3.1 Module Names

1. Embedded Systems Concepts, Design and Tools
2. Embedded C Programming (120 Hrs)
3. Data Structures and Algorithms (90 Hrs)
4. Microcontroller Programming and Interfacing (120 Hrs)
5. Embedded Operating Systems (90 Hrs)
6. Embedded Device Drivers (90 Hrs)
7. Real-Time Operating Systems (60 Hrs)
8. Internet-of-Things (90 Hrs)
9. Project and Seminar (150 Hrs)
10. Aptitude and Effective Communication (90 Hrs)

Note

- Each weekday (Monday to Friday) consists of 2 hours of theory lectures/ demos and 4 hours of practicals in the Lab.
- For Embedded C & Data structures the theory hours will be 4hrs per day and 4 hours of practicals in the Lab.

3.2 Module Contents

3.2.1 Embedded Systems Concepts, Design and Tools

Trends in Embedded Systems, Challenges and Design Issues in Embedded Systems, Assemblers, Compilers, Linkers, Loaders, Debuggers, Embedded In-Circuit Emulators & JTAG, Profilers and Test Coverage Tools, Build Tools for Embedded Systems, GNU Cross-Tool chain, Version Control using Git, Embedded Systems architectures, and design: Event driven architectures, Layered architectures, Time-triggered Embedded Systems, State Machines, Publish Subscribe Models, Sequence Diagrams

Note

- The topics covered under this module may be spread across the course
- No separate exam for this module

3.2.2 Embedded C Programming

Basics of Program Writing & Coding Practices, Overview of C Programming language, Introduction to GNU Toolchain and GNU Make utility, Linux environment and vi editor, Tokens of C - Keywords, Data-Types, Variables, Constants, Operators, Identifiers, Storage Class Specifiers, Control Flow Statements, Arrays, Multidimensional arrays, Data Input & Output, Strings, Loops, Functions and Recursion, Pointers - Introduction, Pointer Arithmetic, Pointers and Arrays, Pointers and Functions, Pointers and Strings, Structures, Unions, Enum, Typedef, Bit field operators and pointers with structures, Preprocessors, C and Assembly, Files, I/O, Variable number of arguments, Command Line arguments, Error handling, Debugging and Optimization of C programs, Bit operations, Handling portability issues in C, Hardware, Time, Space and Power aware Programming

3.2.3 Data Structures and Algorithms

Introduction to Data Structures, Algorithms and Abstract Data Types, Complexity of Algorithms, Linked Lists, Stacks, Queues, Searching and Sorting Algorithms, Hashing, Trees

3.2.4 Microcontroller Programming and Interfacing

Overview of Microcontrollers, Microprocessors and SoC, RISC vs CISC, Harvard vs Princeton Architectures, Overview of Computer Architecture, Embedded Memories, Timers/Counters, UART, SPI, PWM, WDT, Input Capture, Output Compare Modes, I2C, CAN, LED, Switches, ADC, DAC, LCD, RTC, Bus Standards (USB, PCI), Programming in Assembly and Embedded C,

ARM: Overview of ARM Architecture and Organization, Introduction to Cortex-M Architecture, Programming Model and Instruction Set Architecture, Alignment and Endianness, Register access, State, Privilege, Stack, System Control Block, Power Modes, Memory Model, NVIC, Exception Handling, Bit- Banding, Peripheral Programming, SVCall, SysTick, PendSv, MPU, DMA, Mixing Assembly and C programs, Introduction to CMSIS & CMSIS Components, Overview of Cortex A & R architectures

RISC V: Why RISC-V processor, RISC-V processor overview, ARM vs RISC-V, Modes in RISC-V, Setting up of necessary tools, RISC-V register set and calling convention, Instruction formats and type, Build Process, Practical examples of instructions, Detail description on Control and Status Registers, Exception handling, Examples in assembly for exception handling, Interrupts, Interrupt Entry and Exit procedure, Introduction to C-DAC VEGA processors

