**Module 1: System Development Lifecycle (V-Model)**

* SDLC - Development Lifecycles and Frameworks [5 hours]
* SDLC Overview and its Phases [1 hour]
* SDLC Models
* Estimation & Planning
* Agile - an iterative and responsive software development methodology [3 hours]
* Agile Methodology
* Scrum and Kanban
* Test Driven Development (TDD), Behavior Driven Development (BDD), Model Driven Development
* Development Bible [8 hours]
* Architecting and Designing the System
* Reviews
* Languages and Static Code Analysis (SCA) (Program Testing)
* Code Repository, Build and Deployment Process (GIT and GIT HUB)
* Measurement and Metrics
* Roles and Responsibilities
* Development and Operations [3 hours]
* DevOps Process Flow
* Embedded Testing [1 hour]
* Overview
* Project - Digital Door Locking System or Develop a Mobile Phone

**Module 2: Hardware & Laboratory Familiarization**

* Introduction to Embedded Systems [1 hr]
* What are embedded system?
* Basic Embedded System Architecture
* Standard Interfaces
* Understanding schematics/datasheet
* Electrostatic Discharge Essentials [X hrs]
* What is ESD and why we deal with ESD.
* Causes of ESD, ESD can occur at all levels of integration
* How to prevent ESD, outfitting an effective workspace
* Fundamentals of Booting for Embedded Processors [1.5 hr]
* Host and Target Development Setups
* System boot components and understanding workspace
* Securing Embedded Systems [30 min]
* What is Hardware Security in Embedded Systems?
* Techniques to improve embedded system security

**Module 3: System Programming Using C**

* Building an executable [3 Hr]
* Linkers and Memory layouts
* Map files
* Building libraries
* Debugging & Tracing [3 Hr]
* Debugging with gdb
* Trace
* Disassembly of C code
* Memory Management [4 Hr]
* Memory Architecture
* Memory allocation
* Memory Profiling
* Advanced Data Types [3 Hrs]
* Variable length data structures
* Structures and Unions
* Referencing data and functions [3 Hrs]
* Various pointers
* Traversing lists with pointers
* Dynamic binding
* Working with registers [3 Hrs]
* Bit level operations
* Handling special registers
* Context management [4 Hrs]
* Bringing up CPU
* Requirements of C startup
* What is execution context?
* Interrupt Context
* C library functions [4 Hrs]
* Useful primitives
* System Calls
* OOPS Concept
* Interprocess communications
* Refering to implementations
* Coding practices [3 Hrs]
* Coding guidelines
* Secure and Safe Coding
* Develop optimal code

**Module 4: Embedded OS Fundamentals**

* Pre-requisite [1 Hr]
* Sub-topic 1

1. Overview of freeRTOS
2. Download and install freeRTOS
3. Compilation and build tool setup
4. Pre-requisit - student should have gone through the ""Advanced C Programming"" course

* Simulator / Emulator Requirements
* Hardware setup and demo
* Sub-topic 2

1. freeRTOS source code organization
2. freeRTOS ports
3. Demo Applications
4. Configuring freeRTOS
5. Building a simple demo application

* Simulator / Emulator Requirements
* Hardware setup and demo
* Concept of Real-time OS [30 min]
* Sub-topic 1

1. Hard real time vs soft real time
2. Multi-threading/ Multi-tasking / Concurrent execution
3. Schedular introduction
4. Context switching introduction
5. Features of freeRTOS
6. Value proposition of freeRTOS
7. Understanding of bootup sequence and flashing/downloading executable image on HW

* Memory Management [1 Hr]
* Sub-topic 1

1. Pre-requisite - ""Advance C covering memory management""
2. Understanding of different memory available on MCU (include Endianness)
3. Introduction to Heap vs Stack memory, program memory vs data memory

* Sub-topic 2

1. freeRTOS Heap Memory Management (dynamic vs static, predictability, fragmentation, different memory allocation schemes)
2. freeRTOS Heap Utility Functions
3. Optimizing memory
4. Debugging memory
5. Memory Map
6. Concepts - architecture flat memory model, no distinction between kernel and user memory in RTOS, memory protection unit etc.

* Task Management [1 Hr]
* Sub-topic 1

1. concept of freeRTOS Tasks
2. freeRTOS Tasks APIs
3. Creating Tasks, Task Priorities
4. Task State Transitions

* Sub-topic 2

1. Schedular
2. Idle Task
3. Scheduling algorithms
4. Time measurement and Tick Interrupt

* Inter Task (Process) Communication (synchronization) [1.5 Hr]
* Sub-topic 1
* Queue Management

1. Introduction to Queue (1 hr theory 4 hrs lab)
2. freeRTOS Queue APIs
3. Data storage for Queue (data of fixed length, variable length, mem copy vs by reference etc.)
4. Blocking read, write
5. Receiving data from multiple queues
6. Mailbox (using queue)

* Sub-topic 2
* Interrupt Management

1. Events
2. ISRs
3. Tasks vs ISRs
4. Nested Interrupts

* Sub-topic 3
* Semaphores: Binary/Counting Semaphores
* Resource Management [1.5 Hr]
* Sub-topic 1

1. Problem - shared resource curruption (Multi-tasking system)
2. Read, modify, write operations
3. Atomic operations
4. Re-entrant functions or thread safe functions
5. Mutual Exclusion
6. Critical Section

