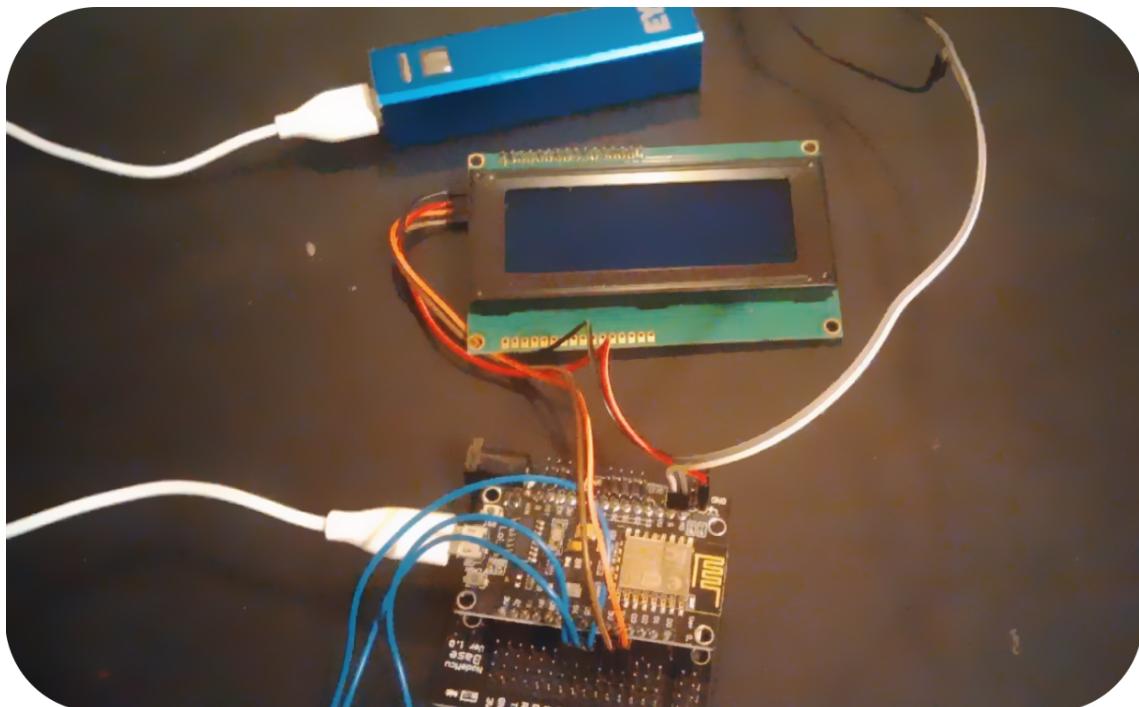


# Dossier industriel

Comparaison énergétique de langages de programmation et création d'un module connecté pour instrument de mesure



Jonathan LÉVY

Candidat n°9118196442

En partenariat avec Orange Labs - Lannion



# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Présentation de l'entreprise . . . . .	3
1.2	Contexte de l'étude . . . . .	3
<b>2</b>	<b>Projet scientifique et technique, système associé</b>	<b>4</b>
2.1	Problématique . . . . .	4
2.2	Pistes de recherches . . . . .	4
2.3	Système utilisé . . . . .	4
<b>3</b>	<b>Développement logiciel</b>	<b>7</b>
3.1	Comparer différents langages de programmation . . . . .	7
3.1.1	Choix des langages . . . . .	7
3.1.2	Choix de la tâche à effectuer . . . . .	7
3.1.3	Calcul de l'énergie consommée . . . . .	8
3.1.4	Résultats et observations . . . . .	8
3.2	Relever la consommation électrique : création d'un instrument connecté . . . . .	10
3.2.1	Besoins associés à la mesure . . . . .	10
3.2.2	Présentation du système de mesure avec l'ESP8266 . . . . .	10
3.3	Intégration des fonctionnalités . . . . .	12
3.3.1	Implémentation du serveur Web et envoi de fichiers . . . . .	13
3.3.2	Script de mesure de la tension de la sonde . . . . .	13
3.3.3	Interface du serveur web . . . . .	15
3.3.4	Pistes d'amélioration envisagées . . . . .	16
<b>4</b>	<b>Séquences pédagogiques</b>	<b>19</b>
4.1	Séquence pédagogique en IUT GEII . . . . .	19
4.1.1	Présentation de la filière . . . . .	19
4.1.2	Le Programme Pédagogique National . . . . .	20
4.1.3	Module <i>Informatique Embarquée</i> . . . . .	21
4.1.4	Contenu de la séquence . . . . .	21
4.1.5	Programmation du module ESP8266 . . . . .	23
4.1.6	Volume horaire attribué . . . . .	23
4.1.7	Déroulement de séquence . . . . .	23
4.1.8	Détail des séances . . . . .	24
4.1.9	Conclusion de la séquence pédagogique . . . . .	30
4.2	Séquence pédagogique en STI2D, Enseignements Technique Transversal . . . . .	31
4.2.1	La filière STI2D . . . . .	31
4.2.2	Thème de séquence et placement . . . . .	31
4.2.3	Compétences travaillées . . . . .	31
4.2.4	Détails de la séquence . . . . .	32
4.3	Séquence pédagogique en Classe Préparatoire aux Grandes Écoles - Informatique . . . . .	34
4.3.1	La nature du programme commun Informatique . . . . .	34
4.3.2	Thème de séquence et savoirs abordés . . . . .	34
4.3.3	Pré-requis et placement . . . . .	34
4.3.4	Déroulement de séquence . . . . .	35
<b>Appendices</b>		<b>39</b>
<b>A Fiche PySéquence complète pour la séquence en STI2D</b>		<b>39</b>

# 1 Introduction

Dans le cadre de l'épreuve orale de l'agrégation externe de Sciences Industrielles pour l'Ingénieur, j'ai réalisé ce dossier industriel avec le centre de recherche et développement Orange Labs de Lannion, dépendant de l'entreprise Orange, spécialiste en télécommunication et réseaux. Il a été l'occasion pour moi de présenter mes services et compétences pour répondre à une problématique de surveillance et réduction des impacts énergétiques des serveurs d'Orange.

## 1.1 Présentation de l'entreprise

Orange Labs est la structure de recherche rattachée à l'entreprise Orange, chaîne de télécommunications et fournisseur de services internet et téléphone. Orange, anciennement France Télécom - Orange, dispose d'un large réseau de téléphonie mobile à travers l'Europe et gère le réseau de téléphonie fixe français historique. Il est également fournisseur d'accès à Internet en France et à l'étranger et propose, en sus de l'accès à son réseau et Internet, des services Internet en hébergeant et fournissant des contenus. L'ensemble des services rendus par Orange peut être regroupé en trois catégories :

- les services de communication résidentiels, c'est-à-dire, la téléphonie fixe, Internet, la téléphonie IP, la visiophonie, la télévision numérique (la TV d'Orange) et les contenus multimédias (ex : vidéo à la demande),
- les services de communication personnels, mobiles, vers lesquels se déplacent également les services et contenus multimédias,
- les services de communication d'entreprise sous la marque Orange Business Services.

Orange commercialise donc un service de communication pour professionnels comme pour particuliers, en accès résidentiel comme mobile.

Le centre de recherche et développement Orange Labs Lannion regroupe plus de 1600 salariés et est divisé en départements de recherche. J'y ai contacté M. Stéphane Le Masson, directeur du département "Energie et Environnement". Parmi ses thèmes de recherche se trouve la consommation énergétique des serveurs.

## 1.2 Contexte de l'étude

Toutes les grandes entreprises, encore plus celles de la communication, possèdent une grande quantité de serveurs, ordonnés dans des datacenters et réalisant des opérations pour répondre aux services demandés par les utilisateurs. Orange, en particulier, propose entre autres des services de stockage et diffusion de contenus. Pour réaliser le traitement en un temps minimum pour garantir la satisfaction de l'utilisateur, l'entreprise investit dans un matériel de pointe pour ses ordinateurs jouant le rôle de serveurs.

Parallèlement à ces ordinateurs toujours plus puissants, on remarque que le code exécuté sur ces machines présente parfois des vices de conception ou de raisonnement. Ces erreurs de programmation pourraient dégrader les performances de l'ordinateur exécutant la tâche.

L'influence de la programmation sur les performances d'un ordinateur sont actuellement difficilement quantifiables, et c'est un domaine que le département "Energie et Environnement" d'Orange Labs commence à aborder.

## 2 Projet scientifique et technique, système associé

### 2.1 Problématique

Dans cette optique de quantification de l'impact d'un programme sur la consommation électrique, la problématique à laquelle j'ai été confrontée a été la suivante : réaliser un diagnostic informatique sur un support donné et mettre en évidence des gains énergétiques possibles en adaptant le code, et en particulier le langage, pour une tâche. Cette tâche a donné lieu à la conception d'un protocole de mesure et la réalisation d'un appareil de mesure abordable et simple d'utilisation. Un objectif prévu par Orange Labs est de pouvoir présenter les différences de consommation d'énergie dans le cadre d'une formation des équipes en charge de rédaction de code pour sensibiliser les programmeurs aux problématiques énergétiques.

### 2.2 Pistes de recherches

Le problème que j'ai dû résoudre était multiple. En effet, réaliser ce diagnostic nécessite d'investiguer sur plusieurs points, qui sont autant de pistes qu'il m'a été proposé d'explorer.

- Quels indicateurs semblent pertinents pour parler de la consommation d'un ordinateur ?
- Quelle consommation mesurer et comment la mesurer ?
- Quels sont les impacts d'un langage à un autre ?
- Quel impact pourrait avoir une sous-optimisation du code ?
- Quel montage pratique permet de mettre ces différences en évidence ?

Le choix de la tâche sur laquelle effectuer ces essais m'a été laissé libre, tout en mentionnant que l'usage de vidéo par les serveurs serait une piste pertinente pour les applications d'Orange. Il m'a donc été conseillé de faire des essais avec par exemple une tâche simple de traitement d'image. Parmi ces pistes de recherche, la différence entre les langages de programmation intéressait beaucoup Orange Labs et il m'a été proposé d'évaluer ces différences avec un protocole de comparaison et des mesures. Dans cette optique, j'ai décidé de restreindre avec Orange Labs le champ d'action à un composant central, siège de la majorité des calculs et de la consommation électrique : le processeur.

### 2.3 Système utilisé

L'environnement et le matériel étant laissés libres, j'ai choisi un ordinateur de bureau classique sur lequel je peux effectuer différentes mesures et lancer des programmes. Il dispose d'un processeur Intel Core 2 Duo E8400 disposant de deux coeurs physiques cadencés à 3,00 GHz. Il exécute le système d'exploitation GNU/Linux Ubuntu 16.04. J'ai fait le choix d'un ordinateur de bureau car le processeur est alimenté par quatre câbles spécifiques où je peux surveiller le courant continu entrant pour avoir une estimation de la puissance demandée. Ces câbles sont visibles en figure 1 : les deux jaunes ont une tension continue de 12V, les deux noirs sont la masse.

Je relève le courant entrant vers le processeur dans les deux câbles jaunes à l'aide d'une pince ampèremétrique à effet Hall visible en figure 2.

Avec la mesure du courant continu, en multipliant par la tension continue appliquée de 12V, j'obtiens une mesure de la puissance instantanée du processeur. Pour justifier l'intérêt de la mesure sur le processeur, je relève la consommation instantanée de l'ordinateur complet grâce à un wattmètre numérique à la prise, et du processeur avec la pince ampèremétrique, au repos et pendant un calcul lourd (ici, le calcul de 8 millions de décimales de Pi par la méthode de Borwein, qui nécessite environ deux minutes à la machine de test). Les résultats sont présentés en table 1.

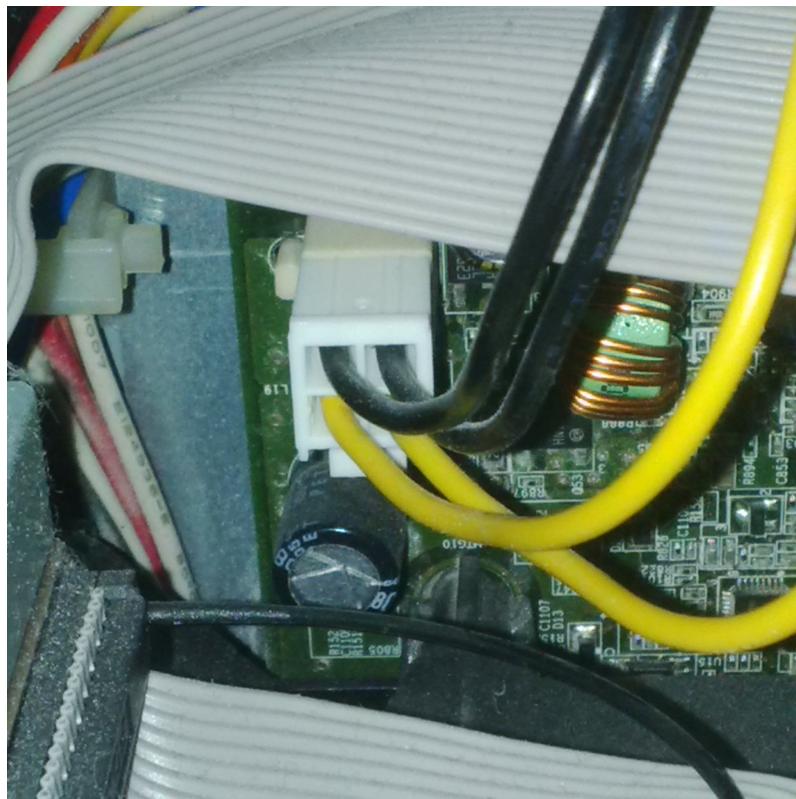


Figure 1 – Câbles d'alimentation spécifiques au processeur.

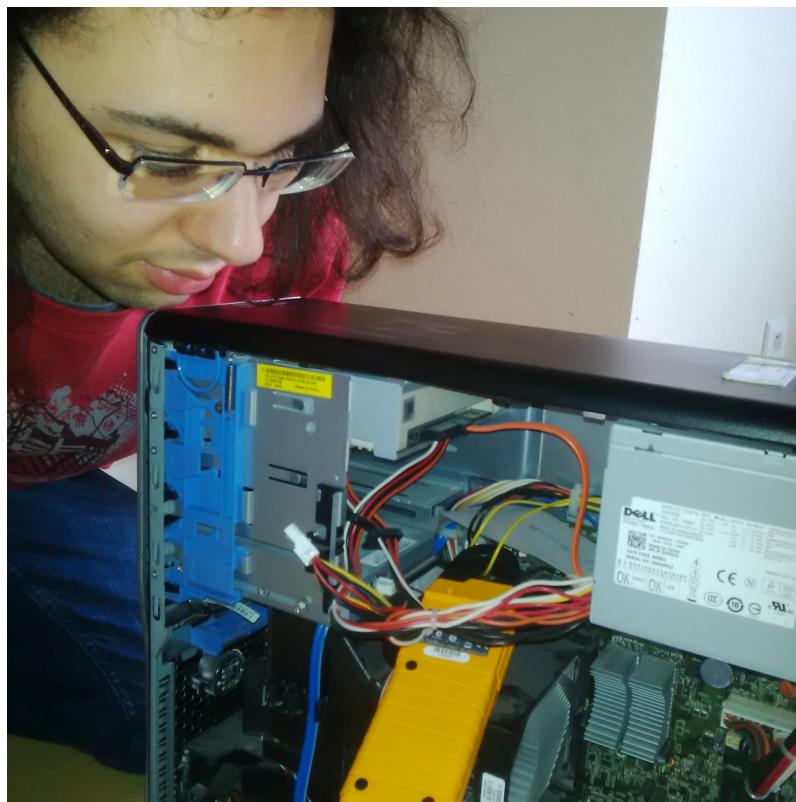


Figure 2 – Mesure du courant par pince ampèremétrique.

