RESEARCH ON BUILDING SIGNAL PROCESSING ANALYSIS MODULE USING RTL-SDR

Trung Thanh Nguyen, Ngoc Thang Bui*

Military Technical Academy, Vietnam

*Corresponding author: ngocthang020901@gmail.com

(Received: July 21, 2025; Revised: March 09, 2026; Accepted: March 26, 2026)

DOI: 10.31130/ud-jst.2026.24(3).380

Abstract - The research focuses on the development and evaluation of a radio signal processing and analysis module, utilizing RTL-SDR equipment in combination with a Raspberry Pi 4 Model B embedded computer. By applying signal processing algorithms such as the Fast Fourier Transform (FFT), FIR/IIR filters, and CFAR detection techniques, the software was tested using signal data captured across various frequency bands. The results demonstrate that the system is capable of detecting real signals with high speed and accuracy. Based on these findings, the study proposes solutions to enhance the application, including improving sensitivity and processing speed, integrating artificial intelligence to analyze complex signals, implementing remote control capabilities, and expanding the system's practical applications in both civilian frequency monitoring and military operations.

Key words - RTL-SDR; digital signal processing; frequency spectrum monitoring; fast Fourier transform (FFT); CFAR.

1. Introduction

With the remarkable development of technology, the number and types of devices using radio waves have increased rapidly. This trend has led to growing competition in the efficient utilization of the radio spectrum, which is a limited resource. At the same time, the demand for spectrum management and monitoring has become increasingly important in order to ensure the stable operation of devices, avoid mutual interference, and maintain the efficient use of frequency resources. Therefore, the development of flexible and efficient systems for radio signal monitoring and analysis has attracted significant attention from both the research community and technology practitioners.

Traditional radio signal reception and processing systems, however, often face many limitations in adapting to new requirements related to frequency, bandwidth, and modulation. These systems are typically built on fixed hardware, which is difficult to reconfigure, while upgrading or changing their functions usually requires high cost and considerable time. In this context, Software Defined Radio (SDR) has emerged as an ideal solution, allowing users to modify operating frequency, signal processing methods, and modulation schemes quickly and conveniently through software, without requiring substantial hardware intervention [1], [2].

The RTL-SDR device is a typical example of the application of SDR technology, with notable advantages including wide frequency range reception, compact size, low cost, and easy integration with various hardware and software platforms. For example, when combined with embedded computers such as the Raspberry Pi 4 Model B, RTL-SDR not only provides flexible deployment under different environmental conditions but also enables the implementation of complex signal analysis algorithms on a compact and low-power platform [1], [2]. With the objective of fully exploiting the advantages of SDR and RTL-SDR technology, this study proposes the development of a software module capable of real-time radio signal analysis, processing, and visualization. The module is designed to meet the requirements of civilian spectrum monitoring, supporting management tasks and the detection of signals that may cause interference or involve unauthorized spectrum use. The digital signal processing algorithms employed include the Fast Fourier Transform (FFT), FIR and IIR digital filters, and the CFAR technique, which significantly enhance the capability to detect and analyze signals in complex noisy environments.

Through the process of developing, testing, and evaluating the module in practice, this study focuses on clarifying the factors that directly affect the efficiency and stability of the system, including parameters such as gain, sampling rate, and FFT resolution. At the same time, the study also proposes improvement directions to optimize system performance, such as integrating artificial intelligence (AI) for the automatic identification of characteristic signal types and developing remote control and monitoring capabilities. These contributions will open up many new opportunities for the practical application of SDR technology and provide active support for spectrum management and related activities in the context of the rapid development of information and communication technology.

2. Overview of signal processing methods

2.1. Fast fourier transform

The Fourier transform converts a signal from the time domain to the frequency domain, thereby providing a clearer understanding of the frequency components of the signal. The Fourier transform is used in the design of digital filters to remove noise and preserve the desired components of the signal [3].

The Fourier transform of a continuous signal is defined by Equation (1) [3].

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi ft} dt \quad (1)$$

$x(t)$: signal in the time domain;

$X(f)$: signal after the Fourier transform in the frequency domain;

f : signal frequency.

