e-Niyama Labs

Engineering the rules of learning

Foundation Course in Embedded Systems

FCM1 : Essentials of C Programming

Goal: An introduction to fundamental C programming concepts essential for embedded systems development

Details: This module helps students rapidly learn and apply embedded system concepts through hands-on C programming. By the end of the module, students will be able to analyse problems and develop effective solutions using advanced features of the C language

Duration: 2 Weeks (Theory + Lab sessions)

Platform: Windows and Linux

Tools: Windows - IDEs, Linux – vi/vim editor, gcc, gdb, git

Module Assessment:

- Theory – Objective + QA + Programs
- Lab exam
- Viva – Module basics + Interview questions

Topics:

- Introduction to Linux environment
- Essential shell commands, Vi editor
- Assembler, Compiler, Linker, Hex generator, Loader
- History of C, Structure of a C program
- Variables, Constants Vs Enumeration
- Data Types and Sizes
- Control Statements, I/O statements, Operators
- Functions and Macros
- Storage Classes
- Scope and Lifetime of a variable
- Static, Global and Volatile
- Recursive Functions
- Stack Frame Analysis
- Drawbacks of Functions
- GDB, Basic Debugging Commands
- Algorithm Designing using UML diagrams – Flowcharts, Sequence diagrams, State Charts
- Debugging a sample C Program
- Coding guidelines, Coding style/documentation

e-Niyama Labs
Engineering the rules of learning

FCM2 : Essentials of Embedded Systems

Goal: Master Microcontroller programming essential for embedded systems development

Details: This module helps students rapidly learn and apply embedded system concepts through hands-on Microcontroller programming. By the end of the module, students will be able to analyse problems and develop effective solutions on Microcontrollers.

Duration: 4 Weeks (Theory + Lab sessions)

Platform: Windows and Linux

Tools: Windows – IDE

Module Assessment:

- Theory – Objective + QA + Programs
- Lab exam
- Viva – Module basics + Interview questions
- Mini Project

Topics:

- What are Micro Processors, Micro Controllers?
- Micro controller Vs Microprocessor
- CPU architectures
- ARM 7/9 Architecture
  - Introduction to ARM Architecture
  - Processor operating states
  - Thumb Vs ARM instruction sets
  - Exceptions, Interrupt latencies
  - Linker & Scatter loading
  - ARM Bus architecture
- Overview of Cortex M architecture
  - Introduction to Cortex M Architecture
  - Processor operating states
  - ThumbV2 instruction sets
  - Vectored Interrupt Controllers
  - Peripheral Interfacing
  - Flash Memory, XiP, NVM RAM
  - IO mapped IO vs Memory mapped IO
- Embedded Protocols
  - Serial/UART, I2C, SPI, CAN
  - USB
  - Networking - TCP/IP, Ethernet, MQTT
  - Wireless Protocols - BT and Wi-Fi

- Peripherals Interfacing
  - Interfacing LED and Display
  - Interfacing a switch and Keypad
  - Interfacing ADC & DC Motor
  - Sensors and Actuators

e-Niyama Labs
Engineering the rules of learning

FCM3 : Advanced C Programming

Goal: Advanced C coding concepts to master Embedded Systems Programming

Details: This module enables students acquire advanced C programming skills required for embedded system programming. By end of this module, students gain expertise in writing C programs to resolve complex embedded problems.

Duration: 2 Weeks (Theory + Lab sessions)

Platform: Windows and Linux

Tools: Windows - IDEs, Linux – vi/vim editor, gcc, gdb, git

Module Assessment:

- Theory – Objective + QA + Programs
- Lab exam
- Viva – Module basics + Interview questions
- Mini Project

Topics:

- Pointers and Arrays
- String Handling
- Function Pointers
- Command line arguments
- Static and Dynamic Memory Allocation
- Memory Leaks and Dangling Pointers
- Structure Padding and Alignment
- Union and Bit fields
- Using Typedefs and Enumerations
- Overview of Data Structures
- Embedded C Programming
    - What is Embedded C
    - Assembly Vs C
    - C Vs Embedded C
    - Need of Cross Compiler
    - What and Why Compiler Directives
    - Memory Models & Memory Type Specifiers
    - Writing Recursive and Reentrant functions
    - Intermixing C and Assembly
    - Calling C function from Assembly and vice versa
- Best Practices and Pitfalls

FCM4 : Linux OS

Goal: Master essential Linux concepts for embedded systems programming

Details: This module equips students with essential Linux OS concepts and Linux internals required for embedded systems programming. By the end of the module, students will be able to apply Linux OS principles to effectively solve complex embedded system challenges.

Duration: 2 Weeks (Theory + Lab sessions)

Platform: Linux OS

Tools: Linux – vi/vim editor, gcc, gdb, git

Module Assessment:

- Theory – Objective + QA + Programs
- Lab exam
- Viva – Module basics + Interview questions
- Mini Project

Topics:

- Introduction to Linux, History of Linux
- Linux Kernel Versions, Features of Linux
- Linux Kernel Architecture
  - User & System Mode
  - System Call.
  - Process Management
  - File Management
  - Using proc filesystem
- Inter Process Communication
  - Pipes and FIFOs
  - Message Queues
  - Shared Memory and Semaphore
  - Signals and Sockets
- Memory Management in Linux
- Virtual File System, VFS Architecture
- Introduction to EXT2FS / EXT3FS
- Basic Build Environment
  - Stages of compilation, C Memory map
  - Basics of Linux build system
  - Methods of debugging – gdb dlt, backtrace

FCM5 : Essentials of C++ Programming

Goal: An introduction to fundamental C++ programming concepts essential for embedded systems development

Details: This module helps students rapidly learn and apply embedded system concepts through hands-on C++ programming. By the end of the module, students will be able to analyse problems and develop effective solutions using advanced features of the C++ language

Duration: 2 Weeks (Theory + Lab sessions)

Platform: Windows and Linux

Tools: Windows - IDEs, Linux – vi/vim editor, gcc, gdb, git

Module Assessment:

- Theory – Objective + QA + Programs
- Lab exam
- Viva – Module basics + Interview questions

Topics:

- History of C++
- Features of C++
- C Vs. C++
- OOP Concepts
- Classes and Objects
- Constructors and Destructors
- Friend functions and inline functions
- Operator Overloading
- Inheritance
- Public, protected and Private inheritance
- Types of inheritance
- Function Overriding
- Polymorphism
- Virtual Functions
- Pure Virtual Functions and Abstract classes
- Templates
- Function Templates
- Class Templates
- Exception Handling
- File Handling

e-Niyama Labs
Engineering the rules of learning

FCM6 : RTOS

Goal: Master essential RTOS concepts for embedded systems programming

Details: This module equips students with essential RTOS concepts and Linux internals required for embedded systems programming. By the end of the module, students will be able to apply RTOS principles to effectively solve complex embedded system challenges.

Duration: 1 Week (Theory + Lab sessions)

Platform: Free-RTOS

Tools: Windows – IDE

Module Assessment:

- Theory – Objective + QA + Programs
- Lab exam
- Viva – Module basics + Interview questions
- Mini Project

Topics:

- What are Real Time Systems
- Types and Examples of Real Time Systems
- Monolithic, Micro and Nano Kernels
- Scheduling algorithms
- Overview of popular RTOS
- Introduction to Free RTOS
- Task Management
- Inter Task Communication
    - Semaphores – Mutex, Counting
    - Queues
    - Priority Inversion
    - Priority Inheritance and Priority Ceiling
- Timer and Interrupt Handling
- Memory Management