AMBIENT-IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 02/04/2026

Register

Embedded Rust Training with Linux Kernel

3 days (21 hours)

Overview

Rust is a modern system language that provides strong guarantees in terms of memory safety, reliability, and performance, making it particularly well suited to industrial embedded systems. In environments where robustness and resource control are critical, Rust is gradually establishing itself as a credible alternative to C and C++ for the development of embedded system and application components.

This embedded Rust training course will teach you how to use Rust in modern embedded architectures, from interacting with hardware to developing robust system components.

You will learn how to compile, deploy, and run Rust applications on embedded platforms, interact with peripherals, and structure reliable and maintainable applications.

The approach is resolutely practical and production-oriented, with an emphasis on real industrial applications: hardware communication, reliability, performance, and operation. At the end of this training, you will be able to integrate Rust into your embedded projects and design robust solutions adapted to field constraints.

Like all our training courses, this one will introduce you to **the latest stable version** of the technology and its new features.

Objectives

- Use Rust in industrial embedded systems
- Develop reliable and high-performance components
- Interact with hardware and peripherals
- Structure and deploy robust embedded applications

Target audience

- Embedded systems engineers
- Firmware and embedded Linux developers
- C/C++ developers who want to use Rust

Prerequisites

- Proficiency in C or C++
- Basic knowledge of embedded systems
- Basic knowledge of Linux

Embedded Rust training with Linux Kernel

[Day 1 - Morning]

Modern embedded environments and Rust positioning

- Overview of current industrial embedded systems
- Differences between firmware, embedded Linux, and applications
- Rust's positioning compared to C/C++
- Industrial use cases: drivers, system services, IoT
- Hardware constraints: CPU, memory, performance
- Hands-on workshop: Preparing the embedded Rust environment.

[Day 1 - Afternoon]

Embedded Rust: toolchain and cross-compilation

- Understanding cross-compilation
- Rust targets for embedded systems
- Dependency management with Cargo
- Differences between std and no_std
- Organizing an embedded Rust project
- Hands-on workshop: Compiling and deploying an embedded binary.

System architecture and interaction with hardware

- Hardware architecture of embedded systems

- Access to peripherals and memory
- Concepts of drivers and hardware abstraction
- Interaction between Rust and the system
- Error handling and robustness
- Hands-on workshop: First hardware access from Rust.

[Day 2 - Morning]

Rust on the system side and user space

- Role of Rust in an embedded system
- Rust development in userland
- Interaction with the kernel and peripherals
- System calls and low-level access
- Memory safety and reliability
- Hands-on workshop: Rust application interacting with a peripheral device.

Drivers and hardware communication

- Principles of embedded drivers
- Hardware buses: SPI, I2C, UART
- Accessing peripherals from Rust
- Interrupt and event handling
- Hardware error handling
- Hands-on workshop: Developing a Rust application that communicates with a hardware device.

[Day 2 - Afternoon] Performance,

security, and reliability

- Rust performance in embedded environments
- Memory and resource management
- Memory security applied to systems
- Isolation, permissions, and hardware access
- Industry best practices
- Hands-on workshop: Analysis and optimization of an embedded Rust component.

Rust and embedded Linux kernel

- Linux kernel architecture in an embedded environment
- Role of the kernel and drivers in an embedded system
- Current status of Rust support in the Linux kernel

- Interaction between kernel space and user space
- Industrial use cases and current limitations
- Hands-on workshop: Loading and using a simple kernel module interacting with a Rust application

[Day 3 - Morning]

Structure and architecture of embedded software

- Modular organization of Rust components
- Logical separation of system and hardware
- Dependency and version management
- Embedded architecture patterns
- Maintainability and scalability
- Hands-on workshop: Structuring a complete embedded application.

[Day 3 - Afternoon]

Debugging, testing, and validation on target

- Embedded debugging and system diagnostics
- Logs, traces, and instrumentation
- Unit tests and system tests
- Validation on real hardware
- Fault and anomaly management
- Hands-on workshop: Debugging and correction on embedded targets.

Deployment, operation, and lifecycle

- Deployment and updating of embedded components
- Version management and compatibility
- Monitoring and operation in production
- Industry best practices
- Limitations and prospects of embedded Rust
- Hands-on workshop: Final deployment and full validation.

Target companies

This training is intended for both individuals and companies, large or small, wishing to train their teams in a new advanced IT technology or to acquire specific professional knowledge or modern methods.

Positioning at the start of training

The positioning at the start of the training complies with Qualiopi quality criteria. Upon final registration, learners receive a self-assessment questionnaire that allows us to assess their estimated level of proficiency in different types of technologies, as well as their expectations and personal objectives for the upcoming training, within the limits imposed by the selected format. This questionnaire also allows us to anticipate certain connection or internal security issues within the company (intra-company or virtual classroom) that could be problematic for the monitoring and smooth running of the training session.

Teaching methods

Practical training: 60% practical, 40% theory. Training materials distributed in digital format to all participants.

Organization

The course alternates between theoretical input from the trainer, supported by examples and discussion sessions, and group work.

Assessment

At the end of the session, a multiple-choice questionnaire is used to verify that the skills have been correctly acquired.

Certification

A certificate will be issued to each trainee who has completed the entire training course.