However, in digital signal processing, signals are discretized in both time and frequency to facilitate computer-based processing. Therefore, the Discrete Fourier Transform (DFT) is applied as expressed in Equation (2) [3]. The DFT transforms a finite discrete signal sequence $x(n)$ of length N from the time domain into the discrete frequency domain:

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi}{N}kn},k=0,1,...,N-1 \quad (2)$$

$x(n)$: signal in the time domain (discrete and finite);

$X(k)$: discrete frequency spectrum;

N: number of signal samples;

k: discrete frequency index.

Nevertheless, one disadvantage of the DFT is its large computational complexity, especially for signals with a large number of samples. To address this issue, the FFT algorithm was proposed and has become one of the most effective tools in modern digital signal processing. The FFT algorithm reduces the number of required computations from $O(N^2)$ to $O(N\log N)$, thereby increasing processing speed and making real-time signal analysis feasible and efficient. The computational speeds of the DFT and FFT are compared in Table 1 [3].

Table 1. *Compare the speed of DFT and FFT [3]*

N	DFT (N^2)	FFT $(N\log N)$	Speed of increase
16	256	64	4 times
1024	1048576	10240	100 times
10^6	10^12	2*10^7	50 000 times

Table 1 shows that the FFT algorithm is significantly faster than the conventional DFT as the signal size increases. For N=16, the DFT requires 256 complex multiplications, whereas the FFT requires 64, making it four times faster. When N=1024, the DFT requires more than one million operations, while the FFT requires only about 10,240 operations, representing an approximately 100-fold improvement in speed. As N increases, the computational cost of the DFT becomes a practical barrier. In the case of N=10^6, the DFT must perform 10^12 operations, which is impractical for most modern hardware, whereas the FFT requires only about 10^7 operations, making it 50,000 times faster. Thus, the FFT not only reduces computational complexity but also transforms real-time signal applications from "impractical" to common.

2.2. Digital filters

Digital filtering is a fundamental and essential technique in digital signal processing, playing an important role in improving signal quality by removing unwanted noise components while preserving only the necessary signal components. Digital filters perform the filtering function by processing digitized signals through digital algorithms instead of using passive electronic components such as resistors, capacitors, and inductors. In digital signal processing, filters are commonly divided into two main groups: Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters.

2.2.1. FIR Filter

An FIR filter is a filter with a finite impulse response, meaning that its impulse response returns to zero after a certain period of time. The impulse response of an FIR filter is illustrated in Figure 1 [4], [5].

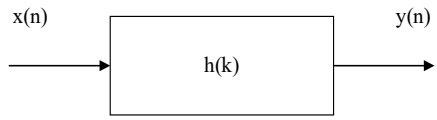


Figure 1. *Impulse response of an FIR filter [4]*

$$y(n)=\sum_{k=0}^{M-1}h(k).x(n-k) \quad (3)$$

$y(n)$: filtered output signal;

$x(n)$: input signal;

$h(k)$: impulse response of the filter.

M: filter order

2.2.2. IIR Filter

An IIR filter is a filter with an infinite impulse response, meaning that its impulse response does not return to zero but extends indefinitely over time. The structure of an IIR filter is shown in Figure 2 [4], [5].

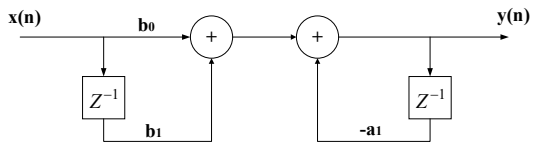


Figure 2. *Structure of an IIR filter [4]*

An IIR filter has an infinite impulse response, which means that the output depends not only on the current input but also on previous outputs, as described by Equation (4) [4].

$$y(n)=\sum_{k=0}^{M}b_k x(n-k)-\sum_{j=1}^{N}a_j y(n-j) \quad (4)$$

$y(n)$: output at sample n;

$x(n-k)$: input signal;

a_j, b_k : filter coefficients.

M, N: filter orders

2.3. CFAR (Constant False Alarm Rate)

CFAR is a signal processing technique widely used in radar systems and radio signal reception systems to maintain a stable false alarm probability, especially under conditions where background noise continuously changes. In practice, radio signals are often affected by environmental noise, thermal noise, and various other interference sources. Therefore, the use of a fixed signal detection threshold may reduce the ability to detect useful

signals or increase the false alarm rate. CFAR addresses this issue by automatically adjusting the detection threshold based on the noise level surrounding the signal position under observation.

