**FACTA UNIVERSITATIS**
Series: **Electronics and Energetics** Vol. 32, N$^o$ 1, March 2019, pp. 1-23
https://doi.org/10.2298/FUEE1901001P

# APPLICATION OF PYTHON PROGRAMMING LANGUAGE IN MEASUREMENTS

**Predrag Pejović**

University of Belgrade, School of Electrical Engineering, Belgrade, Serbia

**Abstract**. *Application of Python programming language in automation of measurement systems and creating virtual instruments is discussed in this paper. Requirements imposed to the software in order to perform these tasks are listed, and Python modules that support them are presented. Application of proposed techniques are illustrated in seven examples in different application areas. Analysis of software evolution, as well as the evolution of professional education yields conclusion that application of Python in automating measurement systems is promising.*

**Key words**: *computerized instrumentation, electric variables measurement, impedance measurement, measurements, measurement techniques, software measurement.*

## 1. INTRODUCTION

Over the years it exists, software evolved. At the beginning of computers, the program was not stored in the machine, instead the functionality had been hard wired for each application. A great step forward occurred with stored programs, at first written in a machine language. Such languages are considered as the first generation of programming languages. The second generation of programming languages involves assembly languages, somewhat more readable than the machine languages, but still heavily dependent on particular instruction set architecture. Finally, the third generation of programming languages, a prominent example of which was FORTRAN, which appeared among the first in this generation and gained huge popularity, provided abstraction that separated programmer from the machine instruction set architecture, enabling code portability. With portable code, software libraries appeared, accumulating knowledge and programming experience, and programming became a social activity. High level libraries, like libraries for numeric computation, are nowadays very rich and complete, and it is the most likely that an everyday problem a programmer faces is already solved and included in a library. In this manner, programming became a social activity: a programmer relies on program development tools, such as compilers and integrated development environments, developed by other programmers,

Received November 12, 2018
**Corresponding author:** Predrag Pejović
University of Belgrade, School of Electrical Engineering, 73 Kralj Aleksandar Blvd, 11000 Belgrade, Serbia
(E-mail: peja@etf.rs)

as well as software libraries, if he or she wants to program efficiently. This focused programming to solving specific tasks, while general, frequently encountered problems, already have readily available library solutions. This led to “gluing” languages, designed to provide efficient inclusion of library solutions, and to glue them together to solve a specific problem. Evolution of software libraries, growing in size and capabilities on daily basis, further supported this concept.

A prominent example of a programming language that supports “gluing” concept is Python [1]. Designed to be readable, with simple and clear syntax, while highly extensible by inclusion of software libraries, named modules, which can be used comfortably using a convenient namespace system. Python modules can be written in Python, but also in C or C++. Furthermore, it is possible to link FORTRAN libraries to Python modules. In this manner, a wast software heritage could be efficiently used in Python applications. A huge list of useful modules are included in Python Standard Library [2]. The modules used in applications focused in this paper are [3–6]. However, real power of Python is in the fact that it enables easy and straightforward inclusion of user contributed modules, outside the Python Standard Library. Application of Python in measurements relies on these modules and their flexibility to adjust to current trends in development of electronics measurement equipment. Review of such external modules [7–17] needed for automating electrical measurements and for creating virtual instruments is presented in this paper. Furthermore, the author of this paper contributed some modules [18–20].

Along with the evolution of computers and computer languages, the people who use computers evolved, too. In Serbia, the last generations that did not learn programming in their high school are getting retired nowadays, and the first generations that learned machine languages, assembly languages, FORTRAN, COBOL, and BASIC in their high school (“Programmer” high school specialization in Serbian high school curricula, lasted from 1977 to 1989) are about 10 years to retirement. Python is about to start to be taught in the sixth grade in elementary schools, and many schools and universities worldwide use Python as the first programming language. Very soon we might expect every professional in any of the technical or science disciplines to be proficient in programming, and the most likely, in Python programming language, which is rapidly becoming a standard language for high level programming.

Many social obstacles are present in automating measurement processes and creating virtual instruments at the time this article is being written. The driving force of this impediment are particular human interests, the process common to automation of any kind. Effects of such temporary impediment are expected to vanish, as they vanished in any other automation process. For example, nowadays simple electrical measurements are performed by a digital multimeter, which contains a microcontroller to process the data. Another example involves building construction, where distance measurements just a few years ago were dominated by measuring tape, while nowadays almost everyone uses digital laser distance meter, being a digital device. On the other end of the process, the measured data are processed by a computer. Who connects the two? In some cases, still a human, collecting the data, writing it down to a notebook, typing it back to a computer. Such jobs are likely to disappear, since computer connectivity enabled multimeters are already available. Common and standardized communication protocols, preferably wireless, and standardized data processing software are still needed, but they are likely to appear, since there are no technological obstacles to provide them. Another option provided by computer supported measurements is creation of virtual instruments. As an example, consider a digital oscilloscope which is a common piece of equipment in any lab and provides signal samples. By acquiring these samples and by processing them on a computer, power, apparent power, reactive power, power factor, displacement power factor, and total harmonic distortion could be measured, which creates virtual instruments that can measure quantities the oscilloscope initially could not measure.

Current state in evolution of measurement equipment, computers, and the people who operate both is such that one might expect that most of the measurements in future would be electronics based, and that the measurement results would be presented in a digital form. Furthermore, all the data are already processed by computers. Connectivity between instruments (which are computers in their construction, microcontrollers), and data processing computers is likely to increase to the level when it becomes an assumed part of any instrument. Some knowledge of programming is already assumed, and it is likely that in a decade every professional would be proficient in Python, limiting the need for graphical programming languages in measurement applications, enabling all the necessary programming tasks to be performed in a general purpose programming language, not requiring any specialized knowledge nor training. All these facts suggest that Python is a convenient choice for a programming language to support measurement automation and virtual instruments. Such conclusion spontaneously and independently appeared in many places, resulting in significant amount of available literature, like [21–27]. According to available literature, it seems that at the moment application of Python in measurements is the most popular in advanced scientific experiments. Examples of applications which might find approach proposed in this paper useful are [28–32].

This paper is written at the tenth anniversary of the author's use of Python in electrical measurements for measurement automation and creation of virtual instruments. All measurements for the experimental results in [33] and all the papers aggregated in it are performed using virtual instruments that post-process the data collected using a digital oscilloscope. The software is ported to Python, as presented in [34]. Furthermore, the same technology is used to create different instruments and systems in [35–40]. This paper aggregates gained experiences and lists all the modules and techniques necessary to design automated measurement systems and virtual instruments, providing some examples. The choice of tools is made to minimize requirements to application specific knowledge, and such that all the tools are free software.

## 2. REQUIREMENTS IMPOSED BY AUTOMATED MEASUREMENT SYSTEMS AND VIRTUAL INSTRUMENTS

At first, let us review functionality required by the design of automated measurement systems and virtual instruments. At first, communication with instruments should be provided in full duplex, proving computers with an ability to send commands to instruments as well as to receive data containing measurement results. The idea is not new, it originated in late 1960s [41], emerging with HP-IB, later renamed to GPIB after a wide acceptance, finally standardized as IEE 488 and IEEE 488.2. Thus, about half a century ago it had been evident that measurements are time consuming and boring, and that these processes should be automated connecting instruments to a computer.

Standardization of commands followed, resulting in SCPI commands in 1990 [42, 43], almost three decades ago. Nowadays GPIB still exists, however the interface is somewhat outdated, being expensive, requiring expensive cables, thus being replaced by general purpose higher bandwidth standard interfaces like USB and Ethernet, where highly applicable communication hardware is available at low prices due to high production volume.

The second task to be performed is data processing and storage. Computers are efficient in that, and many libraries to perform these tasks exist, as well as database utilities which may be required in the case huge datasets are being processed. The third task is data visualization, since providing graphical representation of measurement results is frequently required. The fourth group of tasks which are always required covers timestamping and time control, like providing necessary delays for the system to reach the steady state or providing timed measurements at required time instants, like in climate parameters monitoring. Also, in free software, which is in the focus of this paper, it is common practice to use other general purpose tools to provide specific functionality of the designed system. In an example which will be covered in this text, for automatic report generation a text processing system LaTeX is used. To provide such functionality, communication to the operating system should be provided, to start other programs and to control their execution. In some cases, graphical user interface is needed, especially in cases when the designed system is going to be used by less qualified personnel or by many people, so tools for providing this functionality should be available. For all of the listed tasks, appropriate Python modules are already available as free software.