* Sub-topic 2

1. Mutex (or Lock)
2. Deadlocks, starvation
3. Priority inversion
4. Priority Inheritance
5. Gatekeeper task (for guarding shared resources)

* Sub-topic 3
* Event Groups for Multiple Task Synchronization
* Timer Management [30 min]
* Sub-topic 1

1. Software Timers:

Context of Software Timer

Creation of Software Timer

Resetting Software Timer

1. Hardware timers and Watchdogs

* Task Notifications [30 min]
* Sub-topic 1

1. Tasks communication objects vs direct task notifications
2. Benefits of task notifications

* Advance RTOS concepts [1 Hr]
* Sub-topic 1

1. SMP port of freeRTOS
2. Memory Protection Unit support
3. Low Power Support (Tickess execution feature)
4. Trace and Runtime stats
5. Refresh on device drivers and integration
6. Comparison of different popular RTOSes against evaluation metrics

* RTOS File System (freeRTOS+FAT) [1 Hr]
* Sub-topic 1

1. File System Concept
2. FreeRTOS + FAT (FAT12, FAT16, FAT32 DOS/Windows compatible File System)
3. File System APIs, source code organization, configuration
4. FAT examples: basic file use, stdio API tests, creating a disk etc.
5. Example: FTP server, HTTP Web Server
6. Creating a media driver
7. Introduction of File Systems on Flash, HDD, SSD, eMMC, SD cards

* Project [30 min]
* Sub-topic 1

1. System design example from the perspective of hardware, tasks, interrupts/events, timers, real-time processing (discussion on one concrete open source project example)
2. Common problems encountered (crash, ISR issues, stack overflow issues), debugging (asserts, traces/logs), and profiling etc.

* Sub-topic 2

Deliver a working project along with documentation based on freeRTOS on either Windows emulator or embedded hardware board. These projects will be an existing freeRTOS based open source project to be enhanced further OR a new embedded application to be demonstrated on embedded hardware board.

**Module 5: Driver, Applications and Middleware**

* Introduction to ARM Cortex-M and its Architecture [2 Hr]
* ARM Cortex-M Architecture and pin diagram
* ARM Cortex-M Memory organization
* Introduction to Target board and GPIO programming [1 Hr]
* Interrupts, Timers and counters [1 Hr]
* Optimizations [1 Hr]

**Module6: Verification & Validation**

* Fundamentals of testing
* What is testing?
* Why is testing necessary?
* Seven testing Principles
* Test Process
* The Psychology of Testing
* Testing Throughout the Software Development Lifecycle
* Software Development Lifecycle models
* Test Levels
* Test Types
* Static Techniques
* Static Testing Basics
* Review process
* Test techniques
* Categories of Test Techniques
* Black-box Test Techniques
* White-box Test Techniques
* Experience based Test Techniques
* Test design techniques for embedded systems
* Test Infrastructure
* Embedded software test environments

1. Simulation
2. Prototyping
3. Pre-production
4. Post production

* Tools
* Categorization of test tools
* Test Automation
* Test Monitoring & Control

**Module 7: Packaging & Release along with DevOps**

* Overview of DevOps [1 Hr]
* Learn DevOps Fundamentals
* Introduction to DevOps
* Basic of Virtualization
* Version Control with Git [1.5 Hr]
* Version Control System
* Git and GitHub
* Packaging, Release and Continuous Integration [1 Hr]
* Learn Continuous Integration, Packaging & Release and Hands-On
* Building and Packaging
* Introduction to Jenkins
* Continuous Integration and Delivery with Jenkins

**Module 8: Embedded System: Domain Specific**

* Pre-requisites [1 Hr]
* Machine Learning Basics
* Machine Learning & Embedded Systems [30 min]
* Sub-topic 1

1. Machine Learning on Microcontrollers
2. Dataset and Anomaly Detection
3. Feature Selection and Extraction
4. Theory: How Deploy a Trained Model to RPI/Arduino

* Neural Networks and ML Model [30 min]
* Sub-topic 1

1. Neural Networks and Training Methods
2. Model Evaluation Techniques
3. Overfitting and Underfitting
4. Deploy Model to Embedded System [RPI Board]

* Embedded machine learning: Practice exercises [1 Hr]
* Sub-topic 1

1. Feature Extraction for Audio Data
2. Convolutional Neural Networks Introduction
3. Audio Classification and Sampling Audio Signals
4. MFCCs and CNNs
5. Demo Application on Audio class inferencing

* Automotive Embedded Systems: Introduction [1 Hr]
* Sub-topic 1

1. Basic Introduction
2. Domains in Automotive Systems

* Sub-topic 2

1. Model based Development in MATLAB
2. Software component modelling
3. Simulation & Code Generation

* Autosar Architecture: Introduction [1 Hr]
* Sub-topic 1

1. Architecture overview
2. MCAL Layer
3. Services Layer
4. Autosar Diagnostics & Memstack
5. Autosar RTE

* AUTOSAR Model Deployment in MATLAB [30 min]
* Sub-topic 1

1. Modeling Autosar SWCs in MATLAB
2. Embedded Coder vs Autosar Coder
3. Autosar Editor – Code Mapping
4. Functional Safety

* Project on Embedded Machine Learning or Automotive Embedded Systems
* Sub-topic 1

1. Embedded Machine Learning: Real-time classification of Audio streams
2. Automotive Embedded Systems: Modelling Autosar SWCs in MATLAB