Among CFAR methods, Cell Averaging CFAR (CA-CFAR) is the most common and widely applied approach due to its efficiency and high reliability. CA-CFAR operates based on the following principle: after preliminary processing, the input signal is divided into training cells, a cell under test (CUT), and guard cells. CA-CFAR uses the average value of the surrounding training cells, excluding the guard cells, to establish an appropriate adaptive detection threshold [6]. The block diagram of the CA-CFAR detector is presented in Figure 3.

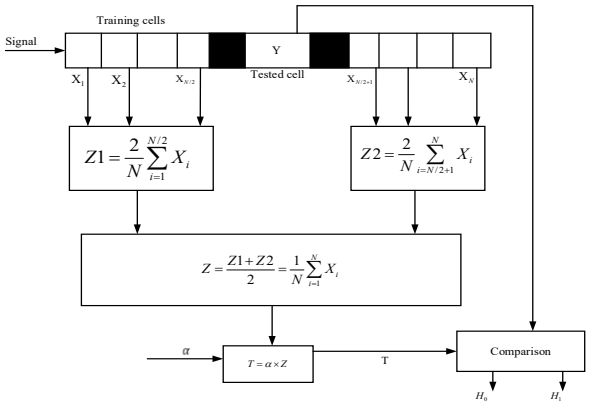


Figure 3. *Block diagram of CA-CFAR [6]*

The CA-CFAR detector operates on the principle that the received signals are stored in a buffer memory, where the average values of the left branch Z_1, the right branch Z_2 and the average value of both branches, denoted by Z, are calculated. This average value Z is then multiplied by a fixed constant α. The result after multiplication by the fixed constant is called the detection threshold [6]. The detection threshold is calculated according to Equation (5).

$$T = \frac{\alpha}{N}\sum_{i=1}^{N} X_i \quad (5)$$

where T is the detection threshold, N is the buffer length, X_i is the input signal amplitude, and α is a fixed constant. The fixed constant α is calculated according to Equation (6).

$$\alpha = N \times \left(P_{fa}^{-1/N} - 1 \right) \quad (6)$$

where P_{fa} is the probability of false alarm. The detection threshold T is then sent to the comparator, where it is compared with Y [6]. At the output of the comparator, the following cases are considered:

If Y > T: the comparator decides that a target is present.

If Y < T: no target is present.

3. Overview of the module

3.1. Overall architecture of the module

The signal analysis and processing module is developed based on the integration of RTL-SDR signal reception hardware and a Raspberry Pi 4 Model B embedded computer. In contrast to traditional spectrum analysis systems, which typically rely on expensive commercial spectrum analyzers, this system is based on self-designed digital processing blocks and a graphical user interface, thereby reducing cost while still meeting real-time signal processing requirements. The overall architecture is described in terms of two main layers:

Hardware layer: includes the receiving antenna, the RTL-SDR device (R820T2 tuner + RTL2832U), the Raspberry Pi 4 Model B, and the USB interface. This layer is responsible for signal reception, amplification, frequency mixing, digitization, and data transmission to the central processor.

Software layer: consists of four sublayers: (1) hardware interface layer, (2) buffer management layer, (3) digital signal processing core (DSP Core), and (4) graphical user interface (GUI) layer. This layer ensures control functions, signal filtering, spectrum analysis (FFT), data display, and visualization.

