

Meta: Discover this insightful guide to explore the most essential programming languages developers should consider.

<https://unsplash.com/photos/person-using-macbook-pro-on-brown-wooden-table-yVxUC9I9Cik>

The Top 4 Programming Languages For Embedded Systems

The significance of programming languages in embedded systems cannot be overstated, as they play a pivotal role in developing software that governs and oversees embedded devices. The selection of programming language can decipher the effectiveness and success of a project. So, when choosing a language for an embedded system, it's essential to consider various factors, including the capabilities and limitations of the target hardware, such as memory capacity, processing capabilities, and software demands.

This guide will delve into the essential programming languages crucial for embedded systems development, offering insights into their strengths, considerations, and real-world applications.

First, let's delve into the most essential programming languages for embedded systems development:

The Most Essential Programming Languages For Embedded Systems

1. C++

C++ is essential for embedded systems due to its efficiency, performance and flexibility, making it well-suited for resource-constrained systems and necessary for IoT and embedded tech. It is important to evaluate and consider project objectives, constraints, and access to required toolchains and libraries. The future of [embedded systems](#) development has started to see the introduction of hybrid approaches, where each language serves its own individual purpose, each coming with its own unique advantages and considerations.

Strengths

Below are some of the key strengths C++ can offer when developing embedded systems:

- **Compact and flexible** - In comparison to other higher-level programming languages, C++ is relatively compact, which is ideal for fitting into the restricted memory constraints of embedded systems. As C++ is a low-level programming language, it allows for accurate control of hardware interactions.
- **Performance** - By providing direct access to hardware, implementing efficient memory management and optimising data structures, C++ is highly efficient for performing critical tasks in real-time.
- **Platform independence** - C++ allows software developers to write their code once and run it on various embedded system platforms, avoiding the need for a complex cross-platform deployment process and creating shorter, more efficient development cycles.

Considerations

Although C++ has its strengths, it also comes with some limitations to consider:

- **Error-Prone** - Manual memory management elements incorporated within C++ mean that the risk of segmentation faults and memory leaks significantly increases, potentially leading to system crashes
- **Security issues** - C++ utilises a manual memory management process and allows direct hardware access, opening up a realm of security vulnerabilities, especially if the software has not been developed in a way that shields cyberattacks.
- **Integration challenges** - While C++ has many benefits, it faces modernisation challenges and struggles to integrate with new technologies and frameworks without requiring additional extensive libraries, significantly hindering innovation within the embedded systems industry.

Applications

Here are some of the real-time applications of C++:

- **Consumer electronics** - C++ is largely utilised in consumer electronics such as game consoles, wearable technology, and smart TVs, as it is ideal for optimising battery life and streamlining the multimedia playback process, making C++ a valuable tool to drive innovation within the industry.
- **Industrial automation** - To implement factory automation technology and robotic machinery, C++ is ideal for developing IoT features to maximise the speed and versatility of machinery, making it an ideal programming language for industrial automation purposes.
- **Automotive Systems** - C++ is well suited to keeping up with the demands of real-time requirements within the automotive field, particularly concerning safety features such as anti-lock braking systems and airbags, where real-time responsiveness is crucial.

To learn more about the importance of automotive-embedded systems and the essential skills employers should look for, discover our insightful guide - [Embedded Systems Recruitment in Automotive Engineering.](#)

2. Python

According to recent research, around [1.4%](#) of websites use Python as a programming language. Its user-friendly nature and extensive ecosystem render it a valuable asset for particular applications and swift development. It is essential to grasp both its strengths and limitations to seamlessly integrate it into projects involving embedded systems.

Strengths

Below are some of the key strengths Python offers:

- **MicroPython** - This Python 3 programming language implementation is designed to operate on microcontrollers and constrained systems. It successfully addresses resource limitations while offering enhanced libraries. It encourages immediate response to commands and rapid prototyping, making Python suitable for diverse embedded systems development.
- **Extensive Libraries** - Bare metal frameworks such as micropython and py-bricks provide vital extensive libraries to ensure direct interaction with hardware, allowing for accurate control. These specialised implementations are suitable for creating innovative embedded systems applications. The

Python Package Index also offers an extensive collection of external components with pre-written code for tasks such as data analysis, machine learning, and web development.

