ÉCOLE NATIONALE D'INGÉNIEURS DE BREST

Ifremer

Rapport de stage assistant ingénieur

29 mars – 9 juillet 2021

GUICHAOUA Tristan

IFREMER *Institut Français de Recherche pour l'Exploitation de la Mer*

Tuteur : Sébastien Prigent

Intitulé du stage : Développement d'une interface utilisateur pour le capteur de température grand fond à affichage temps réel PINT

IFREMER – Centre de Bretagne
Service Ingénierie et Instrumentation Marine
1625 Route de Sainte-Anne, CS 10070
29280 Plouzané

Résumé

Dans le cadre de ma 4e année à l'École Nationale d'Ingénieurs de Brest, j'ai effectué mon stage assistant-ingénieur au sein de l'institut français de recherche pour l'exploitation de la mer (Ifremer), d'une durée de 15 semaines du 29 mars au 9 juillet 2021 ayant l'intitulé « Interface utilisateur d'un capteur de température grand fond à affichage temps réel ».

J'ai donc intégré le service ingénierie et instrumentation marine (SI2M), où j'ai développé l'interface utilisateur pour le capteur de température grand fond PINT.

Remerciements

Je souhaite avant tout remercier mon tuteur de stage Sébastien Prigent ainsi que Hugo Schaaf pour m’avoir accompagné et conseillé tout au long de mon stage.

J’adresse aussi mes remerciements aux membres de l’unité recherches et développements technologiques de l’Ifremer, et tout particulièrement le service ingénierie et instrumentation marine pour leur accueil ainsi que pour leur sympathie.

Table des matières

Glossaire

API	Application Programming Interface	Interface de programmation applicative
BLE	Bluetooth Low Energy	Bluetooth à basse consommation
BMS	Battery Management System	Système de gestion/protection de la batterie
FTP	File Transfert Protocol	Protocole de transfert de fichier
GPL	GNU General Public License	Licence publique générale GNU
LGPL	GNU Lesser General Public License	Licence publique générale limitée GNU
MFTP	Minimal File Transfert Protocole	Protocole de transfert de fichier minimal
PINT	disPlay of In situ Temperature	Affichage de la température in situ
PMMA	PolyMethyl MethacrylAte	Polyméthacrylate de méthyle – *Plexiglas*
ROV	Remotely Operated underwater Vehicle	Véhicule sous-marin téléguidé
TS	TypeScript	
UML	Unified Modeling Language	Langage de Modélisation Unifié
UUID	Universally Inique IDentifier	Identifiant unique universel

Index des figures

1 Introduction

L'IFREMER (*Institut Français de Recherche pour l'Exploitation de la MER*) est un institut public regroupant chercheurs, ingénieurs et techniciens travaillant sur tous les domaines du milieu marin. L'objectif de cet institut est de comprendre et de surveiller, le fonctionnement et l'évolution des océans et populations marines, depuis les zones côtières vers les zones hauturières (zones maritimes éloignées des côtes), de la surface des océans aux grands fonds.

Sur la dorsale médio-atlantique, à la jonction de deux plaques tectoniques, peuvent se trouver des sources hydrothermales (Figure 1) aussi appelées fumeurs. Ces sources d'eau chaude situées en grandes profondeurs (jusqu'à 5 000 m) sont provoquées par une sortie de magma qui se fissure au contact de l'eau froide. L'eau de mer s'engouffre alors dans les fentes et rencontre lors de sa descente à plusieurs centaines de mètres de profondeurs des zones très chaudes pouvant atteindre 400 °C. Cette eau chauffée beaucoup moins dense, remonte vers la surface et jaillit sous forme de fumeur, évacuant ainsi une partie de la chaleur interne de la Terre.

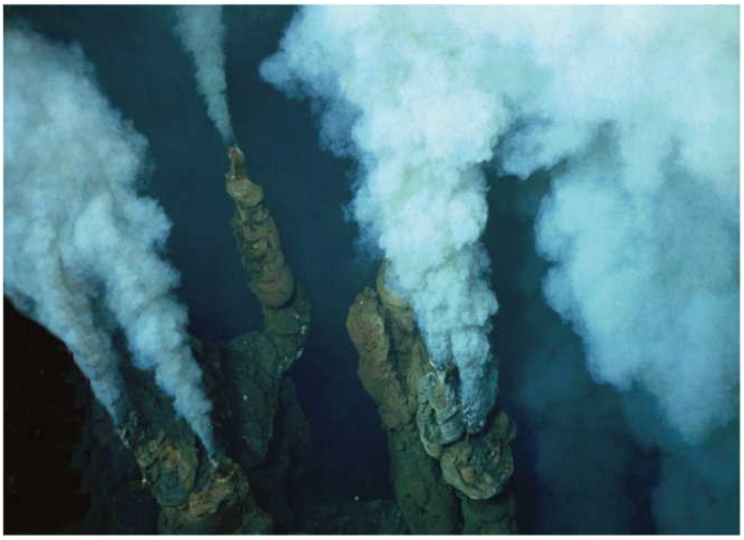


Figure 1: Fumeurs actifs dans le bassin de Lau (îles Tonga, Sud-Ouest Pacifique)

Le fluide chaud au contact de l'eau de mer froide va alors évacuer une grande quantité de minéraux représentant la base de la chaîne alimentaire marine. Cela est appelé la production primaire. Ces sources hydrothermales représenteraient entre 0,1 et 1 % de la production primaire totale des océans, pouvant atteindre 25 % dans les océans profonds (Roussel *et al.*, 2020). Ces milieux extrêmes et hostiles étant très peu connus, le *Laboratoire de Microbiologie des Environnements Extrêmes* (LMEE), de l'unité Étude des Écosystèmes Profonds du département *REM*, étudie ces environnements très spécifiques. Dans le but de caractériser la faune microbienne de ces résurgences, des prélèvements de fluides sont réalisés sur les fumeurs. Pour ce faire, Erwan Roussel (géomicrobiologiste du laboratoire LMEE) a développé un échantillonneur de fluide isotherme isobare IBIS (IsoBaric Isothermal fluidSampler) (Figure 2 et Figure 3). En effet, l'objectif est de remonter la faune microbienne prélevée au plus proche de son environnement naturel, à pression constante.

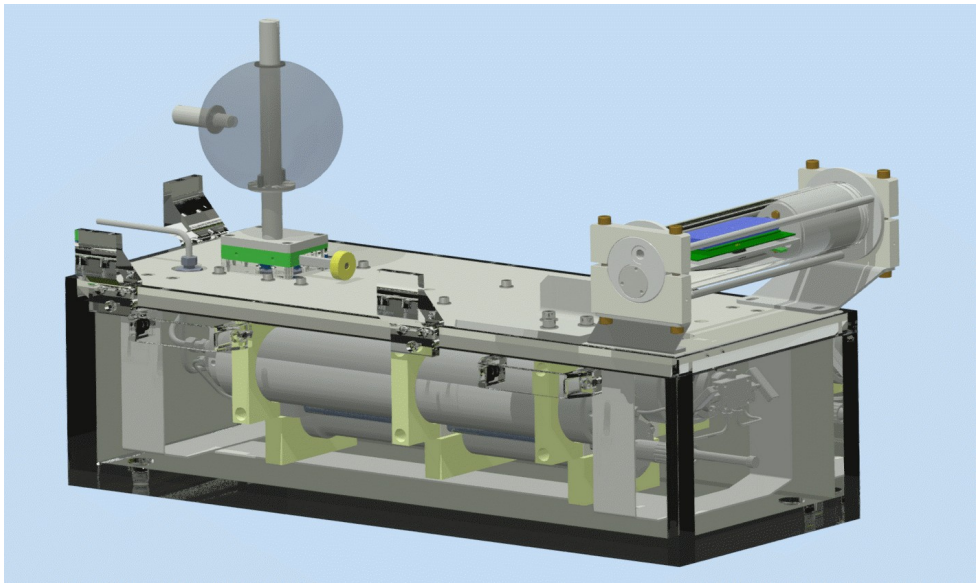


Figure 2: Vue 3D de l'échantillonneur IBIS

La température exacte de l'endroit où sont effectués les prélèvements est un paramètre important pour les scientifiques qui étudient ces échantillons. L'unité SI2M a alors été sollicitée pour développer un capteur de température, pouvant évoluer jusqu'à 6 000 m de profondeur, permettant l'affichage en temps réel de la température mesurée. Les tests du prototype, lors de la campagne *Momarsat* 2018 sur la ride médio-atlantique, ont permis de valider le fonctionnement à 1 700 m de fond. En effet, PINT a pu être déployé grâce au ROV VICTOR 6000 de l'Ifremer sur le site de Lucky Strike. Le bon fonctionnement a permis d'ouvrir d'autres perspectives et de laisser entrevoir d'autres utilisations, moyennant quelques modifications.

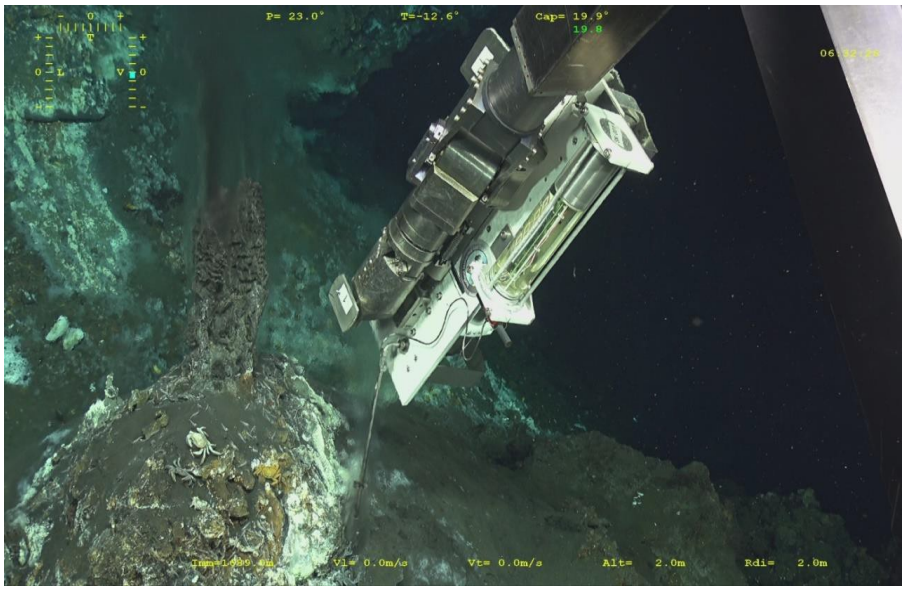


Figure 3: Première plongée d'essai du capteur de température PINT V1, seul, lors de la campagne Momarsat 2018

2 Présentation d’Ifremer

2.1 Statut et missions

L’*Institut Français de Recherche pour l’Exploitation de la MER* (IFREMER) a été créé le 5 juin 1984, suite à la fusion du *Centre National pour l’EXploitation des Océans* (CNEXO) et de l’*Institut Scientifique et Technique des Pêches Maritimes* (ISTPM). L’établissement possède le statut EPIC (Établissement Public à Caractère Industriel et Commercial). Placé sous la tutelle conjointe des ministères de « l’Enseignement Supérieur, de la Recherche et de l'Innovation » ainsi que du « Ministère de la Transition écologique et solidaire (MTES) », l’Ifremer concentre son activité sur la recherche en sciences marines, avec une contribution à l’échelle nationale et européenne. Cet institut a pour objectifs principaux d’étudier et valoriser les océans auprès du public, ainsi qu’observer l’évolution de ceux-ci pour mieux comprendre l’impact sur notre planète.

Figure 4: Le centre de Bretagne IFREMER

2.2 Organisation et implantation

Sous la présidence de François Houllier, l'IFREMER emploie près de 1 500 personnes (dont 595 ingénieurs/chercheurs). L'Institut est composé d'une direction fonctionnelle (DRH, Direction des Affaires…), d'une direction scientifique et de 4 départements scientifiques en lien avec différents domaines du monde marin :

- Ressources Biologiques et Environnement (RBE)
- Ressource physique et Ecosystèmes de fond de Mer (REM)
- Océanographie et Dynamique d'Ecosystèmes (ODE)
- Infrastructures Marines et Numériques (IMN)

En lien direct avec la mer, l'Ifremer dispose de 5 centres principaux répartis sur le territoire français, ainsi que d'une multitude de stations qui opèrent sur les différentes façades maritimes. L'Ifremer couvre une large zone d'action par sa présence sur le continent ainsi qu'en outre-mer. Le siège de l'Ifremer anciennement à Issy-les-Moulineaux a été transféré sur le site de Brest en 2019.

Figure 5: Implantation de l'IFREMER en métropole

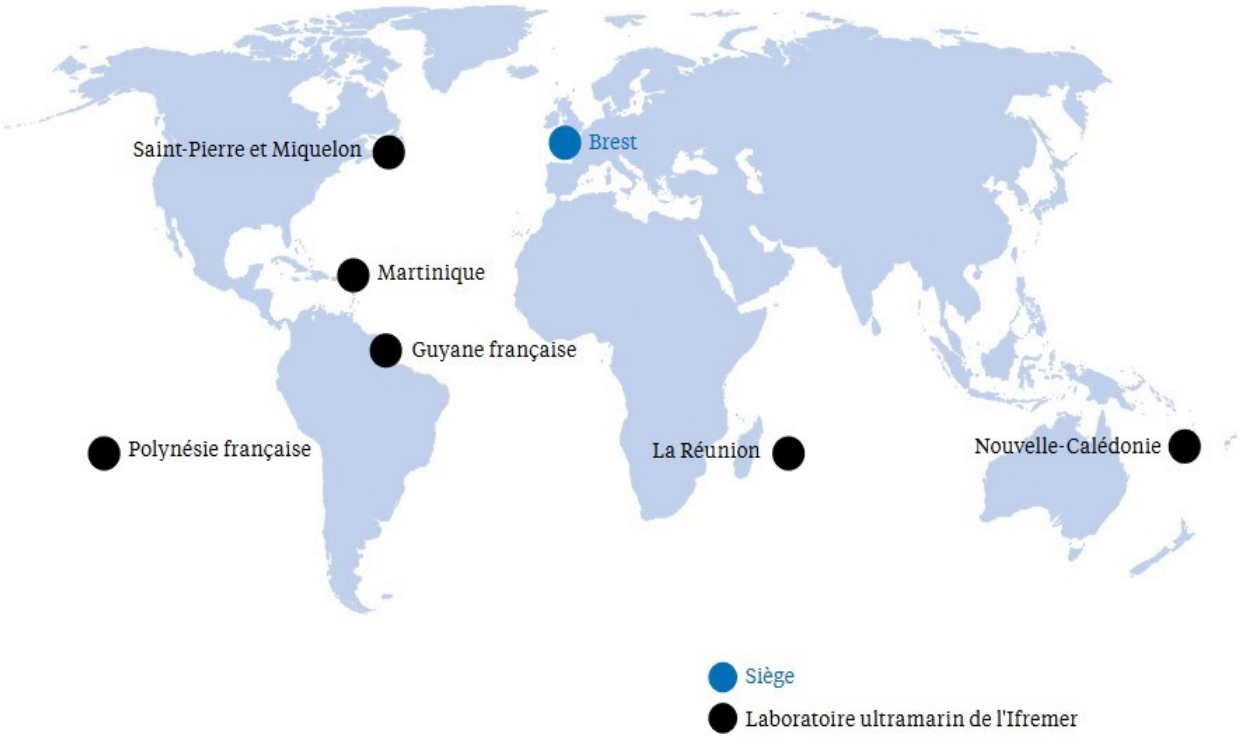


Figure 6: Implantation de l'IFREMER en Outre-Mer

2.3 Le service SI2M

Le *Service Ingénierie et Instrumentation Marine* (SI2M) est chargé des études et de la conception des systèmes instrumentaux, des observatoires océanographiques et des équipements destinés aux engins sous-marins. Le service SI2M est ainsi un acteur important dans la compréhension et l'exploitation des ressources marines, mais aussi dans l'amélioration et l'innovation concernant le matériel de surveillance sous-marine. C'est au sein de ce service que j'ai effectué mon stage. Il travaille en lien avec les unités scientifiques de l'institut, les établissements de recherche nationaux et étrangers, et les industriels. Il intervient de l'estran jusque dans les grands fonds marins en concevant des systèmes miniaturisés ou pouvant atteindre plusieurs dizaines de tonnes. Ce service compte pas moins de 35 personnes dont 21 ingénieurs et 14 techniciens. Il est divisé en 3 équipes :

- Le **Bureau d'études en mécanique** réalise les études, les dimensionnements et les dossiers de plans
- Le **Laboratoire électronique et informatique embarquée** réalise le câblage, l'intégration des systèmes et conçoit les architectures, les cartes électroniques, les logiciels adaptés aux besoins. Il développe autant que possible des systèmes d'acquisition génériques, portables sur un maximum de projets de l'institut.
- L'**Atelier Prototypes** monte et met au point les prototypes. Il prépare les essais tels que ceux effectués en mer. Il dispose de moyens de fabrication (usinage, chaudronnerie et impression 3D) et peut ainsi réaliser certaines pièces mécaniques où suivre des travaux de sous-traitance.

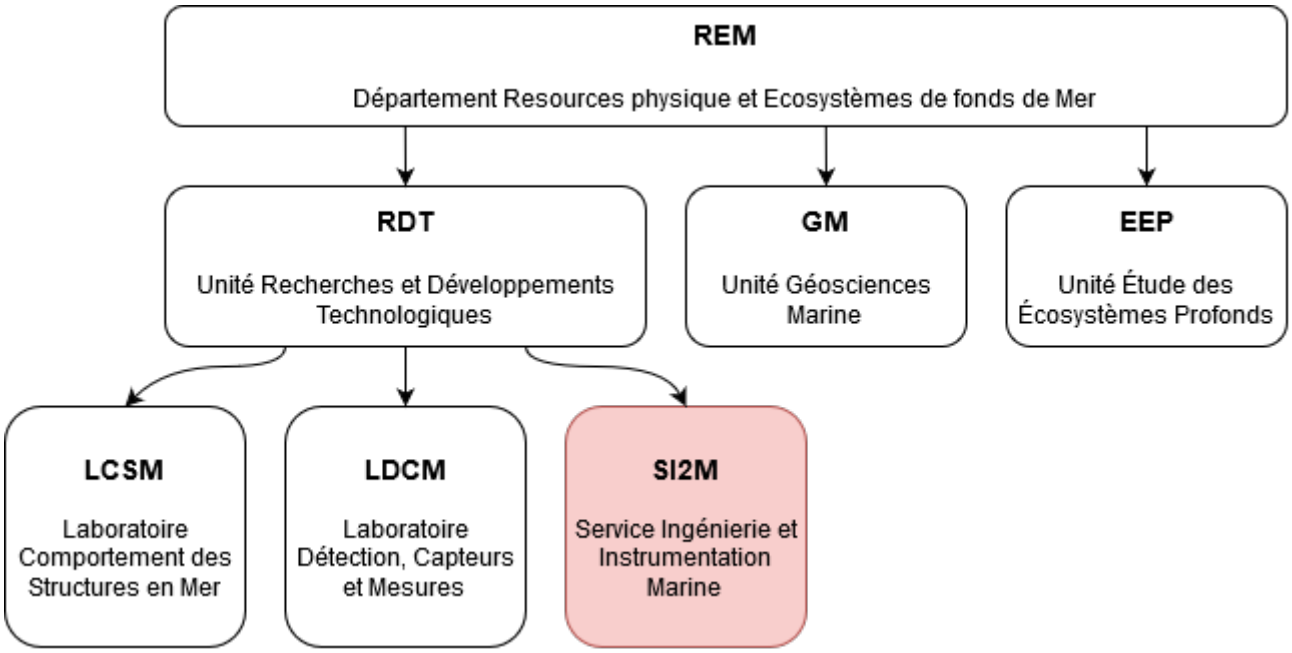


Figure 7: Organigramme du département REM

3 Présentation de la problématique

3.1 Contexte

Le besoin du microbiologiste Erwan Roussel consiste à connaître la température de ses échantillons au moment de leur prélèvement avec IBIS. L'idée est de l'équiper d'un capteur, permettant la visualisation en temps réel de la température mesurée. Pour simplifier au maximum la mise en service ainsi que l'utilisation, le capteur doit être autonome en énergie. Les scientifiques accordent une attention toute particulière à la précision de la température de prélèvement. C'est pourquoi la zone de mesure doit être la plus proche possible de la zone de prélèvement de l'échantillon. Le différentiel de température entre les fumeurs (~300 °C) et l'eau de mer environnante (~5 °C), induit qu'un déplacement de quelques centimètres par rapport à l'endroit du prélèvement peut conduire à une variation de plusieurs dizaines de degrés.

