMENT OF ELECTRICAL ENGINEERING
Indian Institute of Technology Bombay
Mumbai -400076, Maharashtra, India

February 2020

Acknowledgement

Working in the Electrical Department at IIT Bombay was interesting. During these two months of internship I learnt a lot on application development. I specially thank Prof. Anil Kulkarni for supervising and guiding me throughout. I am very much grateful to Mr. Gnana Vinayagan for mentoring me despite his busy schedule. Further I want to thank Mr. Santosh V. Singh for helping me to make this project happen.

Dedicated to my Sister

Abstract

PMU connection tester is a free software available in Windows operating system, which facilitates visualizing the live data received from a PMU or PDC. In this project an open source implementation of *PMU connection tester* in Linux is carried out. As an end product in its first version *iTester* package is developed using C++ programming language. The package developed uses the synchrophasor standard (IEEE C37.118) to parse the data received from PMU or PDC. The role of PMU and PDC are simulated using an open source package *iPDC*. The capability of the *iTester* is to handle one PMU at a time with exactly three numbers of phasors and two numbers of analog data. This report summarizes the synchrophasor standard briefly with some illustrative examples of how the data frames and configuration frames are formulated. Later the developmental stages of *iTester* package is discussed, followed by the package's user manual.

Contents

Chapter 1

Introduction to *iTester*

1.1 Overview of Synchrophasor Technology

High voltage power system networks, topologically spans huge geographical covers which is primarily AC (sinusoids of fixed frequency i.e. 50 Hz nominal in Indian grid). Each node or terminal in these network are referred to as bus. Each bus is connected to other bus through a transmission line which can even be high voltage DC line or short cable. The quantities in these network i.e. voltage and current are time varying. The energy or power transferred from one bus to other bus over a transmission line depends on the phase angle differences between the bus voltages, magnitudes of the voltages and the line impedance between the two buses. In order to ensure smooth functioning of the power system in respect to protection, stability and reliability; monitoring of the network quantities in wide area is of utmost importance.

Earlier wide area measurements were carried by supervisory control and data acquisition (SCADA) system. SCADA system sequentially collects data from different remote terminal units (RTU) placed at specific buses [1]. These data were held for certain time (e.g. 15 min interval in Indian context which corresponds to 96 blocks of measurement on daily basis). Therefore the analytics which are used with SCADA uses only scalar quanties like voltage magnitudes, active and reactive power flow in lines etc. Using these scalar measurements system operator's can back calculate the voltage phasors at various nodes and act accordingly.
Given a bus, phasors can be obtained locally but cannot be related to phasors at other buses unless the time reference is globally common to the network of interest. This time reference can be obtained from global positioning system (GPS) or network time protocol (NTP) or similar derived clocks. This is infact synchrophasor technology in action. Primarily it refers to phasor calculated at buses with a common time reference or in synchronized fashion.

Phasor measurement unit (PMU) is a device used to estimate the voltage and current phasors. Synchrophasors are measured by fast time-stamped devices called phasor measurement units (PMUs) to constitute the basis of real-time monitoring and control actions in the electric grid. These applications include wide-area situational awareness and monitoring, state estimation, fault location and protective relaying, islanding detection etc. PMU measures voltage and current and computes phasor values and send it to the phasor data concentrator (PDC) through internet using IEEE C37.118.2 protocol [3], [4] (standard for synchrophasor data transfer).

The application and role of Phasor Measurement Units and Phasor Data Concentrators are mimiced through computer based applications that are as, PMU Simulator and iPDC. PMU Simulator and iPDC were actually created at IIT Bombay by two students Nitesh Pandit and Kedar V. Khandeparkar [2]. PMU Simulator is a Software for Phasor Measurement Unit. iPDC acts as a server or a client depending upon the requirement and mode of operation. PMU Simulator generates the data automatically or through the experimental recorded `*.csv` files. The data contains the information such as, phasor (magnitude and angle), time stamp, etc.

PMU Connection Tester or iTester plays following roles:

(i) Receive data from the PMU (in here PMU Simulator software)

(ii) Parsing the received configuration and data frames

(iii) Displaying the parsed data in hex packets and human readable form

(iv) Visualizing live and realtime plots of the various quantities parsed from the data frame.

1.2 Literature Survey

1.2.1 PMU Simulator

PMU Simulator is a simulation software for Phasor Measurement Unit. It acts as a server and bind ports for UDP and TCP communication protocol. It listens for UDP connections on UDPPORT and for TCP connections on TCPPORT. The PMU Simulator receives the command frames from PDC and sends the configuration frame and data frames to PDC. It has option to configure PMU by adding PMU ID, phasor, analog, and digital channels, data rate, data format, etc. A single PMU Simulator can have multiple PDC client communicating through TCP and UDP. It sends the configuration frame first once connected to the iPDC then continously transmits a stream of data frames to the iPDC. It either transmits the autogenerated data file or preloaded experimental files `(*.csv)` depending upon what a user chooses.

