The IoT Academy
Connecting The Unconnected

PHYTEC

IASC
SECTOR SKILL COUNCIL

Curriculum

Embedded Full Stack IIOT Analyst

About Us

We are a fast emerging company imparting quality and affordable programs for colleges and individuals for digital economy skills training, internship and guidance focused on helping people develop the skills they need to thrive in the rapidly growing digital economy.

The IoT Academy has made a niche name for itself providing rigorous online/offline training in Data Science, Machine Learning, AI, Python, Java, Internet of Things (IoT), Embedded Systems, Big Data, Data analytics, Industrial IoT, Industry 4.0, Digital marketing, Etc. Based in Delhi NCR since 2017 we have helped 150+ professionals, 600+ students and 100+ faculties across the states get trained, acquire certifications, and upskill their employees.

We have an easy and affordable learning solution that is accessible to millions of learners. With our learners spread across countries like the US, India, UK, Canada, Singapore, Australia, Middle East, Brazil, and many others, we have built a community of over 1 million learners across the globe.

Vision

We aspire to provide learners an edge over the career they choose to stand out amongst today's competitive world through support in training and various academic and industrial collaborations.

Mission

At IoT Academy We believe that the future is about Unified and Converged technologies as reflected in our mission statement. Every aspect of life will eventually lead in bringing forth a unified world of immense possibilities.

About the Program

Embedded Full-Stack Industrial IIoT Analyst By UCT-PHYTEC in collaboration with IASC with execution partner The IoT Academy is responsible for creating and implementing new embedded OS based on system requirements and or industry specifications, building and managing OS drivers for any custom hardware and enhancing existing applications with new features. And also Design and test IoT systems using microcontrollers, sensors, and actuators. Active developer of team Building IoT-based products used in multiple business verticals like Industry-4.0, Energy, Transportation & eMobility, Medical, Smart Cities, Smart Agriculture, Defense & Aerospace. These Experience in programming languages such as C, C++, Python, and Java. Analyze data from sensors and other sources to provide insights and improve system performance. And also Design and implement secure IoT systems that protect data and prevent unauthorized access. And also Design and test IoT systems using microcontrollers, sensors, and actuators. Overall, an Embedded Full Stack IoT Analyst plays a crucial role in designing, developing, and implementing IoT solutions that enable businesses to collect, analyze, and use data to improve their operations, increase efficiency, and reduce costs.

Module 1 : Embedded C Programming

1.1 Basic C BrushUp

- Dynamic Memory
- Datatypes
- Functions
- Arrays
- Pointers
- Storage Classes
- File Handling

1.2 Advance C Programming

- Variable Arguments
- Type Casting

- Enums
- ModTypedefs
- Type Qualifiers
- Other C Libraries
- Function Pointers
- Header Files
- Bit Fields
- Command Line Arguments
- Error Handling

1.3 Hardware Programming

- Accessing Parallel Port, Accessing Serial Port

1.4 Cloud Programming

- Storing data in local database mySQL
- Sending Data to server using HTTP & MQTT
- Local Webserver configuration

1.5 Project work

- Developing Project using learned C programming concept

Module 2 : ARM MCU Programming

2.1 ARM Processor

- SOCs by Semiconductor companies
- ARM Core & ARCH version
- Basic ARM Cortex Arch

2.2 MCU Programming on ARM7 / CortexM0 / CortexM3

- I2C Protocol & Driver implementation
- SPI Protocol & Driver implementation
- GPIO Programming
- UART Programming
- Interrupt Programming
- Timer & Counters Programming
- RTC Programming
- ADC Programming

2.3 Device Interfacing

- Sensor Interfacing (Temp, Humidity, Accelero, Gyro)0
- GSM, GPS, Bluetooth, ZigBee, WiFi
- RFID, Smart Card, Barcode Reader
- Finger PrintSensor
- Keypad, LCD, ADC, DAC

2.4 Project work

- Developing IoT Project using learned C & MCU programming concepts. Home Automation / Smart Wifi Switch / Smart BLE Switch / Industrial Automation Data Logger / Multiprotocol Gateway / BLE Gateway / Modbus Gateway / Modbus Slave Sensor development.

Module 3 : Linux Internals

3.1 Linux Intro & Installation

- How to update Linux and install required packages
- 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]
- Sub systems of Linux [Scheduler, Process, Memory Mgmt, File System, Device Mgmt]

3.2 Linux Shell Commands

- System Commands
- Dir & File Commands
- Basic Commands
- Misc Commands

- Sub systems of Linux [Scheduler, Process, Memory Mgmt, File System, Device Mgmt]

3.2 Linux Shell Commands

- System Commands
- Dir & File Commands
- Basic Commands
- Misc Commands
- Command Line Arguments

3.3 Shell Scripting

- Variables & Operators in Shell scripting
- Writing Basic Linux Shell scripting
- Logical Structures in Shell Scripting

3.4 C Programming in Linux

- Linux Executable format info & tools
- Compiling and executing Linux
- Writing C program on Linux
- Debugging C application on Linux using GDB

3.5 Make Files

- Advanced methods used in writing Make files
- Understanding Make files
- Writing Make files