Les scientifiques doivent aussi pouvoir lire la température sur le capteur depuis le sous-marin habitable de l'Ifremer, le NAUTILE, ou depuis la caméra du robot sous-marin, le ROV VICTOR 6000. Un moyen d'affichage permettant une lecture optimale de la température à distance moyenne (quelques mètres) même à travers du moirage est donc de rigueur.

Un tel système répondant aux besoins du projet et capable de fonctionner dans un environnement aussi hostile n'existe pas dans le commerce. Le service SI2M a donc été sollicité pour développer ce capteur.

3.2 Présentation du PINT V1

L'équipe projet est formée de Sébastien Prigent et Jean-Romain Lagadec, respectivement ingénieur électronicien en instrumentation marine et ingénieur en conception mécanique. Ce capteur a été imaginé et conçu pour répondre aux besoins d'Erwan Roussel : PINT V1, un super thermomètre en quelque sorte. C'est à un cahier des charges exigeant auquel PINT V1 devait répondre :

- Autonomie de fonctionnement : 72H
- Précision de mesure : ±0,5°C en dessous de 50°C, sinon ±1°C
- Échantillonnage de la température à 10Hz
- Affichage de la température à 1Hz
- Résister à une plongée à 6 000m de profondeur (ie. 600 bars de pression)
- Installation sur IBIS
- Rechargeable
- Simple d'utilisation

La carte électronique intègre une solution à microcontrôleur, un grand afficheur 7 segments, un Convertisseur Analogique Numérique (CAN) ainsi qu'un système de recharge de batterie. La source d'alimentation, quant à elle, est une batterie lithium-ion. Le tout est conditionné dans un tube de PMMA. Cependant, pour que le système résiste aux fortes pressions, la partie électronique est

coulée dans la résine transparente et la batterie, qui doit pouvoir être changée au besoin, est protégée dans une pièce en aluminium (Figure 8).

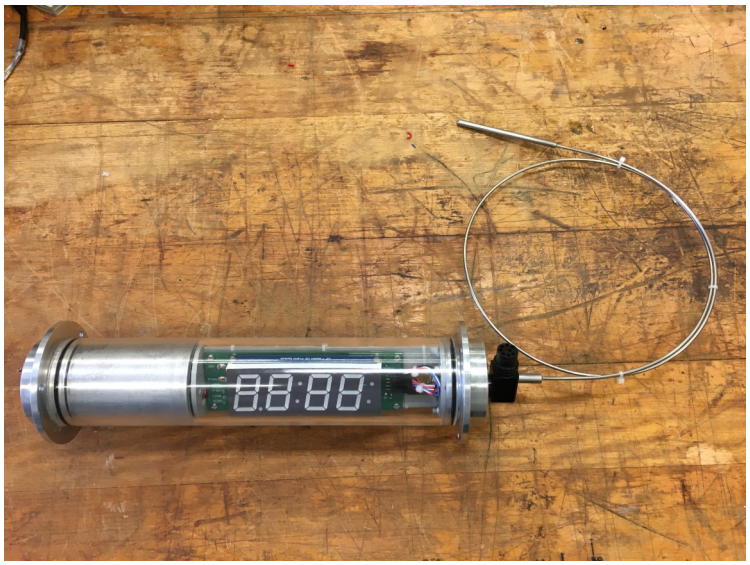


Figure 8: Le capteur de température autonome PINT V1

La précision de mesure est garantie par l'utilisation d'une sonde PT100. Ces sondes à base de platine ont pour propriété la variation linéaire de leur résistance interne en fonction de la température. La mesure de cette résistance est convertie par un Convertisseur Analogique Numérique (CAN) qui rend l'information exploitable par le microcontrôleur. Le rechargement de la batterie et la programmation du capteur se fait grâce à un connecteur étanche de marque *Subconn*. De cette manière, il n'y a pas besoin de démonter le système pour pouvoir le recharger, ce qui a été très apprécié par les utilisateurs. Le connecteur a une 3^{e} fonction qui n'est pas des moindres, puisqu'il permet l'allumage et l'extinction de PINT V1. En y vissant l'un ou l'autre bouchon, un shunt est créé entre la source d'alimentation (la batterie) et la carte électronique.

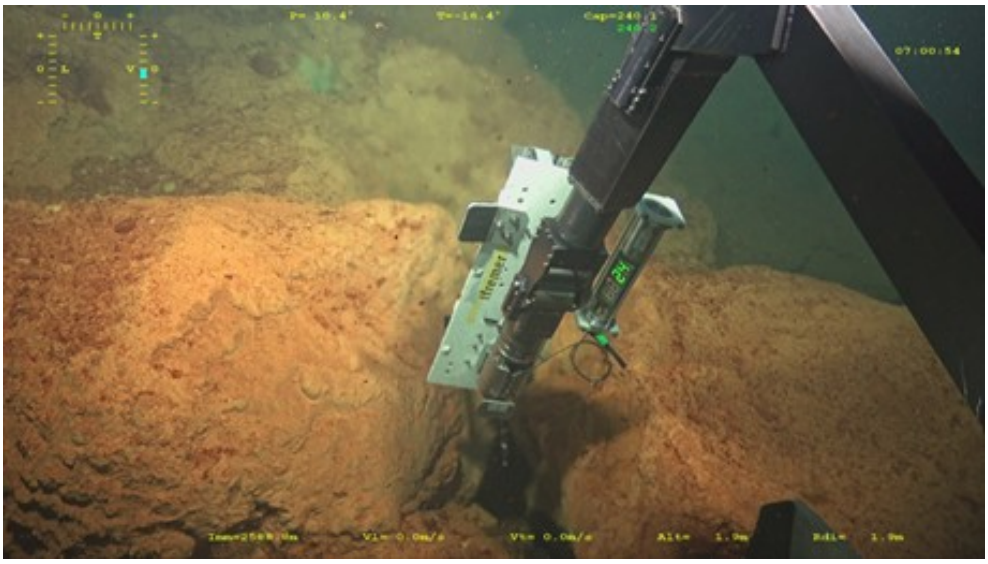


Figure 9: Le PINT V1 en action

3.3 Présentation du PINT V2

La première version du capteur de température, PINT V1, a été déployée à de nombreuses reprises lors des campagnes ESSNAUT 2018, BICOSE 2, MOMARSAT 2018 et 2020, Géoflamme 2021. Grâce à quoi des avancées scientifiques sur la compréhension du milieu ont pu être publiées.

Cependant, la synchronisation et la datation des données avec stockage seraientt un gros plus pour faciliter le post-traitement des données. Aussi, la taille du capteur a été remise en question, car il était assez encombrant (tube d'environ 100 mm de diamètre par 500 mm de long). Erwan Roussel et son équipe ont alors reformulé un nouveau cahier des charges précisant l'évolution souhaitée pour PINT V2 afin de pallier à ces nouvelles problématiques. Il s'agit bien ici d'une évolution donc les points exposés ci-après, constituent un complément au cahier des charges d'origine.

Evolution du cahier des charges

- Stockage des données de températures
- Datation des données de température
- Données restituées au format .csv
- Récupération des données sans fil
- Rechargement de la batterie sans fil
- Communication temps réel sans fil avec le ROV
- Affichage de la température à 10Hz
- Diminution de la taille de l'afficheur
- Miniaturisation du système complet

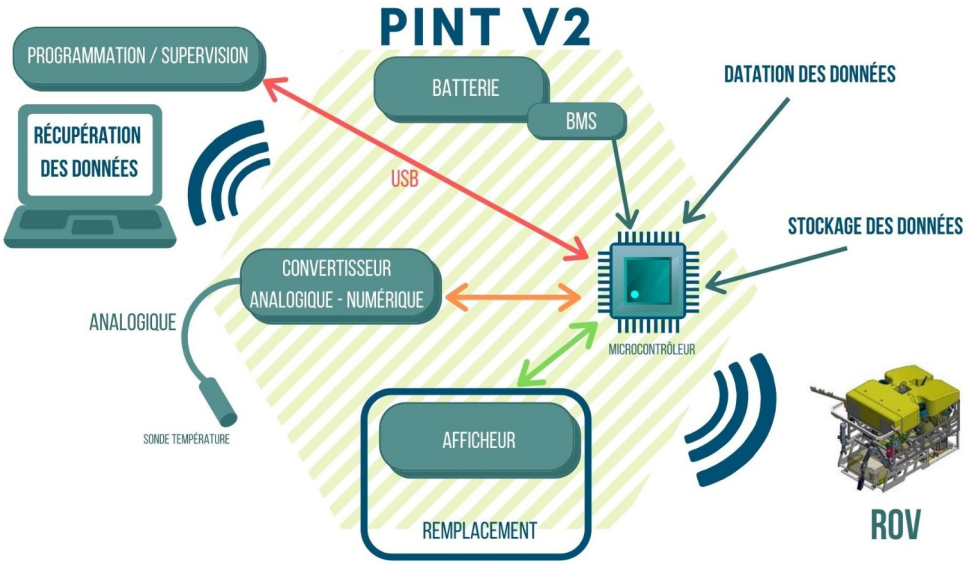


Figure 10: Schéma fonctionnel illustrant le cahier des charges

Les spécifications de la nouvelle version de PINT sont illustrées en Figure 10. L'utilisation d'une communication sans fil permet de garder une grande simplicité d'utilisation, puisqu'elle élimine toute connectique pour interagir avec le système. Les données de mission stockées dans le capteur doivent être récupérées par l'utilisateur. Malgré l'ajout de fonctionnalités, donc possiblement d'éléments consommant, l'autonomie ne doit pas être revue à la baisse. Elle est un élément essentiel, car le capteur peut rester immergé de 72 à 96 h.

Il a été convenu après discussion avec l'équipe projet, qu'il devrait être possible d'enregistrer 30 jours de données datées malgré l'autonomie théorique de 3 jours de PINT V2. Cela ajoute de la flexibilité dans l'utilisation. Effectivement, en conditions réelles, il se peut qu'on ne puisse pas récupérer les données au bout d'une mission avant d'en enchaîner une autre.

Au moment de mon stage, la version 2 est toujours en développement. La partie hardware est en phase de débogage et la partie software est en cours de réalisation.

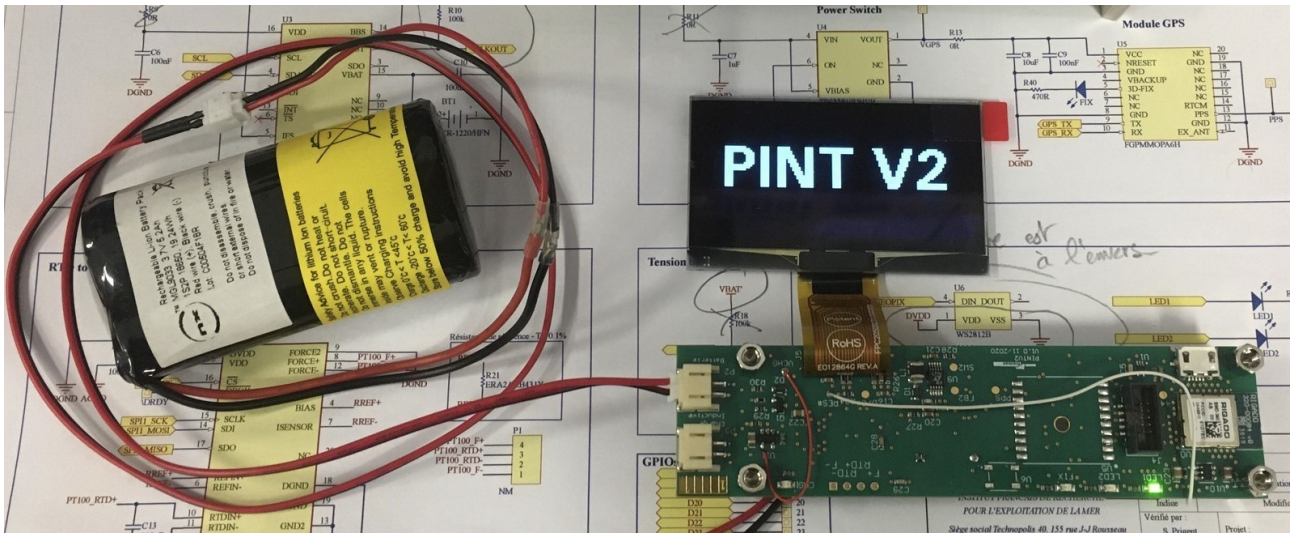


Figure 11: PINT V2

3.4 Objectif du stage

L'objectif de mon stage est la création d'une interface utilisateur sous forme d'une application PC, afin de remplir le point « récupération des données via sans-fil » et « configuration du capteur » du cahier des charges du PINT V2. Cette application devra pouvoir se connecter au capteur via la Bluetooth Low Energy, la technologie de communication sans fil utilisée par le capteur du fait de sa faible consommation d'énergie. L'application permettra à l'utilisateur de consulter les informations techniques du capteur (numéro de série, version du firmware, etc.), de consulter son état (mémoire, batterie), de le paramétrer (ex : la fréquence d'échantillonnage) et de consulter la liste des fichiers d'enregistrement, de pouvoir les télécharger et les supprimer.

L'application pourra aussi servir pour visualiser les données des fichiers téléchargés ; afficher les données sous forme d'un graphique, pouvoir se déplacer sur ce graphique, zoomer sur une portion, etc. Mais aussi avoir une analyse statistique de données (nombre d'entrées, durée, valeur minimum, maximum, moyenne, variance, etc.).

L'application se doit d'être simple et intuitif, on souhaite que l'utilisateur puisse l'utiliser sans avoir besoin d'un guide d'utilisation.

3.5 Contexte du stage

Le stage s’est déroulé en partie en présentielle et en partie en télétravail. J’étais présent sur site les lundis, mardis et vendredis, et en télétravail les mercredis et jeudis.

Hugo Schaaf, un énibien ayant réalisé son stage S8 sur la conception de la version 2 du capteur (hardware et firmware) et continuant son travail actuellement en alternance, présent sur site du mercredi au vendredi, avec qui je communiquais souvent, soit en présentielle le vendredi, soit via *Gitlab chat*, un service de messagerie instantanée.

Avec mon tuteur de stage Sébastien et Hugo, nous organisions des réunions le mercredi matin en vision-conférence, pour faire un point sur l’avancement du projet, discuter des problématiques rencontrées et fixer les prochains objectifs.

Au cours de mon stage, j’ai pu assister à deux réunions avec Christian Podeur, Herve Lintanf et Jean-Romain Lagadec, ingénieurs en conception mécanique en charge de l’intégration mécanique du capteur.

4 Choix de la technologie

La première étape avant toute chose est de réaliser l'état de l'art, c'est-à-dire rechercher les différents moyens qui existent et qui peuvent répondre à notre problématique.

4.1 Serveur web embarqué

Cette solution consiste à embarquer sur le capteur un serveur web. Le capteur agirait alors comme un point d'accès Wi-Fi auquel un appareil pourrait s'y connecter, puis à l'aide d'un simple navigateur web, pourrait accéder à une page web qui agirait alors comme un panneau de configuration pour gérer le capteur. Le gros avantage de cette solution est le fait qu'elle serait multiplatforme sans effort supplémentaire : tout appareil pouvant se connecter en Wi-Fi et disposant d'un navigateur web (PC, smartphone, tablette, …) pourrait alors interagir avec le capteur.

Néanmoins, après discussion avec l'équipe projet, cette solution impliquant soit l'ajout d'un module Wi-Fi, soit le remplacement du microcontrôleur. Cela rendrait caduques les études sur la consommation du capteur, l'autonomie étant l'un des points névralgiques de cette version.

Cette solution est, par conséquent, abandonnée.

4.2 Application et Bluetooth

Cette solution consiste à développer une application PC et d'utiliser le Bluetooth Low Energy du capteur afin de communiquer. J'ai donc listé toutes les technologies qui permettent de créer une interface graphique. Et les ai classées en fonction du langage, ainsi que les plateformes sur lesquelles on puisse exporter l'application (Figure 12).

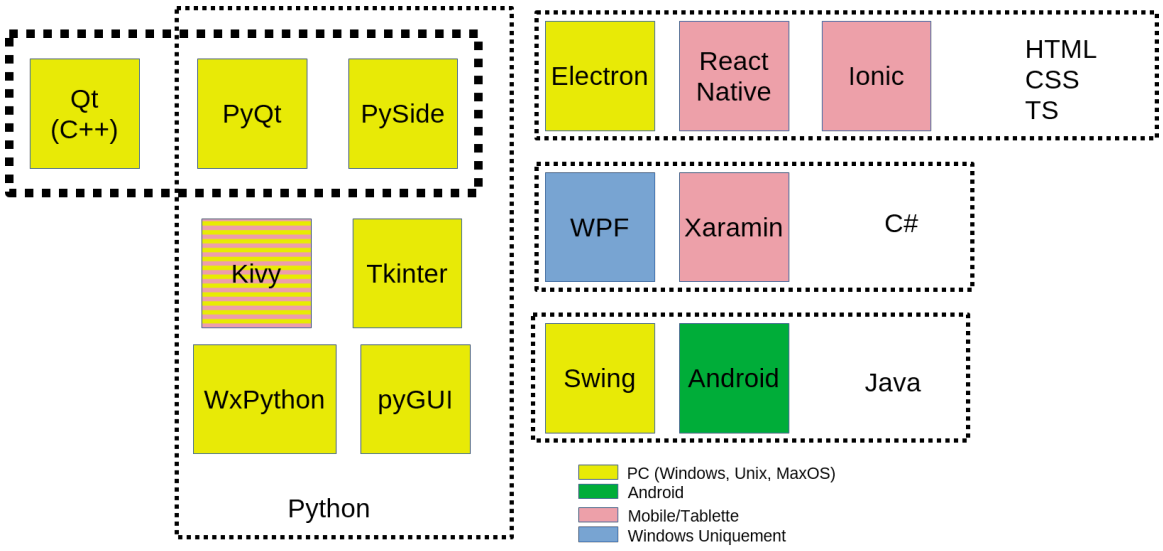


Figure 12: Différentes technologies d'interface graphique

Afin de sélectionner la technologie la plus adaptée, j'ai listé les avantages et inconvénients de chaque technologie en me basant sur les points clé suivants :

- Ergonomie
- Maintenabilité
- Expérience utilisateur

Mon choix s'est donc porté sur les technologies web (HTML, CSS, TypeScript) et sur le Python. J'ai éliminé Electron, car celle-ci étant basée sur chromium, qui consomme beaucoup de ressources à l'exécution. Je me suis tourné vers les technologies basées sur Qt (PyQt et PySide) car celle-ci est très bien documentée et très rependue, ce qui permet de trouver facilement de l'aide. Les autres technologies Python n'avait soit pas reçus de mise à jour depuis un long moment (ex : pyGUI) ou soit était destinée à des interfaces simples (ex : Tkinter).