1.2.2 iPDC

iPDC's act as servers when they are communicating with another iPDC whom they are sending the data/configuration frames. The iPDC receiving data in this case acts as a client. However when the same iPDC sends data to another iPDC, it would act as a server. This pattern will be repeated in the WAMS topology with one peer acting as server and its counterpart a client. The iPDC when acting as a server binds to 2 ports UDPPORT and TCPPORT. It would be listening for UDP connections on UDPPORT and TCP connections on TCP- PORT. The iPDC can then send the combined configuration frames to any number of other iPDC's. Both the communicating peers authenticate each other. iPDC authenti- cates for each received packets irrespective of communcation protocols used(TCP/UDP). When the iPDC starts for the first time the user is prompted to enter iPDC Idcode, UDP Port, TCP Port and Database Server IP. The ports enable iPDC to receive requests from other iPDC and to send the combined data and configuration frames. Database Server IP is the IP address of the machine where the process dbserver is running. The default port on which dbserver is listening for data is 9000 and it is a UDP server. The data which the iPDC receives would also be directed to dbserver for storage in MySQL database. The user is provided with the following options at iPDC Enter iPDC Setup Add a Source Device Remove a Source Device Turn OFF the data Transmission Turn ON the data Transmission Request Configuration frame Add a Destination Device Remove a Destination Device iPDC Connection Table

1.2.3 DBServer

DBServer is the database server module for iPDC. It would run as individual process on local or remote machine. Among the various known open source databases, MySQL has been used for storing the data at the iPDC. The process would have a parser to parse configuration and data frames. After parsing, configuration and data frames entries would be stored in the iPDC MySQL database. The data frames are inserted as they come. This data which is stored in the tables can then be used for later analysis. The data from the database is archived periodically. It will also creating the measurement files in `*.csv` format.

Chapter 2

Revisiting C37.118.2-2011 IEEE Standard

A PMU or PDC may transmit its data in one or more separate data streams. Each stream may have different contents and may be sent at a different rate [4]. The destination of each stream may be a different device and location. Each stream must then be individually controllable, have its own IDCODE, and a separation configuration control.There is actually a format of messages which are sent to and from a PMU or PDC for use in real-time communication of phasor data. Real-time data transmission here is defined as taking place concurrently with the measurement process. The message frames shall be transmitted in their entirety as they are specified. This message protocol may be used for communication with a single PMU or a secondary system that receives data from several PMUs.The protocol allows for necessary identifying information, such as the PMU IDCODE and status, to be embedded in the data frame for proper interpretation of the measured data.

2.1 Message framework

Four message types are defined in this standard: data, configuration, header, and command. The first three message types are transmitted from the PMU/PDC that serves as the data source, and the last (command) is received by the PMU/PDC.

Data messages are the measurements made by a PMU. Configuration is a machine-readable message describing the data types, calibration factors, and other meta-data for the data that the PMU/PDC sends. Header information is human readable descriptive information sent from the PMU/PDC but provided by the user. Commands are machine-readable codes sent to the PMU/PDC for control or configuration. A PMU or PDC may transmit multiple data streams, each with different content, rate, format, etc. Each data stream shall have its own IDCODE so that the data, configu-

ration, header, and command messages can be appropriately identified. Each stream shall be independently operable including command execution and data, header, and configuration messages.

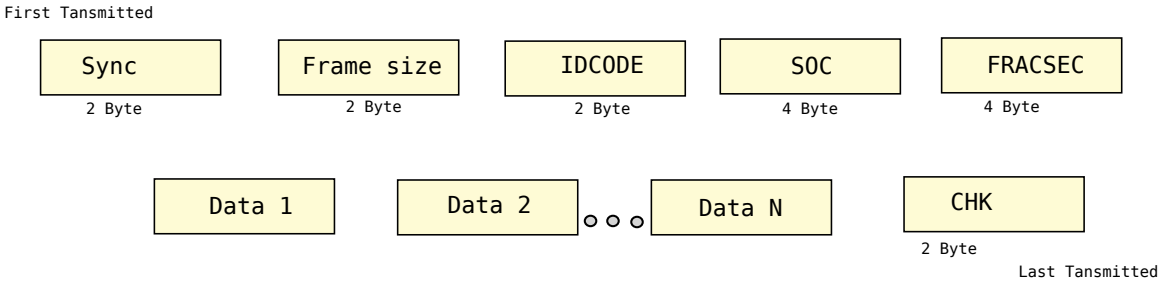


Figure 2.1: General Message Format

2.1.1 Data Frame

A data frame shall contain measured data and shall be identified by having bits 4–6 in the SYNC word set to zero. All fields shall be fixed length as described and no delimiters shall be used. The frame starts with SYNC, FRAMESIZE, IDCODE, SOC and FRACSEC. Refer Figure 2.2 for Data Frame organisation.

2.1.2 Configuration Frame

A configuration frame is a machine-readable BINARY data set containing information and processing parameters for a synchrophasor data stream. Refer the Table 2.1 below for Configuration Frame organisation.

