Embedded Systems and Automotive Protocols

Module 1: C Programming – 2 Hours / Day

C programming is a foundational programming language that serves as the building block for many modern languages. Learning C gives you deep insights into how computers work at a low level, and understanding the following concepts will help you become proficient in programming, debugging, and problem-solving.

DAYS	PARTICULARS
Day 1	C Introduction - C Code execution flow, Variables, Data types & Operators.
Day 2	Decision making- if, if-else, nested if, if-else-if, switch case
Day 3	Looping - for, while, do-while
Day 4	Arrays - 1D,2D
Day 5	Strings
Day 6	Functions in C
Day 7	Storage class, Bitwise Operators
Day 8	Pointers
Day 9	Structures, Union
Day 10	Enumeration, Typedef & Macros – Preprocessors
Day 11	Typedef & Macros – Preprocessors
Day 12-16	Linked List(Node insertion, Deletion, Stack,Queue,PUSH,POP,Sorting algorithms)
Day 17-20	C Revision
Day 21	C Language Assessment
Day 22	C Language Assessment

Note:

- ❖ Practical Hands-on Assignment will be given every day corresponding to the topics taken
- ❖ Candidates will be provided two Attempts of Evaluation once completed C Training
- ❖ If the candidates secured above the cut-off marks in first attempt, they can move directly to second module.

Module 2: EMBEDDED (PIC) – 2 Hours / Day

Learning PIC (Peripheral Interface Controller) programming involves understanding the architecture and functionality of PIC microcontrollers, as well as gaining practical knowledge of embedded systems. Here's an overview of what you'll typically encounter when learning PIC programming:

What is a PIC?
PIC is a family of microcontrollers made by Microchip Technology. PIC microcontrollers are widely used in embedded systems due to their ease of use, low cost, and versatility.

PIC Architecture:
Introduction to the internal structure of a PIC microcontroller, including:

- The CPU (Central Processing Unit)
- Memory organization (Program Memory, Data Memory, EEPROM)
- Registers (General Purpose and Special Function Registers)
- Pin configurations and I/O ports

PIC Variants:
Overview of different PIC families (PIC10, PIC12, PIC16, PIC18, PIC24, PIC32) and their use cases.

DAYS	PARTICULARS
Day 1-5	Basics of Electronics: Active and Passive Components, Switches, Diodes, Digital Electronics, Servo Motors

Day 6	Analog Electronics, Embedded Systems Introduction, Sensors related applications.
Day 7	Importance and Evolution of Embedded Systems, Different Architecture and Elements used in Embedded Systems
Day 8	Embedded C Programming – Operators, Loop, strings, Array, Pointers, Function and Macros, Compilers, Editor and Execute
Day 9	Simulation on software – Basic LED Interfacing, Switches, Seven Segment Interface, Traffic Lights
Day 10	LCD Interface, Header file creation, Interrupt, Timers
Day 11	Graphical LCD, CCP, ADC
Day 12	CCP(Capture,Compare,PWM)
Day 13	UART, SPI
Day 14	I2C
Day 15	EEPROM, RTC interfacing
Day 16 -21	Project Prototype: Sensor (DHT11, LM35, Heart), Motor (DC Motor, Gear Motor, Stepper Motor, Servo Motor), Bluetooth Interfacing, Keypad Interfacing, GSM and GPRS
Day 22	Real Time Application Projects.

Note:

- Practical Hands-on Assignment will be given every day corresponding to the topics taken. Proteus, a widely used simulation software, can be effectively employed to simulate various embedded systems projects.
- Project will be allocated for every candidate which will be on any topic on Real Time Application Projects after the completion of this module.
- If the candidates completed this real time Project with the mentioned time period. He/She will be eligible to go the next module.

Module 3: CONTROLLER AREA NETWORK(CAN) - Windows

DAYS	PARTICULARS
Day 1	Introduction to CAN - Need for Network Arbitration, Introduction to CAN, CAN Data Frame Format, Setup Virtual CAN Bus on Linux, Receiving and Monitoring CAN Bus
Day 2	CANOE Tools - Introduction & Overview, Software simulation Setup and overview
Day 3	CAN Electronic Control Unit (ECU) and IG Block, Trace Window etc.,

Day 4	Introduction to CAPL Programming for simulating a CAN Network
Day 5	Introduction to test specification Environment in CANOE
Day 6	Using a CAPL for Testing a CAN Network
Day 7	CAN Application Development
Day 8	UDS protocol (Frame format)
Day 9	Service ID, Sub function, Request frame format, Response frame format
Day 10	SDLC and STLC – Introduction, Overview, Models involved in SDLC and Application
Day 11	Introduction of Autosar
Day 12	Application layer and Autosar Runtime Environment, BSW Components

CANoe (from Vector) is a powerful development, testing, and simulation tool used in the automotive and embedded systems domain for working with **CAN (Controller Area Network)** protocol, among others (LIN, FlexRay, Ethernet, etc.). Learning CAN protocol using CANoe involves a mix of theory, hands-on simulation, and real-time testing.