Enfin, après quelques recherches, la seule différence notable entre PyQt et PySide est la licence d'utilisation : PyQt est sous licence GPL ou commercial (payante) et PySide est sous LGPL. La licence LGPL offre plus de liberté que la GPL. Entre autres, elle n'impose aucune obligation sur la licence sous laquelle sera publiée l'application. Contrairement à GPL qui impose que l'application soit aussi sous GPL.

Dernière étape, la version à utiliser. À ce jour, il existe trois versions de PySide : PySide basée sur Qt 4.8, PySide2 basée sur Qt 5.12+ et enfin PySide6 basée sur Qt 6.0+. Cette dernière version étant très récente (décembre 2020), j'ai choisi de privilégier la stabilité à la nouveauté avec PySide2.

Pour la version de Python, j'ai choisi la 3.7, car c'est la version la plus récente étant en support longue durée ; c'est-à-dire qu'elle ne reçoit des mise-à-jour que pour des corrections liées à la sécurité et ne subit plus d'évolution.

5 Conception

5.1 Fonctionnalités et scénarios utilisateur

Tout d'abord, nous devons définir clairement les fonctionnalités de l'application, ainsi que les scénarios utilisateur. Cela nous permettra d'avoir une base sur laquelle travailler pour définir l'apparence des différentes pages de notre interface, ainsi que la navigation entre elles.

J'ai donc listé les différentes fonctionnalités en les triant en deux catégories : les *primaires* et les *secondaires*. Les primaires étant celles qui devaient impérativement être présentes dans l'application et les secondaires celles qui serait implémenter en fonction du temps qu'il resterait.

À partir de ces fonctionnalités, j'ai établi différents scénarios utilisateur. Pour cela, je me suis basé sur le modèle *Given When Then*. Pour définir un scénario, on commence par expliciter ce que l'on souhaite réaliser à travers ce scénario. Ensuite, on définit différents critères d'acceptation, chacun représentant un cas de figure possible. Ces critères d'acception sont décrits à l'aide des mots clés Given, When, Then :

- **Given** : (Étant donné) un contexte.
- **When :** (Lorsque) l'utilisateur effectue certaines actions.
- **Then** : (Alors) on doit pouvoir constater telles conséquences.

Prenons un exemple :

Scénario : En tant qu'utilisateur, je souhaite me connecter au capteur de température.

Critère d'acceptation 1 :

- Étant donné que je me trouve sur la page d'accueil,
- Lorsque je clique sur le bouton de connexion,
- Alors le page de connexion s'ouvre et lance la recherche de périphérique.

Critère d'acceptation 2 :

- Étant donné que la recherche de périphérique est lancé,
- Lorsque j'attends 10 secondes et qu'aucun périphérique n'est détecté,
- Alors une avertissement m'indique qu'il est possible que le Bluetooth n'est pas activé ou que la capteur se trouve trop loin pour être capté.

5.2 La maquette de l’application

À partir des fonctionnalités et des scénarios utilisateur que l’on a défini, j’ai construit la maquette de l’application. Cette maquette consiste en des croquis des différentes pages de l’application. Cette maquette nous sert non pas à définir l’esthétique de l’application, mais la position des différents éléments graphiques ainsi que la façon dont l’utilisateur naviguera entre les pages de celle-ci.

Un premier diagramme nous permet de visualiser les différentes pages de l’application et la navigation possible entre elles (Figure 13).

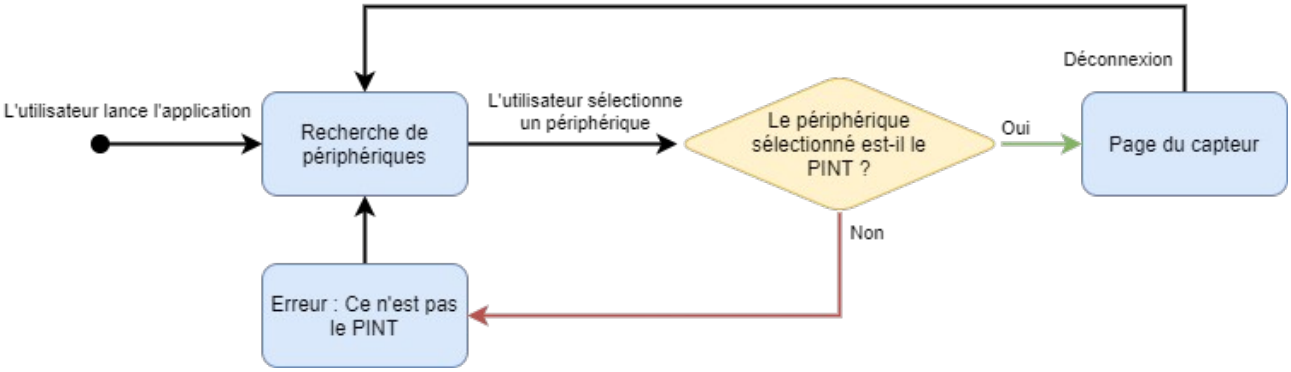


Figure 13: Première version de la navigation de l’utilisateur

Dans cette première version de la navigation, la recherche de périphérique se lance dès le lancement de l'application (Figure 14). Puis l'utilisateur choisit l'un des périphériques détectés, s'il s'agit du capteur PINT, alors la page de gestion du capteur s'affiche (Figure 16). Dans le cas contraire, un message d'erreur s'affiche (Figure 15).

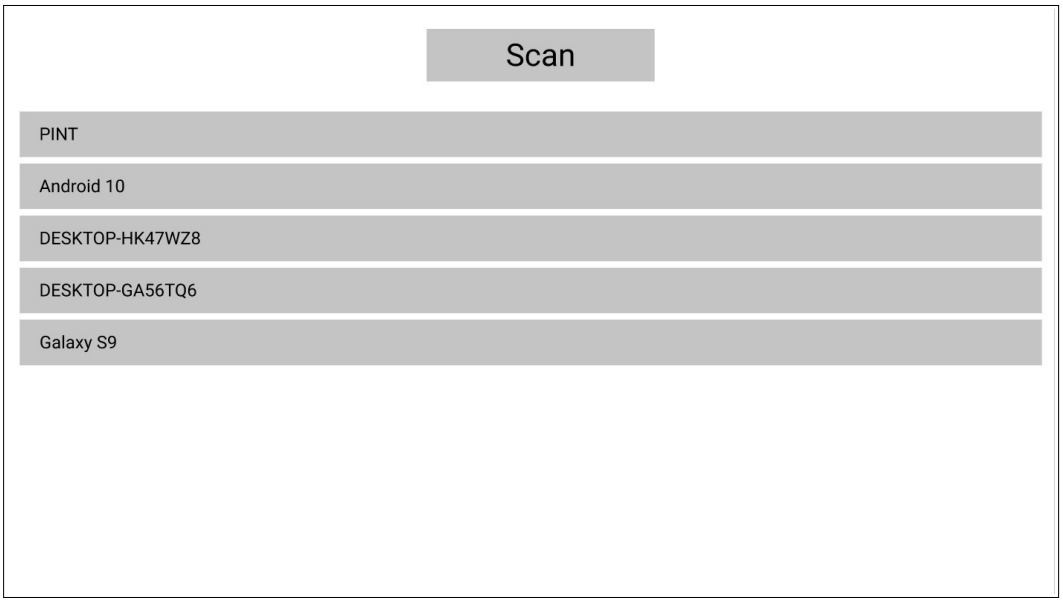


Figure 14: Maquette de la page de recherche de périphérique

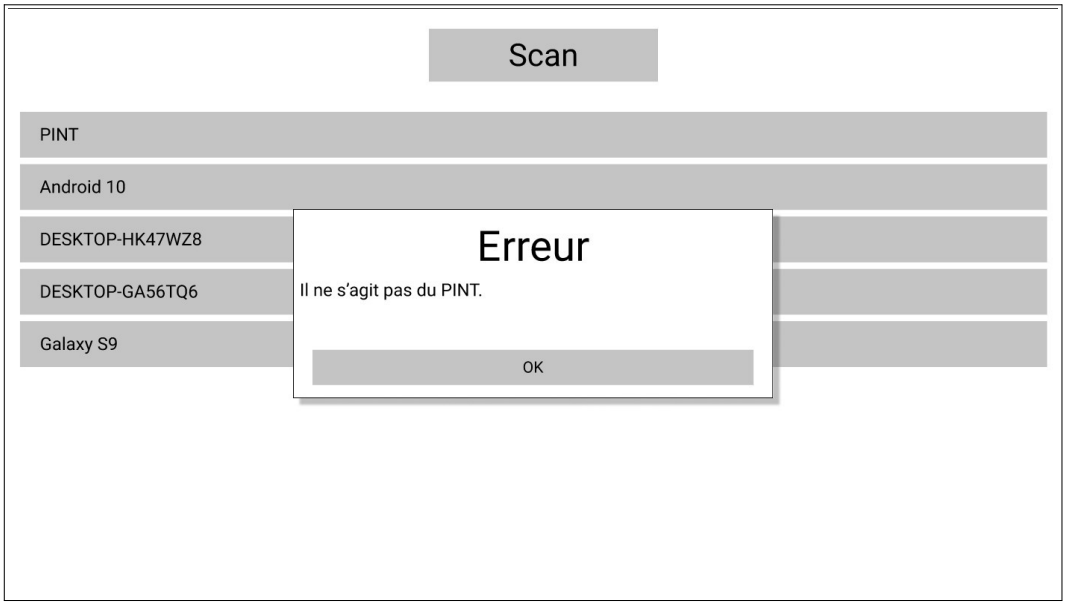


Figure 15: Maquette du message d'erreur si le périphérique est incorrect

La page du capteur (Figure 16) affiche la quantité de mémoire disponible et celle utilisé sous forme graphique, avec une barre de progression, et sous forme textuelle. Un calcule de l'autonomie mémoire, c'est-à-dire le temps qu'il reste avant que la mémoire ne soit pleine, est afficher juste en dessous.

L'utilisateur à aussi la possibilité de choisir l'emplacement de destination des fichiers qui seront téléchargés.

En dessous, se trouve la liste des fichiers présents sur le capteur. L'utilisateur à la possibilité de sélectionner une partie ou tous les fichiers pour les télécharger ou de les supprimer. Il peut aussi les télécharger ou les supprimer individuellement via les boutons qui se trouvent en face de chaque nom de fichier.

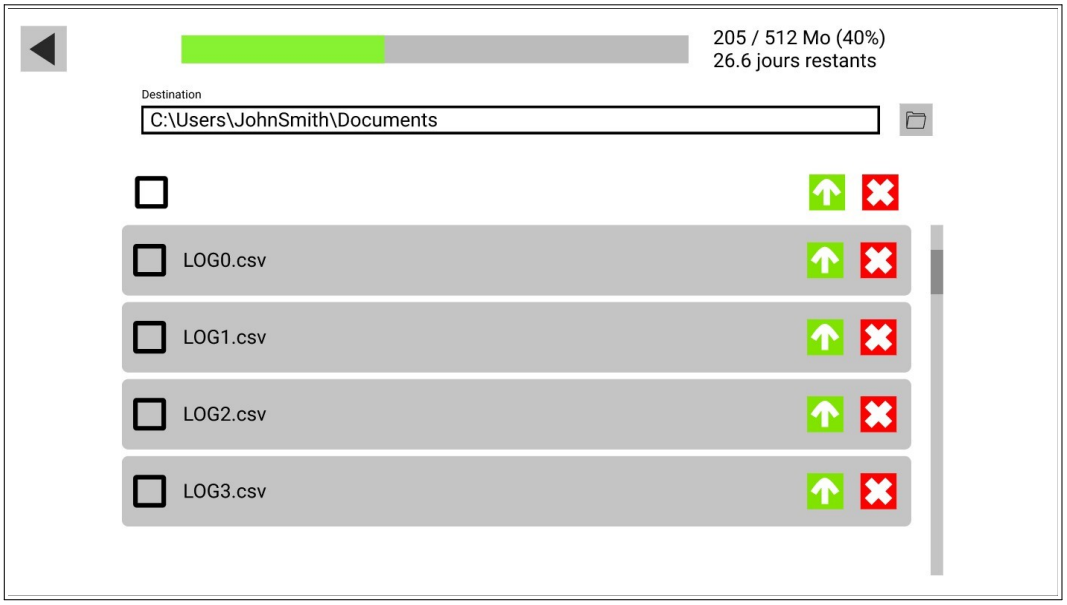


Figure 16: Maquette de la page du capteur

Nous avons ensuite décidé d'ajouter la possibilité de visualiser les fichiers de données, ce qui nous a amenés à créer une seconde version de la navigation (Figure 17).

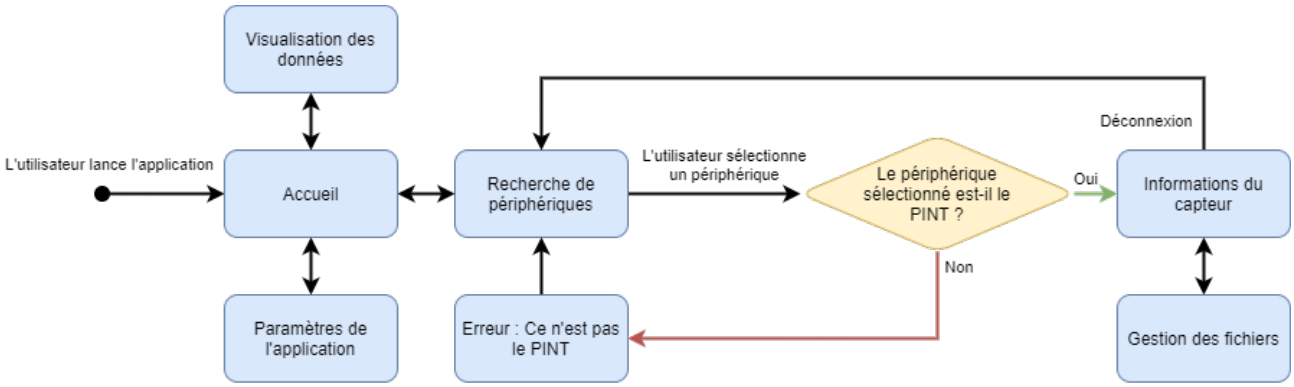


Figure 17: Seconde version de la navigation de l'utilisateur

Dans cette seconde version, l'application s'ouvre sur une page d'accueil, depuis laquelle l'utilisateur peut accéder à toutes les fonctionnalités (Figure 18). Afin de permettre à l'utilisateur d'utiliser en parallèle la connexion au capteur et d'ouvrir plusieurs visualisations de données, j'ai utilisé un système d'onglet similaire à ceux utiliser par les navigateurs web.

La page d'accueil met en évidence deux boutons, un pour chaque fonctionnalité principal de l'application. Le bouton d'accès aux paramètres de l'application est quant à lui plus discret étant un élément secondaire.

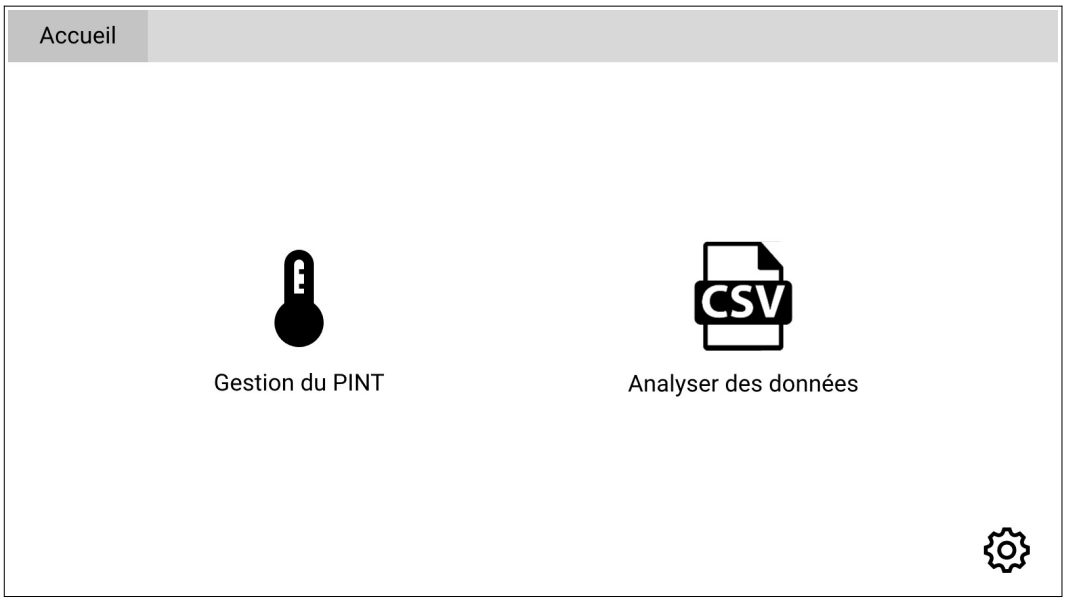


Figure 18: Maquette de la page d'accueil

La page du capteur a été séparée en deux parties pour la rendre plus lisible. La première page (Figure 19) s'affiche lorsque le capteur est connecté. Cette page est organisée autour de trois blocs. Le premier bloc nommé « Général » contient les informations sur le capteur : numéro de série, version du firmware, état de la batterie, état de la mémoire, date et heure. La date et l'heure peuvent être synchronisées avec celles de l'ordinateur avec le bouton « Synchroniser avec le PC ».

Le deuxième bloc permet de configurer le capteur. Pour le moment seule la fréquence échantillonnage est configurable. Le bouton « Sauvegarder » permet d'envoyer la configuration choisie vers le capteur. Le bouton « Reset » remet l'interface dans le même état que la configuration actuelle du capteur.