In some cases, experimental hardware should be reconfigured during the measurement process, like in the cases where devices under test should be switched or a reference value should be changed. An inexpensive way to do that is to use the Arduino platform [44], which could be easily controlled by a python program using [17]. Arduino Mega [45] board is of special interest, since it provides a huge number of 54 digital inputs and outputs at a moderate price.

## 3. Python Modules Useful for Automated Measurements and Virtual Instruments

### 3.1. Communication with Instruments

As already discussed, communication with instruments reduces to exchange of ASCII strings when the instruments support SCPI [42, 43] commands. Thus, while considering instrument purchase, support of SCPI should be an important issue, since it enables the user to create his or her own programs to control the instrument. Nowadays, the most popular means to communicate to instruments are by USB and by Ethernet, which is also an issue in instrument selection.

Communication over the USB interface is provided using usbtmc protocol [46]. Python support for this protocol is provided by python-usbtmc module [7]. In [47], a script to install python-usbtmc on GNU/Linux Debian-based systems (tested on Ubuntu and Linux Mint distributions) is provided. The module provided effective communication to Agilent 33220A signal generator [48], Tektronix TBS 1052B-EDU oscilloscope [49], which is in everyday use in Laboratory for Electronics at the School of Electrical Engineering, University of Belgrade, in Electrical Measurements class [50], as well as the multimeter [51]. A Python module used to support communication and control of the oscilloscope [49] is given in [18].

Communication over Ethernet is provided using VXI-11 protocol [52], implemented in python-vxi11 module [8]. The module has been successfully used in [38] in communication with [48] and [53], and is in everyday use in [50, 54].

Another popular communication interface used with older equipment is the RS-232 interface. Communication over that interface is supported by python-serial [9] module. Besides, this module supports communication over USB to some devices, like the Arduino boards [44]. A Python class that supports communication to Tektronix oscilloscope is provided at [19], and it had been used successfully with TDS 210, TDS 220, TDS 1000, and TPS 2024 oscilloscopes.

Providing communication to measurement equipment is the most specific part of the measurement automation and the design of virtual instruments as proposed in this paper. After the communication has been established, everything else is common general-purpose programming. Communication to instruments according to SCPI [42, 43] reduces to exchange of ASCII strings, and conversion of such strings is readily available in Python even with built in functions, which might be supported with string module of the Python Standard Library if some more complex string operations are needed.

### 3.2. Data Processing

Data processing required by measurement methods is readily provided in Python using numpy module [10], primarily. The module provides numerically efficient array objects and operations over these objects, including basic linear algebra and FFT, among other numerical methods. In the case some advanced numerical algorithms are needed, scipy library [11] is available, although most of the tasks are performed by numpy. It is worth to mention that PyLab programming environment with namespaces set to provide user friendly numerical programming environment is available [12], although recently deprecated for encouraging old fashioned programming styles. In the case excessive data analysis is necessary, Pandas module is available [13].

### 3.3. Data Visualization

To provide data visualization in Python matplotlib [14] seems to be the best known tool. It provides data plotting, both 2D and 3D, and saving the diagrams in a plethora formats. It should be noted that there are other, both well known and mature tools available.

### 3.4. Timestamping and Time Control

Access to the system clock is provided by time module of the Python Standard Library [3]. The module is intuitive and comfortable to program with. In both of the Python versions, 2 and 3, modules with the same names are available for the tasks required by applications considered in this paper, and only versions of modules for Python 2 are cited in this document, assuming equivalent module availability for version 3.

### 3.5. Access to Other Programs

In some case, like the automated report generation, it is necessary to access other programs, like LaTeX [55] or convert [56] for image data format conversion. Such functionality is provided by sys [4] and os [5] modules of the Python Standard Library [2].

### 3.6. Graphical User Interface Design

In the case designed system is intended for specific use, with a limited number of experienced users, it is not likely that creating a graphical user interface (GUI) would be an interesting option. However, if the audience that uses the program is wider, a GUI is required. Fortunately, there are many GUI development tools and modules available for Python, including rapid application development tools. In an example presented in this paper, taken from [38], the GUI had been created using tkinter [6] module, being the simplest and already included in the Python Standard Library. Other very popular and advanced tools are available, like PyQt [15] and wxPython [16], which might be of interest in more complex designs.

## 4. SUPPORTING PROGRAMS

The use of Python programming language in GNU/Linux environment provides an option of a simple interfacing with other free software tools. Only two of such programs would be mentioned: LaTeX [55] which was used for automatic report generation in [34], resulting in [57], and convert used to convert image data formats provided by the digital oscilloscope, as used in [18]. Any other program could easily be invoked from python, and its output used in further processing.

## 5. THE USE OF ARDUINO PLATFORM

Arduino [44] is a very popular prototyping platform, characterized by free software and open hardware, which greatly fueled its popularity. The platform itself can be utilized as an instrument, either using its built-in AD converters, either connecting external high precision converters. However, a different application would be suggested here, based upon availability of a large number of digital ports which could be configured either as an input or as an output: for reconfiguration of the measurement system, and in some cases to facilitate indication of the system state. Easy and direct interfacing with Python might be provided using [17], and with [45] up to 54 digital signals could easily be controlled. Relays operated by Arduino digital output voltage and current levels are readily available, so reconfiguration of measurement system could be easily provided.

## 6. APPLICATION EXAMPLES

### 6.1. Applications in Power Electronics and Electric Power

Power electronics is a principal research area of the author, and he started to use virtual instrumentation in power electronics, to support research in three phase rectifiers that resulted in a number of papers aggregated in [33]. The measurements required to support the research included measurement of power, apparent power, reactive power, power factor, displacement power factor, total harmonic distortion, and efficiency. Additionally, characterization of components, like recording magnetizing curves and analyzing component constitutive relations and losses were required. Specific equipment to perform these tasks are nowadays available, but being narrow in application and highly expensive. For the research purposes, virtual instruments had been created, performing digital post-processing of recorded waveforms. After the Python based instrumentation had been introduced, being entirely based on free software, the methods had been ported to education, to laboratory exercises in Power Electronics 2 [34]. To illustrate automation of measurement process, a 92-page measurement report is automatically generated during a lab exercise that lasts for only two hours, an example being available at [57]. As an example, in Fig. 1 waweforms of voltages and currents at the 6-pulse three-phase rectifier inputs are presented, and their spectra are given in Fig. 2. Effects caused by commutation of the diodes, like the notches in the input voltages and limited slope in the input currents are observable. In the spectra, absence of harmonic components at triples of the line frequency is observable, that matches analytical results. To improve the input current spectra and to reduce the harmonic pollution, 12-pulse rectifiers are applied, and waveforms that correspond to this rectifier are presented in Fig. 3, while corresponding spectra are given in Fig. 4. Reduced distortion is readily available. Collected samples are used to determine input power, output power, efficiency, power factor, displacement power factor and total harmonic distortions (THD) of the input currents and voltages. Signal processing is simplified by the fact that the system frequency is the line frequency, known in advance, and taking an appropriate number of samples spectral leakage is avoided. In systems with variable frequency this issue should be considered, and it will be discussed in this paper in the section that covers frequency response measurement.