Field	Size (bytes)	Comments
SYNC	2	First byte: AA hex Second byte: 01 hex
STAT	2	Bit mapped flags. Bit 15–14: Data error: 00 = good measurement data, no errors 01 = PMU error. No information about data 10 = PMU in test mode (do not use values) or absent data tags have been inserted (do not use values) 11 = PMU error (do not use values) Bit 13: PMU sync, 0 when in sync with a UTC traceable time source Bit 12: Data sorting, 0 by time stamp, 1 by arrival Bit 11: PMU trigger detected, 0 when no trigger Bit 10: Configuration change, set to 1 for 1 min to advise configuration will change, and clear to 0 when change effected. Bit 09: Data modified, 1 if data modified by post processing, 0 otherwise Bits 08–06: PMU Time Quality. Refer to codes in Table 7. Bits 05–04: Unlocked time: 00 = sync locked or unlocked < 10 s (best quality) \01 = 10 s ≤ unlocked time < 100 s 10 = 100 s < unlock time ≤ 1000 s 11 = unlocked time > 1000 s Bits 03–00: Trigger reason: 1111–1000: Available for user definition 0111: Digital 0110: Reserved 0101: df/dt High 0100: Frequency high or low 0011: Phase angle diff 0010: Magnitude high 0001: Magnitude low 0000: Manual
PHASORS	4 / 8	16-bit integer values: Rectangular format: —real and imaginary, real value first —16-bit signed integers, range –32 767 to +32 767 Polar format: —magnitude and angle, magnitude first —magnitude 16-bit unsigned integer range 0 to 65535 —angle 16-bit signed integer, in radians × 10 4 , range –31 416 to +31 416 32-bit values in IEEE floating-point format: Rectangular format: —real and imaginary, in engineering units, real value first Polar format: —magnitude and angle, magnitude first and in engineering units —angle in radians, range –π to + π
FREQ	2 / 4	Frequency deviation from nominal, in mHz Range–nominal (50 Hz or 60 Hz) –32.767 to +32.767 Hz 16-bit integer or 32-bit floating point 16-bit integer: 16-bit signed integers, range –32 767 to +32 767 32-bit floating point: actual frequency value in IEEE floating-point format.
DFREQ	2 / 4	ROCOF, in hertz per second times 100 Range –327.67 to +327.67 Hz per second Can be 16-bit integer or IEEE floating point, same as FREQ above.
ANALOG	2 / 4	Analog word. 16-bit integer. It could be sampled data such as control signal or transducer value. Values and ranges defined by user. Can be 16-bit integer or IEEE floating point.
DIGITAL	2	Digital status word. It could be bit mapped status or flag. Values and ranges defined by user.

Figure 2.2: Data Frame Organisation

Table 2.1: Configuration frame organisation.

Field	**Size (bytes)**	**Short Description**
SYNC	2	Sync byte followed by frame type and version number.
FRAMESIZE	2	Number of bytes in frame.
IDCODE	2	Stream source ID number, 16-bit integer.
SOC	4	SOC time stamp.
FRACSEC	4	Fraction of Second and Message Time Quality.
TIME_BASE	4	Resolution of FRACSEC time stamp.
NUM_PMU	2	The number of PMUs included in the data frame.
STN	16	Station Name—16 bytes in ASCII format.
IDCODE	2	Data source ID number identifies source of each data block.
FORMAT	2	Data format within the data frame.
PHNMR	2	Number of phasors—2-byte integer (0 to 32 767).
ANNMR	2	Number of analog values—2-byte integer
DGNMR	2	Number of digital status words—2-byte integer.
CHNAM	16×(PHNMR+ANNMR+ 16×DGNMR)	Phasor and channel names—16 bytes for each phasor, analog, and each digital channel (16 channels in each digital word) in ASCII format in the same order as they are transmitted. For digital channels, the channel name order will be from the least significant to the most significant. (The firstname is for bit 0 of the first 16-bit status word, the second is for bit 1, etc.,up to bit 15. If there is more than 1 digital status, the next name will applyto bit 0 of the second word and so on.)
PHUNIT	4 x PHNMR	Conversion factor for phasor channels.
ANUNIT	4 x ANNMR	Conversion factor for analog channels.
DIGUNIT	4 x DGNMR	Mask words for digital status words.
FNOM	2	Nominal line frequency code and flags.
CFGCNT	4	Configuration change count.
DATA_RATE	2	Rate of data transmissions.
CHK	2	CRC-CCITT.

2.2 Wireshark Capture of a Data Frame

Data frame starts with i^{th} SYNC, FRAMESIZE, IDCODE, SOC, FRACSEC. To identify the hex values which corresponds to these parameters refer the diagram below.