3.2.5 Embedded Operating Systems

Introduction to Embedded Operating Systems, Anatomy of an Embedded Linux System - Bootloader, Kernel, Root File System, Application -, Process Management, Interprocess Communication & Synchronization, Memory Management, I/O sub- system & Embedded File Systems, POSIX Thread Programming, POSIX Semaphores, Mutexes, Conditional Variables, Barriers, Message Queues, Shared Memory, Debugging and Testing of Multithreaded Applications, Socket Programming, Customizing Embedded Linux based on Yocto, Virtualization: Dockers & Containers

3.2.6 Embedded Device Drivers

The Embedded Linux Software Eco-System, Linux Kernel Modules and Module Programming, Char Device Drivers, Kernel Internals: Dynamic memory allocations, Handling Delays, Timers, Synchronization, Locking, I/O Memory and Ports, Interrupts, Deferred Executions, Driver Debugging Techniques, Drivers for GPIO, I2C, and SPI, Pseudo Filesystems (procfs, sysfs)

3.2.7 Real-time Operating Systems

Introduction to Real-Time Concepts, RTOS Internals & Real Time Scheduling, Performance Metrics of RTOS, Task Specifications, Schedulability Analysis, Application Programming on RTOS, Porting of RTOS, Configuring RTOS, Building RTOS Image for Target platforms

3.2.8 Internet of Things (IoT) & Embedded AI

IoT: IoT Trends, IoT Architecture, IoT Applications, IoT Standards and Protocols, Wireless LAN: IEEE 802.11, Wireless PAN: IEEE 802.15.1 & 802.15.4, Zigbee, Bluetooth, BTLE, LPWAN (LoRa, NB- IoT), 6LowPAN, REST, CoAP, MQTT, Basics of Cryptography, Overview of IoT and Embedded security, Overview of 5G technologies

Embedded AI: AI Fundamentals, Supervised Learning, Unsupervised Learning, Ensemble Techniques, Time Series Forecasting, Neural Networks and Deep Learning, Embedded AI applications, Embedded AI frameworks (CMSIS-NN, AIfES, TensorFlow-Lite, TensorFlow-Lite Micro etc), Feature Engineering, Model Selection & Tuning, Development and deployment of embedded ML models, Case Study on Embedded AI

3.2.9 Project and Seminar

Students are required to execute project work for the duration of four weeks (after the completion of all modules) as a part of this course. For seminar, students need to choose the topic themselves and give the seminar on the respective dates allocated by the concerned faculty members. The topic chosen by the students should be relevant to the Embedded Systems Design. Project work is distributed in the following phases:

1. Study and Requirements Elicitation
2. Design
3. Implementation
4. Testing
5. Project report
6. Viva Voce and Presentation

Students need to submit a project report at the conclusion of the project. Mentors should be allocated within 3 weeks of the course commencement and should be executed throughout the course duration. The students should maintain a logbook, which contains their day-to-day ac-

4.2 Summary of the Module Coverage

4.2.1 Embedded C Programming

1. Basics of Program Writing & Coding Practices
2. Overview of C Programming language
3. Introduction to GNU Toolchain and GNU Make utility
4. Linux environment and vi editor
5. Tokens of C - Keywords, Data-Types, Variables, Constants, Operators, Identifiers
6. Storage Class Specifiers
7. Control Flow Statements
8. Arrays, Multidimensional arrays, Data Input & Output, Strings, Loops - for, while etc.
9. Functions and Recursion
10. Pointers - Introduction, Pointer Arithmetic, Pointers and Arrays, Pointers and Functions, Pointers and Strings
11. Structures, Unions, Enum, Typedef, Bit field operators and pointers with structures
12. Preprocessors, C and Assembly, Files, I/O,
13. Variable number of arguments, Command Line arguments, Error handling
14. Debugging and Optimization of C programs
15. Bit operations & Handling portability issues in C
16. Hardware, Time, Space and Power aware Programming
17. Secure Programming in Embedded C – tools and techniques

4.2.2 Text Books

- Embedded C Programming

1. Programming Embedded Systems in C and C++" by Michael Barr and Anthony Mass
2. The C Programming Language. 2nd Edition, Book by Brian Kernighan and Dennis Ritchie