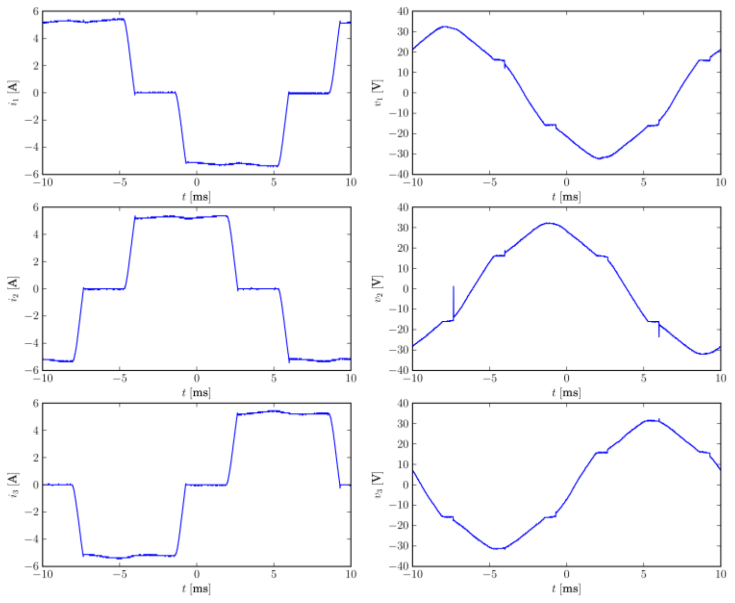


**Fig. 1** Waveforms of the input currents and voltages, 6-pulse rectifier

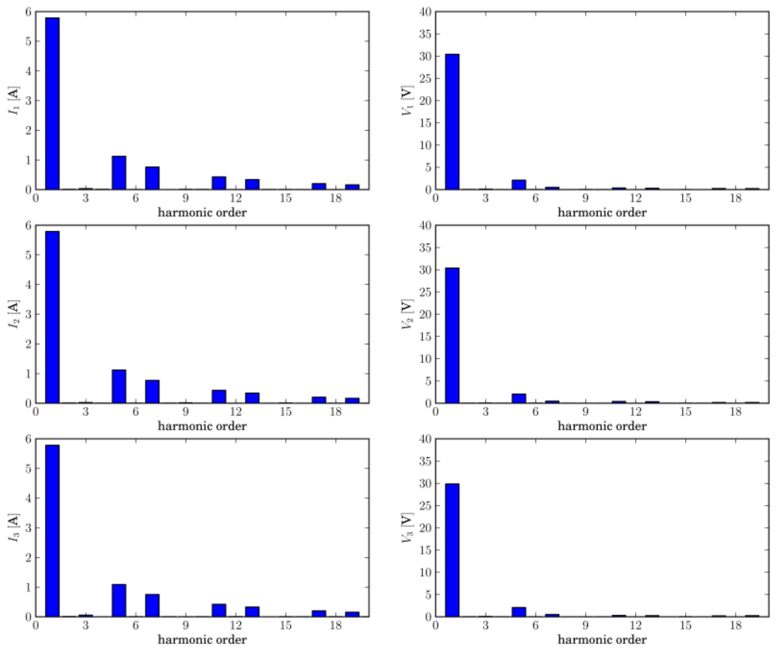


**Fig. 2** Spectra of the input currents and voltages, 6-pulse rectifier

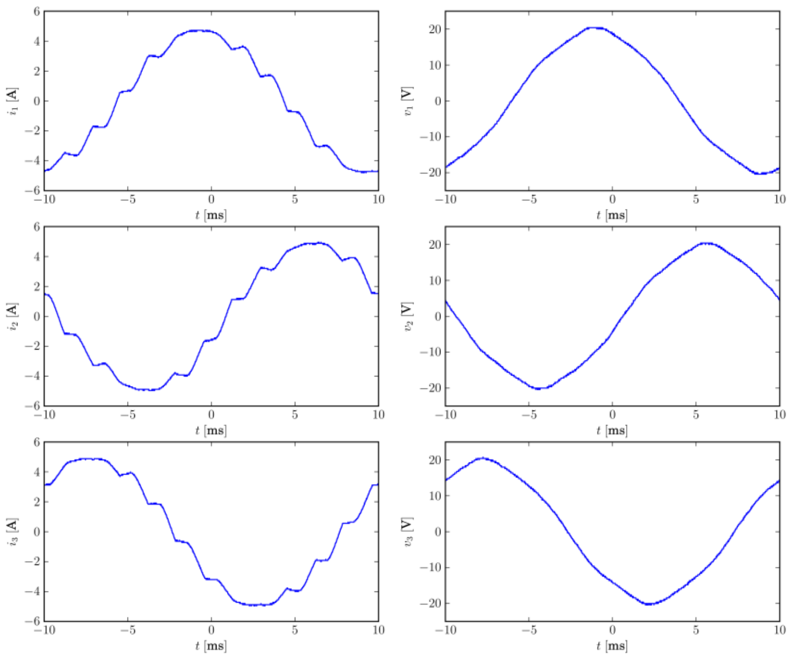


**Fig. 3** Waveforms of the input currents and voltages, 12-pulse rectifier

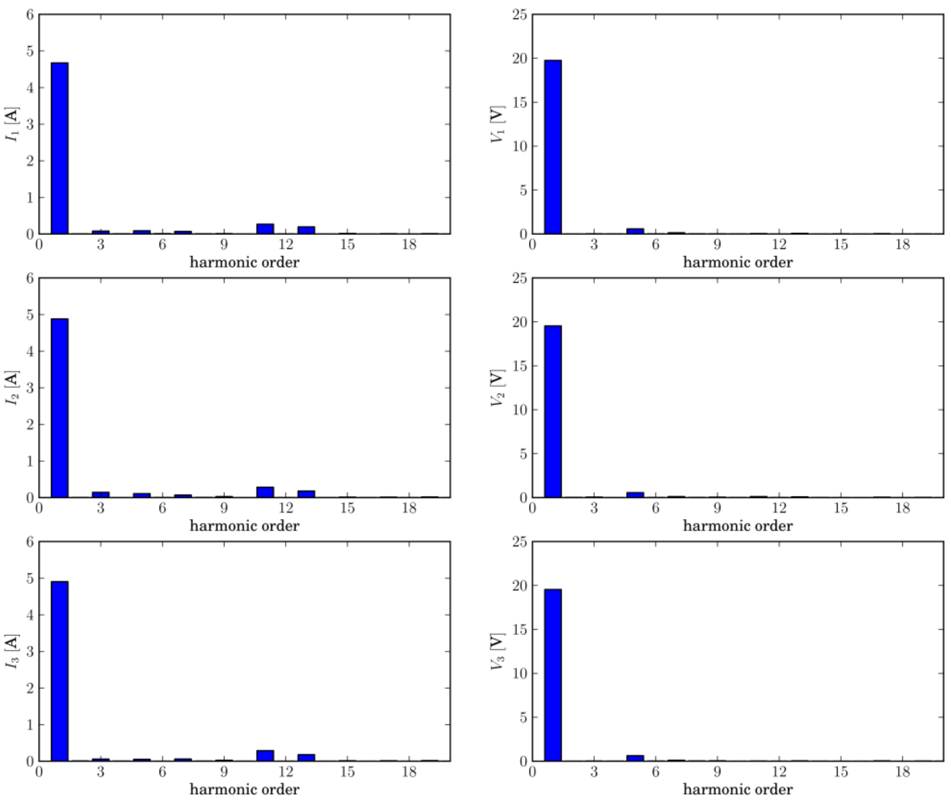


**Fig. 4** Spectra of the input currents and voltages, 12-pulse rectifier

A direct application of the same technology, with minor extension to provide timed measurement and timestamping, is presented in [35], where long lasting measurements, over a week, of the line voltage and its total harmonic distortion (THD) were provided. A diagram presenting measured THD values is presented in Fig. 5, indicating periodic behavior during working days, while having a specific pattern during weekends. To provide the diagram of Fig. 5 measurements were made every minute over a week, and 10080 data points are collected and presented.

To illustrate daily variations of the THD, the waveform of Fig. 5 in the part that corresponds to workdays is plotted in Fig. 6 such that the curves are plotted for each day one atop another. Close to periodic behavior could be observed, illustrating effects of human daily activities on the voltage THD. On the other hand, the THD exposes a different pattern during weekends. To illustrate that, the same methodology as for the workdays, presented in Fig. 6 is applied, and the results is presented in Fig. 7. Significant reduction of the bump from 08 to 16 hours could be readily observed, corresponding to the reduction of business activity during weekends. For the rest of the day, the THD profile remained about the same.

Furthermore, the software based virtual instrument is able to record root-mean-square (RMS) value of the phase voltage at every point. Measurements are made over a week every minute, and the resulting phase voltage histogram is presented in Fig. 8. In this manner, registration of the phase voltage is obtained applying general purpose instruments and some controlling software that provides measurement automation and timestamping.