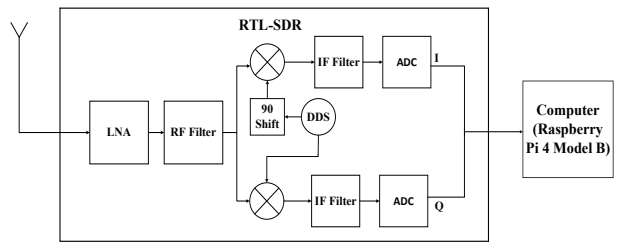


Figure 4. *Overview diagram of the module*

The RF signal received by the antenna is fed into the RTL-SDR device through the R820T2 tuner, where initial processing is performed, including amplification, frequency mixing, and filtering. Next, the signal is digitized by the RTL2832U chip and converted into I/Q data [7], [8]. This digitized I/Q data is then transmitted to the Raspberry Pi 4 embedded computer via the USB protocol. The Raspberry Pi is responsible for receiving the data and performing digital signal processing algorithms such as signal filtering, FFT and CFAR. At the same time, the Raspberry Pi also serves as the control unit for configuring the operating parameters of the RTL-SDR device, such as the center frequency, the number of FFT points, and the gain factor [7], [8], thereby forming a complete and flexible system that effectively supports practical radio signal processing applications. Figure 4 presents the overall diagram of the system.

Thanks to its layered architecture, the system achieves modularity, ease of expansion, ease of maintenance, and high customizability compared with commercial solutions, which are often closed systems.

3.2. Hardware components

3.2.1. RTL-SDR device

The module uses an RTL-SDR device capable of operating over a wide frequency range from 24 MHz to 1.76 GHz, providing flexibility for many common signal reception applications. The available reception bandwidth is 2 MHz, which is sufficient for processing various common types of signals such as FM broadcast signals,

television signals, or signals from wireless communication devices. The signal sampling rate of the system is stably set at 2.048 MSPS (Mega Samples Per Second), enabling the system to process data at high speed and effectively support real-time signal analysis. In particular, the ADC resolution of the RTL2832U chip is 8 bits, providing a sufficient level of detail for digital processing [7], [8], thereby ensuring the reliability and quality of the output signal during analysis and monitoring. Figure 5 shows a specific SDR device.

Figure 5. *RTL-SDR [8]*

3.2.2. Raspberry Pi 4 Model B

Figure 6. *Raspberry Pi 4 Model B*

The Raspberry Pi 4 is equipped with a 1.5 GHz quad-core ARM Cortex-A72 CPU, 4 GB RAM, high-speed USB 2.0 and USB 3.0 ports, and a GPU supporting graphics processing. This configuration is sufficient to run FFT algorithms with sizes of 2048 to 4096 points in real time. Compared with using a laptop computer, the Raspberry Pi offers the advantages of compact size, low power consumption, and easy integration into mobile systems or field deployment scenarios. Figure 6 shows a Raspberry Pi 4B device.

3.3. Software design

3.3.1. Structure and operating principle of the software

The signal analysis and processing software is designed according to a clearly layered model in order to ensure processing performance, stability, and ease of expansion and maintenance. The system structure can be divided into four main layers: (1) hardware interface layer, (2) buffer management layer, (3) signal processing layer (DSP Core), and (4) GUI layer. Data acquired from the RTL-SDR device through the USB interface are pushed into the buffer, processed, and then the final results are visualized and displayed to the user. The layered structure of the system is shown in Figure 7.

At the hardware interface layer, the software uses the pyrtlsdr library to initialize and control the RTL-SDR device. Users can configure parameters such as center frequency, sampling rate, and input gain. During operation, the device continuously sends blocks of digitized signal data (I/Q samples) at a rate of 2 million samples per second (2 MSPS). These data are acquired asynchronously and stored in a ring buffer managed by the software.

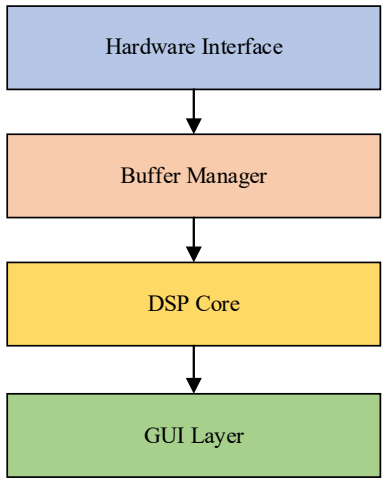


Figure 7. *Layered architecture of the software*

The buffer acts as an intermediate storage region that helps separate the data reading stream from the device (producer) and the signal processing stream (consumer). The buffer is designed with sufficient capacity to ensure that the data are not overwritten before being processed.

The signal processing layer (DSP Core) is the block that executes digital signal processing algorithms such as bandpass filtering, Fast Fourier Transform (FFT), and power spectrum calculation.

