Training Course on Embedded Linux for Edge Computing

Professional Development Training Course Outline

Category	Engineering	**Duration**	10 Days
Base Fee	$3,000 USD	**Delivery Mode**	Online / On-site Hybrid

Course Introduction

Introduction

This intensive training course provides a deep dive into Embedded Linux for Edge Computing, equipping participants with the advanced skills necessary to design, develop, and deploy intelligent solutions at the network's edge. The curriculum focuses on leveraging the power and flexibility of Linux kernel customization, device driver development, and real-time extensions to build robust and responsive edge devices. Attendees will gain hands-on experience with popular embedded Linux platforms like Raspberry Pi and NVIDIA Jetson, mastering critical concepts such as containerization (Docker), secure boot, over-the-air (OTA) updates, and power optimization for resource-constrained environments. Training Course on Embedded Linux for Edge Computing is meticulously crafted to empower engineers to architect and implement high-performance, secure, and scalable edge computing solutions that drive innovation across various industries, from industrial automation to autonomous systems.

The program emphasizes practical application and industry best practices, exploring cutting-edge topics like AI/ML inference at the edge, vision processing, hardware acceleration, and seamless cloud integration. Participants will delve into advanced debugging techniques, performance tuning, and the deployment of complex applications using Yocto Project and Buildroot. By the end of this course, attendees will possess the expertise to design, optimize, and manage embedded Linux systems tailored for demanding edge computing scenarios, ensuring deterministic performance, data privacy, and efficient resource utilization. This training is indispensable for professionals seeking to advance their proficiency in the rapidly evolving landscape of edge AI and distributed intelligence, enabling them to build the next generation of smart, connected devices.

Programme Curriculum

Training Course on Embedded Linux for Edge Computing

Introduction

This intensive training course provides a deep dive into Embedded Linux for Edge Computing, equipping participants with the advanced skills necessary to design, develop, and deploy intelligent solutions at the network's edge. The curriculum focuses on leveraging the power and flexibility of Linux kernel customization, device driver development, and real-time extensions to build robust and responsive edge devices. Attendees will gain hands-on experience with popular embedded Linux platforms like Raspberry Pi and NVIDIA Jetson, mastering critical concepts such as containerization (Docker), secure boot, over-the-air (OTA) updates, and power optimization for resource-constrained environments. Training Course on Embedded Linux for Edge Computing is meticulously crafted to empower engineers to architect and implement high-performance, secure, and scalable edge computing solutions that drive innovation across various industries, from industrial automation to autonomous systems.

The program emphasizes practical application and industry best practices, exploring cutting-edge topics like AI/ML inference at the edge, vision processing, hardware acceleration, and seamless cloud integration. Participants will delve into advanced debugging techniques, performance tuning, and the deployment of complex applications using Yocto Project and Buildroot. By the end of this course, attendees will possess the expertise to design, optimize, and manage embedded Linux systems tailored for demanding edge computing scenarios, ensuring deterministic performance, data privacy, and efficient resource utilization. This training is indispensable for professionals seeking to advance their proficiency in the rapidly evolving landscape of edge AI and distributed intelligence, enabling them to build the next generation of smart, connected devices.

Course duration

10 Days

Course Objectives

1. Master Embedded Linux kernel configuration and compilation for specific hardware.
2. Develop and debug custom Linux device drivers for various peripherals.
3. Utilize Yocto Project and Buildroot for creating highly optimized embedded Linux distributions.
4. Implement secure boot and trusted execution environments on embedded Linux platforms.
5. Deploy and manage containerized applications (Docker/Podman) on edge devices.
6. Integrate AI/ML inference frameworks (TensorFlow Lite, ONNX Runtime) for edge AI applications.
7. Optimize system performance and power consumption for real-world edge deployments.
8. Implement robust over-the-air (OTA) update mechanisms for fleet management.
9. Develop network-aware applications for various connectivity options at the edge.
10. Apply real-time Linux patches (PREEMPT_RT) for deterministic control.
11. Troubleshoot and debug complex kernel and user-space issues on embedded Linux.
12. Design and implement vision processing pipelines on edge AI devices.
13. Leverage hardware accelerators (GPUs, NPUs) for optimized edge computing.

Organizational Benefits

1. Accelerated development of intelligent edge computing products and solutions.
2. Improved performance and reliability of embedded Linux-based systems.
3. Reduced development costs through efficient use of open-source tools and best practices.
4. Enhanced security posture of edge devices against cyber threats.

5. Faster time-to-market for new AI-powered and connected products.
6. Increased internal expertise in cutting-edge embedded Linux and edge AI technologies.
7. Optimized resource utilization on deployed edge devices, leading to lower operating costs.
8. Greater capability to innovate with custom hardware integrations and software features.
9. Competitive advantage in markets requiring real-time processing and local intelligence.
10. Development of more scalable and maintainable embedded Linux solutions.

Target Participants

- Embedded systems engineers
- Software developers
- Hardware engineers
- Network engineers
- System architects

Course Outline

Module 1: Introduction to Embedded Linux and Edge Computing

- **Embedded Linux Overview:** Advantages, components, comparison with RTOS.
- **Edge Computing Paradigm:** Definition, benefits, architecture, use cases.
- **Embedded Linux Distributions:** Yocto Project, Buildroot, Debian/Ubuntu Embedded.
- **Hardware Platforms for Edge:** Raspberry Pi, NVIDIA Jetson, industrial SBCs.
- **Case Study:** Analyzing the architecture of a smart factory edge gateway.