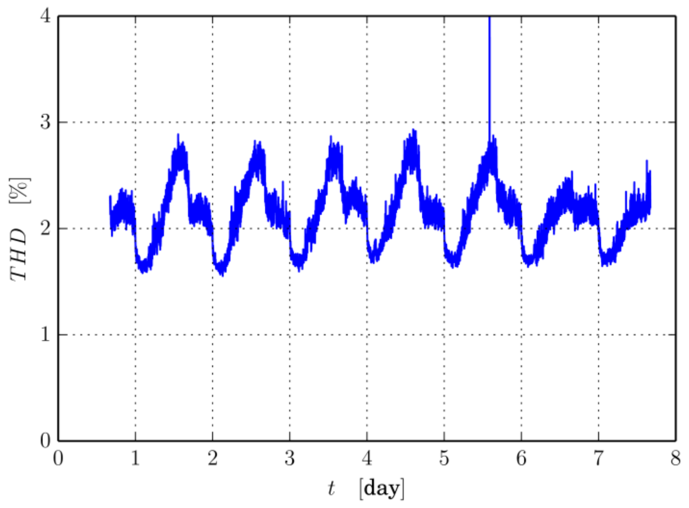


**Fig. 5** THD of the phase voltage

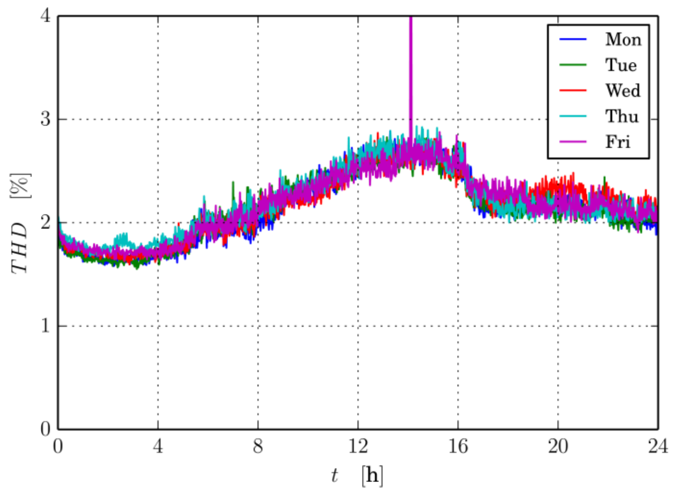


**Fig. 6** THD of the phase voltage, workdays

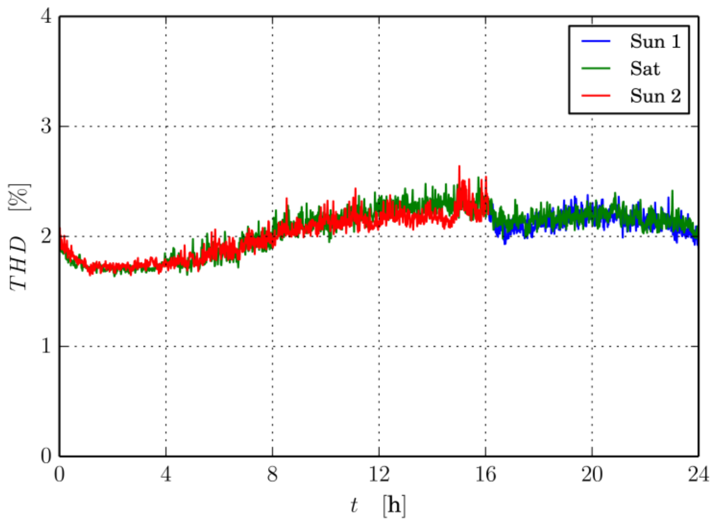


**Fig. 7** THD of the phase voltage, weekend

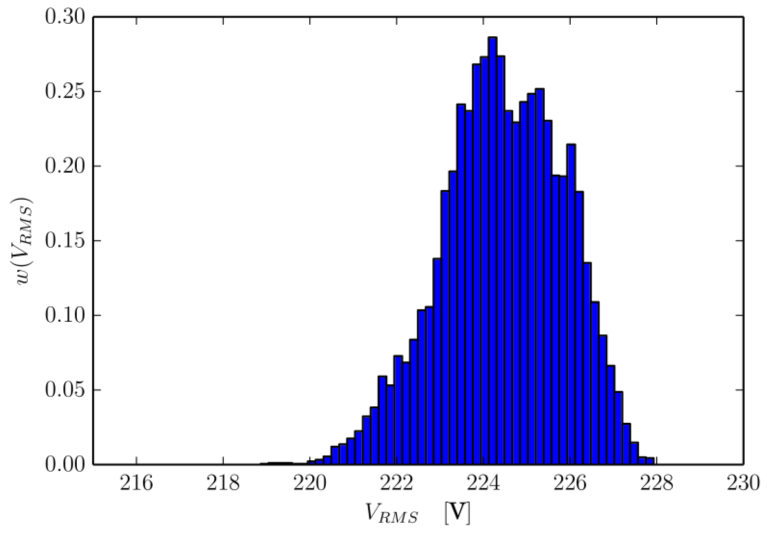


**Fig. 8** Hystogram of the phase voltage RMS value

Proposed measurement methods were extended to cover both measurement and control of a solar power generator, aiming maximum power point tracking of the solar panel [36]. Rapid prototyping is achieved using general purpose instruments and a personal computer to close the loop, which was possible due to the low frequency dynamics in the loop. As a part of the same project, a solar power harvester is designed as presented in [37], where the solar panel is kept at the maximum power point by an adjustable resistive load, and harvested power is measured in order to estimate average, minimum and maximum power that could be harvested in the specified location as it depends on weather conditions.

### 6.2. DC Voltage Calibrator

A different application of the proposed methods is presented in [38] where design of a special instrument is approached using the software tools. A DC voltage calibrator was needed, being an expensive instrument, narrow in application, not worth purchasing for the particular application. A substitution is created closing a loop that included two general purpose instruments, a programmable signal generator [48] and a highly precise multimeter [53]. The system is presented in Fig. 9. The voltage assigned to the signal generator is adjusted in order to generate required voltage, and improvement in accuracy of two orders of magnitude is achieved, placing the generated voltage error within about 500 μV limit, as depicted in Fig. 10. The data of Fig. 11 contain 20001 data points, obtained using an automated system which loops the required calibrator output voltage over all possible values in the available range. Just assuming 30 seconds of manual work per data point, which is fairly optimistic, the measurement process would last for more than 165 man hours.

Since the instrument was intended for use by less qualified personnel, a graphical user interface (GUI) is built using tkinter [6] module. The choice is made considering the application as low demanding, not requiring rapid application development tools, and having in mind that tkinter module is a part of the Python Standard Library [2]. A screenshot of the resulting GUI is shown in Fig. 11, and it presents assigned voltage, measured voltage, and the voltage assigned to the generator, which is an intermediate step governed by the feedback loop. In the example of Fig. 11 an offset of 23 mV had to be added to compensate for the signal generator error and to locate the calibrator error within 500 μV limit.

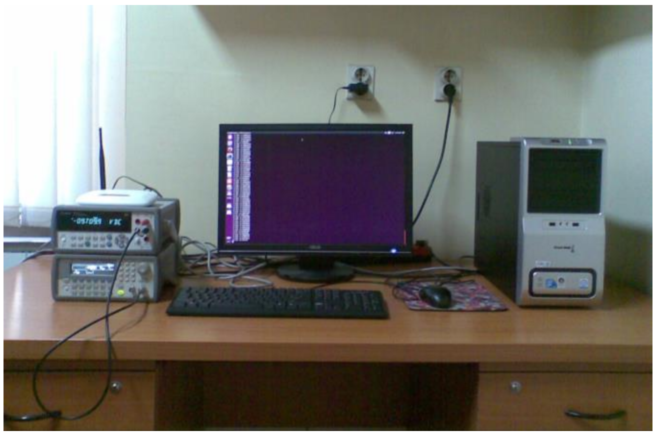


**Fig. 9** The calibrator system

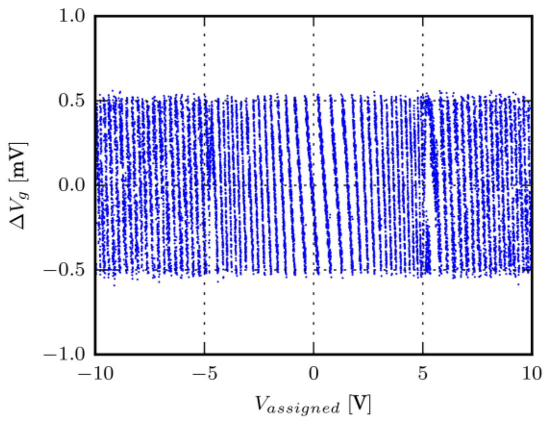
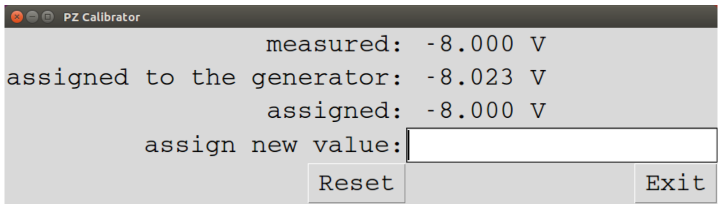


