DémoJ

Rapport de projet

Participants
ADAM Arthur, BOUTINAUD Alexandre,
PALVADEAU Olivier, SALOMÉ Cameron,
MOULHERAT Hadrien, MARTY Jules,
TELLACHE Yasmine, LAINÉ ODIC Yann,
DAUMAND Anaëlle, PIAUD Charly

Avec l'encadrement de Olivier RIDOUX

Master Ingénierie Logicielle, Master Cloud et Réseau
ISTIC
Université de Rennes
2023-2024

Résumé

L'état de l'art de l'analyse des impacts énergétiques des systèmes informatiques distingue les impacts de fabrication des impacts d'utilisation, et parmi ceux-ci les impacts se répartissent en trois couches (trois *tiers* en anglais) : les équipements terminaux, réseaux, et serveurs. Les estimations varient, mais on convient que les impacts de fabrication et utilisation sont du même ordre de grandeur, et que les impacts de l'utilisation se répartissent à part égale entre les trois couches. Le résultat est que les utilisateurs non initiés ne perçoivent facilement que un sixième de l'impact énergétique global des systèmes informatiques.

Nous proposons un démonstrateur qui facilitera la médiation pour faire comprendre les autres impacts de l'utilisation. Il consiste en un système informatique complet (terminal, réseau et serveur) qui expose de façon très visible l'impact énergétique de chacune de ses couches. L'utilisateur verra trois boîtes, une par couche, chacune équipée de jauges lumineuses pour afficher leur consommation électrique et leur température. Ce dispositif permet d'aller au devant du public y-compris dans des lieux non académiques, comme des parcs ou des bars. Le projet DémoJ est de réaliser un tel système.

Ce rapport présente l'analyse et la réalisation des premiers blocs fonctionnels du système DémoJ, depuis l'infrastructure électronique commune à toutes les boîtes, dont les capteurs de consommation d'énergie et les afficheurs à base de leds, jusqu'à la programmation des scénarios de démonstration.

Table des matières

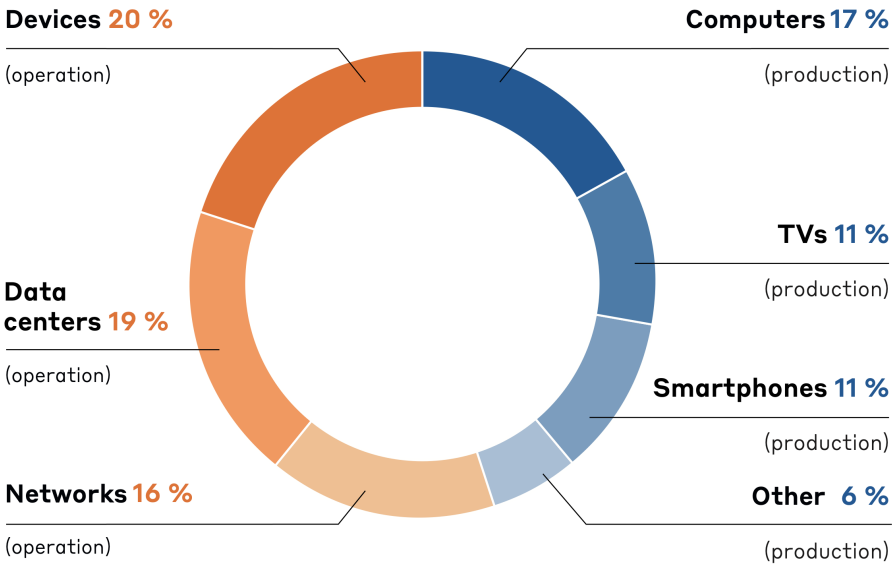

FIGURE 1 – Analyse des impacts énergétiques des impacts des systèmes numériques (source Shift Project [11]). En bleu, les impacts de la production, en orange les impacts de l'utilisation.

1 Introduction

1.1 Contexte

L'Atlas du numérique [2] présente une étude de l'ADEME [1] de 2022 selon laquelle la consommation électrique du secteur du numérique en France représente 10 % de la consommation électrique française. Une valeur plus faible, environ 8 %, est retenue au niveau mondial, mais les 10 % sont largement dépassés dans d'autres pays. Par exemple, en Irlande les seuls data centres ont représenté en 2022 18 % de la consommation d'électricité du pays [2]. Il est donc important de comprendre et faire comprendre d'où viennent ces consommations d'électricité dont la plupart ne se font pas devant les utilisateurs finaux.

On distingue d'abord la consommation pour la fabrication et la consommation pour l'utilisation. Différentes sources, ex. [5] et [11], posent que les deux parts se valent globalement, mais en fait cela dépend beaucoup du type d'appareil considéré et même de ses conditions d'utilisation. Par exemple, pour un smartphone en France (c-à-d. un appareil fabriqué en Chine, souvent en veille, et utilisé dans une région du monde où l'électricité est peu carbonnée), la part fabrication domine largement, environ 80 %, mais pour un serveur en Pologne (c-à-d. un appareil fabriqué en Chine, rarement en veille, et utilisé dans une région où l'électricité est très carbonnée) cela sera l'inverse. Le flou de la définition du secteur numérique ajoute un peu de confusion puisque certains équipements à l'utilisation très impactante comme les téléviseurs sont parfois inclus, parfois non.

Côté utilisation, on distingue trois couches (*tiers* en anglais) des systèmes informatiques : les équipements terminaux, les équipements réseau et les équipements serveurs (lire par exemple [5, 11]). À nouveau, il y a beaucoup de flou sur les proportions des uns

1. ADEME : Agence de l'environnement et de la maîtrise de l'énergie, maintenant appelée Agence de la transition écologique

2. *Central Statistics Office*, `https://www.cso.ie/en/releasesandpublications/ep/p-mec/meteredelectricityconsumption2021/`

et des autres, mais on peut dire que globalement les trois couches font part égale. La figure 1 représente une telle analyse.

Par exemple, l'Atlas du numérique [2] donne les valeurs suivantes :

— 11,55 TWh pour les équipements serveurs, soit 2,4 % de la consommation électrique française.

— de 8,3 TWh à 11,9 TWh pour les équipement réseaux, soit de 1,8 % à 2,5 % de la consommation. L'incertitude vient de l'attribution des objets connectés, 3,6 TWh ; sont-ils des équipements réseau ou non ? Noter aussi que l'Atlas du numérique range les *box* (internet, TV, etc.) parmi les équipement terminaux.

— 30,41 TWh, dont 16,6 TWh pour les équipements d'affichage (télévisions, ordinateurs, autres). Soit 6,5 % avec les écrans, et 2,9 % sans. Il faut aussi noter que cette statistique inclut des objets qui rentrent mal dans le modèle à trois couches, ex. les projecteurs, imprimantes et copieurs, etc. pour un total de 5 TWh.

Ces exemples montrent que l'impact énergétique des systèmes informatiques est important mais difficile à cerner. Le grand public ne voyant que l'utilisation des équipements terminaux n'a une perception que d'un sixième de cet impact. Comment peut-il songer diminuer son impact sur les cinq sixièmes qu'il ne connaît pas ? Comment peut-il comprendre une réglementation contraignante qui s'appliquerait sur les cinq sixièmes qu'il ne connaît pas mais se répercuterait sur lui ? Nous pensons que pour répondre à cela il faut concevoir des médiations simples mais dynamiques : simple pour éviter les problèmes d'attribution ou les difficultés d'interprétation ; dynamique pour ne pas se limiter à une feuille de données, ou à un graphique, mais permettre de faire des expériences.

1.2 Un support de médiation simple et dynamique

Nous proposons pour cela de concevoir un démonstrateur des impacts énergétiques de l'utilisation des systèmes informatiques qui suit à la lettre le modèle à trois couches. Il sera lui-même un système électronique à trois couches que le packaging rendra très visibles, voire tangibles. On pourra toucher chacune des couches ! Chaque couche exposera visuellement l'énergie qu'elle consomme à l'aide d'une jauge lumineuse.

Chaque couche jouera des scénarios typiques de son rôle, ex. servir des fichiers petits ou gros pour la couche serveur. Les répercussions énergétiques de ces scénarios se verront alors sur les autres couches. Nous espérons ainsi permettre une médiation plus active où les participants pourront jouer des scénarios exploratoires : que se passe-t-il si ... ?

Le système permettra aussi de faire agir les participants via leurs smartphones. En effet, ceux-ci pourront exécuter une application qui leur offre les mêmes scénarios que ceux de la couche client. Cela permettra de démultiplier les équipements terminaux et de rendre visibles des scénarios où les effets cumulés de nombreux utilisateurs affectent les infrastructures.

Le système devra pouvoir être utilisé dans des *tiers lieux* [9], y compris des tiers lieux éphémères comme des parcs ou des bars aménagés temporairement pour héberger une médiation. Le système devra être autonome électriquement et informatiquement ; il ne nécessitera donc ni connexion électrique ni connexion réseau. Il n'utilisera pas non plus les supports de médiation usuels qu'on peut trouver dans les espaces professionnels (les *seconds lieux*) ; par exemple, il pourra se passer de vidéo-projecteur. Il sera linguistiquement le plus neutre possible, et s'appuiera sur le moins possible de codification, et, quand il y en a, le plus souvent possible sur des codifications largement utilisées par ailleurs. Par exemple, l'affichage de l'intensité d'un phénomène en vert-orange-rouge, pour inten-

sité faible, intensité moyenne et intensité élevée voire trop élevée, est largement utilisée, depuis l'affichage de l'intensité énergétique des habitations ou des appareils domestiques, jusqu'à l'affichage de l'intensité de sortie d'un amplificateur audio, en passant par les compte-tours des moteurs et le nutriscore.

