✅ COURSE DETAILS

Course Name: STM32 Bare Metal 2026 – Live Bootcamp (Project-Based & Real-Time Mentorship)

Mode: Online Live Training
Level: Beginner to Advanced
Duration : 3 Months
100% Project Based
Lifetime Access to Materials
Certificate of Completion

👉 **Course Link:** STM32 Bare Metal 2026 – Live Bootcamp (Project-Based & Real-Time Mentorship)

COURSE FEES

Original Price: ₹4499
Today's Price: ₹ 3499

INSTRUCTOR

Name :- Chandan (STM32 Instructor)
Experience: 5+ Years in Embedded Systems
Engineer at Mother-son technology

Why This STM32 Course?

Arduino-level knowledge is not enough for **Embedded jobs.**
This course trains you exactly how **companies design, code, debug, and test STM32 firmware** using industry tools.
Learn **STM32** the way it is used in **real products**, not classroom demos.

🎯 Who Can Enrols?

A. Working professionals in **Embedded / ECE / EEE** domains
B. Engineers upgrading from **Arduino / PIC / ECP32/ 8051**
C. Professionals preparing for **Embedded Firmware** Engineer roles
D. Engineers working in **Automation, IoT, Industrial & Automotive domains**
E. **Embedded Engineer** Looking to Up-Skill
F. **Automotive Engineer** Looking to UP-skill
G. **Firmware Engineer** Looking to UP-skill

Electro4u

Module 01: STM32 Fundamentals

- What is a Microcontroller?
- MCU internal building blocks
- CPU, Memory, Peripherals & Buses
- What does “Bare Metal” mean?
- Bare Metal vs HAL vs LL
- Why learn STM32 Bare Metal?
- Typical STM32 application areas
- Embedded firmware execution flow

Module 02: Introduction to STM32

- Introduction to STMicroelectronic
- STM32 product ecosystem
- Cortex-M based STM32 overview architecture
- STM32Cube ecosystem overview (Installation of CubeIdes & Drivers)
- STM32 documentation ecosystem
- Datasheet vs Reference Manual vs Programming Manual

Module 03: STM32 Architecture

- CPU core overview
- Registers inside Cortex-M
- ALU and Control Unit
- Link Register
- General Purpose Registers

Module 04: STM32 Families

- STM32 family overview
- STM32F Series
- STM32G Series
- STM32H Series
- STM32L Series
- STM32U Series
- STM32C Series
- Performance vs power-oriented families
- Selecting the right STM32 MCU
- Understanding MCU datasheet specifications

visit now www.electro4u.net

Module 05 : STM32 Naming Conventions

i) Understanding STM32 part numbers
ii) Family identification
iii) Series identification
iv) Package identification
v) Flash/RAM identification
vi) Pin-count identification
vii) Temperature/grade information

Module 06: STM32 bus architecture

i) What is a system bus?
ii) ARM AHB architecture
iii) APB1
iv) APB2
v) AHB/APB relationship
vi) Peripheral bus mapping
vii) Bus clock configuration
viii) Bus frequency limitations
ix) Peripheral clock enable
x) Understanding the STM32 bus matrix
xi) Practical register-level bus analysis

Module 07: STM32 Clock System

i) Why clock is important?
ii) Internal vs external clock
iii) HSI
iv) HSE
v) LSI
vi) LSE
vii) PLL
viii)Clock tree
ix) SYSCLK
x) HCLK
xi) PCLK1
xii) PCLK2

Electro4u

xiii) Prescalers
xiv)Peripheral clocks
xv) Clock switching
xvi)Clock configuration through registers
xvii)Calculating peripheral clock frequency
xviii) Practical clock configuration

Module 08: STM32 Boot Sequence

i) What happens after reset?
ii) Reset sources
iii) Boot ROM
iv) Bootloader
v) Flash memory startup
vi) Vector table
vii) Reset Handler
viii) Stack initialisation
ix) Startup code
x) System initialisation
xi) `main()` execution
xii) Boot configuration
xiii) Boot pins
xiv) STM32 system boot-loader
xv) Understanding startup files

