Syllabus Si8

Conception de Plateformes

Direction des études (2025-2026)


EPITA

Cadrage

Ce document présente les éléments de contenus du semestre 8 de la majeure Conception de Plateformes. La formation est semestrialisée, chaque semestre contient à un nombre d'UEs (Unité d'Enseignement) qui contiennent chacune à leurs tours des ECUEs (Élément Constitutif d'une UE). Une UE est porteuse d'un nombre d'ECTS (European Credit Transfer System) qui reflète le temps de travail étudiant total sur l'UE. La correspondance admise est de 25-30h de travail étudiant par ECTS. Un semestre est valorisé de 30 ECTS, donc un total maximal de travail étudiant de 900h.

Chaque UE a des objectifs d'apprentissages indiqués en tête de descriptif d'UE. Chacune participe à l'acquisition d'indicateurs de développement du référentiel compétences école, également indiqués dans le descriptif de chaque UE. Dans le descriptif d'une ECUE on retrouve :

un résumé des contenus abordés dans l'ECUE
les AAVs (Acquis d'Apprentissages Visés) de l'ECUE
le plan de cours de l'ECUE
les pré-requis de l'ECUE
les modalités de validation et le calcul de la note de l'ECUE
la bibliographie associée au cours.
le coefficient permettant de calculer la moyenne pondérée sur l'UE dont fait partie l'ECUE.

On trouve par ailleurs quelques éléments relatifs au cadrage temporel de l'ECUE et du référent de celle-ci. Parmi les éléments décrivant la ventilation des heures associées à l'ECUE :

CM : nombre d'heures de Cours Magistraux, cela correspond au nombre d'heures d'enseignement magistral à effectuer en présentiel.
TD : les heures de TD, d'activités interactives qui attendent une production de la part des étudiants.
TP : les heures de TP qui attendent une production de la part des étudiants. Un TP peut faire l'objet d'un rendu hors séance de cours qui sera évalué.
Suivi : les heures dédiées au suivi de TP, de projet, où l'enseignant guide les étudiants et répond à leurs éventuelles questions. Il peut aussi s'agir de l'étude d'une correction après un rendu.
Travail perso : le temps de travail hors séance estimé nécessaire pour compléter les objectifs pédagogiques du cours : complétion de TP, réalisation de projet, révisions…
VHT : Volume Horaire Total d'implication des étudiants sur l'ECUE ; cela correspond au temps total estimé d'investissement des étudiants.

Chaque UE a un coordonnateur attitré, chaque ECUE son référent cours. Ils sont à disposition des différents intervenants de l'école pour répondre à leurs questions.

Table des matières

[26_ANS8] Architecture et Noyaux Systèmes

Semestre	S8
Crédits	3
CODE	26_ANS8
Coord.	Geoffrey Le Gourriérec

[HTS] – Harmonisation Technologie Système

Niveau	M1
Semestre	S8
UE	26_ANS8_HTS
Coeff ECUE	1
Référent	Marc ESPIE

Cours magistral	0
TD	24
TP	0
Cours intégré	0
Suivi	0
Soutenance	0
Travail perso	5
Volume horaire total	29

Prérequis

Programmation en C.
Scripting en shell (sh).
Connaissances de base des systèmes UNIX.

Résumé du cours

Le cours d'harmonisation vise à ré-inculquer des bases en programmation système et la culture informatique UNIX nécessaire. Il prend la forme d'un atelier abordant les thématiques fonctionnelles principales que sont l'installation de système, le scripting, la fourniture de services à l'espace utilisateur dans un OS dit "riche", ainsi que le réseau.

La plateforme logicielle du cours est OpenBSD, un OS relativement simple qui permet d'aborder les thématiques du cours sans craindre de lire le code source du noyau. L'essentiel du cours consistera à pratiquer des compétences de base sur OpenBSD.

Comme son nom l'indique, HTS est un cours d'harmonisation : son contenu est adapté en séance selon les compétences démontrées par les élèves. Une fois que les acquis d'une séance ont été démontrés, l'apprenant est libre de passer à la séance suivante.

Contenu et plan du cours

1. Introduction à OpenBSD : choix de conception, compilation, installation
2. Appels systèmes, espaces utilisateur et noyau
3. Scripting pratique avec sh
4. Rappels de programmation réseau

Format des activités

TD avec de brefs rappels théoriques : 24h

TP à compléter chez soi : 5h

Les TD font l'objet de brefs rapports qui permettent à l'enseignant de valider la démarche. Les rendus prendront la forme d'archives *.tar.gz* contenant code source et rapport en prose.

Les TP, dont les sujets seront donnés hors séance, produiront l'évaluation pour le cours. Ils demanderont l'exercice de plusieurs des compétences pratiquées en TD.

Acquis d'apprentissages visés (AAVs)

Les objectifs d'apprentissage du cours sont les suivants :

A l'issue de ce cours, les étudiants sont capables de :

- Installer et configurer OpenBSD sur une machine physique ou virtuelle,
- Visualiser l'interaction générale entre une application et un noyau complexe lors d'un appel système,
- Localiser le code d'un appel système dans le noyau,
- Automatiser des tâches simples de façon portable sur UNIX avec sh,
- Visualiser la machine à états de l'API réseau haut-niveau POSIX.

Évaluation des Acquis d'apprentissages visés (AAVs)

L'évaluation se fera en continu à travers les TP.

Calcul de la note finale

TP 100 %

Références et bibliographie

Les références de cours seront fournies par l'enseignant en séance.

[IOTG] – Internet des Objets : composants et architectures

Niveau	M1
Semestre	S8
UE	26_ ANS8_ IOTG
Coeff ECUE	1
Référent	Eric GAILLARD

Cours magistral	12
TD	0
TP	0
Cours intégré	0
Suivi	0
Soutenance	6
Travail perso	10
Volume horaire total	28

Prérequis

Connaissance de base sur le fonctionnement d'Internet (protocoles IP, TCP et UDP)
Notions en réseaux (trame, commutation, routage, encapsulation)

Résumé du cours

Le cours débute par une présentation générale du monde connecté et du concept d'IoT où les systèmes, embarqués ou non, ont et auront de plus en plus la capacité de détecter, d'apprendre, d'agir et d'interagir avec les personnes et les environnements.

Ensuite le cours poursuit son but de démystification de l'IoT par une présentation des composantes technologiques qui le sous-tendent. En particulier il vise à ouvrir les esprits aux architectures de communications sous-jacentes aux différentes architectures IoT..

Cet exposé des architectures de communication spécifiques aux échanges entre les dispositifs de production/consommation/stockage de données (capteurs, actionneurs, plateformes d'exécution, centre de données, Cloud/Edge Computing, etc.) est fait de manière différentiée au niveau L1/L2 (BLE, 802.15.4, 802.11ah, etc.), L3/L4 (IPv6, 6LoWPAN, 6Lo, TCP, UDP …) et L5/L7 (MQTT, CoAP, …).

Les principales solutions du marché qui intègrent ces différentes technologies sont également présentées en l'état de leur maturité (Matter, Thread, …).

Le cours est complété d'une sensibilisation marquée à la dimension de Cybersécurité relative à l'IoT ; les menaces sur les flux IoT sont exposées ainsi qu'une introduction aux possibles parades.

