Embedded Linux Engineering

Training Highlights:

- Learn through Practical's.
- Work on Latest ARM Cortex Processors

  A5/A7/A8/A9/A15/A17/35/53/55/72
- Open Source Projects Development
- Assured Post Training Support
- Unlimited Access to the Hardware Boards **vLAB**
- Lifetime access to LMS eLinux Module
- Bi-Weekly Interaction with Industry Guru's

Module-1: Linux Internals

Linux Intro & Installation	- What is Linux, how it has been evolved, GNU License, Kernel - How Linux was designed, - Sub systems of Linux [Scheduler, Process, Memory Mgmt, File System, Device Mgmt] - Ways to Install Linux [1. Dual Boot, 2. Within Windows, 3. Using Virtual Machine] - How to update Linux and install required packages
Linux Shell Commands	- Basic Commands - Dir & File Commands - System Commands - Misc Commands
Shell Scripting	- Writing Basic Linux Shell scripting - Variables & Operators in Shell scripting - Command Line Arguments - Logical Structures in Shell Scripting
C Programming in Linux	- Writing C program on Linux - Compiling and executing Linux - Linux Executable format info & tools - Debugging C application on Linux using GDB
Make Files	- Understanding Make files - Writing Make files - Compiling Multiple src Dir using Make file - Advanced methods used in writing Make files
Process Management	- Understanding Linux Process - How to create child process using [system, exec, fork & clone] - Managing Linux process
File Operation	- How to write application to access files in Linux - System Calls used in Linux to control special files like device nodes

Signals	- How to write a serial port access program in Linux - Signals in Linux - Registering & Handling Signals - Implementing new Signals
Linux Scheduler & Memory Management	- Linux Kernel Scheduling Policies - Scheduler System calls - MMU Subsystem - Understanding Virtual Memory Concept - System calls for Memory Management
Linux Multi-Threading Programming	- Basics of Multithreading in Linux - How to create multi-threading applications in Linux - Managing & communication between Multiple threads
Inter Process Communication	- Data sharing between Multiple processes using IPC Mech. - Writing apps using PIPEs, FIFOs, Msg Queues, Shared Memory
Network Programming in Linux	- How to develop client server-based network application in Linux - When and how to use TCP and UDP Protocols

Module-2: eLinux Porting

Introduction, Setup & Hardware	- Introduction to Embedded Linux - ARM Processor Basics & Families - ARM Board Details and Schematic Overview - Boot Process - Host PC Setup for eLinux Development
Toolchain & Hardware Practical's	- Board Boot Options - Flashing Bootloader & Linux Kernel on Board - Setting up TFT and Running Application on Board - Toolchain & its components

	- How to build toolchain
Bootloader U-Boot	- Introduction to Bootloader - Primary Bootloader (TI X-Loader) - Bootloader Commands and their usage
U-Boot Porting	- Bootloader Source Code Structure - Compiling Bootloader - How to port Bootloader on ARM Based Hardware - Patching Bootloader
Customizing Bootloader	- Modifying Bootloader for new feature - Modifying Bootloader to support new device - Command Line Arguments & ATAG - Booting with SD Card - Setting up NFS Server - Booting with NFS Server - Linux Kernel Compilation
Linux Kernel	- Introduction to Linux Kernel Arch - Kernel Dir Structure - Kernel Layers H/W dependent and independent (BSP) - Kernel Build System (KConfig)
Kernel Porting & Compilation	- How to configure and compile for ARM Hardware - Type of kernel images (vmlinux, zImage, uImage) - Kernel initialization process - How to port Kernel on New ARM Hardware
Kernel Modification	- How to modify the Kernel code - How to integrate new driver / module in kernel image - Building static and dynamic kernel modules
Root File System	- Components of RootFS -Types of RootFS -Different types of Flash Device (NOR / NAND)

	- Building RootFS from scratch and using Build System (Buildroot)
Embedded Application Development	- How to develop embedded applications - Debugging application on target using GDB - Running sample Web-Server Application - Using Eclipse for embedded application development

