# Real Time College

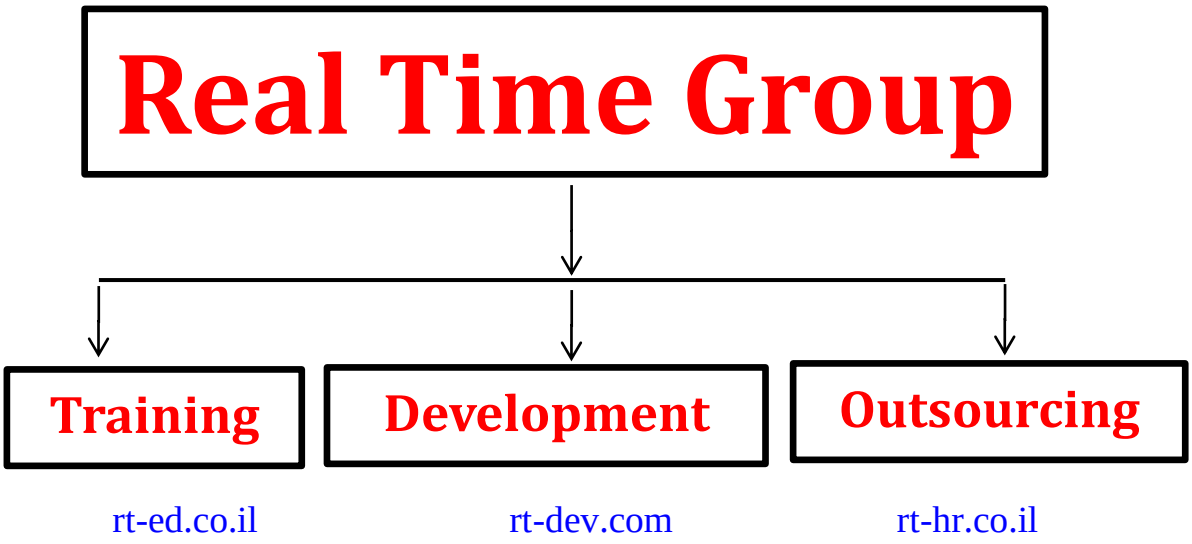
## Course: Embedded Systems (ARM)

**Duration:** 90 Academic Hours,
Hands-On-Training: 75%

Real Time Group is a multi-disciplinary dynamic and innovative Real-Time O.S. and Embedded Software Solutions Center, established in 2007.

Providing Bare-Metal and Embedded Linux solutions, professional services and consulting, end-to-end flexible system infrastructure, outsourcing, integration and training services for Hardware, Software and RT-OS \ Embedded Systems.

The company is divided into the following three Divisions:




## Training Division:

Professional Training Services for Hardware, Software, RT-OS and Embedded systems industries.

We provide the knowledge and experience needed to enable professional engineers to Develop, Integrate and QA Hardware, Software and Networking Projects.

In order to insure experience, all courses are practical – hands-on-training.
The latest Development, QA and Automation equipment which are adopted by the industry are used.

All students are supplied with Development-Boards for home-work and course projects.

## Course Overview:

The course provides a complete guide to Hand-on software development for Embedded Systems, it focuses on giving you real world coding experience by implementing ARM based Hardware Drivers.

The course will be taught using an Evaluation board for practical exercises.

During the course we'll discuss the techniques, development tools, the working environment and debugging tools, typical problems faced in embedded development and their solutions.

The participants will learn how to implement Drivers, Synchronize hardware and software input/output, use the appropriate technologies for their development needs, communicate with sensors & motors through different buses, implement Boot-loaders and peripherals such as GPIO, I2C, UART, SPI, TIMERS and more.

## Who should attend:

- This course is intended for Engineers who need to verify their HW, implement Drivers for peripheral devices, implement boot-loaders, implement user applications for Embedded Systems.
- Engineering and software students who want to break into the field of Embedded systems design.

## Prerequisite:

- Knowledge of the C programming language.
- Experience of using Linux or Windows O.S. .

# Curriculum for Embedded Systems (ARM)

1. **Introduction to Embedded Development**
    a. Overview of the course.
    b. Commonly used architectures
    c. ARM Architecture
    d. Conventional registers used
    e. Introduction to ARM Based Processors
        1) None Cortex Processors
        2) Cortex-M Based Processors
        3) Core Registers
        4) Memory Map
        5) Bit-Banding
        6) Memory Alignment
        7) Pipeline

2. **Setting-Up the Tool-Chain**
    a. Install i386 Run-Time Libraries
    b. Eclipse Installation + Eclipse Plug-Ins
    c. GCC ARM Embedded
    d. Linker file – drill down
    e. map file reading
    f. memory consideration
    g. Drivers Installation
    h. OpenOCD Installation
    i. STM32CubeMX Tool Installation

3. **STM32CubeMX Tool**
    a. Introduction to CubeMX Tool
    b. Pinout View
    c. Chip View
    d. IP Tree
    e. Clock View
    f. Configuration View