Le cours se conclut par une mise en perspective des sujets enseignés sur des usages existants, émergents ou futurs illustrée notamment par des travaux de l'IEEE.

Contenu et plan du cours

1. Introduction à l'Internet des objets et au monde connecté
2. Composants et Architecture IoT (Cloud Comp. Edge Comp. , …)
3. Architectures de communication spécifiques à l'IoT
4. Considérations marché
5. La dimension de Cybersécurité dans les architectures IoT (menaces /parades)
6. Cas d'usage / perspectives

Format des activités

Formats des activités pédagogiques :

- Cours magistraux
- Exercice de modélisation d'une architecture IoT (sous forme d'une étude de cas)

Acquis d'apprentissages visés (AAVs)

À l'issue de ce cours, les étudiants sont capables de :

1. Acquérir une vision large et concrète de ce que recouvre la notion d'internet des objets
2. Identifier les composantes des architectures IoT
3. Différencier les technologies de connectivité selon les différents cas d'usage de l'IoT
4. Comprendre la dimension de Cybersécurité appliquée aux architectures IoT
5. Projeter les notions acquises pour architecturer des solutions IoT

Évaluation des Acquis d'apprentissages visés (AAVs)

Le cours IOTG est évalué par un examen sur table de 1h, et par une étude de cas.

L'étude de cas (modélisation d'une architecture IoT) est un travail par équipe se déroulant sur une période de 3 semaines avec un rendu écrit qui privilégiera la synthèse, la structure et la valeur ajoutée technique. Une présentation orale est faite au cours d'une session ad hoc de 6h avec corrections/coaching par l'enseignant.

Les questions/réponses en séances, le support de cours, le corrigé de l'examen de l'année antérieure ainsi qu'une liste de références fournie par l'enseignant (sites, documents) permettent aux étudiants de préparer leur examen et leur exercice de modélisation.

Calcul de la note finale

70% examen + 30% exercice de modélisation

Références et bibliographie

- Support de cours IOTG
- https://www.rfc-editor.org/rfc/rfc9006.html
- https://securityintelligence.com/the-importance-of-ipv6-and-the-internet-of-things/
- http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/internet-of-things-risk-and-value-considerations.aspx
- https://www.rfc-editor.org/rfc/rfc8576
- https://datatracker.ietf.org/doc/html/rfc9019
- https://www.rfc-editor.org/rfc/rfc8138.html
- https://www.rfc-editor.org/rfc/rfc4944
- https://datatracker.ietf.org/doc/html/rfc8025
- https://standards.ieee.org/wp-content/uploads/import/documents/other/iot.pdf
- https://openthread.io/?hl=fr
- https://csa-iot.org/all-solutions/matter/
- https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/iot

[NSE] - Noyau et Systèmes d'exploitation

Niveau	M1
Semestre	S8
UE	26_ANS8_NSE
Coeff ECUE	1
Référent	Ivan BOULE

Cours magistral	12
TD	0
TP	0
Cours intégré	0
Suivi	0
Soutenance	1
Travail perso	10
Volume horaire total	23

Prérequis

Concepts de base d'architecture des ordinateurs : processeur, registres, mémoire vive, bus, périphériques d'entrée/sortie, interruptions matérielles, segmentation de la mémoire.

Concepts de base des systèmes d'exploitation : ordonnanceur, processus, pagination, mémoire virtuelle.

La théorie générale derrière ces concepts, sans étude d'une implémentation choisie, est supposée acquise en ING1. Aucune pratique n'est attendue avant le cours.

Résumé du cours

Le cours NSE a pour but l'étude approfondie des systèmes d'exploitation avec un point particulier sur l'ordonnancement de plusieurs tâches concurrentes.

Contenu et plan du cours

1. Rappels : Architecture des ordinateurs
    A. Architecture de Von Neumann
    B. Conceptions CISC et RISC
    C. Modes d'exécution et niveaux de privilège
    D. Interruptions, exceptions
    E. Appels systèmes
    F. Entrées/Sorties et communication avec périphériques
2. Principes des systèmes d'exploitation
    A. Fonctions essentielles d'un OS
    B. Comment différencier les OS ?
    C. L'OS machine logique par-dessus le matériel
    D. ABI et interface avec l'espace applicatif
    E. Rappels : chaîne de compilation, édition de lien, production de binaires
    F. Concept de processus
    G. Espaces d'adressage mémoire
3. Introduction au parallélisme
    A. Concept de thread
    B. Statut d'un thread et machine à états
    C. Concepts d'ordonnancement et de préemption
    D. Politiques d'ordonnancement de base
    E. Les API threads POSIX et Windows

4. Gestion de la concurrence
    A. Les problèmes d'un environnement multi-threads
    B. Définition d'un modèle d'exécution
    C. Primitives de base : mutex, lock, section critique
    D. Masquage d'interruptions et systèmes multi-coeurs
    E. Primitives POSIX : pipe, sémaphore
    F. Paradigmes producteur/consommateur et lecteur/rédacteur
5. Granularité des techniques de synchronization
    A. Concepts de "coarse-grained" et "fine-grained"
    B. Illustration par une API d'allocation mémoire avec deadlock
    C. Illustration sur projet avec l'OS temps réel Chorus

Format des activités

Formats des activités pédagogiques :

- Cours magistraux chaque séance
- Exercices avec corrigés dispensés hors séance pour vérifier les acquis
- Examen sur table pour évaluer le cours

Acquis d'apprentissages visés (AAVs)

A l'issue de ce cours, les étudiants sont capables de :

1. Identifier les services fondamentaux d'un OS concret et leurs caractéristiques distinctives
2. Sélectionner une politique d'ordonnancement adaptée aux contraintes de latence/performance d'un système applicatif complet
3. Délimiter la couverture d'abstraction matérielle offerte par un OS concret
4. Exprimer une séquence d'exécution de tâches contrainte avec un vocabulaire applicable à tout OS
5. Reconnaître un problème de synchronisation inter-tâches lors de la conception et du développement d'un système applicatif
6. Proposer et implémenter des solutions de synchronisation adaptées aux contraintes de performance, de complexité et de flux d'entrées/sorties d'un système applicatif

Évaluation des Acquis d'apprentissages visés (AAVs)

Le cours NSE est évalué par un examen sur table.

Calcul de la note finale

Examen (100%)

Références et bibliographie

Les références du cours seront accessibles via les supports de cours.

[26_ASET8] Architecture des Systèmes Embarqués Temps réel

Semestre	S8
Crédits	4
CODE	26_ASET8
Coord.	Geoffrey Le Gourriérec

[CSTR] – Conception Système Temps Réel

Niveau	M1
Semestre	S8
UE	26_ASET8_CSTR
Coeff ECUE	2
Référent	Dumitru POTOP-BUTUCARU
Enseignants	Dumitru POTOP-BUTUCARU Fabien SIRON

Cours magistral	16
TD	0
TP	16
Cours intégré	0
Suivi	0
Soutenance	0
Travail perso	15
Volume horaire total	47

Prérequis

Les compétences suivantes seront utiles au bon suivi du cours :

- Programmation en langage C
- Construction d'un système de build avec GNU Make
- Programmation concurrentes et primitives de synchronisation
- Concept de tâche, thread et et d'ordonnancement

Résumé du cours

Le cours de Conception de Systèmes Temps Réel offre une approche holistique de la conception d'un système devant répondre à des contraintes de latence dure, qui doivent garantir ce qu'on appelle de la sûreté de fonctionnement. Cette question de la sûreté est une exigence très forte dans quelques domaines dont par exemple l'aéronautique, le ferroviaire, et le pilotage de machines-outils.

