การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 48
The 48th Electrical Engineering Conference (EECON-48)
วันที่ 19-21 พฤศจิกายน 2568 ณ โรงแรมฟูราม่า จังหวัดเชียงใหม่

Real-Time Analysis S-Parameters via Remote Control of a Vector Network Analyzer Using API Integration

Artit Rittiplang[1], Nakrop Jinaporn [1], and Atipong Suriya[1]

[1]Department of Electrical Engineering, Faculty of Engineering, Ubon Ratchathani University, Thailand,
artit.r@ubu.ac.th, nakrop.j@ubu.ac.th, atipong.s@ubu.ac.th

Abstract

Nowadays, programming commands are widely used in automation control instruments for real-time data acquisition in automatic test equipment. This control is typically implemented using SCPI, which simplifies code writing, but usually requires the vendor's software to be running. Therefore, our focus is on using an API that enables direct control of the instrument without the need to launch vendor software. In this demonstration, the PicoVNA 106 is used as the instrument, and we develop the API protocol using the Python programming language to integrate the system independently of vendor or commercial software. This approach enhances flexibility and accessibility for educational and research applications, supporting hands-on learning in programming, automation, and real-time measurement systems. Furthermore, it can significantly reduce costs associated with commercial software packages.

Keywords: Automation Control, Programming Command, Vector Network Analyzer, S-parameters, API, SCPI.

1. Introduction

In the field of electrical and electronic engineering, automated test equipment plays a crucial role in measurement, calibration, and validation processes. As the demand for faster and more reliable testing grows, real-time instrument control has become increasingly important [1]-[7]. Traditional methods often rely on manual operation or proprietary software provided by instrument manufacturers, which can limit flexibility and hinder integration into custom or automated systems.

These challenges have become a key topic in engineering education, where students are increasingly expected to learn how to control measurement instruments through application programming interfaces (API). Unlike Standard Commands for Programmable Instruments (SCPI), which often requires the manufacturer's software to be running during communication [8]-[9]. API can provide more direct and lightweight access to device functions without launching external software. API and SCPI processes are shown in Fig. 1.

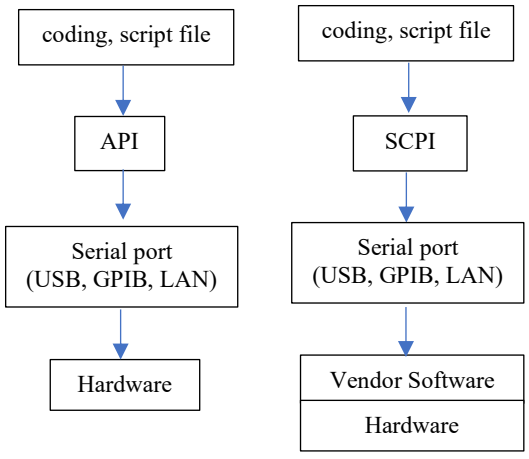


Fig 1. Instrument control by programing (a) API protocol, (b) SCPI protocol

As shown in Fig.1(b), the SCPI protocol requires the vendor's software to be running in order to manage SCPI commands for hardware control. Typically, the vendor's software is integrated into the device's display interface, such as the screen on an oscilloscope, and the front panel of a DC power supply and multimeter.

In this work, we demonstrate an API-based approach , as shown in Fig.1(a) with Python for instrument control using the PicoVNA106 vector network analyzer (VNA). A custom Python API is developed to interface with the VNA, allowing users to configure measurements and acquire S-parameters and real-time response, without relying on the official PicoVNA software [9]-[10].

This approach offers a more flexible and scalable solution for both academic and industrial applications. It is particularly beneficial in educational environments, where students gain practical experience in real-time instrumentation control, data acquisition, and the integration of measurement systems into broader automation workflows. Additionally, it helps reduce costs associated with commercial software licenses such as MATLAB and LabVIEW [5]-[7], by promoting the use of open-source and custom-built alternatives

2. The Design of API on Windows OS

This design is currently compatible with Python versions 3.8 to 3.11. Firstly, it is necessary to download the appropriate set of files from the PicoVNA5 SDK v5.3.1 [10] corresponding to the specific version of Python being used. In this demonstration it is based on

*Corresponding Author
Atipong Suriya, atipong.s@ubu.ac.th

Python 3.11 version and Windows 11. Those PicoVNA5 SDK files should include and place in the same directory as the Python script (.py file) as shown in Fig.2 to ensure correct operation and successful integration with the PicoVNA API.