Module-3: Linux Device Drivers

Introduction and Arch of Linux Device Drivers	- Introduction to Kernel Space and User Space - Memory mgmt in Kernel - How to develop Kernel Device Driver - Layers of LDD - Processor Memory Layout - Device Register Access from Code
Kernel Module Programming	- Kernel Module Programming - Module Parameters - Exporting Symbols between modules
Character Device Drivers	- Linux Kernel Device Driver Framework - Virtual File System as bridge between Driver and Application - Implementing basic character driver - Writing Makefile to compile Device driver - Compiling and running on X86 - Cross Compiling and running on ARM Hardware - Implementing advance api like ioct in character device driver - Standards to follow while implementing ioctl - Writing and testing LED driver with IOCTL on ARM Hardware
Interrupts in Device Driver	- Interrupts in ARM Processor - Interrupts Mechanism in Linux Kernel - How to implement Interrupts in device driver

Interrupt Handling & Bottom Half	- Writing and testing Interrupt for Button press on ARM Target - Writing and testing multiple Intterupts in single driver - How to implement Shared Interrupts - How to handle lengthy ISR using Bottom Half (Soft IRQ, Tasklet & Workquees)
Special File Systems ProcFS & SysFS	- Ram based files systems in Linux - Using procfs for special purpose and accessing kernel data structure - How to implement procfs - Sysfs implementation in device drivers for easy application access.
LDDM (Linux Device Driver Model)	- Platform Data - Platform Device - Platform Driver - Modify SLED Driver as platform driver
Board File	- Structure of Board File - Writing a simple Board File and testing on Board
Device Tree	- Understanding Device Tree Structure - Nodes in DTS - Properties of Nodes - Device Tree examples: memory-mapped devices, I2C, SPI, pin-muxing, clocks, etc. - Kernel API's to process device tree data - Compiling Device Tree and Flashing
Advance Device Drivers	- Walkthrough **MMC** domain in AM335x & its implementation - Lab Add SD-CARD support to Board file and enable root file system to be mounted from SD-Card partition. - Understanding **UARTs** in AM335x and its driver components - Lab Modify Board file to configure UART-2 & UART-3 on WEGA Board and test it using Linux user application. - Intput Subsytem in Linux - Lab: Modify Board file to Configure Switches on WEGA board to generate input events & test

Advance Device Drivers	it from user app. - I2C Subsytem in Linux - Lab: Modify Board file to add support of i2c based EEPROM or RTC and test it using user app. - SPI Subsytem in Linux - Lab: Modify Board file to add SPI based External ADC device to WEGA Board and test it from user app. - Display Sub-System in Linux - Lab: Configure the 7” LCD Display and test it using fbtest utils in linux. - Introduction to block and network device drivers - Case study of Network Device Drivers - Debugging Techniques like debugfs / target debugging

Module-4: Yocto

Yocto Architecture	- Bitbake - OpenEmbedded Core - Poky reference project - Configuration files - local.conf - machine.conf - distro.conf - Bitbake usage
Recipes defines everything in Yocto	- Understanding Recipes - Recipe tasks - Writing new recipe

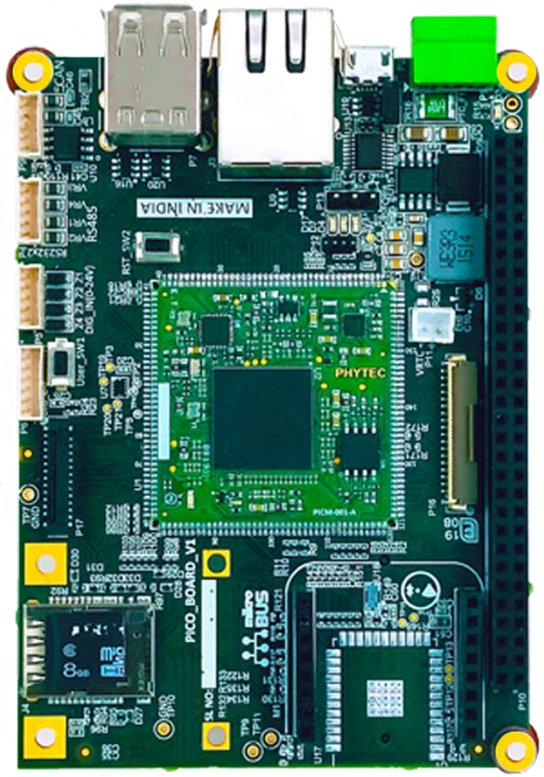
Layers makes Yocto Modular & Structured	- Basic examples recipes(hello.bb) - Customizing recipes - Customizing existing recipes (.bb) - Create .bbappend file for existing recipe - Yocto recipe classes - Layer in Yocto - Creating new Layers - Using existing Layers - Git usage in yocto - Create and apply patches
Adding new Hardware support using BSP Layers	- **Bootloader** - Adding bootloader to machine - Create and Apply patches to bootloader - Setting New defconfing to bootloader - Providing configuration fragments - Adding new bootloader versions to machine - **Kernel** - Adding linux kernel to machine - Create and Apply patches to kernel - Provide new defconfing to kernel - Providing configuration fragments - Adding new kernel versions to machine - **Building Root File System** - Modify existing rootfs image recipe - Selecting types of root file system images - Integrating IOT Packages to Yocto (MQTT, libcoap...) - Menu config support for kernel and Bootloader