Pour répondre à ces contraintes de latence, le cours propose d'étudier les limites de la programmation séquentielle sur OS généraliste, qui est le mode d'orchestration des tâches le plus répandu, pour se pencher sur les avantages offerts par la programmation dite synchrone en flot de contrôle, à l'aide du langage Heptagon, proche de Lustre.

En complément, le cours propose de formaliser la conception de systèmes exclusivement dépendant d'entrées-sorties, comme souvent le cas dans l'industrie lourde, par l'étude des systèmes dits réactifs. Nous verrons que conception (réactif) et réalisation (programmation synchrone, en flot de contrôle) doivent s'unir pour atteindre les exigences.

Après une introduction pratique à la programmation en Heptagon, l'étudiant(e) est invité(e) à implémenter des politiques d'ordonnancement offrant différents niveaux de garanties pour des systèmes réactifs simples, avant de conclure sur une mise en oeuvre compatible multi-coeurs sur plateforme ARM 64 bits, sans OS.

Contenu et plan du cours

1. Introduction

    a) Système embarqué, système temps réel, système réactif : définitions

b) Ordonnancements en ligne et hors ligne

c) Introduction à la programmation synchrone

d) La programmation synchrone en flot de données : une façon de présenter les algorithmes

e) Introduction au langage Heptagon

f) La chaîne de compilation Heptagon et la transpilation en langage C

g) TP 1 : myfun, sum2, count

2. Focus sur les propriétés temps réel

a) Notion d'exigence non-fonctionnelle

b) Implémenter un système par un mapping temps/espace

c) Choix d'une politique d'ordonnancement, étude de la complexité

d) Concept de temps logique : discret, synchrone, LET

e) La gestion du temps et la conditionnalité d'exécution avec Heptagon

f) Concept d'horloge logique, application en Heptagon

g) TP 2 : rcounter, fs_hs, hs_handler

3. Focus sur la réponse à spécifications

a) Les spécifications fonctionnelles : rappels, application à un système contraint

b) Modélisation d'un contrôleur embarqué temps réel

c) Illustration avec le système GNC

d) Programmation avancée en Heptagon : structures de données, paramètres statiques

e) TP 3 : GNC

4. Ordonnancement temps réel pratique

a) Les propriétés recherchées dans une politique d'ordonnancement temps réel

b) Techniques en ligne et hors ligne

c) Gestion de la préemption

d) Programmer un ordonnanceur en Heptagon

e) TP 4 : ordonnanceur RM, EDF

5. Conception système intégrée

a) Les défis d'un système complexe : coût, garanties, difficulté d'intégration

b) Approche intégrée et approche fédérée

c) Concept de partitionnement spatial et temporel

d) Illustration dans l'avionique avec ARINC-653

e) La programmation synchrone et la vision système

f) Préparation du framework temps réel multi-coeurs ARM 64 bits

6. Mise en oeuvre d'un système intégré partitionné

a) La chaîne de compilation pour cible multi-coeurs ARM 64 bits

b) Interface de programmation via Heptagon et configuration

c) Concept IMA : 4 niveaux logiques dans un système

d) Premiers exemples et étude du code C généré : gestion de la mémoire, de la synchronisation

e) Configuration d'une politique d'ordonnancement

f) TP 5 : implémentation de système temps réel sur plateforme multi-coeurs

Format des activités

Les séances de CSTR sont animées par 2 heures de CM suivies de 2h de TP.

Les travaux pratiques commencés en classe pourront être poursuivis en autonomie chez soi, puis rendus avant la séance suivante.

Acquis d'apprentissages visés (AAVs)

Les objectifs d'apprentissage du cours sont les suivants :

- Identifier et quantifier les contraintes de latence d'un système
- Modéliser un système réactif en flot de données
- Concevoir le déploiement d'un système réactif simple par un schéma de sous-systèmes partitionnés en mémoire et en temps au niveau logique
- Implémenter un système réactif simple par la programmation synchrone en flot de données
- Maîtriser les primitives de synchronisation du langage C (mutex, lock, section critique...) dans un cadre d'exécution concurrente haute performance

Évaluation des Acquis d'apprentissages visés (AAVs)

L'évaluation se fera en évaluation continue à travers les TPs.

Calcul de la note finale

TP 100%

Références et bibliographie

La bibliographie et les références figureront dans les supports de cours.

[VHDL] – VHDL : Introduction au logic design

Niveau	M1
Semestre	S8
UE	26_ASET8_VHDL
Coeff ECUE	2
Référent	Yann DOUZE

Cours magistral	15
TD	0
TP	15
Cours intégré	0
Suivi	0
Soutenance	2
Travail perso	20
Volume horaire total	52

Prérequis

- Base de l'électronique numérique (circuit combinatoire, circuit séquentiel, portes logiques, bascules)
- Connaître les bases du développement en langage assembleur
- Numération binaire, BCD et hexadécimal

Résumé du cours

Le cours de VHDL vise à apprendre l'utilisation d'un langage de description matérielle afin de décrire des circuits numériques séquentiel et combinatoire. Il propose également de comprendre la structure interne des composants programmables et maitriser les outils logiciels qui permettent d'implémenter des circuits numériques sur des FPGA.

Le cours met l'accent sur l'apprentissage du VHDL comportemental qui permet de décrire des circuits avec un plus haut niveau d'abstraction. Il vise également à apprendre les règles élémentaires qui permettent d'écrire du code VHDL synthétisable (VHDL RTL). Les bases de la vérification par banc de test sont également abordées dans ce cours afin de vérifier par simulation chaque description avant implémentation.

Contenu et plan du cours

1. Introduction sur les origines du langage et la structure interne des composants programmables (FPGA)
    A. A quoi sert le VHDL ?
    B. Historique sur les langages HDL
    C. Les enjeux de la synthèse logique
    D. Le flot de conception HDL
    E. Les différentes cibles matérielles (ASIC, FPGA, CPLD, etc…)
    F. L'architecture interne des FPGA
    G. Évolution des méthodes de conception
    H. Qu'est-ce qu'une IP matérielle
2. Structure d'une description VHDL
    A. Les bibliothèques
    B. Déclaration de l'entité