L'énergie consommée par un système informatique se dissipant essentiellement en chaleur, on s'attachera à rendre visible l'énergie électrique consommée instantanée (donc essentiellement une puissance), et la température du système. Formellement, ces deux mesures ne sont pas exactement des mesures de l'énergie, mais elles y sont liées et il nous semble qu'elles sont assez concrètes pour l'auditeur non scientifique. Pour faire de la puissance instantanée une mesure de l'énergie, il faudrait l'intégrer par rapport au temps. Et la mesure de température introduit des phénomènes complexes comme son inertie, le fait qu'elle soit très peu uniforme dans le système, et le fait que pour s'en protéger les systèmes informatiques sont organisés pour évacuer la chaleur qu'ils produisent. On essaiera donc de mesurer la température à la source de la dissipation de chaleur et on prendra le risque de concevoir un système qui ne cherche pas trop à évacuer cette chaleur.

On appellera ce système **DémoJ**, **J** comme joule.

1.3 Le projet DémoJ

Le projet DémoJ a été présenté au Collectif Numérique Responsable du Poool [3] [4], et y a rencontré un accueil favorable avec une invitation à présenter le projet au printemps 2024. Il a aussi été présenté au projet AIR de l'université de Rennes [5] dans la perspective d'une utilisation du système DémoJ dans les enseignements liés à la transition environnementale qui sont destinés à monter en puissance dans l'Université de Rennes. Il y a aussi rencontré un accueil favorable.

Une première tentative de réaliser le système DémoJ a été faite en 2022-23, mais n'a pas abouti. Elle a cependant produit des analyses sur lesquelles nous nous appuyons [10]. En particulier, nous reprenons l'idée de placer le projet DémoJ dans le contexte de la (pseudo-)entreprise DémoTech, dont le domaine d'activité est celui du support technique à la médiation et l'enseignement des technologies. Le projet DémoJ est donc un projet de la société DémoTech.

2 Le projet DémoJ

Ce chapitre décrit notre plan d'action pour réaliser le système DémoJ. Cela concerne l'organisation générale du système, ainsi que l'organisation de notre équipe.

3. Poool : successeur de la technopole Rennes Atalante pour le domaine du numérique, `https://lepoool.tech`. Porteur du label La French Tech du ministère de l'Économie, des Finances et de la Souveraineté industrielle et numérique : `https://lafrenchtech.gouv.fr`

4. Collectif Numérique Responsable : groupe de travail du Poool sur le thème du numérique responsable, `https://lepoool.tech/collectif-numerique-responsable/`

5. AIR : Augmenter les interactions à Rennes, `https://www.univ-rennes.fr/actualites/projet-air-experimenter-pour-transformer-grande-echelle-la-pedagogie-numerique`. Ce projet répond à l'appel à projet national « Démonstrateurs numériques dans l'Enseignement Supérieur ».

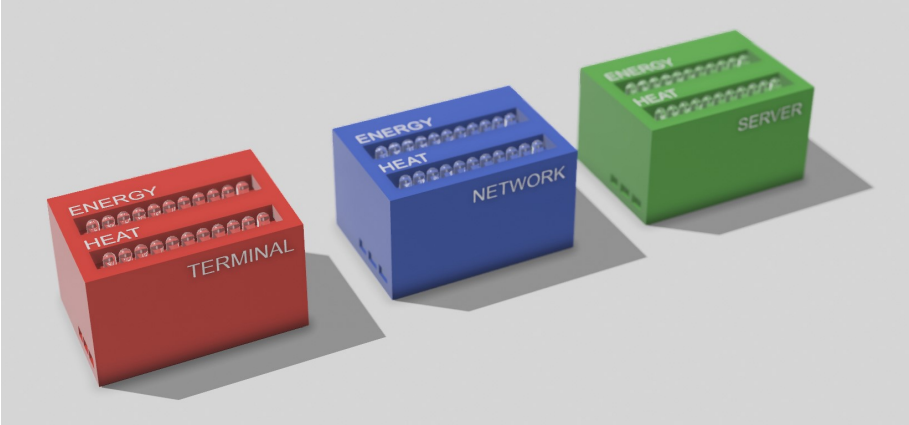

FIGURE 2 – Les trois boîtes du système DémoJ (vision d'artiste ; source, projet DémoJ 2022-23)

2.1 Les composants du système DémoJ

Vu de l'extérieur, le système DémoJ doit prendre la forme de trois boîtes matérielles affichant chacune des jauges qui dévoilent leur consommation d'énergie. Rappelons que ces trois boîtes correspondent aux trois couches de l'analyse usuelle qui est faite de la consommation d'énergie due à l'utilisation des systèmes informatiques : les équipements terminaux, les équipements réseaux, et les équipements serveurs. À l'intérieur de chacune de ces boîtes un système électronique-informatique permet de capturer les données énergétiques de la boîte, et un système informatique permet de former un réseau entre les trois boîtes. Ce dernier système permet aussi d'exécuter des applications spécifiques qui donnent à chacune de ces boîtes son rôle de terminal, réseau ou serveur.

Nous décrivons dans cette section notre analyse de ces composants et nos premières décisions les concernant.

2.1.1 Les boîtes

Afin de rendre tangible les trois couches systèmes de l'analyse des impacts énergétiques du fonctionnement des systèmes informatique, nous les réalisons par des sous-systèmes indépendants disposés dans des boîtes distinctes. Puisqu'elles doivent être vues de loin tout en restant intelligibles, ces boîtes ont un affichage réduit à des jauges lumineuses d'assez grande taille, et une interface de contrôle déportée sur le terminal de l'opérateur de la démonstration. La figure 2 donne une idée de ce que pourraient être ces boîtes.

Nous avons envisagé dans un premier temps de réaliser ces boîtes sur la base de conteneurs plastiques du commerce. L'objectif était de profiter de la géométrie de ces objets déjà faits, et de gagner du temps pour se consacrer aux composants plus fonctionnels. Cependant, cette proposition a reçu des remontées négatives, et nous avons décidé de fabriquer nos propres boîtes en bois. La plus répétée des objections était d'ordre plutôt symbolique ; pourquoi utiliser des objets en plastique, qui n'ont eux-même pas une bonne réputation vis-à-vis de l'écologie et qui ne sont pas forcément facilement remplaçables, pour initier aux impacts environnementaux des systèmes informatiques ? Cette objection revenant trop souvent, nous avons décidé de prendre les devants et d'éviter le plastique, privilégiant le bois.

Les boîtes seront réalisées au fablab de l'université de Rennes. Elles seront constituées de pièces de bois conçues et découpées numériquement. Après quelques tâtonnements,

nous avons choisi un format de parallélépipède vertical, inspiré d'une tour, doté de deux ouvertures verticales pour les jauges. Cette disposition offre une grande visibilité, permet de changer le sens du support, et s'accorde bien avec le principe des jauges lumineuses. Celles-ci donneront l'impression de se remplir plus ou moins selon l'intensité énergétique des opérations exécutées.

On se propose d'utiliser des couleurs distinctives pour chacune des boîtes, par exemple, rouge pour la boîte terminal, bleu pour la boîte réseau et vert pour la boîte serveur. Cela pourra consister en une coloration globale de chacune des boîtes (comme sur la figure 2) ou en la coloration des signalétiques imprimées sur chaque boîte.

Ces boîtes devront héberger tout l'équipement électronique-informatique de façon à le protéger tout en le laissant accessible, par exemple pour la maintenance, ou visible, dans le cas des jauges. Enfin, elles devront prévoir des passages pour des connecteurs et des boutons de contrôle.

2.1.2 Le système électronique-informatique

Chaque boîte contient un exemplaire d'une plate-forme commune que nous concevons et réalisons, et des composants spécifiques, surtout logiciels, qui correspondent au rôle joué par chaque boîte. La plate-forme commune comprend une couche électronique, ex. alimentation, carte processeur, capteurs et circuits d'affichage (voir figure 3), une couche système et une couche réseau, ex. de quoi faire fonctionner un réseau ad hoc qui réunit les trois boîtes, un terminal pour l'opérateur animateur de la démonstration, et éventuellement des terminaux opérés par le public.

Les composants spécifiques correspondent aux rôles d'équipement terminal, ex. envoyer des requêtes et exploiter les réponses, d'équipement réseau, ex. acheminer les requêtes et les réponses entre les équipements terminaux et serveurs, et le rôle d'équipement de service, ex. un serveur web.