The GUI layer uses PyQt5 in combination with Qt Designer to design the graphical interface. The main interface consists of two parts: the control block (frequency, gain) and the graph display block.

3.3.2. Software interface

The software provides an intuitive adjustment tool through the QDial knob, allowing users to quickly change the center frequency value (No. 1, Figure 8).

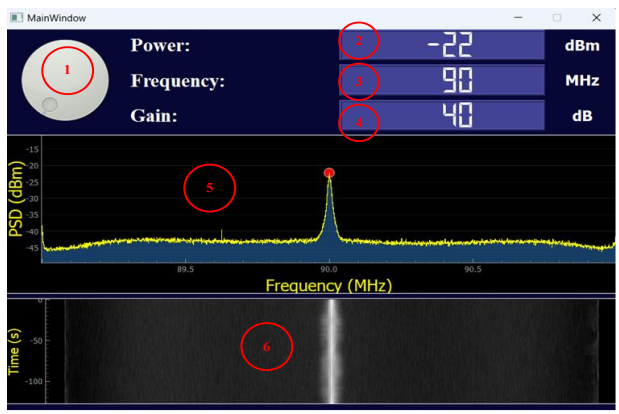


Figure 8. *Software interface*

Any change in value is immediately reflected on the digital displays (QLCDNumber), while also triggering

corresponding changes in the signal processing flow or hardware settings, thereby creating a smooth and responsive interactive experience (No. 3, Figure 8).

The received signal power level is displayed on the Power display (No. 2, Figure 8).

The input signal gain level is adjusted and displayed on the Gain display (No. 4, Figure 8).

One of the most important functions is the signal spectrum view, which is a graph showing the result after applying the Fast Fourier Transform (FFT) to the received I/Q signal. The horizontal axis represents frequency (MHz), while the vertical axis represents the power spectral density (PSD) in dBm. Strong signals appear as distinct peaks in the spectrum, allowing the operator to easily detect characteristic signals such as FM, digital modulation signals, and signals from transmitters or other devices in the environment (No. 5, Figure 8).

In addition, the software also displays a waterfall view, which helps observe spectrum variation over time. Each pixel row corresponds to an FFT slice at a specific moment, while the color represents signal intensity. This feature is particularly useful for detecting transient signals, rapidly varying signals, or frequency-hopping signals. All display components are continuously updated in real time (No. 6, Figure 8).

4. Experimental results

4.1. Laboratory setup

The experimental procedure was carried out as follows: the RTL-SDR device was connected directly to the receiving antenna to receive signals from a standard signal generator. A portion of the signal from the standard signal generator was fed to a spectrum analyzer for comparison with the results obtained from the system. The system was configured with specific technical parameters, including a gain in the range of 30–40 dB, a sampling rate of 2.048 MSPS, and an FFT size of 2048 points to ensure appropriate spectral resolution for detailed evaluation of the received signals. The measurements were conducted under normal laboratory conditions, avoiding unusual interference factors in order to ensure the objectivity and reliability of the results. The experimental arrangement is shown in Figure 9. IFR 2023A standard signal generator (No. 1, Figure 9), handheld spectrum analyzer (No. 2, Figure 9), RTL-SDR device (No. 3, Figure 9), and signal analysis and processing software (No. 4, Figure 9).