   C. Description de l'architecture
   D. VHDL : un langage concurrent
3. Les opérateurs de base
   A. Affectation simple et opérateurs logiques
   B. Opérateurs relationnels
   C. Affectation sélective
   D. Affectation conditionnelle
   E. Signaux internes
   F. Description structurelle
   G. Instanciation de composant
4. Les instructions séquentielles
   A. Rappel sur les bascules D
   B. Définition et rôle du process
   C. Instructions séquentielles : if, case, while, for
   D. Les règles pour décrire un process purement combinatoire
   E. Affectation incomplète et l'assignement par défaut
   F. Les règles pour décrire un process synchrone
   G. La description RTL (Transfert par registre)
   H. Les mauvais styles de process
   I. Les process non synthétisables
   J. La description des bancs de test
   K. La vérification par assertion
5. Fichiers et librairies
   A. La librairie de travail
   B. La clause de contexte
   C. Ordre de compilation
   D. Style VHDL recommandé
6. Les variables
   A. Les délais et le delta délais
   B. Les variables
7. Les Types
   A. Les types énumérés
   B. Les types définis par défaut
   C. Les valeurs initiales
   D. Les opérateurs arithmétiques
   E. Utilisation de la librairie NUMERIC_STD
   F. Conversion de type
   G. Fonctions de conversion
8. Les Process synchrones
   A. Design synchrone
   B. Les contraintes de temps
   C. La synthèse RTL
   D. Inférence des registres à la synthèse
   E. La resynchronisation des entrées
   F. La détection de front lent
9. Les Sous-Types
   A. Les packages
   B. Les sous-types entiers
   C. Synthèse des sous-types
   D. Sous-types d'un tableau
   E. Description et modélisation des mémoires
   F. Les attributs de types
   G. Les agrégats
10. Les descriptions génériques

   A. Les paramètres génériques
   B. Instanciation d'un composant générique
   C. Instruction Generate
   D. Description d'un additionneur structurel générique
11. Les machines à états (MAE)
   A. Machine de Mealy
   B. Machine de Moore
   C. Représentation par diagramme d'état
   D. Description d'une MAE en VHDL
   E. Machine de Mealy resynchronisée
12. Description d'un cœur de processeur en VHDL (Architecture ARM7TDMI)
   A. Architecture ARM7TDMI
      a. Unité de traitement
      b. Banc de registre
      c. Instructions ARM7TDMI
      d. UAL et registre d'état
      e. Instructions conditionnelles
      f. Unité de commande
   B. Concevoir un processeur étape par étape
      a. Codage binaire des instructions
      b. Composants du chemin de données (datapath)
      c. Assemblage des composants pour réaliser le datapath de chaque instruction
      d. Établir la logique de contrôle

Travaux pratiques : Compteur BCD, Interface VGA et génération de signaux PWM.
Projet : description d'un cœur de processeur 32 bit simplifié (Architecture ARM7TDMI)

Format des activités

Formats des activités pédagogiques :
- Cours magistraux/MIMO : à chaque séance
- Travaux dirigés : à chaque séance
- Travaux pratiques : 2 séances de 4h
- Projet : débuté dès la cinquième séance

Acquis d'apprentissage attendus

A l'issue de ce cours, les étudiants sont capables de :
1. Evaluer la pertinence du langage VHDL dans une application embarquée
2. Écrire du code VHDL comportemental synthétisable
3. Ecrire du code VHDL pour la vérification (banc de test) et la modélisation (non synthétisable)
4. Maitriser une suite logicielle de synthèse logique (Intel Altera)
5. Maitriser un environnement de simulation HDL (Modelsim ou GHDL/GTK Wave)
6. Décrire, simuler et synthétiser en VHDL les composants de base d'un ordinateur (multiplexeur, additionneur, décodeur, registres, compteur, mémoire, machine à états, etc..)
7. Décrire, simuler et synthétiser en VHDL un cœur de processeur (ARM7TDMI)
8. Programmer une interface pour afficher une image sur un écran via VGA

Évaluation des acquis d'apprentissage attendus

Le cours VHDL est évalué par deux travaux pratiques, un examen QCM sur Moodle et un projet.

Le premier TP consiste à décrire un compteur BCD et un générateur PWM qui permet de mettre en œuvre plusieurs concepts vus pendant le cours et prendre en main une suite logicielle de synthèse logique (Intel Altera).

Le deuxième TP consiste à comprendre le protocole VGA et décrire une interface qui permet dans un premier temps de générer une mire de couleur horizontale et verticale, dans un second temps d'afficher une image sur l'écran VGA.

Les travaux pratiques sont évalués pendant la séance en validant les différentes étapes décrites dans le sujet.

Le projet propose de décrire le cœur d'un processeur ARM7TDMI en plusieurs étapes :

1. Décrire et simuler les composants de bases qui constituent le processeur (Multiplexeur, Unité Arithmétique et Logique, extension de signe, registre, banc de registre, mémoire d'instructions, mémoire de données, etc…)
2. Décrire et simuler l'unité de traitement qui rassemble les composants de bases précédemment décrits
3. Décrire et simuler l'unité de gestion des instructions : incrémentation du compteur de programme et gestion les sauts et les offsets
4. Décrire l'unité de contrôle avec le décodeur des instructions
5. Assembler les différentes et simuler l'exécution d'un petit code assembleur.
6. Synthèse du processeur et implémentation sur carte FPGA.
7. Description et simulation d'un périphérique UART

Livrables du projet :

- Code source, banc de test et script de simulation. Résultats de synthèse (fichier .sof)
- Rapport de projet qui répond aux différentes questions posées dans le sujet et précise par des captures d'écrans les simulations réalisées (cf ce quelle ont permis de vérifier).
- README décrivant : membres du groupe + paliers accomplis

Un examen QCM sur Moodle qui permet de vérifier que les connaissances théoriques de bases sont acquises. Il aura lieu lors d'une séance de cours, pas lors de la semaine de partiels.

Formule de calcul de la note finale

- TP 20%
- Projet 50%
- QCM 30%

Références et bibliographie

Les références du cours sont accessibles sur la page Moodle du cours.

[26_CIEL8] Culture de l'Ingénieur et Electronique

Semestre	S8
Crédits	5
CODE	26_CIEL8
Coord.	Geoffrey Le Gourriérec

[ELEC] – Introduction à l'électronique

Niveau	M1
Semestre	S8
UE	26_CIEL8_ELEC
Coeff ECUE	2
Référent	Corentin VIGOURT
Assistants TP	Dany Jorge Pierre-Olivier Koenig

Cours magistral	13
TD	3
TP	11
Cours intégré	0
Suivi	10
Soutenance	3
Travail perso	20
Volume horaire total	60

Prérequis

Bases théoriques de l'électronique

Résumé du cours

Ce cours a pour but d'apprendre les bases de la manipulation de circuits électroniques. Il aborde notamment l'utilisation de matériel de laboratoire, la réalisation de circuit de test ou encore la conception de prototype de cartes électroniques.

