ORIGIN ACADEMY · COURSE SYLLABUS

Embedded Systems Engineer

From your first line of C to shipping firmware on ARM Cortex-M and embedded Linux, with a hardware kit, an enterprise capstone and a job-readiness rating.

30 weeks · about 15 hours a week · live cohort of up to 10 · hardware kit included

Curriculum last updated 19 November 2026

Availability

This course opens soon. Join the waitlist on the course page to hear first about dates and prices.

Before you start

- No coding experience needed: Phase 0 starts from the basics
- English good enough to follow technical documentation and live sessions: about a credit (C6 or better) in WASSCE or NECO English Language, or an equivalent such as KCSE or IGCSE English (roughly CEFR level B1). You don't need the certificate: Phase 0 is free, so you can check for yourself
- A laptop running Linux, macOS, or Windows with WSL

How the course works

- A live cohort of up to 10 students, with a mentor.
- At least half your time is hands-on: every module ends with a lab.
- Live sessions are recorded, so you can rewatch them or catch up.
- Questions between sessions are answered within 24 hours on weekdays.
- A mentor reviews every phase project, like a pull request at work.

Phase 0 is free for everyone, before you enroll.

Curriculum

PHASE 0 · WEEKS 1–2 · FREE

Foundations

Linux CLI, Git, how computers work, number systems, bitwise logic.

- How computers work: CPU, memory and buses
- Number systems and bitwise logic
- The Linux command line
- Git and GitHub workflow
- Setting up your embedded toolchain

You build: Dev environment + first commit

PHASE 1 · WEEKS 3–7

C for Embedded

Pointers, memory layout, volatile, bit manipulation, structs/unions, the build process and linker scripts.

- C the embedded way
- Pointers and memory layout
- Fixed-width types, const and volatile
- Bit manipulation and register access
- Structs, unions and memory-mapped I/O
- The build: compiler, linker scripts and Make
- Unit testing C with Unity

You build: Ring buffer + register bit-field library with unit tests

PHASE 2 · WEEKS 8–10

Electronics Essentials

Ohm's law, pull-ups, transistors, reading datasheets and schematics, multimeter and logic analyzer.

- Voltage, current and Ohm's law
- Dividers, pull-ups and pull-downs
- Diodes, transistors and driving loads
- Reading datasheets and schematics
- Using a multimeter and logic analyzer

You build: Breadboard circuits, debounced inputs

PHASE 3 · WEEKS 11–15

Microcontrollers & ARM

Cortex-M architecture, memory map, GPIO, clocks, NVIC, timers, bare-metal startup, CMSIS.

- Cortex-M architecture and the memory map
- Clocks and reset
- GPIO from the registers up
- Interrupts and the NVIC
- Timers and SysTick
- Bare-metal startup code and CMSIS

You build: Bare-metal blinky, interrupt-driven input

PHASE 4 · WEEKS 16–19

Peripherals & Protocols

UART, SPI, I2C, ADC, PWM, DMA, CAN intro.

- UART
- SPI
- I2C
- ADC and PWM
- DMA
- CAN bus basics

You build: Sensor driver from the datasheet

PHASE 5 · WEEKS 20–21

RTOS & Real-Time

FreeRTOS tasks, queues and semaphores, Zephyr, and what real-time means on Linux with PREEMPT_RT.

- Why an RTOS: tasks and scheduling
- FreeRTOS tasks, queues and semaphores
- Zephyr: devicetree and Kconfig
- Real-time Linux with PREEMPT_RT

You build: Multi-task data logger

PHASE 6 · WEEKS 22–23

Embedded Linux

Cross-compiling toolchains, the boot flow and U-Boot, building the kernel and Device Tree, a root filesystem from scratch with BusyBox, filesystems for flash and SD, Buildroot, licence compliance and debugging on the target.

- How an embedded Linux system fits together
- Cross-compiling toolchains and C libraries
- The boot flow: ROM code, U-Boot and the kernel
- Configuring and building the kernel and Device Tree
- A root filesystem from scratch with BusyBox
- Filesystems for SD and flash: ext4, SquashFS, UBIFS
- Building the whole system with Buildroot
- Open-source licences and compliance
- Debugging on the target: gdbserver, strace and perf

You build: Custom Linux image for an ARM board, built with Buildroot

PHASE 7 · WEEKS 24–25

Professional Practice

MISRA C, static analysis, unit testing, firmware CI, GDB/JTAG, low power, security, OTA.

- MISRA C and static analysis
- Firmware CI
- Debugging with GDB and SWD/JTAG
- Low-power design
- Firmware security and OTA updates

You build: Harden an earlier project

PHASE 8 · WEEKS 26–30

Enterprise Capstone

Team project to an enterprise brief: sprints, code review, design doc, panel demo.

- Design doc and sprint planning
- Sprints with code review
- Integration and field testing
- Panel demo and readiness review

You build: Industrial sensor node with RTOS, wireless uplink, OTA and CI

Your certificate

You earn it by doing two things:

- Attend every live session. If you miss one, make it up by watching the recording and submitting that week's lab within 7 days. You can do this up to 2 times per course.
- Score more than 50% on the final evaluation at the end of the course.

Your Enterprise Readiness Rating is separate: it scores your projects, capstone and assessments so employers can see how job-ready you are.

Questions

Email academy@originclouds.io, or see the course online: https://academy.originclouds.io/courses/embedded-systems-engineer