

PROJET MYCELIUM 2.0

Rapport de conception

Equipe projet

BOIZUMAULT Maiwenn

JULIEN Léo-Paul

TUMOINE Ivy

Encadrant

PARLAVANTZAS Nikolaos

Avec la participation de

MOUREAU Julien et LONGUEVERGNE Laurent
- Université de Rennes 1

CAROFF Nicolas, RAOUL Thibaut, VILLE Grégoire
- INSA Rennes

13 février 2023



Table des matières

1	Glossaire	2
2	Introduction	3
3	Architecture globale	4
3.1	Architecture matérielle	4
3.1.1	Architecture matérielle de Mycélium 1.0	5
3.1.2	Architecture matérielle de Mycélium 2.0	6
3.2	Architecture logicielle	7
4	Ajouts de Mycélium 2.0	8
4.1	Les nouveaux capteurs	8
4.1.1	Architecture induite par les nouveaux capteurs	8
4.1.2	FoxyFind	10
4.1.3	Capteurs de l'OSUR	11
4.2	Scalabilité de Mycélium	12
4.3	Nouveaux scénarios	14
4.3.1	Scénario <i>orage</i>	14
4.3.2	Scénario <i>écart aux normales saisonnières</i>	15
5	Conclusion	16
6	Annexe	19
6.1	Retard par rapport à la planification	19

1 Glossaire

- **ChirpStack** : logiciel open-source pour la gestion des appareils utilisant LoRaWAN ;
- **FaaS** : *Function as a service*, modèle de déploiement de cloud computing en microservices serverless ;
- **Gateway** : passerelle de communication entre deux réseaux distincts ;
- **IoT** : *Internet des objets*, décrit le réseau d'objets physiques constitués de capteurs, de logiciels et d'autres technologies dans le but de connecter et d'échanger des données avec d'autres appareils et systèmes sur Internet ;
- **LoRaWAN** : protocole de communication basse consommation, longue portée et faible bande passante très utilisé pour l'IoT. Il se base sur le réseau LoRa (Long range) ;
- **MQTT** : *Message Queuing Telemetry Transport*, protocole de messagerie publish-subscribe basé sur le protocole TCP/IP. Ce protocole très léger est énormément utilisé en IoT ;
- **OpenFaaS** : logiciel open-source pour les fonctions serverless sur le cloud ;
- **Raspberry Pi** : ordinateur peu coûteux de la taille d'une carte de crédit, pouvant effectuer toutes les actions d'un ordinateur classique. Sa taille, son faible coût, les modules complémentaires et sa communauté en font un outil de choix pour les projets d'IoT ;
- **Scalabilité** : capacité d'un système à augmenter ou diminuer ses ressources en fonction de l'usage de ce système ;
- **Serverless** : modèle d'exécution pour les applications sans gestion de serveur par l'utilisateur, avec des coûts payés selon l'utilisation ;
- **VPS** : *Virtual Private Server*, c'est une machine virtuelle, créée sur un serveur physique et employant ses ressources pour offrir à ses utilisateurs les mêmes fonctionnalités qu'un serveur dédié.

2 Introduction

La construction de la ligne B du métro Rennais a généré des émissions de gaz à effets de serre et une disparition de certains espaces verts. Cependant, dans le contexte actuel de crise climatique, la ville de Rennes s'est engagée à compenser ces impacts néfastes sur l'environnement, notamment en renaturant d'anciens espaces verts. Ainsi, le projet **Mycélium** se concentre sur le suivi de la **renaturation de la Croix Verte** (*figure 1*) située sur le campus de Rennes 1, face à l'INSA. Le projet Mycélium est réalisé en collaboration avec le laboratoire de Géosciences de Rennes et l'Observatoire des Sciences et de l'Univers de Rennes, l'OSUR [1]. Il vise à suivre au mieux l'évolution de la zone de la Croix Verte en y installant un maillage de capteurs et en traitant les informations issues de ces capteurs.

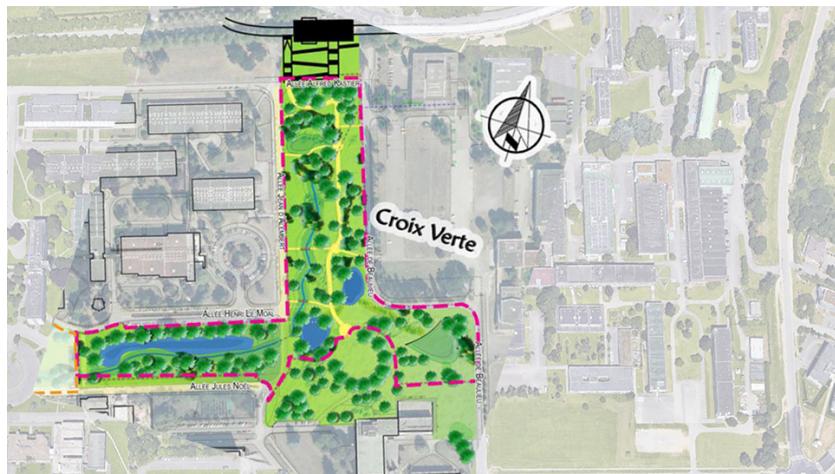


FIGURE 1 – La Croix Verte

A plus grande échelle, notre travail s'inscrit dans le projet de suivi environnemental national TERRA FORMA [2] coordonné par Laurent LONGUEVERGNE. TERRA FORMA vise à partager un état des lieux de nos ressources naturelles (capital sol, eau, biodiversité...) continuellement mis à jour.