On remarque alors que la puissance consommée par le processeur représente plus d'un tiers de la puissance totale. D'autre part, on remarque qu'au repos, le processeur consomme très peu par rapport

	Puissance du PC (W)	Puissance du processeur (W)	Part du processeur
Au repos	78,4 W	8 W	10,7%
En charge	120 W	45 W	37,5%

Table 1 – Relevés de la puissance du processeur et de l'ordinateur complet.

au reste du PC. Il s'agit donc d'un point crucial sur lequel des économies d'énergies auront un fort impact lors de demande de calculs.

L'enregistrement des mesures et le traitement seront des points abordés dans la section suivante.

### 3 Développement logiciel

#### 3.1 Comparer différents langages de programmation

##### 3.1.1 Choix des langages

Une proposition de comparaison a été de regarder les différences entre différents langages de programmation. J'en ai choisi quatre. Je voulais mettre en valeur des langages assez différents, c'est pour cela que mon choix s'est porté sur :

- Un langage compilé : le C++,
- Un langage interprété : le Python,
- Un langage compilé en code binaire exécuté dans une machine virtuelle : le Java
- Un autre langage compilé à la volée dans la même machine virtuelle : le Scala

D'autres différences existent mais ne seront pas traitées ici, en particulier la gestion de la mémoire et la présence de ramasse-miette, ou leur tendance à présenter leur code (paradigme de programmation).

##### 3.1.2 Choix de la tâche à effectuer

Le choix de la tâche à faire accomplir à chaque programme m'a également été laissé libre ; cependant, il m'a été conseillé de m'inspirer du domaine de l'image et de la vidéo pour y apporter une dimension concrète pour les ingénieurs en formation. Afin de garder une tâche facilement réalisable, assez simple, mais relativement lourde, j'ai choisi de faire appliquer un flou sur une image haute définition de 3840 par 1600 pixels avec une convolution par noyau telle qu'expliquée par la figure 3. On calcule la moyenne des pixels aux alentours et on stocke cette moyenne dans le pixel courant. Cette opération est chronométrée par le programme qui affiche son temps.

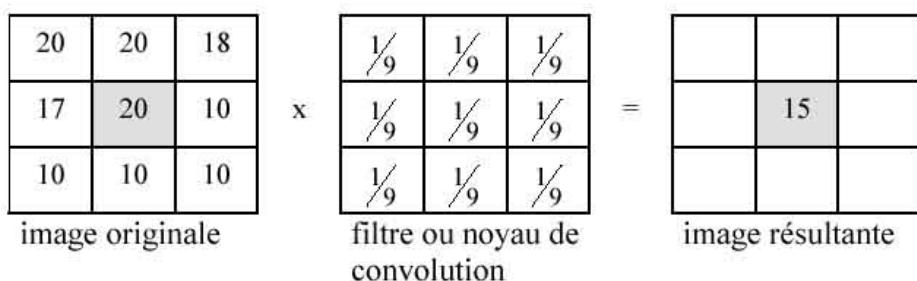


Figure 3 – Exemple de convolution de noyau pour du traitement d'image.

Pour être sûr des opérations menées par l'ensemble des programmes, les étapes réalisées sont les suivantes :

- Chargement de l'image
- Conversion en niveaux de gris et en valeurs entières entre 0 et 255, pour s'assurer que le traitement se fait sur le même type de données,
- Application du flou 5 fois, chronométré
- Affichage de l'image originale et de l'image floue

Les quatre programmes sont disponibles en annexe et en format numérique à l'adresse suivante :  
[https://github.com/j-levy/languages\\_performance\\_tests](https://github.com/j-levy/languages_performance_tests)

### 3.1.3 Calcul de l'énergie consommée

Le calcul de l'énergie consommée pour réaliser la tâche a pour vocation de comparer les programmes entre eux. Pour un temps d'exécution  $t$ , une puissance consommée en charge  $P_{charge}$ , l'énergie dépensée à effectuer la tâche s'écrit :

$$E_{tache} = t * P_{charge}$$

L'énergie de la tâche correspond donc à l'énergie dépensée pendant l'exécution. On fait alors l'hypothèse que les serveurs ont globalement un temps d'attente entre deux tâches négligeable et sont presque toujours en train de réaliser une tâche calculatoire. Cela se traduit par le fait que, lorsque la tâche est réalisée, le serveur peut passer à la réalisation d'une autre tâche. Cette hypothèse est globalement réaliste dans la plupart des datacenters ; et elle reste valide dans le cas où la puissance de calcul est louée (dans ce cas, le bailleur qui aura terminé sa tâche plus vite pourra louer à quelqu'un d'autre, et on ne paye pas de coût supplémentaire).

### 3.1.4 Résultats et observations

Les mesures de temps d'exécution et de puissance donnent les résultats en table 2.

Programme	C++	Python	Java	Scala
Puissance en charge (W)	43,5	30	39,4	37,3
Temps (s)	2,5	168	12,7	13,8
Energie consommée (J)	150	5040	501	517

Table 2 – Relevés de consommation et d'énergie pour les quatre langages.

On remarque que les puissances sont relativement similaires mais que les durées varient sur trois ordres de grandeur : la seconde, la dizaine de secondes, et la centaine de secondes. En positionnant chaque langage dans un graphe Puissance-Temps, on obtient la figure 4. Le temps est mis en échelle

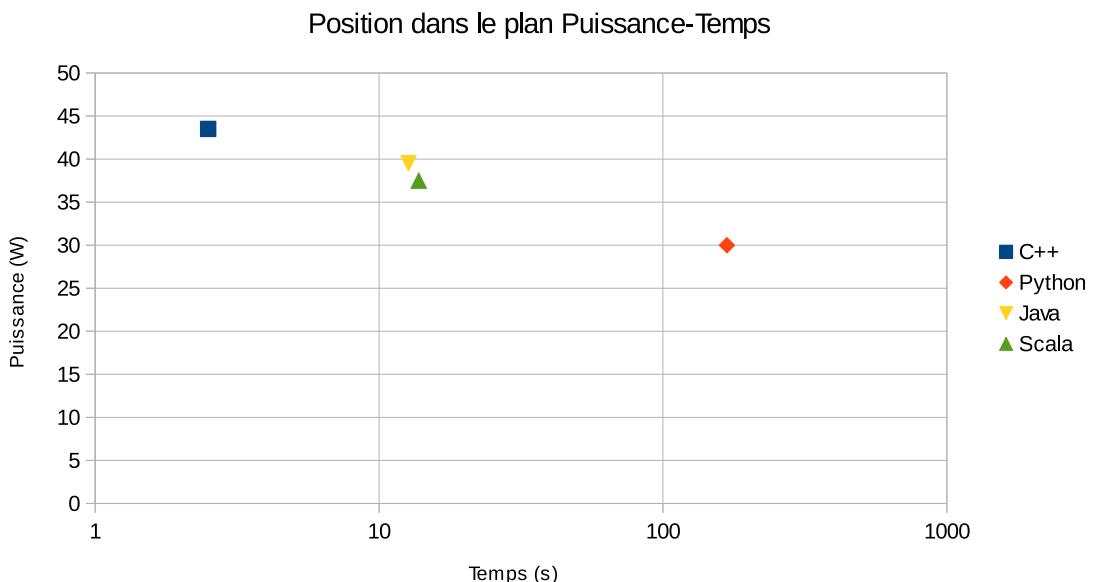


Figure 4 – Position du langage dans le plan Puissance-Temps.

logarithmique pour refléter ces différents ordres de grandeur. Sur ce graphe, plus un langage se

rapproche du coin inférieur gauche, meilleur il est.

Traçons maintenant les énergies consommées pour exécuter cette tâche de flou dans le temps maximal (figure 5)

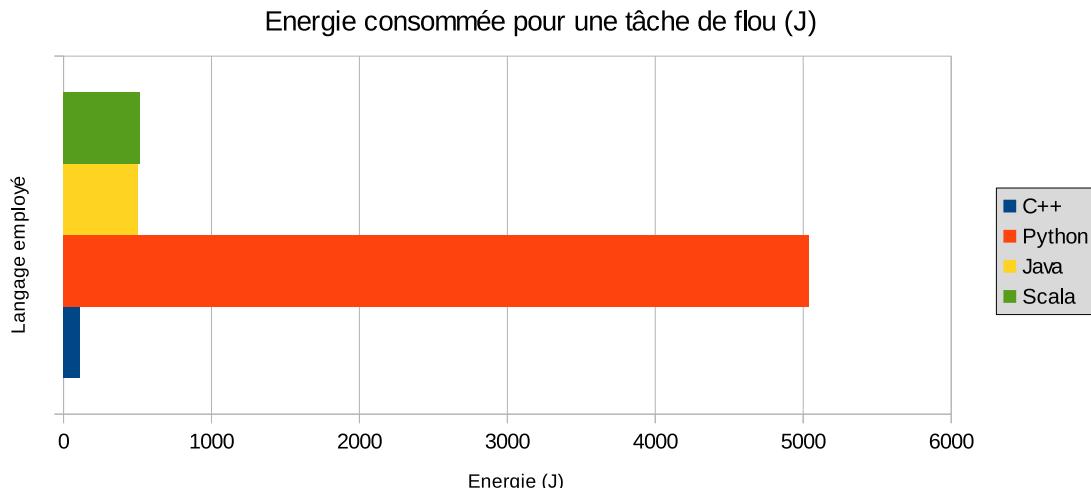


Figure 5 – Quantité d'énergie consommée pour l'exécution de la tâche dans le temps maximal.

Bien que les puissances mises en jeu soient du même ordre de grandeur, le temps mis par les différents programmes change drastiquement l'énergie totale consommée. Par exemple, Python (en orange sur les graphes) a besoin de moins de puissance (30W contre les 43,5W du C++), mais il met tellement de temps à s'exécuter que l'énergie consommée revient à dix fois celle de ses concurrents. On observe très peu de différences entre Java et Scala, ce qui pourrait venir des optimisations faites par la machine virtuelle Java quel que soit le langage. Enfin, le C++ est le plus économique, quand bien même la puissance demandée est la plus haute, l'énergie consommée est minimale grâce à son temps d'exécution de 2,5 secondes sur cette tâche.

Finalement, on constate que le langage a une influence sur la puissance demandée par le processeur. Différents langages nécessitent différentes puissances. Cependant, ces puissances mises en jeu sont du même ordre de grandeur, et n'ont finalement que peu d'influence sur la consommation d'énergie. Le facteur le plus important reste la rapidité d'exécution.

Une explication possible serait que, dans le cas du C++, le programme est de taille plus réduite qu'une machine virtuelle, donc les données de l'image peuvent être potentiellement chargées en plus grande partie dans le cache. Le traitement est alors fait sur une mémoire physiquement très proche du processeur, et est très rapide. Si on suppose qu'à l'inverse, une machine virtuelle comme Python prend de la place en cache et charge l'image en RAM puis que le processeur y accède en faisant des requêtes vers la RAM, il va attendre pendant longtemps que les informations arrivent depuis la RAM. Il est alors possible que sa consommation diminue, mais relativement peu, alors que son temps d'attente grandit, ici de manière exponentielle (on est deux ordres de grandeurs au dessus). Et globalement la consommation électrique s'en ressent.

## 3.2 Relever la consommation électrique : création d'un instrument connecté

### 3.2.1 Besoins associés à la mesure

Pour relever la consommation du processeur, je mesure le courant entrant à l'aide d'une sonde de courant à effet Hall. Elle délivre une tension image du courant. Pour la lire et enregistrer ses valeurs, le premier montage que j'ai réalisé utilisait l'entrée analogique d'une carte Arduino Leonardo pilotée par le logiciel Matlab. Ce dernier installait un logiciel propriétaire sur la carte Arduino et se chargeait de la communication entre les deux. L'Arduino est dotée d'un convertisseur analogique-numérique (CAN) sur 10 bits, offrant 1024 niveaux avec une pleine échelle de 0V à 5V. La figure 6 montre un diagramme de blocs internes SysML qui présente ce mode de mesure.

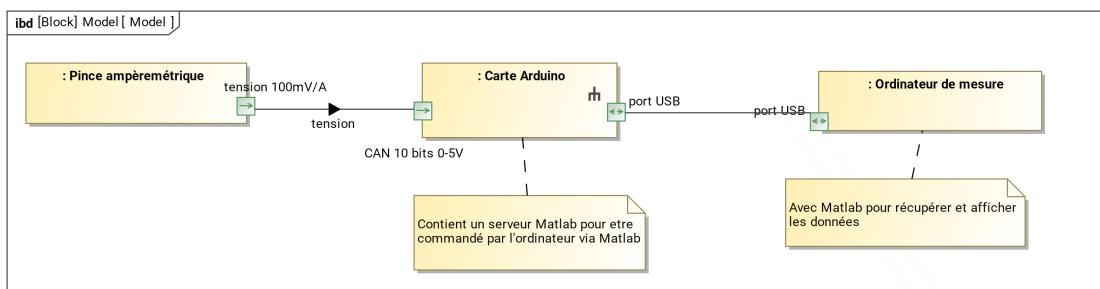


Figure 6 – Diagramme de blocs internes présentant le premier système de mesure par carte Arduino et Matlab.

Bien que le système fût fonctionnel, il avait plusieurs inconvénients, comme l'utilisation d'un logiciel propriétaire coûteux et la nécessité d'être physiquement proche de la machine à mesurer. J'ai alors changé de microcontrôleur et éliminé le besoin d'un logiciel tiers particulier pour les mesures. Pour cela, j'ai échangé la carte Arduino contre une carte NodeMCU basée sur le microcontrôleur ESP8266.

L'ESP8266 est un microcontrôleur 32 bits cadencé à 80 MHz disposant de fonctionnalités Wi-Fi et d'un système de gestion de fichiers pour sa mémoire de 4Mo. Il peut se connecter sur un point d'accès ou devenir lui-même point d'accès Wi-Fi, et servir des fichiers par différents protocoles, par exemple HTTP pour un service Web. C'est un microcontrôleur adapté à l'Internet des Objets. En outre, la carte complète NodeMCU est commercialisée par le constructeur à environ 3€, là où une carte Arduino coûte entre 20€ et 35€ .

Enfin, pour permettre un retour d'information simple, j'ai rajouté à la carte NodeMCU un écran LCD affichant diverses informations, comme l'adresse IP pour se connecter ou la valeur mesurée en cours. Il est piloté grâce à une bibliothèque dédiée et se connecte en I<sup>2</sup>C.

L'ensemble est alimenté à travers un port USB. Sur la photo en figure 7, j'utilise une batterie USB.

Pour la programmation, j'ai embarqué sur la carte l'interpréteur Lua NodeMCU largement documenté qui donne son nom à la carte. Il permet d'avoir accès très facilement au système de fichiers, chose cruciale pour servir des pages web et stocker des mesures. En outre, le langage Lua est tourné vers la programmation événementielle, qui évite de bloquer le microcontrôleur pendant une opération et révélera beaucoup d'intérêts par la suite pour cette application.

### 3.2.2 Présentation du système de mesure avec l'ESP8266

La carte NodeMCU embarque un microcontrôleur ESP8266 du fabricant Espressif, ainsi que le matériel nécessaire à sa programmation (en particulier, un pont port USB vers liaison série pour le programmer). C'est un matériel open source, plusieurs constructeurs fabriquent des cartes identiques. La figure 8 présente le modèle du fabricant LoLin.

