PARUVATHTHE PAYIR SEI®
ENCOURAGE ENRICH ENLIGHTEN

10/12/2025

2Embedded Systems Training using STM32 Microcontroller

1. Program Overview

Program Title:

Embedded Systems using STM32 Microcontroller

Target Audience:

B.E / B.Tech – Electrical & Electronics Engineering (EEE) Students

Duration:

6 Days (36 Hours – 6 Hours per Day)

Pre-requisites:

- Basic Electrical & Electronics
- Fundamentals of C Programming (Desirable)

2. Program Objectives

This program aims to:

- Build strong embedded system fundamentals for EEE students
- Provide deep understanding of STM32 architecture
- Explain clear mapping between software (C code) and hardware behavior
- Prepare students for embedded system placements and core jobs
- Enable students to interface basic sensors and peripherals confidently

3. Expected Outcomes

At the end of the program, students will be able to:

- Explain ARM Cortex-M architecture
- Understand STM32 internal block diagram & memory map
- Configure and control GPIO, Timers, ADC, UART
- Relate each line of Embedded C code to hardware functionality
- Interface simple sensors and peripherals
- Answer placement-oriented embedded system questions confidently

4. Tools & Platform

- Microcontroller: STM32 (STM32F103 / Nucleo series)
- IDE: STM32CubeIDE
- Programming Language: Embedded C
- Debugger: ST-Link
- Peripherals: LED, Push Button, LDR / Potentiometer, UART

5. Day-Wise Detailed Syllabus

DAY 1 – Introduction to Embedded Systems & STM32 Architecture

Theory:

- What is an Embedded System?
- Embedded Systems in EEE Applications
- Microcontroller vs Microprocessor
- Introduction to ARM Cortex-M Architecture
- RISC vs CISC
- Harvard Architecture
- STM32 Internal Block Diagram
- Flash, RAM, Registers
- Clock System Overview
- Reset & Boot Process

Hands-On:

- STM32CubeIDE installation
- Exploring STM32 datasheet & reference manual
- Identifying memory map and GPIO registers

Outcome:

✓ Students understand how a program executes inside STM32

DAY 2 – GPIO & Digital Output (LED Interfacing)

Theory:

- GPIO pin internal structure
- Pin modes (Input, Output, Alternate Function)
- Push-Pull vs Open-Drain
- Pull-up and Pull-down resistors
- Peripheral Clock Enable concept

Hands-On:

- LED interfacing
- GPIO configuration using CubeMX
- LED blink program
- Understanding HAL functions and underlying registers

Outcome:

✓ Students understand how C code controls voltage at a pin

DAY 3 – Digital Input (Button Interfacing & Logic Control)

Theory:

- Digital input pin operation
- Floating input problem
- Hardware debouncing concept
- Polling vs Interrupt (Introduction)

Hands-On:

- Push button interfacing
- Reading input pin status
- LED control using button
- Simple decision-making logic in embedded C

Outcome:

✓ Students understand hardware signal → software logic → output action

DAY 4 – Timers & Delay Concepts

Theory:

- What is a Timer?
- Timer block diagram
- Prescaler, Counter, Auto-reload
- Software delay vs Hardware timer
- Importance of timers in real-time systems

Hands-On:

- LED blinking using HAL_Delay()
- LED blinking using Timer interrupt
- Observing timing accuracy

Outcome:

✓ Students understand real-time behavior & timer-based execution

DAY 5 – Analog Interfacing (ADC + Sensor)

Theory:

- Analog vs Digital signals
- ADC working principle
- Resolution, reference voltage
- Sampling and quantization
- ADC block inside STM32

Hands-On:

- Interfacing LDR / Potentiometer / LM35
- Reading ADC value
- Converting ADC count to voltage
- Threshold-based LED control

Outcome:

✓ Students understand sensor → ADC → register → software processing

DAY 6 – Serial Communication (UART) & Mini Project

Theory:

- Need for communication in embedded systems
- UART working principle
- Baud rate concept
- TX & RX hardware

Hands-On:

- UART configuration
- Sending sensor data to PC
- Viewing output in serial monitor

Mini Project:

Basic Embedded Monitoring System

- Digital input (Button)
- Digital output (LED)
- Timer-based operation
- Analog sensor input
- UART-based data display

Outcome:

✓ Students integrate all learned concepts into one system

6. Teaching Methodology

- Architecture-first approach
- Register-level explanation (conceptual)
- Code-to-hardware mapping
- Live debugging & signal explanation
- Question-driven learning (placement focus)

7. Assessment & Evaluation

- Daily hands-on verification
- Oral explanation of code logic
- Hardware–software relation questioning
- Final mini-project demonstration

8. Career & Placement Relevance

This program supports roles such as:

- Embedded Engineer
- Firmware Engineer
- Junior Hardware Engineer
- Automation & Control Engineer

9. Conclusion

This training builds conceptually strong embedded engineers with:

- Clear STM32 architecture understanding
- Strong Embedded C fundamentals
- Hardware–software integration skills
- Industry & placement readiness