Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра комп'ютерно-інтегрованих технологій, автоматизації, робототехніки та безпекової інженерії

I Всеукраїнська конференція «Інтелектуальні технології цивільної безпеки та робототехнічні системи аварійно-рятувальних робіт»

I All-Ukrainian Conference “Intelligent Civil Safety Technologies and Robotic Systems for Emergency and Rescue Operations”

ICSTRO

2026

I All-Ukrainian Conference

February 12 - 13, 2026

Kharkiv

УДК: 005:004.896:62-65:338.3

Інтелектуальні технології цивільної безпеки та робототехнічні системи аварійно-рятувальних робіт 2026: матеріали І-ої Всеукраїнська конференція, Харків, 12-13 лютого 2026 р.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)].-Харків: [електронний друк], 2026. – 192 с.

У збірник включені тези доповідей, які присвячені сучасним тенденціям розвитку технологій та засобів моделювання, прогнозування та управління ризиками у сфері цивільної безпеки; техногенна та виробнича безпека: технічні засоби, оцінка ризиків, експертиза; інтелектуальні та робототехнічні системи аварійно-рятувальних робіт; кіберфізичні системи, інформаційна безпека та цифровий захист виробництв; інформаційно-комунікаційні технології в системах управління та моніторингу надзвичайних ситуацій; сталий розвиток, екологічна безпека та соціальна відповідальність у сфері цивільної безпеки; інтелектуальні системи прийняття рішень у сфері цивільного захисту.

Intelligent Civil Safety Technologies and Robotic Systems for Emergency and Rescue Operations 2026: Proceedings of I st All-Ukrainian Conference, Kharkiv, February 12 - 13, 2026: Thesises of Reports / [Ed. I.Sh. Nevlyudov (chief editor).] .- Kharkiv .: [electronic version], 2026. - 192 p.

The collection includes the thesises of reports on devoted to current trends in the development of technologies and tools for modeling, forecasting, and risk management in the field of civil safety; industrial and technological safety, including technical means, risk assessment, and expert evaluation; intelligent and robotic systems for emergency and rescue operations; cyber-physical systems, information security, and digital protection of industrial facilities; information and communication technologies in emergency management and monitoring systems; sustainable development, environmental safety, and social responsibility in the field of civil safety; and intelligent decision-support systems in civil protection.

Харківський національний університет радіоелектроніки
Кременчуцький національний університет імені Михайла Остроградського
Національний університет «Запорізька політехніка»
Національний університет «Львівська політехніка»
Державне підприємство «Південний державний проектно-конструкторський та науково-дослідний інститут авіаційної промисловості»
Головне управління ДСНС України у Харківській області

Всеукраїнська конференція «Інтелектуальні технології цивільної безпеки та робототехнічні системи аварійно-рятувальних робіт» (ICSTRO-2026)

All-Ukrainian Conference “Intelligent Civil Safety Technologies and Robotic Systems for Emergency and Rescue Operations” (ICSTRO-2026)

COMPARATIVE ANALYSIS OF PROGRAMMING LANGUAGES FOR DEVELOPING SYSTEM USER INTERFACES

A. Fesenko, S. Sotnik
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av., 14
E-mail: alina.fesenko@nure.ua

***Annotation*:** The work considers the issue of choosing a programming language for developing a system's user interface. A comparative analysis of the programming languages C++, C#, and Python has been conducted in terms of their suitability for creating interfaces, taking into account the requirements for performance, ease of development, cross-platform compatibility, and the possibility of integration with system components. The main advantages and disadvantages of each language have been identified, and the areas of their appropriate application in developing interfaces for various systems have been outlined. The results obtained can inform a well-informed choice of software development tools.

Key words: user interface, programming language, C++, C#, Python, GUI, software systems.

ПОРІВНЯЛЬНИЙ АНАЛІЗ МОВ ПРОГРАМУВАННЯ ДЛЯ РОЗРОБКИ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ СИСТЕМ

А. О. Фесенко, С. В. Сотник
Харківський національний університет радіоелектроніки,
Україна, 61166, Харків, пр. Науки 14
E-mail: alina.fesenko@nure.ua

***Анотація*:** У роботі розглянуто питання вибору мови програмування для розробки користувацького інтерфейсу системи. Проведено порівняльний аналіз мов програмування C++, C# та Python з точки зору їх придатності для створення інтерфейсів, з урахуванням вимог до продуктивності, зручності розробки, кросплатформеності та можливості інтеграції з системними компонентами. Визначено основні переваги та недоліки кожної мови, а також окреслено області їх доцільного застосування при розробці інтерфейсів систем різного призначення. Отримані результати можуть бути використані при обґрунтованому виборі інструментів розробки програмного забезпечення.

***Ключові слова*:** інтерфейс користувача, мова програмування, C++, C#, Python, GUI, програмні системи.

In the context of widespread implementation of automation, monitoring, and control systems, the requirements for the quality and functionality of user interfaces are increasing [1-10]. The system interface must ensure clear data visualization, prompt response to operator actions, and stable operation during prolonged use [11-15]. Therefore, selecting a programming language for interface development is a crucial design stage, justifying the relevance of researching and comparing modern programming languages.