**Fig. 10** Voltage error, closed loop calibrator



```
                measured:  -8.000 V
assigned to the generator:  -8.023 V
                assigned:  -8.000 V
        assign new value:
```

**Fig. 11** Graphical user interface of the calibrator

### 6.3. Applications in Education

Developed techniques proved to be successful in education, since free access to all of the source codes is available, the code could be analyzed in classes and shared to students, and the time spent in the laboratory, limited due to the huge lab burden, could be effectively utilized, illustrating key concepts instead of spending time on trivial repetitive tasks. The first application of the proposed methods in education is made in Power Electronics 2 [34], where lab exercises were introduced to illustrate the theory presented in the course, as shown here by Figs. 1–4. After this successful implementation, course of Electrical Measurements [50] is reformed, as reported in [39]. After a year, the course is further updated, since new oscilloscopes were obtained, providing much faster data acquisition, enabling introduction of even more experiments since the intellectually and educationally idle processing time had been reduced further. A set of nine new laboratory exercises is created [54]. According to student questionnaires, they enjoyed the concept which reduced hard work and increased the number of experiments, focusing to the essence instead to the trivia.

### 6.4. Measurement of Frequency Response

Another application example of the proposed techniques, used both in education [39, 50] and in practice is an automated system for frequency response measurement [40]. The system is intended to measure frequency response of transmittance and immittance, and a numerically intensive technique is used to measure amplitude and phase, extracting the first harmonic. Such approach is applied to remove influence of noise in the amplitude and phase measurements, which is going to be illustrated as significantly present in measurements encountered in practice. In this manner, precise measurements are obtained, since all of the collected samples affect the result, filtering the noise out.

The algorithm starts with selecting the time scale such that the the minimal number of signal periods is covered by the oscilloscope time frame. The frequency is assigned to the signal generator, being an independent variable, thus the signal period is known. Time span of the oscilloscope screen belongs to a discrete set of values achievable by the given oscilloscope, and the span that includes the lowest number of whole signal periods is selected, determining the oscilloscope time scale. For the oscilloscope applied [49] the number of periods covered by a screen is either one or two, depending on the signal frequency, as depicted in Fig. 12.

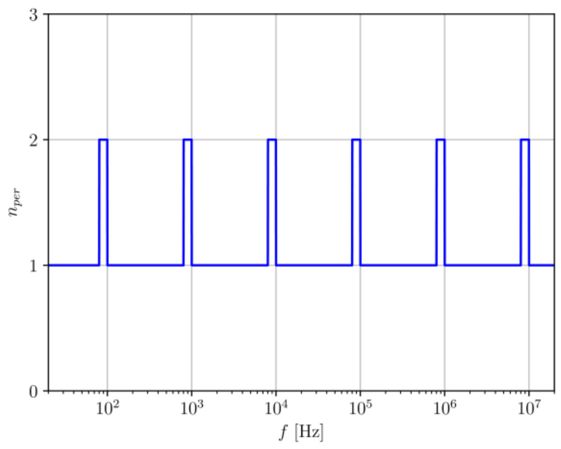


**Fig. 12** Number of periods covered by the oscilloscope screen

After the time scale has been selected, the number of samples taken into account is computed by rounding $2500 \times n_{per} \times T_0 / T_{span}$, where 2500 is the number of samples per time frame for the given oscilloscope, $n_{per}$ is the number of signal periods per time frame, shown in Fig. 12, $T_0$ is the signal period, and $T_{span}$ is the time span covered by the time frame. The number of samples is solely dependent on the signal frequency, and the diagram is shown in Fig. 13. In [40], an older version of the algorithm is presented, reducing the scope to only one signal period, but in cases when more than one signal period is covered by the oscilloscope screen, due to the limitations imposed by the discrete set of available time scale values, better results are obtained by taking two periods into account, and the improved algorithm is presented in this paper.

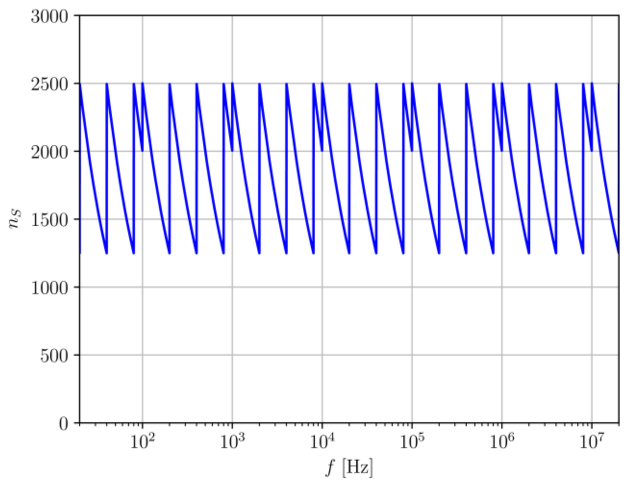


**Fig. 13** The number of considered samples

The algorithm assumes that the number of considered samples $n_s$ is known, and that samples of signals $x(t)$ and $y(t)$ are available as $x_k$ and $y_k$ for $k \in \{0, \ldots\, n_s - 1\}$. Waiting functions are computed next, according to

$$c_k = 2\cos\left(2\pi n_{per} \frac{k}{n_S}\right) \tag{1}$$

and

$$s_k = 2\sin\left(2\pi n_{per} \frac{k}{n_S}\right) \tag{2}$$

According to the Fourier analysis, for signal $x(t)$ cosine component is obtained as

$$X_C = \frac{1}{n_S} \sum_{k=0}^{n_S - 1} x_k c_k \tag{3}$$

while the sine component is

$$X_S = \frac{1}{n_S} \sum_{k=0}^{n_S - 1} x_k s_k \;. \tag{4}$$

After the cosine and sine components are determined applying the Fourier analysis, effectively filtering the noise out, the signal amplitude is obtained as

$$X_m = \sqrt{X_C^2 + X_S^2} \tag{5}$$

and the phase is obtained as

$$\varphi_x = \operatorname{atan}2(X_S, X_C) \tag{6}$$

using the $\operatorname{atan2}$ function that takes two arguments and provides the result in the range $(-\pi, \pi]$.

The same signal processing is performed over signal $y(t)$, resulting in values of $Y_C$, $Y_S$, $Y_m$, and $\varphi_y$. Finally, the transfer function magnitude is obtained as

$$|H(j\omega_0)| = \frac{Y_m}{X_m} \tag{7}$$

and the phase is obtained as

$$\varphi_{H0} = \varphi_y - \varphi_x \tag{8}$$

The value $\varphi_{H0}$ is named "raw phase" since it takes value in the range $-2\pi < \varphi_{H0} \le 2\pi$ since $-\pi < \varphi_x, \varphi_y \le \pi$. The value is correct, due to the phase periodicity over $2\pi$, but it is convenient to provide the phase value in the range $-\pi < \varphi_H \le \pi$. In this aim, phase adjustment by appropriate shifting for $2\pi$ is performed according to

$$\varphi_H = \begin{cases} \varphi_{H0} & -\pi < \varphi_{H0} \le \pi \\ \varphi_{H0} - 2\pi & \pi < \varphi_{H0} \\ \varphi_{H0} + 2\pi & \varphi_{H0} \le -\pi. \end{cases} \tag{9}$$

This concludes the algorithm for the one point, for the specified frequency value. The algorithm is repeated for specified frequency range and the specified number of data points.