Le troisième bloc affiche une estimation de l'autonomie de la mémoire et de la batterie en se basant sur la configuration que l'utilisateur choisit.

Enfin, en bas de la page, se trouve un bouton pour se déconnecter du capteur (à gauche) et un pour accéder à la page de gestion des fichiers (Figure 20).

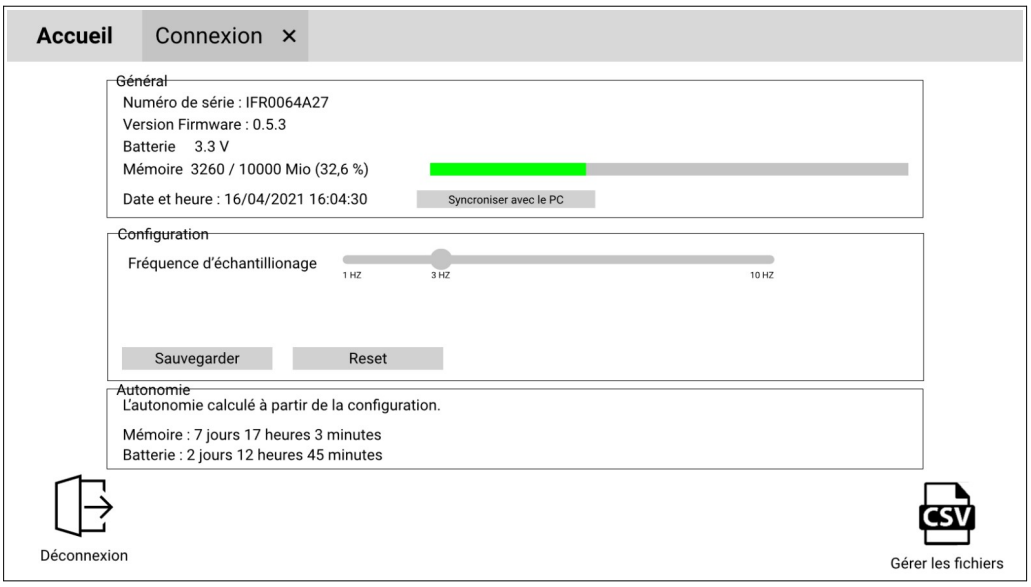


Figure 19: Maquette de la page du capteur (deuxième version)

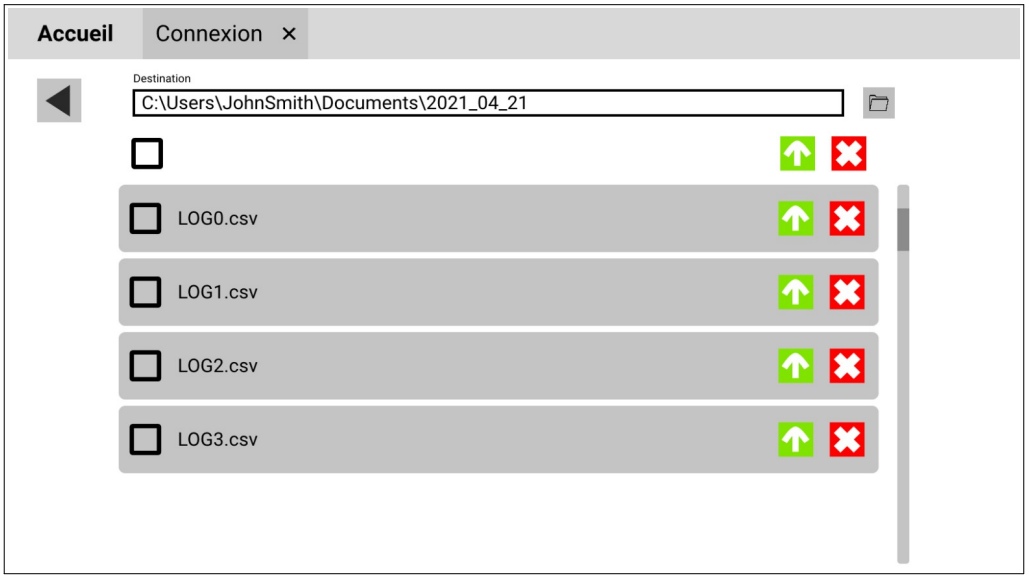


Figure 20: Maquette de la page de gestion des fichiers du capteur

La page de visualisation des données (Figure 21) affiche un graphique avec la courbe des températures qui occupe le plus d'espace possible et des informations statistique en dessous.

Figure 21: Maquette de la page de visualisation de données

5.3 Changements au moment de l'implémentation

Entre le moment où les maquettes ont été créées et le moment où les pages ont été faites, certains changements sont apparus.

L'ensemble des captures d'écran des différentes pages de l'application se trouvent en Annexe 1 : Captures d'écran de l'application.

5.3.1 Gestion des fichiers

La page de gestion des fichiers (Figure 22) comprend un bouton pour actualiser manuellement la liste des fichiers (à côté des boutons de téléchargement et de suppression). Dans le cas où une erreur se produirait lors de l'actualisation de la liste des fichiers (au moment de la connexion ou après la suppression d'un fichier), l'utilisateur est notifié de cette erreur et la liste des fichiers est vidée. L'utilisateur a alors la possibilité de retenter d'actualiser la liste.

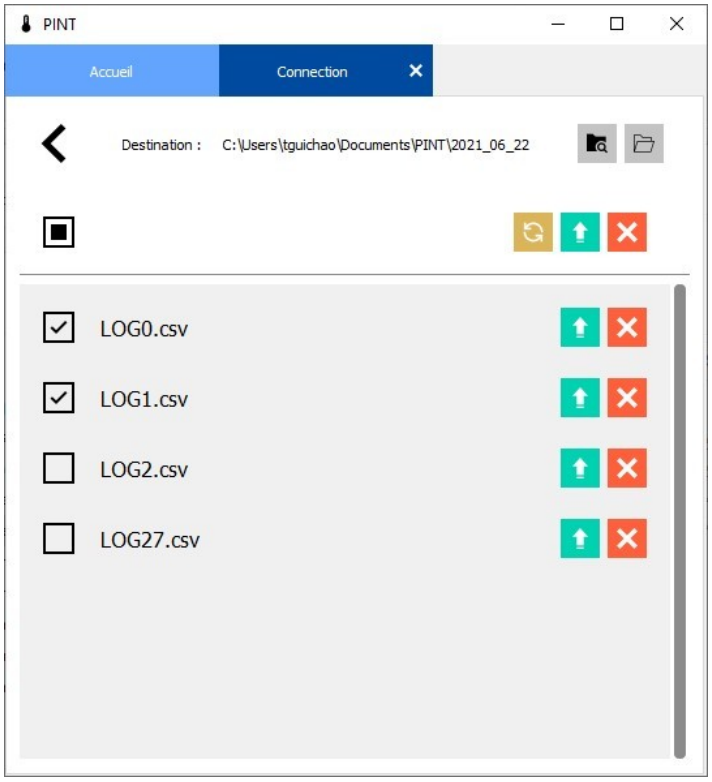


Figure 22: Capture d'écran de la page de gestion des fichiers du capteur

5.3.2 Visualiseur de données

Étant donné toutes les fonctionnalités pour naviguer dans le graphique, j'ai ajouté la liste des contrôles directement sur la page (Figure 23). Aussi, puisque les données de statistiques et la liste des contrôles ne sont pas des informations qu'il est utile d'avoir sous les yeux en permanence, il est possible de rétracter cette section pour gagner de la place pour le graphique (Figure 24).

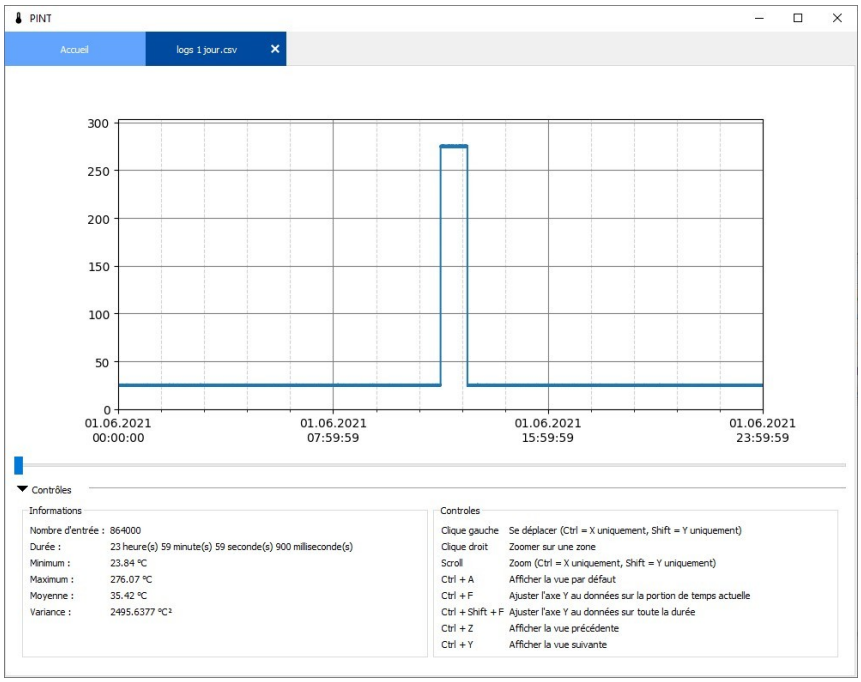


Figure 23: Capture d'écran du visualiseur de données avec les contrôles affichés

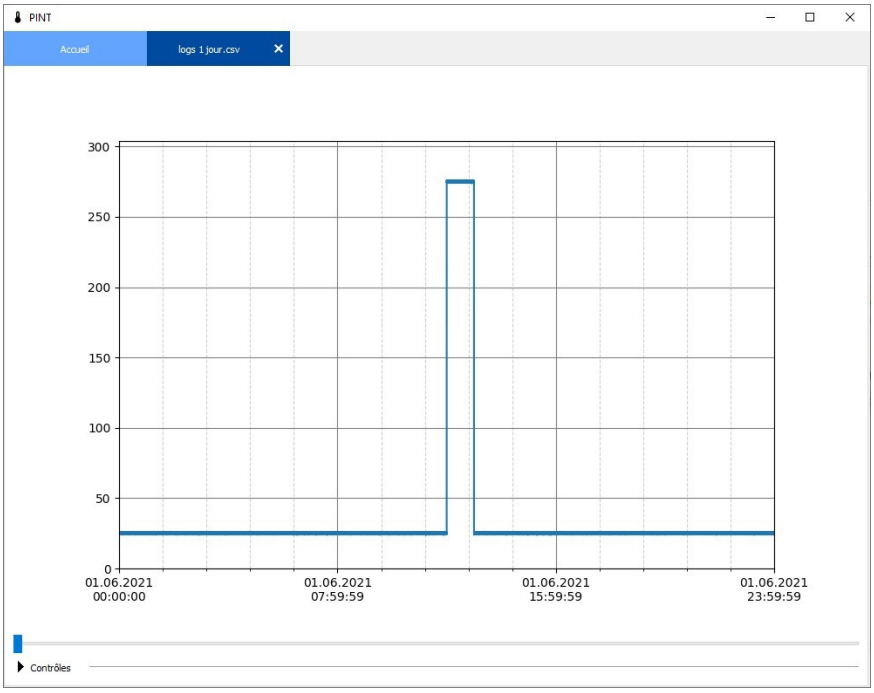


Figure 24: Capture d'écran du visualiseur de données avec les contrôles cachés

5.4 Architecture logiciel

L'un des plus gros problèmes en algorithmie est celui du couplage. Deux portions d'un programme sont dit couplées, si la modification de l'un, peut nous forcer à modifier l'autre. On cherche donc à réduire le couplage au maximum pour facilité la maintenance du code et son développement. Pour cela, il existe différentes méthodes et architectures pouvant variées en fonction du type de programme que l'on souhaite créer.

Pour la réalisation de l'interface utilisateur, j'ai choisi d'utiliser une architecture très utilisée dans les interfaces utilisateur : l'architecture Modèle-Vue-Contrôleur (MVC). Elle se présente comme suit :

- **Modèle** : Stock les données et définit comment elles peuvent être accédées et modifiées.
- **Vue** : Affiche les données et se met à jour quand les données du modèle changent.
- **Contrôleur** : Traite les requêtes de l'utilisateur et modifie le modèle en conséquence.

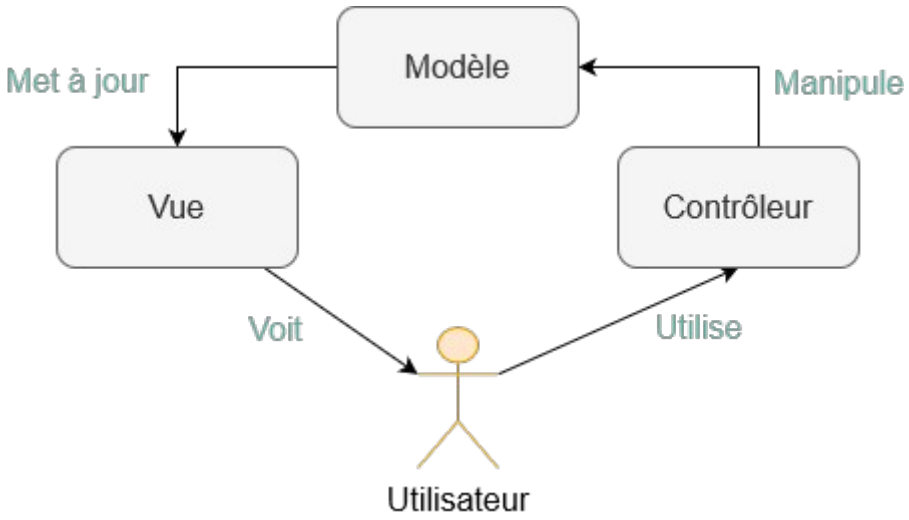


Figure 25: Schéma de l'architecture MVC

L'application est divisée en différents modules conçus de manière à être réutilisable en étant le plus indépendant possible, cela permet aussi de rendre le code le plus clair possible en regroupant les éléments semblables ensemble. Les modules sont :

- **app :** Les éléments visuels de l'application.
- **ble :** Tout ce qui est en lien avec la communication Bluetooth Low Energy.
- **pint_com :** Les services pour communiquer avec le capteur.
- **utils :** Différents modules contenant des éléments pratiques en tout genre.

5.5 Interface de communication avec le capteur

Au moment où j'ai commencé le développement de l'application, le capteur n'était pas prêt pour la connexion. Pour tout de même avancer, j'ai créé une interface de communication. Cette interface indique à l'application ce qu'elle peut faire (par exemple, récupérer le numéro de série du capteur) tout en lui cachant ce qui se passe réellement. On peut donc changer l'implémentation de l'interface (bloc bleu sur la Figure 26) sans avoir à changer quoi que ce soit dans l'application.

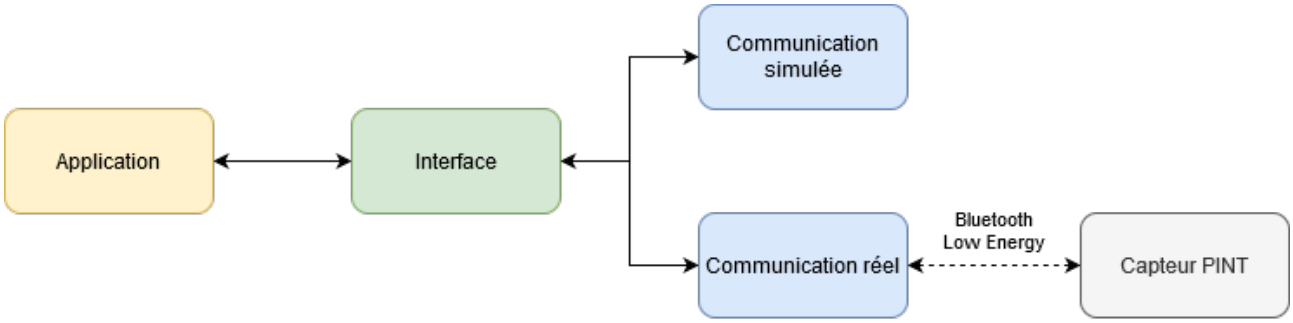


Figure 26: Schéma du fonctionnement d'une interface

Dans un premier temps, j'ai utilisé une communication simulée qui renvoyer à l'application des données factices. Puis lorsque le capteur fut prêt, j'ai remplacé le bloc simulé par le véritable bloc de communication.

Le détail de l'interface est disponible en Annexe 2 : Interface de communication avec PINT.

6 Communication avec le capteur

6.1 Communication via Bluetooth Low Energy

La technologie de communication utilisée par le capteur PINT est le Bluetooth Low Energy (BLE), une technologie de communication par ondes radio, à l'instar du Bluetooth « classique », ayant une consommation réduite. Néanmoins, sont fonctionnement est différent de ce dernier.

Dans une connexion BLE, il existe deux types d'appareil, chacun jouant un rôle différent :

- le **central** qui initie la connexion est émet des requêtes (ex : ordinateur, smartphone).
- le **périphérique** celui qui propose des services et traite les requêtes (ex : capteur de fréquence cardiaque, capteur de température).

Le périphérique possède des *caractéristiques* qui stockent une valeur, que le central peut lire ou écrire. Les caractéristiques de même nature sont regroupées ensemble dans des *services* (Figure 27). Chacun de ces caractéristiques et services possède un identifiant unique (un UUID). Certaines caractéristiques et services étant très repandus, ils ont été normalisés. Par exemple, le niveau de la batterie est représenté par une caractéristique dont l'UUID est « 00002a19-0000-1000-8000-00805f9b34fb », la valeur que contient cette caractéristique est un octet (un nombre entier entre 0 et 255) qui représente le pourcentage de la batterie.

Dans le cas où la caractéristique a une valeur qui change fréquemment, par exemple la fréquence de rythme cardiaque, elle peut émettre une notification à laquelle le central peut s'abonner pour savoir lorsque la valeur change.

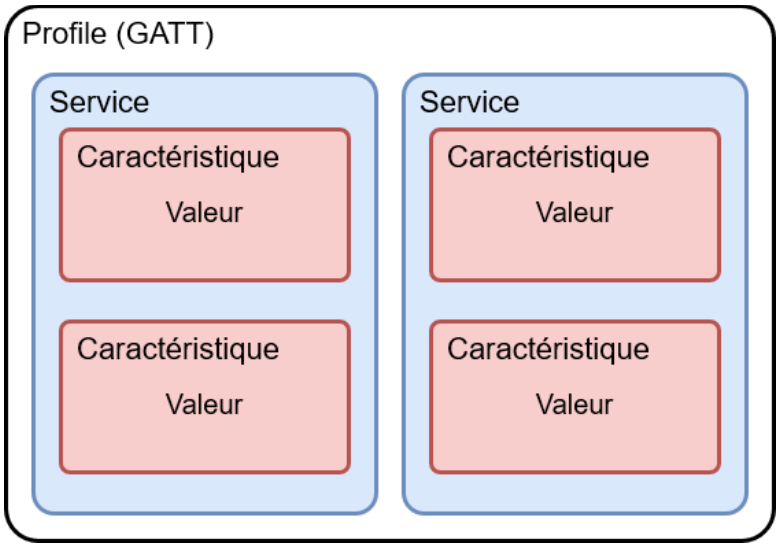


Figure 27: Schéma de profile Bluetooth Low Energy

6.2 Minimal File Transfert Protocol (MFTP)

Il n'existe aucun protocole pour réaliser un transfert de fichier avec le BLE. Nous avons donc créé notre propre protocole qui se base sur le service BLE Nordic UART, qui émule une liaison série. Ce service comporte deux caractéristiques, l'une servant à l'envoi des données et l'autre à la réception, à l'instar des broches TX et RX dans le cas d'un liaison série classique.

