Description du module, disponible en: FR

Informatique embarquée avancée

Informations générales

Nombre de crédits ECTS

3

Code du module

TSM_AdvEmbSof

Valable pour l'année académique

2026-27

Dernière modification

2025-10-14

Coordinateur/coordinatrice du module

Serge Ayer (HES-SO, serge.ayer@hefr.ch)

Explications concernant les langues d'enseignement par site :

- L'enseignement est dispensé dans la langue indiquée ci-dessous pour chaque site et chaque exécution du module.
- Les supports de cours sont disponibles dans les langues indiquées ci-dessous pour chaque site et chaque exécution du module. Lorsque plusieurs langues sont utilisées, la proportion de contenu disponible dans chaque langue est précisée (100 % = ensemble des supports de cours).
- Les examens (questions et réponses) sont entièrement rédigés dans la langue indiquée ci-dessous pour le site et l'exécution du module concernés. Ils se déroulent en présentiel.

	Lausanne			Lugano	Zurich		
Leçons		**X** F 100%					
Documentation		**X** F 100%	**X** E 100%				
Examen		**X** F 100%					

Catégorie de module

TSM approfondissement technico-scientifique

Leçons

2 leçons et 1 leçon de pratique par semaine

Compétences préalables

Connaissances préalables, compétences initiales

- Connaissance du langage de programmation C et d'un langage orienté objet.
- Bonne connaissance de l'architecture des ordinateurs et des microprocesseurs.
- Compréhension des concepts de base des systèmes d'exploitation (ordonnancement, processus/thread).
- Connaissances de base en programmation concurrente

Brève description du contenu et des objectifs

Un système embarqué est un système informatique spécialisé avec du matériel embarqué. Il existe une grande variété de systèmes embarqués, mais en général, ces systèmes sont des systèmes capables de détecter les entrées physiques de leur environnement, d'effectuer certains calculs et de communiquer les résultats. Habituellement, les systèmes embarqués sont conçus pour exécuter des tâches répétitives, périodiquement ou spontanément, pour un faible coût, une faible consommation et des performances optimales.

Dans ce module, nous étudions comment les systèmes embarqués basés sur des microcontrôleurs peuvent être développés, en mettant l'accent sur les points suivants :

- Fournir une approche logicielle flexible avec un système multi-tâches permettant une exploitation efficace des capacités matérielles du système.
- Fournir une extensibilité du système.
- Faciliter la détection des erreurs et les capacités de débogage et de tests.
- Fournir une portabilité avec l'utilisation d'un système d'exploitation embarqué et permettre au programmeur d'abstraire les détails matériels de chaque plate-forme.

Objectifs, contenus, méthodes

Objectifs d'apprentissage, compétences à acquérir

Les étudiant-e-s apprendront les caractéristiques les plus importantes d'un système d'exploitation temps réel (Zephyr RTOS) moderne en mettant en œuvre leur propre scénario sur une plateforme de développement IoT offrant un large éventail de capacités de détection, de traitement et de communication. En partant d'une implémentation basique de type super boucle, les étudiants découvriront rapidement les limites de cette réalisation. Ces limites seront étudiées et des solutions améliorées utilisant l'ordonnancement, le threading et la synchronisation seront mises en place par les étudiants pour le développement d'un logiciel robuste, portable et facilement maintenable. En outre, les étudiant-e-s devront également

- Mettre en œuvre des outils et méthodes pour un contrôle de qualité continu.
- Mettre en œuvre des méthodes d'analyse de l'utilisation du CPU et de la mémoire du système.
- Développer des méthodes pour les tests automatisés, y compris des tests unitaires et des tests d'intégration, dans une perspective CI/CD pour les systèmes embarqués.

