Paper No. T.4.3, pp. 1-5

17th INTERNATIONAL SYMPOSIUM on POWER ELECTRONICS - Ee 2013

NOVI SAD, SERBIA, October 30th – November 1st, 2013

VIRTUAL INSTRUMENTS FOR POWER ELECTRONICS BASED ON FREE SOFTWARE TOOLS

Predrag Pejović, Mirjana Simić
University of Belgrade, Faculty of Electrical Engineering, Belgrade, Serbia

Abstract: *Virtual instruments described in this paper utilize a digital oscilloscope as an acquisition module and a personal computer to process the data. Described instruments measure mean value, root-mean-square value, power, apparent power, power factor, displacement power factor, total harmonic distortion, efficiency, and compute the waveform spectra. The measurements are organized at several stages, to provide verification at intermediate result level. Measurement process is fully automated, covering data acquisition, data processing, and report generation. The system is realized applying free software exclusively, and all the programs presented in the paper are freely available.*

Key Words: ***Education/Free Software/Power Line Measurements/Virtual Instrumentation***

1. INTRODUCTION

Widely available computing power in the form of personal computer (PC), associated with digital instruments capable of being connected and controlled by a PC, resulted in tremendous progress in virtual instrumentation [1]. Virtual instruments are, among many other applications, applied in power electronics and electric power engineering [2, 3, 4]. However, in all of the mentioned cases, they were based on proprietary software tools, which was a widely repeated pattern. Application of proprietary software increases the system cost, causes dependence on the program vendor and the program version, limits portability, and significantly reduces the designer's ability to have all the parameters of the measurement system under immediate control. On the other hand, one might argue that development cost of virtual instruments is higher if the instruments are based on free (or "open source", to avoid common confusion between free software and freeware) software tools. This is a quantitative issue, frequently discussed in qualitative and ideological terms. To make a slight contribution to the discussion, the authors of this paper would like to share their experience: the first of the authors did not have any experience with proprietary software tools used for virtual instrumentation, while the second author had a wast experience. On the other hand, the first author had an experience in general application of free software tools. Facing a dilemma about the choice of the software tools, initial condition the first author had, i.e. familiarity with Python programming language [5], accompanied by other benefits, resulted in a decision to use free software tools. The decision is based upon an estimate that the time required to master proprietary tools would be at least the same as to master free software tools and to develop the system based on them. The result is presented in this paper, and the authors would like to share it in the spirit of free software.

Development of the virtual instrumentation system described in this paper were initiated by a requirement to support measurements for two lab exercises in Power Electronics 2 class [6]. The lab exercises are focused to three-phase rectifiers, six-pulse and twelve-pulse, and of equipment only an oscilloscope for data acquisition and a PC equipped with free software tools were available. Available oscilloscope had two channels, and it was equipped with a current probe and a 100:1 voltage probe. Having this hardware, it was desired to provide measurements of:

1. mean value,
2. root-mean-square (RMS) value,
3. power,
4. apparent power,
5. power factor (PF),
6. displacement power factor (DPF),
7. total harmonic distortion (THD),
8. efficiency (η), and
9. compute waveform spectra.

Besides that, post-processing of the data should enable computation of the rectifier output port Thévenin equivalent parameters.

The first implementations of virtual instruments like proposed in this paper was used for research purposes, the results of which were summarized in [7]. However, these measurements were performed using chunks of undocumented software, with frequent changes. For classroom use [6], the software system ought to be documented and well structured, which was a chance to enhance its features and to provide capacity to generate measurement reports automatically. Besides, it was intended to provide a platform to extend features of the program to make it suitable for other applications.

Resulting software [8, 9] is available under GPLv3 license, thus being free (copyleft) software.

2. ARCHITECTURE OF THE MEASUREMENT SYSTEM

As mentioned, available hardware to develop the measurement system consists of a digital oscilloscope, and supported versions include Tektronix TDS 210 and Tektronix TDS 1000. The oscilloscope should be equipped with a current probe, either Agilent 1146A or Tektronix A621, and a 100:1 voltage probe. The oscilloscope supports two channels only, which is a significant constraint when three-phase measurements are aimed. It also contains a communication port that supports RS-232 communication with a computer.

The computer was equipped with Ubuntu 12.04 LTS operating system, and the requirement was to develop virtual instruments relying to the software already available in the repository, only. All the virtual instruments are implemented using Python programming language [5], using pySerial module [10] to facilitate communication between the oscilloscope and the PC, and PyLab package [11] to process the data and provide diagrams. The choice is made having in mind that Python is a modern interpreted multiple-paradigm programming language, with rich standard library, further extended with external packages. To generate the lab report automatically, processing of the measurement results and graphs is provided using LaTeX, which results in a report of the form [12]. In this manner, measurement and documenting processes are completely automated, and everything is done using free software exclusively.