Notre protocole est inspiré du *Trivial File Transfert Protocol*, qui est une version simplifiée du protocole FTP. Le client envoie au serveur une requête comportant le type d'opération à réaliser (lecture, écriture ou suppression) et le nom du fichier sur lequel opérer. La délimitation de la trame de requête est inspirée du *Serial Line Internet Protocol*, qui définit certains octets spéciaux qui servent à indiquer la fin d'une trame.

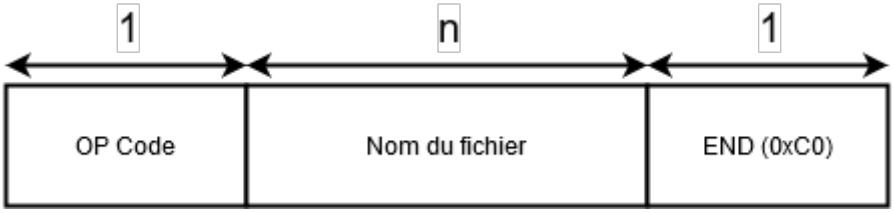


Figure 28: Trame de requête MFTP

Le serveur accuse réception de la requête, soit en l'acceptant, soit en la refusant si par exemple le fichier n'existe pas ou si celui-ci ne doit pas être modifié ou supprimé. Si la requête est acceptée, le serveur et/ou le client envoie les données en fonction du type de requête (Figure 29, 30, 31 et 32). Dans le cas d'un envoi de fichier (lecture ou écriture), la taille du fichier est transmise en premier et est de taille fixe : 4 octets. Ainsi, le récepteur connaît le nombre d'octets qu'il s'apprête à recevoir et sait lorsque le transfert est terminé sans que l'émetteur ai à le signaler. Dans le cas de la suppression, le serveur renvoie un octet pour signaler la fin de la suppression.

Pour la récupération de la liste de fichiers, nous avions pensé dans un premier temps utiliser une requête dédiée. Puis on s'est rendu compte que cela faisait doublon avec la requête de lecture. Nous avons donc ajouté le concept de fichier virtuel à notre protocole. Ces fichiers ne sont pas présents sur le système de fichier du capteur, mais leur contenu est généré au moment où la requête est faite. Par exemple, le fichier « ~FILELIST » contient la liste des noms des fichiers présents sur la carte SD du capteur séparés par le caractère « : ». Les requêtes de lecture sur ce fichier sont toujours acceptées, mais les requêtes d'écriture et de suppression sont toujours refusées.

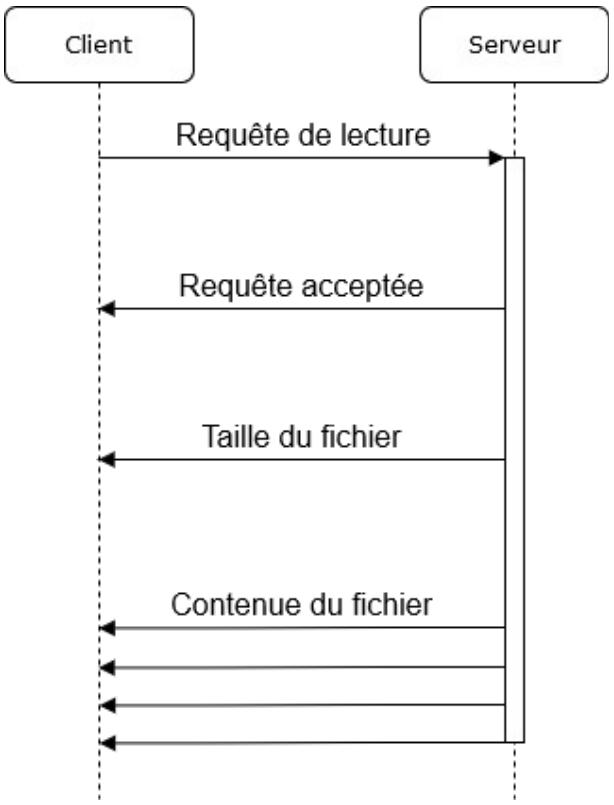


Figure 29: MFTP : requête de lecture

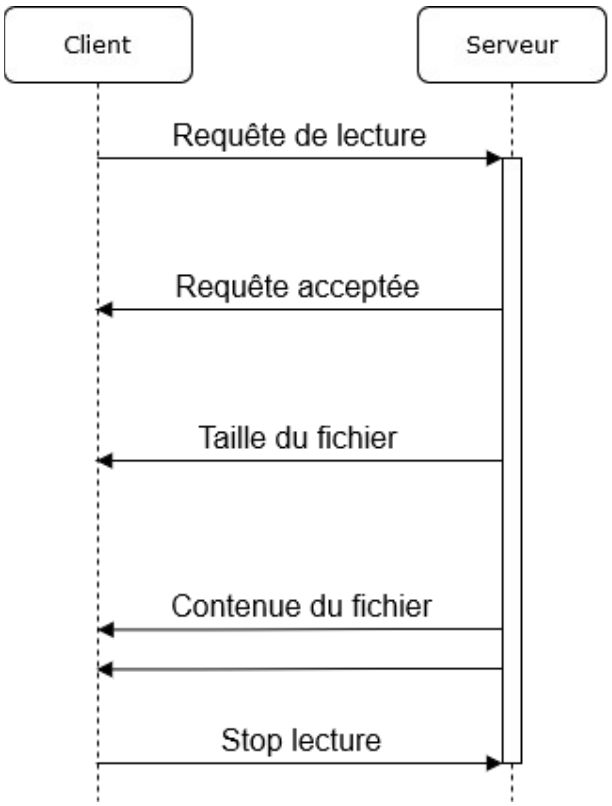


Figure 30: MFTP : interruption de la lecture

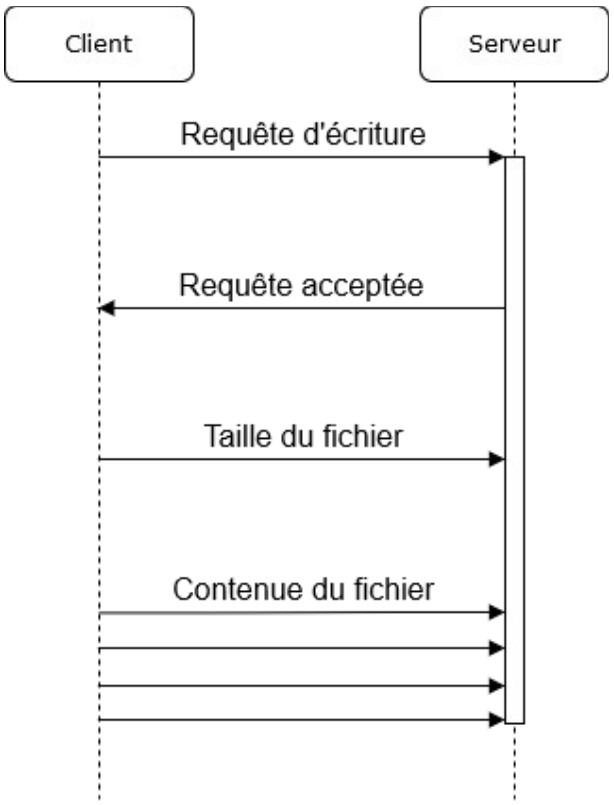


Figure 31: MFTP : requête d'écriture

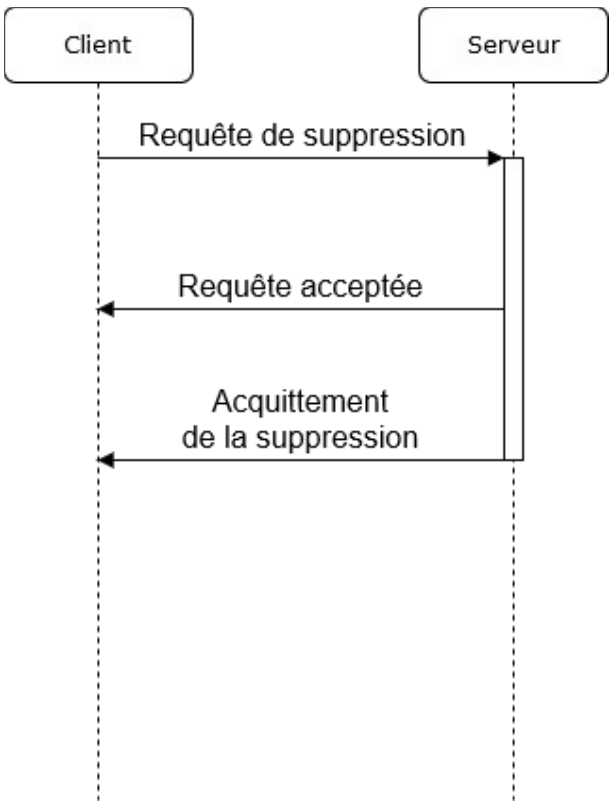


Figure 32: MFTP : requête de suppression

7 Tâches annexes

La création de l'application m'a pris moins de temps que prévu, et il me restait un peu moins d'un mois avant la fin de mon stage. Cela a permis à Hugo de me déléguer une partie de son travail sur le capteur, lui libérant du temps pour travailler sur les autres aspects du capteur.

7.1 Veille technologique

Ayant terminé l'application, j'ai profité de certains après-midi pour faire de la veille technologique. Cela m'a permis de commencer à apprendre le Rust, un langage de programmation récent qui tendrait à concurrencer le C et le C++.

7.2 Implémentation de service BLE custom

Dans le cadre de la communication avec l'application, nous avions besoin d'ajouter au capteur des services BLE qui ne sont pas normalisés.

J'ai suivi un tutoriel pour apprendre à utiliser l'API pour créer un service BLE. J'ai dans un premier temps créer un service custom ayant pour but de transmettre des informations de débogage. J'ai repris le code du service UART du SDK Nordic en retirant les éléments superflus, comme par exemple la caractéristique de réception de données. Étant donné que ce service transmet des données uniquement vers l'application. J'ai aussi créé un service pour transmettre l'état de la mémoire (espace disponible et utilisé).

7.3 Prototypage du serveur MFTP

Une autre de mes tâches a été de coder un prototype du serveur MFTP avec Arduino. Lorsque je travaillais encore sur l'application, Hugo avait réalisé une première version qui permettait de lire des fichiers et d'interrompre la lecture. En reprenant ce qu'il avait fait, j'ai créé un module qui implémente toutes les fonctionnalités du protocole.

Ayant l'habitude de programmer sur PC avec des langages haut niveau, j'ai pris l'habitude de coder de manière générique et polyvalente, c'est-à-dire de rendre mon code réutilisable facilement dans diverses situations. En discutant avec Hugo de cette première version du prototype, il m'a conseillé d'aller vers l'essentiel, d'avoir un prototype certes moins polyvalent, mais beaucoup plus robuste car ayant moins de cas de figure à prendre en compte. En gardant cette idée en tête, j'ai révisé le code du module.

Le serveur fonctionne sur le principe d'une machine à état et va traiter les données qu'il reçoit en fonction de son état actuelle et mettre à jour cet état au besoin. Le serveur peut se trouver dans l'un des états suivants :

- **non initialisé** : le serveur attend d'être initialisé
- **en attente d'une requête** : le serveur attend une requête de la part du client
- **lecture** : la lecture d'un fichier est en cours
- **écriture** : l'écriture d'un fichier est en cours

La gestion des fichiers et l'envoie de donnée au client sont délégués au code applicatif pour rendre le code du serveur découplé de tout système de fichier ou de système de communication avec le client. Le code applicatif définit pour ça deux gestionnaires :

- *request_handler* : appelé lorsque le serveur identifie une requête en transmettant la nature de la requête (lecture, écriture ou suppression) et le nom du fichier. Le gestionnaire répond pour indiquer si la requête est acceptée ou non, et avec la taille du fichier s'il s'agit d'une requête de lecture.
- *event_handler* : appelé à diverses occasions pour indiquer un certain événement :
  - SEND_DATA : indique que des données doivent être envoyées au client
  - DATA_WRITTEN : indique que des données doivent être écrites dans le fichier
  - READ_END : indique la fin d'une opération de lecture
  - WRITE_END : indique la fin d'une opération d'écriture
  - READ_CANCELLED : indique que la lecture a été annulée par le client
  - DELETE_FILE : indique qu'un fichier doit être supprimé

Le détail du fonctionnement du serveur se trouve en Annexe 3 : Fonctionnement du serveur MFTP.

8 Gestion du projet

8.1 Gestion du temps

Figure 33: Écran de lancement de GanttProject

La gestion du temps est le point clé de tout projet. Afin de répartir correctement mon temps sur les tâches à accomplir, j'ai utilisé *GanttProject*.

GanttProject est un logiciel de gestion de temps basé sur le diagramme de Gantt. Il permet de visualiser la durer des tâches, celles pouvant être réalisées en parallèle et leurs dépendances les unes aux autres.

J'ai pris la première semaine pour me familiariser avec le sujet, lire les rapports de stages des stagiaires qui ont travaillé sur le projet. J'ai dans un premier temps découpé mon temps restant en trois sections : la conception, le développement et la fin du stage ; puis ai détaillé chaque section. Sur le conseil de mon tuteur de stage, j'ai réservé les deux dernières semaines pour la rédaction de mon rapport et la préparation de ma soutenance.

Pour la phase de développement, j'ai réservé la première journée pour mettre en place l'environnement de travail. J'ai ensuite réparti les différentes fonctionnalités qui devaient être implémentées en fonction de leur priorité en essayant d'estimer le temps qu'elle me prendrait.

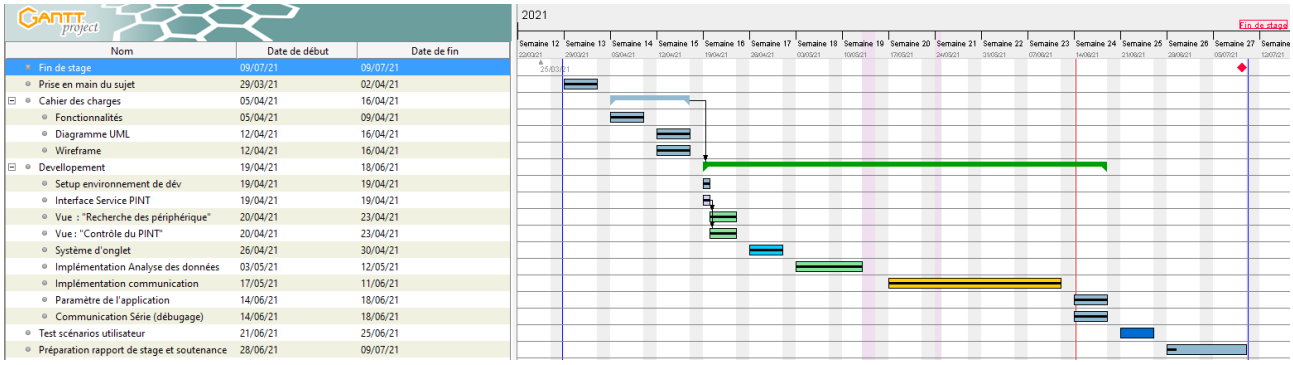


Figure 34: Diagramme de gantt de mon stage

8.2 Outils collaboratifs

Afin de mettre en commun plus facilement notre travail, nous avons mis en place un tableau de kanban (Figure 35). Mes tâches sont représentées en bleu sur le tableau et celle d'Hugo en orange. Le tableau permet de ranger les tâches en fonction de leur avancement, allant de gauche à droite : backlog (les tâches prévues, mais pas planifiées), à faire, en cours, test, révision du code et terminé.

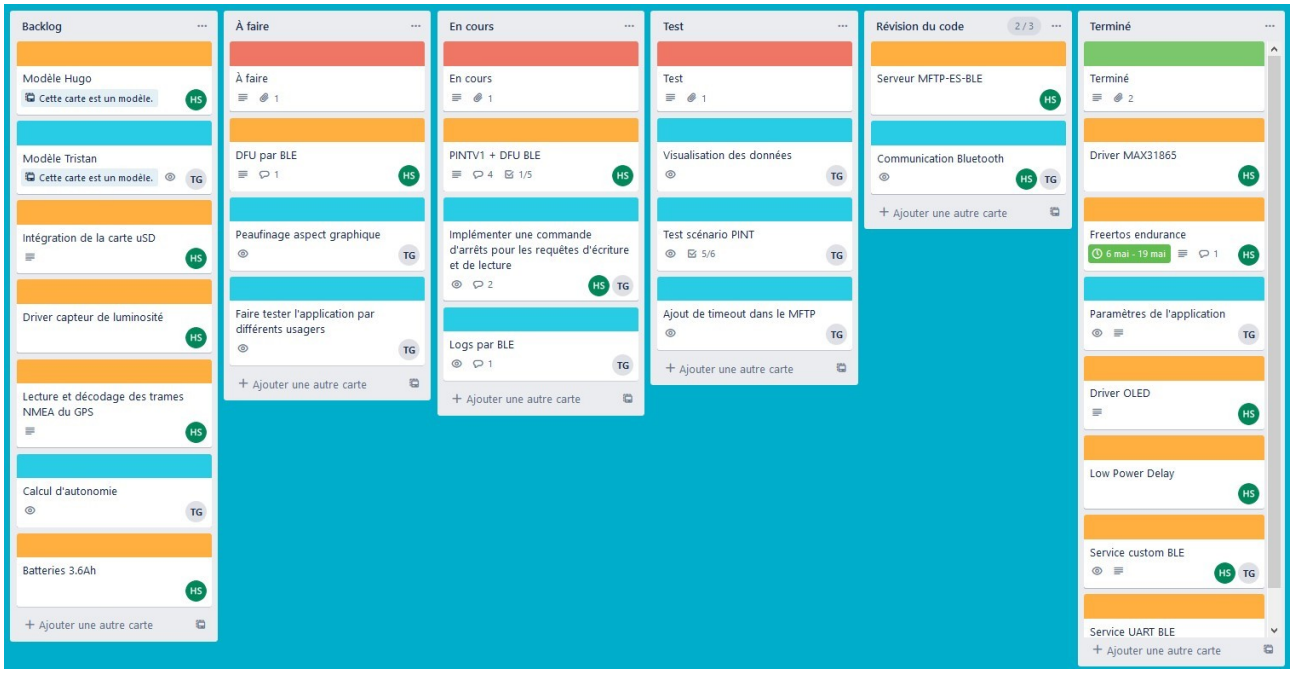


Figure 35: Tableau de kanban du projet

8.3 Gestion du code source

Figure 36: Logo de Git

Git est un logiciel libre de gestion de versions décentralisé. C'est-à-dire qu'il permet de conserver la chronologie de toutes les modifications apportées aux fichiers de notre projet sur l'ordinateur de tous les contributeurs du projet.