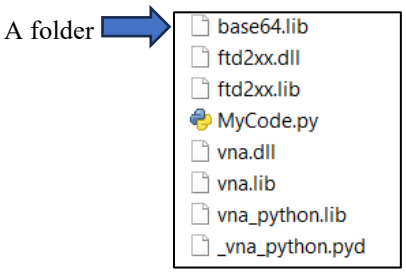


Fig 2. Picovna files and Python script must be in the same folder.

Python must install the vna package (pip install vna) for interfacing and communicating. Then, we have two typical program structures for considering [9].

1) Synchronous mode: this mode waits for the entire measurement sweep to complete before processing any measurements.

2) Asynchronous mode: it is processing measurements point-by-point without waiting for the entire sweep to complete.

In this design, we use asynchronous mode for faster real-time measurements compared to synchronous mode, which is a bit more difficult to implement, but it is worthwhile for real-time applications in industrial or medical settings. That mode has a flowchart as shown in Fig.3. Then, we will design the real-time displaying for S-parameters and time-domain of responses of low pass filter, band pass filter.

Table 1 Required Python packages

Library	Purpose
tkinter	Standard Python GUI library
tkinter.ttk	buttons, labels, entries
threading	Enables background measurement without freezing the GUI.
time	optional delays
matplotlib.pyplot	Used to create plots
matplotlib.backends .backend_tkagg .FigureCanvasTkAgg	Integrates matplotlib plots into tkinter windows.
PIL.Image	Loads and displays image (e.g., our university logo) in the GUI.
vna	Provides access to the VNA device and related functions