As the first example, consider a circuit of Fig. 14, used to illustrate frequency response effects caused by the capacitor, to identify frequency range where it behaves approximately as an open circuit and the range where it behaves approximately as a short circuit. The program is run, and the frequency response is obtained as presented in the diagram of Fig. 15, clearly indicating areas of flat frequency response where the capacitor could be considered either as open circuit, bellow 1 kHz in the considered case, or as short circuit, which occurs above 100 kHz in the considered case.

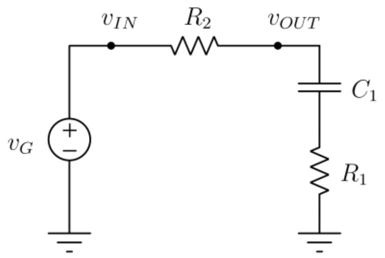


**Fig. 14** The circuit, $R_1 = R_2 = 1\,\text{k}\Omega$, $C_1 = 10\,\text{nF}$

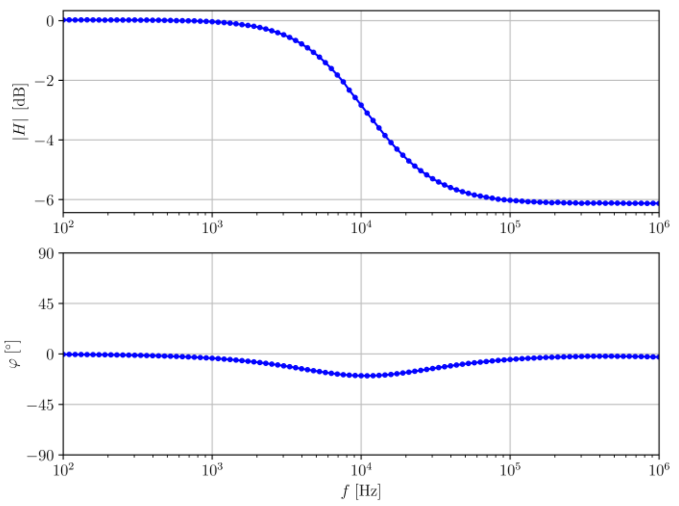


**Fig. 15** Frequency response of the circuit

The same system could be used for immittance measurements, for impedance and admittance, using the circuit of Fig. 16. In the circuit of Fig. 16 $R$ is used as a reference resistor, and the current through the measured impedance is computed as

$$i = \frac{v_1 - v_2}{R} \tag{10}$$

The same data processing algorithm as for the transfer functions is applied, taking signals $v_2(t)$ and $i(t)$ as $y(t)$ and $x(t)$ if impedance computation is the goal.

Application of the method to analyze electronic components provides insight in their operation and suggest suitable modeling strategies. As an example, in Fig. 17 frequency response of a capacitor $C$ = 1 nF impedance is presented. The result matches expectations, and barely noticeable deviations of measured phase from $-90^\circ$ at the beginning and at the end of the diagram are caused by a huge difference of the capacitor impedance at considered frequency and the impedance reference of $R$ = 20 kΩ. This is expected, since measured impedance varies for four decades, i.e. $10^4$ times over the considered frequency range, and a constant reference impedance is used. To improve the result, suggested approach that uses Arduino to reconfigure the circuit by adapting the reference impedance value to the measured impedance should be applied.

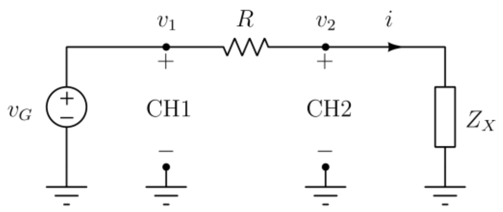


**Fig. 16** Circuit structure for impedance measurement; $R$ is the impedance reference value

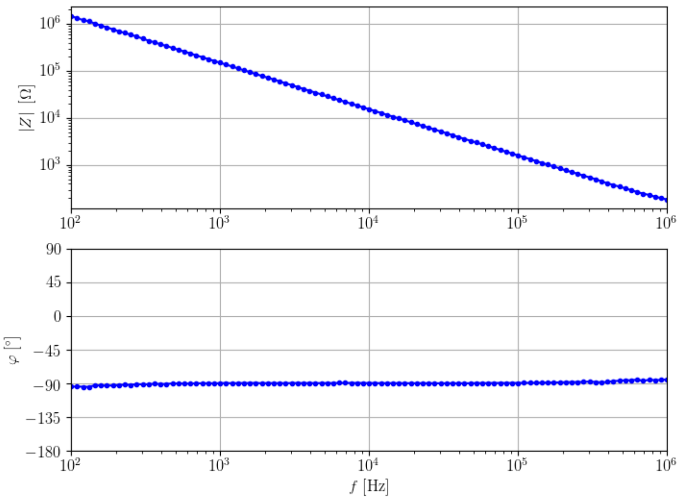


**Fig. 17** Frequency response of a capacitor impedance, $C = 1\text{nF}$

In contrast to the capacitor impedance frequency response, which follows the ideal model, impedance of an inductor is presented in Fig. 18. The inductor has rated inductance of 10 mH, but it exposes inductive behavior only in the frequency range from about 1 kHz to about 200 kHz. At low frequencies, parasitic resistance of the winding dominates the impedance, while at high frequencies parasitic capacitance of the winding dominates the response, resulting in capacitor-like frequency response above the resonant frequency of about 400 kHz. The results are obtained using a reference resistor of $500\Omega$.

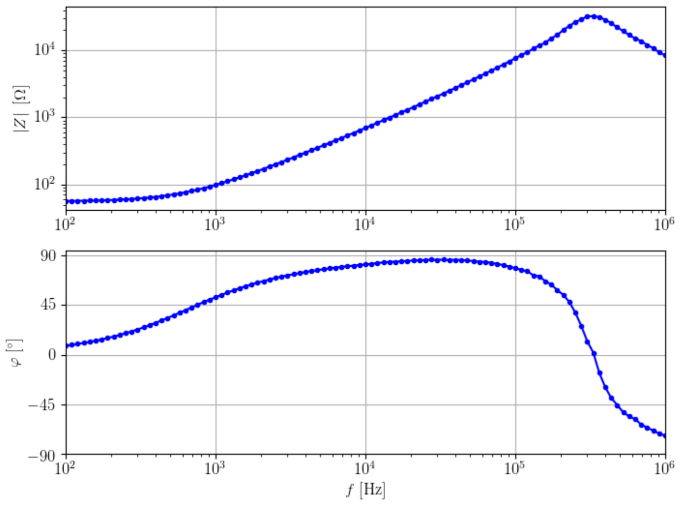


**Fig. 18** Frequency response of an inductor impedance, $L = 10\text{mH}$

As a final example that covers impedance measurements, consider frequency response of an electrolytic capacitor impedance, presented in Fig. 19. To measure impedance of the electrolytic capacitor, a DC offset of 2 V has been applied, and the measurements are made with 0.5 V amplitude of the signal generator AC component. The capacitor shows dominantly capacitive behavior only at frequencies lower than 300 Hz, and in the frequency range from 300 Hz to about 300 kHz equivalent series resistance slightly above 1 Ω dominates the impedance. Above 300 kHz, equivalent series inductance starts to dominate the impedance behavior.

To illustrate waveforms captured during the measurement process and noise filtering, the waveforms recorded while measuring the electrolytic capacitor impedance at the frequency of 289.087 Hz are presented in Fig. 20. Red trace corresponds to the capacitor current, while the yellow trace is the capacitor voltage. The cyan trace is the input voltage AC component. Significant presence of noise in the capacitor voltage waveform could be readily observed. Similar situation occurs in the frequency range from 300 Hz to 300 kHz, when the capacitor voltage is low. Regardless the noise, consistent measurements of amplitude and phase are presented in Fig. 19, indicating that the noise is successfully removed by the signal processing, not affecting the measurement result.