Il utilise un système de branche et de commit. Un commit est une modification que l'on apporte au projet, il peut s'agir d'un ajout de fonctionnalité, d'une suppression de fichier, etc. Les commits sont ensuite ajouter à l'historique de la branche courante.

Une branche représente un historique des modifications apportées au projet. C'est donc la suite de commits depuis la création du projet. On créer une branche en copiant une branche existante, puis on peut lui ajouter de nouveaux commits sans modifier la branche original. Enfin, il est possible de fusionner deux branches pour en obtenir une qui contient alors l'ensemble des commits des deux branches.

Un projet Git classique possède une branche principale généralement nommée « master » ou « main » et qui contient le projet dans un état fonctionnel. On créer alors une nouvelle branche lorsque l'on souhaite réaliser une modification (ajout d'une fonctionnalité, correction de bug, etc.). Lorsque cette modification est jugée stable, la branche est fusionnée sur la branche principale. Ce système de cloisonnement en branche permet de travailler sur plusieurs points différents en parallèles et avec d'autres contributeurs en réduisant considérablement le risque de conflit.

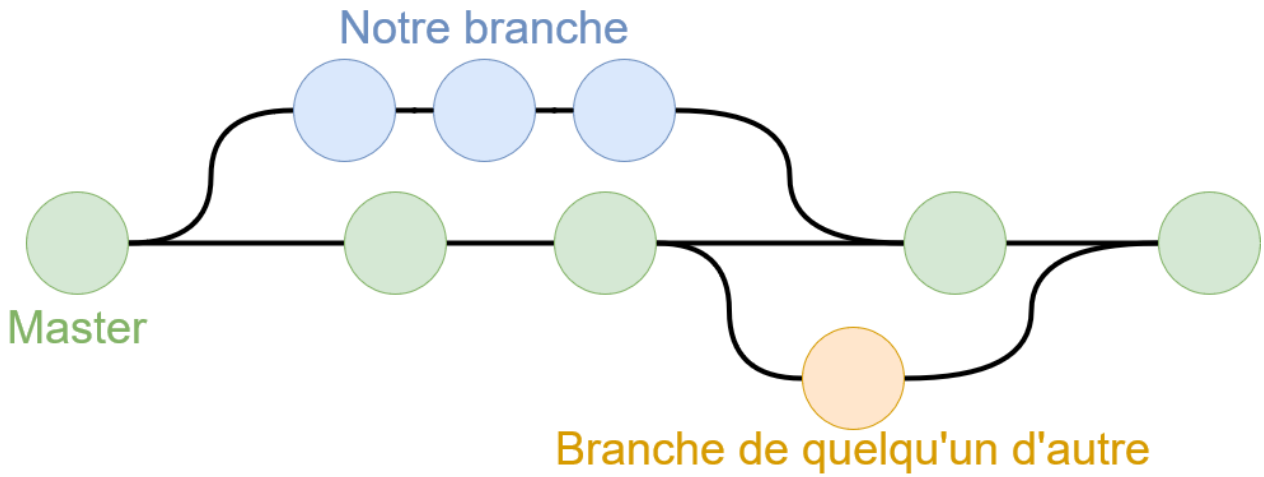


Figure 37: Exemple simple de collaboration avec Git

9 Pistes d’amélioration

9.1 Plateformes portables

Étant donné l’utilisation de l’application, il pourrait être intéressant de l’adapter pour les plateformes portables (smartphone, tablette, etc.). Cela rendrait son utilisation plus simple sur le terrain pour pouvoir régler les paramètres du capteur jusqu’au dernier moment.

9.2 Les paramètres utilisateur

Il pourrait être intéressant pour l’utilisateur qu’il puisse customiser certains points du comportement de l’application. Par exemple, lorsque les fichiers de température sont téléchargés, ils sont renommés avec un format de date basé sur la date du premier échantillonnage du ficher ; on pourrait donc laisse la possibilité à l’utilisateur de configurer le format des noms de fichier.

10 Conclusion

10.1 Bilan du projet

Tous les objectifs du projet, on été remplis. Les différents scénarios imaginés, on put être testés, incluent ceux de communication avec le capteur utilisant le code réel et pas seulement un prototype pour tester les fonctions de communication.

Seul bémol, la fonctionnalité qui calcule l'autonomie batterie et mémoire en fonction des paramètres choisis par l'utilisateur est implémenté (on a un affichage des valeurs) mais les formules pour calculer ces autonomies ne sont pas codées. Elles seront ajoutées plus tard par Hugo, lorsqu'il s'occupera de ces aspects.

10.2 Bilan personnel

Ce stage a été enrichissant. Premièrement, d'un point de vue technique, j'ai pu mettre en pratique et faire évoluer mes méthodes de développement.

Mais aussi d'un point de vue humain, Ifremer offre un environnement de travail ouvert et convivial, la discussion avec nos collègues est très facile. On peut ainsi obtenir un point de vue extérieur sur ce que l'on fait, apporter son point de vue sur le projet des autres.

Bibliographie

Hugo Schaaf *Rapport de stage assistant.e ingénieur.e, Evolution du capteur de température grands fonds PINT*, 2020

waytolearnx *Différence entre MVC et MVVM*, 10 juin 2020 [https://waytolearnx.com/2020/06/difference-entre-mvc-et-mvvm.html]

institut agile Given - When – Then [http://institut-agile.fr/gwt.html]