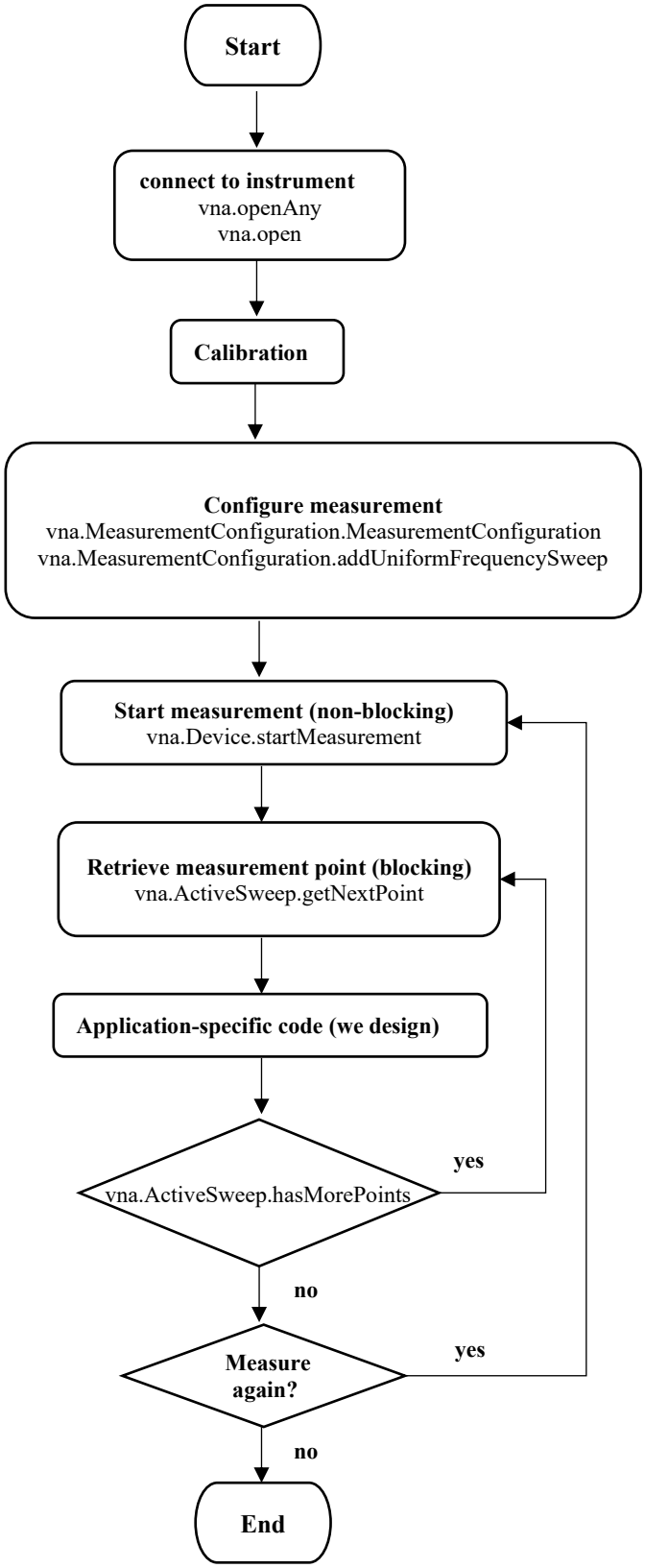


Fig 3. Asynchronous mode for real-time measurements

Table 1-2 shows Python packages and functions used in this work

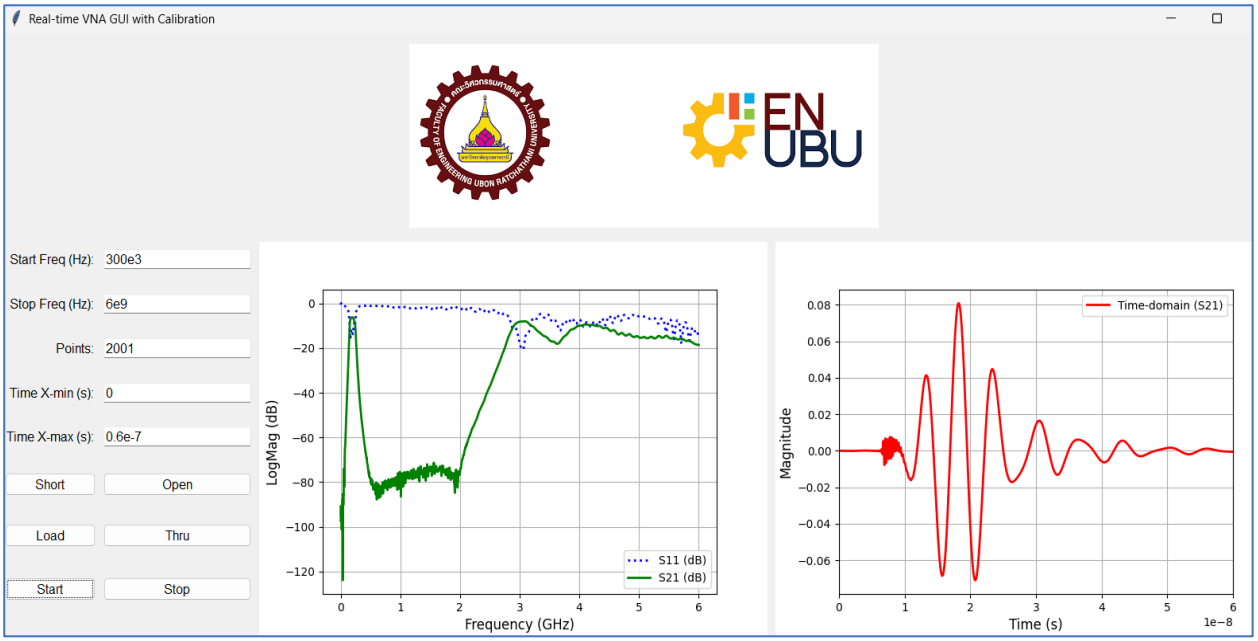


Fig 4. the real-time GUI of S11 S21 (left side) and time domain S21 (right side)