- **Cloud Integration** - Python's strong backing for cloud connectivity facilitates the straightforward and secure provisioning of embedded systems with the cloud, enabling the seamless transmission of data upstream. This proves especially advantageous for IoT devices and other systems seeking smooth integration with the cloud, allowing complex functions to be performed on resource-constrained devices.

Considerations

Although Python offers many advantages for embedded systems development, some limitations should be considered:

- **Real-time Processing** - It is important for employers to consider the specific function of the embedded system. In terms of real-time responsiveness, Python may not be the most suitable option for systems that require strict timing and response, as Python's latency is not the most sufficient for this.
- **Memory Management** - In Python, Automatic Garbage Collection (GC) is employed for memory management, where the interpreter releases unused memory while the program runs. While this streamlines development, it poses a challenge in real-time embedded systems due to the potential for latency spikes introduced by GC cycles. This is particularly critical in systems where quick response is vital and unpredictable pauses are unreliable.
- **Hardware Access** - Python is typically not the best option to go for when requiring direct hardware and low-level resource control in embedded systems, which significantly limits debugging and interfacing with particular hardware components. This means Python is less suitable for writing performance-critical code, which could be better when dealing with real-time responsiveness.

Applications

Now that we have discussed the strengths and considerations, let's delve into some of Python's real-world applications:

- **Industrial automation** - Python has various applications in industrial automation. For example, it is employed in High-Frequency Trading to automate trading strategies and swift buy and sell orders by analysing real-time market data. In Robotics Control, Python programs empower robots to adapt to their environment, utilising libraries like RoboPy.
- **Online gaming** - Python plays a key part in the functionality of online gaming. For example, it is used for various tasks such as AI, scripting and server-side logic. Python also offers an extensive variety of tools and frameworks used for building web frameworks and online game backends.
- **Video processing** - Python's simplicity and versatile features make it suitable for [electronic](#) tasks such as real-time video manipulation, frame-by-frame processing, and the implementation of advanced image processing methods.

3. Rust

When it comes to embedded systems programming languages, Rust is a suitable choice for embedded system development overall due to several advantageous features such as memory management, static analysis, and performance. However, it is equally important to consider the considerations and limitations to assess whether the language can meet specific project requirements.

Strengths

Take a look below to discover the key strengths that Rust offers to embedded systems development:

- **Memory Management and Security** - Rust offers an ownership and borrowing system, significantly reducing the risk of memory-related security issues and enhancing memory safety. Rust is a suitable choice in embedded systems development, particularly those that function on reliability and security.
- **Static Analysis** - Rust provides strong static analysis, meaning code is revised before being executed, ensuring code is suitable and meets specific requirements beforehand. Rust also offers powerful concurrency, meaning several tasks can be executed simultaneously, making it valuable for embedded systems in reducing errors and enhancing reliability.

- **Resource Management** - Rust can be used on bare-metal systems and real-time operating systems, meaning developers can specifically manage the use of resources, making it a suitable application for embedded systems that require close monitorisation and control of CPU usage and memory footprint.

Considerations

While Rust provides several advantageous features that benefit embedded systems development, it also comes with various drawbacks within the sector:

- **Compile Times** - When comparing programming languages, Rust has longer compliance times than languages such as C++. This is usually due to implementing a borrow checker, which negatively impacts the development workflow. This means Rust is not an ideal programming language for embedded systems that require quick responsiveness.
- **Low Hardware Support** - A further limitation of Rust as a programming language for embedded systems is limited interaction with specific hardware components, meaning developers are currently working on integrating lower-level drivers to advance this area.
- **Compatibility** - Although Rust offers a combination of C++ and existing code, the programming language needs more extensiveness, meaning the process of integration with other languages is not a simple process, posing challenges when it is necessary to merge pre-existing code.

Applications

Here are some key examples of the applications of Rust:

- **Industrial Automation** - Rust is a commonly used programming language for industrial automation, and it is particularly used in robotics and machinery, where memory safety and real-time responsiveness and analysis are critical.
- **Internet of Things (IoT)** - Rust's robust memory management and security are ideal for IoT devices, where memory vulnerabilities are common. Rust also offers static analysis and concurrency features, effectively reducing power consumption and making handling complex IoT deployments easier.

