INDIA's one & only
INDUSTRIAL Embedded Systems Training Institute
KERNEL MASTERS

Courses

Course Name	Duration	Fees	Syllabus	Eligibility
KERNEL MASTERS Certified Embedded Systems Professional. (KMESD) (Realtime projects training)	5 Weekends	16,000 /-	Modules 1 - 3	Working professional (or) Non Fresher's

All sessions are highly interactive & hands-on with hardware.

Contact Numbers: 9949062828, 9963111083
Mail ID: info@kernelmasters.org
Directions: https://www.google.com/maps/place/Kernel+Masters/

Detailed Syllabus

Module 1: Linux Device Drivers — Duration: 32 Hours

Introduction to Linux Kernel
- Two Roles of Kernel.
- Kernel Programming
- Linux Source tree Overview
- /proc and /sys virtual file system.

Introduction to Device Drivers
- What is Device Driver?
- Types of Device Drivers
- Classes of Device Drivers
- The Role of the Device Driver
- Types of Kernels
- Configuring, Compiling and Booting the Linux Kernel
- Kernel Configuration
- Booting the kernel.

Module Programming
- What is a Kernel Module?
- User mode vs Kernel mode
- Our First Linux Driver
- Module parameters

Character Device Drivers
- The Complete connection.
- Major and Minor numbers.
- Implementation of Character Driver.

Concurrency and Race Conditions
- Concurrency and Its Management
- Semaphores and Mutex
- Completions
- Spinlocks

Advanced Character Device Drivers
- ioctl
- Blocking I/O
- poll and select
- Asynchronous notification

Communicating with Hardware
- I/O Ports and I/O Memory
- An I/O Port Example

Interrupt Handling
- Installing an Interrupt Handler
- Implementing a Handler
- Top and Bottom Halves
- Interrupt Sharing
- Interrupt-Driven I/O

Kernel Mechanisms
- Kernel Threads
- Kernel Timers
- Delaying Execution
- Tasklets
- Workqueues

Network Device Drivers
- Implementation of network Device Drivers

Adding a Driver to the Kernel Tree
- kernel layout for drivers
- Modifying the Makefile
- Adding it to configuration options - the Kconfig file

Module 2: Embedded Linux — Duration: 32 Hours

Embedded Linux Development Environment

Introduction to embedded Linux.
- Embedded Hardware and Software.
- C libraries. Building a cross-compiling tool chain.
- Setting Up a Target Development Board.
- X86 vs Embedded Boot Sequence.

Boot loaders
- Board Support Packages
- U-boot commands and Environment variables.
- U-boot Source code flow
- U-boot Customization
- Device Trees

Linux Kernel
- KBuild System
- Configuring, (cross) compiling and booting a Linux kernel
- Kernel boot-up flow

Root File System
- Creating a simple, Busybox based root file system from scratch
- Flash file systems – Manipulating flash partitions
- mmc vs emmc
- NAND vs NOR
- Developing and debugging applications for the embedded system
- Implementing real-time requirements
- Boot time optimizations

Embedded Linux Build System
- Buildroot - Configuring Buildroot for bbb
- Yocto **Project/Open Embedded** Poky and bitbake.

Developing Embedded Linux Device Drivers		
BSP/Bootloader ▪ Board bring up ▪ Device Tree ▪ RTC, Clock, Timers, watchdog and Interrupts. ▪ Mux Configuration **GPIO Driver** ▪ Linux GPIO Management ▪ Accessing GPIO interfaces	**Linux Input Subsystem** Matrix Keypad **Communication protocols** ▪ UART Framework ▪ I2C Framework ▪ SPI Framework ▪ CAN Framework	**Sensors** ▪ Touch panel ▪ Proximity ▪ Accelerometer **Display/LCD** Frame buffer Architecture **Ethernet Drivers**
Embedded Linux Test Environment		
▪ Setup Linux Test project (LTP) on Kernel Masters Embedded Linux Development Board. ▪ Self-Diagnostic Tool. ▪ Developing Linux Device Drivers Test Cases using Shell & Python Scripting.		
Embedded Linux Troubleshooting		
▪ u-boot Level ▪ Kernel Level ▪ User Level		

Module 3: Debugging Techniques		Duration: 8 Hours
User space tools ▪ GDB, ▪ gdb server ▪ Strace ▪ /proc & /sys ▪ Valgrind	**Android Debugging** ▪ adb ▪ logcat ▪ avd **Remote Debugging** ▪ ssh ▪ Ethernet	**Kernel Space tools** ▪ Printk ▪ Kernel Panic ▪ Kernel OOPS ▪ KDB ▪ KGDB ▪ Kprobes & Jprobes ▪ Crash dump Analysis