Embedded and AUTOSAR for Fresher's

Duration:	**8 Weeks**
Delivery Format:	**Classroom/Online**
Target Audience:	**Graduates with no experience**
Learning Outcomes:	**C Programming , Embedded C , Microcontrollers, Peripherals, Debugging , Automotive , Automotive Protocols- CAN & UDS, Automotive SDLC, GIT, AUTOSAR Architecture, AUTOSAR Stack Configuration , Implementation & Testing with more emphasis on Communication and Diagnostics Stack**
Entry criteria:	**Graduates with basic C programming skills.**

Training Curriculum:

Week 1 - C Programming

Objective: Refreshing C, Understanding Memory Management, Multiple Source file Handling, Data Consistency

Day 1
Brush-up C Concepts

- Why & What are the features of C Language
- How Embedded C Differ from Normal C
- What is compilation?
- Compile with Console
- IDE - Eclipse
- Hello World example
- Input and output to Console
- Data Types
- Keyword
- Variables & Storage Classes
- Constant
- Operators & Precedence
- Format Specifiers
- Escape Sequence
- Programming Errors
- Conditional Operator

- Control Statements
- Functions
- Library Functions
- Use Cases – Use cases Based on Data types, and Operators – Protocol Frame Creation
- Use Cases - Layered Architecture
- Solving Problems on DataTypes

Day 2
Functions

- Working on different types of functions with use cases
- Function Handling in Multiple. c files

Day 3
Pointers

- Using pointers to solve problems
- Using Pointers in Functions
- Using pointers for Automotive use cases
- Types of Pointers - Character pointers, double pointer etc

Day 4 Arrays

- Solving problems on Arrays
- Solving Problems on Arrays using Pointers

Day 4 Bitwise operators

- Logical Operators
- Comparing Logical Operators and Binary Operators
- Bitwise operators - AND - &, OR - |, NOT - ~, XOR - ^
- Left Shift - << :: Loose the MSB and Add 0 at LSB
- Right Shift - >> :: Loose the LSB and Add 0 at MSB
- Set a Bit
- Clear a Bit
- Toggle a Bit
- Check the bit is set or not::
- append a byte at LSB /MSB

Day 5 MACROS and Storage Classes

- Storage Classes
- Enum
- Typedef
- Constants

Day 5 Strings:

- Working with Strings

Day 6 Structures

- Structures
- Unions
- Array of structures
- Automotive Use cases

Week 2, Week 3 and Week 4 - Embedded C and AUTOMOTIVE

Embedded Exercises

1. Setting up a Project.
   Control an LED using GPIO. Write a function that implements various LED flashing routines - number of times the LED is flashed, duty cycle (ratio of On to Off time), period (On + Off time), number of flashes etc. Write a file with functions to control and LED and create a header file with function prototypes to be able to call these functions from other files, like main.c.

2. Control an LED using PWM. Write functions to set PWM frequency and duty cycle and create header file to access these functions from other files. In main, call this LED function to control the LED in various patterns like flashing at 1Hz, control brightness of the LED, fade in and fade out. Use the same function to control a buzzer, by configuring frequency and number of beeps.

3. Measure an analog voltage using ADC. Write a file that has functions to read from ADC and return the value. The function should take channel number as input and return the measured value. Write a header file with configurable values for ADC resolution, reference voltage, scaling factor etc.

4. Combine #2 and #3. Measure an analog signal and control the brightness of an LED

5. Do #1 using interrupts.

6. Measure input frequency and duty cycle using timer capture. Write a header file with function prototype of a function that will return the frequency and duty cycle values of an input signal. Use a function generator to generate a square wave with variable frequency and connect this signal to the development board through its expansion header

7. Transmit through UART. Write functions that will transmit a debug string on a UART port. Configure print to print these debug strings

8. Write a program to read and write to an RTC through I2C. Use I2C write to set the time. Use I2C read to read the time

9. Write a program to store and read data into an EEPROM using I2C.

10. Write a program to read and write to an SPI peripheral. Display data in a LCD using SPI interface

11. Write a program to measure resistance. Write a header file that has a function prototype to measure and return the resistance value.