2.1.3 Interface utilisateur

L'interface d'entrée du système DémoJ se limite à ce qui permet de l'allumer/éteindre et à ce qui permet de le lancer/relancer (voir côté gauche de la figure 3). Le contrôle plus fonctionnel du système DémoJ se fera à l'aide d'une application qui sera installée sur un terminal de l'opérateur de la médiation, par exemple son smartphone. L'application de contrôle devra donc être très portable, et pour cela nous optons pour une application web qui s'exécuterait dans le navigateur du terminal de l'opérateur et agirait sur des variables du système DémoJ.

Nous avons aussi prévu que les auditeurs puissent interagir avec le système DémoJ. Ils le feront en tant que équipement terminal. Pour cela, une variante de l'application de contrôle leur sera mise à disposition pour téléchargement. On espère ainsi créer des situations plus réalistes que celle où il y a un équipement terminal pour un équipement serveur. En effet, il y a en réalité beaucoup plus d'équipements terminaux que d'équipements réseaux ou de service. Selon une étude de l'association GreenIT [3], il y avait en 2019 plus de 30 milliards d'équipements terminaux, plus de 6 milliards si on ne compte que les équipement ayant un écran (les autres sont les objets connectés, souvent enfouis dans autre chose : ex. bâtiments, véhicules, appareils domestiques), pour plus de 1 milliard d'équipements réseau, environ 200 millions si on se limite aux équipements de cœur de réseau (les autres sont les équipements de la périphérie du réseau : ex. *box* internet). Il n'y aurait que quelques milliers de datacentres qui hébergeraient 70 millions de serveurs.

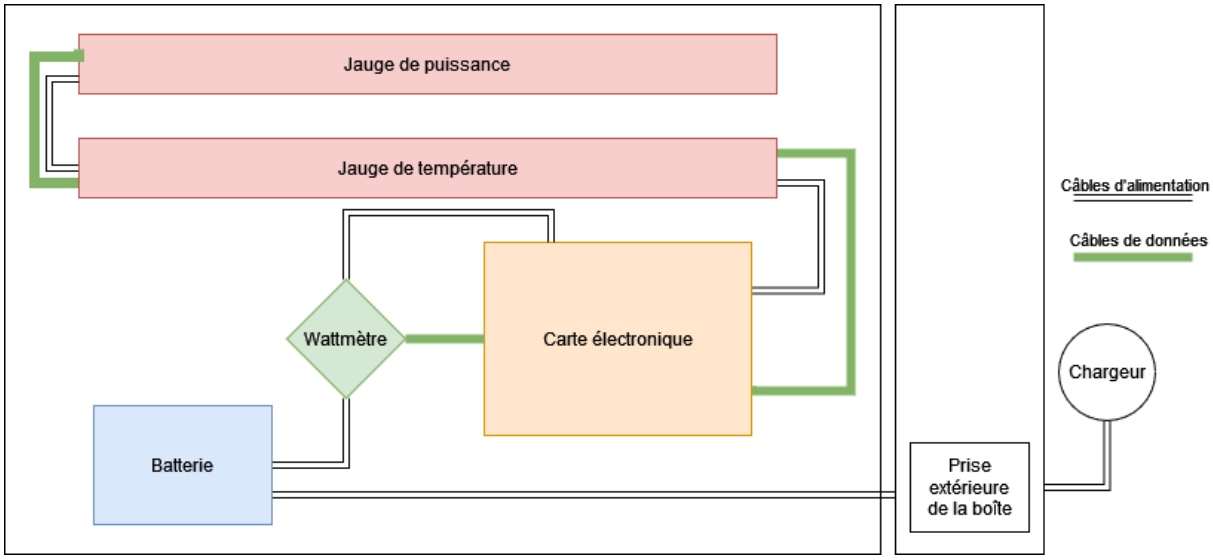

FIGURE 3 – Schéma synthétique de l'électronique des boîtes

Un système DémoJ proportionnel à ces quantités pourrait avoir une boîte serveur, deux ou trois boîtes réseau et une centaine d'équipements terminaux. Une boîte serveur, une boîte réseau, une boîte terminal, plus une dizaine de smartphones des auditeurs en feront une approximation raisonnable.

2.2 Le système électronique

Le système DémoJ est fondamentalement hybride. Il est composé d'une couche électronique qui offre capteurs, afficheurs et puissance de calcul, et d'une couche informatique qui contrôle le tout.

2.2.1 Organisation globale

À l'intérieur de chaque boîte, nous allons avoir un même système électronique contenant plusieurs composants principaux : une batterie pour permettre le fonctionnement hors réseau électrique (bloc bleu dans la figure 3), un système monocarte, ou *single-board computer*, pour rendre chaque boîte programmable (bloc orange dans la figure 3), des jauges d'affichage (bloc rose dans la figure 3) et des capteurs d'énergie consommée, dont un wattmètre (bloc vert dans la figure 3).

Concernant la mesure et l'affichage de la consommation d'énergie, nous avons choisi de montrer la consommation d'électricité instantanée, c'est-à-dire une puissance (des watts), et la chaleur dissipée par le processeur (des °C). La chaleur mesurée n'est pas exactement la quantité d'énergie consommée puisqu'elle se dissipe en dehors du système DémoJ, même en l'absence de dispositifs pour favoriser cette dissipation (ventilateur et/ou radiateur). Malgré tout, nous pensons que la chaleur est la forme d'énergie dont l'auditeur a l'expérience la plus concrète après le niveau d'essence dans une voiture.

Il faudra donc que l'intervenant qui utilise le système DémoJ explique ces concepts.

Énergie : ressource qui permet à un système de changer d'état ou de résister à des changements d'états spontanés. L'énergie se mesure en principe en joule (abréviation J), mais de nombreuses autres unités sont utilisées, ex. le wattheure pour l'énergie électrique.

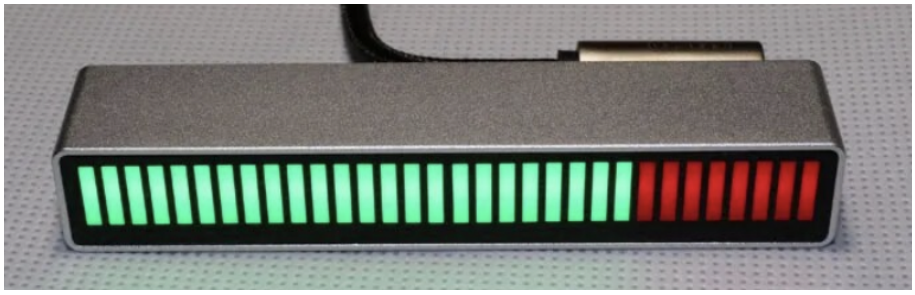

FIGURE 4 – Exemple de jauge visuelle (source AliExpress)

L'énergie existe sous de nombreuses formes, et de nombreuses conversions sont possibles, jamais sans perte [8].

Puissance : vitesse à laquelle l'énergie est consommée, ou produite. La puissance se mesure en watt (abréviation W), mais ici aussi de nombreuses autres unités sont utilisées, ex. le cheval-vapeur. La puissance électrique est le produit de l'intensité par la tension, et lorsque la tension est constante, il arrive souvent de mesurer la puissance en ampère (abréviation A) comme si c'était une intensité. Par exemple, la puissance des batteries ou d'un abonnement domestique est souvent exprimée en ampère.

2.2.2 Les jauges

Les jauges sont le seul dispositif d'affichage vers le public. Nous utiliserons un format inspiré des jauges de table de mixage (ex. figure 4), mais que l'on retrouve dans de nombreux autres dispositifs. On pourra par exemple avoir une moitié de la longueur qui s'affiche en vert, le quart suivant en orange, et le dernier quart en rouge.

2.3 Gestion du projet

2.3.1 Organisation de l'équipe

Afin de mener à bien le projet DémoJ, nous avons adopté un cycle de développement agile, en se rapprochant d'une méthode de développement dirigée par les démonstrations (DDD [6]). La table 1 rappelle rapidement le vocabulaire et les rites de l'agilité. Dans ce cadre, le DDD consiste à caractériser les objectifs de chaque *sprint* par un protocole de démonstration. Si la démonstration se passe bien, la tâche à réaliser est réputée achevée.

Nous nous sommes donné des rôles organisationnels dès le début du projet, mais nous avons retardé le plus longtemps possible la spécialisation technique afin que la prise en main du projet soit partagée par tous. Les rôles organisationnels sont les suivants :

- *scrum master* : (définition dans la table 1), Arthur Adam et Alexandre Boutineau ;
- *doc master* : le *scrum master* de toutes les communications vers l'extérieur, par exemple, rapports, posters, démos, Olivier Palvadeau ;
- *infra master* : le *scrum master* des infrastructures, par exemple, dépot git, ressources matérielles, acquisition de nouvelles ressources, Arthur Adam.

Bien noter que de même que le *scrum master* n'est pas l'expert de tout et encore moins le développeur de tout, ces trois rôles sont d'abord des rôles d'animation et de supervision. Ils sont supposés être soutenus par des rôles d'expertise technique.

6. DDD : *Demo Driven Development*

Les spécialisations se font progressivement selon les besoins du projet et les goûts de chacun : couche électronique, couches systèmes et réseaux, couche applicative, packaging des boîtes, et communication.