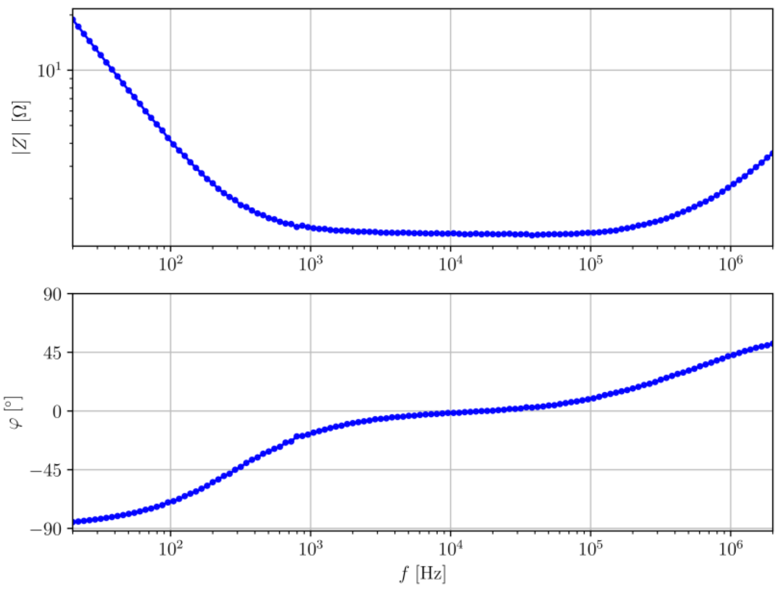


**Fig. 19** Frequency response of an electrolytic capacitor impedance, $C = 470\mu\text{F}$

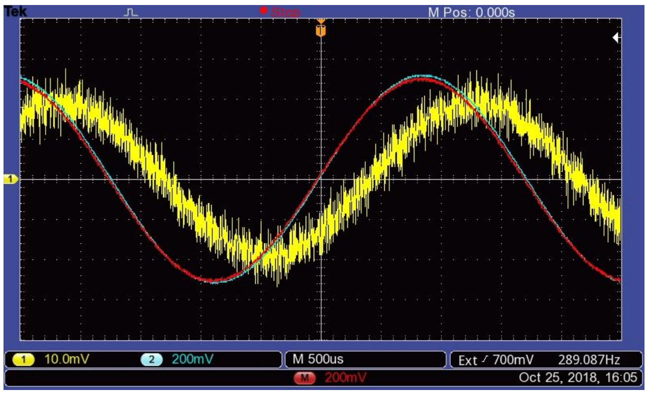


**Fig. 20** Waveforms recorded during the electrolytic capacitor impedance measurement: yellow — capacitor voltage; red — signal proportional to the capacitor current; cyan — voltage of the signal generator

In educational application of [39, 50], measurements of a transmission line transfer function is an experiment that attracts lots of student attention, and has an educational value of connecting courses that cover circuit theory to engineering practice. Due to the nature of the problem, linear frequency scale is appropriate, and the transfer function of an open transmission line is presented in Fig. 21. Resonances and nonlinear phase response could be readily observed. Repeating the experiment with properly terminated transmission line, results of Fig. 22 are obtained, indicating flat amplitude response and linear phase response, corresponding to close-an-ideal transmission system.

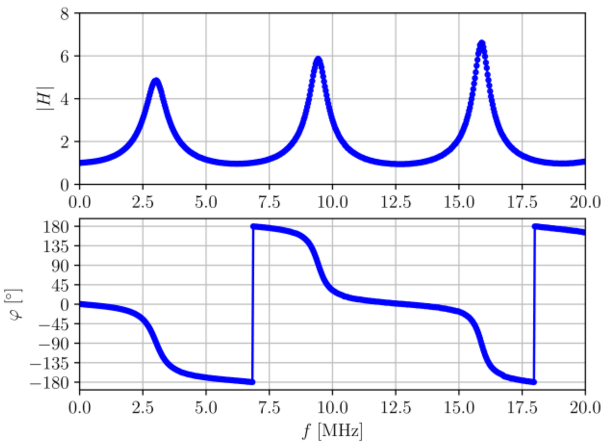


**Fig. 21** Frequency response of an open transmission line

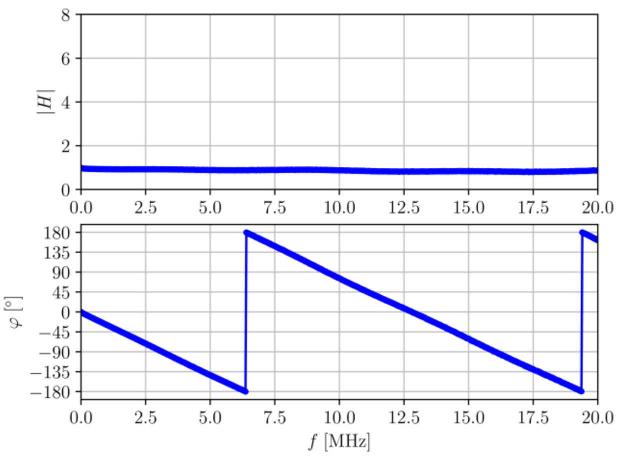


**Fig. 22** Fequency response of properly terminated transmission line

## 7. Conclusions

In this paper, application of Python programming language in creating automated measurement systems and virtual instruments is discussed. It is shown that to create such systems a set of specific tasks should be performed, not frequent in common application programming. The tasks are listed, and the Python modules that support performing them are looked for. It is shown that for all of the specific tasks there are Python modules readily available, either from the Python Standard Library, either from external sources, some of them highly specialized to support communication with instruments. Effective methods of including modules and arranging them in separate namespaces turned out to be useful in considered application. It is also shown that other programs, like LaTeX for text processing might be useful in creating automated measurement tools, to provide automatic report generation, which might be of use in certifying laboratories. The use of Arduino platform is proposed to provide measurement system controlled automatic reconfiguration and indication of the system state and performance. Tools for controlling Arduino platforms directly from Python are identified.

Application of the proposed methods is illustrated in four different areas, as reported by the author in seven papers. Applications started in power electronics, and positive experiences spread to metrology, to the design of a DC voltage calibrator, to education, where the methods were used in modernizing two courses, and in measurements of system frequency response, as applied in electronics, acoustics, and control system design. In some of these applications, selection of the time scale and the number of considered samples in the case of variable signal frequency is controlled by an updated algorithm presented in this paper.

Overall conclusion is that Python is an adequate tool for creating automated measurement systems and virtual instruments, due to its modular structure and openness for contribution of modules. In the choice of programming tools, the attention has been made to favorize general purpose tools and techniques, to minimize specific knowledge requirements. Having in mind evolution of software and the people who work in metrology, it is likely to expect wide application of the proposed approach and methods, which already started in several places independently.