Module 09: STM32 Pin Configuration

i) GPIO fundamentals
ii) GPIO port architecture
iii) GPIO registers
iv) GPIO Pin Swapping
v) Input mode
vi) Output mode
vii) Alternate Function mode
viii) Analog mode
ix) Push-pull vs Open-drain
x) Pull-up and Pull-down
xi) GPIO speed
xii) Alternate Function mapping
xiii) Pin multiplexing
xiv) Reading STM32 datasheet pin tables
xv) Register-level GPIO configuration

Project 9.1: STM32 GPIO & Pin Configuration — Register-Level Implementation & Live Demonstration

Electro4u

Module 10: STM32 debugging techniques

i) What is embedded debugging?
ii) Debug vs Release build
iii) Breakpoints
iv) Hardware breakpoints
v) Watchpoints
vi) Step Into
vii) Step Over
viii) Step Out
ix) Register inspection
x) Memory inspection
xi) Peripheral register debugging
xii) Variable/watch window
xiii) HardFault analysis
xiv) Using SWD for debugging
xv) Practical debugging exercises

Project 10.1: STM32 Embedded Debugging & Fault Analysis — Practical SWD Debugging Demonstration

Module 11: STM32 Register Overview

i) What is a hardware register?
ii) Memory-mapped registers
iii) Register addresses
iv) Register width
v) Bit fields
vi) Read/Write permissions
vii) Reserved bits
viii) Register reset values
ix) RCC registers
x) GPIO registers
xi) EXTI registers
xii) NVIC registers
xiii) Timer registers
xiv) UART registers
xv) ADC registers
xvi) Register-level programming methodology
xvii) Reading a Reference Manual effectively

Project 11.1: STM32 Register-Level Peripheral Control — Hands-On Implementation & Live Demonstration

Module 12: STM32 Project Overview

i) STM32CubeIDE setup
ii) MCU selection
iii) Toolchain overview
iv) GCC ARM compiler
v) Linker
vi) Startup files
vii) CMSIS
viii) Project structure
ix) Build process
x) Compilation
xi) Linking
xii) ELF/BIN/HEX files
xiii) Flashing firmware
xiv) Debug configuration
xv) Build configuration
xvi) Debug vs Release

Project 12.1: STM32 Firmware Development Pipeline — Build, Flash & Debug

Module 13: STM32 Digital I/O

i) GPIO architecture
ii) GPIO input
iii) GPIO output
iv) LED control
v) Button interfacing
vi) Debouncing
vii) Bit manipulation(Bit Set/Reset)

Project 13.1: GPIO Input — Button State Monitor

Read a push-button state and detect **HIGH/LOW** using GPIO input.

Project 13.2 : GPIO Output — Digital Output Controller

Configure GPIO pins and control digital outputs directly

Project 13.3: LED Control — Single & Multiple LED Controller

- Single LED ON/OFF
- LED blinking
- Multiple LED control
- Different LED patterns

Electro4u

Project 13.4: Button Interfacing — Button-Controlled LED

Use a push button to control an LED:
Button Press → LED ON
Button Release → LED OFF

Project 13.5. Debouncing — Reliable Push-Button Interface

Implement software debouncing so one physical press produces one reliable digital event.

Project 13.6. Bit Manipulation — Binary LED Controller

Use **bit set, bit clear, bit toggle, masking, and shifting** to control multiple LEDs.

Module 14: STM32 Interrupts

i) Why interrupts?
ii) Polling mode
iii) polling vs Interrupt
iv) Why interrupts?
v) Interrupt sources
vi) Cortex-M exception model
vii) NVIC
viii) Interrupt priority
ix) Preemption
x) EXTI
xi) GPIO external interrupt
xii) ISR
xiii) Interrupt vector table
xiv) Nested interrupts
xv) Interrupt latency
xvi) Clearing interrupt flags
xvii) Common interrupt mistakes
xviii) Practical interrupt-based project

Project 14.1: STM32 Multi-Interrupt Event Management System

Subtitle:
EXTI, NVIC Priority, Preemption & Nested Interrupts — Hands-On Implementation