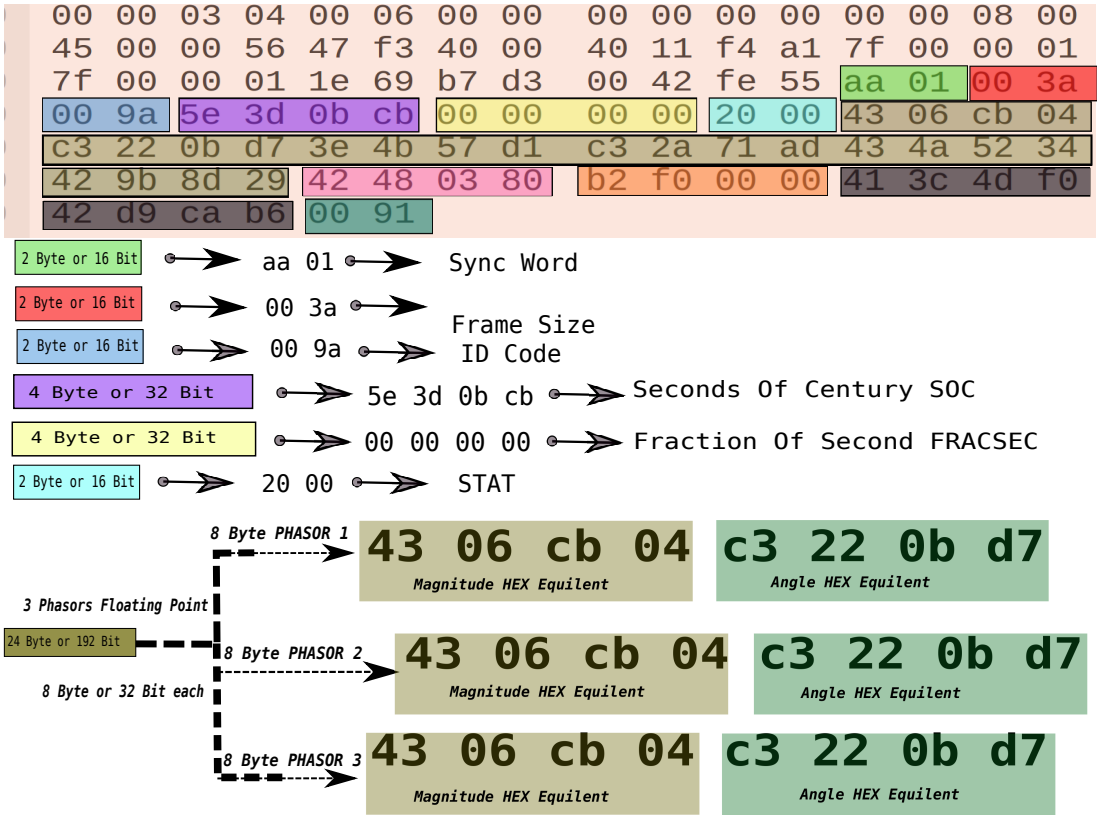


Figure 2.3: Data frame Capture in Wireshark

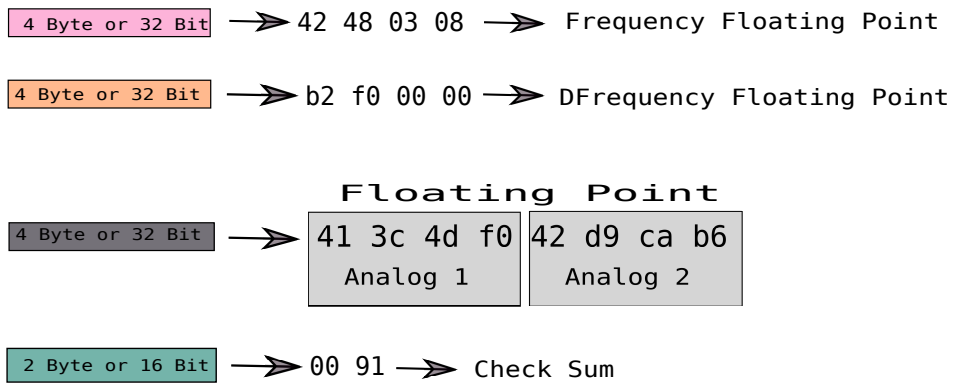
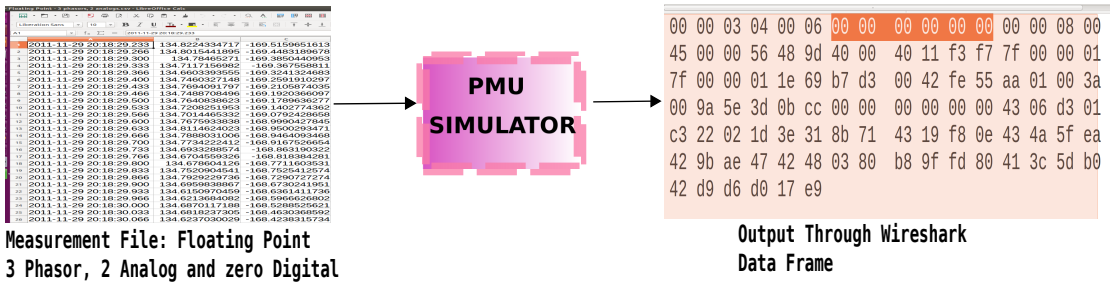


Figure 2.4

2.3 Illustrative Examples

Considering a specific case of 3 phasor (in polar form) and 2 analog data expressed in floating point representation. The values of various electrical parameters like magnitude, angle, analog values etc are transferred into proper hex packets. Those hex values can be converted into decimals, floating points, ASCII values and so on. The measurement file determines whether the obtained values are scaled versions or not, this all depends on what kind of input format has been chosen. For fixed type format the value of the conversion factor is given in the configuration frame. For floating type format the value of the conversion factor is equal to 1. In floating type format the input data is same as retrieved one.

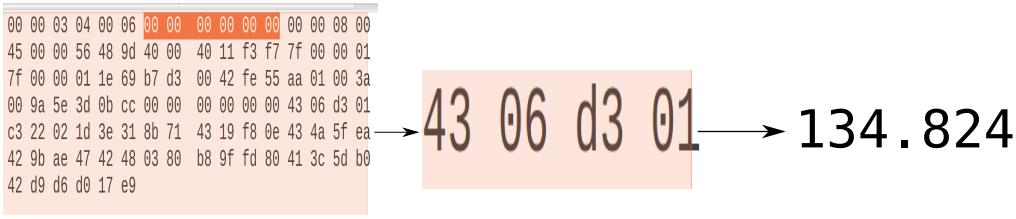
Example 1 : This example defines where a phasor data such as phasor magnitude is situated in a string of hex data packet received. The input measurement file fed to the PMU simulator is 3 phasor, 2 analog and zero digital (floating point) in polar format. The Hex magnitude values are converted into floating format so to get the desired values. In this example the scaling factor is equal to 1 therefore conversion factor is also equal to 1, hence the input data is same as the output retrieved.