Annexe 1 : Captures d’écran de l’application

Figure 38: Capture d'écran de la page d'accueil

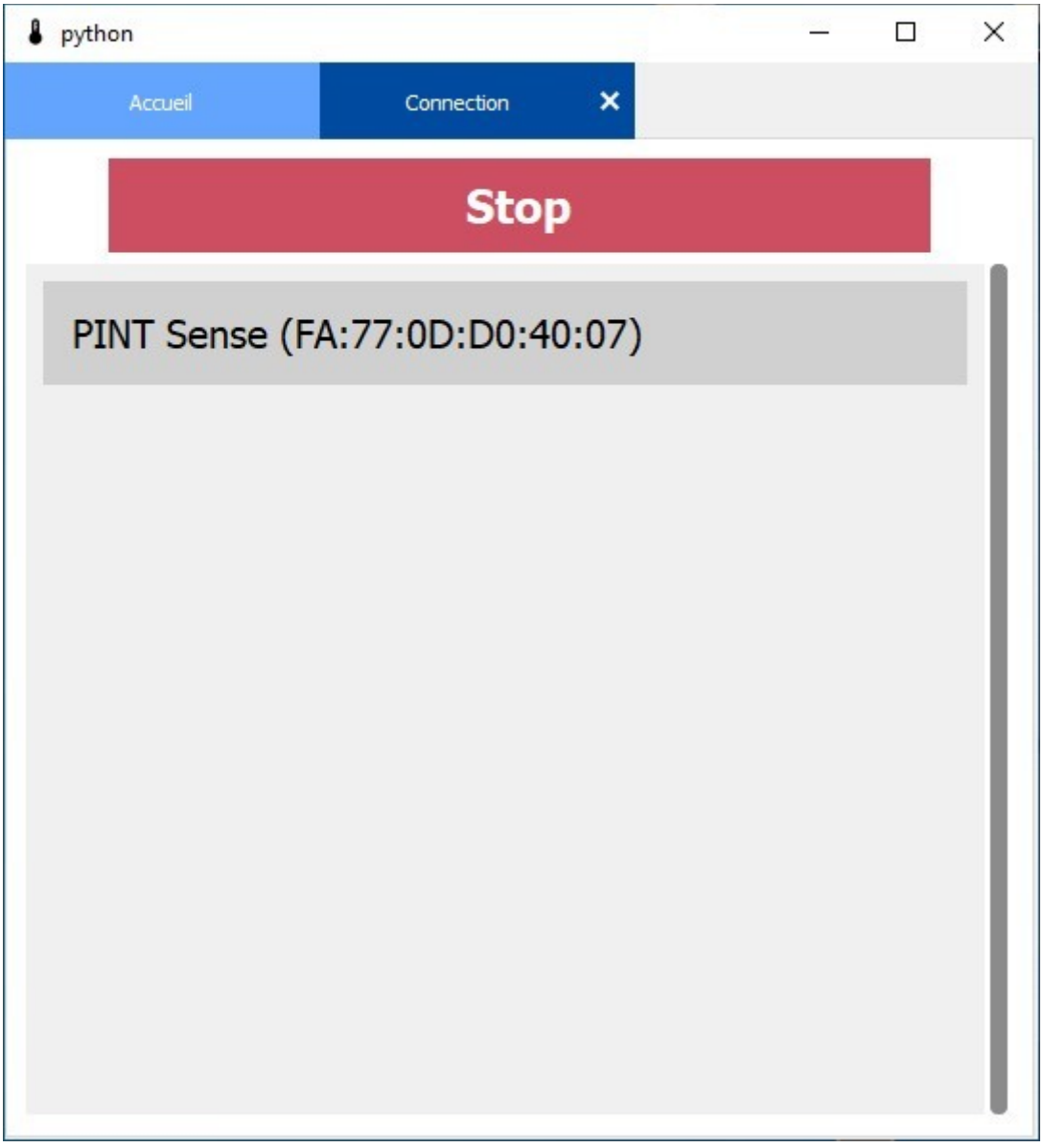


Figure 39: Capture d'écran de la page de scan

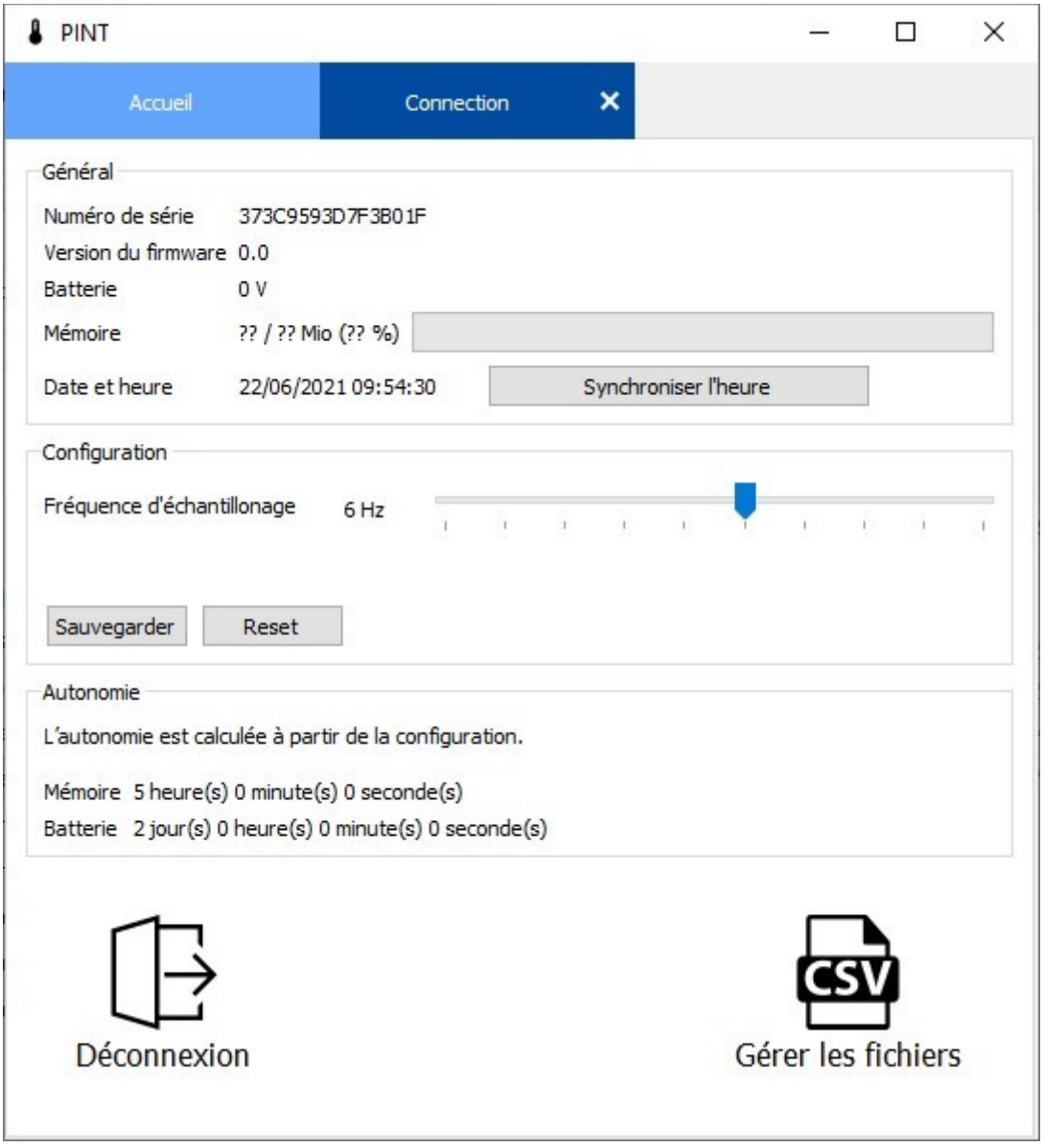


Figure 40: Capture d'écran de la page de gestion du capteur

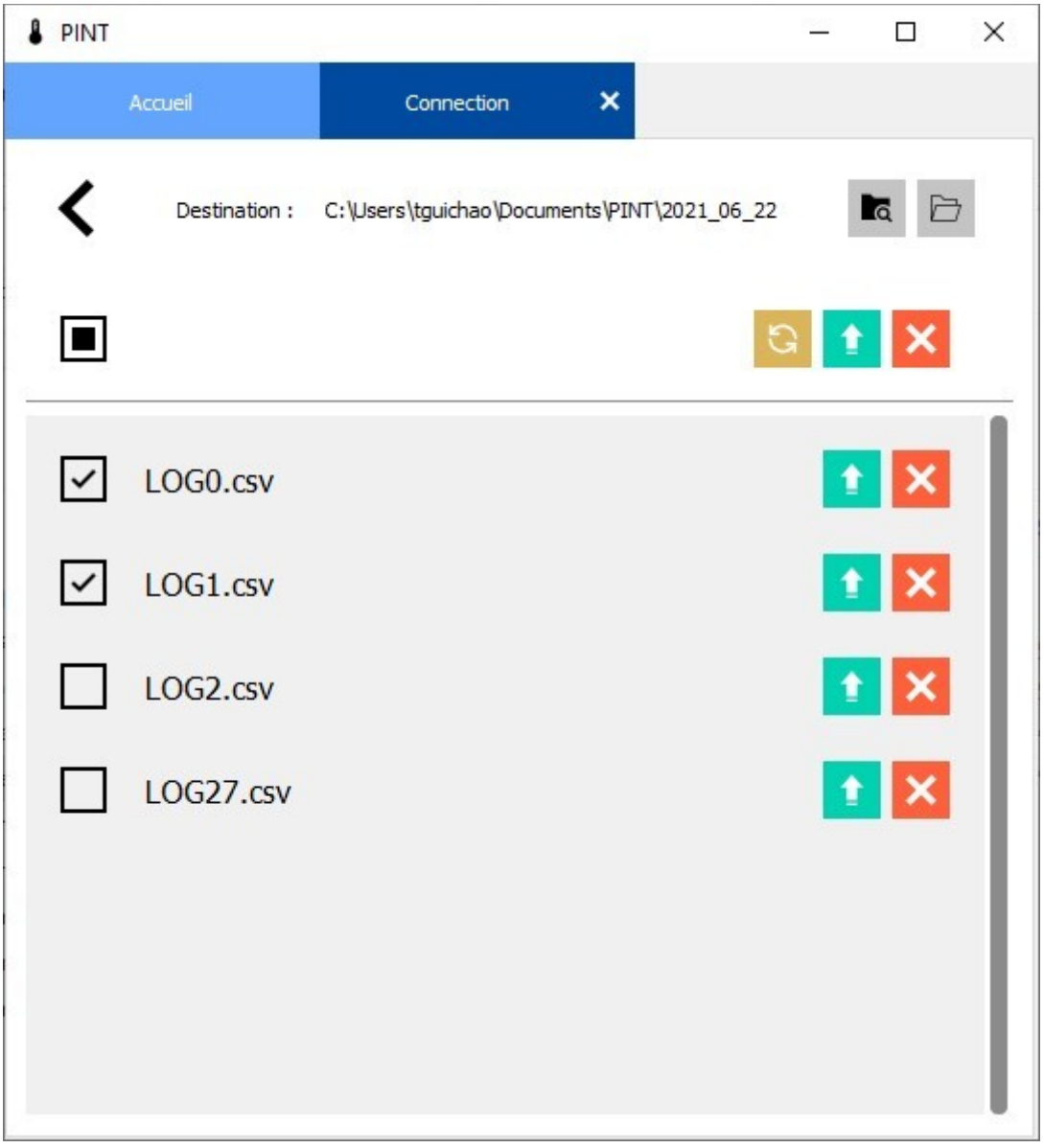


Figure 41: Capture d'écran de la page de gestion des fichiers du capteur

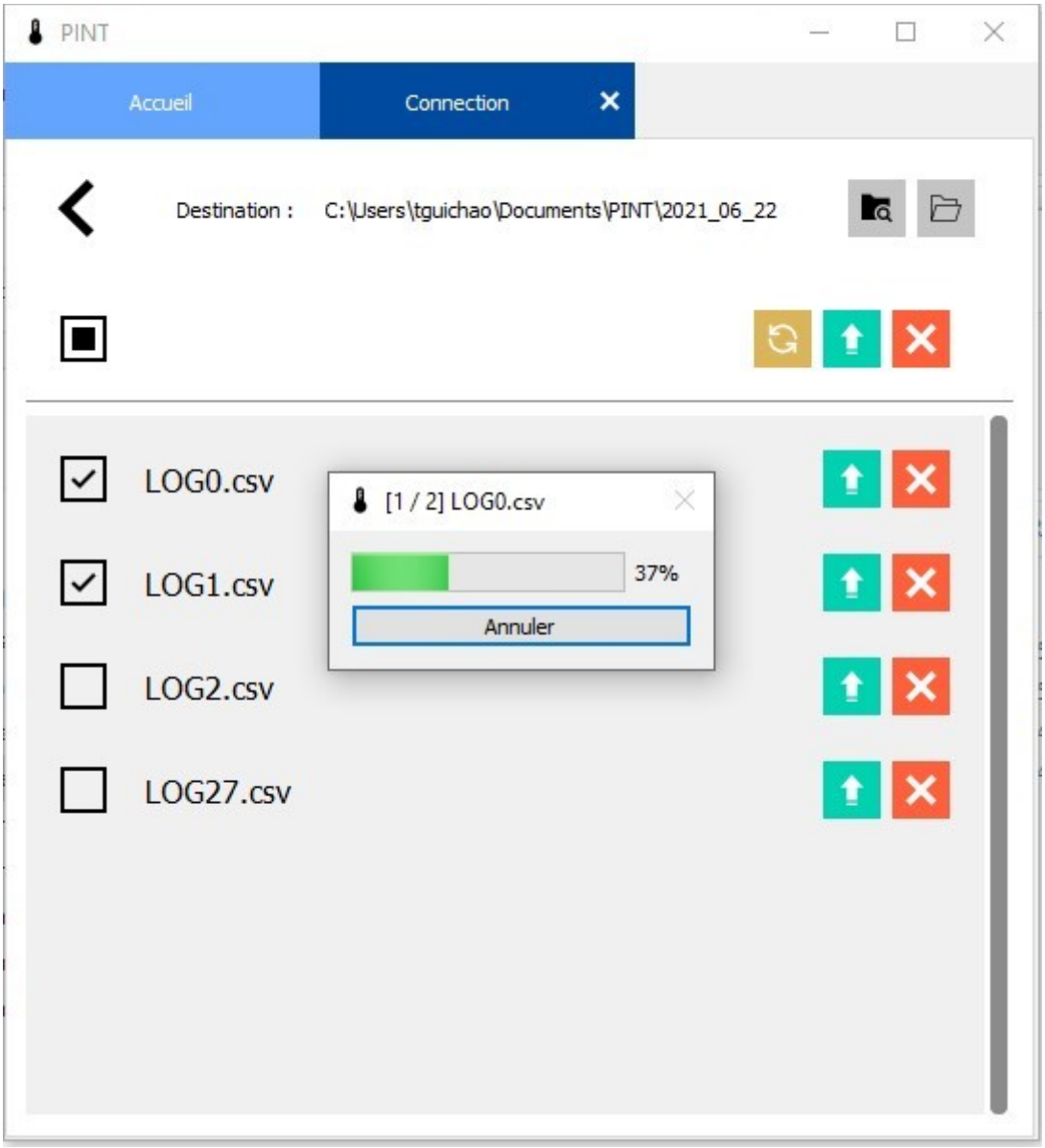


Figure 42: Capture d'écran d'un téléchargement de fichier

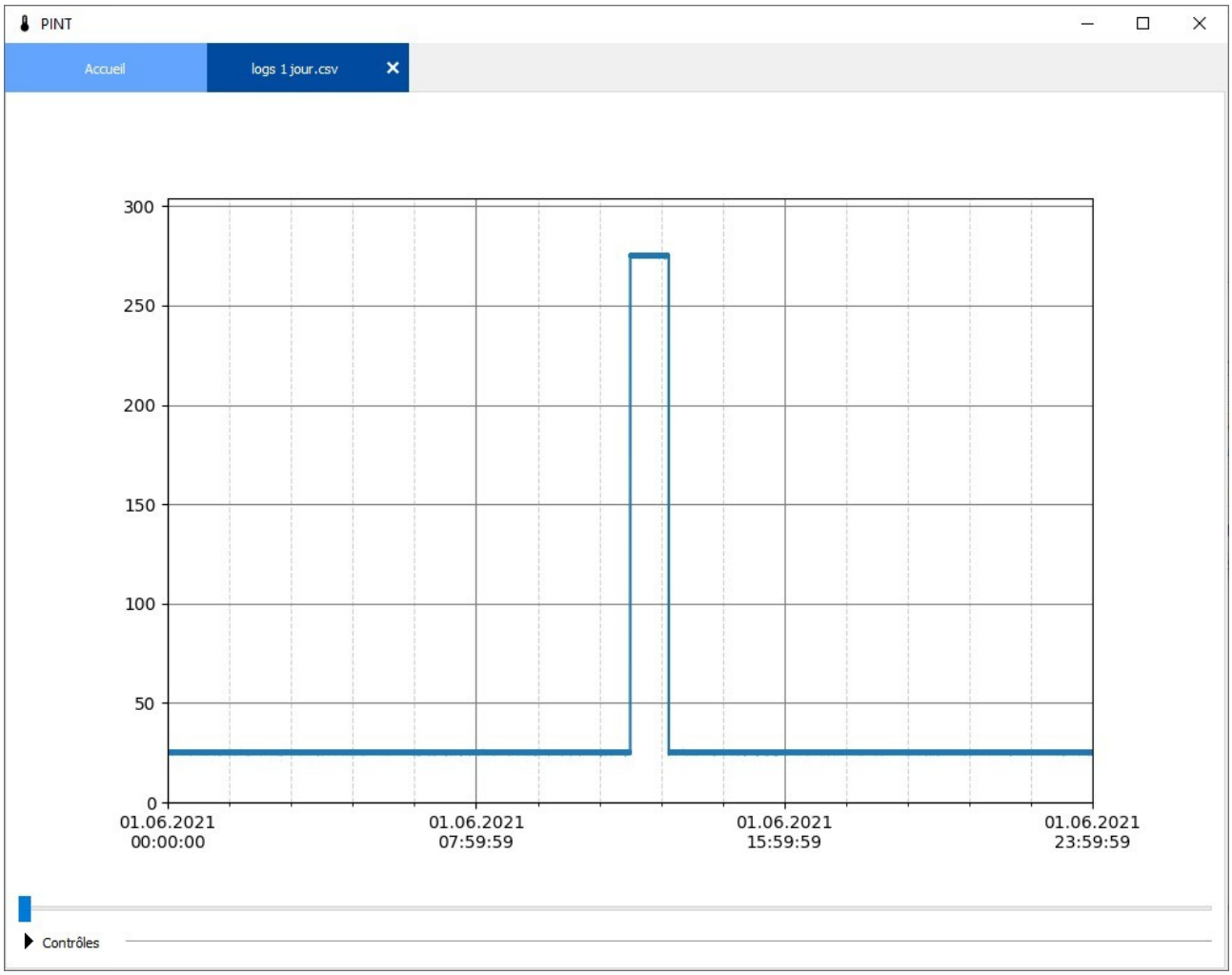


Figure 43: Capture d'écran du visualiseur de données

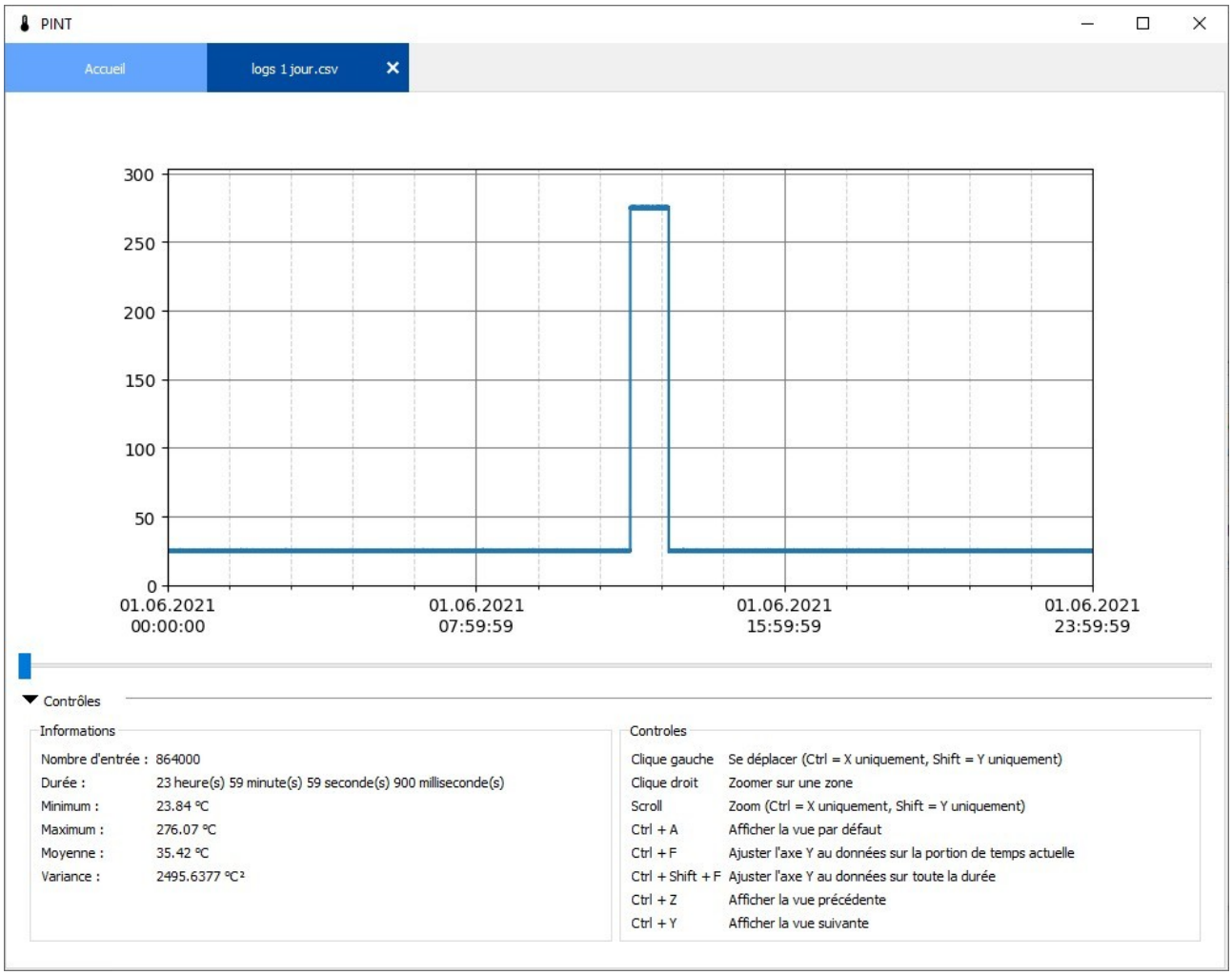


Figure 44: Capture d'écran du visualiseur de données avec les contrôles affichés

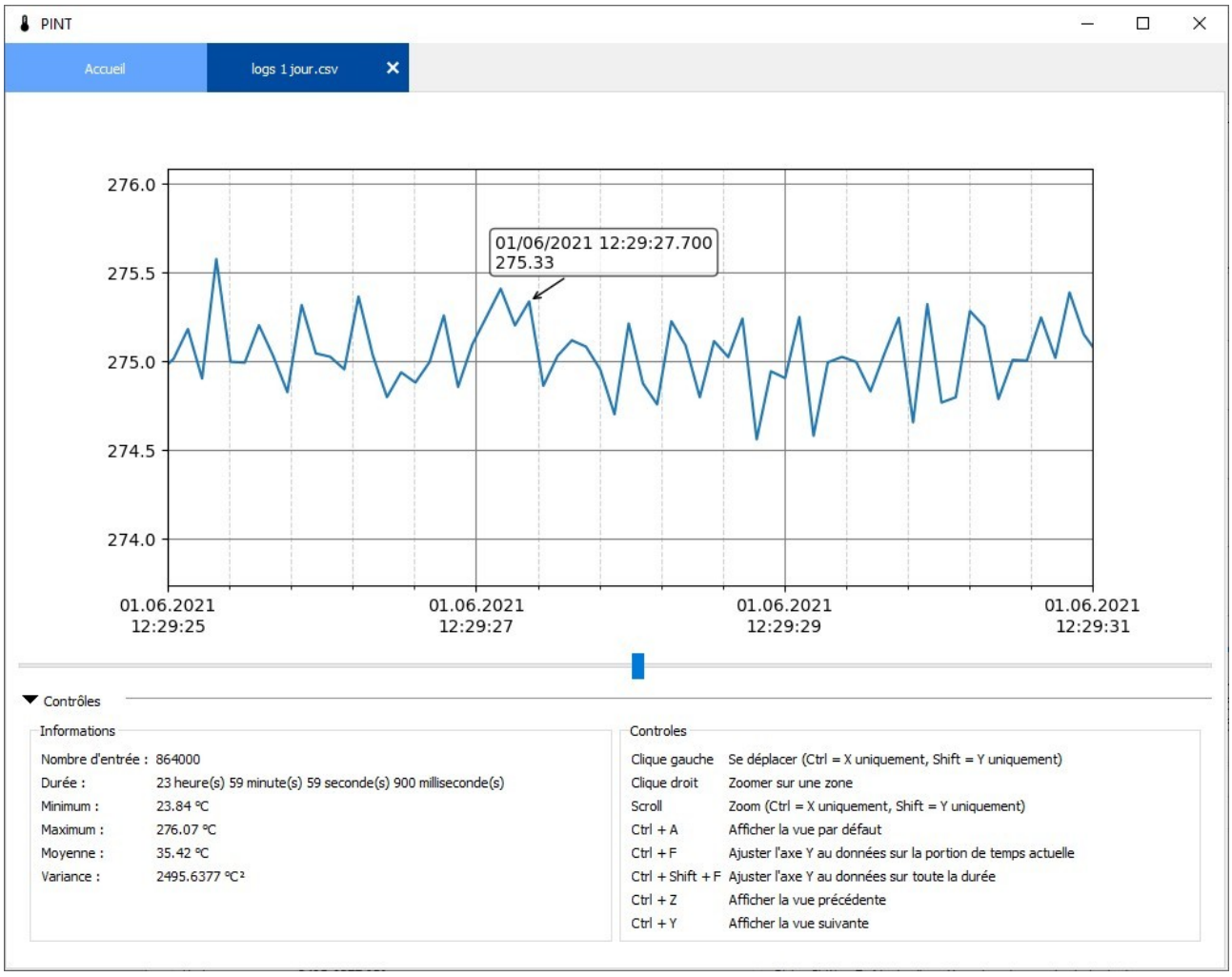


Figure 45: Capture d'écran du visualiseur de données avec affichage d'un point de donnée

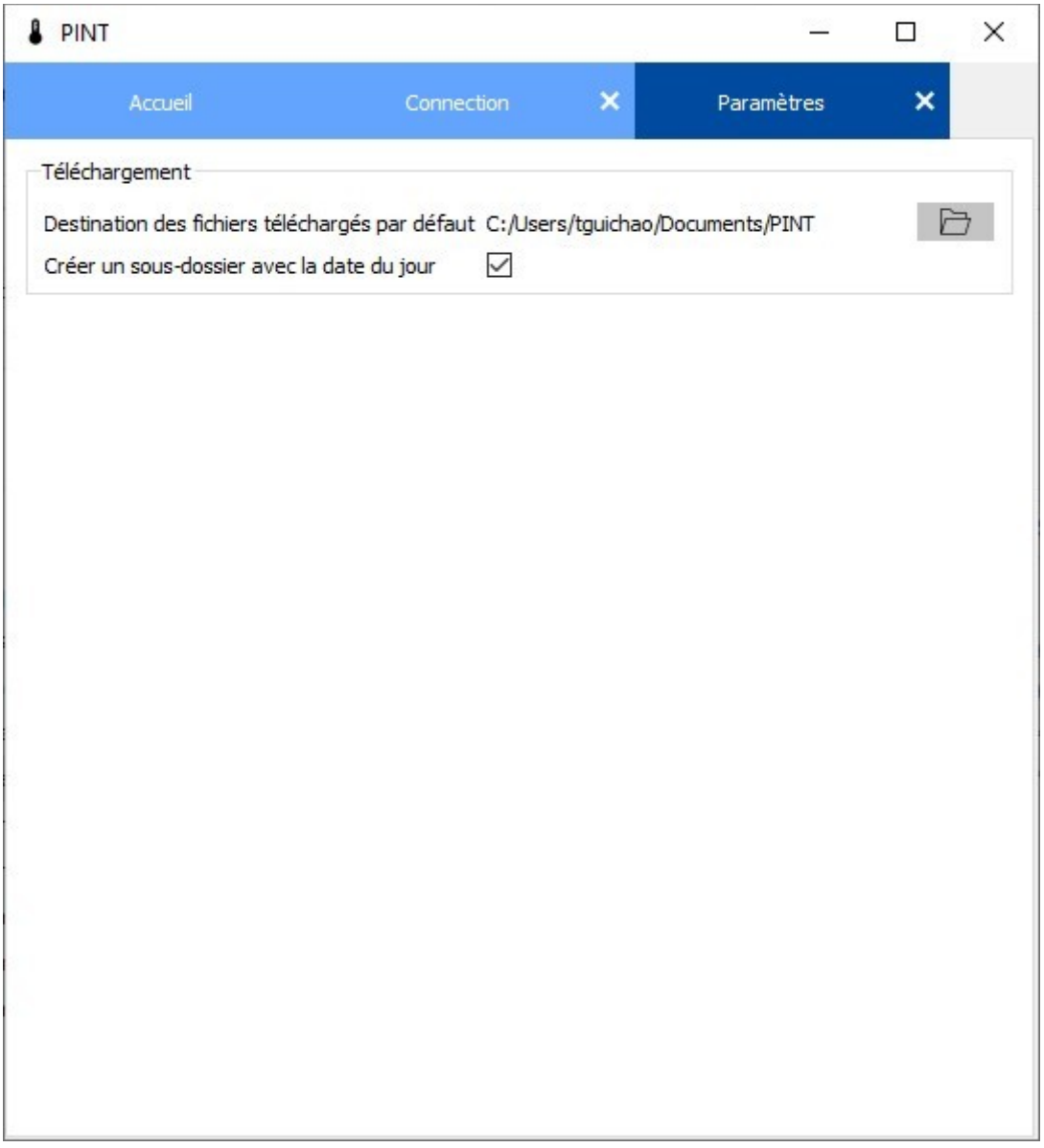
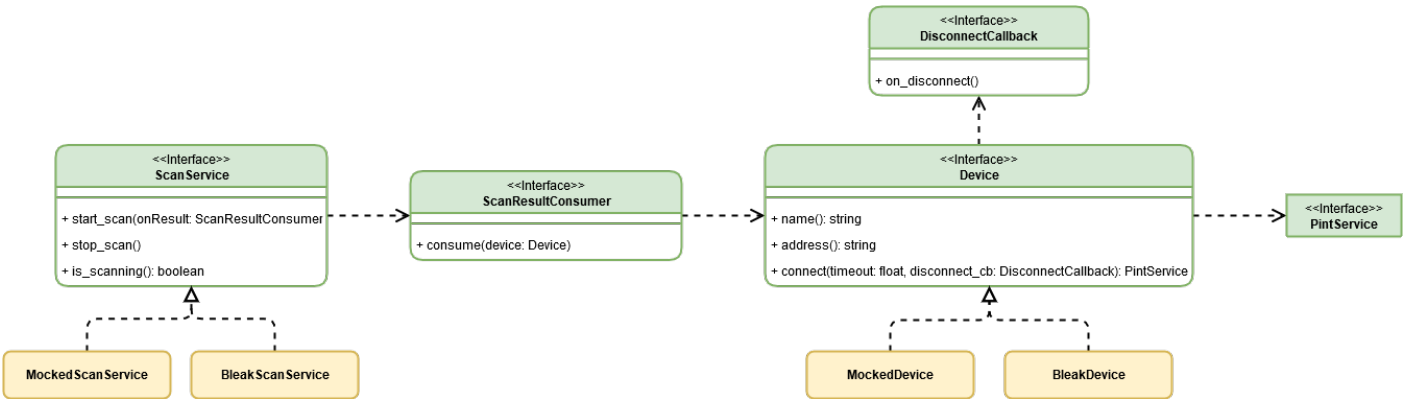
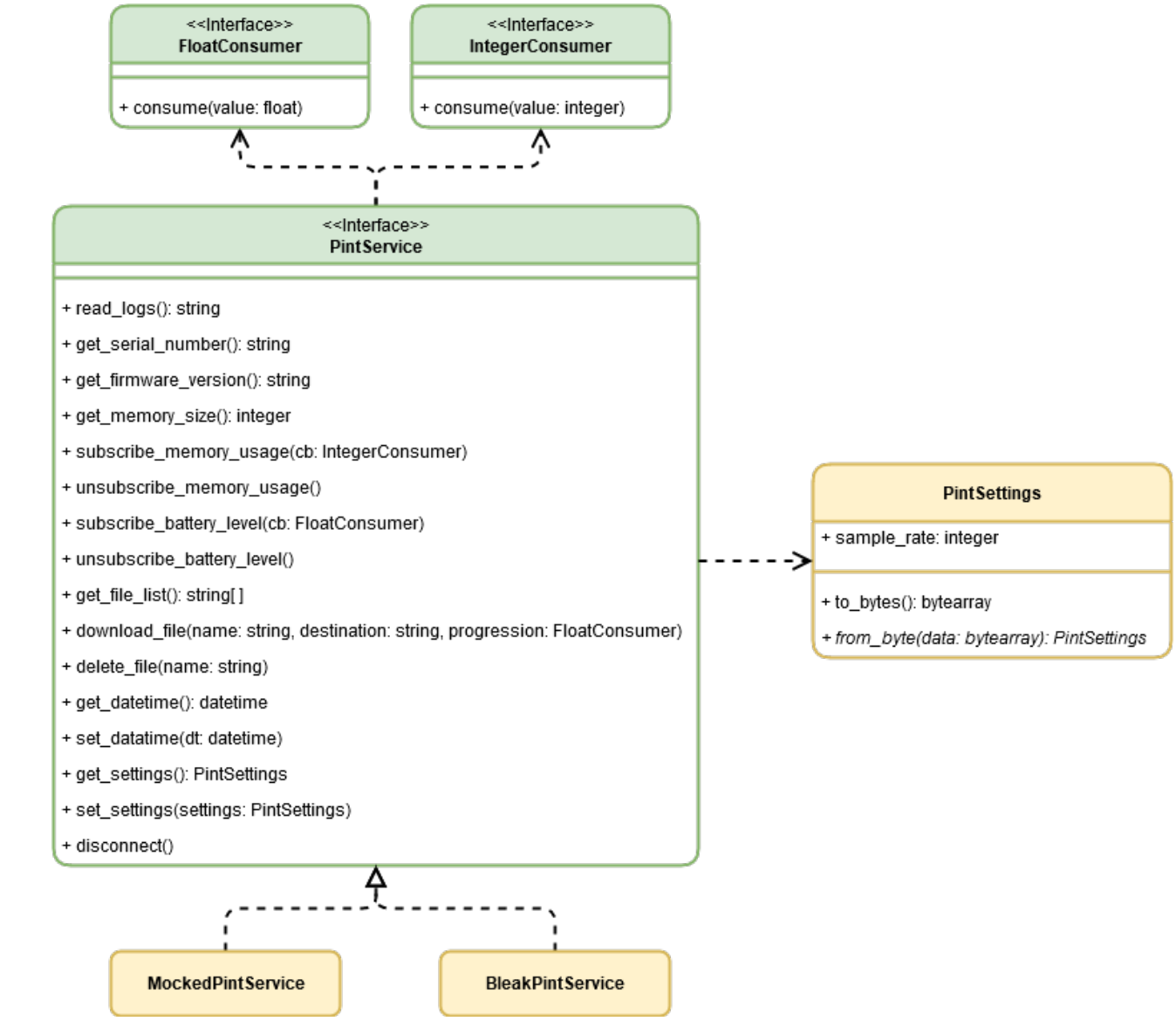


