Fiche du cours

70 h

Titre :

CPP410 - C++ Embarqué & Temps Réel

Description :

Développer du C++ moderne pour l'embarqué ARM Cortex-M en environnement temps réel. De la mise en route bare-metal (startup, linker, registres) à l'OS temps réel (FreeRTOS ou Zephyr), en passant par les drivers matériels (GPIO, UART/I²C/SPI, ADC/PWM, TIMERS, DMA), la gestion des interruptions, la mémoire sans heap, la sécurité & robustesse (watchdog, brown-out, CRC), les règles MISRA C++ et l'industrialisation (CMake cross-compile, logs, traces, tests sur cible, CI de base).
Capstone : un firmware temps réel complet "capteurs → traitement → protocole", mesuré (latence/jitter) et packagé.

Objectifs :

- Toolchain & mémoire : projet CMake cross-compile, compréhension startup, linker script et carte Flash/SRAM.*
- Drivers sûrs : C++ moderne (RAII sans heap) pour GPIO/UART/I²C/SPI/ADC/PWM/TIMERS, avec DMA.*
- Interruptions : NVIC, priorités, latence, et sections critiques non bloquantes.*
- RTOS : FreeRTOS/Zephyr (tâches, queues, semaphores, timers), priority inversion, tickless & low-power.*
- Architecture & robustesse : sans allocation dynamique (ou contrôlée) — std::array, std::span, pools — + watchdog, brown-out, logs non bloquants, recovery.*
- Qualité & validation : MISRA C++ (principes), revue de code, tests host + HIL, mesures latence/jitter, traces, et firmware documenté & reproductible.

Chapitres :

1. Boot & toolchain : startup, linker script (Flash/SRAM), registres, CMake cross, debug (gdb/OpenOCD)*
2. HAL/LL & registres : GPIO, timers de base, clocking/PLL, interruptions (NVIC, priorités)*
3. C++ pour l'embarqué : RAII sans heap, std::array/span, constexpr, enum class, erreurs sans exceptions*
4. Drivers UART/I²C/SPI : polling → IRQ → DMA, protocoles simples, buffers circulaires*
5. TIMERS/ADC/PWM : capture/compare, conversion périodique, synchronisation, timestamping*

6. FreeRTOS/Zephyr I : tâches, piles statiques, queues/semaphores, timers, hooks (stack overflow, idle)*
7. FreeRTOS/Zephyr II : priorités, priority inversion (mutex PI), tickless idle, low-power*
8. Architecture & FSM : couches HAL→drivers→services, machines à états (debounce, protocole), interfaces pimpl*
9. Robustesse & sécurité : watchdog, brown-out, CRC, politiques d'erreurs (expected-like), redémarrage sûr*
10. Logs & traces : SWO/RTT/UART non bloquant, horodatage (std::chrono), mesure latence/jitter*
11. Qualité & MISRA : règles clés MISRA C++, clang-tidy/cppcheck, code reviews, tests host (GoogleTest) & HIL*
12. Industrialisation & Capstone : packaging binaire, scripts flash, feuille de tests, perf & conso, doc utilisateur

À la fin :

Vous saurez développer un firmware C++ moderne temps réel sur ARM Cortex-M : drivers propres (IRQ/DMA), RTOS (tâches, IPC, priorités), mémoire maîtrisée (sans heap), robustesse (watchdog, recovery), traces & mesures (latence/jitter) et conformité MISRA (principes).
Vous livrerez un capstone exploitable (code, binaire, scripts, doc, feuille de tests) — prêt pour l'intégration produit ou une démo client.