Contenu et plan du cours

1. Introduction à la manipulation de circuit
    a. Rappels sur l'électroniques
    b. Comment validons-nous un design électronique
    c. Premiers circuits
    d. Utilisation du matériel de laboratoire
2. Circuit et composants usuels (1)
    a. Qu'est-ce qu'un schématique
    b. Qu'est-ce qu'un composant électronique
    c. Résistances
    d. Diodes
    e. Condensateurs
    f. Inductances
    g. Lois de Kirchoff (rappels)
3. Composants usuels (2)
    a. Moteurs
    b. Autres types d'actionneurs

   c. Interrupteurs
   d. Transistors
   e. Montages intéressants
4. ICs
   a. ICs
   b. Etude de cas : NE555
   c. Logique booléenne
5. Microcontrôleur
   a. Transmission de données
   b. ICs avancés
   c. Microcontrôleur
   d. Familles de microcontrôleur
   e. Etude de cas : ATTiny85
6. Transformation et filtrage
   a. Alternatif vs continu
   b. Traitement du signal
   c. Filtres
7. Conception de PCB
   a. Qu'est-ce qu'un PCB
   b. Choix des composants
   c. Ecriture d'un schématique
   d. Design du PCB
8. Atelier pratique de soudure

Format des activités

Formats des activités pédagogiques :

- Cours magistraux/MIMO : à chaque séance
- Travaux pratiques : à chaque séance (sauf la dernière)
- Projet : après la dernière séance avec soutenance à la fin
- Atelier soudure : au moment requis dans le projet

Acquis d'apprentissages visés (AAVs)

A l'issue de ce cours, les étudiants sont capables de :

1. Lire et comprendre le schématique d'un circuit électronique
2. Faire des circuits de tests
3. Tester et valider des circuits simples
4. Concevoir des prototypes de cartes
5. Interagir et communiquer avec des équipes d'électroniciens

Évaluation des Acquis d'apprentissages visés (AAVs)

Le cours est évalué à la fois par des TPs effectués durant les séances de cours (sauf la dernière) et par un projet qui débute à la fin de la dernière séance.

Travaux pratiques

Commenté [gl1]: J'ai redécoupé les paragraphes entre TPs et projet, sans toucher au contenu.

Commenté [cv2R1]: Ok

Chaque séance se divise en deux parties, une partie cours et une partie TP pour pratiquer les points abordés durant la partie cours. Les étudiants devront réaliser des circuits donnés et faire valider chaque partie du TP auprès de l'enseignant et de ses assistants. Il sera possible de valider les manipulations avant le cours suivant.

Un TP utilisera les connaissances données pendant la séance de cours associée.

Un sujet sera fourni aux étudiants pour chaque TP. Ils utiliseront le kit de matériel fourni pour le cours ainsi que les équipements de laboratoires disponibles.

Projet

Le projet consiste en la conception et la réalisation d'une carte électronique simple. Les étudiants pourront au choix :

1. Faire le circuit type proposé par l'enseignant (+ simple)
2. Proposer un circuit utilisant les composants disponibles, dans le matériel du cours ou au laboratoire (+ complexe),

Le projet utilise les connaissances acquises durant tout le module mais exploite essentiellement le contenu de la dernière séance, sur la conception de carte.

Les supports de cours, les sujets de TPs, le kit de matériel fourni, les équipements de laboratoire disponibles ainsi que des ressources techniques permettent de mener à bien le projet. Il est aussi possible de demander de l'aide si blocage.

Le PCB sera commandé par l'enseignant puis remis aux étudiants pour souder leur carte puis venir la faire valider le jour de la soutenance.

Livrables :

- Fichiers de conception de la carte électronique (schématique, design de carte)
- Fichiers de fabrication de la carte (fichier gerber, BOM)

Calcul de la note finale

- TP 60%
- Projet 40%

Références et bibliographie

Les références du cours sont accessibles sur la page Moodle du cours.

[REDAC] – Rédaction technique

Niveau	M1
Semestre	S8
UE	26_CIEL8_REDAC
Coeff ECUE	1.5
Référent	Geoffrey LE GOURRIEREC

Cours magistral	0
TD	6
TP	0
Cours intégré	0
Suivi	0
Soutenance	0
Travail perso	20
Volume horaire total	26

Prérequis

Aucun prérequis.

Résumé du cours

L'ingénieur est amené à rédiger quantités de documents plus ou moins techniques, assumant ainsi un rôle de communiquant indispensable pour dialoguer avec d'autres ingénieurs, voir avec d'autres corps de métier *(manager,* testeur, électronicien, DevOps...)

Parfois, il est aussi amené à écrire dans une forme plus libre: article de blog, comparatif, ou étude pour la presse technique sont autant d'exemples de prose nécessitant de maîtriser quelques bases de l'écriture.

Le cours REDAC propose d'étudier des exemples d'écriture publique, notamment des articles, et d'apprendre à se poser les bonnes questions avant de prendre la plume, que ce soit pour soi ou pour son travail en entreprise.

Le cours REDAC n'étudie pas les *mails* ou autres formes d'écriture rapportées à un échange dans le temps.

Contenu et plan du cours

A. Pourquoi écrire sur la technique ?
    a. Ce dont le cours ne parlera pas
    b. Ecrire pour soi-même
    c. Ecrire pour le travail
    d. Ecrire pour la presse
B. Les différentes formes d'écriture
    a. Tutoriels et introductions
    b. "Tech dives"
    c. Ecrire sur l'actualité
    d. Comparatifs et guides techniques

e. Spécifications

C. Techniques d'écriture

a. Quel message passer ?

b. Cibler son public

c. Ton et style

d. L'acrostiche WISDOM

Format des activités

- Supports de cours à consulter chez soi, avant chaque séance.
- En séance, TD sur une étape précise de la rédaction de votre article, et coaching par l'enseignant.
- Poursuite du travail entamé chez soi.

Acquis d'apprentissages visés (AAVs)

A l'issue de ce cours, les étudiants sont capables de :

- Identifier un format et un style d'écriture adaptés pour communiquer à un public choisi
- Doser le niveau de détail technique dans une communication écrite
- Rédiger un article technique court (< 3000 mots) style article de blog

Évaluation des Acquis d'apprentissages visés (AAVs)

Le cours REDAC est évalué par l'écriture d'un article pour le blog GISTRE (https://blog.gistre.epita.fr/). Le choix de publier ou non est laissé à la discrétion de l'étudiant(e).

Avant toute chose, il est recommandé de lire 2 ou 3 articles piochés sur le blog, pour se faire une idée du style et des thématiques abordées. Voyez par là-même les sujets déjà traités.

Chacune des 3 séances lancera une des étapes de rédaction ci-dessous.

Trouver un sujet

Un sujet doit d'abord être soumis par *mail* à l'enseignant et accepté par lui. L'enseignant jouant le rôle de poser une ligne éditoriale, le sujet doit en effet être estimé "compatible" avec les thématiques de la majeure GISTRE: programmation système, OS, toolchain, hardware. Embarqué comme débarqué sont bienvenus. Aucun sujet n'est prédéfini : discutez-en avec l'enseignant !

Une fois le sujet accepté, créez un ticket sur Gitlab avec comme titre celui de votre futur article.

Rédiger un plan

Un plan détaillé montre à l'enseignant que vous avez réfléchi sur les messages clefs de votre article, et sur la structure adaptée pour faire passer ces messages.

Un plan détaillé doit être soumis dans la description de votre ticket Gitlab. Notifiez l'enseignant (@pseudoenseignant) dans un message du fil de discussion du ticket, pour qu'il puisse le revoir.

Rédiger l'article

Une fois le plan accepté, l'article est rédigé et soumis via une *merge request* sur le dépôt Gitlab du blog. Le guide de contribution complet est accessible ici : https://gitlab.cri.epita.fr/ing/majeures/gistre/blog/-/blob/main/CONTRIBUTING.md

2 personnes vous reliront : l'enseignant, et un(e) camarade de classe. Leurs remarques utiliseront le système de revue offert par Gitlab. La personne faisant une demande de changement est aussi celle qui décidera de clore (*solve*) le fil de discussion (*thread*) afférent ou non. La pertinence de chaque demande est naturellement sujette à discussion. L'exercice est difficile : soyez constructifs et bienveillants.