3. COMMUNICATION WITH THE OSCILLOSCOPE

In virtual instruments described in this paper, the oscilloscope is used as an acquisition module that provides information about digitalized waveform shown in the oscilloscope screen. The oscilloscopes supported by the system provide this information in the form of 2500 equidistant samples over the waveform, each sample in 8-bit resolution. To communicate with the oscilloscope pySerial module is used [10]. Language used to communicate with the oscilloscope is fairly standard for a range of oscilloscope types [13], and provides an easy and intuitive way to issue commands to configure the instrument, as well as to get answers on queries and to collect the waveform data. Programs that facilitate communication with the oscilloscope are `take.py` of [8] and `uzmi.py` of [9]. The waveform data are rescaled according to the vertical axis settings, and the results are stored in `.npy` format [11] to facilitate further processing. In the case of `take.py` of [8], the program provides some additional information to the user, some waveform parameters, in order to simplify logical verification of the data collected. This turned out to be useful for students with moderate lab experience that tend to collect wrong signals.

As implied in Introduction, proposed techniques are focused to measurements on rectifiers, where all the processes are synchronized with the line frequency. This simplifies the analysis greatly, since the frequency of the waveforms is known and stable. The optimal frame of the oscilloscope (Tektronix TDS 210 and similar) with respect to time resolution over period covers $25\,\text{ms}$, having 2500 samples, and only 2000 samples are used in order to avoid spectral leakage. This provides time resolution of $T_S = 10\,\mu\text{s}$. In cases where the signal frequency is not known in advance, special care should be taken to measure the signal frequency precisely, to adjust the time scale, and to reduce the time frame to an integer number of periods in order to avoid spectral leakage. This issue is a topic of further development of the instrument, to extend its application to converters other than rectifiers. Spectral leakage is a frequent error in application of virtual instruments to periodic waveforms.

4. DATA PROCESSING

After the data samples are collected, computing listed waveform parameters is simple, and reduces to digital signal processing implementation of the parameter definitions. Experimental experience indicates that to obtain better results in measurements of two-waveform parameters, like power, apparent power, etc., to provide better sampling synchronism it is a good practice to stop the data collection (Run/Stop command of the oscilloscope) and to take the data from the oscilloscope while the sweep is stopped. This technique provides better power balance at the end of the measurements, and thus it has been programed in the acquisition scripts to be done automatically. Implementation of the measurements reduces to data processing, which is a simple task to perform with proposed software tools. Somewhat more complex computations are spectra related: displacement power factor and the total harmonic distortion. Just to illustrate how simple the code for data processing is, everything relevant for measurements is coded as:

```
def rms(x):
    return sqrt(mean(x ** 2))

def apower(x, y):
    return mean(x * y)

def thd(x):
    X1rms2 = 2 * X[1] ** 2
    Xrms2 = sum(X ** 2)
    return 100 * sqrt(Xrms2 / X1rms2 - 1)

def dpf(x, y):
    X = fft(x)
    Y = fft(y)
    return cos(angle(X[1]) - angle(Y[1]))
```

which covers computation of the waveform rms value (function `rms`), average power (function `apower`), waveform THD (function `thd`), and the displacement power factor (function `dpf`). Apparent power, power factor, and efficiency are computed by simple arithmetic operations according to their definitions.

Functions `rms` and `apower` that compute rms value and average power are completely time-domain based, which results in really simple implementation, just a line of code. Besides, high level array functions of [11] are used (NumPy package of the PyLab), which results in