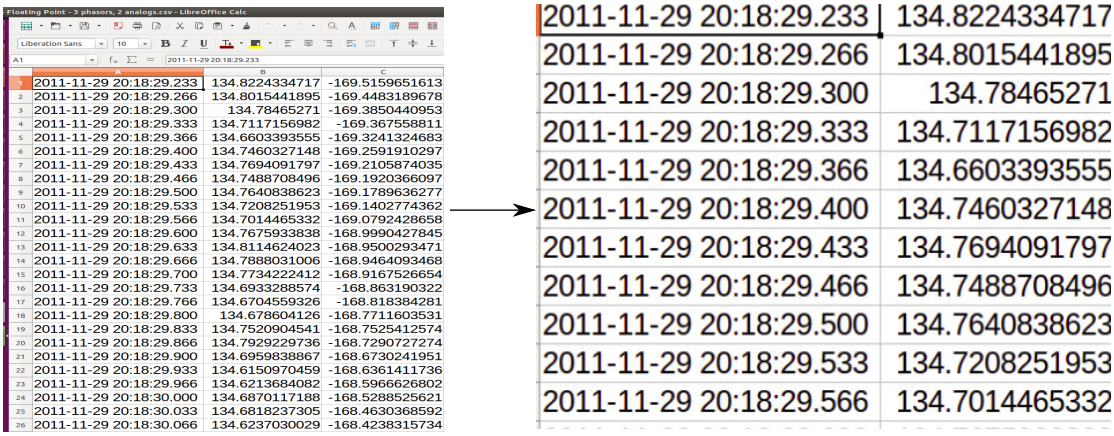
Measurement File: Floating Point
3 Phasor, 2 Analog and zero Digital

Output Through Wireshark
Data Frame

- Consider a case in which the measurement file choosen is 3 Phasor, 2 Analog and zero Digital in Polar Representation Format.
- The output of the PMU Simulator is visualized through Wireshark which shows a Hexadecimal representation of Data Frame received.



Wireshark capture of Data Frame **Phasor Magnitude in Hexadecimal Equilent** **Floating Point Equilent**



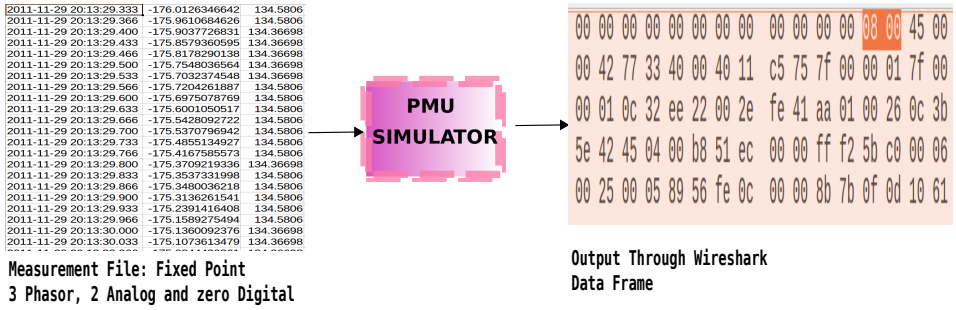
1	2011-11-29 20:18:29.233	134.8224334717
2	2011-11-29 20:18:29.266	134.8015441895
3	2011-11-29 20:18:29.300	134.78465271
4	2011-11-29 20:18:29.333	134.7117156982
5	2011-11-29 20:18:29.366	134.6603393555
6	2011-11-29 20:18:29.400	134.7460327148
7	2011-11-29 20:18:29.433	134.7694091797
8	2011-11-29 20:18:29.466	134.7488708496
9	2011-11-29 20:18:29.500	134.7640838623
10	2011-11-29 20:18:29.533	134.7208251953
11	2011-11-29 20:18:29.566	134.7014465332
12	2011-11-29 20:18:29.600	134.7675933838
13	2011-11-29 20:18:29.633	134.8114624023
14	2011-11-29 20:18:29.666	134.7888031006
15	2011-11-29 20:18:29.700	134.7734222412
16	2011-11-29 20:18:29.733	134.6933288574
17	2011-11-29 20:18:29.766	134.6704559326
18	2011-11-29 20:18:29.800	134.678604126
19	2011-11-29 20:18:29.833	134.7520904541
20	2011-11-29 20:18:29.866	134.7929229736
21	2011-11-29 20:18:29.900	134.6959838867
22	2011-11-29 20:18:29.933	134.6150970459
23	2011-11-29 20:18:29.966	134.6213684082
24	2011-11-29 20:18:30.000	134.6870117188
25	2011-11-29 20:18:30.033	134.6818237305
26	2011-11-29 20:18:30.066	134.6237030029

2011-11-29 20:18:29.233	134.8224334717
2011-11-29 20:18:29.266	134.8015441895
2011-11-29 20:18:29.300	134.78465271
2011-11-29 20:18:29.333	134.7117156982
2011-11-29 20:18:29.366	134.6603393555
2011-11-29 20:18:29.400	134.7460327148
2011-11-29 20:18:29.433	134.7694091797
2011-11-29 20:18:29.466	134.7488708496
2011-11-29 20:18:29.500	134.7640838623
2011-11-29 20:18:29.533	134.7208251953
2011-11-29 20:18:29.566	134.7014465332

- The Floating Equilent of Hexdecimal Phasor magnitude is approximately equal to magnitude values present in the measurement file which was fed to the PMU Sim. as an input.