Une fois un article prêt, l'auteur pourra passer la *merge request* au statut *ready* (!= *draft*) pour indiquer que la publication est souhaitée. Dans le cas contraire, laisser en *draft*.

Un article ne sera publié que si :

- Toutes les remarques ont été traitées,
- L'auteur a passé la *merge request* en statut *ready,*
- La note obtenue suite à l'évaluation est supérieure ou égale à 12/20.

/!\ Il faudra en moyenne 2 allers-retours avec l'enseignant pour que l'article atteigne la qualité rédactionnelle demandée. Comptez 1 semaine pour obtenir un retour. /!\

Calcul de la note finale

Article de blog 100%

Références et bibliographie

La bibliographie et les références figureront dans la page Moodle du cours.

[TELE] – Extraction d'information sur des images de télédétection

Niveau	M1
Semestre	S8
UE	26_CIEL8_TELE
Coeff ECUE	1.5
Référents	Laurent BEAUDOIN Loïca AVANTHEY

Cours magistral	6
TD	21
TP	0
Cours intégré	0
Suivi	0
Soutenance	4
Travail perso	10
Volume horaire total	41

Prérequis

Aucun.

Résumé et contexte

Le cours de Télédétection passe en revue les multiples capteurs et vecteurs des différentes échelles de télédétection : spatiale, aérienne et rapprochée. On couvre la partie historique pour mieux comprendre le panorama actuel, les concepts importants liés à leurs utilisations et les champs d'applications. On apprend à exploiter les informations contenues dans des images sur des applications concrètes (tsunamis, mer d'Aral, pyramides, cyclone, etc.). On découvre ce qu'est un système d'acquisition d'un point de vue systémique (éléments, rôles et connections) à travers un *serious game*. Et enfin on fait un point sur la législation des drones volants en France ainsi que les bases de pilotage.

Contenu et plan du cours

1. History: on passe en revue dans l'histoire les différentes avancées des domaines de la télédétection pour arriver à ce que l'on a aujourd'hui.
2. Serious Game : pour comprendre les éléments système d'une plateforme d'acquisition.
3. Spatial Remote Sensing : on aborde les principes spécifiques au domaine spatial et on apprend à extraire les informations d'images satellites.
4. Close Range Remote sensing : on aborde les principes spécifiques de la télédétection rapprochée par drone.
5. Beyond horizons : on voit les autres échelles de la télédétection (vecteurs, capteurs et applications pour la télédétection exoplanétaire et sous-marine).
6. Législation et pilotage de drone.

Format des activités

Formats des activités pédagogiques:

1. Cours magistraux/MIMO
2. Travaux dirigés
3. Ateliers
4. Projet

Acquis d'apprentissages visés (AAVs)

Une fois ce cours suivi, l'étudiant(e) sera capable de:

- Comprendre les différentes échelles de télédétection et leurs applications
- Connaître les différents capteurs et vecteurs ainsi que leurs avantages et inconvénients
- Maîtriser les concepts fondamentaux de la télédétection : acquisition et interprétation des images (trajectographie, GSD, mesures physiques, etc.)
- Extraire des informations (mesures, etc.) d'images satellites
- Conceptualiser un système d'acquisition (éléments systèmes, adaptation à une application, etc.)
- Connaître les bases de la législation française en ce qui concerne les drones volants et savoir où trouver les informations spécifiques
- Maîtriser les bases du pilotage de drone
- Réaliser et programmer un vecteur roulant pour suivre une trajectoire à l'aveugle

Évaluation des AAVs

Le cours se termine par un projet final qui demande aux élèves de monter un petit robot roulant (voiture) et de le programmer de manière à ce qu'il puisse suivre à l'aveugle une trajectoire connue à l'avance. Le jour de l'évaluation, une soutenance teste le projet en conditions réelles à l'école.

Le matériel (robot roulant, batteries) est fourni au préalable par les enseignants sous forme de *pack* individuel et doit être rendu à l'issue de la soutenance.

La soutenance dure 4h, le temps de faire passer tous les groupes.

Livrables: code source, rapport de projet, soutenance.

Calcul de note finale

Projet 100%

Références et bibliographie

- Remote Sensing and Image Interpretation, Thomas Lillesand, Wiley
- Remote Sensing: Principles, Interpretation, and Applications, Floyd F. Sabins Jr., James M. Ellis, Waveland Press Inc.

[26_PROGS8] Programmation Système

Semestre	S8
Crédits	8
CODE	26_PROGS8
Coord.	Geoffrey Le Gourriérec

[ARM] – Technologies ARM

Niveau	M1
Semestre	S8
UE	26_PROGS8_ARM
Coeff ECUE	2
Référent	Geoffrey LE GOURRIEREC

Cours magistral	9
TD	9
TP	0
Cours intégré	0
Suivi	2
Soutenance	0
Travail perso	30
Volume horaire total	50

Prérequis

- Programmation en langage C, niveau confirmé
- Electronique pratique et signaux numériques, niveau débutant

Résumé du cours

Le cours ARM est une introduction à la programmation sur microcontrôleur (MCU), avec comme technologie centrale pour l'illustrer l'architecture matérielle ARM sur cœur Cortex-M3. La question à laquelle le cours souhaite donner réponse : j'ai un MCU dans les mains, comment puis-je m'en servir ?

Le cours ARM met ainsi l'accent sur la pratique.

On utilisera comme matériel des kits de développement pour STM32F429, des capteurs, et des câbles Dupont classiques pour monter des prototypes. Quelques opérations de soudures ponctuelles peuvent être nécessaires.

On utilisera comme logiciel l'environnement de développement STM32 Cube IDE, qui comprend notamment le configurateur / générateur de code STM32 CubeMX, une toolchain, ainsi que la HAL/LL adaptés pour le MCU visé.

Contenu et plan du cours

1. Introduction
    a. Goals of the course
    b. The limits of the PC architecture
    c. Characteristics of MCUs
    d. Intoducting the STM32F429
    e. Setting up our development environment
2. The basic toolset
    a. What's in STM32 Cube IDE ?
    b. Blinking LED
    c. Hardware corner: GPIO
    d. The code behing blinking LED
    e. Configuring pins with STM32 CubeMX
    f. Software corner: C weak symbol

3. Debugging
    a. Debug chains for embedded
    b. Debugging blinking LED
    c. Hardware corner : test point
    d. Software corner : SWD
    e. The STM32F429 devkit debug chain
    f. Flashing firmware
4. Clocks
    a. How can a computer control time ?
    b. Hardware corner: clock
    c. Configuring clocks with STM32 CubeMX
    d. Hardware corner: timer
    e. Blinking LED without main()
5. Communicating with a PC
    a. Meet the dinosaur of cable communication
    b. Hardware corner: RS232
    c. Software corner: UART
    d. Using UART for debugging
    e. Driver instanciation in STM32 CubeMX
    f. Launching the ARM project
6. Programming techniques for embedded
    a. Specific goals of embedded software
    b. MCU firmware design primer
    c. Memory saving
    d. Error handling
    e. Code portability
    f. Testing
7. Reading a sensor
    a. UART is not for everyone
    b. Hardware corner: I2C
    c. Looking at I2C with a logic analyzer
    d. Programming the MPU6050
8. Writing portable code
    a. Where does portability come from ?
    b. Designing for portability
    c. Software corner: Hardware Abstraction Layer
    d. The STM32 HAL
    e. The STM32 LL
9. Conclusion
    a. Skills we obtained
    b. Skills we don't have
    c. Employment perspectives in embedded
    d. Project Q&A

Format des activités

- Chaque séance est un mélange de cours magistral et de TD.

