AI1 - AI-Assisted Embedded Development

Master AI coding agents for embedded systems

Objectives

- Use AI coding assistants (Claude Code, Copilot and Codex)
- Prompting to validated Firmware
- Spec-driven development
- Functional Specification Document (FSD)
- Configure MCP servers
- Design AI agents
- Build reusable Skills
- Validate firmware on target
- Apply IP and data rules

Prerequisite

- C language
- Embedded target experience
- Git basics
- Command line
- No AI experience needed

Course Environment

- Theoretical course
  - PDF material in English (printed for face-to-face); online over Teams.
  - Trainer assistance throughout.
- Practical activities (40-50% of duration)
  - Code examples, exercises and solutions.
  - Remote: one online Linux PC per trainee, with emulated or physical board as needed.
  - Face-to-face / on-site: one PC (one per two beyond six trainees), target board and install manual as needed.
- Downloadable preconfigured VM to redo the labs afterwards.
- Each session starts with a trainee check-in.

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

The AI Tooling Landscape for Embedded Engineers

- AI assistant families
- Interaction modes
- Why embedded is different
- From coding to orchestrating
- Strengths and limitations

Exercise: give the same driver prompt to AI tools and compare the results

Prompting for Embedded Problems

- Anatomy of a prompt
- Few-shot prompting
- Iterative refinement
- Prompting with embedded artefacts
- Reviewer-mode prompting
- Plan before code
- Detecting hallucinations

Exercise: rewrite three requests as structured prompts and measure the improvement

Spec-Driven Development and Functional Specification Documents

- Why specs come first
- Anatomy of a spec
- From requirement to spec
- Spec as contract
- Living specifications
- Role of the FSD
- FSD structure
- Generating a first-draft FSD
- Reviewing the FSD
- Iterating the FSD
- Handing off the FSD

Exercise: turn a feature request into a structured specification
Exercise: find the hidden assumptions in a requirement and make them explicit
Exercise: generate a first-draft FSD with Claude
Exercise: review and harden the FSD into a ready-to-build version

AI Assistants in Your IDE and CLI

- GitHub Copilot in VS Code
- Claude in VS Code
- Configuring the IDE
- Privacy and licensing
- What a CLI assistant adds
- Scaffolding a new firmware project
- Claude Code
- OpenAI Codex CLI
- Project memory files
- Encoding the constraints
- Permissions and autonomy
- Slash and custom commands
- Hooks (build, clang-format, lint)
- Plan Mode and Extended Thinking
- Claude Code on the web
- Context-window economics

Exercise: set up Copilot and Claude in VS Code, then build an I²C driver with each and compare
Exercise: configure CLAUDE.md and test hooks, then let Claude Code build a small driver

Second Day

Datasheets, Reference Manuals and Project Context

- The datasheet problem
- Feeding the right pages

- Per-project knowledge corpus
- Trust heuristics

Exercise: generate a DMA config

Model Context Protocol (MCP)

- MCP as the agent's interface
- MCP server categories
- Designing the MCP toolbox
- Writing an MCP server
- Sharing MCP configs

Exercise: connect Claude Code to three MCP servers and verify each one
Exercise: write a small MCP server

Designing AI Agents for Embedded Development

- What an agent is
- The agent loop
- Single vs multi-agent
- Subagents and delegation
- Autonomy boundary
- Agent configurations
- Failure modes

Exercise: build a two-agent workflow (one implements the FSD, the other reviews it)
Exercise: run an agent on a build-flash-test loop, inject a failure, and watch it recover

Third Day

Skills, Reusable Configuration and Multi-Tool Orchestration

- The Skill concept
- Skills for embedded teams
- Installing and using plugins
- Plugins vs Skills
- Project-level config files
- Encoding team standards
- Versioning and sharing Skills
- Three tools, one workflow
- Where each fits
- Switching tools mid-task
- Decision guide
- Cost and licensing trade-offs
- IP and data residency

Exercise: write a Skill that turns a one-line request into an FSD, and test it on three cases
Exercise: write a CLAUDE.md of coding conventions and see how Claude Code's output changes
Exercise: take one feature through all three tools (Claude for the FSD, Codex to build, Copilot to refactor)

Validating AI-Generated Embedded Code

- What review means
- Hardware failure modes
- Concurrency failure modes
- Timing failure modes
- Review workflow

Exercise: find and fix three planted defects (register, concurrency and timing) in a generated driver
Exercise: have a second AI tool review the first's output and compare what each missed

Data Handling and IP

- IP and data handling
- Your company’s AI policy

Exercise: classify ten scenarios as safe, borderline or forbidden, and draft a one-page house rule