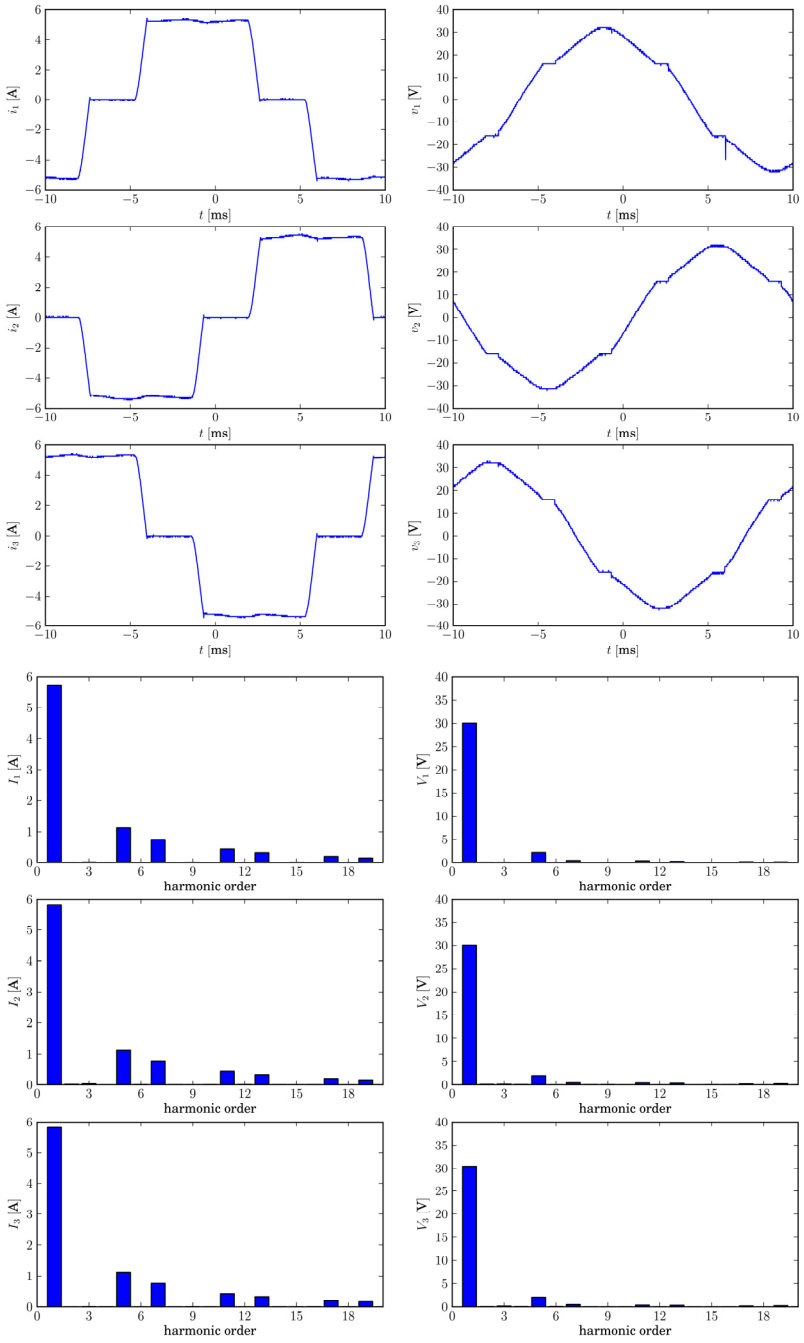

Figure 1.Waveforms and spectra of the input currents and voltages of a six-pulse three-phase rectifier with a constant current load.

fast execution, although this is less relevant in offline processing. On the other hand, functions `thd` and `dpf` rely on Fourier analysis of the signals, and special care should be taken regarding location and scaling of spectral components in the resulting array. This is particularly visible in the case of `thd` function, where square of `X[1]` is taken twice due to the double sided spectrum obtained by `fft` function of NumPy [11]. However, the spectral components were not rescaled to their physical meaning in neither of the functions, since in the case of the `thd` function the scaling factor cancels out, while in the case of `dpf` function amplitudes of spectral components are irrelevant.

A part of PyLab is matplotlib [14], which was used to provide high quality diagrams. As an example, waveforms and spectra of a six-pulse three-phase rectifier are shown in Fig. 1, while the corresponding diagrams for the twelve-pulse rectifier are shown in Fig. 2. The program provides figures with LaTeX lettering, which can easily be included in the final report, being compatible in format and style. Besides the illustration of matplotlib capabilities, Figs. 1 and 2 illustrate the educational point of the lab exercises: that