- Un projet est démarré au milieu du cours ARM, qui sera travaillé sur le temps personnel.
- Le projet fait l'objet d'une présentation de lancement, et d'une correction, lors de deux petites séances distinctes en classe.

Acquis d'apprentissages visés (AAVs)

A l'issue de ce cours, les étudiants seront capables de :

- Développer un logiciel de type *bare* metal sur microcontrôleur,
- Identifier les leviers de conception logicielle envisageables pour un système embarqué à base de microcontrôleur,
- Identifier la cause racine d'un dysfonctionnement dans un logiciel de microcontrôleur,
- Modéliser un système à base de microcontrôleur mono-cœur et équipé de quelques périphériques,
- Piloter un périphérique matériel de type capteur simple,
- Valider manuellement un logiciel pour microcontrôleur.

Évaluation des Acquis d'apprentissages visés (AAVs)

Projet de programmation système sur microcontrôleur STM32F429 avec capteur MPU6050.

Calcul de la note finale

Projet 100 %

Références et bibliographie

Les références seront fournies sur la page Moodle du cours ARM.

[BOOT] – Séquences de démarrage

Niveau	M1
Semestre	S8
UE	26_PROGS8_BOOT
Coeff ECUE	1
Référent	???

Cours magistral	6
TD	0
TP	6
Cours intégré	0
Suivi	0
Soutenance	0
Travail perso	15
Volume horaire total	27

Prérequis

TODO

Résumé du cours

TODO

Contenu et plan du cours

TODO

Format des activités

TODO

Acquis d'apprentissages visés (AAVs)

TODO

Évaluation des Acquis d'apprentissages visés (AAVs)

TODO

Calcul de la note finale

TODO

Références et bibliographie

TODO

[IOT1] – Internet des Objets et Bluetooth Low Energy

Niveau	M1
Semestre	S8
UE	26_PROGS8_IOT1
Coeff ECUE	1
Référent	Yann DOUZE
Assistant TP	Geoffrey LE GOURRIEREC

Cours magistral	10
TD	0
TP	13.5
Cours intégré	0
Suivi	0
Soutenance	0.5
Travail perso	5
Volume horaire total	29

Prérequis

- Programmation en langage C
- Bases en réseau
- Cours ARM

Résumé du cours

Étudier l'ensemble des techniques qui permettent d'appréhender le domaine de l'internet des objets (Internet Of Things : IoT). Mettre en œuvre par la pratique toutes les couches matérielles et logicielles qui existent entre un capteur physique et la représentation de ses données sur une application web.

Contenu et plan du cours

1. Introduction IoT
    A. Naissance et définition du nom IoT
    B. Objectifs de l'IoT
    C. Architecture des solutions IoT
    D. Les marchés de l'IoT
    E. Les technos autour de l'IoT : Big Data, IA, Energy Harvesting et les plateformes Cloud
    F. Exemple IoT : la voiture connectée
    G. Les défis de l'IoT : interopérabilité, cybersécurité, connectivité et acceptation sociétale
    H. Les acteurs français de l'IoT
2. Les technologies radio pour l'IoT
    A. Les technologies de communication radio

   B. Les bandes de fréquences
   C. Les caractéristiques des technos radio
   D. Les réseaux LPWAN (Sigfox, LoRa et NB-IoT)
   E. Quelle technologie choisir en fonction du cas d'usage

3. Le WiFi
   A. Architecture WLAN
   B. Le modèle OSI et le WiFi
   C. Les versions du WiFi
   D. Les canaux WiFi
   E. Adresse IP et adresse MAC
   F. Les modules WiFi : ESP8266 et ESP32
   G. Quelques exemples de codes utilisant le WiFi
4. Le Business Model des solutions IoT
   A. Les 3 couches de l'IoT
   B. Quelques cas d'usages : Fitbit et Netatmo
   C. Les coûts des solutions IoT
   D. Les modèles de revenus des solutions IoT
   E. Quelques exemples de solutions IoT pour le futur
5. Les protocoles REST, MQTT et CoAP
   A. Format des données (JSON, XML)
   B. Le protocole HTTP REST
   C. Le protocole MQTT
   D. Les protocole CoAP
6. Le réseau LoRaWAN en détail
   A. La modulation LoRa
   B. Les trames LoRa et LoRaWAN
   C. Le protocole LoRaWAN

D. Le serveur d'application et le serveur réseau

E. Les 3 classes LoRaWAN

F. Activation d'un module LoRaWAN (ABP ou OTAA)

G. Les réseaux LoRaWAN en France

7. Les protocoles LPWAN Sigfox et LoRaWAN

A. Architectures LPWAN

B. Les bandes de fréquence et le rapport cyclique (duty cycle)

C. Historique de Sigfox et LoRa

D. Les différences de modulation

E. Les spécificités du réseau Sigfox

F. Les spécificités des réseaux LoRaWAN

G. Les réseaux LTE-M et NB-IoT

Format des activités

Formats des activités pédagogiques :

- Cours magistraux au début de chaque séance avec une mise en pratique des concepts théoriques
- Un TP de prise en main de la carte ESP32
- Un projet débuté en séance, et à compléter chez soi

La pratique consiste plus précisément en :

- Travaux pratiques : Prise en main d'un module WiFi ESP32
- Mini-Projet 1: Mise en place de communications bidirectionnelles avec les protocoles TCP-IP REST, MQTT et CoAP
- Mini-Projet 2: Mise en place de communications bidirectionnelles avec la technologie LoRaWAN

Acquis d'apprentissages visés (AAVs)

A l'issue de ce cours, les étudiants sont capables de :

1. Comprendre l'architecture d'une solution IoT
2. Connaître les différentes technologie IoT et leurs particularités
3. Comprendre le business model des solutions IoT
4. Savoir choisir la bonne technologie de communication en fonction du cas d'usage de la solution IoT

5. Être capable d'envoyer et recevoir des données depuis une carte microcontrôleur Wifi vers et depuis un Cloud en utilisant les protocoles REST et MQTT.
6. Savoir mettre en place un broker MQTT
7. Connaître les caractéristiques et les modulations des réseaux LPWAN Sigfox et LoRaWAN.
8. Connaître et comprendre les différences entre les réseaux LPWAN Sigfox et LoRaWAN
9. Être capable d'envoyer et recevoir des données vers un Cloud en utilisant la technologie LoRaWAN

Évaluation des AAVs

Le cours IoT est évalué par un projet, et un examen QCM sur Moodle qui permet de vérifier que les connaissances théoriques de bases sont acquises.