2.3.2 Stratégie agile

Le projet DémoJ est largement exploratoire et certains de ses aspects débordent de nos compétences académiques. À cause de cela, il est difficile de prévoir à l'avance le travail, et donc le temps, nécessaire au développement de chacun de ces composants. Nous nous sommes donc organisés autour d'une démarche agile [1] qui permet de se voir avancer et de gérer rapidement les blocages qui pourraient survenir.

Le vocabulaire et les rites de l'agilité sont bien connus, mais cela laisse ouvertes les dimensions plus stratégiques. Beaucoup de documentations sur l'agilité disent assez bien comment avancer pas à pas dans une direction, mais elles ne disent pas comment choisir la direction. Nous détaillons un peu plus la stratégie dans les lignes qui suivent. Il faut noter aussi que nous n'appliquons pas les principes de l'agilité à la lettre, ne serait-ce que parce que le développement de ce projet n'est qu'une de nos activités d'étudiants. Par exemple, nos *daily meetings* ne sont pas quotidiens.

La stratégie que nous suivons est fondée sur deux principes :

— Commencer par les fonctions qui recèlent le plus d'incertitude sur la façon de les réaliser. En effet, réaliser ces fonctions nécessite un apprentissage difficile à évaluer et possiblement de nombreux essais et erreurs. Il ne faudrait pas se trouver bloqué parce qu'il ne resterait pas assez de temps pour cela.

— Traiter ces fonctions de bout en bout, c'est-à-dire en ajoutant tous les composants nécessaires pour que leur mise en œuvre produise un effet intelligible pour le client.

Pour nous qui sommes plutôt spécialisés dans le développement d'applications logicielles ces fonctions qui recèlent le plus d'incertitude sont d'abord celles qui comportent des éléments électroniques, puis les fonctions plutôt réseau, puis les fonctions plutôt systèmes. Par exemple, la fonction de mesure de l'énergie consommée est prioritaire à cause des technologies mises en œuvres, dont un montage électronique, et la traiter de bout en bout nécessite de réaliser le montage, le connecter matériellement au processeur, programmer un module de gestion pour lire ces données, et un module d'affichage pour les présenter au client dans le cadre d'une démonstration. De la même façon, l'affichage sur un bandeau de leds est prioritaire à cause des technologies mise en œuvres, dont un montage électronique, et le traiter de bout en bout nécessite de réaliser le montage, le connecter matériellement au processeur, et programmer un module de gestion pour afficher des données déterminées. Cela montre qu'à la priorité pour le *product owner* et l'effort pour les développeurs s'ajoutent d'autres indicateurs, comme l'incertitude ou l'urgence technique. Cette démarche conduit à développer des blocs fonctionnels pour tous les verrous qu'il est possible d'envisager a priori. Elle nous rassure sur notre capacité à réaliser le système DémoJ, et cela rassure aussi notre *product owner*. Nous avons aussi remarqué que cela facilitait la communication avec l'atelier d'électronique de l'ISTIC.

Dernier point, la démarche agile repose largement sur le management visuel, et en particulier la représentation de l'état d'avancement du projet par un *scrum board*. Mais nous de disposons pas d'un espace de travail stable où un *scrum board* pourrait être affiché à demeure. Il existe des solutions numériques comme Trello, mais le côté très tangible du *scrum board* mural nous manquait. Nous avons donc proposé une autre représentation des avancées du projet plus directement inspirée des étiquettes *kanban*.

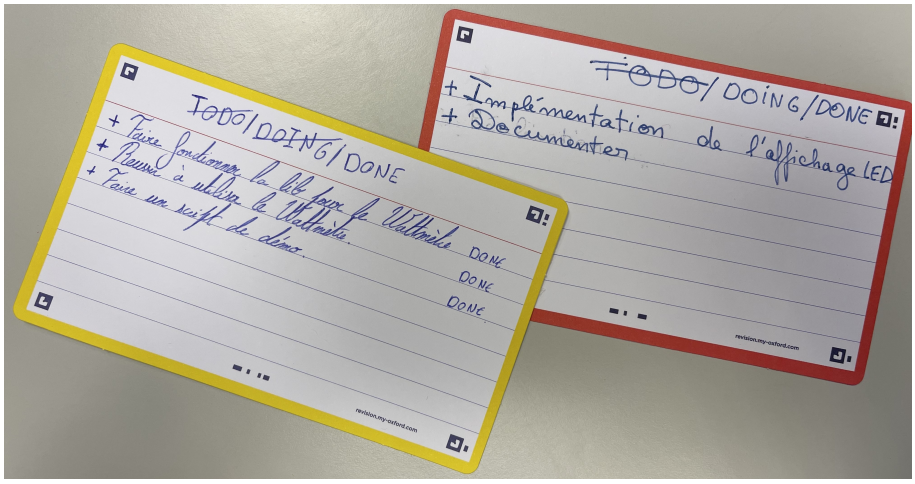

FIGURE 5 – Deux cartes de *user stories*

À l'image d'un *scrum board*, où différentes tâches à réaliser sont visibles et attribuées à des personnes, chacune des cartes représente une tâche spécifique, associée à un individu ou à un groupe. Les états de chaque tâche *(todo, doing, done)* sont également clairement indiqués sur chaque carte. Les cartes arborent des couleurs distinctes, chaque teinte correspondant à une équipe spécifique (informatique, électronique, réseau, communication). Cette méthode permet une évaluation rapide de la charge de travail de chaque équipe, facilitant ainsi l'ajustement des effectifs. Au cours des réunions de travail, les cartes sont posées sur la table pour former un *scrum board* éphémère, offrant à chacun une vision d'ensemble de l'avancement du projet.

TABLE 1 – Vocabulaire et rites de l'agilité.

Début	
sprint	Unité temporelle de développement, de durée constante fixée a priori. Les *sprints* s'enchaînent, en principe avec l'objectif de se rapprocher progressivement, mais avec confiance, des exigences du client.
product owner	Le client ou un représentant de celui-ci. Ce qui est nouveau avec l'agilité est de considérer qu'il « est dans la boucle », c'est-à-dire qu'il est accessible à tout moment. Il est au moins accessible à la fin de chaque sprint, et comme les sprints se répètent, la présence du *product owner* aussi.
scrum master	L'interlocuteur du client au sein de l'équipe de développement. Il en est aussi l'animateur pour garantir le bon fonctionnement de la démarche agile, c'est-à-dire à la fois le respect des exigences du *product owner* et le respect des exigences de durabilité de l'équipe de développement. Le *scrum master* n'a pas besoin d'être un expert technique ; cette compétence doit se trouver dans l'équipe de développement.
user stories	Les exigences du *product owner* formalisées en unités expérimentales testables. Le *product owner* peut en formuler autant qu'il veut quand il veut. Il peut aussi en supprimer, ou les transformer.
priorité	Le *product owner* assortit chaque *user story* d'une priorité. La priorité est exprimée par un nombre, mais sans interprétation métrique. Par exemple, 4 est plus prioritaire que 2, mais pas 2 fois plus prioritaire.

Suite de la table 1	
backlog	L'ensemble des *user stories* formulées par le client, mais pas encore réalisées. Peut évoluer constamment (voir *user stories*).
effort	L'équipe de développement évalue l'effort nécessaire du développement des *user stories* formulées par le *product owner*. L'effort est exprimé par un nombre, avec une interprétation métrique. Par exemple, 4 est un effort plus grand que 2, et même 2 fois plus grand. Il est recommandé de ne pas chercher à exprimer de façon trop précise les efforts. Pour cela, on utilise souvent des échelles non linéaires, ex. la suite de Fibonacci.
vélocité	Nombre de points d'effort que l'équipe peut soutenir par *sprint*. Excessivement difficile à estimer, mais l'espoir est qu'en répétant l'exercice l'équipe le fera de mieux en mieux. La vélocité n'est pas constante. Elle varie avec l'effectif de l'équipe, la disponibilité de ses membres, l'arrivée de nouveaux membres, le départ de membres expérimentés, etc.
sprint planning	Opération de brainstorming par laquelle le *product owner* et l'équipe de développement se mettent d'accord sur les objectifs du *sprint* à venir. Les objectifs sont une sélection de *user stories* qui sont encore dans le *backlog*. Le *sprint* planning prend en compte les priorités du *product owner*, et l'effort évalué par l'équipe de développement. On cherche à maximiser la satisfaction du *product owner* tout en s'adaptant à la vélocité de l'équipe. Celle-ci n'est qu'estimée, surtout en début de projet, ou pour une équipe nouvellement constituée.
planning poker	L'estimation de l'effort de développement d'un logiciel est une des choses les plus difficiles à faire. C'est en partie à cette difficulté que s'attaque l'agilité en général. Le *planning poker* est un rituel de brainstorming qui permet de le faire de façon argumentée. Il demande que chaque membre de l'équipe de développement se forge individuellement une estimation de l'effort, puis dans une phase collective, que les porteurs des estimations extrêmes s'expliquent. Des éditeurs proposent des jeux de cartes qui peuvent servir de support au *planning poker*. Plus conceptuellement, l'agilité permet un suivi des estimations successives et leur évaluation. On espère que grâce à cela l'équipe de développement montera en maturité et fera des estimations toujours plus correctes.
time boxing	L'idée que les *sprints* ont une durée fixée à l'avance. Un sprint ne s'arrête pas parce qu'il a atteint ses objectifs, mais parce que il a épuisé son crédit de temps. L'agilité oblige à en faire le constat et à analyser ce qui n'a pas bien marché quand les objectifs et *time box* ne s'accordent pas.
test	Le passage obligé de toute fin de sprint, avec la démo. À chaque fin de sprint, l'évaluation se fait par rapport aux *user stories* du sprint. Celles-ci doivent être testables, sinon elles sont mal définies, et elles doivent être testées. Le test n'est donc pas une activité annexe mal intégrée, mais une activité complètement intégrée au cycle de vie agile.
démonstration ou *démo*	L'autre passage obligé de toute fin de sprint. Fondamentalement, test et démo appartiennent au même spectre de la validation empirique. La démo doit absolument parler au *product owner* en se plaçant dans son domaine de compétence.