Table 5.1: Duration of Data Structures and Algorithms module

TYPE	DURATION (HRS)
Lectures and Demonstrations	30
Labs	60
Total Duration	**90**

Table 5.2: Evaluation details of Data Structures and Algorithms module

MODULE/ SUB-MODULE	MAX MARKS
Data Structures and Algorithms	100
Module Total	**100**

5.1 Summary of the Module Coverage

5.1.1 Data Structures and Algorithms

1. Introduction to Data Structures
2. Algorithms and Abstract Data Types
3. Complexity of Algorithms
4. Linked Lists, Stacks, Queues
5. Searching and Sorting Algorithms
6. Hashing, Trees

6 | Microcontroller Programming and Interfacing

6.1 Duration and Evaluation Details

Note

The total duration of the module is 120 Hrs with 2 hours of lectures and 4 hours of practical labs per day

Table 6.1: Duration of Microcontroller Programming and Interfacing

Type	Duration (Hrs)
Lectures and Demonstrations	40
Labs	80
Total Duration	**120**

Table 6.2: Evaluation details of Microcontroller Programming and Interfacing

Module/ Sub-Module	Max Marks
Microcontroller Programming and Interfacing	100
Module Total	**100**

6.2 Summary of the Module Coverage

1. Overview of Microcontrollers
2. Microprocessors and SoC, RISC vs CISC, Harvard vs Princeton Architectures
3. Overview of Computer Architecture
4. Embedded Memories
5. Timers/Counters, Input Capture, Output Compare Modes
6. LED, Switches, ADC, DAC, LCD, RTC
7. UART, SPI, PWM, WDT, I2C, CAN

8. Bus Standards (USB, PCI)
9. Programming in Assembly and Embedded C
10. ARM
    - Overview of ARM Architecture and Organization
    - Introduction to Cortex-M Architecture
    - Programming Model and Instruction Set Architecture
    - Alignment and Endianness, Register access,
    - States and Privileges
    - Stack, System Control Block, Power Modes
    - Memory Model
    - NVIC, Exception Handling
    - Bit-Banding, Peripheral Programming
    - SVCall, SysTick, PendSv,
    - MPU, DMA
    - Mixing Assembly and C programs
    - Introduction to CMSIS & CMSIS Components
    - Overview of Cortex A & R architectures
11. RISC V
    - Why RISC-V processor, RISC-V processor overview
    - ARM vs RISC-V
    - Modes in RISC-V
    - Setting up of necessary tools
    - RISC-V register set and calling convention
    - Instruction formats and type
    - Build Process
    - Practical examples of instructions
    - Detail description on Control and Status Registers
    - Exception handling
    - Examples in assembly for exception handling.
    - Interrupts
    - Interrupt Entry and Exit procedure
    - Introduction to C-DAC VEGA processors

6.3 Online/MOOCS Courses

- Embedded Hardware Design: HANDS-ON Course on CHARiOT Platform
  https://meghsikshak.in/chariot/jsp/chariot/courses/Hands-on.jsp

6.4 Text Books

- The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, Third Edition, Joseph Yiu
- Computer Organization and Design RISC-V Edition: The Hardware Software Interface, David A. Patterson, John L. Hennessy

7.1 Duration and Evaluation Details

Note

The total duration of the module is 90 Hrs with 2 hours of lectures and 4 hours of practical labs per day

Table 7.1: Duration of Embedded Operating Systems

TYPE	DURATION (HRS)
Lectures and Demonstrations	30
Labs	60
Total Duration	**90**

Table 7.2: Evaluation details of Embedded Operating Systems

MODULE/ SUB-MODULE	MAX MARKS
Embedded Operating Systems (EOS)	100
Module Total	**100**

7.2 Summary of the Module Coverage

1. Introduction to Embedded Operating Systems
2. Anatomy of an Embedded Linux System: Bootloader, Kernel, Root File System, Application
3. Process Management
4. Interprocess Communication & Synchronization
5. Memory Management
6. I/O sub- system & Embedded File Systems
7. POSIX Thread Programming, POSIX Semaphores, Mutexes, Signals