Electro4u

Module 15: STM32 Timers, PWM & Input Capture

i) Timer fundamentals
ii) Timer architecture
iii) Prescaler
iv) Counter
v) Auto-reload register
vi) Timer overflow
vii) Timer interrupts
viii) Output Compare
ix) PWM generation
x) PWM frequency calculation
xi) PWM duty-cycle calculation (LED brightness control)
xii) Input Capture
xiii) Measuring frequency

Project 15.1 — STM32 Timer-Based LED Blinker

Steps: Clock → Prescaler → Counter → ARR → Overflow → Timer Event

Project 15.2 — STM32 PWM LED Brightness Controller

Steps: Timer → PWM → Frequency → Duty Cycle → LED Brightness

Project 15.3 — STM32 RGB LED Colour Controller

Steps: PWM Channel 1 + PWM Channel 2 + PWM Channel 3 → RGB Colour Mixing 🌈

Project 15.4 — STM32 Input Capture Frequency Meter

Steps: Period → Frequency → Signal Analysis

Module 16: UART Protocol

i) UART fundamentals
ii) Baud rate
iii) Start/Stop bits
iv) Parity
v) UART registers
vi) Polling UART
vii) Interrupt-driven UART
viii) UART debugging
ix) UART DMA

Project 16.1: Transmit & Receive Data using UART Polling Mode
Project 16.2: Transmit & Receive Data using UART Interrupt mode
Project 16.3: Transmit & Receive Data using UART DMA Mode High-Speed

Electro4u

Module 17: SPI

i) SPI architecture
ii) Master/Slave
iii) SCK
iv) MOSI
v) MISO
vi) CS
vii) SPI modes
viii) Clock polarity & phase
ix) SPI register configuration
x) Sensor communication

Project 17.1: Transmit & Receive Data using SPI Polling Mode
Project 17.2: Transmit & Receive Data using SPI Interrupt Mode
Project 17.3: Transmit & Receive Data using SPI DMA Mode High-Speed

Module 18: I²C

i) I²C architecture
ii) SDA/SCL
iii) Start/Stop conditions
iv) Addressing
v) ACK/NACK
vi) Master/Slave
vii) Pull-up resistors
viii) I²C register configuration
ix) Sensor communication

Project 18.1: Transmit & Receive Data using I²C Polling Mode
Project 18.2: Transmit & Receive Data using I²C Interrupt Mode
Project 18.3: Transmit & Receive Data using I²C DMA Mode High-Speed

Module 19: USB CDC

x) USB fundamentals
xi) USB device architecture
xii) USB endpoints
xiii) USB descriptors
xiv) USB device classes
xv) STM32 USB peripherals
xvi) USB communication overview
Note: Physical without Demonstration

Module 20: STM32 Analog Systems

i) Analog vs Digital signals
ii) ADC fundamentals
iii) ADC architecture
iv) Resolution
v) Reference voltage
vi) Sampling
vii) Conversion time
viii) ADC channels
ix) ADC registers
x) Single conversion
xi) Continuous conversion
xii) Scan mode
xiii) ADC interrupts
xiv) DMA-based ADC
xv) Voltage measurement
xvi) Sensor interfacing
xvii) ADC calculation

Project 20.1: STM32 ADC-Based Voltage & Sensor Monitoring System — Practical Data Acquisition

Module 21: STM32 Low-Power & Sleep Modes

i) Why low-power design matters
ii) Sleep mode
iii) Stop mode
iv) Standby mode
v) Shutdown mode
vi) Clock gating
vii) Wake-up sources
viii) External interrupt wake-up
ix) RTC wake-up
x) Timer wake-up
xi) Power consumption measurement
xii) Low-power firmware architecture

Project 21.1: STM32 Low-Power Wake-Up & Power Management System

Active Mode → Sleep/Stop → Low Power → Wake-Up Event → Resume→ Measure Power Consumption 🔋⚡

Module 22: STM32 Power modes

i) STM32 power architecture
ii) Run mode
iii) Sleep mode
iv) Stop mode
v) Standby mode
vi) Shutdown mode
vii) Clock gating
viii) Voltage scaling
ix) Wake-up sources
x) Power consumption considerations
xi) Choosing the appropriate power mode
xii) Program Counter
xiii) Stack Pointer