Module 2: Linux Kernel Basics for Embedded Systems

- **Kernel Architecture:** Monolithic vs. Microkernel, system calls, kernel modules.
- **Kernel Configuration and Compilation:** make menuconfig, .config file, cross-compilation.
- **Boot Process:** Bootloader (U-Boot), kernel loading, initramfs/rootfs.
- **Kernel Source Code Navigation:** Understanding drivers, arch, fs directories.
- **Case Study:** Compiling a custom Linux kernel for a Raspberry Pi.

Module 3: Device Driver Development Fundamentals

- **Character Device Drivers:** Registering, open, read, write, close operations.
- **Platform Devices and Drivers:** Managing devices on the bus, device tree overlays.
- **Interrupt Handling in Kernel Space:** IRQ controllers, ISRs, deferred work.
- **Memory Management in Drivers:** kmalloc, vmalloc, DMA buffers.
- **Case Study:** Developing a simple GPIO character device driver for a custom LED.

Module 4: Advanced Device Driver Topics

- **SPI, I2C, and UART Drivers:** Interfacing with common serial peripherals.
- **USB Device and Host Drivers:** Enumeration, URBs, class drivers.
- **Network Device Drivers:** Ethernet, Wi-Fi drivers, NAPI.
- **Block Device Drivers:** Interfacing with storage devices.
- **Case Study:** Writing a custom I2C sensor driver and integrating it with the Linux kernel.

Module 5: Building Embedded Linux Systems with Yocto Project

- **Yocto Project Overview:** Layers, recipes, bitbake, OpenEmbedded build system.
- **Setting up a Build Environment:** Poky, meta-openembedded.
- **Customizing Images and Packages:** Creating new recipes, adding features.
- **Layer Management and Best Practices:** Organizing custom layers.
- **Case Study:** Building a minimal embedded Linux image with a custom application using Yocto.

Module 6: Building Embedded Linux Systems with Buildroot

- **Buildroot Overview:** Simplicity, configuration options, package selection.
- **Buildroot Toolchain and Root Filesystem:** Understanding the build process.
- **Adding Custom Packages:** Integrating third-party applications.
- **Bootloader and Kernel Integration:** Configuring boot parameters.
- **Case Study:** Creating a lightweight embedded Linux system for a specific IoT gateway.

Module 7: Filesystems and Storage for Embedded Linux

- **Root Filesystems:** JFFS2, UBIFS, YAFFS2, Ext4, Read-Only Filesystems.
- **Flash Memory Technologies:** NOR, NAND, eMMC, SD cards.
- **Flash Translation Layers (FTL):** Managing flash wear and tear.
- **Overlay Filesystems:** UnionFS, OverlayFS for read-only rootfs with writeable overlay.
- **Case Study:** Configuring an embedded Linux system to use an optimized NAND flash filesystem.

Module 8: Embedded Linux Networking and Connectivity

- **Network Configuration:** ip command, netplan, network interfaces.
- **Wireless Connectivity:** Wi-Fi, Bluetooth, cellular (4G/5G) setup.
- **Network Services:** SSH, DHCP, DNS, NTP on edge devices.
- **VPN and Secure Tunnels:** Protecting data in transit.
- **Case Study:** Setting up an embedded Linux device as a secure network gateway with Wi-Fi and VPN.

Module 9: System Management and Debugging on Embedded Linux

- **System Initialization (systemd, SysVinit):** Services, targets, unit files.
- **Logging and Monitoring:** journalctl, syslog, performance metrics.
- **Debugging Tools:** GDB, strace, ltrace, perf, ftrace.
- **Memory Analysis:** valgrind, memleak, identifying memory issues.
- **Case Study:** Debugging a persistent application crash on an embedded Linux device using gdb and strace.

Module 10: Real-Time Linux and Determinism

- **Introduction to Real-Time Systems:** Hard vs. Soft Real-Time, latency.
- **PREEMPT_RT Patch:** Kernel preemption, priority inheritance.
- **Real-Time APIs:** POSIX Real-Time Extensions, sched_setscheduler.
- **Measuring Real-Time Performance:** Latency, jitter analysis.
- **Case Study:** Modifying a standard Linux kernel with PREEMPT_RT for a motor control application.

Module 11: Security in Embedded Linux Edge Devices

- **Secure Boot Chain:** Root of Trust, authenticated boot.

- **Disk Encryption:** dm-crypt, LUKS for data at rest.
- **Access Control:** Users, groups, sudo, SELinux/AppArmor.
- **Firmware Update Security:** Digital signatures, rollback protection, A/B updates.
- **Case Study:** Implementing a secure boot process on an NVIDIA Jetson platform.

Module 12: Containerization for Edge Applications

- **Docker Fundamentals:** Images, containers, Dockerfile, volumes.
- **Container Runtimes for Edge:** Docker Engine, containerd, Podman.
- **Orchestration at the Edge:** K3s, MicroK8s, Balena.
- **Resource Constraints and Optimization for Containers:** Minimizing image size.
- **Case Study:** Deploying a machine learning inference application inside a Docker container on an edge device.

Module 13: AI/ML Inference at the Edge

- **Overview of Edge AI:** Benefits, challenges, use cases.
- **AI Frameworks for Edge:** TensorFlow Lite, ONNX Runtime, OpenVINO.

Upcoming Scheduled Sessions

Start Date	End Date	Location	Price
21 Sep 2026	02 Oct 2026	Online	$2,000
21 Sep 2026	02 Oct 2026	Physical	$3,000
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0

Ready to enroll or request a customized program? Book online or contact us:

Register Online Now

Email: info@fineskilltrainingcenter.com | Website: www.fineskilltrainingcenter.com