Before choosing a programming language and development environment for creating a system interface, it is necessary to define the main evaluation criteria upon which the comparative analysis will be conducted. These criteria enable an objective assessment of the capabilities of different technologies, their suitability for implementing a user interface, as well as their compliance with requirements for performance, reliability, and usability:

- performance and resources, because for the system interface, it is important how quickly the program responds to button presses, element movement, graph updates, displaying data from sensors,

and so on. It also matters how much memory and processor time the application consumes, especially if the system runs on a weak computer or performs other tasks in parallel (logging, signal processing, database work);

- development speed is an important criterion, since in real projects it is often necessary to first create an interface prototype, show it, collect feedback, and only then bring it to industrial quality. Therefore, it is important how quickly windows, forms, tables, diagrams can be created, styles configured, input validation and event handling logic implemented;

- the convenience of GUI tools is a significant criterion because a programming language by itself "does not draw" the interface – this is done by frameworks and tools. It is important whether there is a form designer, which elements are available (buttons, menus, tables, graphs, tabs), how easily styles can be configured, and whether there is support for modern approaches (templates, data binding, modularity);

- cross-platform capability, as the interface must work not only on Windows but also on Linux/macOS, making it important to choose a stack that ensures minimal differences in behavior and appearance, as well as simple building and deployment of the program on different operating systems;

- ecosystem and support, because it is important for the framework and language to have documentation, examples, an active community, and also for it to be easy to find ready-made solutions (libraries for graphs, serialization, network operations, hardware integration), which reduces the risk of «reinventing the wheel»;

- low-level integration is critical if the interface is part of a control system and requires interaction with microcontrollers, sensors, ports, and specific protocols, where it is important that working with system APIs, drivers, and COM/USB/Serial interfaces is as simple as possible, and that there is the ability to directly call the necessary operating system functions for full control over resources;

- reliability and security are fundamental requirements for system programs, since the application must run stably without crashes for hours or days, completely avoid memory leaks and dangerous errors, which is achieved through an appropriate memory model, the presence of automatic resource management, strict typing, and effective testing and debugging tools.

C++ is often chosen for a system interface when maximum efficiency, control, and integration with "hardware" or system components are required. It is a typical language for engineering applications, industrial control panels, CAD/CAE systems, simulators, and programs with heavy computations or critical latency (for example, fast real-time updates of sensor data graphs). From a UI perspective, C++ is most often used with Qt because Qt provides a cross-platform interface, a large number of ready-made elements (tables, graphs, style customization), as well as tools for creating forms. Other options also exist: wxWidgets for cross-platform, WinAPI/MFC for "classic" Windows, or Dear ImGui for rapid tool panels. The advantage of C++ is that it allows precise control over memory and resources, which is useful for systems that need to run stably and for long periods. The downsides are that UI development in C++ usually takes more time and requires more attention to architecture, resource management, and testing.

C# is one of the most convenient options for creating desktop interfaces, especially if the main platform is Windows. In C#, the ecosystem for UI is very well-developed: you can quickly create windows, menus, input forms, tables, lists, as well as more complex screens with data binding, templates, and styles. The most common approaches are WinForms and WPF. WinForms is simpler and allows for quick creation of a classic form-based interface, but its capabilities in modern design are limited. WPF is much more powerful: it supports modern styles, element templates, and the MVVM approach, which helps keep logic code separate from the visual part, which is useful for large systems. A separate plus of C# is a relatively low entry barrier, specifically for UI: many things are done in Visual Studio through the designer. Also, C# is usually easier to maintain in a team: less

«manual» memory management, a stable set of tools, and many examples. The downside can be the tie to Windows (if using WinForms/WPF) and more difficult integration with low-level components.

Python is often used when you need to quickly create an interface as a shell for scripts, automation, or data processing. It is a very popular option for prototypes: you can put together a working window in a short time, connect data reading, chart building, logging, and control buttons. There are several ways to create a GUI in Python. Tkinter is built into Python and allows you to quickly make a simple interface, but its appearance is often «basic». PyQt / PySide are the most powerful options because they have many ready-made components, good functionality, support for tables, tabs, and complex forms. There are also solutions like Kivy or Dear PyGui, which are convenient for certain types of tasks (for example, touch interfaces or tool panels). The main drawback of Python is performance and packaging it into a finished product. If the interface is large, complex, and needs to function as an industrial system, Python may require more optimization and careful consideration, and building an installer and deployment can be more complicated. But as a language for prototypes, internal tools, lab systems, and rapid automation, Python is a very strong choice. Table 1 presents a comparative overview of the programming languages C++, C#, and Python based on the key criteria that are important in system interface development.