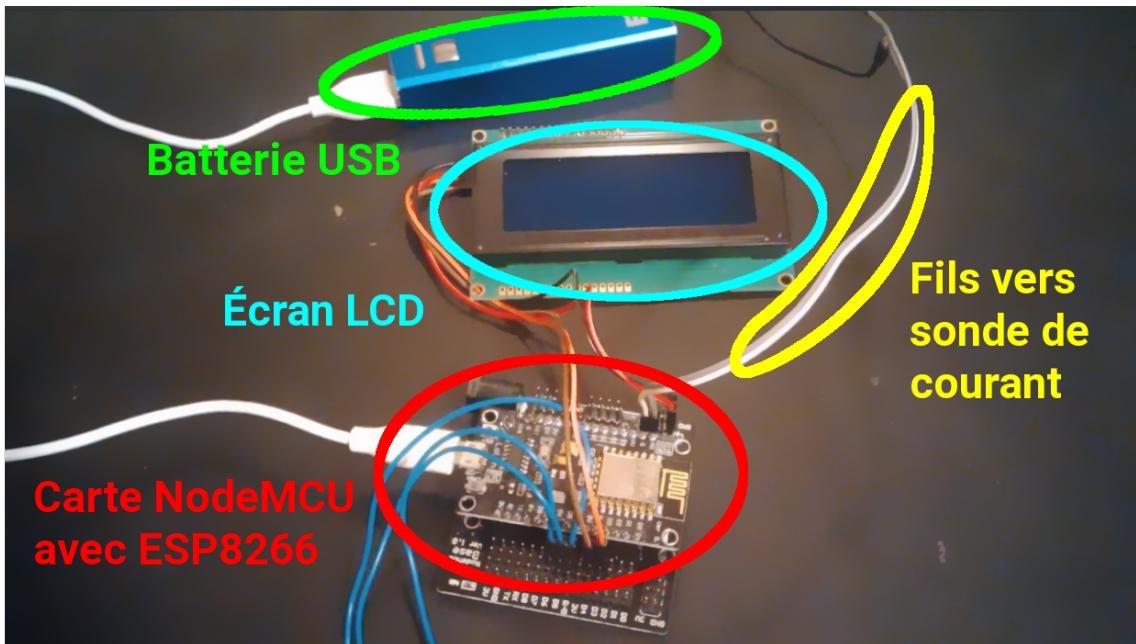


Figure 7 – Montage de l'ensemble ESP8266, sonde de courant, écran LCD et alimentation USB.

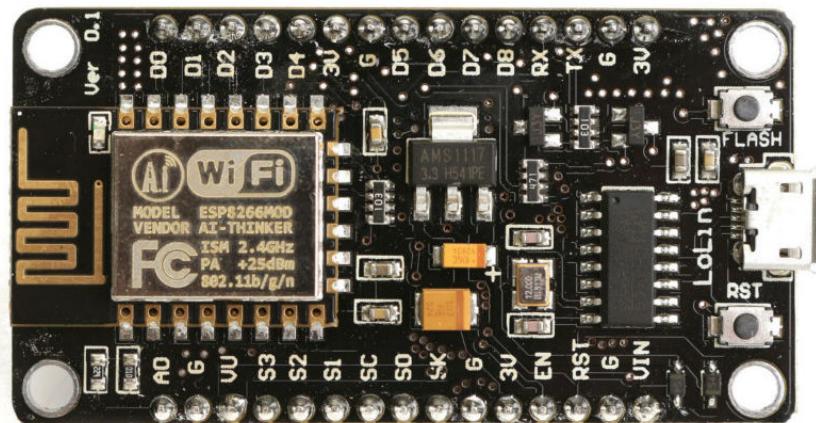


Figure 8 – Photographie de la carte NodeMCU de LoLin.

Il dispose d'une broche d'entrée analogique avec un CAN sur 10 bits, avec une pleine échelle de 0V à 3,3V. Les données sont lues et stockées dans un fichier sur l'ESP8266. Ce dernier utilise ses capacités Wi-Fi pour créer un point d'accès sur lequel peut venir se connecter un terminal Wi-Fi (un ordinateur, un smartphone...), et communiquer avec un serveur Web réalisé dans l'ESP8266. Le terminal affiche donc une page web disposant de boutons pour réaliser différentes actions, comme lancer ou arrêter une mesure, récupérer les mesures stockées, ou effacer les mesures. Enfin, il peut également demander à afficher la dernière mesure, ou celle courante s'il y en a une, sous forme d'un graphe directement sur la page web.

Le système de mesure et son environnement sont détaillés dans le diagramme de blocs internes SysML de la figure 9.

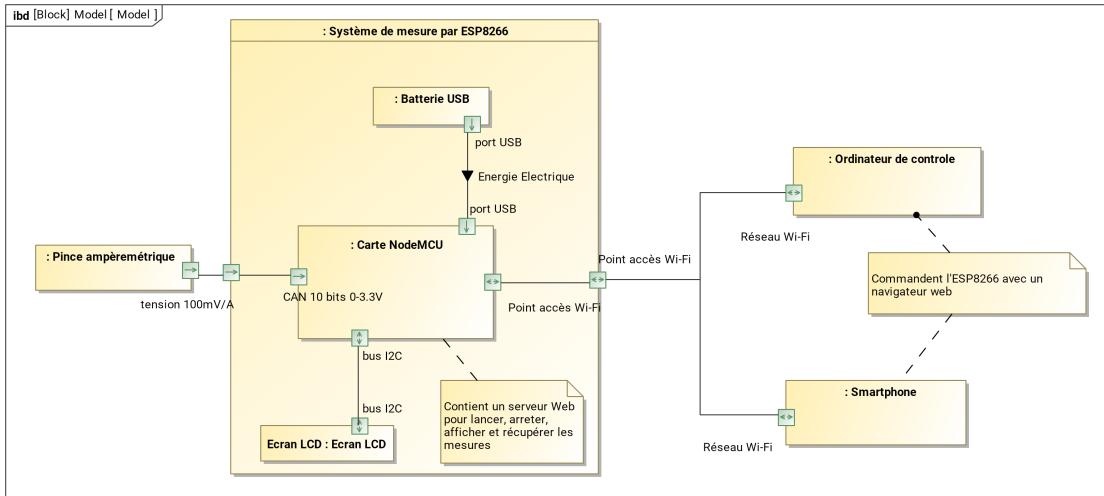


Figure 9 – Diagramme de blocs internes présentant l'architecture de la solution de mesure par carte NodeMCU.

### 3.3 Intégration des fonctionnalités

La NodeMCU se programme à l'aide de scripts Lua stockés dans sa mémoire et interprétés avec le micrologiciel NodeMCU embarqué. Pour envoyer ces scripts et avoir un retour de fonctionnement, la carte dialogue avec l'ordinateur à travers son port USB utilisé comme interface série, de la même manière que la carte Arduino. Le constructeur ne propose pas d'environnement de développement intégré, donc j'ai utilisé un environnement créé par la communauté, *ESPlorer*, visible en figure 10. Il facilite le télémétrage de fichier et l'accès à ceux-ci.

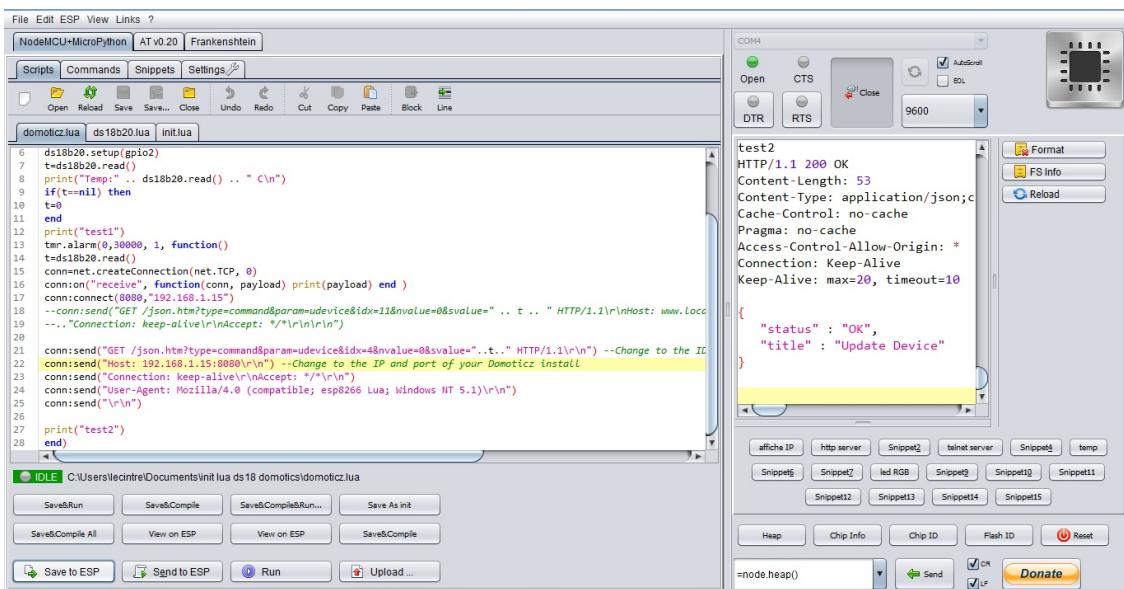


Figure 10 – L'environnement de développement ESPlorer<sup>1</sup>.

1. <https://esp8266.ru/esplorer/>

### 3.3.1 Implémentation du serveur Web et envoi de fichiers

Au démarrage, la carte exécute le script `init.lua`, qui se chargera d'appeler les autres scripts. La mise en place du serveur web se crée dans le script `AP_server.lua` qui écoute le port 80 et crée une connexion TCP quand il a une requête. La requête HTTP est ensuite analysée et l'ESP8266 va alors servir le fichier demandé, et, ou exécuter un script.

Le diagramme d'activité SysML en figure 11 montre les réponses de l'ESP8266 par son serveur web sous une forme proche de l'algorigramme. Il met en évidence l'approche événementielle de la solution.

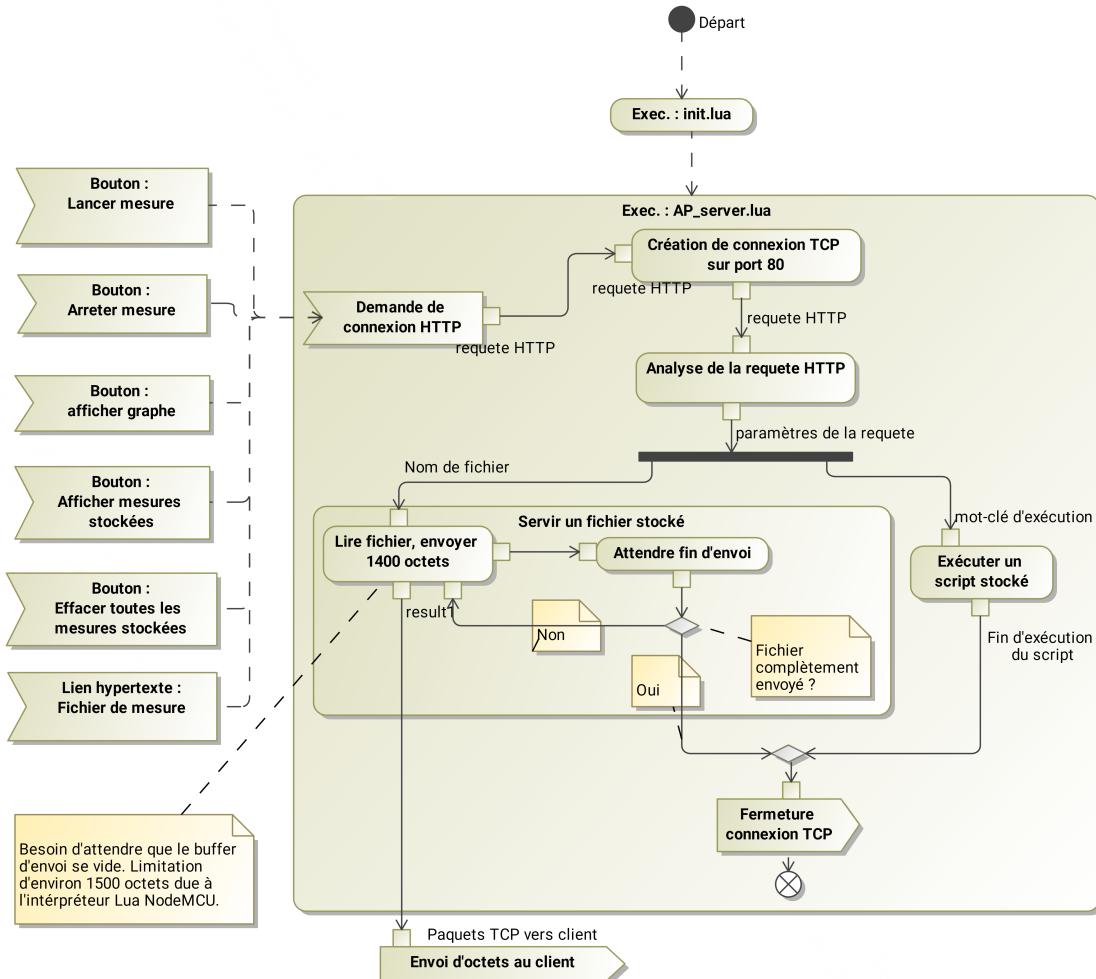


Figure 11 – Description algorithmique du fonctionnement du serveur web de l'ESP8266.

On voit sur la figure 11 que l'envoi d'un fichier doit se faire par blocs d'environ 1400 octets, et on doit attendre que chaque envoi soit terminé avant d'en exécuter un nouveau. Sans vérification d'envoi, la mémoire tampon dédiée à l'envoi de paquets de l'ESP8266 se remplit, jusqu'à saturation et levée d'une exception, ce qui provoque le redémarrage de l'interpréteur Lua.

Si le fichier demandé n'existe pas, le serveur web renvoie l'utilisateur sur la page principale.

### 3.3.2 Script de mesure de la tension de la sonde

Pour mesurer la sonde, on utilise l'entrée disposant d'un CAN sur l'ESP8266. Le microcontrôleur peut effectuer jusqu'à 80 mesures par seconde, mais cela provoque beaucoup d'écritures sur la mémoire

interne et génère des gros fichiers qui prennent de la place et du temps à envoyer. La fréquence de mesure est donc choisie à 1Hz, ce qui permet d'avoir des mesures relativement fréquentes, mais assez lentes pour ne pas encombrer la mémoire et pouvoir être affichées sur l'écran LCD.

Puisqu'une fois la mesure lancée, elle doit continuer de mesurer toutes les secondes jusqu'à interruption externe, le pseudo-code de mesure pourrait ressembler à ceci :

```
while condition_arret do
    --effectuer la mesure
    --écrire la mesure dans un fichier
    --attendre 1 seconde
end
```

Cependant, utiliser une boucle `while` bloque l'interpréteur Lua dans cette boucle, et sitôt qu'une mesure commence, le serveur web ne peut plus répondre.

Il a donc fallu adopter une représentation évènementielle. Pour faire un relevé, on appelle la fonction `acquisition()` mentionnée dans l'extrait ci-après. L'appel s'effectue grâce à une minuterie (un *timer*) nommée `tmr_measure` qui se déclenche toute les secondes. Au sein de la fonction `acquisition()` se trouve la vérification de la condition de fin, qui arrêtera le *timer*.