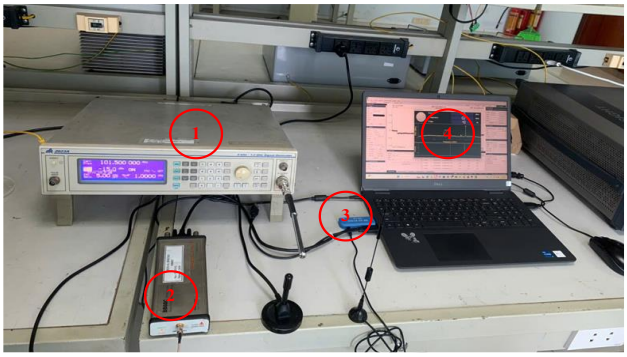


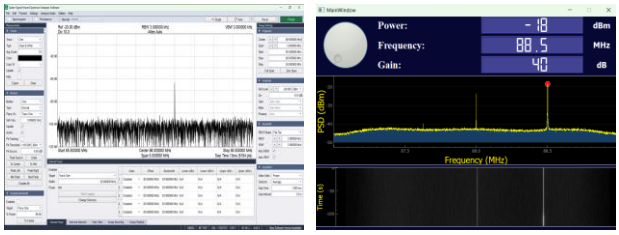
Figure 9. *Experimental setup*

The parameter settings on the IFR 2023A standard signal generator and the results obtained from the software are presented in Table 2. The standard signal generator output is shown in Figure 10. The signals received on the handheld spectrum analyzer and on the signal analysis and processing module at 88.5 MHz are shown in Figure 11, and at 101.5 MHz in Figure 12.

Table 2. *Transmit - receive parameters*

Transmitter		**Receiver**	
Frequency	Power	Frequency	Power
88.5 (MHz)	-16 dBm	88.5 (MHz)	-18 dBm
101.5 (MHz)	-15 dBm	101.5 (MHz)	-16 dBm

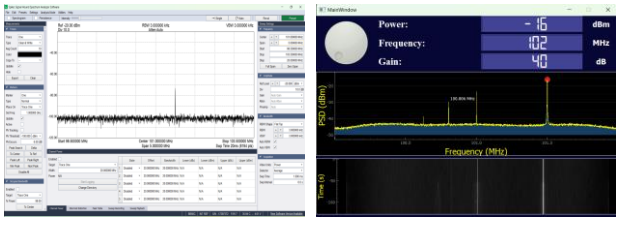
Figure 10. *IFR 2023A standard signal generator*



a: Spectrum analyzer *b: Software*

Figure 11. *Results obtained at 88.5 MHz frequency*

A comparison between the measurement results obtained from the signal analysis and processing module and those from the spectrum analyzer shows similarity in both the frequency and power level of the received signals. Specifically, at 88.5 MHz and 101.5 MHz, the signals received by the module were consistent with the results from the spectrum analyzer. The measured power levels showed only small deviations from the transmitted power due to impedance matching and propagation loss in the environment. This indicates that the system is capable of identifying and displaying radio signals and satisfactorily meets signal analysis requirements.



a: Spectrum analyzer *b: Software*

Figure 12. *Results obtained at 101.5 MHz frequency*

4.2. Real reception of VOV signals in Hanoi

In this section, reception tests were conducted for VOV radio signals in Hanoi. The VOV radio stations are summarized in Table 3. The receiver was located at

236 Hoang Quoc Viet, and the arrangement is shown in Figure 13.

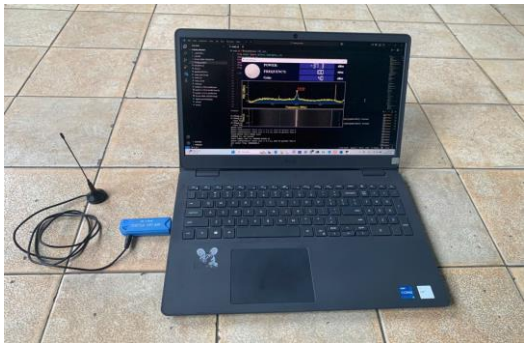


Figure 13. *Real signal reception*

Table 3. *Parameters of some VOV stations in Hanoi*

Station name	Frequency	Distance to receiver
VOV1	100 MHz	41-43 Ba Trieu, Hanoi (7.6 km)
VOV2	96.5 MHz	41-43 Ba Trieu, Hanoi (7.6 km)
VOV3	102.7 MHz	58 Quan Su, Hanoi (6.8 km)
VOV5	105.5 MHz	45 Ba Trieu, Hanoi (7.6 km)

The reception results for several VOV stations in the Hanoi area are presented in Table 4 and Figure 14.

