Embedded Training for Freshers

Duration:	**8 Weeks**
Delivery Format:	**Classroom/Online**
Target Audience:	**Graduates from College**
Learning Outcomes:	**C Programming, Embedded C , Microcontrollers, Peripherals, Debugging , Automotive, Automotive Protocols- CAN & UDS, Automotive SDLC, GIT**
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 to week 8- Embedded C and AUTOMOTIVE

Embedded Exercises

Week 2

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

---

Week 3

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 capture temperature from SHT21 and display on the UART

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

10. Implement Page read and write in EEPROM. Create an array of n bytes and write an algorithm to read and write the data in a particular location in EEPROM

Week 4

11. Segment a portion of the Flash and make it work as EEPROM. Store and retrieve data from the EEPROM/FLASH

12. Write a Program to monitor status of the microcontroller and if there is a flag is set→ reset the micro controller.

13. Implement micro-controller reset using Watch Dog

14. Implement Watch Dog. When the micro controller restarts capture the last stored values in RAM and load the RAM.

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

Week 5

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

17. Write a program to read switch status

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

19. Function Pointers and Call back function Implementation

20. Building low level drivers for different Peripherals. Write a Hal Driver for GPIO, ADC and CAN

==========

Week 6

21. RTOS Implementation

    a. Implement the 5 examples as implemented by Bangalore Interns

==========

Week 7

22. Intro to Automotive, V cycle. Communication protocols in Automotive. CAN protocol in detail and Write a program to read and write to CAN bus.

23. UDS protocol and Implementation → Theory and 3 services

Week 8

24. Using Boot loaders

25. 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