A la fin du module, les étudiant-e-s seront capables de :

- Développer une application multi-tâches sur un système embarqué basé sur un microcontrôleur, en utilisant un RTOS.
- Utiliser les capacités de débogage et construire l'environnement de test pour une application embarquée.
- Comprendre l'organisation et l'utilisation de la mémoire de leur application.
- Développer un cadre pour la mise à jour des applications embarquées, y compris une application de bootloader.

Contenu des modules avec pondération du contenu des cours

Introduction aux systèmes et logiciels embarqués

- Applications et attributs des systèmes embarqués
- Options pour la construction de systèmes embarqués
- Systèmes embarqués basés sur des microcontrôleurs
- Systèmes embarqués et systèmes d'exploitation (OS)
- Introduction à un système d'exploitation (RTOS) moderne

Tests

- Niveaux de test
- Test unitaire et d'intégration à l'aide du RTOS
- CI/CD pour les systèmes embarqués

Qualité du software

- Importance, outils et méthodes pour un contrôle continu de la qualité.
- Directives de programmation au travers d'exercices.
- Mise en œuvre à l'aide de différents outils.

Ordonnancement pour les systèmes embarqués

- Modèles de programmation des systèmes embarqués
- Aperçu des algorithmes d'ordonnancement
- Ordonnancement cyclique statique, piloté par événements et préemptif, avec comparaisons.

Tâches et concurrence

- Conception de logiciels embarqués en tâches multiples
- RTOS : multitâches, threads, ordonnancement et mécanismes de synchronisation
- Inversion de priorité et protocoles d'accès aux ressources

Mémoire des systèmes embarqués

- Principes de gestion de la mémoire
- Structure de l'image du programme Cortex-M
- Modèle de mémoire du système d'exploitation
- Unité de protection de la mémoire des processeurs Cortex-M

Bootloader

- Déploiement des mises à jour dans les systèmes embarqués
- Principes du bootloader
- Exigences et modèles de mémoire pour le développement d'une application bootloader
- Réalisation d'un bootloader à l'aide du RTOS

Méthodes d'enseignement et d'apprentissage

Ce module utilise des notes de cours et des exercices pratiques qui sont donnés sous la forme de codelabs. Les étudiant-e-s doivent développer leur propre logiciel sur la base d'une spécification, avec l'aide des notes de cours et du matériel des codelabs, et ils doivent rendre le logiciel développé dans le cadre d'un projet en deux phases.

Bibliographie

Les références sont données dans les notes de cours et dans codelabs

Evaluation

Évaluation supplémentaire pendant le semestre

Le module comprend une ou des évaluation(s) supplémentaire(s) pendant le semestre. La note obtenue pour la ou les évaluation(s) supplémentaire(s) est valable à la fois pour l'examen final et pour l'examen de répétition.

Description de l'évaluation supplémentaire pendant le semestre

Au cours du semestre, les étudiants travaillent par groupes sur un projet. Le projet est organisé en trois étapes, qui sont évaluées séparément. La note finale du projet représente 30 % de la note finale du module.

Principe pour les examens

En règle générale, tous les examens réguliers de fin de module se déroulent sous forme écrite. Concernant les examens de répétition, leur format (écrit ou oral) sera communiqué par l'enseignant-e en même temps que le calendrier des examens.

Examen de fin de module régulier et examen écrit de répétition

Type de l'examen

Examen écrit

Durée de l'examen

120 minutes

Aides autorisées

Les aides suivantes sont autorisées:

Aides électroniques autorisées

Pas d'aide numérique

Autres aides autorisées

Deux pages A4 recto-verso de notes personnelles

Exception : En cas d'examen électronique sur Moodle, des modifications des aides autorisées peuvent survenir. Dans ce cas, les aides autorisées seront annoncées par les enseignant-e-s avant l'examen.

Cas spécial: examen de répétition oral

Type de l'examen

Examen oral

Durée de l'examen

30 minutes

Aides autorisées

Les aides suivantes sont autorisées:

Aides électroniques autorisées

Aucune aide électronique autorisée

Autres aides

Deux pages A4 recto-verso de notes personnelles