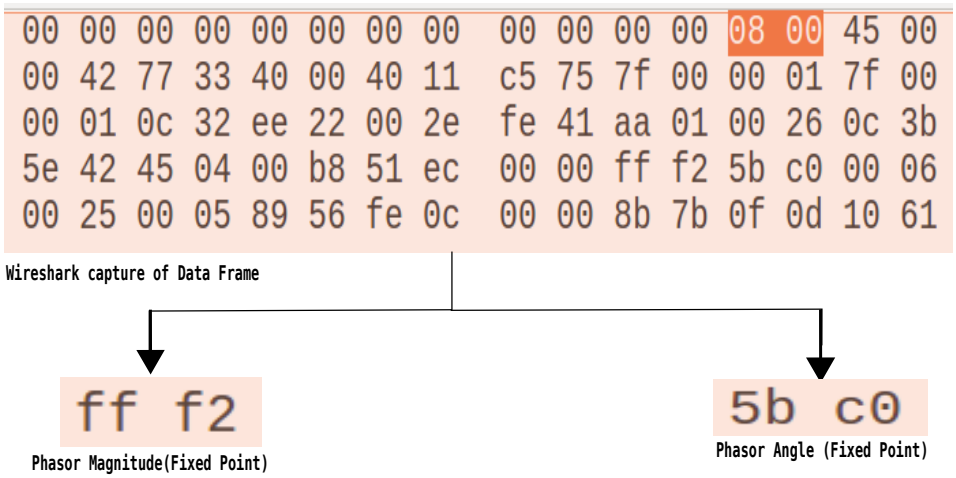
Figure 2.5: Example 1

Example 2 : This example defines where a Phasor data such as Phasor magnitude, the phase angles can be located in a string of hex data packet received. The input measurement file fed to the PMU simulator is 3 phasors (in polar form) and 2 analogs in **fixed point representation**. The hexadecimal values are converted into floating format so as to get the desired values. In this example the scaling factor is read from configuration frame organisation table. The maximum input values set for 3 phasors and 2 analogs are 400 kV and 100 RMS. The corresponding conversion factors so generated for these set of values are 1220703e-5 (for magnitude) and $\frac{180}{\pi} \times 10^4$. The value so obtained is approximately equal to the value present in the measurement file.



Consider a case in which the measurement file choosen is 3 Phasor, 2 Analog and zero Digital in Polar Representation, Fixed Point Format.

The output of the PMU Simulator is visualized through Wireshark which shows a Hexadecimal representation of Data Frame received.



FFF2 (HEXADECIMAL) ⟶ 65522 (DECIMAL EQUIVALENT)

65522-65536 = -14

-14 × 12.20703 (*Phasor Magnitude conversion factor determined through Configuration Frame*)

= 170.89842 (*Calculated Phasor Mangitude value from Data Frame*).

5BC0 (HEXADECIMAL) ⟶ 23488 (DECIMAL EQUIVALENT)

23488 × 180/(pi) (*Phasor Angle conversion factor determined through Configuration Frame*)

= 134.644586 (*Calculated Phasor Angle value from Data Frame*).

The Calculated values of Phasor Magnitude and Angle are approximately equal to the values present in the measurement file which is fed as an input to the PMU Sim.

Figure 2.6: Example 2

Chapter 3

iTester

- The alphabet 'i' in iTester refers to IIT BOMBAY.
- iTester is basically a Linux based PMU connection tester.
- iTester has been created to generate live plots of the data coming from PMU simulators. It also displays the raw hex data packets and their respective decoded values in human readable form
- It gives live plots of various parameters such as Frequency, Magnitude, Angle and Analogs with respect to time.

iTester has been created in Qt. It has been coded in C++ platform. From the coding point of view iTester actually fetches the raw data from PMU simulator and stores it into a string. The data stored into string is then parsed accordingly. Parsing process is completely done by considering the protocol IEEE Std C37.118.2-2011 for Synchrophasor Data Transfer for Power Systems.

Features : iTester has a very simple and self explanatory user interface. The user interface of iTester comprises of a Digital clock fixed at the top right extreme of the window. There are four kinds of plotting areas in the iTester application. Each plot has its own essence. Plots of frequency, magnitude, angle, analogs with respect to time can be obtained by virtue of this application. Five types of buttons are present there. Each button has its own unique functionality. Description of various buttons is given below:

- Connect button: connects to the port address entered.
- View configuration frame button: Displays the Configuration Frame.
- Rescale Button: rescales the window to according to the intercept values of max and min.

- Preset button: resizes the window to its original size

- Exit button: closes the application.

iTester supports an interactive way of visualizing a live plot through zooming. Zoom in and zoom out feature enables a user to get a better interpretation of a live plotted data.It has a feature of displaying the packets received into hexadecimal format. iTester has a separate window for displaying the complete information of a data frame into proper human readable form.