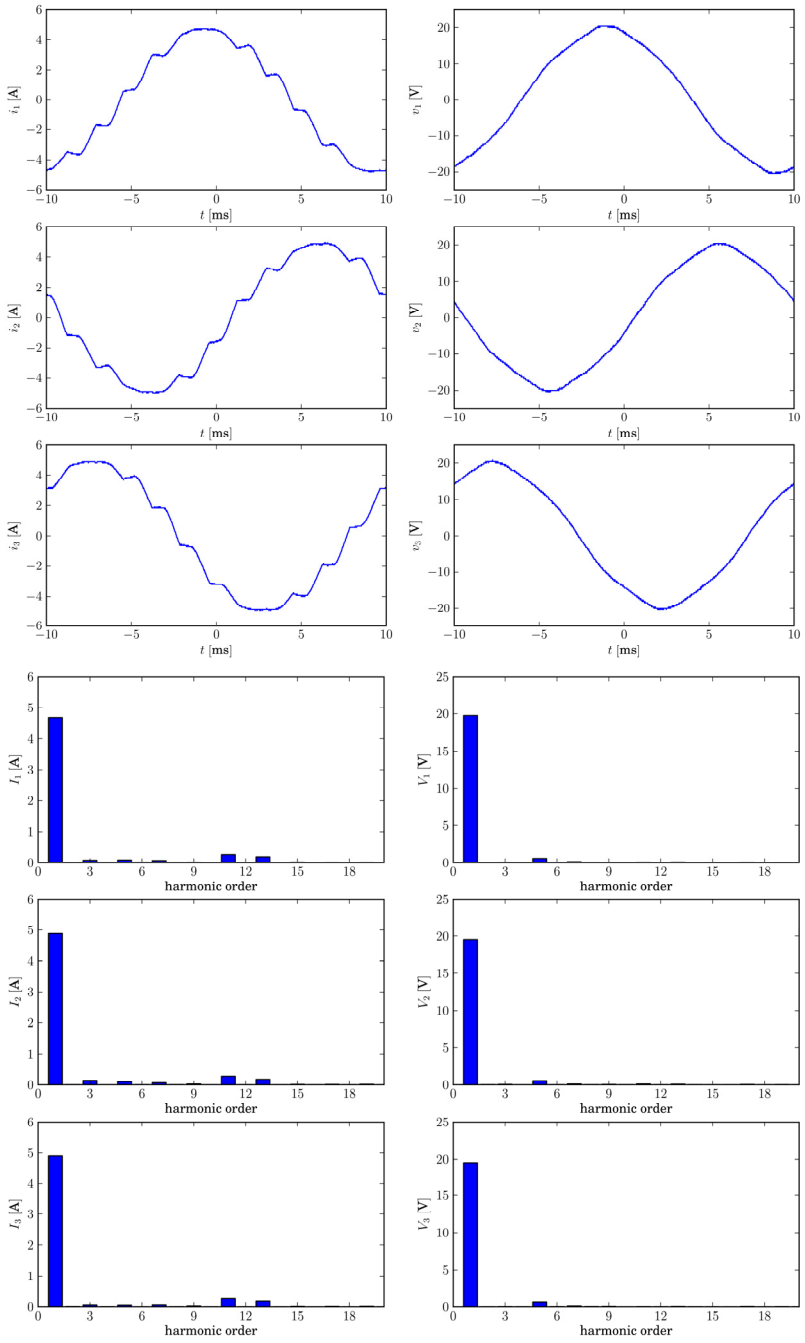

Figure 2. Waveforms and spectra of the input currents and voltages of a twelve-pulse three-phase rectifier.

the twelve-pulse rectifier provides less distorted input current waveforms than the six-pulse rectifier. In Fig. 1, the waveforms are recorded in the case of maximally allowed output current, which was $I_{OUT} = 5$ A. In that case, parasitic effects that cause notches in the input voltages are pronounced, and perfectly illustrate the background theory. Spectra of the input currents are in agreement with the theoretical predictions, having harmonic components at $6k \pm 1$ multiples of the line frequency. This causes distortion of the input voltages at the same frequencies, due to a finite internal impedance of the voltage sources. Some harmonics outside theoretically predicted locations are also observable, being caused by second-order effects, like unbalance of the input voltages. This is also an educationally valuable point, to compare theoretical predictions with experimental results.

5. ORGANIZATION OF THE MEASUREMENTS

At this point of the system development, the measurements are performed in several phases. The first phase is data collection, where students are required to position probes at appropriate places, and to run adequate program to collect the data. In the case of [8], the program is `take.py`, which also provides some information about the collected waveforms: power, rms

values, power factor, and so on. These information are important to prevent propagating errors, since completely unexpected results may warn the students to double-check what they were doing.

After the data are collected for one measurement setup, characterized by the output current and the filter type, another program is started, `point.py` that processes all the data collected for the setup, enabling computation of the rectifier efficiency which requires data about the input power at all three of the phases and the output power, i.e. four distinct measurements. Besides that, diagrams of the recorded waveforms, as presented in Fig. 1, are generated at this stage.

Next level in the process is assembling all the data about the rectifier with specified filter type, which is done by running program `assemble.py`. The program assembles the data collected at the data points specified by the output current, and provides cumulative diagrams, like the dependence of the rectifier efficiency on the output current, and the parameters of Thévenin equivalent of the rectifier output port.