8. Conditional Variables, Barriers, Message Queues, Shared Memory
9. Debugging and Testing of Multithreaded Applications
10. Socket Programming
11. Virtualization: Dockers and Containers

7.3 Text Books

- Mastering Embedded Linux Programming, Second Edition, Chris Simmonds, Packt Publishing

or

- Building Embedded Linux Systems, Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, Philipp Gerum, OReilly Publications

7.4 Reference Books and Papers

1. **https://bootlin.com/doc/training/embedded-linux/**

7.5 Detailed Syllabus

- Concept of baremetal and multi-tasking kernels, OS Architectures: Monolithic vs Micro-Kernel
- Anatomy of an Embedded Linux System: Bootloader, Kernel, Root file system, Application
- Overview of Bootloader, Single Stage or Multi-Stage bootloaders, Functional Blocks of bootloaders (Main Function Module, I/O Channel Driver Module, Memory Device Driver Module), Overview of Board Support Packages, Overview of U-Boot, Yocto based customization of Embedded Lunux
- Embedded Linux boot process: First and second stage bootloaders, Kernel Decompression, Architecture-Specific Initialization, Board-Specific Initialization, Kernel Entry Point, Mount Initial RAM File System & Root Filesystem, init/ systemd
- Static and dynamic libraries, User and Kernel Mode, Libary vs System calls
- Process / Task Management, Synchronization, Interprocess Communication
- Scheduling Algorithms
- Memory management, Virtual Memory, Paging
- I/O sub-system & Embedded File Systems
- POSIX Semaphores, Mutexes, Conditional Variables, Barriers, Read-Write Locks
- Pipes, FIFOs, Shared Memory, Message Queues
- TCP/ UDP Socket programming based on Embedded Linux
- Debugging and Testing of Multithreaded Applications
- Concept of Device Trees and Building Root Filesystem
- Concept of Hardware Abstraction Layers (HAL)
- Application development & Debugging on Embedded Linux
- Build process of Embedded application project
- Executable and Linking Format (ELF), Relocatable file, Executable file, Shared object file
- Concept of Instruction Set Architecture (ISA), Application Binary Interface (ABI) and Application Programming Interface (API)

8.1 Duration and Evaluation Details

Note

The total duration of the module is 90 Hrs with 2 hours of lectures and 4 hours of lab per day

Table 8.1: Duration of Embedded Device Drivers

TYPE	DURATION (HRS)
Lectures and Demonstrations	30
Labs	60
Total Duration	**90**

Table 8.2: Evaluation details of Embedded Device Drivers

MODULE/ SUB-MODULE	MAX MARKS
Embedded Device Drivers	100
Module Total	**100**

8.2 Summary of the Module Coverage

1. The Embedded Linux Software Eco-System
2. Linux Kernel Modules and Module Programming
3. Char Device Drivers
4. Kernel Internals: Dynamic memory allocations
5. Handling Delays, Timers
6. Synchronization, Locking, I/O Memory and Ports
7. Interrupts, Deferred Executions, Driver Debugging Techniques

8. USB device driver, Drivers for GPIO, I2C, and SPI
9. Pseudo Filesystems (procfs and sysfs)
10. GPIO drivers

8.3 Text Books

- Linux Device Drivers Development, John Madieu, Packt Publishing

or

9 | Real-Time Operating Systems (RTOS)

9.1 Duration and Evaluation Details

Note

The total duration of the module is 60Hrs with 2 hours of lectures and 4 hours of lab per day

Table 9.1: Duration of Real-Time Operating Systems (RTOS)

Type	Duration (Hrs)
Lectures and Demonstrations	20
Labs	40
Total Duration	**60**

Table 9.2: Evaluation details of Real-Time Operating Systems (RTOS)

Module/ Sub-Module	Max Marks
Real-Time Operating Systems (RTOS)	100
Module Total	**100**

9.2 Summary of the Module Coverage

1. Introduction to Real-Time Concepts
2. RTOS Internals & Real Time Scheduling
3. Performance Metrics of RTOS
4. Task Specifications
5. Schedulability Analysis
6. Application Programming on RTOS
7. Porting of RTOS