Final Project 22.1: STM32 Intelligent Low-Power Embedded Control & Monitoring System

GPIO → Interrupts → Timers/PWM → UART → SPI → I²C → ADC → Debugging → Register-Level Programming → Low-Power Modes → Wake-Up → Power Management

Optional Advance Module

Project 23: STM32 GSM & GPS-Based Location Tracking System
Project 24: STM32 GSM Modem Control Using AT Commands
Project 25: STM32 GSM-Based TCP Data Transmission System
Project 26: STM32 GSM-Based HTTPS Data Communication System
Project 27: STM32 GSM-Based Voice Call & Call Control System

visit now www.electro4u.net

STM32 Project List

1. **Project 9.1:** STM32 GPIO & Pin Configuration — Register-Level Implementation
2. **Project 10.1:** STM32 Embedded Debugging & Fault Analysis — Practical SWD Debugging
3. **Project 11.1:** STM32 Register-Level Peripheral Control — Hands-On Implementation
4. **Project 12.1:** STM32 Firmware Development Pipeline — Build, Flash & Debug
5. **Project 13.1:** GPIO Input — Button State Monitor
6. **Project 13.2:** GPIO Output — Digital Output Controller
7. **Project 13.3:** LED Control — Single & Multiple LED Controller
8. **Project 13.4:** Button Interfacing — Button-Controlled LED
9. **Project 13.5:** Debouncing — Reliable Push-Button Interface
10. **Project 13.6:** Bit Manipulation — Binary LED Controller
11. **Project 14.1:** STM32 Multi-Interrupt Event Management System
12. **Project 15.1:** STM32 Timer-Based LED Blinker
13. **Project 15.2:** STM32 PWM LED Brightness Controller
14. **Project 15.3:** STM32 RGB LED Colour Controller
15. **Project 15.4:** STM32 Input Capture Frequency Meter
16. **Project 16.1:** Transmit & Receive Data using UART Polling Mode
17. **Project 16.2:** Transmit & Receive Data using UART Interrupt Mode
18. **Project 16.3:** Transmit & Receive Data using UART DMA Mode High-Speed
19. **Project 17.1:** Transmit & Receive Data using SPI Polling Mode
20. **Project 17.2:** Transmit & Receive Data using SPI Interrupt Mode
21. **Project 17.3:** Transmit & Receive Data using SPI DMA Mode High-Speed
22. **Project 18.1:** Transmit & Receive Data using I²C Polling Mode
23. **Project 18.2:** Transmit & Receive Data using I²C Interrupt Mode
24. **Project 18.3:** Transmit & Receive Data using I²C DMA Mode High-Speed
25. **Project 20.1:** STM32 ADC-Based Voltage & Sensor Monitoring System Practical Data Acquisition
26. **Project 21.1:** STM32 Low-Power Wake-Up & Power Management System
27. **Final Project 22.1:** STM32 Intelligent Low-Power Embedded Control & Monitoring System

Optional Advanced Modules

Project 23: STM32 GSM & GPS-Based Location Tracking System

Project 24: STM32 GSM Modem Control Using AT Commands

Project 25: STM32 GSM-Based TCP Data Transmission System

Project 26: STM32 GSM-Based HTTPS Data Communication System

Project 27: STM32 GSM-Based Voice Call & Call Control System

Electro4u

📚 HOW TO REGISTER

Step 1 — Login

Visit **www.electro4u.net** and log in via Google.

Step 2 — Enroll & Pay

Open the course link → Scroll to the payment section → Complete payment.
Link : **https://electro4u.net/course/stm32-mastery-2026-project-based-job-ready**

Step 3 — Get Access

Send your **payment confirmation** to our support team to receive class & course details.

🔗 Direct Registration

Complete payment through the provided payment link and share the payment confirmation with us.
Direct Payment link : **https://rzp.io/rzp/R5q7aBK2**

📞 Support

📧 info@electro4u.net

📱 +91 63648 21851, +91 91232 88116

Electro4u

visit now **www.electro4u.net**