Finally, the measurement report is automatically generated running program `report.py`, which is the topic of the next section.

In the case of twelve-pulse rectifier, structure of the measurement is similar, and the students are required to follow a checklist of measurements before they run a program to process the data and generate the report.

Experience with current measurement protocol is that measurements pass smoothly, without any problem. One idea for further development of the system is to provide a graphic user interface. The issue is non-technical and primarily ergonomic, thus at this stage priority of the task is set to low.

6. AUTOMATIC REPORT GENERATION

Another feature of the proposed measurement system is automatic report generation. The feature might be useful for certified laboratories that provide measurement reports. Due to the flexibility of Python, it was easy to provide this option. To process the report, LaTeX is used, and program `report.py` of [8] generates the `.tex` file which contains general data about the measurement (student names, date and time of the processing provided by the system), automatically generates required tables, and includes the figures. Furthermore, using `os` module, the program runs LaTeX system and provides the report in `.pdf` format. Without any scientific confirmation, we believe that the students were interested to see how a 49-page report in the case of [8] and 92-page report in the case of [9] are automatically generated within the time frame of several seconds.

7. CONCLUSIONS

In this paper, two measurement systems based on virtual instruments, intended for educational [6, 8, 9] and research [7] applications are described. Virtual instruments utilize a digital oscilloscope as an acquisition module, and a personal computer to process the data. Described virtual instruments measure mean value, root-mean-square value, power, apparent power, power factor, displacement power factor, total harmonic distortion, efficiency, and compute the waveform spectra. The measurements are organized at several stages, to provide verification at the intermediate result level, which prevents error propagation. The virtual instrumentation system is fully automated, covering the whole measurement process, which includes data acquisition, data processing, and report generation. The automatically generated report is LaTeX processed, includes text, tables, and diagrams. At this stage, the system lacks graphical user interface, and the need for such interface is doubtful. The system is realized exclusively applying free software, and all the programs described in the paper are freely available at [8, 9]. It is demonstrated that it is possible to develop virtual instruments exclusively using free software tools.

8. REFERENCES

[1] Harold Goldberg, "What is virtual instrumentation?," *IEEE Instrumentation & Measurement Magazine*, vol. 3, no. 4, pp. 10–13, Dec. 2000.

[2] Jen-Hao Teng, Shun-Yu Chan, Jin-Chang Lee, Roy Lee, "A LabVIEW based virtual instrument for power analyzers," *Proceedings of the International Conference on Power System Technology, PowerCon 2000*, vol. 1, pp. 179–184, 2000.

[3] Pavol Spanik, Libor Hargas, Miroslav Hrianka, Ivan Kozehuba, "Application of virtual instrumentation LabVIEW for power electronic system analysis," *12th International Power Electronics and Motion Control Conference, EPE-PEMC 2006.*, pp. 1699–1702, 2006.

[4] Mohamed Zahran, Yousry Atia, Abdullah Al-Hussain, Ihab El-Sayed, "Labview based monitoring system applied for pv power station," *The 12th WSEAS International Conference on Automatic Control, Modelling & Simulation (ACMOS'10)*, 2010.

[5] *Python Programming Language — Official Website*, [Online]. Available: http://www.python.org/

[6] Predrag Pejović. Power Electronics 2 web site [Online]. Available: http://tnt.etf.rs/~ms1ee2/

[7] Predrag Pejović, *Three-Phase Diode Rectifiers with Low Harmonics — Current Injection Methods*, Springer, 2007, ISBN 978-0-387-29310-3.

[8] Predrag Pejović. *Three-Phase Rectifier, lab exercise software* [Online]. Available: http://tnt.etf.rs/~ms1ee2/trofazni-ispravljac-init.zip

[9] Predrag Pejović. *Twelve-Pulse Rectifier, lab exercise software* [Online]. Available: http://tnt.etf.rs/~ms1ee2/12-pulse-init-2.zip

[10] *pySerial* [Online]. Available: http://pyserial.sourceforge.net/

[11] *PyLab web page* [Online]. Available: http://wiki.scipy.org/PyLab

[12] *Twelve-Pulse Rectifier, Power Electronics 2 Lab Report* [Online]. Available: http://tnt.etf.rs/~ms1ee2/report-12-pulse-2.pdf

[13] Tektronix, *Programmer Manual TDS 200-Series Digital Real-Time Oscilloscope*, 071–0493–01

[14] *matplotlib* [Online]. Available: http://matplotlib.org/