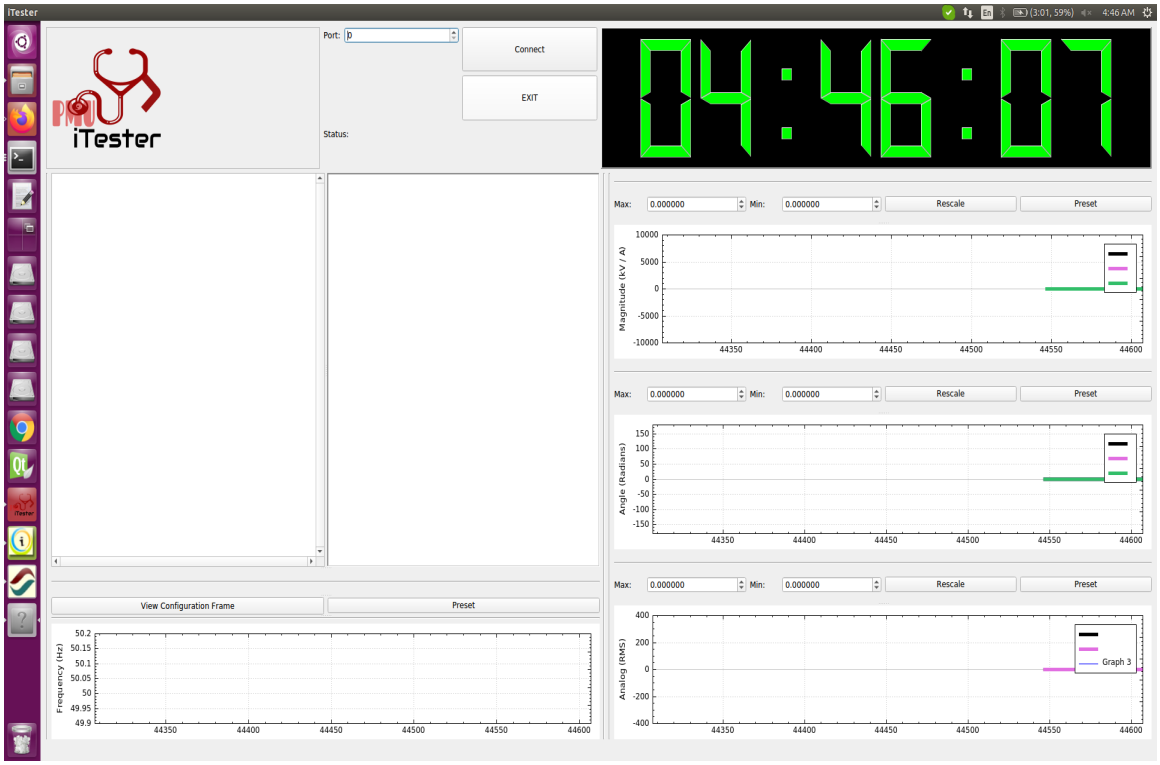


Figure 3.1: ScreenShot of iTester

3.1 Interfacing iTester with PMU and iPDC

Certain sequence of steps has to be followed in order to interface iTester with PMU and iPDC.

- Open PMU Simulator and configure it.

Figure 3.2: ScreenShot of PMU Simulator

- Open iTester and put it on listening mode by entering the address of DBServer that is 9000.

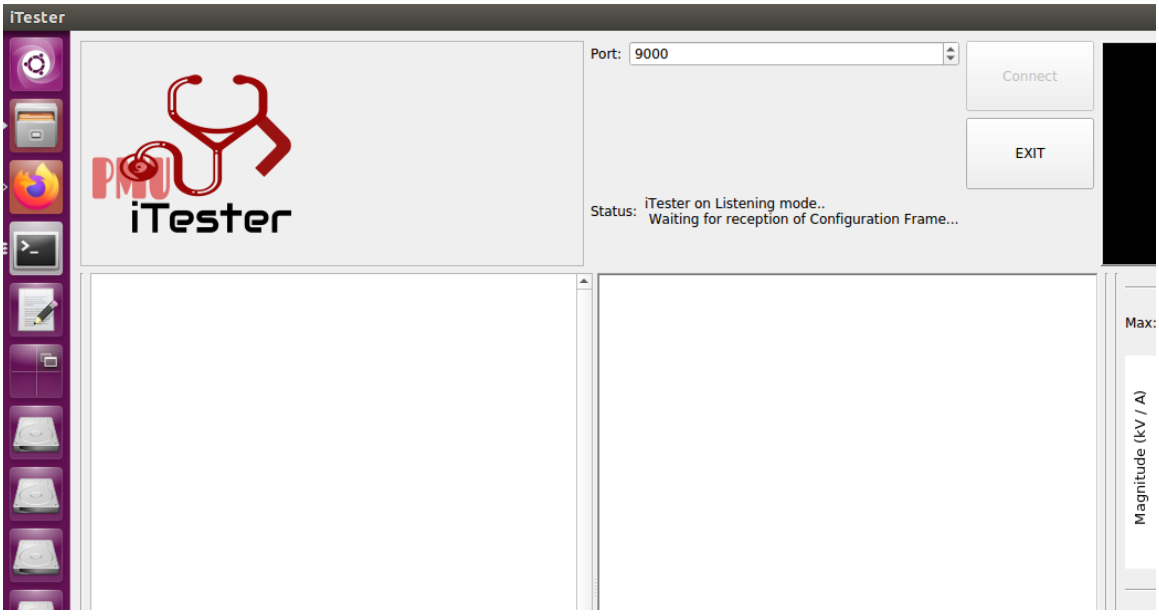


Figure 3.3: iTester in Connecting mode

- Open iPDC and load the PMU profile into it so that it can start streaming the data and iTester would be able to receive it.