Suite de la table 1	
rétrospective	À la fin de chaque *sprint*, il est recommandé de se demander ce qui a bien marché, ce qui aurait pu mieux marcher, et comment, etc.
célébration	La dimension humaniste de l'agilité ne doit pas être négligée. Célébrer les fins de *sprint*, même si le résultat n'est pas excellent, est une façon de maintenir ou remonter le moral de l'équipe de développement.
stand-up meeting ou *daily meeting*	Un autre rituel à dimension humaniste. Ses deux appellations rappellent ses deux caractéristiques. Il s'agit d'une rencontre quotidienne, qui se tient d'habitude debout. On y présente les actions du jour, on peut y solliciter de l'aide pour les problèmes qui résistent, et on peut même réaffecter les effectifs pour faire sauter un verrou. Il se tient debout pour favoriser une posture dynamique et ne pas traîner en longueur. On y adjoint des rites auxiliaires, comme celui de matérialiser la prise de parole par l'acquisition d'un objet, ex. une balle.
burn-down chart	On fait l'hypothèse que l'effort nécessaire à réaliser les *user stories* se répartit uniformément dans la durée du *sprint*. L'effort restant est donc proportionnel au temps qui reste. Cela forme une droite dans un diagramme temps-effort. Un *burn-down chart* superpose à ce diagramme les valeurs d'effort des *user stories* qui restent à réaliser. Cela visualise au quotidien le retard, ou l'avance, pris sur la planification du *sprint*.
scrum board	Un tableau avec des colonnes *BACKLOG*, *TODO*, *DOING*, *TESTING*, *DONE* (il peut y en avoir moins ou beaucoup plus), et où les *users stories* circulent d'une colonne à l'autre. Le *scrum board* est un exemple de « management visuel ». Il rend visible l'état d'avancement d'un projet. Il peut prendre la forme d'un mur décoré de post-its, ou la forme d'outils numériques, comme Trello [7]. Des éditeurs proposent des *scrum boards* pré-formatés pour tableau papier, tableau magnétique, etc.
Fin	

3 Premiers blocs fonctionnels

Suivant la stratégie développée à la section 2.3.2, nous avons identifié les points les plus incertains du système DémoJ de façon à les prototyper en premier sous forme de blocs fonctionnels. Nous présentons ici les premiers blocs fonctionnels réalisés.

3.1 Les boîtes

Le prototype de boîte (voir figure 6) a été fabriqué en faisant de la découpe laser de planches de carton. Les boîtes finales seront faites en utilisant la même technique et les mêmes outils, mais avec du bois. Sa réalisation a permis de se familiariser avec les outils de conception et fabrication numérique du fablab. Ces outils incluent

— une découpeuse laser, qui est une machine qui s'interface comme une imprimante mais qui grave ou découpe un matériau en fonction des traits (épaisseur et couleur) du fichier imprimé ;

7. Trello : `https://trello.com/`

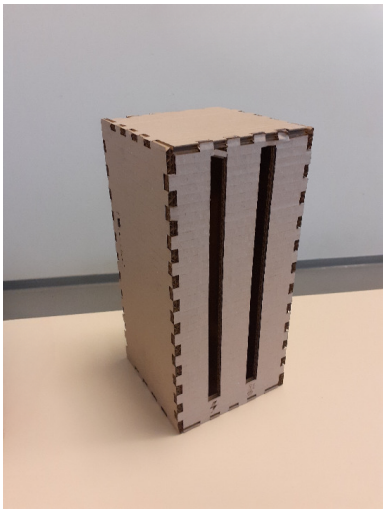

FIGURE 6 – Premier prototype de boîte (conception et découpe numérique)

— et le logiciel gratuit Inkscape, un logiciel de graphisme permettant de manipuler des images matricielles ou vectorielles pour avoir le maximum de précision sur le plan de découpe.

De plus, en parallèle de l'apprentissage de l'utilisation des machines et logiciels, nous avons réfléchi aux premières contraintes et comment les intégrer dans le design. Pour l'ouverture, et par extension la fermeture de la boîte, nous avons eu plusieurs idées qui devront être testées avec des maquettes :

— un couvercle à glissière, ce qui permet d'avoir une forme assez droite sans grosse contrainte technique mais la tenue n'est pas garantie avec les mouvements ;
— un couvercle à charnière, mais il faudrait un système de verrouillage car sinon la boite est facilement ouvrable mais pas sécurisée ;
— un couvercle avec des petits aimants néodyme (format des aimants des magnets de réfrigérateur), mais cette idée a été évoquée assez tardivement, donc elle n'a pas encore été réellement explorée.

Après plusieurs séances de réflexion de groupe, la solution privilégiée serait le couvercle à glissière avec un système d'aimants pour faire tenir le couvercle en place quand la boîte est fermée.

De plus, il faudra des ouvertures pour faire les interfaces avec les utilisateurs, comme des interrupteurs pour démarrer la boîte, un port de recharge de la batterie, les ouvertures pour les jauges. Celles-ci devront à la fois les rendre visibles et les protéger.

Ce prototype nous a aussi appris quelques contraintes que nous n'imaginions pas : et le gabarit de la découpeuse laser ne permet pas d'envisager des pièces trop grandes.

— Le processus de découpe (ou gravure) est assez long ; la découpe des pièces du prototype en carton à pris au total plus de 30 minutes. Le carton étant une matière plus facilement découpable que le bois, les boites finales en bois et ayant des plans plus complexes prendront plus de temps.
— La machine limite les dimensions des pièces qu'elle peut usiner ; les pièces ne peuvent dépasser 60 cm de longueur, 30 cm de largeur, et l'épaisseur maximale dépend du type du matériau utilisé et de l'action effectuée (gravure ou découpe).

3.2 Le système mono-carte

Dans le but de produire quelque chose de cohérent par rapport à la réalité, et d'avoir un environnement confortable pour le développement, nous utilisons des cartes **Raspberry Pi** [6]. Chaque boîte contient donc une carte Raspberry Pi. L'avantage de ces cartes est qu'il est possible de les utiliser à la fois comme des ordinateurs standards et comme des composants électroniques. En effet, on peut y installer un système d'exploitation tel que **Raspberry OS** qui est une variante de Linux, et elles rendent accessibles des ports de connexion physique, les ports GPIO [8]. L'environnement de développement des composants logiciels spécifiques est donc celui d'une machine Linux standard. Et l'environnement de développement de la plate-forme commune est celui de l'électronique numérique. Cela permet de créer des circuits électroniques contrôlables par la carte Raspberry Pi. Dans notre cas, cela permet de programmer facilement la mesure de la chaleur produite et de l'énergie consommée et de générer un affichage lumineux dynamique. La carte Raspberry Pi est alimentée par une batterie, et contrôle sans les alimenter les capteurs et les jauges. Ceux-ci sont alimentés par la batterie sur un circuit distinct.

Prévoyant des mises à jour fréquentes pendant la phase de développement, et des mises à jours plus rares mais réalisées par un opérateur non expert pendant la phase d'utilisation, nous avons réfléchi à une manière simple de réinstaller un système d'exploitation. Ici, simple signifie que cela devrait être possible sans connaissance approfondie et sans risque d'erreur de manipulation. Il a donc été décidé de créer un script bash qui effectue toutes les actions nécessaires pour créer le disque amorçable de chaque boîte.

3.3 Le système électronique

3.3.1 Vue d'ensemble

Le système électronique est organisé autour des composants suivants :
— batterie externe USB 57976 10000 mAh (bloc bleu dans la figure 7) ;
— carte Raspberry Pi 4 B - 2 GB (bloc orange) ;
— ruban NeoPixel RGB 1 m, 60 leds programmables ADA1138 (bloc rose) ;
— module wattmètre I2C Gravity SEN0291 (bloc vert).

Ces composants ont été commandés, mais nous ne les avons pas encore reçus.

Le ruban NeoPixel est un composant relativement complexe. Chaque led programmable est en fait constituée de trois ou quatre leds monocolores (rouge, vert, bleu, et parfois blanc selon modèle) plus un micro-contrôleur. Les leds programmables sont elles-mêmes assemblées dans des composants macroscopiques comme des rubans ou des matrices. Elles y sont assemblées en série ; c'est-à-dire que la sortie de l'une est connectée à l'entrée de la suivante. Un ruban NeoPixel doit être connecté par une extrémité à un contrôleur, ici la carte Raspberry Pi. Le contrôleur envoie des ordres comme *led 12 : rgb(52, 152, 36)*. Cet ordre passera de led en led, et seule la led programmable concernée, celle qui porte le numéro 12, l'exécutera et s'éclairera selon la consigne spécifiée.