Voici l'extrait commenté du script Lua lançant une mesure :

```
-- On vérifie l'état du timer pour ne pas relancer une mesure si une est
-- déjà active
running, mode = tmr_measure:state()
if running then
    print("Mesure déjà lancée")
else
    -- Création du fichier où les mesures sont écrites à la volée. Le
    -- mode W+ efface le contenu du fichier existant et permet la
    -- lecture et l'écriture. C'est ce fichier qui sera lu pour créer
    -- un graphe.
    log = file.open("last_measure.html", "w+")

    --[[ A l'arrêt de la mesure, on copiera tout le contenu du fichier
    -- temporaire vers un fichier final. Son nom sera donné par la
    -- valeur d'uptime (nombre de secondes écoulées depuis l'allumage)
    -- avec "tmr.time()" :
        filename = "meas_"..tmr.time()..".txt"
    Cela permet d'avoir un nom de fichier probablement unique. Si le
    -- fichier existe, on incrémente le nombre donné par tmr.time()
    -- jusqu'à trouver un nom disponible. Le préfixe "meas_" permet de
    -- le retrouver lorsqu'on cherchera les mesures stockées.]]]

    if not(log) then
        print("Fichier indisponible")
    end
    delay = 1000 --ms

    --On enregistre le timer : il doit lancer toutes les secondes la
    --fonction acquisition().
    tmr_measure:register(delay, tmr.ALARM_AUTO, function() acquisition
        () end)
    --Enfin, on le démarre. Dans la fonction acquisition() se trouve
    --un test pour l'arrêt du timer.
    tmr_measure:start()
end
```

L'écriture dans le fichier est réalisé en utilisant un format de base de données adapté à un fichier, le format JSON (JavaScript Object Notation). Cela permet d'utiliser directement ce fichier pour l'affichage en graphe.

### 3.3.3 Interface du serveur web

Le serveur web envoie au client une page HTML contenant en son sein le code HTML qui structure la page, le code CSS qui la met en forme et le code JavaScript qui permet de la rendre interactive. Au vu des capacités limitées du programme serveur, j'ai choisi de regrouper tout le code d'une page dans un seul fichier, pour éviter d'envoyer plusieurs fichiers en même temps.

L'interface est visible sur la figure 12. Elle s'adapte aux différentes tailles d'écran pour être exploitable sur un ordinateur comme sur un smartphone.

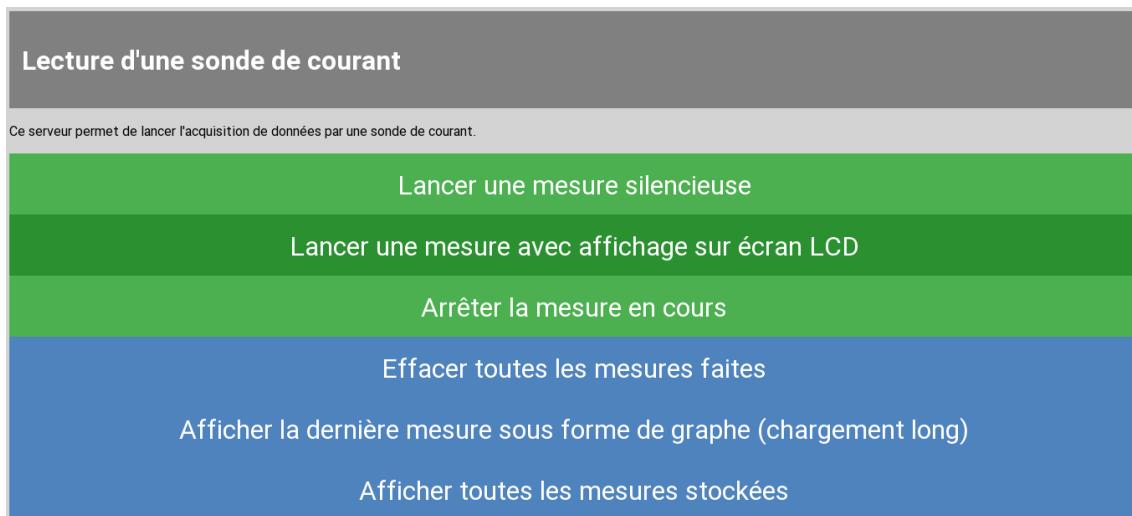


Figure 12 – Interface du serveur web : menu d'accueil.

L'affichage des mesures stockées est re-généré à chaque fois qu'on demande la page. L'appui sur le bouton déclenche la récupération des noms de tous les fichiers présents en mémoire, puis leur sélection suivant leur nom : les fichiers de mesure ont leur nom commençant par `meas_` suivi d'un nombre. Enfin, ces noms sont ajoutés dans une chaîne de caractères créant ainsi une page HTML.

Un exemple de page servie contenant les noms stockés est disponible ci-après.

```
<!doctype html>
<html lang="fr">
    <head>
        <title>Lecture de sonde de courant</title>
        <meta charset="UTF-8">
    </head>
    <body bgcolor=white>
        <p>Voir les mesures sur le stockage interne :
        <ul>
            <!-- lignes re-générées automatiquement -->
            <li> <a href="meas_45.txt">meas_45.txt</a>
            <li> <a href="meas_13.txt">meas_13.txt</a>
        <!-- fin des lignes re-générées automatiquement-->
        </ul>
    </body>
</html>
```

Enfin, l'affichage de la mesure sous forme de graphe utilise la bibliothèque JavaScript Canvasjs.js<sup>1</sup>, que j'ai incorporée dans le fichier HTML et que j'ai allégée en enlevant les fonctionnalités qui ne me servaient pas (par exemple l'affichage des graphes camembert) pour ne garder que le graphe ligne. Les points sont fournis au format JSON.

La page est ainsi générée :

- Duplication du fichier HTML contenant le code d'affichage de graphes
- Insertion des points de mesure dans le fichier dupliqué
- Ajout de la fermeture du fichier (balises fermantes)

Le résultat est visible en figure 13. Passer la souris sur un point du graphe (ou le toucher sur un smartphone) donne ses coordonnées.

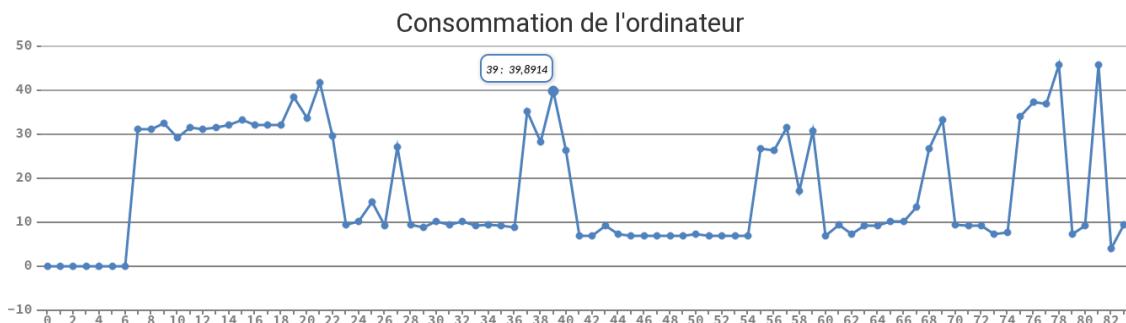


Figure 13 – Exemple de graphe obtenu de consommation (en W) en fonction du temps (en secondes).

Bien qu'une fois allégé, le fichier contenant tout le code nécessaire à l'affichage ne pèse plus que 110 ko, l'ESP8266 met environ 2 secondes à le dupliquer et 8 secondes à l'envoyer. Cela représente un temps d'attente conséquent, qui est dû aux capacités limitées du microcontrôleur. Un appareil plus conséquent, comme un ordinateur monocarte Raspberry Pi, aurait un temps de réaction bien plus court. Cependant, ce dernier coûte dix fois le prix de la carte NodeMCU.

### 3.3.4 Pistes d'amélioration envisagées

Plusieurs pistes d'amélioration sont envisageables. Certaines visent à améliorer les performances du systèmes, d'autres pourraient lui rajouter des fonctionnalités.

Une idée d'amélioration de performance serait d'utiliser un convertisseur analogique-numérique dédié codant sur plus que 10 bits. Par exemple, l'ADS1115 de Texas Instrument est un CAN sur 16 bits transmettant ses informations par un bus I<sup>2</sup>C, jusqu'à 860 échantillons par seconde. On peut le trouver sous forme de carte prête à souder sur PCB pour moins de 3 €. Le passage de 10 à 16 bits multiplierait la résolution de mesure par un facteur 2<sup>6</sup>, sans véritable contrainte sur la vitesse d'acquisition (puisque l'écriture sur mémoire interne est bien plus lente et limitante).

La mise en réseau de la NodeMCU n'a pas été réalisée pour des questions d'indépendance d'environnement. Telle que présentée, la solution fonctionne dans toutes les situations. Cependant, connecter la carte NodeMCU à Internet à travers un point d'accès Wi-Fi est possible ; mais il faudrait alors qu'elle vienne s'enregistrer sur un serveur DHCP libre tel que <http://www.noip.com>. Elle pourrait alors bénéficier d'une URL pour accéder à son contenu. De plus, les fichiers dépendants stockés pourraient être distribués sur Internet, en particulier les codes JavaScript lourds à transférer : on gagnerait en vitesse d'exécution puisque la NodeMCU devrait envoyer seulement le code HTML avec les données, et un serveurs tiers enverrait le JavaScript.

Une fonctionnalité testée mais non-implémentée est la synchronisation entre l'équipement de mesure et le programme à mesurer. L'ESP8266 peut dialoguer sur le port USB d'un ordinateur, en l'utilisant

1. <http://canvasjs.com/>

comme port série. On peut alors envoyer une commande à l'interpréteur Lua vers le port série en utilisant la description des ports en tant que fichiers, sous les systèmes UNIX. Par exemple, pour lancer une mesure, il suffit d'exécuter dans un terminal la commande suivante :

```
echo "dofile(\"measure_LCD.lua\")" > /dev/ttyUSB0
```

Cela revient à rediriger la sortie de la commande `echo`, qui ne fait qu'afficher un texte, vers un fichier `via >`. Le fichier choisi est le port série de l'ESP8266, ici `/dev/ttyUSB0`. L'interpréteur Lua va ainsi lire la commande `dofile()` qui exécute le script passé en paramètre. On peut remarquer les caractères d'échappement `\` pour envoyer les guillemets correctement. Cette commande a été testée, elle est effectivement fonctionnelle, mais elle n'a pas été implémentée dans les programmes de test précédents.

En fin de compte, on peut intégrer ce genre de commande dans un environnement de développement qui, pendant le programme, lance et arrête les mesures, et affiche la consommation en direct en ouvrant une page web. Ces possibilités d'extension ont été appréciées par Orange Labs et ouvrent également des portes sur l'appréciation de la qualité d'un code au moment de sa conception.

## Synthèse du projet scientifique et technique

Cette étude m'a permis de réaliser deux livrables pour Orange Labs. Ces deux parties correspondent aux deux besoins résultant de la même problématique : comment comparer différents langages de programmation du point de vue énergétique.

D'une part, j'ai réalisé et mis à leur disposition une série de programmes avec code documenté servant de test de performance. Ils réalisent la même opération sur la même ressource, une image haute définition de 3840 par 1600 pixels dans notre cas. Une attention particulière a été portée à la correspondance entre les algorithmes utilisés dans chacun des programmes de test, ainsi qu'aux types de données manipulés.

D'autre part, j'ai réalisé un module connecté pour un instrument de mesure, la pince ampèremétrique, permettant de relever le courant sur les fils de tension continue dans un PC de bureau. La puissance se déduit en effectuant le produit du courant continu par la tension continue. Cet instrument connecté permet de stocker des mesures au format JSON et de les afficher sur un serveur web embarqué dans un microcontrôleur. Ce dernier se distingue par son prix très contenu : en effet, l'ensemble des fonctions de mesure, de Wi-Fi et de serveur web sont embarqués dans une carte à microcontrôleur qu'on peut trouver pour environ 3€.

Cela permet d'envisager d'en utiliser un grand nombre pour vérifier des consommations sur un vaste parc. L'intérêt de la connectivité Wi-Fi utilisée réside alors dans l'agrégation possible des données de multiples instruments au sein d'un seul terminal. Un navigateur web suffit pour interagir avec l'objet, que ce soit pour le commander ou récupérer les mesures. Cela le rend utilisable à travers la plupart des terminaux connectés. (ordinateur, smartphone, ou même un autre objet Wi-Fi).

Cette étude m'a permis de mettre en œuvre un large panel de compétences, parmi lesquelles du traitement d'image, de la programmation procédurale (C++), objet (Java), événementielle (Lua), web (HTML). Au cours de la réalisation, j'ai rencontré et résolu divers problèmes, par exemple concernant l'utilisation de bibliothèques, l'envoi de gros fichiers par la carte NodeMCU, ou la gestion côte-à-côte des fonctions de serveur web et de mesure. Il en résulte une grande richesse scientifique et technologique exploitable au sein de diverses séquences pédagogiques présentées ci-après.

## 4 Séquences pédagogiques

La première séquence développée se place en IUT Génie Electrique et Informatique Industrielle (GEII). Elle a pour thème principal la programmation de microcontrôleurs.

Une autre séquence en classe de STI2D en Enseignement Technique Transversal sera présentée. Elle permet d'aborder les notions liées au développement durable et à la consommation énergétique des systèmes.

Enfin, une troisième séquence pédagogique a été élaborée en cours d'informatique commun à toutes les classes préparatoires aux grandes écoles (CPGE). Elle porte sur la manipulation de fichiers et de structures tableaux 2D pour des images.

### 4.1 Séquence pédagogique en IUT GEII

La séquence pédagogique proposée en IUT GEII traite de la programmation d'un microcontrôleur moderne dans un système embarqué.

#### 4.1.1 Présentation de la filière

L'Institut Universitaire de Technologie est une filière se situant après le Baccalauréat et se déroulant sur deux ans. Il mène au Diplôme Universitaire de Technologie (DUT). Faisant partie d'une Université, l'IUT appartient à l'enseignement supérieur en France.

Dans le cas de l'IUT GEII, les étudiants sélectionnés proviennent majoritairement de baccalauréat scientifique et STI2D. Une partie des étudiants y accède également depuis un baccalauréat professionnel. Au terme de la formation, la grande majorité des étudiants poursuivent leurs études vers une licence professionnelle ou une préparation vers les écoles d'ingénieurs. Quelques uns décident de se lancer dans la vie active. Le positionnement de l'IUT GEII dans le parcours des étudiants est résumé en figure 14.

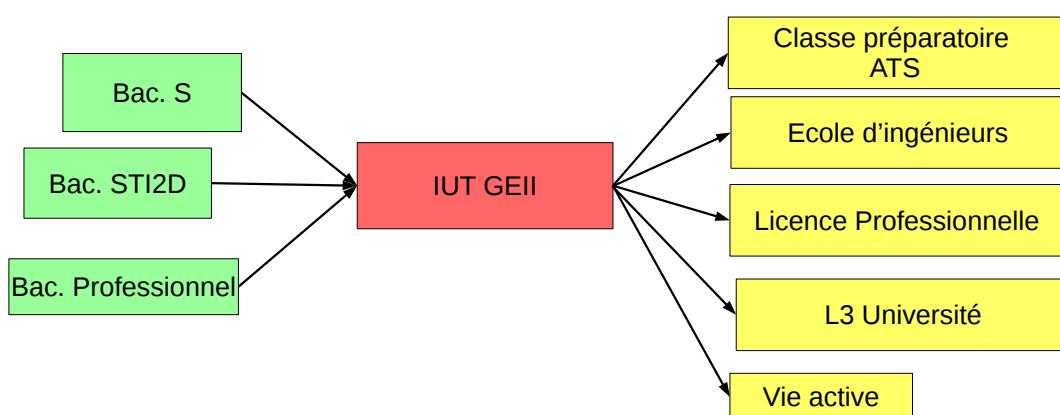


Figure 14 – Parcours possibles avec l'IUT GEII.