- **Automotive Embedded Systems** - In automotive systems such as safety-critical systems like airbags and anti-lock brakes, Rust's memory safety features are essential in reducing accidents caused by software bugs. Rust also offers several features used to develop driver assistance systems (DAS), where strong predictive performance contributes to efficient real-time performance. Recent research suggests that DAS will prevent an estimated [37 million](#) crashes over the next **30** years.

<https://unsplash.com/photos/black-mercedes-benz-vehicle-steering-wheel-SkKTh9ZyTxU>

4. Java

One of the key embedded systems programming languages is Java, which offers unique strengths depending on the functionality and specific requirements of the embedded system development cycle, particularly those focusing on maintainability and security. However, it is also important to consider potential limitations when evaluating development needs.

Strengths

Here are some strengths that label Java as one of the best programming languages for embedded systems:

- **Security** - Java offers various security features, such as sandboxing, to promote security in resource-constrained settings. For example, sandboxing limits applications from accessing system resources directly, reducing potential attacks from malicious code.
- **Object-oriented** - Java's object-oriented characteristics contribute to the organisation and modularisation of code, effectively encouraging reuse and maintainability. These qualities prove advantageous in the development of intricate embedded systems.
- **Cross-platform development** - Java provides various cross-platform capabilities, providing several benefits for embedded systems development, such as using existing code that can potentially be implemented across several systems with varied architecture, which is highly time and cost-efficient.

Considerations

Although Java is one of the go-to programming languages for embedded systems, usage comes with notable drawbacks:

- **Performance** - When comparing programming languages, Java has lower performance capabilities than C++ due to interpretation and garbage collection, leading to performance pauses. This is a huge limitation for embedded systems that require predictable timing.
- **Memory** - Java poses a risk to resource overhead, as The Java Virtual Machine (JVM) necessitates memory and processing power for its operation. This becomes a potential issue for embedded systems with constrained resources, specifically limited memory and processing capabilities.
- **Hardware Compatibility** - Java has low hardware compatibility for embedded systems, particularly those with low-level hardware features. Java's memory overhead and performance also contribute to further inhibiting compatibility.

Applications

Below are some of the real-world applications of Java in embedded systems:

- **Medical Devices** - Java has several reliability and security features, meaning it is one of the best programming languages for embedded systems in medical devices, such as infusion pumps and imaging equipment. However, it is important to assess beforehand whether real-time responsiveness is suitable.
- **Gateways for the Internet of Things (IoT)** - Java's extensive libraries and object-oriented qualities make it an ideal programming language for creating gateways that collect data from IoT devices and build complex systems within them.
- **Industrial Automation** - Java is commonly used in automation systems to control industrial machinery. Java allows for embedded systems development across multiple platforms and is object-orientated, meaning code can be reused, creating ease of development.

Final Thoughts on Programming Languages For Embedded Systems

The strengths of each programming language demonstrate their significance in embedded systems development. C++ stands out for its efficiency, performance, and platform independence, making it a preferred choice for resource-constrained systems in various industries such as consumer electronics, industrial automation, and automotive systems. Python's user-friendly nature, extensive libraries, and cloud integration capabilities make it well-suited for rapid prototyping and IoT applications despite real-time processing and hardware access limitations.

Rust offers robust memory management, security features, and resource management, which is advantageous for safety-critical systems in industrial automation, IoT, and automotive embedded systems. Meanwhile, Java excels in security, object-oriented design, and cross-platform development, making it a preferred language for applications in medical devices, IoT gateways, and industrial automation.

When comparing programming languages, it is crucial to acknowledge the considerations and limitations associated with each. From potential security vulnerabilities to memory management challenges, it's essential to carefully weigh these factors against project requirements. Despite these challenges, the applications of embedded systems programming languages in real-world scenarios demonstrate their versatility in addressing the complex demands of development.

Connecting Businesses to Top Talent in Programming Languages

Our recruitment specialists have the expertise and extensive networks to connect top talent to embedded systems programming language jobs. By tapping into our global contacts, we have the resources to meet your needs.

Whether you are a candidate looking to embark on a new career journey or a business looking for your next talented hire, [contact us](#) today to discuss your requirements.