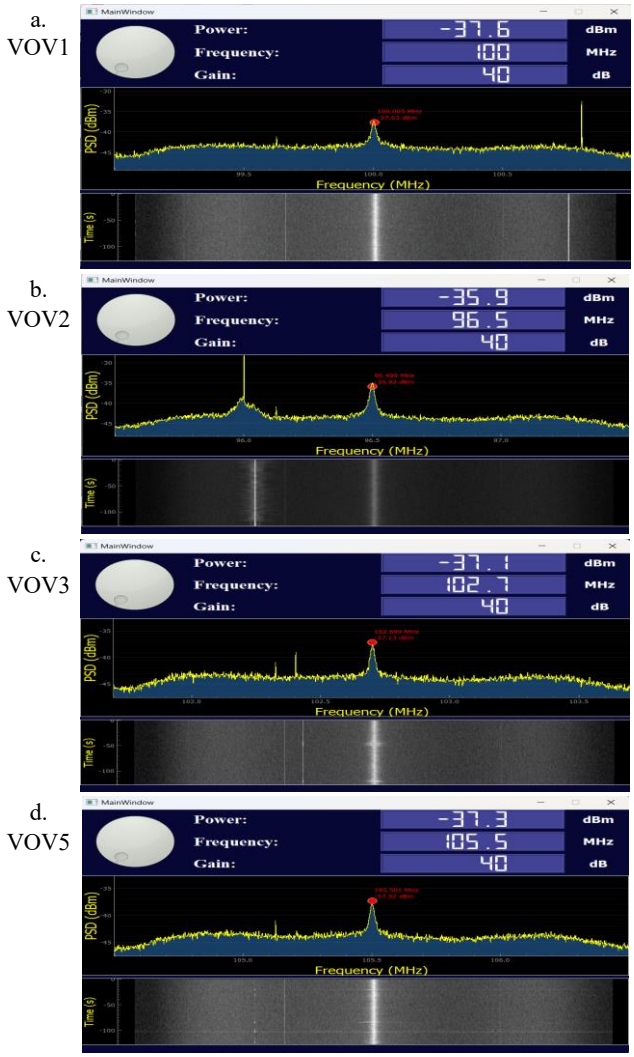


Figure 14. *Reception results of several real VOV signals*

Table 4. *Signal reception results of some VOV stations in Hanoi*

Station name	Frequency	Power
VOV1	100 MHz	-37.6 dBm
VOV2	96.5 MHz	-35.9 dBm
VOV3	102.7 MHz	-37.1 dBm
VOV5	105.5 MHz	-37.3 dBm

The experimental results from the module demonstrate its effectiveness in receiving signals from VOV radio stations in the Hanoi area. These signals were all accurately detected and displayed on the spectrum interface, demonstrating the system's practical applicability in signal monitoring and analysis.

4.3. Results, limitations, and novel contributions of the study

4.3.1. Achieved results

The experiments show that the module is capable of accurately receiving and processing radio signals in the VHF band. The system simultaneously displays both the power spectrum and the waterfall spectrum in real time, enabling rapid and intuitive monitoring of signal variations. When compared with a dedicated spectrum analyzer, the recorded power error is only on the order of a few dB, demonstrating high reliability while the total investment cost is significantly lower. The software interface is designed to be intuitive, allowing users to change parameters instantly while smoothly observing the corresponding response on the screen.

4.3.2. Limitations

Although many positive results were achieved, the system still exhibits several limitations. First, the 8-bit ADC reduces the dynamic range, making it difficult to analyze weak signals located near strong signals. Second, the 2 MHz bandwidth limitation means that the module does not yet satisfy the requirements for wideband signals. Finally, the processing capability of the Raspberry Pi remains limited; for large-size FFT operations (≥8192 points) or complex algorithms such as artificial intelligence, processing latency becomes clearly noticeable. This also motivates future improvement directions such as exploiting GPU, FPGA, or multiplexing techniques to enhance performance.

4.3.3. Novel contributions of the study

Compared with previous studies, this research offers several notable differences. First, the software is designed according to a four-layer architecture - hardware interface, buffer management, DSP core, and user interface - making the system modular and easier to maintain and expand. Second, the comparison results show that the performance of the module approaches that of a commercial spectrum analyzer, while the deployment cost is only a small fraction of that of such equipment, thereby increasing its applicability in civilian spectrum monitoring, education, and research. Third, the system is oriented toward further development with modern functions such as the integration of artificial intelligence for signal identification, long-term data storage mechanisms, and remote control capability, thereby extending its application scope to defense-related tasks such as electronic reconnaissance and early warning.