Un ruban NeoPixel peut être découpé en des points spécifiques entre deux leds consécutives. Il comporte 60 leds pour 1 mètre de long. Une utilisation optimale serait donc d'en diviser 2 en 3 tronçons de 20 leds, ce qui fait 6 tronçons pour les 2×3 jauges. Cela implique que les boîtes devront faire au moins 33 cm de haut. On peut évidemment faire des tronçons plus courts, en utilisant les rubans de façon moins optimale.

8. GPIO : *General Purpose Input/Output*

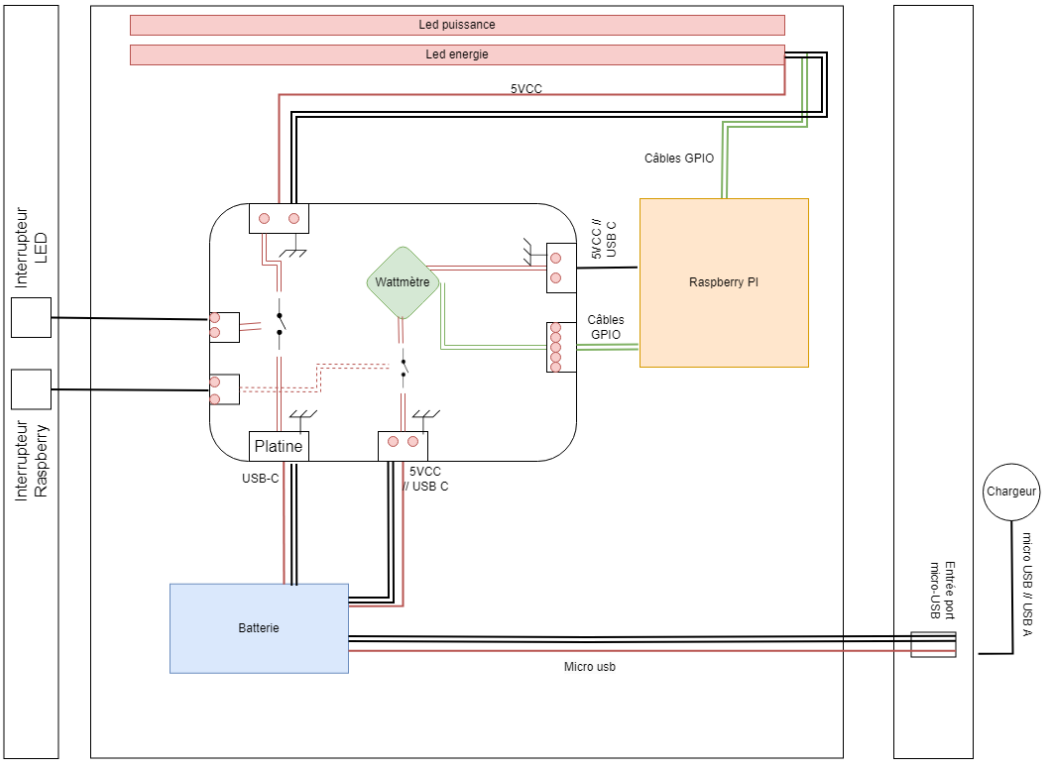

FIGURE 7 – Circuit électronique des boites

Les rubans NeoPixel sont potentiellement d'importants consommateurs d'énergie. En effet, chaque led programmable peut consommer jusqu'à 60 mA. Avec 2 jauges de 20 leds par boîte, cela fait un maximum de 2,4 A, mais qui n'est atteint que pour un éclairage à pleine puissance en couleur blanche alors que nous n'utiliserons que du vert, de l'orange et du rouge, et qu'une partie variable des leds programmables ne seront pas allumées. Du côté Raspberry Pi, la documentation indique une consommation maximale de 1,8 A. La batterie délivrant un maximum de 3 A, il faudra s'assurer que les jauges ne consomment jamais plus de 1,2 A. Noter aussi que le maximum de consommation des jauges devrait coïncider avec le maximum de consommation du Raspberry Pi.

3.3.2 Interface du circuit électronique

Les composants seront connectés à une interface contenant le wattmètre ainsi que des interrupteurs externes, de telle sorte qu'on puisse éteindre indépendamment le système mono-carte et les leds. Cette interface sera d'abord réalisée sur une plaque d'expérimentation (*bread-board*), puis en circuit imprimé.

Une prise externe est aussi prévue pour l'alimentation de la batterie afin de pouvoir la charger directement depuis l'extérieur. Ces aménagements ont été conçus pour ne pas devoir ouvrir les boîtes à chaque extinction ou allumage du système DémoJ.

L'affichage vers les auditeurs se fera uniquement via les jauges, et on a vu que celles-ci afficheront puissance et température.

3.3.3 Les capteurs

Capteur de température Avant même d'afficher la température du processeur, il faut commencer par retrouver sa valeur. Le Raspberry Pi étant contrôlé par un système

d'exploitation Linux, il est possible de récupérer les résultats des capteurs thermiques directement dans les fichiers du système dans le répertoire /**sys**/**class**/**thermal**/. Les fichiers des capteurs de températures sont organisés en zones de température. Dans le cas du Raspberry Pi, il y a quatre cœurs, mais une seule zone thermale : **thermal_zone0**. Chaque zone thermale possède un fichier "**temp**" contenant la valeur de température mesurée en kilo degré Celsius. Nous avons rédigé un programme en Python permettant de récupérer cette valeur afin de la fournir à un afficheur. La fréquence de mise à jour dépend du système d'exploitation. Il faut donc le prendre en compte dans nos programmes.

Nous avons développé un bloc fonctionnel de mesure de la température. Il montre une grande sensibilité aux conditions d'expérimentation. On observe aussi qu'il faudra sans doute installer les Raspberry Pi sans radiateur ni ventilateur pour que les effets de réchauffement en cas de stress soient visibles. En effet, en protégeant le circuit intégré, les radiateurs et ventilateurs rendent invisible la dégradation de l'énergie en chaleur.

Capteur de consommation électrique Pour montrer l'impact énergétique des couches systèmes que représente le système DemoJ, la puissance instantanée, en **milliwatts**, de l'alimentation du Raspberry Pi est mesurée. Il est important que les jauges ne soient pas directement alimentées par le Raspberry Pi mais par une source externe comme la batterie. Ainsi on mesurera ce que consomme chaque boîte pour les calculs et les services qu'elle propose plutôt que ce qu'elle consomme pour afficher sa consommation.

On a vu que le processeur du Raspberry Pi rendait visible sa température, mais au contraire des processeurs de la famille Intel qui rendent aussi visible leur consommation électrique via les données RAPL [9] [7] il ne donne pas accès à sa consommation électrique. Il faut donc la mesurer par des moyens externes à la carte Raspberry Pi.

Nous avons donc utilisé le capteur **INA219** [10], qui mesure la puissance avec une bonne précision (mW). Il peut être contrôlé en utilisant la librairie **DFRobot INA219**. Ce capteur communique en série avec le Raspberry Pi selon le protocole **I2C** [11]. Il faut donc activer sur le Raspberry Pi les canaux I2C sur les broches GPIO 2 (**SDA** [12]) et 3 (**SCL** [13]) pour la communication avec le wattmètre. Nous avons programmé un contrôleur de ce capteur, à partir de l'exemple fournit par DFRobot. Nous avons aussi programmé un script d'initialisation qui permet d'activer la communication I2C si elle n'était pas déjà activée et de rendre la librairie **DRobot INA219** visible des autres programmes. D'après la description du capteur INA219, celui ci ne nécessite pas de protection électronique particulière ; en effet il peut recevoir jusqu'à 26 V et 8 A. Or l'alimentation choisie pour le projet fournit au maximum 5 V et 2,5 A.

Nous avons observé que contrairement à la température, la mesure de l'énergie est plutôt stable et que lors d'un stress de la machine, elle change de façon instantanée et non progressivement comme le fait le capteur de température.

3.3.4 Les jauges

Avant de mettre en place et de contrôler une bande de leds programmables, il a fallu repartir des bases en faisant des essais simples, comme faire clignoter une led, puis en

9. RAPL : *Running Average Power Limit*
10. INA219 : `https://how2electronics.com/how-to-use-ina219-dc-current-sensor-module-with-arduino/`
11. I2C : *Inter-Integrated Circuit*, `https://les-electroniciens.com/sites/default/files/cours/cours_i2c.pdf`
12. SDA : *Serial Data Line*
13. SCL : *Serial Clock Line*

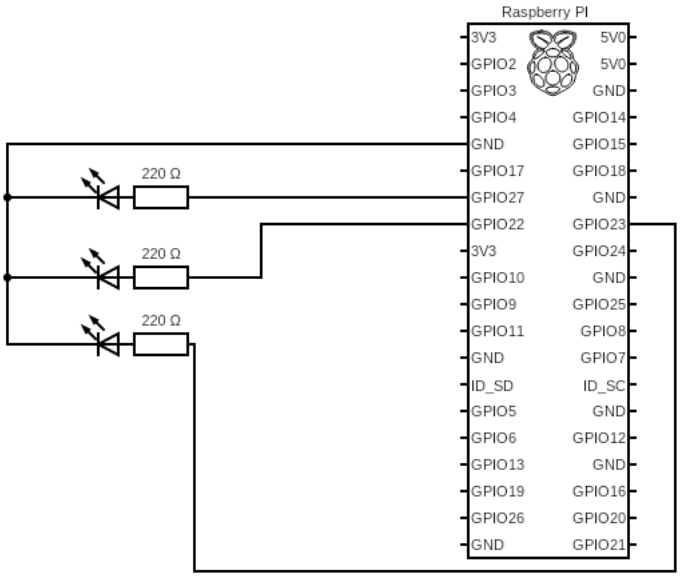

FIGURE 8 – Diagramme du montage du visualiseur de température

contrôler quatre individuellement, etc.