Livrables du projet

- Code source, schéma d'architecture des solutions
- Copie d'écran des différents tableaux de bords obtenus
- Donner les informations qui permettent de refaire les solutions sous la forme d'un petit tutoriel
- README décrivant : membres du groupe + paliers accomplis

Calcul de la note finale

Projet : 70 %
Examen QCM sur Moodle : 30 %

Références et bibliographie

Les références seront données dans la page Moodle du cours IOT1.

[PFE_GISTRE] – Projet de Fin d'Etudes : extension de majeure

Niveau	M1
Semestre	S8
UE	26_PROGS8_PFE
Coeff ECUE	4
Référent	Geoffrey LE GOURRIEREC
Enseignants / Tuteurs	Geoffrey LE GOURRIEREC Corentin VIGOURT Loïca AVANTHEY Laurent BEAUDOIN

Cours magistral	2
TD	0
TP	0
Cours intégré	26
Suivi	18 (par groupe)
Soutenance	6
Travail perso	60
Volume horaire total	112

Prérequis

Aucun, chaque projet de fin d'études apportant ses propres défis.

Toutefois, la pratique régulière sur les projets courts réalisés en ING1 doit permettre un lancement rapide de chaque PFE, qui nécessite par définition un ensemble de compétences pratiques.

Résumé du cours

Le Projet de Fin d'Etudes est un projet d'un an qu'on trouve dans toutes les majeures de l'EPITA. En majeure Conception de Plateformes, ce projet est particulièrement important et son volume horaire est augmenté en conséquence : vous lisez ici l'extension de cours propre à la majeure.

Un PFE vise à réaliser, en un an, un projet de qualité professionnelle. Le sujet peut être interne à l'EPITA, en partenariat avec une des divisions du LRE, ou avec une entreprise; dans ce dernier cas on le nomme PFEE (pour Entreprise). Le projet doit venir résoudre un problème concret, pour lequel il n'existe pas de solution toute faite.

En majeure Conception de Plateformes, un rythme proche du monde de l'entreprise est adopté : rendus réguliers, vérification de l'avancement, démos… Pour aider les étudiants, 2 ressources : les cours pratiques, et l'aide personnalisée du tuteur.

Tout le long du projet, un(e) tuteur(trice) est en effet désigné(e) pour accompagner les étudiants. Cette personne agit comme une ressource et pas comme une contributrice, ni même une *tech lead*. Le PFE est une démonstration d'autonomie de la part des étudiants.

L'organisation de ce temps de travail est laissée au soin de l'équipe d'étudiants. Pour la réussite du projet, maintenir un rythme hebdomadaire est clef.

Contenu et plan du cours

Les cours sont listés dans l'ordre chronologique.

- Réunion de lancement
  - Présentation des sujets proposés
  - Présentation des travaux attendus
  - Lancement des votes pour constituer les équipes
- Comment rédiger un cahier des charges ?
  - A quoi sert votre projet ?
  - Le concept d'exigence
  - A quoi ne sert pas votre projet ?
  - Rédiger correctement ses exigences
  - Exemple : AJA
- Comment faire un état de l'art ?
  - Se renseigner sur un domaine
  - Evaluer les technologies existantes
- Comment s'organiser ?
  - Eviter les réunions inutiles
  - Partager une compréhension commune
  - Prendre des décisions raisonnées
  - Exemple : votre projet
- Comment rédiger des spécifications ?
  - Cycle de vie d'un produit logiciel
  - Pourquoi écrire des spécs ?
  - Du besoin au code, la traçabilité en action
  - Exemple : robot Evolutek
- Comment bien utiliser Gitlab ?
  - Tour des services basés sur git
  - Gitlab en gros
  - Tickets et merge requests
  - Mono vs multi repo
  - Exemple : votre projet
- Comment définir une architecture HW ?
  - Pourquoi définir l'archi HW ?
  - Quand définir l'archi HW ?
  - Alimentation
  - Unités de calcul
  - Bus de communication
  - Capteurs et actionneurs
  - Modularité
  - Exemple : robot Evolutek
- Comment construire une timeline ?
- Comment définir une architecture SW ?
- Comment pitcher son projet ?
  - Pourquoi apprendre à présenter son travail ?
  - Les défis de la présentation orale

- Identifier son public
- Contrôler la voix
- Maîtriser la gestuelle
- Faire une démo cool
- Adapter la technicité du discours
- Le fil rouge
- L'argumentation solide
- Exemple : votre projet

Format des activités

- Cours magistraux immédiatemment suivis d'une application au projet de chaque groupe.
- Suivi hebdomadaire de 30min avec tuteur(trice).
- Livrables jalonnant le semestre : cahier des charges, spécifications techniques, revue de conception. Après chaque livrable, debrief avec tuteur(trice).
- Soutenance de 30min en fin de semestre.

Acquis d'apprentissages visés (AAVs)

A l'issue de ce cours, les étudiants sont capables de :

- Proposer et ajuster un cahier des charges
- Rechercher et analyser l'existant pour définir un état de l'art; en conséquence, proposer des choix technologiques cohérents avec les contraintes (temps, difficultés, priorités) du projet
- Concevoir un logiciel à partir du cahier des charges: spécifications fonctionnelles, et spécifications techniques
- Documenter un projet pour atteindre une qualité suffisante à la poursuite du projet par d'autres ingénieurs
- Définir un plan de test en accord avec les spécifications à atteindre
- Prototyper (Proof of Concept) un projet pour démontrer la faisabilité de fonctionnalités
- Maîtriser la dette technique d'un projet et prioriser des axes d'amélioration

Évaluation des Acquis d'apprentissages visés (AAVs)

- Livrable : cahier des charges (fin mars)
- Livrable : spécifications techniques (fin avril)
- Livrable : revue de conception (début juin)
- Livrable : rapport (pour soutenance)
- Livrable : slides (pour soutenance)
- Livrable : vidéo de démo (pour soutenance)
- Soutenance de 30min face au jury, composé de tous les enseignants du cours

Calcul de la note finale

La note est divisée en 7 critères de part égale. On évalue l'équipe et non pas les individus :

- Suivi : régularité du suivi, capacités d'organisation, clarté des points d'avancement, facilité de communication

- Rapport : clarté de l'historique du projet, du problème cible, de l'état de l'art et des choix technologiques défendus ; précision des problèmes rencontrés et des solutions retenues ou envisagées ; statut global vs ambitions de départ ; perspectives futures
- Slides : clarté des messages clefs, concision, esthétique
- Oral : clarté de la voix, des propos ; démo en *live* si pertinent ; conviction
- Cahier des charges : introduction claire aux objectifs du projet ; transparence concernant les questions encore floues qui ont besoin d'être clarifiées ; concernant les exigences fonctionnelles et non-fonctionnelles : complétude, non-ambiguïté, classification
- Spécifications techniques : traçabilité entre exigences techniques et exigences fonctionnelles ; schémas d'architecture sous différents points de vue (temporel, structurel…) ; concernant les exigences techniques : précision, non-ambiguïté
- Preuve de concept : est-ce que le code marche 😊

Références et bibliographie

Les supports de chaque cours fourniront les références spécifiques utilisées.


EPITA
ÉCOLE D'INGENIEURS EN INFORMATIQUE