



COURSE BROCHURE

EMBEDDED SOFTWARE DEVELOPMENT COURSE



Hands on Training



1:1 Mentoring



**Placement
Assistance**



The **Embedded Software Development** course at **RLS Embedded Systems** is designed to provide learners with a strong foundation and practical expertise in developing software for embedded systems. The course focuses on building efficient, reliable, and real-time embedded applications that closely interact with hardware components such as microcontrollers, sensors, actuators, and communication interfaces.

This curriculum begins with the fundamentals of embedded systems, including microcontroller architecture, memory organization, and peripheral interfacing. It progressively advances to embedded C programming, interrupt handling, timers, communication protocols, and an introduction to Real-Time Operating Systems (RTOS). Special emphasis is placed on writing optimized, maintainable, and resource-efficient code suitable for real-world embedded environments.

At **RLS Embedded Systems**, the course follows a hands-on, industry-oriented learning approach. Learners work with real hardware platforms and professional development tools to gain practical experience in debugging, testing, and deploying embedded software. Through lab sessions, assignments, and mini and major projects, students develop the skills required for embedded applications in domains such as automotive systems, industrial automation, consumer electronics, and IoT.

By the end of the course, learners will be equipped with the technical knowledge, practical skills, and confidence required to design, develop, and debug embedded software solutions that meet industry standards and performance requirements.

WHAT IS EMBEDDED SYSTEMS?

An embedded system is a dedicated computing system designed to perform a specific function within a larger product or application. Unlike general-purpose computers, embedded systems are optimized for real-time operation, reliability, and efficiency. They consist of a microcontroller or microprocessor, memory, input/output peripherals, and embedded software that controls the system's functionality.

Embedded software is typically developed using programming languages such as C or C++ and interacts directly with hardware through registers, interrupts, timers, and communication interfaces. Embedded systems are widely used in various domains including automotive systems, consumer electronics, industrial automation, medical devices, and Internet of Things (IoT) applications. These systems play a crucial role in modern technology by enabling intelligent, automated, and connected products.

WHY SHOULD YOU CHOOSE RLS EMBEDDED SYSTEMS AS YOUR TRAINER?

INDUSTRY-ORIENTED EMBEDDED SYSTEMS TRAINING

The Embedded Software Development course at RLS Embedded Systems is designed and delivered by experienced industry professionals. The curriculum is aligned with current job requirements and updated regularly to match emerging technologies and industry trends.

EFFECTIVE TRAINING METHODOLOGY AND INFRASTRUCTURE

RLS Embedded Systems follows a practical training approach with strong emphasis on hands-on learning. Learners spend significant time working on lab exercises and real hardware using industry-standard embedded boards, tools, and software environments.

PRACTICAL AND PROJECT-BASED LEARNING

The course includes projects and real-time embedded applications that help learners strengthen programming, debugging, and system-level understanding through practical implementation.

PLACEMENT AND CAREER GUIDANCE

RLS Embedded Systems provides placement assistance and career guidance to help learners prepare for interviews and entry-level embedded software roles through resume support and technical mentoring.

CONTINUOUS MENTORING AND SUPPORT

Personalized mentoring and technical support are provided throughout the course. Additional guidance, mock interviews, and communication skill sessions help learners become industry-ready professionals.

COURSE DETAILS

ANALOG & DIGITAL ELECTRONICS

- Numerical Representations in Computing (Number Systems)
- Digital Logic Gates
- Design of Combinational Circuits
- Design of Sequential Circuits
- Practical Uses of Digital Circuits
- Electrical fundamentals (voltage, current, resistance)
- Passive components (R, L, C)

INTRODUCTION TO EMBEDDED SYSTEMS

- Definition and overview of embedded systems
- Characteristics of embedded systems
- Applications of embedded systems
- Classification of embedded systems
- Embedded systems vs general-purpose computing systems
- Embedded system architecture (basic block diagram)
- Embedded hardware and software components
- Microprocessors, microcontrollers, and System-on-Chip (SoC) with a comparison of microcontroller vs microprocessor

C PROGRAMMING BASICS

- Introduction to C
- Structure of a C program
- Data types
- Variables and constants
- Input / Output
- Operators
- Control statements
- Looping statements
- Arrays
- Strings
- Functions
- Basic pointers
- Type casting
- Preprocessor directives

COURSE DETAILS

C PROGRAMMING ADVANCED

- Advanced pointers
- Dynamic memory allocation
- Structures
- Unions
- Enumerations
- File handling
- Storage classes
- Command-line arguments
- Bitwise operators
- Recursion
- Data Structures
- Error handling
- Multi-file programming
- Memory management

ARM CORTEX-M4 ARCHITECTURE & EMBEDDED CONCEPTS

- Embedded C concepts
- ARM Cortex-M4 Architecture
- Overview of ARM Cortex-M4
- ARM Processor modes
- ARM Registers
- ARM Instruction set
- SysTick Timer
- NVIC (Nested Vector Interrupt Controller)
- Interrupt Service Routines (ISR)
- Timing constraints in embedded systems
- Hardware Abstraction Layer
- Embedded coding standards (MISRA overview)
- Code optimization techniques
- Embedded debugging basics

STM32 MICROCONTROLLERS & DEVELOPMENT ENVIRONMENT

- STM32 family overview
- ARM Cortex-M architecture in STM32 context
- Memory map
- Clock system (RCC)
- STM32CubeIDE
- STM32CubeMX
- ST-Link debugging

COURSE DETAILS

PERIPHERAL INTERFACES

- GPIO input/output modes
- External interrupts
- ADC
- DAC
- DMA
- Timers
- PWM generation
- Input capture
- Output compare
- Encoder interface
- Watchdog timers

COMMUNICATION PROTOCOLS

- UART
- SPI
- I2C
- CAN
- Error detection and handling
- Protocol debugging techniques

FREE RTOS

- RTOS fundamentals
- Task creation and scheduling
- Task priorities
- Context switching
- Inter-task communication
- Queues
- Semaphores and mutexes
- Software timers
- Memory management in RTOS
- Priority inversion and deadlocks
- RTOS debugging and tracing

COURSE DETAILS

PYTHON

- Variables
- Data Types
- Operators
- Loops
- Conditions
- Functions
- Lists
- Tuples
- Dictionaries
- Sets
- File Handling

LINUX BASICS

- Linux fundamentals
- Linux file system
- Basic Linux commands
- GCC compilation
- Cross-compilation basics
- Shell scripting basics
- Git version control

EMBEDDED SYSTEMS PROJECTS

- Specification Analysis
- System Design
- Coding and Testing