3.6 Process Management

- Compiling Multiple src Dir using Make file
- How to create child process using [system, exec, fork & clone]
- Understanding Linux Process
- Managing Linux process

3.7 File Operation

- How to write application to access files in Linux
- System Calls used in Linux to control special files like device nodes

3.8 Signals

- Registering & Handling Signals
- Signals in Linux
- How to write a serial port access program in Linux
- Implementing new Signals

3.9 Linux Scheduler & Memory Management

- Understanding Virtual Memory Concept
- System calls for Memory Management
- Linux Kernel Scheduling Policies
- Scheduler System calls
- MMU Subsystem

3.10 Linux MultiThreading Programming

- How to create multithreading applications in Linux
- Managing & communication between Multiple threads
- Basics of Multithreading in Linux

3.11 Inter Process Communication

- Data sharing between Multiple processes using IPC Mech.
- Writing apps using PIPEs, FIFOs, Msg Queues, Shared Memory

3.12 Network Programming in Linux

- How to develop client serverbased network application in Linux
- When and how to use TCP and UDP Protocols

Module 4 : Embedded Linux Porting

4.1 Introduction, Setup & Hardware

- Host PC Setup for eLinux Development
- Boot Process
- Introduction to Embedded Linux
- ARM Processor Basics & Families
- ARM Board Details and Schematic Overview

4.2 Toolchain & Hardware Practical's

- Setting up TFT and Running Application on Board
- How to build toolchain
- Toolchain & its components
- Board Boot Options
- Flashing Bootloader & Linux Kernel on Board

4.3 Bootloader UBoot

- Primary Bootloader (TI XLoader)
- Introduction to Bootloader
- Bootloader Commands and their usage

4.4 UBoot Porting

- How to port Bootloader on ARM Based Hardware
- Compiling Bootloader
- Bootloader Source Code Structure
- Patching Bootloader

4.5 Customizing Bootloader

- Setting up NFS Server
- Linux Kernel Compilation
- Modifying Bootloader for new feature
- Booting with NFS Server
- Modifying Bootloader to support new device
- Command Line Arguments & ATAG

Booting with SD Card

4.6 Linux Kernel

- Kernel Layers H/W dependent and independent (BSP)
- Kernel Build System (KConfig)
- Introduction to Linux Kernel Arch
- Kernel Dir Structure

4.7 Kernel Porting & Compilation

- Type of kernel images (vmlinux, zImage, uImage)
- How to port Kernel on New ARM Hardware
- Kernel initialization process
- How to configure and compile for ARM Hardware

4.8 Kernel Modification

- How to integrate new driver / module in kernel image
- How to modify the Kernel code
- Building static and dynamic kernel modules

4.9 Root File System

- Building RootFS from scratch and using Build System (Buildroot)
- Components of RootFS
- Types of RootFS
- Different types of Flash Device (NOR / NAND)

4.10 Embedded Application Development

- Running sample WebServer Application
- Debugging application on target using GDB
- How to develop embedded applications
- Using Eclipse for embedded application development

Module 5 : Linux Device Drivers

5.1 Introduction and Arch of Linux Device Drivers

- Device Register Access from Code
- Introduction to Kernel Space and User Space
- Memory management in Kernel
- How to develop Kernel Device Driver
- Layers of LDD
- Processor Memory Layout

5.2 Kernel Module Programming

- Module Parameters
- Exporting Symbols between modules
- Kernel Module Programming

5.3 Character Device Drivers

- Implementing advance api like ioct in character device driver
- 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
- Standards to follow while implementing ioctl
- Cross Compiling and running on ARM Hardware
- Writing and testing LED driver with IOCTL on ARM Hardware

5.4 Interrupts in Device Driver

- Interrupts Mechanism in Linux Kernel
- How to implement Interrupts in device driver
- Interrupts in ARM Processor

5.5 Interrupt Handling & Bottom Half

- Writing and testing multiple Intterupts in single driver
  How to implement Shared Interrupts
- How to handle lengthy using Bottom Half (Soft IRQ, Tasklet & Workquees)
- Writing and testing Interrupt for Button press on ARM Target
- Writing and testing Interrupt for Button press on ARM Target

5.6 Special File Systems ProcFS & SysFS

- Using procfs for special purpose and accessing kernel data structure
- How to implement procfs • Sysfs implementation in device drivers for easy application access
- Ram based files systems in Linux
- Ram based files systems in Linu

5.7 LDDM (Linux Device Driver Model)

- Platform Device
- Modify SLED Driver as platform driver
- Platform Data
- Platform Driver

5.8 Board File

- Structure of Board File
- Writing a simple Board File and testing on Board

5.9 Device Tree

- Device Tree examples: memorymapped devices, I2C, SPI, pinmuxing, clocks, etc. •Kernel API's to process device tree data
- Compiling Device Tree and Flashing
- Understanding Device Tree Structure
- Nodes in DTS
- Properties of Nodes
- Kernel API's to process device tree data