Table 1 – Comparison of programming languages by criteria

Criterion	C++	C#	Python
Performance	Very high	High	Very low
GUI development speed	Average	Very high	High
Convenience of GUI tools	High	Very high	Average
Cross-platform capability	High	Low	High
Ease of maintenance	Low	High	Average
«Low-level», system APIs	Very high	Average	Very low

Based on the data provided in Table 1, it can be concluded that each of the considered programming languages has its own appropriate area of application in system interface development. Comparison based on key criteria allows for a clear tracing of the compromise between performance, development speed, and software maintainability. In particular, languages with a high level of resource control provide better performance and deeper integration with system components, but require greater implementation and maintenance complexity. Conversely, solutions focused on rapid interface development simplify the creation and maintenance of programs, but may have limitations in terms of performance or low-level integration capabilities. Thus, the results of the comparative analysis confirm that the choice of a programming language for implementing a system interface should be made taking into account the priority requirements of a specific project, particularly the target platform, load level, operational lifespan, and complexity of system interaction.

Given the results of the comparative analysis and the necessity to work with C++ in a cross-platform environment, Visual Studio Code with the PlatformIO plugin has been chosen for the development of the system interface, which provides convenient project organization, build automation, and support for hardware-oriented solutions.

Thus, for developing a C++ system interface, it is the strongest option if maximum performance, resource efficiency, and deep integration with operating systems or hardware are required, especially in industrial or engineering applications. C# is the most convenient choice for creating classic desktop interfaces under Windows due to its powerful development tools and rapid GUI creation, making it ideal for system applications, control panels, and business clients. Python is best suited for rapid interface prototyping or creating small tools and utilities, especially if the interface is related to automation or data processing; however, in large UI projects, it often loses out in terms of performance and maintenance convenience. Considering the above, it can be said that the choice of

language depends on the priority: maximum efficiency and control – C++, fastest development of a full-fledged UI for Windows – C#, fastest prototypes and utilities – Python.

REFERENCES

1. Lashyn, Z. V. & et al.. Automation capabilities of equipment with built-in robot for manufacture of microelectronics products. Proceedings of the XVII International scientific and practical conference «Information technologies and automation – 2024», 2024. PP. 283-286
2. Sotnik, S. Evaluating relational database scaling strategies in web engineering / S. Sotnik, M. Rudenko. International Conference on Advanced Trends In Radioelectronics and Infocommunications (ATRIC-2025) (May 21–22, 2025), Lviv Polytechnic Publishing House, Lviv, Ukraine, 2025. – PP. 224-228
3. Sotnik, S. V. Features of using REST architecture for development of ARS for information systems. Міжнародна науково-практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, 9–13 вересня 2024 р. Збірник праць. – Харків: ХНУРЕ, 2024. с. 42-45
4. Lykho, T. A. & et al.. Pattern recognition and computer vision technologies in decision support systems of robotic systems. Proceedings of the XVII International scientific and practical conference «Information technologies and automation – 2024», 2024. PP. 645-648
5. Tverdokhlib, A. & et al.. Intelligent tools for optimizing information and search engines. Manufacturing & Mechatronic Systems 2024: Proceedings of VIII st International Conference, Kharkiv, October 25-26, 2024. – PP. 28-31
6. Khalimonov, Y. & et al.. Approaches to ensuring proper working conditions using sensor technologies IoT. International Conference «DIGITAL INNOVATION & SUSTAINABLE DEVELOPMENT 2024», 2024 – PP. 24-25
7. Sotnik, S. V. Development of automated control system and registration of metal in continuous casting. Radio Electronics, Computer Science, Control, 2024. – PP. 197-211
8. Konieva, A. & et al.. Main trends in the development of automated image processing systems. «Computer-integrated technologies, automation and robotics» CITAR-2025, 2025. – PP. 68-72
9. Polikanov, K. A. & et al.. Overview of modern technologies for residential automation. «Computer-integrated technologies, automation and robotics» CITAR-2025, 2025. – PP. 85-89
10. Rudenko, M. Overview of algorithmic approaches to forecasting in CRM systems / M. Rudenko, S. Sotnik // «Computer-integrated technologies, automation and robotics» CITAR-2025, 2025. – PP. 101-105
11. Shrubkovskyi, Y. V. & et al.. Development of a structural scheme for automatic dosing of liquid components. Період трансформаційних процесів в світовій науці: задачі та виклики: збірник наукових праць з матеріалами V Міжнародної наукової конференції, м. Кропивницький, 6 червня, 2025 р. / Міжнародний центр наукових досліджень. – Вінниця: ТОВ «УКРЛОГОС Груп, 2025. – PP. 242-246
12. Marunich, R.V. & et al.. Modern IoT technologies for creating automated access systems. Sustainable smart cities and communities: business and innovation solutions 2025: Proceedings of I st I International Conference, Kharkiv, April 21, 2025: Thesises of Reports, 2025. – PP. 38-39
13. Yechevskyi, A., Maksymova, S. & et al.. Analysis of the data collection process about products at different stages of production. Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports, 2025. – PP. 38-41
14. Rudenko, M. & et al.. Classification of CRM systems. Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports, 2025. – PP. 42-45

15. Cherednichenko, T. & et al.. Features of automatic working time control systems. Manufacturing & Mechatronic Systems 2025: Proceedings of IX st International Conference, Kharkiv, October 25-26, 2025: Thesises of Reports, 2025. – PP. 54-57