5. Conclusion

This study has successfully developed a radio signal analysis and processing module based on the integration of an RTL-SDR device and a Raspberry Pi 4 Model B embedded computer. Digital signal processing algorithms, including the Fast Fourier Transform, FIR/IIR digital filters, and the CFAR technique, were scientifically integrated and optimized, thereby enhancing the capability of signal detection and processing in real-world environments. One of the advantages of this study is the ability to develop the software independently instead of purchasing commercial software, which is often costly and inflexible. The software was fully developed and allows users to configure technical parameters such as gain, reception bandwidth, and sampling rate. Developing the software proactively not only helps reduce cost but also allows users to adjust and expand its functions according to practical needs, thereby creating a flexible, efficient, and easily updatable system.

Based on the practical experimental results, the study has demonstrated that the system operates stably with fast and accurate processing capability, while the interface is designed intuitively to facilitate the monitoring and management of received signals. Although positive results have been achieved, the study proposes several improvements to further enhance the efficiency and practicality of the system. First, the integration of data storage capability is proposed. The addition of this storage function would support post-measurement data analysis, enabling users to perform detailed evaluations more easily and preserve important results for research and practical applications. Next, the study recommends the application of artificial intelligence (AI) and machine learning to develop automatic signal recognition algorithms. The integration of AI would improve the ability to analyze, detect, and classify signal types while reducing the time and effort required from the operator. In addition, upgrading the interface is also considered an important improvement. The interface should include more intuitive control features, detailed information on the operating status of the system, and warning functions to help users promptly respond when problems arise. Finally, developing remote control and monitoring capability through the Internet or local networks would improve the deployment and management of the system at different locations without requiring direct physical presence, thereby broadening the application potential of the system in many fields such as civilian, scientific, educational, and military applications.

REFERENCES

[1] T. F. Collins, R. Getz, D. Pu, and A. M. Wyglinski, *Software-Defined Radio for Engineers*. Norwood, MA, USA: Analog Devices, 2018.

[2] J. M. Reyland, *Software Defined Radio: Theory and Practice*. Norwood, MA, USA: Artech House, 2024.

[3] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2007.

[4] L. B. Jackson, *Digital Filters and Signal Processing: With MATLAB Exercises*, 3rd ed. New York, NY, USA: Springer, 1995.

[5] D. Schlichthärle, *Digital Filters: Basics and Design*, 2nd ed. Berlin/Heidelberg, Germany: Springer, 2011.

[6] B. R. Mahafza, *Radar Signal Analysis and Processing Using MATLAB*. Boca Raton, FL, USA: CRC Press, 2008.

[7] C. Laufer, *The Hobbyist's Guide to the RTL-SDR: Really Cheap Software Defined Radio*, 4th ed. [S.l.]: RTL-SDR Blog, 2018.

[8] *The Complete RTL-SDR Handbook: Affordable Software-Defined Radio*, [S.l.]: Independently published, 2023.

[9] RTL-SDR Blog, "RTL-SDR Blog", *RTL-SDR.com*, 2018. [Online]. Available: https://www.rtl-sdr.com [Accessed June 2025].

[10] Osmocom Project, "RTL2832U Technical Documentation", *Osmocom.org*, 2020. [Online]. Available: https://osmocom.org/projects/rtl-sdr/wiki [Accessed June 2025].

[11] Raspberry Pi Foundation, "Raspberry Pi 4 Model B Specifications", *RaspberryPi.com*, 2021. [Online]. Available: https://www.raspberrypi.com [Accessed June 2025].

[12] NumPy Developers, "NumPy Reference Documentation", *NumPy.org*, 2023. [Online]. Available: https://numpy.org/doc/ [Accessed June 2025].

[13] SciPy Community, "SciPy Reference Guide", *SciPy.org*, 2023. [Online]. Available: https://docs.scipy.org [Accessed June 2025].

[14] PyQtGraph Developers, "PyQtGraph Documentation", *ReadTheDocs.io*, 2023. [Online]. Available: https://pyqtgraph.readthedocs.io [Accessed June 2025].

[15] Pyrtlsdr Project, "pyrtlsdr: Python interface for the RTL-SDR", *ReadTheDocs.io*, 2023. [Online]. Available: https://pyrtlsdr.readthedocs.io [Accessed June 2025].