Table2 Required VNA functions

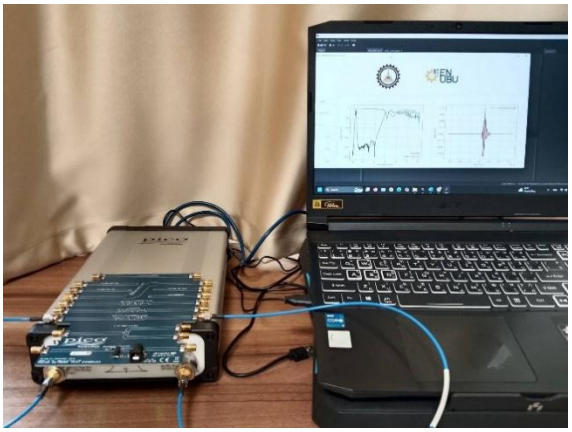
Function	Role
vna.Device.openAny()	connects to VNA device.
vna.MeasurementConfiguration()	configuration for frequency sweep
vna.toLogMag(pt.s11 / pt.s21)	Converts complex S-parameter to log magnitude (dB)
vna.transform(options, parameter, measurements)	Converts S-parameter to time-domain
vna.TimeDomainOptions()	Sets time-domain transform options (e.g., HANNING window)

We have adopted Python as our software platform, starting and building upon the example codes provided in the official datasheet repository [10]. Finally, the project has been completed, with the application successfully displaying S11, S21, and time-domain results in Fig4.

3. Experimental Setup

The PicoVNA106 instrument in Fig5(b) is firstly connected to a bandpass mounted on a circuit toolkit and a laptop in Fig5(a). When the application is executed, a real-time graphical user interface (GUI) is displayed, as shown in Fig 4. Firstly, we set the start frequency to 300 kHz and the stop frequency to 6 GHz, with 2001 sampling points, by entering the values on the screen. Secondly, we then perform calibration using short, open, load, and through standards from the circuit toolkit in Fig5(c) to ensure accurate S11 and S21 data.

In the left plot of Fig4, the S11 parameter is represented by a dotted line, while the S21 parameter is shown as a solid line. The right plot illustrates the time-domain response of the S21 parameter. For the bandpass filter measurement in Fig. 6 (left), the solid line shows a passband between 150 MHz and 250 MHz, with an



(a)

(b)

(c)

Fig 5. (a) real experiment (b) PicoVNA106 (c) circuit toolkit.

insertion loss of approximately –5 dB. In the time-domain plot of S21 (right), the response is obtained by applying an inverse Fourier transform (IFFT), resulting in a short pulse. This indicates that the frequency content is concentrated within the 150 MHz to 250 MHz range. These results are consistent with the datasheet of the circuit training kit [11].

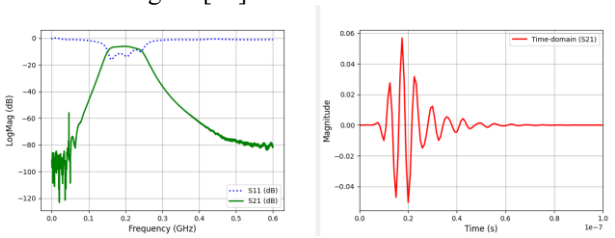


Fig 6. the measurement range of bandpass.

For the low-pass filter in Fig. 7, the solid line (left) shows a passband from DC to approximately 260 MHz. In the time-domain response (right), obtained via inverse Fourier transform (IFFT), low-frequency signals pass through while high-frequency components are attenuated. This demonstrates the typical behavior of a low-pass filter, where the response resembles a step voltage.