## REFERENCES

[1] Python Programming Language — Official Website, [online] Available: http://www.python.org/
[2] The Python Standard Library, [online] Available: https://docs.python.org/3/library/
[3] time — Time access and conversions, [online] Available: https://docs.python.org/2/library/time.html
[4] sys — System-specific parameters and functions, [online] Available: https://docs.python.org/2/library/sys.html
[5] os — Miscellaneous operating system interfaces, [online] Available: https://docs.python.org/2/library/os.html
[6] Graphical User Interfaces with Tk, [online] Available: https://docs.python.org/2/library/tk.html
[7] python-usbtmc, [online] Available: https://github.com/python-ivi/python-usbtmc
[8] Python VXI-11, [online] Available: https://github.com/python-ivi/python-vxi11
[9] pyserial, [online] Available: https://pythonhosted.org/pyserial/
[10] NumPy, [online] Available: http://www.numpy.org/
[11] SciPy, [online] Available: https://www.scipy.org/
[12] SciPy: PyLab, [online] Available: https://scipy.github.io/old-wiki/pages/PyLab
[13] Pandas, [online] Available: https://pandas.pydata.org/
[14] matplotlib, [online] Available: https://matplotlib.org/
[15] PyQt's Modules, [online] Available: http://pyqt.sourceforge.net/Docs/PyQt4/modules.html
[16] wxPython, [online] Available: https://wxpython.org/
[17] Python-Arduino-Proto-API-v2, [online] Available: https://github.com/vascop/Python-Arduino-Proto-API-v2
[18] P. Pejović, oscusb, Python module to support communication with oscilloscopes over USB, [online] Available: http://tnt.etf.bg.ac.rs/~oe2em/oscusb.py
[19] P. Pejović, oscusb, Python module to support communication with oscilloscopes over RS-232, [online] Available: http://tnt.etf.bg.ac.rs/~oe2em/oscrs232.py
[20] Pr. Pejović, oscusb, Python module to support presentation of numbers in engineering notation, [online] Available: http://tnt.etf.bg.ac.rs/~oe2em/engineeringnotation.py
[21] J. M. Hughes, Real World Instrumentation with Python: Automated Data Acquisition and Control Systems. O'Reilly Media, Inc., 2010
[22] G. Real, L. Raviola, M. F. Jauré, and A. O. Vitali, "Data acquisition system for didactic laboratories based on open-source hardware and free software," In Proceedings of the 2015 XVI IEEE Workshop on Information Processing and Control (RPIC), 2015, pp. 1-6.
[23] J. L. Johnson, H. T. Wörden, and K. V. Wijk, "PLACE: an open-source python package for laboratory automation, control, and experimentation," *Journal of laboratory automation*, vol. 20, no. 1, pp. 10-16, 2015.
[24] I. J. Koenka, J. Sáiz, and P. C. Hauser. "Instrumentino: an open-source software for scientific instruments," *CHIMIA International Journal for Chemistry*, vol. 69, no. 4, pp. 172-175, 2015.
[25] I. J. Koenka, J. Sáiz, and P. C. Hauser. "Instrumentino: An open-source modular Python framework for controlling Arduino based experimental instruments," *Computer Physics Communications*, vol. 185, no. 10 pp. 2724-2729, 2014.
[26] F. J. F. Martín, M. V. Llopis, J. C. C. Rodríguez, J. R. B. González, and J. M. Blanco, "Low-cost open-source multifunction data acquisition system for accurate measurements," *Measurement*, vol. 55, pp. 265-271, 2014.
[27] A. J. Lewis, M. Campbell, and P. Stavroulakis, "Performance evaluation of a cheap, open source, digital environmental monitor based on the Raspberry Pi," *Measurement*, vol. 87, pp. 228-235, 2016.
[28] V. Davidović, D. Danković, S. Golubović, S. Djoric-Veljkovic, I. Manić, Z. Prijić, A. Prijić, N. Stojadinović, and S. Stanković, "NBT Stress and Radiation Related Degradation and Underlying Mechanisms in Power VDMOSFETS," *Facta Universitatis, Series: Electronics and Energetics*, vol 31, no. 3, pp. 367-388, 2018.
[29] S. K. Mohapatra, K. P. Pradhan, and P. K. Sahu, "Resolving the bias point for wide range of temperature applications in high-k/metal gate nanoscale DG-MOSFET," *Facta Universitatis, Series: Electronics and Energetics*, vol. 27, no. 4, pp. 613-619, 2014.
[30] S. K. Mohapatra, K. P. Pradhan, and P. K. Sahu, "ZTC bias point of advanced fin based device: The importance and exploration," *Facta Universitatis, Series: Electronics and Energetics*, vol. 28, no. 3 pp. 393-405, 2015.

[31] I. Manić, D. Danković, V. Davidović, A. Prijić, S. Djorić-Veljković, S. Golubović, Z. Prijić, and N. Stojadinović, "Effects of pulsed negative bias temperature stressing in p-channel power VDMOSFETs," *Facta Universitatis, Series, Electronics and Energetics*, vol. 29, no. 1, pp. 49-60, 2015.
[32] X. Saura, M. Riccio, J. Suñé, A. Irace, and E. Miranda, "Study on the spatial generation of breakdown spots in MIM capacitors with different aspect ratios," *Facta Universitatis, Series Electronics and Energetics*, vol. 28, no. 2 pp. 177-192, 2015.
[33] P. Pejović, "Three-Phase Diode Rectifiers with Low Harmonics - Current Injection Methods," Springer, 2007.
[34] P. Pejović, M. Simić, "Virtual Instruments for Power Electronics Based on Free Software Tools," In Proceedings of the17th International Symposium on Power Electronics, Ee 2013, Novi Sad, October-November 2013.
[35] P. Pejović, M. Simić, "A System for Measuring Mains Voltage Parameters and Logging the Data," In Proceedings of the 18th International Symposium on Power Electronics, Ee 2015, Novi Sad, October 2015.
[36] V. Lazarević, M. Bjelica, P. Pejović, "Maximum Power Point Tracking Control System of Photovoltaic Module Using Free Software and Standard Laboratory Equipment," In Proceedings of the 18th International Symposium on Power Electronics, Ee 2015, Novi Sad, October 2015.
[37] P. Pejović, M. Bjelica, "A Simple System to Estimate On-Site Solar Energy Harvesting," In Proceedings of the 18th International Symposium on Power Electronics, Ee 2015, Novi Sad, October 2015.
[38] P. Pejović, A. Zeković, "Software Supported DC Voltage Calibrator," In Proceedings of the XI International Symposium Industrial Electronics, INDEL 2016, Banja Luka, November 3-5, 2016.
[39] P. Pejović, "Electrical Measurements Revisited — Experiences from Modernizing the Course," In Proceedings of the IEEE EUROCON 2017, Ohrid, Republic of Macedonia, 6-8 July 2017, pp. 838-844.
[40] P. Pejović, "An Automated System for Frequency Response Measurement Based on Free Software Tools," In Proceedings of the XII International Symposium Industrial Electronics, INDEL 2018, Banja Luka, November 1-3, 2018.
[41] Wikipedia contributors, IEEE-488, [online] Available: https://en.wikipedia.org/wiki/IEEE-488
[42] Wikipedia contributors, Standard Commands for Programmable Instruments, [online] Available: https://en.wikipedia.org/wiki/Standard_Commands_for_Programmable_Instruments
[43] Standard Commands for Programmable Instruments (SCPI), [online] Available: http://www.ivifoundation.org/docs/scpi-99.pdf
[44] M. Banzi, Getting Started with Arduino, Second Edition, O'Reilly Media, 2011
[45] Arduino Mega 2560 Rev3, [online] Available: https://store.arduino.cc/arduino-mega-2560-rev3
[46] Universal Serial Bus Test and Measurement Class Specification (US-BTMC), Revision 1.0, April 14, 2003, [online] Available: http://sdpha2.ucsd.edu/Lab_Equip_Manuals/USBTMC_1_00.pdf
[47] P. Pejović, usbtmcinstall.zip, [online] Available: http://tnt.etf.bg.ac.rs/~oe2em/usbtmcinstall.zip
[48] Agilent Technologies Agilent 33220A 20 MHz Waveform Generator User's Guide, [online] Available: http://cp.literature.agilent.com/litweb/pdf/33220-90002.pdf
[49] TBS1000B-EDU Series Datasheet, [online] Available: https://www.tek.com/datasheet/digital-storage-oscilloscope-0
[50] P. Pejović, Electrical Measurements, course web site, [online] Available: http://tnt.etf.bg.ac.rs/~oe2em/
[51] Keysight Technologies Digital Multimeters, 34460A Digital Multimeter, 6 (1/2) Digit, Basic Truevolt, [online] Available: https://literature.cdn.keysight.com/litweb/pdf/5991-1983EN.pdf
[52] VMEbus Extensions for Instrumentation TCP/IP Instrument Protocol Specification VXI-11, Revision 1.0, The VXIbus Consortium, 1995, [online] Available: http://www.vxibus.org/files/VXI_Specs/VXI-11.zip
[53] Agilent 34410A and 34411A Multimeters, [online] Available: http://cp.literature.agilent.com/litweb/pdf/ 5989-3738EN.pdf
[54] P. Pejović, "Laboratorijske vežbe iz električnih merenja" [online] Available: https://zenodo.org/record/ 1311557/files/prirucnik.pdf?download=1
[55] CTAN Comprehensive TeX Archive Network, [online] Available: https://ctan.org/
[56] ImageMagick convert, [online] Available: https://imagemagick.org/script/convert.php
[57] Twelve Pulse Rectifier - Lab Report Example, [online] Available: http://tnt.etf.bg.ac.rs/~ms1ee2/report-12-pulse-2.pdf