Here's a structured description of learning **CAN protocol** with **CANoe**:

By the end of this course, you will be able to:

- Understand and interpret CAN protocol communication.
- Simulate CAN bus networks using CANoe.
- Configure and use DBC files for decoding CAN messages.
- Write custom simulation scripts using CAPL.
- Perform real-time diagnostics and analysis of CAN traffic.
- Test physical ECUs using hardware interfaces and CANoe.

Module 4: EMBEDDED (STM 32 Using HAL programming) - 2 Hours / Day

STM32 HAL (Hardware Abstraction Layer) programming using **STM32CubeIDE** allows developers to interact with the STM32 microcontrollers in an easier and more structured way. STM32CubeIDE, a development tool from STMicroelectronics, integrates code generation, debugging, and programming features in one environment, making it ideal for learning HAL programming.

Here's a structured description of learning **STM32 HAL programming** using **STM32CubeIDE**.

 Overview of STM32 Microcontrollers:

- Introduction to the STM32 family of microcontrollers, their architecture, and peripheral features.
- Application areas of STM32, including automotive, industrial, and IoT.

 STM32CubeIDE:

- STM32CubeIDE is an integrated development environment (IDE) that combines the functionality of **STM32CubeMX** (code generator) with advanced debugging and programming features.
- Familiarization with the IDE layout: Project Explorer, Editor, Peripheral Configuration, and Debugger.

 STM32 HAL (Hardware Abstraction Layer):

- HAL is a high-level library that abstracts the low-level peripheral registers of STM32, providing simpler functions to configure and use the peripherals.
- The benefits of HAL over direct register manipulation (ease of use, portability).
- **STM32CubeMX Overview**:

- Introduction to STM32CubeMX as a graphical tool for peripheral configuration, clock setup, and middleware integration.
- Using STM32CubeMX to select an STM32 MCU or development board.

Project Setup in STM32CubeIDE:

- Create a new project in STM32CubeIDE using STM32CubeMX.

- Choosing the right **STM32 MCU/board** (e.g., STM32F4, STM32L4, STM32F7, Nucleo boards).

Peripheral Configuration:

- Graphically configure peripherals (GPIO, UART, I2C, SPI, Timer, ADC, etc.) using STM32CubeMX.
- Clock configuration (HSE, LSE, PLL settings) using CubeMX clock configuration tool.
- Automatically generate initialization code for peripherals.

DAYS	PARTICULARS
Day 1	STM32 Basic Overview – STM32F, – OSI Layers and Applications (Advance than other controller), STM32CubeMX CONFIG, STM32CUBEIDE Programming and Debugging
Day 2	STM 32 Internal Architecture and Interface Protocol
Day 3	STM32 GPIO Configuration (LED,Switch,7-segment,LCD,Matrix keypad)
Day 4	ADC (Sensor interfacing)
Day 5	Timer and its interrupts
Day 5	CCP module(Capture, Compare, PWM)
Day 7	DAC
Day 8	Communication protocols(UART,I2C,SPI)
Day 9	I2C With OLED
Day 10	SPI with EEPROM
Day 11	CAN protocol implementation
Day 12	Bootloader overview
Day 13	Blue tooth module, GSM module interfacing
Day 14	GPS,Lora module interfacing
Day 15	RFID interfacing, Motor control
Day 16-Day 25	Project Work Prototype

Module 5: EMBEDDED (STM 32 – Baremetal) - 2 Hours / Day

Baremetal programming involves directly interacting with the microcontroller's hardware, without the use of high-level abstractions such as hardware abstraction layers (HAL) or operating systems like FreeRTOS. In

STM32 baremetal programming, you work closely with the microcontroller's registers to control peripherals and write efficient code. This gives you full control over the hardware, allowing for high performance and precision, but requires a deeper understanding of the microcontroller's architecture.

Here is a structured overview of what learning **STM32 baremetal programming** typically involves:

DAYS	PARTICULARS
Day 1	STM32 Basic Overview – STM32F, – OSI Layers and Applications (Advance than other controller), STM32CubeMX CONFIG, STM32CUBEIDE Programming and Debugging
Day 2	STM 32 Internal Architecture and Interface Protocol
Day 3	STM32 GPIO Configuration (LED,Switch,7-segment,LCD,Matrix keypad)
Day 4	ADC (Sensor interfacing)
Day 5	Timer and its interrupts
Day 5	CCP module(Capture, Compare, PWM)
Day 7	DAC
Day 8	Communication protocols(UART,I2C,SPI)
Day 9	I2C With OLED
Day 10	SPI with EEPROM
Day 11	CAN protocol implementation
Day 12	Bootloader overview
Day 13	Blue tooth module, GSM module interfacing
Day 14	GPS,Lora module interfacing
Day 15	RFID interfacing, Motor control
Day 16-Day 25	Project Work Prototype

Outcome of Learning STM32 Baremetal Programming

Upon completing this learning path, you will:

- Have a deep understanding of STM32 microcontroller architecture and peripheral handling.

- Be capable of configuring and controlling peripherals manually by writing directly to the hardware registers.
- Understand and implement interrupts, timers, communication protocols (UART, SPI, I2C) without any abstraction layers.
- Be able to write highly optimized, low-level code that is efficient and precise for embedded systems applications. This approach gives you full control over the microcontroller hardware and teaches you to write highly efficient, resource-conscious applications.