8. Configuring RTOS and Building RTOS Image for Target platforms

9.3 Text Books

- Real-Time concepts for Embedded Systems, Qing Li, Caroline Yao, CMP Media LLC

or

- Real-Time Systems Design and Analysis: Tools for the Practitioner, 4th Edition, Philipp A. Laplante, Seppo J. Ovaska, Wiley Publisher

9.4 Reference Books and Papers

1. Real-Time Embedded Systems: Design Principles and Engineering Practices, Xiaocong Fan
2. The FreeRTOS Reference Manual, FreeRTOS online
3. Use of FreeRTOS in Teaching Real-time Embedded Systems Design Course, Dr. Nannan He, Minnesota State University, Mankato
4. RTOS Program Models Used in Embedded Systems, József Kopják, Dr. János Kovács

9.5 RTOS Detailed Syllabus

- Introduction to Real-Time Operating Systems
- Factors affecting the choice of Bare metal vs multi-tasking for a specific application
- RTOS vs Embedded OS vs General purpose OS
- Hard vs Soft Real-Time Systems
- Examples of Hard and Soft Real-Time Systems with timing constraints and consequence of missing deadlines
- Real-time embedded systems development process
- RTOS Internals, Priority based Premptive Scheduling, Real Time Scheduling

- RTOS booting sequence: Hardware initialization, RTOS initialization, and Application initialization
- Task and Interrupt Level Scheduling
- Interrupt Latency, Response and Recovery
- Static and Dynamic Priority driven Scheduling
- Rate Monotonic, Deadline Monotonic, Round Robin & FIFO Scheduling in RTOSs, Sporadic Server Scheduling Policy, Minimum Laxity First, Earliest Deadline First
- Real-Time Task classification: Periodic tasks, Sporadic tasks, Aperiodic tasks
- Real-Time Task design: eg. How many tasks should be considered? How should one assign priorities to those tasks? How should one document the task design and the timing constraints, Idle tasks
- Task Specifications: Periodicity, Deadlines, Worst Case Execution Time, Release Time
- Schedulability analysis: eg. with the identified timing constraints, is the set of tasks schedulable?
- Application Programming on RTOS, Task Management, Real-Time Synchronization and Interprocess Communication, Timer APIs, Data sharing with ISR, Software Timers
- Priority Inversion, Priority Inheritance, Priority Ceiling
- Patterns for Time-Triggered systems
- RTOS aware device drivers
- Priority Inversion, Priority Inheritance, Priority Ceiling
- Patterns for Time-Triggered systems
- RTOS aware device drivers and RTOS systems design based on MPU

10 | Internet of Things (IoT) & Embedded AI

10.1 Duration and Evaluation Details

Note

The total duration of the module is 90 Hrs with 2 hours of lectures and 4 hours of lab per day

Table 10.1: Duration of Internet of Things (IoT)

Type	Duration (Hrs)
Lectures and Demonstrations	30
Labs	60
Total Duration	**90**

Table 10.2: Evaluation details of Internet of Things (IoT)

Module/ Sub-Module	Max Marks
Internet of Things (IoT)	100
Module Total	**100**

10.2 Summary of the Module Coverage

IoT

1. IoT Trends, IoT Architecture, IoT Applications
2. IoT Standards and Protocols
3. Wireless LAN: IEEE 802.11
4. Wireless PAN: IEEE 802.15.1 & 802.15.4, Zigbee,
5. Bluetooth, BTLE, LPWAN (LoRa, NBIoT), 6LowPAN
6. REST, CoAP, MQTT
7. Basics of Cryptography, Overview of IoT and Embedded security

8. Overview of 5G technologies

Embedded AI

9. AI Fundamentals
10. Supervised Learning, Unsupervised Learning, Ensemble Techniques, Time Series Forecasting, Neural Networks and Deep Learning
11. Embedded AI applications
12. Embedded AI frameworks (CMSIS-NN, AIfES, TensorFlow-Lite, TensorFlow-Lite Micro etc)
13. Feature Engineering, Model Selection & Tuning
14. Development and Deployment of embedded ML models
15. Case Study on Embedded AI

10.3 Text Books

- Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security, Perry Lea