Chaque année se forment des promotions d'environ 120 élèves dans les IUT GEII, réunis en amphithéâtre pour les cours, et organisées en classes de 24 élèves en moyenne pour les travaux dirigés. Ces classes sont ensuite réduites à des demi-groupes de 12 étudiants pour les séances de travaux pratiques.

#### 4.1.2 Le Programme Pédagogique National

Le Programme Pédagogique National (PPN) précise les modalités d'enseignement en IUT GEII. Le chapitre "Référentiels d'activités et de compétences" décrit les activités que doivent accomplir les étudiants sous forme de paire Activité-Compétences, de base et spécifiques. J'ai retenu pour ma séquence pédagogique la paire spécifique présentée en figure 15.

Développement de petits systèmes embarqués (limité aux cas à complexité modérée)	<ul style="list-style-type: none"> <li>modéliser un système dans son environnement</li> <li>conduire une démarche de développement logiciel (analyse, algorithme, codage, test)</li> <li>utiliser un outil de développement croisé</li> <li>utiliser un langage de description matérielle des circuits (conception, simulation)</li> <li>intégrer ensemble matériel et logiciel</li> </ul>
--	--

Figure 15 – Paire Activité-Compétences spécifique associée aux systèmes embarqués en IUT GEII.

La formation est divisée en trois thèmes, un par semestre, comme indiqué dans la figure 16.

Enseignements	S1 Initiation	S2 Développement	S3 Approfondissement	S4 Renforcement	Total
	heures	heures	heures	heures	heures
Thème 1 : Composants, systèmes et applications	240	240	240		720
Thème 2 : Innovation par la technologie et les projets	150	135	150	180	615
Thème 3 : Formation scientifique et humaine	120	135	120	90	465
(UE41 : Stage)					
<b>Total Heures encadrées</b>	<b>510</b>	<b>510</b>	<b>510</b>	<b>270</b>	<b>1800</b>

Figure 16 – Thèmes d'enseignement sur les quatre semestres

Chaque semestre est découpé en Unité d'Enseignements, elles-mêmes divisées en modules. Pour ma séquence, j'ai retenu le module M2103 sur l'informatique embarqués. Il se place au semestre 2 et son volume horaire est précisée en figure 17. Ce module est placé au sein de l'Unité d'Enseignement *Composants, systèmes et applications - Développement*. Elle se situe au deuxième semestre.

Référence module	Nom module	Coef.	CM	TD	TP	Volume horaire Etudiant en formation encadrée	dirigée
<b>Semestre 2</b>							
<b>UE21 Composants, systèmes et applications – Développement</b>							
M 2101	Energie	3	16	24	20	60	
M 2102	Automatisme	3	12	20	28	60	
M 2103	Informatique embarquée	3	12	20	28	60	
M 2104	Systèmes électroniques	3	15	24	21	60	
<b>Total UE21</b>		<b>12</b>	<b>55</b>	<b>88</b>	<b>97</b>	<b>240</b>	

Figure 17 – Placement du module dans l'Unité d'Enseignement UE21.

#### 4.1.3 Module *Informatique Embarquée*

Le PPN fournit une fiche descriptive des contenus pour chaque module. Le module intitulé *Informatique Embarquée*, numéroté M2103, est présenté en figure 18. Les pré-requis pour aborder ce module sont au premier semestre, et sont résumés en table 3. Principalement, j'ai besoin que les étudiants soient à l'aise en programmation séquentielle courante, sur la séparation du code en fonctions, et sur les opérations binaires allant de pair avec la représentation des nombres en machine.

M1102 : Informatique	Variables, boucles, conditions Programmation procédurale Fonctions et découpage en modules
M1103 : Systèmes d'Information Numériques	Opérations combinatoires Algèbre de Boole Représentation des nombres

Table 3 – Pré-requis à la séquence pédagogique.

Sa position dans le semestre est relativement libre. Je la placerais vers le mois de février.

#### 4.1.4 Contenu de la séquence

Dans ce module, je vais axer ma séquence pédagogique sur la mise au point d'un système domotique de mesure de température. Il s'agit d'un système relativement similaire à celui que j'ai réalisé pour le compte d'Orange Labs, avec cependant une partie capteur alternative.

L'idée est d'utiliser une carte type NodeMCU comprenant l'ESP8266 pour acquérir, sur pression d'un bouton, les données d'un capteur de température type LM35<sup>1</sup> délivrant une tension proportionnelle à la température. Cette valeur est lue par le CAN de l'ESP8266. Il s'agit ensuite d'afficher cette valeur sur une page web. Pour s'y connecter, on affichera l'adresse IP sur un écran LCD relié en I<sup>2</sup>C.

C'est un support dont le coût reste modéré, avec un prix total inférieur à 10 €. Le détail est présenté en table 4. Les prix sont indicatifs et ont été relevés chez les boutiques internet de revendeurs de matériel électronique.

Composant	Prix
Carte avec ESP8266	4 €
Ecran LCD 16x2 caractères	2 €
Contrôleur I <sup>2</sup> C pour l'écran LCD	1 €
Capteur de température LM35	1,50 €
Total	8,50 €

Table 4 – Prix moyens, constatés sur les boutiques internet des revendeurs.

Les compétences travaillées dans cette séquence portent sur les points suivants :

- Architecture d'un microcontrôleur
- Interfaces et périphériques : I<sup>2</sup>C, port série, CAN
- Manipulation d'interruptions : minuterie, GPIO
- Utilisation d'un outil de mise au point

Ce sont ces mêmes points qui sont encadrés en rouge sur la figure 18. Ils sont directement liés aux objectifs du module, encadrés en bleu, et qui sont tous travaillés lors de cette séquence.

1. <http://www.ti.com/lit/ds/symlink/lm35.pdf>

Référence de l'UE <b>UE21</b>	Nom de l'UE Composants, systèmes et applications  Matière : <b>Informatique industrielle</b>	Volume Horaire <b>60h (12CM, 20TD, 28TP)</b>
Référence du module <b>M 2103 (Info2)</b>	Module <b>Informatique embarquée</b>	Semestre <b>S2</b>
<b>Objectifs du module :</b> Comprendre l'architecture d'un système à microcontrôleur. Maîtriser l'utilisation des périphériques d'un microcontrôleur. Savoir modéliser une application embarquée. Comprendre les mécanismes d'interruption.		
<b>Compétences visées :</b> Développer une application en langage évolué pour une cible à microcontrôleur, Gérer les périphériques d'entrées – sorties pour s'interfacer avec un environnement, Mettre en œuvre le mécanisme de fonctionnement en régime d'interruption de programme, Utiliser un outil de développement croisé		
<b>Pré-requis :</b> Modules M 1103 (Info1 : Informatique) et M 1102 (SIN1 : Système d'information numérique).		
<b>Contenus :</b> Démarche d'élaboration d'une application informatique embarquée :  <ul style="list-style-type: none"> <li>- Compréhension de l'architecture matérielle de la cible.</li> <li>- Compréhension des fonctions de gestion des périphériques types (entrées/sorties TOR, convertisseurs analogique numérique et numérique analogique, timer, communication série, PWM...),</li> <li>- Analyse d'un cahier des charges,</li> <li>- Identification des ressources matérielles nécessaires et des mécanismes de leur mise en œuvre (scrutation ou interruption),</li> <li>- Modélisation de l'application,</li> <li>- Codage dans un langage évolué,</li> <li>- Utilisation d'une méthode de validation prédéfinie,</li> <li>- Utilisation avec un outil de mise au point (type débuggeur),</li> <li>- Documentation des fichiers sources.</li> </ul>		
<b>Modalités de mise en œuvre :</b> Programmation de périphériques, Implémentation de graphes d'états, Mise en œuvre des interruptions. La mise en œuvre des applications en TP est souhaitable via un environnement de développement sur machine hôte, pour toute la richesse dans les techniques de mise au point qu'il apporte (développement en simulation, au travers d'un débuggeur via le téléchargement sur cible).		
<b>Prolongements possibles :</b> Tous les modules complémentaires d'informatique industrielle (semestres 3 ou 4), Module M 3103 (Res3 : Réseaux), Tous les modules d'ER (M3203, M4203) et de projets tutorés (M3207, M4207) des semestres 3 et 4.		
<b>Mots clés :</b> Microcontrôleur, périphériques, architecture matérielle, registres, interruptions, développement croisé.		

Figure 18 – Fiche du module *Informatique Embarquée* du PPN GEII.

#### 4.1.5 Programmation du module ESP8266

Le PPN propose l'utilisation d'un "langage de programmation évolué". Il me semble intéressant de proposer une programmation de l'ESP8266 en Lua. C'est un langage moderne, dont la réalité industrielle s'affirme au travers de logiciels comme Adobe Lightroom, le micrologiciel pour routeurs OpenWrt, ou les scripts du programme Domoticz. Il est également utilisé pour la programmation des calculatrices de la gamme Nspire de Texas Instrument. La déclinaison embarquée du Lua (Embedded Lua ou eLua) est disponible sur une grande liste de microcontrôleurs<sup>1</sup>. Elle permet une programmation évènementielle apte à mettre en évidence le concept d'interruption. De plus, l'utilisation du Lua comme interpréteur permet l'exécution d'un programme pas-à-pas en envoyant chaque ligne par le port série.

Si toutefois l'utilisation du Lua posait problème lors de la séquence, il est envisageable de coder l'ESP8266 dans un langage proche du C, à l'aide de l'environnement de développement Arduino. Il faudrait alors lui rajouter les bibliothèques, le compilateur et le téléverseur associés à l'ESP8266, et cette dernière se comporterait comme une carte Arduino augmentée de fonctions Wi-Fi. Cependant, on perdrat les capacités d'interprétation et les messages d'erreur utiles pour la résolution de problèmes.

#### 4.1.6 Volume horaire attribué

Cette séquence va permettre de lancer le module d'Informatique Embarqué. Afin de développer les compétences précédemment mentionnées, j'attribue :

- 6 heures en promotion entière pour les cours et l'évaluation,
- 6 heures de TD en classe de 24 élèves,
- 12 heures de TP en demi-groupe.

Le reste des heures servira pour l'approfondissement bas niveau des notions vues pendant cette séquence, notamment l'utilisation des registres. Ces concepts seront plus facilement abordables une fois que les étudiants les auront manipulés d'un point de vue utilisation, avant de rentrer dans leur réalisation matérielle.

#### 4.1.7 Déroulement de séquence

La figure 19 présente le plan de déroulement de la séquence.

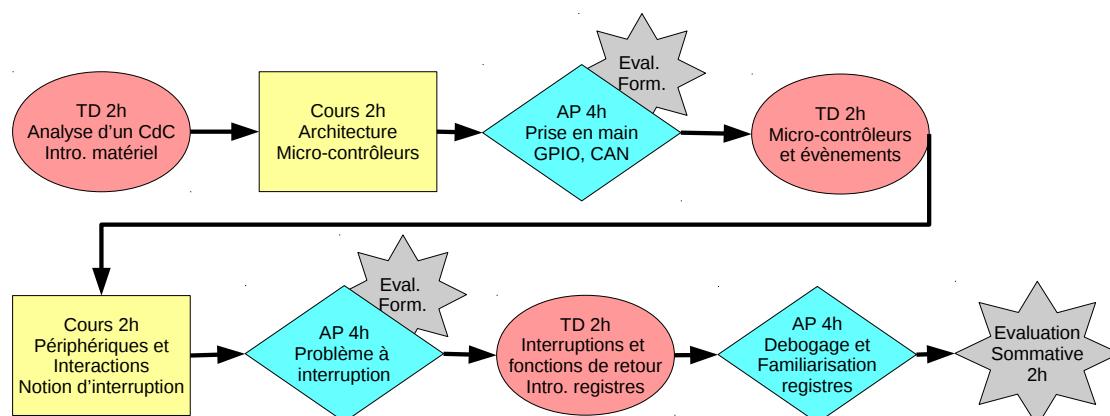


Figure 19 – Découpage de la séquence pédagogique.

1. <http://wiki.eluaproject.net:80/Boards>

#### 4.1.8 Détail des séances

##### 1) TD 2h : Analyse d'un cahier des charges

La séance proposée sert d'introduction à la séquence en présence d'une classe de 24 élèves. Elle est l'occasion d'introduire les besoins auxquels vont répondre les microcontrôleurs.

Le client veut mesurer les températures de différentes chambres contrôlées et reporter ces valeurs sur un système de supervision à distance. Pour cela, il aurait besoin d'une nuée de capteurs connectés, commandables, intégrés dans un réseau assimilable à celui d'une usine. Cependant, il ne peut pas câbler l'intérieur des chambres à température contrôlée pour des raisons d'isolation thermique et de coûts engendrés. Le système de mesure devrait donc avoir des capacités de communication sans fil, et être autonome en énergie.

Au fil des questions-réponses avec le client dont je joue le rôle, les élèves dressent ensemble au tableau une liste des fonctions devant apparaître dans un cahier des charges. Je fournis ensuite le cahier des charges fonctionnel minimal visible en table 5 dans lequel on devrait retrouver la plupart de leurs propositions.

Référence	Fonction	Critères	Niveaux	Flexibilité
FP1	Mesurer la température	Précision, Plage	$\pm 1^\circ\text{C}$ , -40°C à 85°C	F0
FP2	Transmettre l'information	Liaison, Fréquence d'envoi	Wi-Fi, 1 Hz, sur une page web	F1
FC1	Afficher des informations sur place	Ecran	16x2 caractères, LCD	F2
FC2	Etre autonome	Durée	1 mois	F1
FC3	S'intégrer à l'environnement	Volume	<20cm³	F1

Table 5 – Cahier des charges fonctionnel du capteur de température connecté.

Après un point sur les stratégies mises en place pour avoir une bonne autonomie, devrait émerger l'idée d'utiliser un microcontrôleur. Vu le passé des étudiants en S-SI et STI2D, il est probable que certains aient cette expérience, dans ce cas, ils peuvent essayer d'expliquer ce que c'est aux autres étudiants. Dans le cas où les étudiants n'y pensent pas, je l'introduis. On peut alors discuter des idées qu'ils ont d'un microcontrôleur, à quoi cela ressemble, qu'est-ce que cela inclut. Cette séance, servant d'évaluation diagnostique, servira à monter la séance suivante de cours pour apporter une connaissance structurée sur la question.

##### 2) Cours 2h : Architecture et composition d'un microcontrôleur

Ce cours regroupe toute la promotion de 120 élèves en amphithéâtre. Le cours présente quelques microcontrôleurs courants en situation, avec les définitions des termes et la présentation des composants liés. En prenant un exemple, il apporte une description de ce qu'on y trouve, par exemple :

- un microprocesseur comportant des unités de calculs et une mémoire interne servant à l'exécution,
- Une mémoire externe pour stocker un programme à exécuter,
- Des convertisseurs analogiques-numériques, numériques-analogiques
- Des broches d'entrée / sortie générales (*General Purpose Input/Output* ou *GPIO*),

- Des supports de communication (souvent port série, voire d'autres, utilisés avec des bibliothèques pré-écrites, ici au mieux mentionnés.)
- Parfois plus encore (par exemple des composants dédiés au Wi-Fi...)