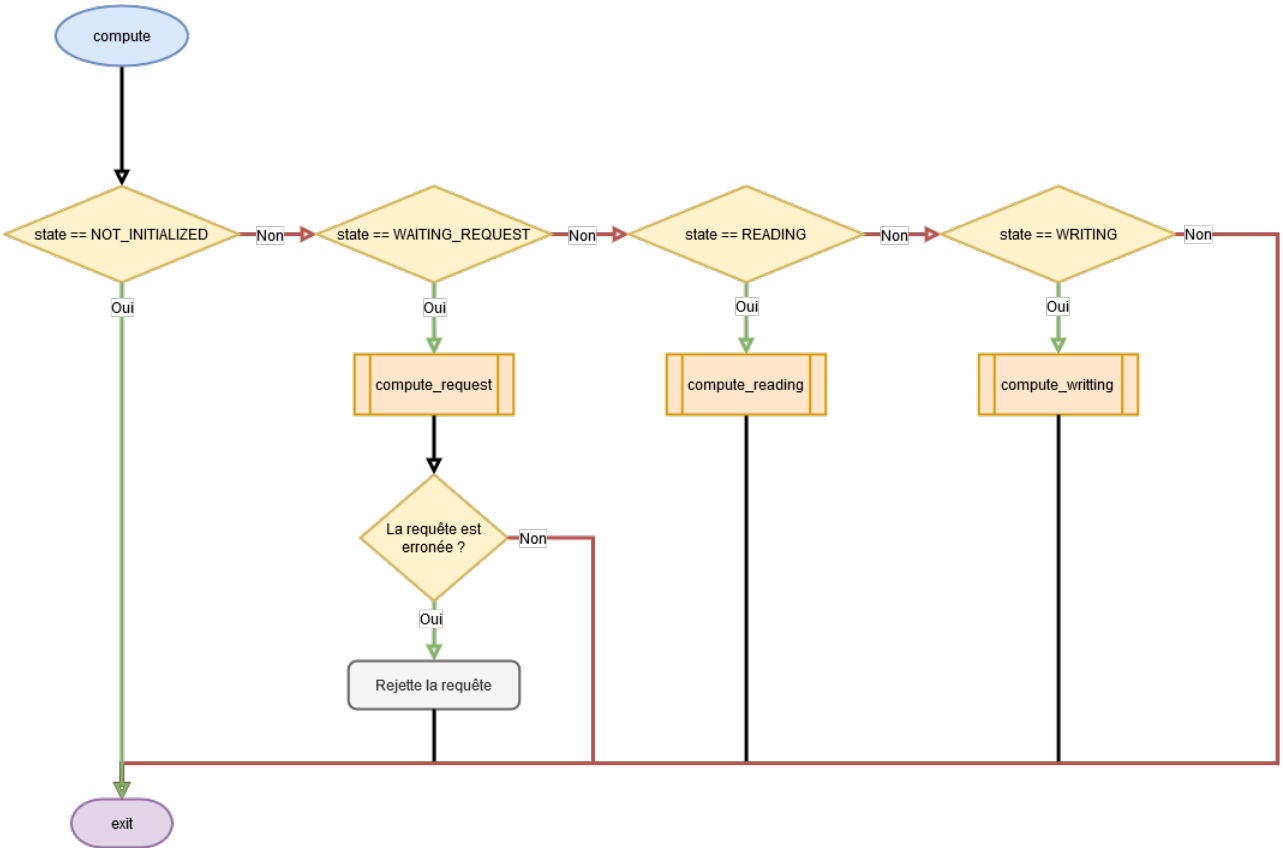
Figure 46: Capture d'écran de la page de paramètre de l'application

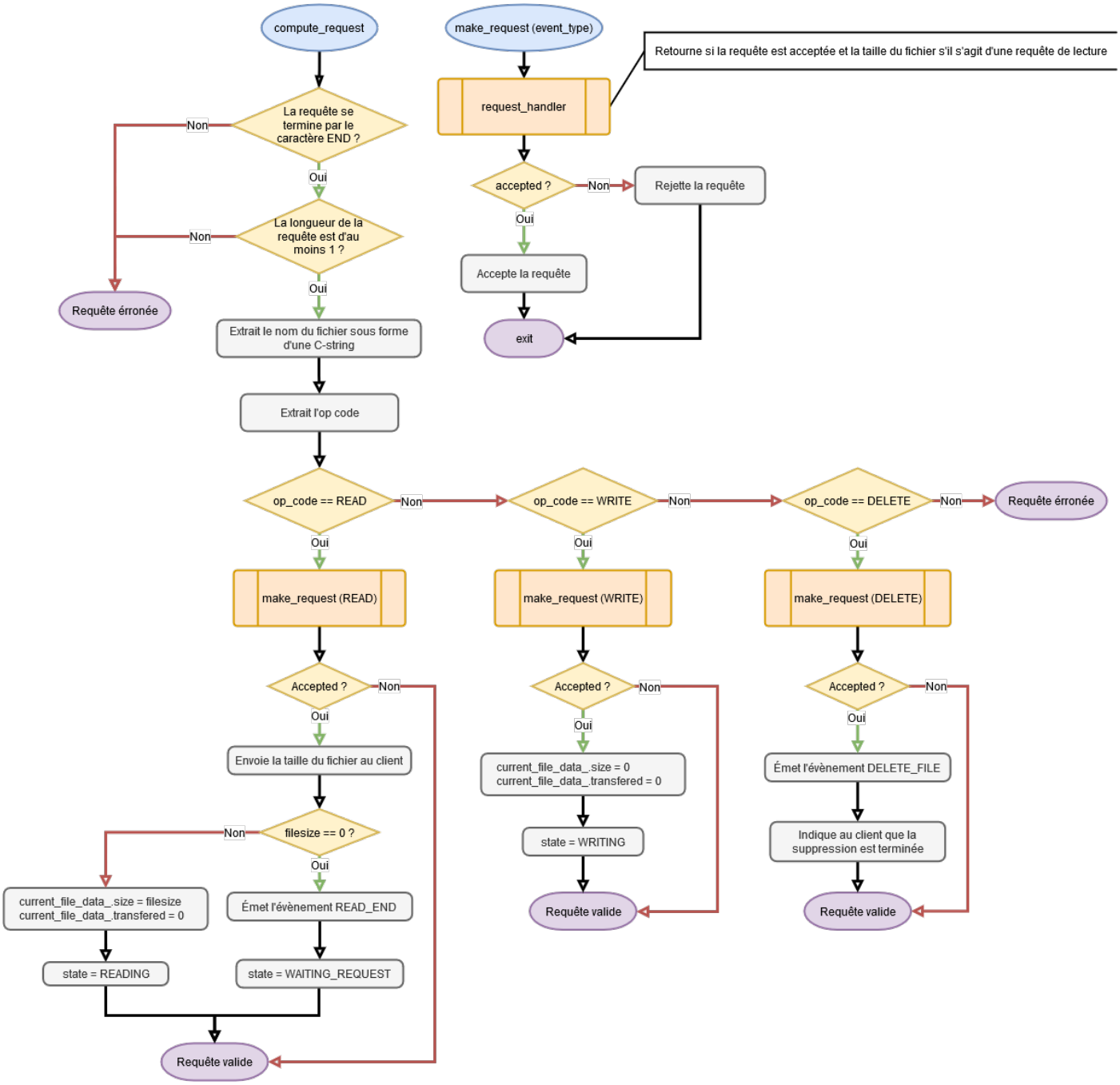
Annexe 2 : Interface de communication avec PINT

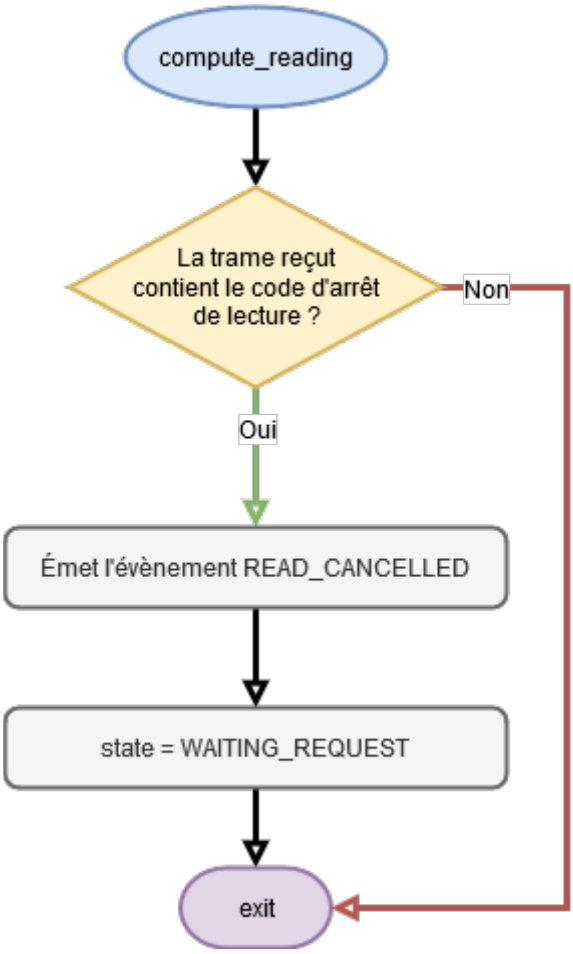


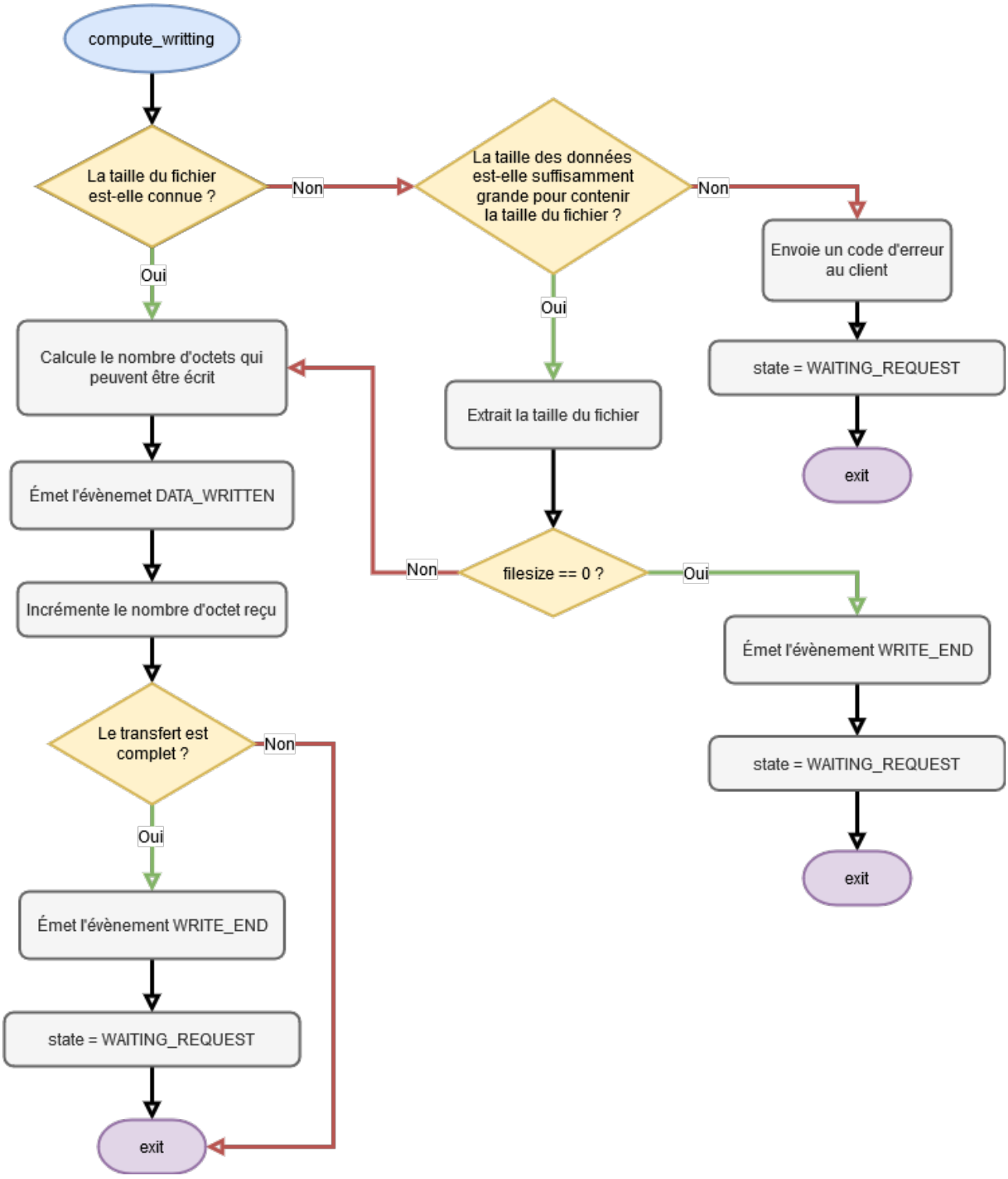


Annexe 3 : Fonctionnement du serveur MFTP




compute_request
La requête se termine par le caractère END ?
Non
Oui
La longueur de la requête est d'au moins 1 ?
Non
Oui
Requête érronée
Extrait le nom du fichier sous forme d'une C-string
Extrait l'op code
op_code == READ
Non
Oui
make_request (READ)
Accepted ?
Non
Oui
Envoie la taille du fichier au client
filesize == 0 ?
Non
Oui
current_file_data_.size = filesize
current_file_data_.transfered = 0
state = READING
Émet l'évènement READ_END
state = WAITING_REQUEST
Requête valide
make_request (event_type)
request_handler
Retourne si la requête est acceptée et la taille du fichier s'il s'agit d'une requête de lecture
accepted ?
Non
Rejette la requête
Oui
Accepte la requête
exit
op_code == WRITE
Non
Oui
make_request (WRITE)
Accepted ?
Non
Oui
current_file_data_.size = 0
current_file_data_.transfered = 0
state = WRITING
Requête valide
op_code == DELETE
Non
Oui
make_request (DELETE)
Accepted ?
Non
Oui
Émet l'évènement DELETE_FILE
Indique au client que la suppression est terminée
Requête valide
Requête érronée


compute_reading
La trame reçut contient le code d'arrêt de lecture ?
Non
Oui
Émet l'évènement READ_CANCELLED
state = WAITING_REQUEST
exit


compute_writting
La taille du fichier est-elle connue ?
Non
La taille des données est-elle suffisamment grande pour contenir la taille du fichier ?
Non
Envoie un code d'erreur au client
state = WAITING_REQUEST
exit
Oui
Oui
Calcule le nombre d'octets qui peuvent être écrit
Extrait la taille du fichier
Émet l'évènemet DATA_WRITTEN
Non
filesize == 0 ?
Oui
Incrémente le nombre d'octet reçu
Émet l'évènement WRITE_END
state = WAITING_REQUEST
exit
Le transfert est complet ?
Non
Oui
Émet l'évènement WRITE_END
state = WAITING_REQUEST
exit