Faysal Aziz

Embedded Linux and Device Driver Mastery

From Bootloader to Kernel Driver, Practical Embedded Linux on Real Beaglebone Black

FA-0184 · 3-day

DevOps, Cloud & Infrastructure

Introduction

Introduction

Embedded Linux systems power industrial controllers, robotics platforms, smart devices, and countless production systems. Understanding how Linux boots, how it interfaces with hardware, and how device drivers integrate into the kernel is no longer optional for serious embedded developers.

This 3-day intensive program focuses exclusively on practical implementation using the Beaglebone Black platform. Participants will move from understanding the boot sequence and filesystem construction to developing kernel modules, character drivers, platform drivers, and working with Device Tree. The training is delivered by an instructor with over 30 years of industry experience, ensuring the content reflects real-world engineering practices and industry demand rather than purely academic theory.

Learning outcomes

By the end of this course, participants will be able to:

- Explain and demonstrate the Linux boot process on the Beaglebone Black
- Configure bootloaders, kernels, and root filesystems
- Build minimal embedded Linux systems using Busybox and Buildroot
- Interface peripherals using GPIO and I2C

- Develop and compile Linux kernel modules
- Implement character device drivers
- Create and manage multiple device nodes
- Develop platform drivers
- Understand and integrate Device Tree with drivers
- Work with Device Tree overlays
- Use the Linux device model and sysfs within drivers

Prerequisites

Participants should have:

- Basic knowledge of Linux command line usage
- Basic C programming skills
- Familiarity with compiling programs in Linux
- Access to a Beaglebone Black board (recommended for hands-on practice)

Training outline

1. 1. Beaglebone Black Overview and Environment Setup

- Board architecture and hardware overview
- Peripheral interfaces and connectivity
- Serial communication setup
- Terminal tools configuration
- Ubuntu host setup using VirtualBox
- Host-to-target communication setup

---

2. 2. AM335x Architecture and Linux Boot Process

- AM335x functional block overview
- Boot requirements and boot modes
- ROM bootloader sequence
- SPL role and functionality
- U-Boot overview and configuration

- Linux kernel boot process
- Boot logs analysis

---

3. 3. Booting Techniques and System Configuration

- Booting Linux from eMMC
- Booting Linux from SD card
- Updating system images
- Serial boot procedure
- TFTP boot configuration
- Linux Device Tree basics
- uEnv.txt configuration
- initramfs overview

---

4. 4. Embedded Linux Tools and Build Systems

- Busybox architecture and compilation
- Busybox configuration
- Buildroot introduction
- Buildroot configuration
- Generating minimal root filesystems

---

5. 5. Peripheral Interfacing on Beaglebone Black

- GPIO subsystem overview
- GPIO programming techniques
- I2C subsystem overview
- Using I2C tools
- LED interfacing
- 7-segment display interfacing
- LCD interfacing
- MPU6050 sensor interfacing
- Tilt sensing implementation
- EEPROM programming

---

6. 6. Device Driver Development Environment

- Host system preparation for driver development
- Toolchain setup
- Disk partitioning using gparted
- Kernel preparation for module development
- Target board setup for driver testing

---

7. 7. Linux Kernel Module Fundamentals

- Kernel space vs user space
- Structure of a Linux kernel module
- Module initialization and exit macros
- Writing a basic Hello World module
- Compiling kernel modules
- Makefile structure for kernel modules
- Inserting and removing modules
- Inspecting module information

---

8. 8. Character Device Driver Development

- Introduction to character devices
- Allocating device numbers
- Registering character devices
- File operations structure
- Implementing open function
- Implementing read function
- Implementing write function
- Implementing lseek
- Creating device files
- Testing character drivers
- Cleanup routines and error handling

---

9. 9. Pseudo Character Drivers and Multiple Devices

- Designing a pseudo character driver
- Memory management within driver
- Handling multiple device instances
- Creating multiple device nodes
- Testing multiple character devices

---

10. 10. Platform Drivers

- Platform bus architecture
- Platform device registration
- Platform driver registration
- Matching devices and drivers
- Implementing platform driver probe
- Implementing platform driver remove
- Testing platform drivers

---

11. 11. Device Tree Fundamentals

- Device Tree purpose and structure
- Node hierarchy
- Properties and syntax
- Parent and child node relationships
- Compiling Device Tree sources

---

12. 12. Device Tree and Driver Integration

- Binding drivers with Device Tree
- Writing compatible strings
- Parsing Device Tree properties in drivers
- Testing Device Tree based drivers

---

13. 13. Device Tree Overlays

- Concept of overlays
- Overlay creation
- Integrating overlays with bootloader
- Testing overlays on target

---

14. 14. Linux Device Model and Sysfs

- Overview of Linux device model
- kobjects fundamentals
- sysfs structure
- Creating sysfs entries in drivers
- Defining driver attributes
- Testing sysfs attributes

© 2026 Faysal Aziz · Kuala Lumpur