Durant ce cours, je présente les notions de mémoire de plus ou moins grande taille, jusqu'aux *registres* présentés comme des mémoires internes servant à faire des calculs directs, connaître la position dans le programme, ou encore se renseigner sur l'état des broches.

Le cours mentionne également l'intérêt économique et environnemental d'utiliser un ensemble intégré à faible consommation pour une application ciblée.

Le support diaporama sera distribué en version lacunaire aux étudiants en début de séance. Il devra être rempli et annotée pendant le cours.

## 2) Activité Pratique n°1, 4h : prise en main d'un microcontrôleur

Cette activité pratique est réalisée par binômes (6 groupes). Chaque binôme a à sa disposition une carte NodeMCU sur lequel se trouve le microcontrôleur ESP8266, avec un ordinateur dédié à la programmation équipé de l'environnement de développement ESPlorer présenté précédemment.

L'objectif de cette séance est de réaliser un premier montage électrique avec la carte NodeMCU et de manipuler les fonctionnalités suivantes :

- Les messages sur le port série - USB
- Les broches d'entrées - sorties (GPIO)
- le CAN intégré à la carte sur la broche A0

Pour cela, les étudiants réalisent le montage en figure 20. L'alimentation du montage se fait par le port USB sur lequel est branché la carte NodeMCU, qui apporte l'énergie aux autres composants.

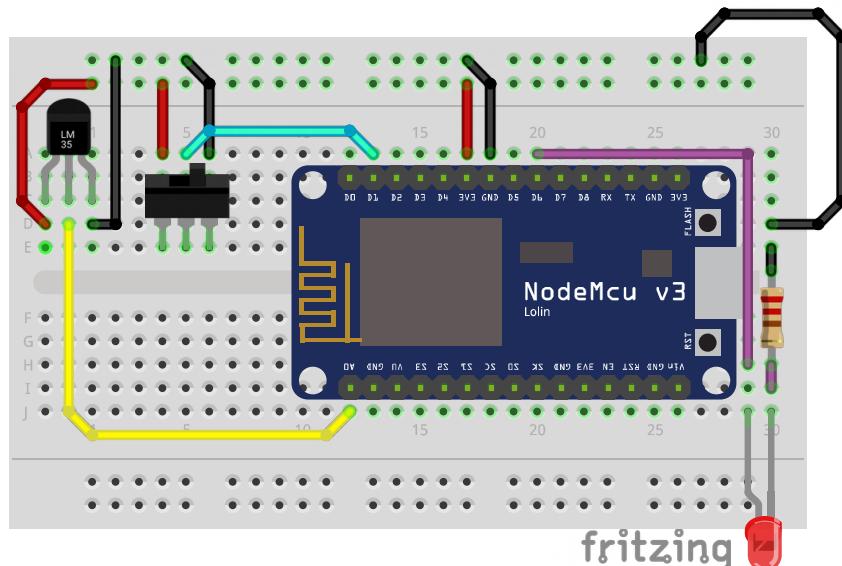


Figure 20 – Montage électrique, application pratique n°1.

Le schéma électrique équivalent est visible en figure 21.

Ensuite, ils prennent en main le code que je fournis, en Lua. Il contient les lignes documentées pour :

- Mettre une broche en entrée ou en sortie

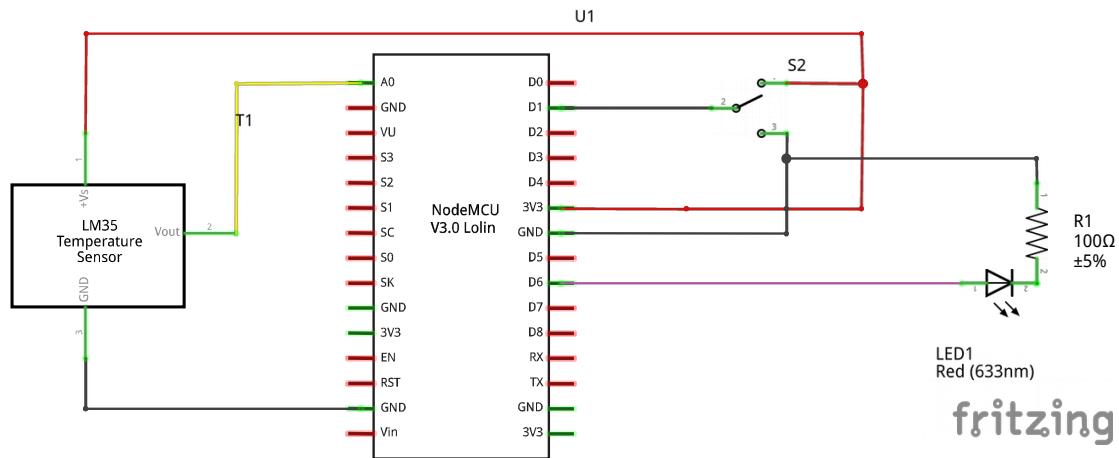


Figure 21 – Schéma électrique du montage, activité pratique n°1.

- Définir l'état d'une broche de sortie (1-0)
- Lire l'état d'une broche en entrée (1-0)
- Lire la valeur de la broche avec CAN

De plus, les structures classiques (boucles, conditions, définitions de variables) sont données aux étudiants. Elles sont relativement transparentes et ont une syntaxe simple.

L'objectif est de maîtriser les fonctions de ce microcontrôleur. Le support d'activité présente rapidement le Lua et son intérêt dans les microcontrôleurs modernes. Il met en valeur le fait qu'aujourd'hui, il existe des microprocesseurs assez performants pour exécuter un code à travers un interpréteur, et faciliter la réutilisation et la portabilité du code à travers différents supports. Comme souligné précédemment, l'interpréteur Lua est disponible sur beaucoup de microcontrôleurs très différents. L'activité permet de se familiariser avec les concepts du microcontrôleur sans s'embarrasser des particularités d'une référence spécifique.

Avancement	Travail demandé	Résultat attendu
Etape 1	Se connecter au microcontrôleur. Afficher le programme stocké dessus ( <code>init.lua</code> ). Que fait-il ?	Affiche un message à l'allumage, définit des pins en entrée et en sortie. Rien de plus.
Etape 2	Afficher un message avec <code>print()</code> . Essayez de changer la vitesse de transmission. Que constatez-vous ?	Message lisible pour une seule valeur. Besoin d'être synchronisé des deux côtés.
Etape 3	Comment allumer la LED ?	Ajouter : broche D6 à l'état 1
Etape 4	Comment allumer la LED seulement si l'interrupteur est à droite ?	Ajouter : lire la valeur de la broche où l'interrupteur se trouve, avec boucle
Note : point de vocabulaire donné : cette technique s'appelle <i>scrutation</i> (ou <i>polling</i> en anglais).		
Etape 5	Chercher dans la documentation les informations sur le CAN, sur le capteur.	CAN 10 bits : valeurs entre 0 et 1023, pleine échelle 3.3V. Capteur avec résolution de 10mV/°C.
Etape 5	Lire la valeur sur la broche avec CAN. L'afficher par le port série. Que voyez-vous ?	Une valeur du CAN.
Etape 6	Lorsque la valeur du CAN augmente de 1, de combien varie la tension ?	$3,3/1024=0,0032V$ soit environ 3,2mV.
On arrondit cette valeur à 3,3mV et on suppose pour simplifier que le capteur a une résolution de 9,9mV/°C. Donc augmenter de 1°C augmente la valeur lue de 3.		
Etape 7	Implémenter dans le code la conversion valeur du CAN vers °C.	Code Lua correspondant
Etape 8	Modifier votre programme pour afficher la température (une seule fois !) quand l'interrupteur est actionné.	Astuce : utiliser une variable pour stocker l'état de l'interrupteur avant de le lire, et comparer pour savoir s'il a changé. Code Lua correspondant.

Dans cette activité, un problème peut survenir autour du CAN, qui n'est pas un composant familier pour les étudiants. La démarche doit être la plus guidée possible pour leur introduire le CAN. Si un problème de langage émerge, la syntaxe d'un cas précis sera donnée et inscrite au tableau. Les broches entrées / sorties sont vues et manipulées tout au long de l'activité.

Cette activité pratique fait l'objet d'un compte rendu pour chaque groupe à rendre sous 24h sur une plate-forme en ligne de dépôt de documents.

#### 4) TD 2h : gestion des événements

A la suite de cette activité pratique, les étudiants auront été confrontés à diverses fonctions remplies par un microcontrôleur et à quelques périphériques. La seule méthode que les étudiants connaissent pour se renseigner sur le niveau d'une broche est la scrutation.

La séance commence par un résumé rapide de ce qui a été vu en activité pratique. Ensuite, suivant les notions mal assimilées, des exercices sont distribués, en particulier sur les CAN (qui sont un peu plus ardues que les broches Tout-Ou-Rien.). La partie exercices se terminera par un exemple de mise en place de scrutation.

J'annonce alors aux élèves que la scrutation est coûteuse en énergie puisqu'elle oblige le microprocesseur à attendre en exécutant du code en boucle. Le terme "*"attente active"*" est alors introduit (mais sera défini et gardé en trace écrite lors de la séance suivante de cours). Ce mode pose plusieurs soucis :

- Une consommation électrique accrue, le micro-processeur étant bloqué dans une boucle,
- L'impossibilité de mieux utiliser le processeur en faisant autre chose en attendant,
- Parfois le blocage des autres fonctionnalités qui devraient s'exécuter même quand on attend pour une fonctionnalité particulière.

Pour remplir le cahier des charges posé en première séance, cela pose problème car on rajoute une

difficulté sur l'autonomie du capteur connecté (puisque l'on consomme plus), et il est possible que la partie serveur web ne réponde plus lors de l'attente active. J'introduis alors la notion d'*interruptions* : les microcontrôleurs disposent de mémoires particulières (les registres, introduits lors du premier cours) leur permettant, suivant leur état, de détourner le flot d'exécution du programme. Ces mémoires voient leur état changer sur des événements. Donc, si au lieu d'écrire un programme d'un bloc, on écrit des morceaux à exécuter en fonction des événements, on devrait pouvoir faire mieux que de la scrutation. C'est ce que permettent les *interruptions*.

## 5) Cours 2h : périphériques, interactions, interruptions

Ce cours apporte les connaissances sur les interruptions : ce que c'est, comment les mettre en place. En utilisant des portions en pseudo-code dans le cours, puis quelques exemples en Lua, le but est de familiariser les étudiants avec un paradigme qu'ils n'ont pas l'habitude de rencontrer : la programmation événementielle.

Il permet l'apport des définitions d'interruption sur minuterie et sur niveau d'une broche (niveau haut, bas). Je présente aussi le lien avec le registre qui informe sur l'état (entrée, sortie) et le niveau.

D'autre part, le cours va introduire des manières de communiquer suivant les périphériques. En prenant appui sur les broches d'entrée / sortie génériques (GPIO), j'introduis la notion de mot à envoyer, puis de protocole de communication, indispensable pour permettre à deux appareils de communiquer. Je mentionne l'existence d'un bus, le bus I<sup>2</sup>C, adapté aux microcontrôleurs, disposant d'un fil de synchronisation (appelé "horloge", communément SCL), et d'un fil de données (communément SDA). Je pose ainsi les généralités permettant de rentrer plus dans le détail pour une séquence ultérieure. Ces généralités permettront d'observer un cas d'utilisation pour I<sup>2</sup>C lors de l'activité pratique suivante.

## 6) Activité pratique n°2, 4h : problème à interruptions

L'objectif est de manipuler les interruptions et d'écrire des fonctions de rappel.

Les étudiants commencent par réaliser le montage présenté en figure 22. Le contrôleur I<sup>2</sup>C et l'écran sont déjà soudés entre eux. Par rapport au montage précédent, la LED disparait, et un écran LCD est ajouté sur les broches D4 et D5. Les autres composants sont inchangés.

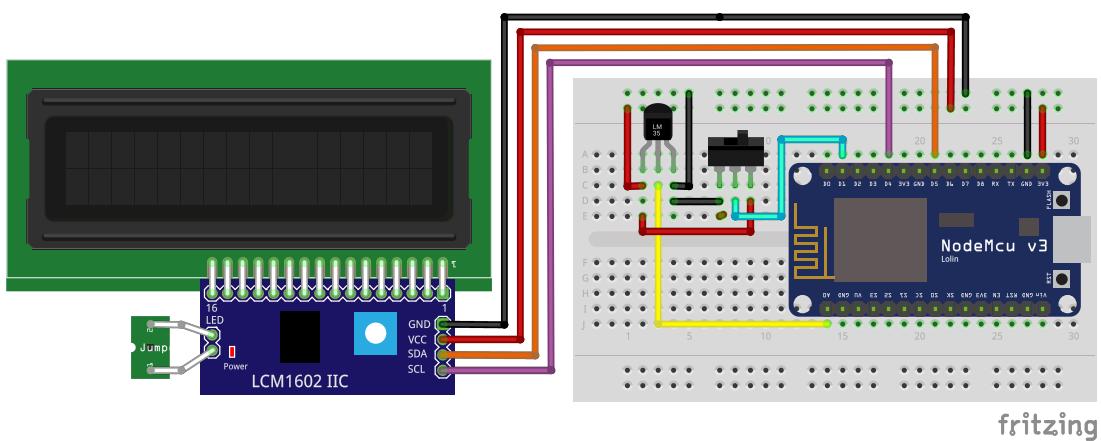


Figure 22 – Montage de l'activité pratique n°2 : interruption et cas d'utilisation d'un écran LCD.

A présent, la carte NodeMCU contient un programme générant un serveur web sur connexion TCP. Il est équivalent au serveur réalisé en partie technique, avec des commandes adaptées au cas présent. Les étudiants n'ont pas à le modifier.

La démarche d'activité pratique est résumée ci-après.

Travail demandé	Résultat attendu
Se connecter au microcontrôleur. Afficher le programme stocké dessus ( <code>init.lua</code> ). Que fait-il ?	Affiche un message à l'allumage, définit des pins en entrée et en sortie, et définit une connexion pour l'écran LCD, puis lance le script de serveur web.
Implémenter la lecture de température comme précédemment, avec affichage sur le port série. Le résultat doit être stocké dans la variable fournie (lue par le serveur web)	Code Lua correspondant. Voit si le CAN a été compris.
Lancer le script. Se connecter au serveur web (IP sur l'écran LCD). Que voyez-vous ?	Serveur ne répond pas : scrutation bloquante.
Note : ici on fournit la syntaxe pour écrire en Lua une fonction et pour exécuter la fonction sur interruption. On l'appelle <i>fonction de rappel</i> (en anglais <i>callback function</i> )	
Ecrire la fonction à appeler lors de l'interruption, et le code qui lie l'état de la broche avec la fonction. Que se passe-t-il maintenant ?	Serveur web accessible, on lit la température. Page web qui s'actualise toutes les 5 secondes.
On peut déclencher une interruption avec autre chose qu'une broche, par exemple une minuterie (dite <i>timer</i> ).	
Implémenter une interruption sur minuterie pour lire la température toutes les 2 secondes.	Code Lua correspondant.
Regarder comment est commandé l'écran au début du script faisant serveur web. Regarder ensuite le script bibliothèque LCD. Ajouter dans l'interruption- <i>timer</i> l'affichage de la température sur l'écran LCD	Affichage sur la 2e ligne de l'écran de la température

Les étudiants expérimentent l'usage d'interruptions et de fonctions de rappel (*callback functions*). Ils réutilisent les broches d'entrée-sortie, récupèrent l'application du CAN faite précédemment, et commencent à écrire des fonctions. La syntaxe est rappelée dans le cours précédent, ou au tableau s'il y a un blocage.