12. Write a program to read and write to CAN bus.

13. Write a program to read switch status

14. Write a program to control a DC motor. The function should be able to control speed and direction.

15. Write a program to store and read data from Internal Flash

16. Building low level drivers for different Peripherals

17. AUTOMOTIVE and Automotive Use cases.

18. Function Pointers

19. Using Boot loaders

20. Building an application using all the concepts learned

Programming best practices

- Initializing variables
- Expose only the required functions to other modules
- Commenting
- Function headers
- File headers with copyright

Debugging Concepts

- Using JTAG or SWD
- Using debug UART

Error recovery and fault tolerant system

- Watchdog
- Cold boot, warm boot

Week 5 – Automotive and AUTOSAR (ASWC, RTE, OS, IO and Cdd)

Day 1

AUTOSAR-Intro & ASWC

- Overview and Introduction to Architecture
- Application Design in VFB Level
- Software Component
- Interfaces in AUTOSAR

Day 2

RTE & OS

- RTE Layer
- Implement RTE
- OS
- Implement the ASWC with OS and RTE

Day 3 and Day 4

RTE & OS Implementation

- Implementing the ASWC Exploring all the possible RTE implementation.
- Integration of ASWC with other stacks
- Code navigation in Rte. Demonstrating how RTE Events trigger the runnables with the help of OS
- Code navigation of access points.
- Building wrappers for code migration

Day 5

Cdd and IO Stack

- IoHwAb Layer
- DIO Driver
- PORT Driver
- PWM Driver
- ICU/OCU Driver
- ADC Driver
- Implementation- Managing the IO Stack using IO Abstraction Layer
- Complex Device Drivers
- Implementation- Managing the IO Drivers using Complex Device Drivers- Seat HeaterApplication

Evaluation of Week 1

- Requirements to software requirements
- VFB design
- Accessing the hardware peripherals through CDD and IO
- Generating the code and testing the same

Week 6 –Communication Stack

Day 1

COMMUNICATION and CAN Stack with Implementation

- Communication Module
- PDUR
- CANIF
- CAN Driver
- CAN Transceiver
- IPDUM

- CAN TP
- Implementation of Communication Stack
- Loading the dbc file and monitoring the code flow from Com Module to CAN using ECUC reference
- Compare the dbc file and the PDUs and make sure the data constraints are applied and the PDU config in stack and dbc are same.

Day 2

Configuring the Com Module, PDUR , CANIF and CAN controller

- Trace the Signal/PDU in the com stack
- Configure the Com Module for requirements
- Configure the other modules and fix all the dependency errors
- Do Data Mapping
- Generate the Code
- Differentiate Com Send and Com Receive signal Behaviour in code
- Monitor Update Bit behaviour in code
- Gateway Functionality – Signal Routing , Application Routing and PDU Routing
- Application to Check the status of the COM manager and implement the logic to transmit data only if COMM is in FULL COM

Day3

Handling Call Backs and Callouts

- Writing Call back functions for signal failure and other use cases
- Triggering IPDU callouts
- Handling Notifications
- Testing the Com stack using CANoe

Day 4

Mode Management and Implementation

1. COM Manager
2. CAN SM
3. NMIF
4. CAN NM
5. BswM
6. EcuM
7. StdM
8. SecOC
9. Implementation of Mode Management – Configuring different wakeup sources

10. Write application code to trigger the CAN bus and check the status of the Can bus before Can transmission.

Day 5

BSW Manager and ECU Manager

1. Configuring the BSW Manager for Communication Stack and a mode Switch condition as given by ANCIT team
2. ECU Manager understanding

Week 7 – Memory, Security and other managers

Day 1 and Day 2

Communication Stack Evaluation

Day 3

Memory Stack and Implementation

- EEPROM driver
- Flash Driver
- Fee
- EA
- MemIf
- NVM
- Implementation of Memory Stack from ASWC
- Boot Loader Introduction

Day 4

WatchDog Stack and Crypto

- WDG Drive
- WDG Manager
- WDG If
- Crypto Stack – CsM and CryIF
- Implementation

Day 5 and Day 6 – UDS Theory

Week 8

Diagnostics Stack

Day 1, Day 2 and Day 3

UDS, Diagnostic Stack and Implementation

1. DEM
2. DCM
3. FIM
4. DET
5. Implementation of Diagnostic Stack
    a. DTC Implementation
    b. NRC checks
    c. Service Id implementation

Day 4 and Day 5 Implementation of specific use case as evaluation