Custom Distribution & Images	- Understanding Distro Layers - Image types - Custom Image type for custom Board & Applications - Linux package management tools - Package groups - Package release versioning
Creating SDK using Yocto for Application Development	- Generating SDK using Yocto - Using SDK for Application development

Industrial Single Board Computers on Discounted Price for Participants

RuggedBOARD-A5D2x

[Low Cost Industrial IoT Gateway]



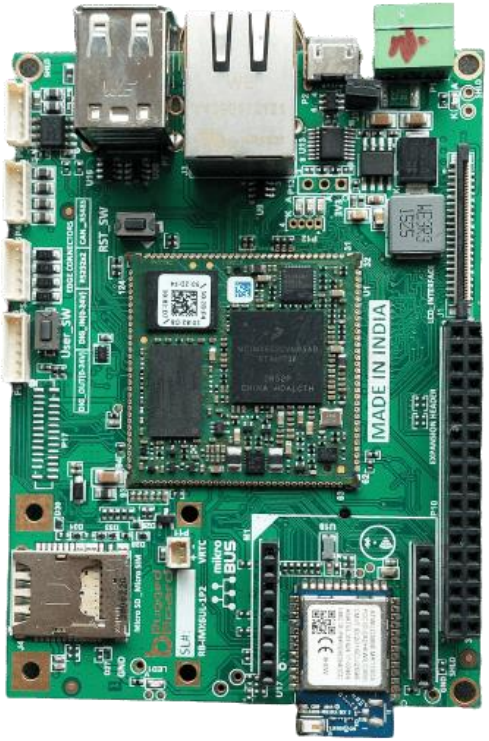
- A5D2x @500MHz
- Cortex - A5
- 64MB Ram
- 32MB Flash
- 2 x RS232
- 1x RS485
- 1 x CAN
- 1 x Ethernet
- TFT & Cap Touch
- 1 x MicroSD Slot
- 2 x USB
- DC & USB Power
- Expansion Header
- mikroBUS Conn.
- mPCIe conn.
- Micro Sim Slot

[Fore More Details Click Here]

RuggedBOARD-i.MX6UL

[Low Power Industrial IoT Gateway]

- Processor: NXP-i.MX 6 UL
- SOM: phyCORE-i.MX 6UL
- Processor Architecture: ARM-Cortex A7
- Clock Frequency: 686 MHz
- OS Build: Yocto Linux
- Kernel-Version: Linux Kernel 4.9
- RAM: 512MB
- ROM: 512MB NAND Flash
- Industrial Interfaces:
  - 1x CAN
  - 1x RS485
  - 2x RS232
  - 8x DIO
  - 1x Ethernet (10/100)
  - 2x USB
  - GPIOs (Number Configurable)
  - Standard mikro-BUS
  - 1x mPCIe
  - 1x SIM and SD card slot
  - Expansion: I2C, SPI, UART, PWM



phyBoard-WEGA

[Industrial HMI ARM Single Board Computer]

Best solution for all your advance fancy User Interface in Control & Automation Systems. Cut down & Simplify your application development time using QT / Android / WinCE

phyBoard-WEGA Features:

- **ARM Cortex - A8@720MHz [TI-AM335x]**

onBoard Devices

- 1 x USB Host / 1 x USB OTG
- 2 x 10/100 Ethernet / Micro SDCARD

Display Interface on Expansion Connectors

- LCD / VGA / HDMI

Communication Interfaces

- UART1(RS232) UART0 console (TTL)
- 1xi2C
- 1xSPI
- 1xCAN Interface

Expansion Connectors

- 1x ADC(12Bit,8Channel)
- GPIOs JTAG

Fore More Details Click Here

phyBoard-Polis

[SBC for Edge Computing AI Applications]

phyBoard-Polis Features:

CPU Cores in i.MX8M Mini

- Quad Cortex™-A53 1.8 GHz,
- Cortex™-M4F 400 MHz
- GC Nano Ultra 3D GPU and GC320 2D GPU
- 1080p VPU

onBoard Devices

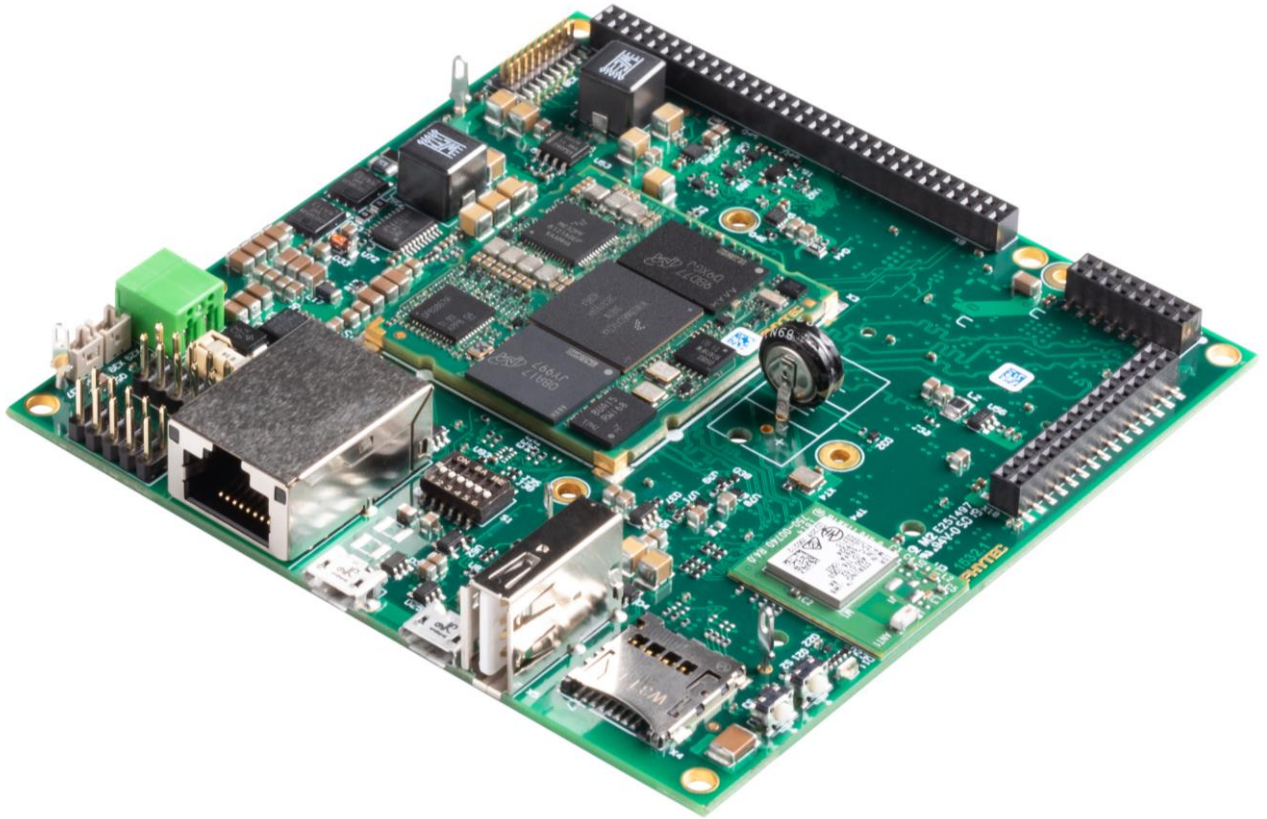
- 1 x USB Host / 1 x USB OTG
- 1 x 10/100 /1000 Ethernet / Micro SDCARD
- WiFi + BLE-4.2 Micro SDCARD

Display Interface on Expansion Connectors

- LCD / HDMI

Communication Interfaces

- UART1(RS232) UART0 console (TTL)
- 1xi2C
- 1xSPI
- 1xCAN Interface
- miniPCIe Connector
- Mipi CSI Camera Connector
- OnBoard TPM



Fore More Details Click Here