4. **The Evaluation board**
    a. Where is everything
    b. Connecting to the EVB using the serial port
    c. Downloading your Image and running it on the EVB

Lab Exercise-1: Running your first Embedded Application
Lab Exercise-2: Building a new Embedded "Hello World" Application

5. **Dealing with hardware registers**
    a. What are Hardware Registers
    b. Difference between HW registers to Memory
    c. Memory mapped I\O Registers
    d. Accessing Hardware Registers

Lab Exercise-3: Reading & Writing to registers

6. **GPIO (General Purpose Input Output).**
    a. What are GPIO pins and how to use them
    b. Configuring GPIO registers
    c. GPIO Mode
    d. GPIO Alternate Function
    e. GPIO Speed
    f. Driving a GPIO
    a. Reading from \Writing to GPIO Data Registers
    b. What is a Pull-Up Register, when do we need it?
    c. Using the Electronic schemas

Lab Exercise-4: Turning the LED on\off\blinking
Lab Exercise-5: Using Buttons to Turn on\off\blink the LED

7. **Interrupts Management**
    a. What are interrupts and in what aspect are they used?
    b. What's the difference between Hardware and Software Interrupts
    c. NVIC Controller
    d. Vector Table
    e. Configuring Interrupts With CubeMX
    f. Masking Interrupts

Lab Exercise-6: Implementing timers with Interrupts
Lab Exercise-7: Implementing UART with Interrupts

8. **Direct Memory Access (DMA)**
    a. hat is DMA and what is it used for ?
    b. The DMA Controller
    c. Transferring in Polling Mode
    d. Transferring in Interrupt Mode
    e. Transferring between Peripherals
    f. Using CubeMX to Configure DMA Requests
    g. Memory Allocation of DMA Buffers

9. **High resolution Timers**
    a. What are Timers, and where are they used?
    b. Calculating the Timer Input frequency
    c. Using Timers in Polling Mode
    d. Using Timers in Interrupt Mode
    e. Using CubeMX to Configure Timers
    f. understanding different timer's configuration

Lab Exercise-8: Implementing a mSecond delay function

10. **Pulse-width modulation (PWM)**
    a. What is Pulse width Modulation and where is it used?
    b. How Can we Create PWM with timers.
    c. Using CubeMX to Configure the PWM Mode.

Lab Exercise-9: Implementing PWM

11. **UART- RS232 Serial Communication**
    a. What is the Universal Asynchronous Receiver/Transmitter and where is it used?
    b. Synchronous communication ver Asynchronous communication.
    c. UART Protocol – Start bit, Data bits, Parity, Stop bit
    d. UART Configuration Using CubeMX
    e. UART Communication in Polling Mode
    f. UART Communication in Interrupt Mode
    g. Hardware and Software Flow control protocols

Lab Exercise-10: Implementing UART communication in polling mode.
Lab Exercise-11: Implementing UART communication in Interrupt mode.

## 12. The Real-Time Clock (RTC)

a. What is the RTC and what is it used for?
b. Reading from the RTC registers
c. Supporting A Leap Year Generator

Lab Exercise-12: using the RTC for timing.

## 13. The Analog to Digital Converter (ADC)

a. What is the ADC, what is good for?
b. Sampling and Conversion issues
c. Sensitivity of the ADC
d. Conversion Modes – Single\Continues
e. ADC Resolution and Conversion Speed
f. A/D Conversions in Polling
g. A/D Conversions in Interrupt
h. A/D Conversions in DMA Mode.
i. Possible Error issues in ADC
j. Using CubeMX to Configure ADC Peripheral

Lab Exercise-13: Using the ADC .

## 14. The Watchdog Timer (WDT)

a. What is the WDT used for?
b. Calculating the required time
c. Setting the Watch dog timer in the mini2440 EVB

Lab Exercise-14: using the Watchdog timer.

## 15. The Inter Integrated Circuit (I²C)

a. What is the I²C and what is it used for?
b. The I²C protocol
c. I²C Start and Stop Conditions
d. Using the I²C Peripheral in Master
e. Using the I²C Peripheral in Slave Mode
f. Using CubeMX to Configure the I²C Peripheral

Lab Exercise-15: Using the I2C bus.

## 16. The Serial Programming Interface (SPI)

a. The SPI specification and what is it used for?
b. Using SPI as Slave device
c. Using CubeMX to Configure the I²C Peripheral

Lab Exercise-16: Using the SPI bus.

## 17. The Memory Management Unit (MMU)

a. What is the MMU and what is it in charge of?
b. Bare-Metal ver O.S. based Memory modals
c. The Translation look-aside buffer (TLB)

## 18. Debugging & Tracing

a. Using JTAG for debugging.
b. Break points, watch windows..
c. Call stack usage
d. ARM Semi-hosting
e. Debugging using external equipment (Scope, Logic Analyzer..)
f. Conditional break points and how to use SW & HW breakpoints
g. Add hard faults and asserts – what are they and how to use them
h. ARM Semi-hosting
i. Debugging using external equipment (Scope, Logic Analyzer..)