5.10 Advance Device Drivers

- Walkthrough MMC domain in AM335x & its implementation
- Lab : Add SDCARD support to Board file and enable root file system to be mounted from SDCard partition.
- Understanding UARTs in AM335x and its driver components
- Lab :Modify Board file to configure UART2 & UART3 on WEGA Board and test it using Linux user application.
- Intput Subsytem in Linux

- Intput Subsytem in Linux
- Lab : Modify Board file to Configure Switches on WEGA board to generate input events & test 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.
- Lab :Modify Board file to add SPI based External ADC device to WEGA Board and test it from user appSPI Subsytem in Linux
- Display SubSystem in Linux
- Lab : Configure the 7” LCD Display and test it using fbtest utils in linux.
- Introduction to block and network device drivers
- Debugging Techniques like debugfs / target debugging
- Case study of Network Device Drive

Module 6 :YOCTO

6.1 Yocto Architecture

- Bitbake usage
- Configuration files
- machine.conf

6.1 Yocto Architecture

- local.conf
- Bitbake
- OpenEmbedded Core
- Poky reference project
- distro.conf

6.2 Recipes defines everything in Yocto

- Recipe tasks
- Writing new recipe
- Understanding Recipes

6.3 Layers makes Yocto Modular & Structured

- Git usage in yocto
- Basic examples recipes(hello.bb)
- Customizing existing recipes (.bb)

- Create bbappend file for existing recipe
- Yocto recipe classes
- Layer in Yocto
- Creating new Layers
- Using existing Layers

6.3 Layers makes Yocto Modular & Structured

- Crete and apply patches
- Customizing recipes

6.4 Adding new Hardware support using BSP Layers

6.4.1 Bootloader

- Setting New defconfing to bootloader
- Adding new bootloader versions to machine
- Adding bootloader to machine
- Create and Apply patches to bootloader
- Providing configuration fragments

6.4.2 Kernel

- Providing configuration fragments
- Adding new kernel versions to machine
- Adding linux kernel to machine
- Create and Apply patches to kernel
- Provide new defconfing to kernel

6.4.3 Building Root File System

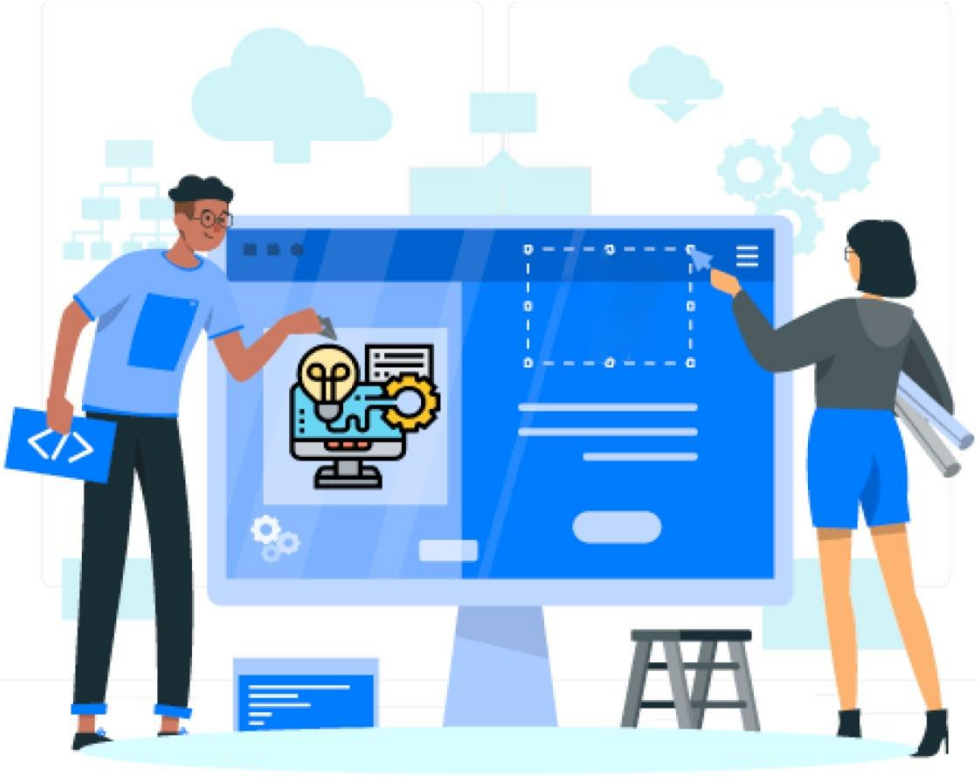
- Integrating IOT Packages to Yocto (MQTT, libcoap...)
- Menu config support for kernel and Bootloader
- Modify existing rootfs image recipe
- Selecting types of root file system images

6.5 Custom Distribution & Images

- Custom Image type for custom Board & Applications
- Package release versioning
- Understanding Distro Layers
- Image types
- Package groups
- Linux package management tools

6.6 Creating SDK using Yocto for Application Development

- Using SDK for Application development
- Generating SDK using Yocto



Program Information:

Embedded Full Stack IIOT Analyst Course

By The IoT Academy, Phytec and IASC

Follow us on:

www.theiotacademy.co info@theiotacademy.co enquiry@theiotacademy.co