L'application pratique donne lieu à un compte-rendu comme précédemment, et qui me permet d'adapter le TD suivant.

## 7) TD 2h : interruptions et fonctions de rappel - introduction aux registres

Cette séance commence par un résumé des techniques et méthodes mises en place lors de l'activité pratique précédente. L'intérêt de l'interruption est bien mis en valeur. Pour asseoir les connaissances sur ces points, des exercices sur l'écriture de fonctions de rappel sont faits en classe.

Les étudiants ont réalisé un prototype de thermomètre connecté à l'AP précédente. Nous faisons ensemble un résumé des étapes de réalisation : commande des broches, lecture de la valeur du CAN, conversion en °C, scrutation d'un bouton, interruption sur bouton puis sur minuterie, affichage sur écran LCD.

J'explique que ces opérations ont été grandement simplifiées par l'interpréteur Lua pour rendre la programmation plus simple. Quand bien même il existe de plus en plus de microcontrôleurs assez bien dotés en mémoire et en puissance de calcul pour utiliser un interpréteur Lua, tous n'en sont pas capables.

Finalement, le minimum reste d'avoir 1 bit pour définir l'état d'une broche, donc un mot de 8 bits pour les états de 8 broches. Un autre exemple possible est par exemple un mot de 16 bits qu'on incrémente pour faire un compteur, une minuterie. Les microcontrôleurs utilisent des registres parce que cela permet d'avoir la fonctionnalité voulue en dépensant le moins possible : en complexité, en temps, en ressources, et en argent.

C'est pour cela que la prochaine activité pratique aura pour but de se familiariser avec quelques registres sur un microcontrôleur plus modeste, par exemple le Freescale 9S12.

## 7) Activité pratique n°3, 4h : débogage et manipulation de registres

Cette séance permet aux étudiants de manipuler la notion de registre à travers un microcontrôleur et son environnement de développement dédié. Ici, le choix s'est porté sur le Freescale 9S12, mais rien n'empêche de réaliser la même activité pratique sur un microcontrôleur concurrent (par exemple, avec un représentant de la famille PIC16 ou MSP430), suivant le matériel déjà disponible.

Le microcontrôleur sera monté dans une carte comprenant interrupteurs et diodes sur différentes broches, et fourni avec des programmes écrits en C. C'est un langage connu des étudiants suite au module d'informatique M1103 fait au premier semestre.

Les étapes de l'activité seraient les suivantes :

- Prendre en main l'environnement de développement.
- Lire dans le programme l'état du port contenant les broches. Retrouver cet état dans l'afficheur de registres du logiciel.
- Utiliser l'exécution du programme pas-à-pas. Observer le changement du registre du port concerné (par exemple PORTB pour les GPIO associées).
- Dé-commenter (rajouter dans le code à exécuter) une partie pour rajouter l'interruption. Lancer le programme pas-à-pas, et activez l'interruption sur la broche. Que se passe-t-il au niveau du registre correspondant ?
- L'interruption est signalée par un bit. Que se passerait-il si on demandait une deuxième interruption, et une troisième pendant qu'on est en train de répondre à la première ? (on dit que l'interruption est *en attente*, ou "*pending*")
- Selon vous, quelle serait "une bonne manière" d'écrire une fonction de rappel d'interruption, dans ce cas ? (la plus petite et rapide possible)

Exploration de nouveaux registres :

- Lancer le programme "essai calculs" qui réalise une somme de deux entiers. Exécuter le programme pas-à-pas, observer les registres indiqués (par exemple A). Où retrouvez-vous les nombres sommés ?
- Indiquer sur le schéma-blocs du micro-processeur le cheminement des opérandes du calcul.
- Changer le contenu du registre indiqué. Qu'observez-vous ? (le calcul demandé est exécuté mais avec le nouveau nombre rentré)

La manipulation et l'observation des registres permettra aux étudiants d'appréhender pour la séquence suivante d'autres registres plus délicats comme le pointeur d'instruction et le pointeur de pile.

## 9) Evaluation sommative : 2h

Cette évaluation clôt la séquence avec une interrogation sur table de 2 heures. Elle s'assure que les compétences ont bien été acquises :

- Connaissance des définitions : microcontrôleur, GPIO, CAN, registres, interruption, minuterie (*timer*), attente active, fonction de rappel.
- Calcul de résolution d'un CAN, choix d'un capteur parmi une liste
- Rédaction d'une interruption en Lua (syntaxe donnée) : définition de l'interruption, écriture de la fonction de rappel
- QCM sur les registres : nature, utilité, exemple de registre, à quoi sert le registre PORTB par exemple ?

### 4.1.9 Conclusion de la séquence pédagogique

Cette séquence d'introduction aux microcontrôleurs part d'une problématique industrielle simple et progresse des concepts jusqu'à leur mise en place matérielle. Elle permet aux étudiants de comprendre

les intérêts et possibilités offertes par les microcontrôleurs, mais aussi leurs limites. Cela oriente la prochaine séquence directement sur les concepts bas niveaux tels que la gestion des instructions et de la pile. Ces notions seront abordées avec des activités pratiques autour du langage Assembleur.

Ces points seront moins évidents à traiter avec l'ESP8266 qui garde un haut niveau d'abstraction. Il pourra cependant être réutilisé pour le module M3207 sur la supervision. C'est un bon candidat pour ce genre de tâches, et il permettra de mettre en pratique la compétence "Créer un server web embarqué" mentionnée dans le PPN.

## 4.2 Séquence pédagogique en STI2D, Enseignements Technique Transversal

### 4.2.1 La filière STI2D

La filière de STI2D désigne des classes de 1<sup>re</sup>) et de Terminale, et elle est axée autour des technologies et de leur intégration dans la société. Quatre spécialités sont proposées :

- ITEC : Innovation, Technologie et Eco-Conception,
- SIN : Systèmes d'Information et Numériques,
- EE : Energie et Environnement,
- AC : Architecture et Constructions.

Cette séquence est en Enseignement Technique Transversal (ETT). Tous les élèves de STI2D la suivent, peu importe leur option.

Les enseignements de STI2D sont articulés autour du triptyque Matière-Energie-Information.

Les élèves de STI2D, à la suite du baccalauréat, continuent pour la plupart leurs études vers les filières IUT et STS et en Université.

### 4.2.2 Thème de séquence et placement

Cette séquence pédagogique concerne l'ensemble de la classe et a pour thème le compromis coût-efficacité-complexité. Elle nécessite d'avoir été initié aux concepts autour du développement durable, ce qui nécessite les pré-requis indiqués dans la fiche de séquence page suivante. D'autre part, elle représente un point central où se croisent les trois piliers du développement durable : économie, environnement et société. Il est intéressant que les élèves puissent s'y référer lors du choix de leur projet en Terminale. C'est pour cela que je place la séquence en milieu de 1<sup>re</sup>.

Cette séquence a été conçue pour une classe de 32 élèves.

### 4.2.3 Compétences travaillées

Les niveaux taxonomiques mentionnés en table 6 sont ceux des connaissances dans les compétences relevées. Il y a plusieurs compétences derrière celles présentées ici, mais leurs niveaux taxonomiques sont identiques entre elles.

n° référentiel	Compétence	Taxonomie (début->fin/max)
1.1.3	Paramètres de compétitivité	0→2/2
1.2.3	Utilisation raisonnée des ressources	0→2/2
3.1.4	Traitement de l'information	2→3/3

Table 6 – Compétences travaillées dans la séquence STI2D-ETT.

#### 4.2.4 Détails de la séquence

Pour la concevoir, j'ai utilisé le logiciel PySéquence<sup>1</sup>. Il permet de générer une fiche séquence pour les professeurs de Sciences de l'Ingénieur. L'intégralité de cette fiche est disponible en annexe.

L'en-tête de la fiche PySéquence en figure 23 résume les centres d'intérêts, prérequis et objectifs de la séquence. Pour parler de consommation énergétique, il est nécessaire que les élèves aient vu en Physique les notions d'énergie et puissance. Pour les enseignements technologiques transversaux, les premières notions de développement durable devront être vues et comprises, en particulier les paramètres de compétitivité et le cycle de vie des systèmes, avec la notion d'Analyse du Cycle de Vie (ACV).

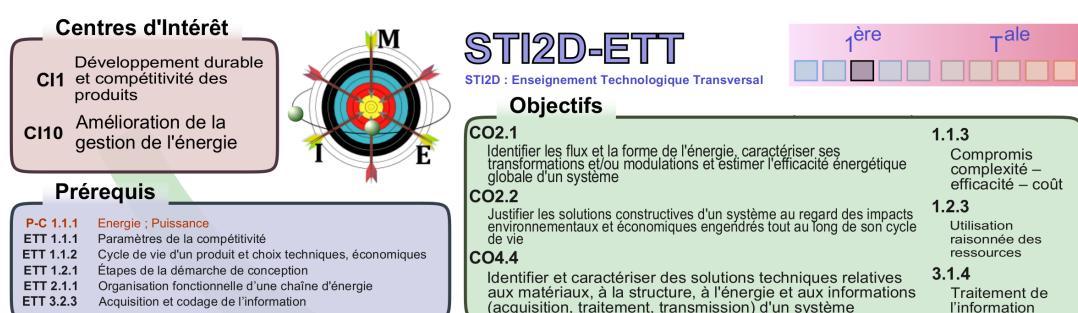


Figure 23 – Présentation des contenus et pré-requis de la séquence.

Le découpage en séance est donné en figure 24 . Il est suivi du détail de ces séances page suivante.

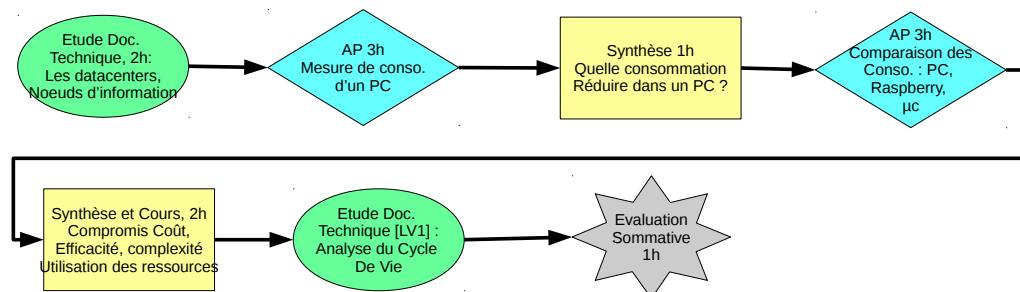


Figure 24 – Déroulement de la séquence Développement Durable en STI2D.

1. Logiciel dédié à la construction de séquences pédagogiques en SII. [http://eduscol.education.fr/sti/ressources\\_pedagogiques/pysequence-aide-lelaboration-de-sequences-pedagogiques-et-de-suivie-de](http://eduscol.education.fr/sti/ressources_pedagogiques/pysequence-aide-lelaboration-de-sequences-pedagogiques-et-de-suivie-de)

<b>1) Etude de dossier technique sur les datacenters - 1/2 classe, 2h</b>	<ul style="list-style-type: none"> <li>— Rappels sur l'organisation des composants d'un ordinateur</li> <li>— Description d'un datacenter : ensemble de serveurs</li> <li>— Serveur = composants d'un PC, dans une forme empilable. Pas de différence fondamentale avec PC normal.</li> <li>— Documents sur la norme PUE<sup>2</sup> : introduction à la notion de rendement</li> <li>— Questionnements sur les limites du PUE : efficacité du traitement par les serveurs ?</li> </ul>
<b>2) Activité Pratique 1 : Mesure de consommation de composants - 4 binômes, 3h</b>	<ul style="list-style-type: none"> <li>— Mesure avec pince ampèremétrique (à emprunter du laboratoire de physique) sur fils à tension constante d'un PC de bureau</li> <li>— 3 cas de mesure : au repos, en calcul processeur, en calcul carte graphique</li> <li>— Quelle est la plus grande puissance et dans quel cas ?</li> <li>— Chronométrer la tâche donnée, calculer l'énergie consommée (<math>E = P * t</math>),</li> <li>— Sachant que <math>1 \text{ kW} * 1\text{h} = 0,15\text{\euro}</math>, donner le coût en <math>\text{\euro}</math> pour accomplir cette tâche</li> </ul>
<b>3) Synthèse de l'activité menée - classe entière, 1h</b>	<ul style="list-style-type: none"> <li>— Présentation des résultats par un groupe</li> <li>— Comment réduire la consommation énergétique d'une tâche ?</li> <li>— Propositions attendues : rendre les tâches plus rapides ? rendre les tâches moins demandante en puissance ? Diviser les tâches ? Utiliser un matériel adapté ?</li> </ul>
<b>4) Activité pratique 2 : comparaison des consommation - 4 binômes, 3h</b>	<ul style="list-style-type: none"> <li>— Mesure puissance, temps de traitement : PC, Raspberry Pi, microcontrôleur</li> <li>— Lequel consomme le plus ? Est-ce surprenant ? Dans l'industrie, quel genre de machine est utilisé ?</li> <li>— Combien de microcontrôleur pour faire aussi vite que le PC (si on divise) ?</li> <li>— Code non optimal : le changer. Différence ? (temps d'exécution)</li> <li>— Lire résultats d'ACV, combien d'énergie pour produire 1 PC ? N microcontrôleurs ?</li> </ul>
<b>5) Synthèse - cours : compromis coût-complexité-éficacité - classe entière, 2h</b>	<ul style="list-style-type: none"> <li>— Quelles ressources sont mis en jeu ? (énergétique, matérielles, humaines)</li> <li>— Quels coûts ? (Financiers, écologiques, sociaux)</li> <li>— Quelle complexité ? (Besoin d'experts, temps)</li> <li>— Rappel de l'ACV (notion vue précédemment), et de la place qu'elle prend dans le choix d'une solution</li> </ul>
<b>6) Etude de documents techniques : analyses de cycle de vie en LV1 - 1/2 classe, 1h</b>	<ul style="list-style-type: none"> <li>— Confrontation à des ACV, résolution de problème de coûts, choix de complexité d'une solution.</li> <li>— Cette séance permet une remédiation pour les points mal compris en AP.</li> </ul>
<b>7) Evaluation sommative - classe entière, 1h</b>	<ul style="list-style-type: none"> <li>— Définitions : énergie grise, acronyme ACV, rendement (vu avec PUE), lien puissance-temps-énergie</li> <li>— Analyse d'impact fournie, quelques questions autour</li> <li>— Pour une installation informatique, quels sont les leviers d'action pour consommer moins d'énergie (à service égal) ? <ul style="list-style-type: none"> <li>(Optimisation du code, réduction du nombre de machines simultanées si elles sont assez performantes, parallélisation de la tâche sur plusieurs machines si elles sont assez sobres, utilisation de matériel spécifique type carte graphique)</li> </ul> </li> </ul>

2. Power Usage Effectiveness, qui traduit le rendement d'un datacenter comme étant le rapport de l'énergie consommée pour les serveurs sur l'énergie totale utilisée (comprenant refroidissements, lumières...)