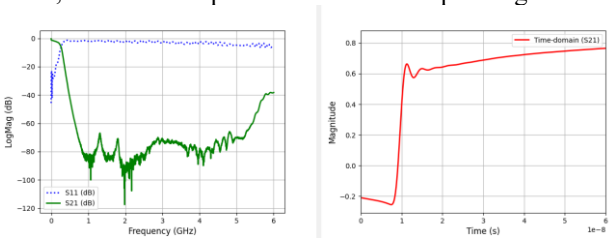


Fig 7. the measurement range of band-pass filter.

This software can be extended and further developed for suitable applications using Python coding. In comparison, other commercial software is ready to use for reading significant parameters, but it is not real time if data is imported via a flash drive for analysis. Therefore, this work is valuable for fostering open, customizable, and cost-effective research solutions in real-time medical measurements, such as tissue dielectric analysis and electrochemical sensors.

4. Conclusions

This paper presents the development of API software using Python for real-time graphical visualization of S11, S21, and time-domain analysis. While the approach itself is not novel, it offers a practical and accessible foundation for developers seeking to extend its capabilities to other applications, such as dielectric material characterization or biomedical tissue measurement, without the need for proprietary vendor software.

References

[1] H. C. Georgiana, B. Ana-Maria, and L. Ioan, “Automatic testing of automotive electronic modules for cranking conditions,” 2018 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2018, pp. 1–4, doi: 10.1109/ISETC.2018.8583856.

[2] C.-C. Chang, Y.-T. Chiu, and C.-C. Wei, “Design instrument control software interface based on SCPI commands to reduce development time,” 2021 7th International Conference on Applied System Innovation (ICASI), Chiayi, Taiwan, 2021, pp. 97–100, doi: 10.1109/ICASI52993.2021.9568467.

[3] A. S. Rao et al., “Development of Python-based applications for virtual instrument control using PyQt5, PyVISA, and SCPI protocol,” 2024 Second International Conference on Emerging Trends in Information Technology and Engineering (ICETITE), Vellore, India, 2024, pp. 1–7, doi: 10.1109/ic-ETITE58242.2024.10493634.

[4] J. L. Schmalzel and R. Trafford, “SCPI: IoT and the déjà vu of instrument control,” 2021 IEEE Sensors Applications Symposium (SAS), Sundsvall, Sweden, 2021, pp. 1–6, doi: 10.1109/SAS51076.2021.9530061.

[5] F. Zaiming, Z. Zhixiang, Z. Yijiu, and M. Min, “The merging design method of instrument software based on the SCPI command set,” 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), Yangzhou, China, 2017, pp. 44–48, doi: 10.1109/ICEMI.2017.8265709.

[6] B. A. Balaji, S. Sasikumar, and K. Ramesh, “SCPI-based integrated test and measurement environment using LabVIEW,” IOP Conference Series: Materials Science and Engineering, vol. 1045, no. 1, p. 012036, 2021, doi: 10.1088/1757-899X/1045/1/012036.

[7] W. Cai, B. Wang, and S. Zhang, “Remote control and data acquisition of multiple oscilloscopes using LabVIEW,” 2017 International Conference on Computer Technology, Electronics and Communication (ICCTEC), Dalian, China, 2017, pp. 920–924, doi: 10.1109/ICCTEC.2017.00203.

[8] Pico Technology Ltd., “PicoVNA Vector Network Analyser (PicoVNA 5) user’s guide,” [Online]. Available: https://www.picotech.com/download/manuals/picovna-vector-network-analyser-picovna-5-users-guide.pdf. [Accessed: Jul. 7, 2025].

[9] Pico Technology Ltd., “PicoVNA Vector Network Analyzer programmer’s guide,” [Online]. Available: https://www.picotech.com/download/manuals/picovna-vector-network-analyzer-programmers-guide.pdf. [Accessed: Jul. 7, 2025].

[10] Pico Technology, “picovna5-examples: Example code for PicoVNA 5 SDK,” GitHub repository, 2023. [Online]. Available: https://github.com/picotech/picovna5-examples.

[11] Pico Technology Ltd., “Network Metrology Training Kit user’s guide (PQ186),” [Online]. Available: https://www.picotech.com/download/manuals/network-metrology-training-kit-users-guide.pdf. [Accessed: Jul. 12, 2025].