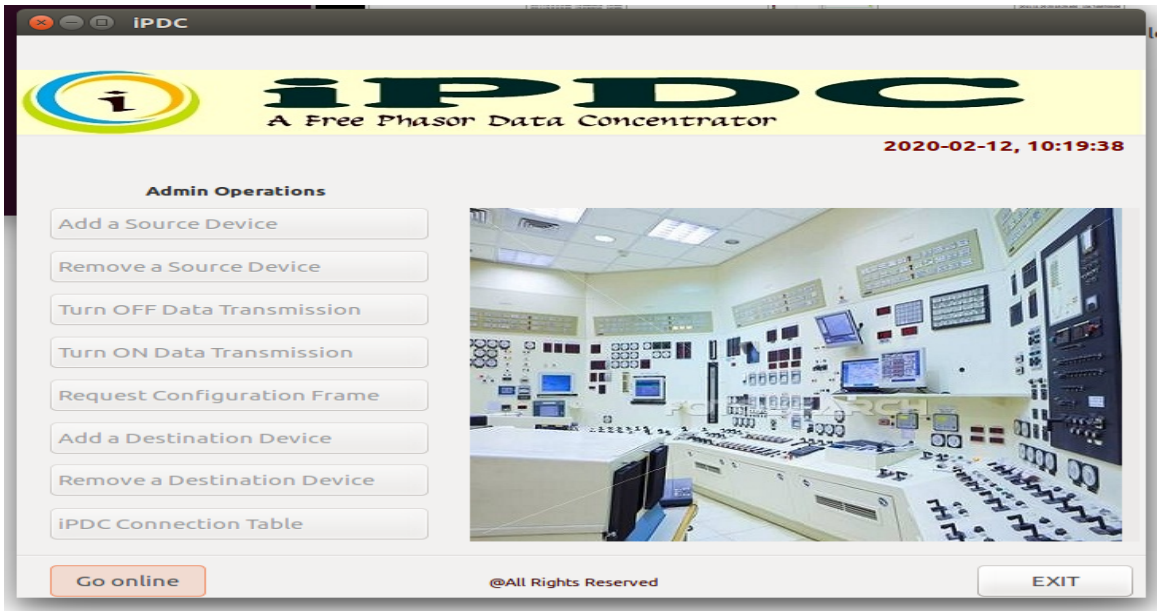


Figure 3.4: ScreenShot of iPDC

- Once the configuration frame is received iTester starts displaying the data as plotting it simultaneously.

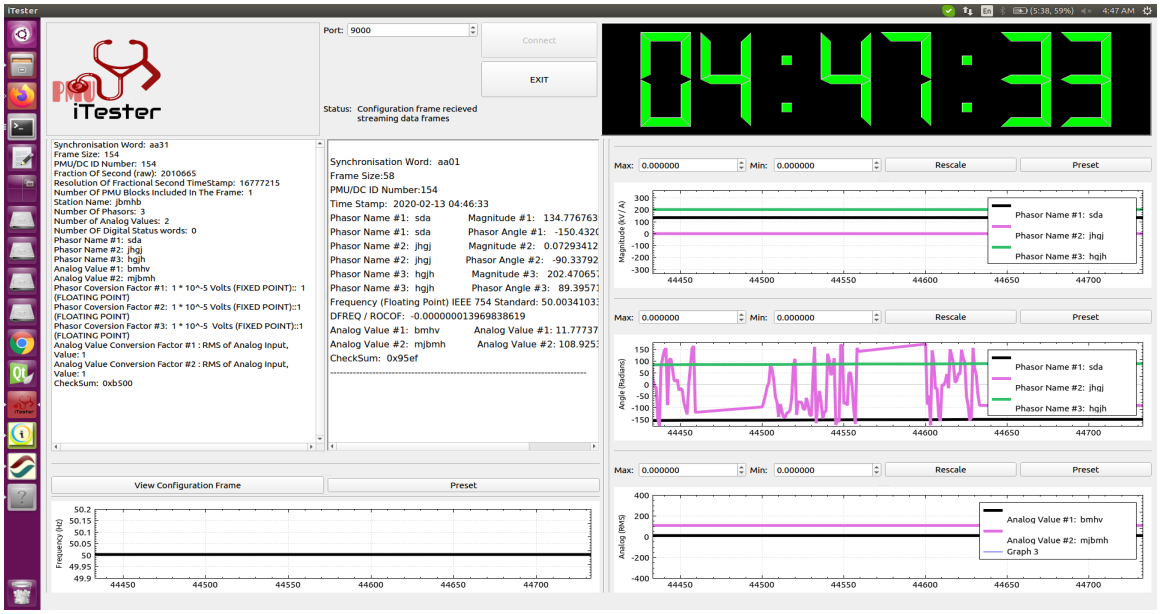


Figure 3.5: iTester in Streaming mode

Chapter 4

Conclusion and Future Work

4.1 Conclusion

- iTester at present handles a specific case of only 3 phasor, 2 analog and no digital (floating point) only.
- It always binds to a port address of 9000 (port address of DBServer). It listens to UDP port only.

4.2 Future Work

- Presently the data is being sent from PMU to PDC and from PDC to visualizer tool. Currently iTester is able to handle dataframe containing 3 Phasors and 2 analog values of single PMU only. In actual scenario, a data frame may contain any number of PMU data and any number of phasor, analog and digital data per PMU. iTester has to be modified to handle such scenarios.
- Changes has to be made such that the iTester has the ability to display phasor, analog and digital values of all PMUs.

Bibliography

[1] Phadke, Arun G., and James S. Thorp, *Synchronized phasor measurements and their applications*, Vol. 1. New York: Springer, 2008.

[2] Pandit, Nitesh, and Kedar Khandeparkar, "Design and implementation of IEEE C37. 118 based phasor data concentrator & PMU simulator for wide area measurement system," *Indian Institute of Technology Bombay Technology Report* (2012).

[3] Martin, K. E., et al."An overview of the IEEE standard C37.118.2 synchrophasor data transfer for power system," *IEEE Transactions on Smart Grid 5.4* (2014): 1980-1984.

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

[5] Alliance, Grid Protection."PMU connection tester," (2014).