## **4.3 Séquence pédagogique en Classe Préparatoire aux Grandes Écoles - Informatique**

### **4.3.1 La nature du programme commun Informatique**

Les classes préparatoires aux grandes écoles (CPGE) scientifiques sont des filières en deux ans après le baccalauréat. Les élèves y rentrant, majoritairement de baccalauréat S, se forment aux problématiques de l'ingénieur pendant deux ans. Ils poursuivent pour la plupart cette formation en intégrant une école d'ingénieur, mais peuvent se réorienter par exemple vers l'Université.

Aujourd'hui, toutes les CPGE scientifiques bénéficient d'un enseignement commun de l'informatique. Le but est d'initier les étudiants aux algorithmes, aux structures de données, et à l'écriture de programmes pour les confronter à la réalité informatique du monde d'aujourd'hui. Cet enseignement scientifique est pluridisciplinaire et est mené conjointement par les professeurs de sciences (Mathématiques, Physique-Chimie et Sciences de l'Ingénieur).

### **4.3.2 Thème de séquence et savoirs abordés**

L'objectif de la séquence est d'appréhender l'utilisation de fichiers en programmation ainsi que les structures de données inhérentes à leur manipulation. L'exploitation de fichiers permet la conception d'algorithmes simples pour lier algorithmique et programmation. Les centres d'intérêts de cette séquence sont présentées en table 7

<b>Imaginer et concevoir</b>	une solution algorithmique modulaire utilisant des méthodes de programmation et des structures de données adaptées
<b>Traduire</b>	un algorithme dans un langage de programmation moderne et généraliste

Table 7 – Centres d'intérêts de la séquence en Informatique en CPGE.

Les élèves pourront ainsi manipuler une structure de données type tableaux à deux dimensions, pour réaliser des opérations basiques sur des fichiers images type Bitmap (sans compression particulière).

De manière plus détaillée, les compétences travaillées du référentiel abordés sont encadrés en rouge dans la figure 11, avec leur savoir associé encadré en bleu.

### **4.3.3 Pré-requis et placement**

Pour aborder sereinement les notions de fichiers et d'algorithmes pour les manipuler, il est nécessaire d'avoir travaillé avant les représentations des nombres en informatique ainsi que les structures classiques de programmation (variables, boucles, conditions, fonctions) et leur écriture avec le langage utilisé en CPGE, Python.

J'aurais aussi besoin que les élèves aient été introduits aux structures indexées comme les tableaux, typiquement dans la séquence précédente. Cette séquence permettrait de voir l'utilisation des fichiers ainsi que de la structure tableau 2D utilisée pour les images.

Pour ce faire, je place la séquence en fin du premier semestre, par exemple vers la fin de novembre de la première année de CPGE.

<p><b>Manipulation de quelques structures de données</b> : chaînes de caractères (création, accès à un caractère, concaténation), listes (création, ajout d'un élément, suppression d'un élément, accès à un élément, extraction d'une partie de liste), tableaux à une ou plusieurs dimensions.</p>	<p>On met en évidence le fait que certaines opérations d'apparence simple cachent un important travail pour le processeur.</p> <p>On met à profit la structure de tableau d'entiers à deux dimensions pour introduire la notion d'image ponctuelle (« bitmap »). Les algorithmes de traitement d'image seront abordés plus tard.</p>
<p><b>Fichiers</b> : notion de chemin d'accès, lecture et écriture de données numériques ou de type chaîne de caractères depuis ou vers un fichier.</p>	<p>On encourage l'utilisation de fichiers en tant que supports de données ou de résultats avant divers traitements, par exemple graphiques. L'utilisation de bases de données sera étudiée plus tard.</p>

Figure 25 – Savoirs et compétences travaillés dans la séquence.

#### 4.3.4 Déroulement de séquence

L'enseignement d'informatique dispose de 2 heures par semaine, la plupart du temps consécutives. Le déroulement proposé est celui en figure 26. Les 10 heures de cours nécessitent donc 5 semaines, qui peuvent être les quatre dernières semaines avant les vacances d'hiver.

Lors des séances d'informatique, la classe serait divisée en trois tiers. Si on suppose un nombre moyen d'étudiants dans une classe préparatoire à 42, on a alors trois groupes de 14 étudiants.

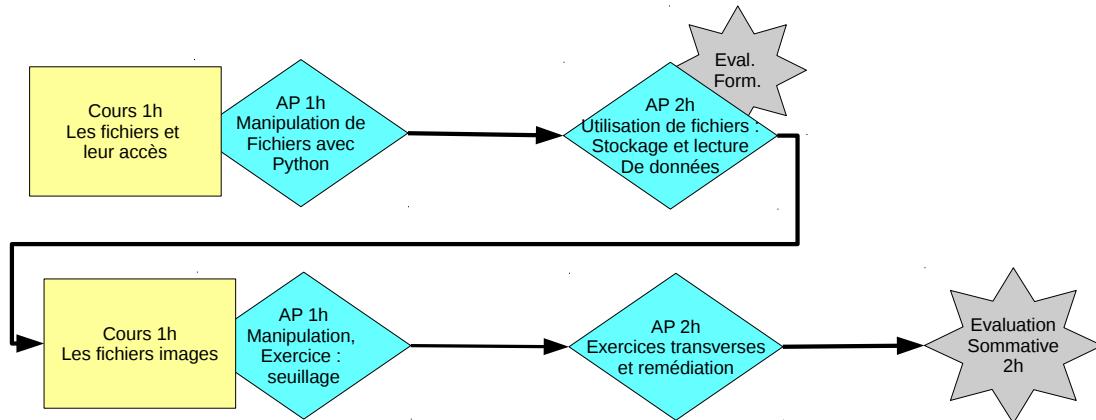


Figure 26 – Déroulement de la séquence en CPGE.

Une présentation séance par séance est proposée page suivante.

<b>1) Représentation des fichiers - 1h</b>	<ul style="list-style-type: none"> <li>— Pourquoi des fichiers : stocker durablement</li> <li>— Définition : chemin relatif / absolu, type de fichier, extension</li> <li>— Modes d'ouverture : <i>Read, Write, Add</i></li> <li>— Permissions : <i>Read, Write, Execute</i></li> </ul>
<b>2) Activité Pratique 1 : Python et les fichiers - 1h</b>	<ul style="list-style-type: none"> <li>— Ouverture, fermeture de fichiers</li> <li>— Lire et écrire avec différents modes d'ouverture</li> <li>— Manipuler des chemins relatifs, absous</li> <li>— Voir les permissions avec Linux (<code>ls -l</code>)</li> </ul>
<b>3) Activité pratique 2 : utilisation de fichiers - 2h</b>	<ul style="list-style-type: none"> <li>— Appréhender les fonctions de lecture, écriture, ouverture, fermeture de fichiers</li> <li>— Créer un programme simple qui affiche un fichier lu, puis qui ajoute une ligne à la fin</li> <li>— Ajouter la lecture d'un 2e fichier de configuration, avec traitement du contenu pour modifier le comportement du programme (lire des chaînes de caractère et modifier une variable en conséquence)</li> </ul>
<b>4) Cours : la représentation des images - 1h</b>	<ul style="list-style-type: none"> <li>— Composition d'une image : pixels = valeurs numériques pour définir couleur, luminosité...</li> <li>— Exemple d'encodage : niveaux de gris [0,255]</li> <li>— Accéder aux pixels : tableau 2D, comme une feuille de calcul d'un tableur</li> <li>— Besoin d'une structure de donnée indexée : accéder à un pixel par sa position</li> <li>— On peut évoquer rapidement des couleurs (encodage RVB par exemple)...</li> </ul>
<b>5) Exercice sur une image : seuillage</b>	<ul style="list-style-type: none"> <li>— Utiliser <code>PIL.Image</code> pour charger une image donnée (documentation fournie).</li> <li>— Essayer les fonctions données pour acquérir la valeur d'un pixel, stocker la valeur dans un pixel, et afficher l'image.</li> <li>— Opérations sur les pixels en parcourant toute l'image</li> <li>— Réalisation d'un seuillage (principe expliqué, algorithme à écrire en Python)</li> </ul>
<b>6) Exercices transverses, 2h</b>	<ul style="list-style-type: none"> <li>— Autres exercices sur les images : <ul style="list-style-type: none"> <li>• Haute valeur = plus lumineux. Multiplier la valeur de chaque pixel par 2, en restant en dessous de 255. Afficher l'image.</li> <li>• Stocker dans chaque pixel la valeur (pixel de gauche - pixel de droite). Exclure les bords. Afficher l'image. Quelle caractéristique est mise en valeur ? (contours verticaux)</li> </ul> </li> <li>— Autres exercices sur les fichiers : <ul style="list-style-type: none"> <li>• Ouvrir, lire et écrire dans un fichier simple</li> <li>• On donne un jeu en Python, créer un fichier où stocker les 10 meilleurs scores et les afficher. Lorsqu'un joueur termine sa partie, son score doit être comparé à ceux du fichier, placé au bon endroit, puis on doit supprimer le score le plus mauvais (quitte à supprimer le score du joueur).</li> </ul> </li> </ul>

<b>7) Evaluation sommative - classe entière, 1h</b>	<p>Vérifier que les élèves savent :</p> <ul style="list-style-type: none"> <li>— accéder aux éléments d'un tableau 2D type image</li> <li>— ouvrir un fichier avec un mode adéquat pour son utilisation (<i>Read, Write, Add</i>)</li> <li>— quelles sont les étapes de traitement d'un fichier : ouverture, lecture/écriture, fermeture</li> <li>— qu'il existe des permissions d'accès aux fichiers (on ne peut pas toujours tout faire avec n'importe quel fichier)</li> <li>— écrire le code en Python pour réaliser les opérations énumérées.</li> </ul>
---	---

## Conclusion

Pour ce dossier technique, j'ai travaillé en partenariat avec Orange Labs Lannion sur un projet de comparaison de langages de programmation d'un point de vue énergétique. Pour cela, j'ai conçu des programmes similaires dans leur fonctionnement, mais différents par leur langage, pour mesurer la consommation d'une machine arbitraire qui les exécute.

D'autre part, j'ai réalisé un système sous forme d'objet connecté pour effectuer des mesures de courant à distance de manière simple. Il permet l'accès aux mesures et est commandable par Wi-Fi au sein d'un navigateur web standard. Son prix réduit permet d'envisager son utilisation en grand nombre pour surveiller beaucoup de machines.

Ce projet scientifique et technique m'a permis de travailler sur un large spectre de notions adaptables à plusieurs formation de l'enseignement secondaire et supérieur. Je me suis appuyé sur mes développements et mes réalisations pour construire des séquences pédagogique en IUT GEII, en STI2D et en CPGE.

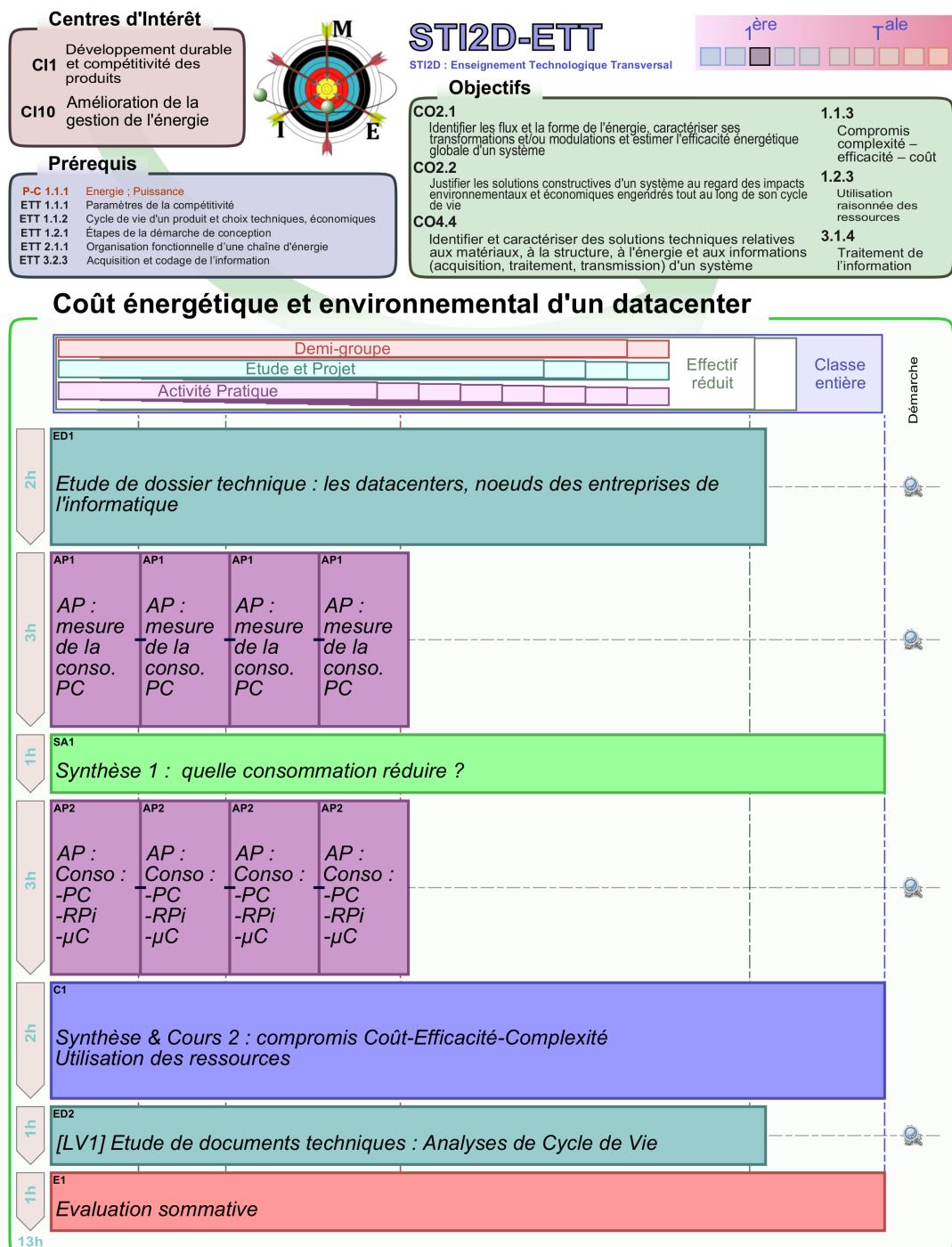
J'ai utilisé ce système de mesure en l'adaptant dans une séquence pédagogique en IUT GEII, autour des microcontrôleurs et de la programmation événementielle. Cette dernière est introduite progressivement, en partant du concept et de l'écriture haut niveau, pour ensuite commencer à développer son implémentation au sein de la puce.

Les concepts d'énergie consommée et d'utilisation des ressources trouvent leur place en Enseignement Technique Transversal de STI2D. Les élèves sont amenés à réfléchir sur l'ordinateur et sa place dans le développement durable.

Enfin, j'ai réutilisé les concepts de traitement d'image et de manipulation de fichiers dans une séquence en Informatique commune à toutes les classes préparatoires scientifiques. Elle permet aux étudiants de manipuler des algorithmes au sein du langage de programmation Python, et de manipuler des structures de données communes telles que les chaînes de caractères et les tableaux.

# Appendices

## A Fiche PySéquence complète pour la séquence en STI2D



Commentaires : Cette séquence aborde le sujet du développement durable dans la filière STI2D à l'aide d'un problème concret de la consommation électrique d'un datacenter et les coûts engendrés. Elle permet d'aborder des soucis de relation coût-efficacité-complexité et d'aborder la compétitivité d'un système sur le plan technique, économique, et développement durable. Elle est aussi l'occasion d'aborder les premières notions de traitement de l'information : codage et traitement programmé.

Fiche créée avec le logiciel pySequence (<https://github.com/cedrick-f/pySequence>)