Contexte électronique Pour contrôler des composants électroniques tels que des sorties (ex. leds) ou des entrées (ex. capteurs), il faut utiliser le port GPIO de la carte Raspberry Pi. Chaque broche du port GPIO est alimentée par une tension de 3 V et une intensité de 16 mA. Pour ne pas endommager le matériel, il est conseillé qu'une broche du port GPIO ne soit pas utilisée pour alimenter plus de deux leds avec résistance. En effet, les broches GPIO ont surtout pour but de contrôler des composants plutôt que de les alimenter. La résistance à utiliser pour une led peut dépendre de sa couleur. Dans notre cas, nous avons utilisé des leds rouges de 1,6 V. D'après la loi d'Ohm, il faut alors une résistance d'au moins 106,25 Ω. Cependant, il est plus simple de se procurer des résistances qui ont des valeurs standardisées, c'est pourquoi nous avons utilisé des résistances de 220 Ω à la place dans nos premières réalisations.

Un montage naïf pour observer la température La figure 8 montre un premier essai de bloc fonctionnel capable de mesurer et donner une représentation de la température. Ce premier montage permet d'observer la température du CPU affichée sur trois diodes lumineuses pour faire une jauge à quatre niveaux (0, 1, 2 ou 3 leds allumées). Pour tester le bon fonctionnement de ce bloc fonctionnel, il faut d'abord un programme permettant d'imposer au Raspberry Pi une charge de calcul calibrée afin de le faire chauffer. Il existe des programmes faisant cela, comme les programmes **stress**[14] et **stress-ng**[15]. Ces programmes exécutent plusieurs autres sous-programmes et il est possible de choisir combien de cœurs du processeur on souhaite stresser. Un second programme contrôle les leds en fonction de la température. Il récupère la température du CPU à une fréquence déterminée à l'avance et allume ou éteint les leds selon la valeur lue.

14. stress : `https://github.com/mattixtech/stress`
15. stress-ng : `https://github.com/ColinIanKing/stress-ng`

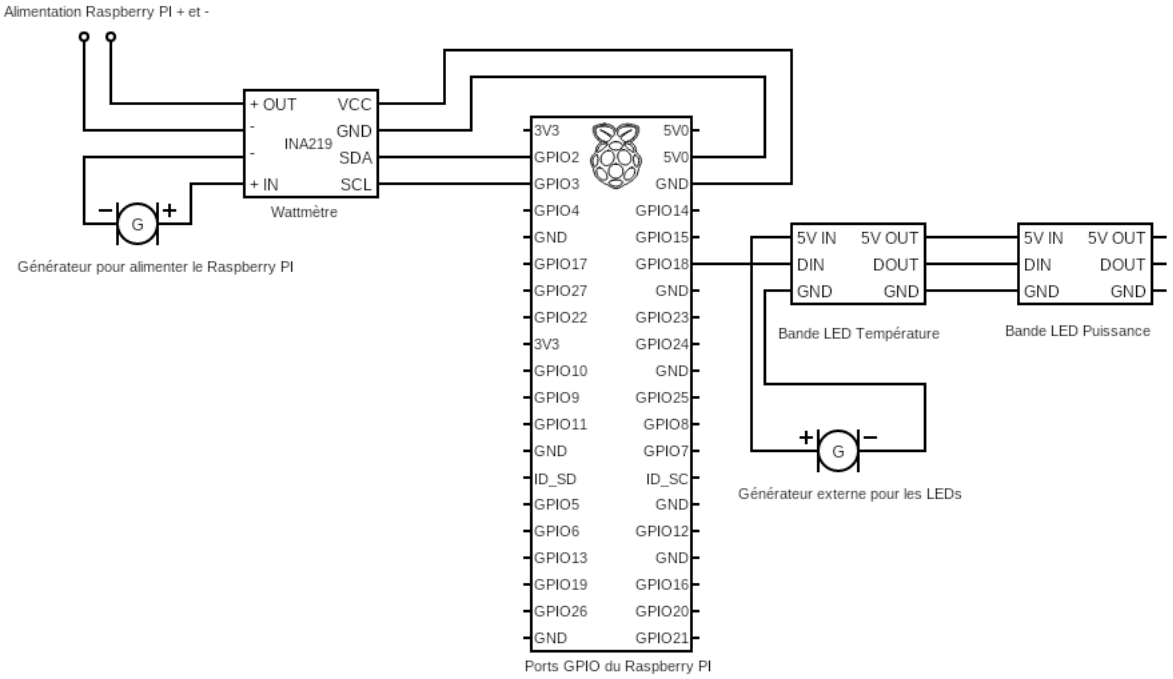

FIGURE 9 – Diagramme du montage réaliste des jauges

Nous avons pu observer que l'affichage de la température est très instable. C'est pourquoi, pour montrer une progression plus fluide et sans clignotement, on se propose de lisser l'affichage en utilisant une moyenne mobile : par exemple, **EMA** [16].

Nous avons aussi observé que la dynamique de la température n'était pas distribuée uniformément sur la plage des températures possibles (de 20 °C à 80 °C). Par exemple, à la mise sous tension du processeur la température monte rapidement vers 30-40 °C, mais sans activité applicative. Nous envisageons donc un affichage qui ne soit pas linéaire. Par exemple, moins de leds seraient consacrées aux températures de 20 °C à 30 °C qu'aux températures de 50 °C à 60 °C. Comme les jauges seront constituées de quelques dizaines de leds seulement, ex. 20, on pourra obtenir cet effet à l'aide d'une table de quantification.

Un montage réaliste pour les jauges Passer de trois leds monocolores à quelques dizaines de leds programmables en couleur et en intensité nécessite de changer d'architecture. Quand les leds sont simples et peu nombreuses, il est possible d'attribuer à chacune une broche du port GPIO, chaque broche alimentant sa led. Mais quand les leds sont trop nombreuses et complexes, cette organisation ne marche plus pour plusieurs raisons :

— Une raison logique simple : il n'y a pas assez de broches disponibles.

— Une raison énergétique : le port GPIO dans son ensemble ne peut pas délivrer assez de puissance pour toutes les leds, même si il n'y a pas plus de leds que de broches. En effet, le port GPIO pris globalement ne peut pas délivrer plus de 50 mA, et chacune de ses broches ne peut pas délivrer plus de 16 mA. Chaque led programmable NeoPixel consomme environ 60 mA par point de lumière, donc de l'ordre de 2 A au total pour 30 leds programmable. C'est-à-dire bien plus que ce

16. EMA : *Exponential Moving Average*, ou moyenne pondérée exponentiellement des mesures ou $\Sigma_{i\in[0..t[}\alpha^i \times m_{t-i}$

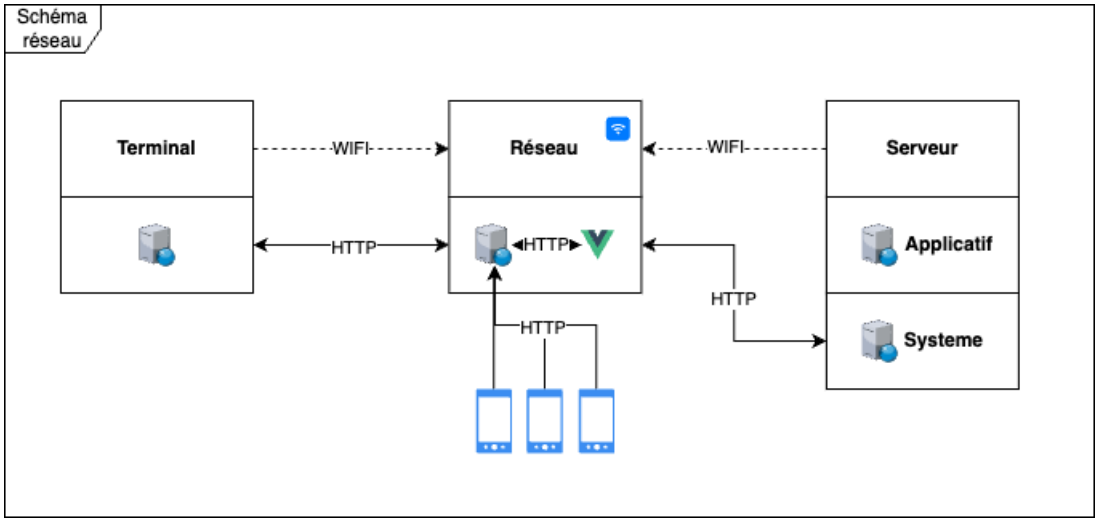

FIGURE 10 – Schéma réseau

que peut délivrer le port GPIO dans sa totalité.

Le bloc fonctionnel avec affichage sur un ruban NeoPixel est donc très différent du bloc fonctionnel avec affichage sur quelques leds monocolores. Seule la première led programmable du ruban est physiquement connectée au port GPIO. Ensuite, le ruban est alimenté séparément de la carte Raspberry Pi. Enfin, là où le script du montage naïf identifie chaque led à une broche différente, le script du montage réaliste identifie tout le ruban à une unique broche. La figure 9 représente une réalisation de ce bloc fonctionnel.

Le programme permettant de relier l'affichage des leds avec les mesures de température et de puissance réutilise les programmes réalisés plus haut pour récupérer les valeurs des mesures. Il utilise la librairie **rpi_ws281x**[17] pour contrôler le ruban NeoPixel. Mais afin de rendre plus simple son utilisation nous avons réalisé une couche supérieure d'abstraction de fonctions telles que afficher la température ou la puissance.

3.4 Réseau

3.4.1 Routeur

L'établissement de la communication entre les trois Raspberry Pi est crucial pour le bon fonctionnement du système DémoJ, et pour cela, nous avons configuré le module **Réseau** en tant que **Hotspot Wifi**. Cette configuration, réalisée à l'aide de la bibliothèque **RaspAp**[18], permet aux modules **Terminal** et **Serveur** de se connecter au réseau constitué par le Raspberry Pi **Réseau**. Nous utilisons des adresses IP statiques pour faciliter la communication entre les modules dans les scripts Python.

Nous avons délibérément opté pour des adresses IP statiques, écartant l'utilisation du DHCP[19], car dans le réseau privé du système DémoJ, la connaissance préalable des adresses IP était essentielle pour établir une communication fiable via le protocole HTTP[20], garantissant ainsi une connectivité stable. Il a également été crucial de garantir que chaque Raspberry Pi puisse se connecter automatiquement au signal Wifi du mo-

17. rpi_ws281x : `https://github.com/jgarff/rpi_ws281x`
18. RaspAp : `https://raspap.com`
19. DHCP : *Dynamic Host Configuration Protocol*
20. HTTP : *Hypertext Transfer Protocol*

dule **Réseau**. Cette fonctionnalité est gérée directement par le système d'exploitation, assurant ainsi une connexion automatique lors du démarrage.

Enfin, nous avons envisagé la possibilité d'ajuster les paramètres spécifiques du réseau. Cela pourrait inclure la capacité d'influencer délibérément le niveau de perte de paquets, le niveau de latence du réseau, voire même la possibilité de limiter la bande passante. Ces ajustements pourraient être réalisés sans compromettre la stabilité du système.

3.4.2 Communication inter Raspberry

Nous avons pris la décision stratégique de faire communiquer les Raspberry Pi via le protocole HTTP en utilisant un serveur Web développé avec Flask [21] en Python [4]. Bien que les Websockets aient été envisagés pour obtenir des informations en temps réel depuis l'application de contrôle, nous avons choisi de ne pas les intégrer pour éviter de mélanger les protocoles de communication. Cette décision a été prise en tenant compte du fait que le rafraîchissement des informations côté client a été efficacement géré, garantissant l'intégrité et la disponibilité des données sans nécessiter de complexités supplémentaires.

3.5 Application DémoJ Connect

L'application web de contrôle porte le nom de **DemoJ Connect**. Ce choix s'explique par la volonté de souligner la connexion entre les différents modules et de renforcer la cohérence de la marque.

3.5.1 Architecture de l'application

Pour garantir une expérience utilisateur fluide, nous avons opté pour une architecture de type **SPA** [22] plutôt que de développer une application mobile dédiée. Cette décision s'est appuyée sur la plus grande accessibilité d'une application web depuis n'importe quel dispositif via un navigateur, évitant ainsi les contraintes liées au développement mobile et permettant une accessibilité universelle.

La représentation des données en **JSON** [23] a été sélectionnée en raison de sa haute modularité et de son adéquation avec Javascript, le langage principal utilisé dans l'application DemoJ Connect. Ce choix a été précédé par une période de réflexion visant à trouver une structure complète, tout en restant facile à maintenir et à mettre à jour.

Un fichier JSON, contenant des informations cruciales telles que le statut de connexion, les paramètres des modules, et les détails des scénarios, reste sous la responsabilité du module Réseau. Ce dernier est chargé de le mettre à jour via les requêtes HTTP provenant des autres modules et de l'application DemoJ Connect.

3.5.2 Requêtes et scénarios

Afin de mettre en place l'éxecution des scénarios depuis DemoJ Connect, il nous fallait décider de la façon dont les modules et l'application communiqueront.

Ainsi, DemoJ Connect a la responsabilité de gérer le fichier de base de données puisque l'application et ce dernier partagent le même format (JSON). De ce fait, les routes dispo-

21. Flask : `https://flask.palletsprojects.com/en/3.0.x/`
22. SPA : *Single Page Application*
23. JSON : *Javascript Object Notation*

nibles sur le serveur correspondent à la récupération du fichier de base de données JSON, à l'exécution d'un scénario ainsi qu'à la modification des paramètres des modules.

Le fichier étant suffisamment léger, il ne s'agit pas d'un problème de le transmettre via le réseau.

3.5.3 Environnement de développement

En l'absence des composants physiques du projet, nous avons utilisé des **machines virtuelles** Linux pour simuler les Raspberry Pi manquants. Ces machines ont été configurées pour communiquer sur le même réseau, similaire à celui qui sera émis par le module réseau. Cette approche a permis de progresser dans la mise en place de la partie réseau et de l'application DemoJ Connect.

4 Conclusion

Le projet DémoJ consiste à fournir une maquette programmable de système informatique pour faire des démonstrations d'impact énergétique dans un cadre de médiation ou d'enseignement scientifique. Nous avons exploré avec notre client ce que cela impliquait afin de rester aligné avec ses attentes. En effet, notre client a montré de grandes attentes concernant la réalisation d'un prototype du système DémoJ, y compris en communiquant hors de notre formation et même hors de l'université. Un rendez-vous est même pris pour une présentation publique au printemps prochain.

C'est un projet qui comporte des aspects de packaging externe, des aspects systèmes et même électroniques, ainsi que des aspects applicatifs. Dans cette première phase, nous avons exploré ce qui nous semblait être des verrous technologiques par rapport à notre spécialisation. Pour tous ces verrous, nous avons réalisé des blocs fonctionnels qui prototypent des solutions pour les lever. Nous sommes donc confiants dans notre capacité à mener à bien le développement du système DémoJ.

Les prochains blocs fonctionnels à réaliser sont des assemblages ou des versions plus approfondies des blocs fonctionnels déjà réalisés : carte électronique principale de chaque boîte (voir centre de la figure 7), finalisation des boîtes, y-compris leur décoration, l'application de contrôle DemoJ Connect, les applications types pour les scénarios de médiation et le packaging de l'ensemble (documentation utilisateur, supports de médiation, ...).

Remerciements : Nous remercions Mme Camille Bisson, manager du fablab de Beaulieu, et M. Régis Legave, responsable de l'atelier d'électronique de l'ISTIC, pour leur accueil, et leurs conseils concernant des technologies que nous ne connaissons pas bien.

Références

[1] Kent Beck et al. *Manifesto for Agile Software Development*. 2001. URL : `https://agilemanifesto.org`.

[2] Dominique Cardon et al. *Atlas du numérique*. Les presses de SciencesPo, 2023.

[3] GreenIT. *The environmental footprint of the digital world*. 2019. URL : `https://www.greenit.fr/wp-content/uploads/2019/11/GREENIT_EENM_etude_EN_accessible.pdf`.

[4] Miguel GRINBERG. *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc., 2018.

[5] Udit GUPTA et al. « Chasing Carbon: The Elusive Environmental Footprint of Computing ». In : *International Symposium on High-Performance Computer Architecture (HPCA 2021)*. 2021.

[6] Gareth HALFACREE. *LE GUIDE OFFICIEL du débutant Raspberry Pi*. Raspberry Pi Press, 2020. URL : https://www.framboise314.fr/docs/BeginnersGuide-4thEd-FR_v5.pdf.

[7] Kashif Nizam KHAN et al. « RAPL in Action: Experiences in Using RAPL for Power Measurements ». In : *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2 (2018).

[8] Éric LUYCKX. *L'énergie expliquée aux enfants*. Ministère de la Région Wallonne, 2004. URL : https://www.academia.edu/308998/L%C3%A9nergie_Expliqu%C3%A9e_Aux_Enfants.

[9] Ray OLDENBURG. *The Great Good Place: Cafés Coffee Shops Bookstores Bars Hair Salons and Other Hangouts at the Heart of a Community*. Group West, 1999.

[10] The DemoJ PROJECT. *Rapport final*. 2023.

[11] The Shift PROJECT. *Expanding digital sufficiency*. 2021. URL : https://theshiftproject.org/en/article/implementing-digital-sufficiency/.