Mycélium 2.0 est la reprise du projet de l'année précédente Mycélium 1.0 [3]. Lors de ce projet, nos prédecesseurs ont déployé un premier système collectant des données sur la Croix Verte. Ce premier système était composé d'un nœud de capteurs basse consommation communiquant via le réseau LoRa, d'une gateway recevant les données récoltées et d'un cluster de Raspberry Pi [4], chargé de traiter ces données et d'alerter les utilisateurs finaux en cas de mesures exceptionnelles. Prenons un exemple : des chutes de neiges peuvent être observées lorsque la température est inférieure à 0°C et que le taux d'humidité est important, ainsi lorsque ces critères sont observés, l'utilisateur final est notifié de la forte probabilité de neige. Les critères de mesures exceptionnelles définissent ce que l'on nomme des "scénarios", l'exemple précédent correspondant au scénario neige. Les enjeux de ce projet étaient d'obtenir un système basse consommation et adapté à l'environnement afin qu'il puisse durer dans le temps. C'est à ces enjeux que répondent les choix des capteurs et du réseau LoRa. L'objectif de Mycélium 2.0 est alors d'**améliorer** et d'**étendre** le système précédent, notamment en y intégrant de **nouveaux capteurs** et de **nouveaux scénarios** de mesures exceptionnelles.

Ce rapport de conception vise à décrire l'architecture interne du projet, sa modélisation et l'interfaçage des différents modules entre eux. Il commencera par présenter l'architecture globale de Mycélium en détaillant succinctement chaque module qui le compose, tout en distinguant les anciens modules de Mycélium 1.0 et les ajouts effectués. La section 4 du rapport permettra de revenir plus en détail sur les ajouts. Tout d'abord, elle se penchera sur de nouveaux types de capteurs et comment ces derniers sont intégrés au projet, elle traitera ensuite comment le système existant sera développé dans le Cloud, et s'en suivra enfin une section s'intéressant aux nouveaux scénarios mis en place.

3 Architecture globale

La section qui suit a pour but de présenter l'architecture globale de Mycélium, tant du point de vue matériel que logiciel. Cette section va également permettre de comprendre quels modules appartiennent à Mycélium 1.0 et quels modules sont en cours de développement. Les modules principaux et essentiels de Mycélium 1.0 seront détaillés ; concernant les modules de Mycélium 2.0, leurs principes de fonctionnement sera évoqué puis approfondi dans la seconde partie du rapport.

3.1 Architecture matérielle

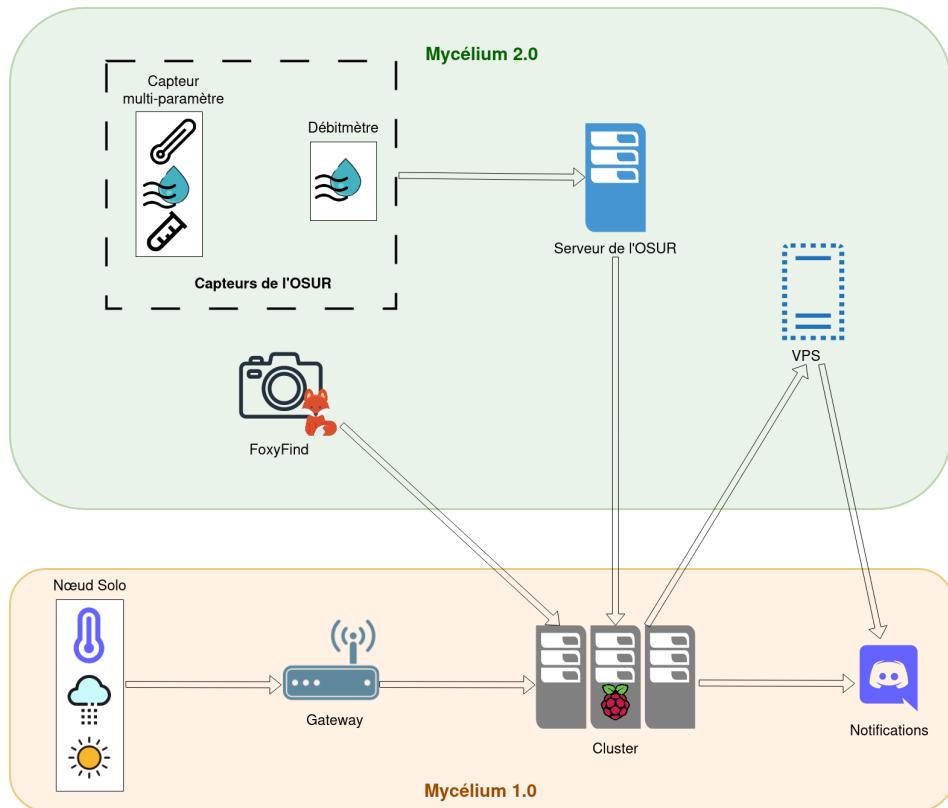


FIGURE 2 – Architecture matérielle de Mycélium

La *figure 2* présente l'architecture matérielle de Mycélium. Le cadre orangé inférieur représente les éléments présents dans Mycélium 1.0 [5], tandis que la partie verte supérieure représente les éléments ajoutés cette année dans le cadre de Mycélium 2.0.

3.1.1 Architecture matérielle de Mycélium 1.0

Mycélium 1.0 se compose donc d'un nœud de capteurs nommé nœud SoLo qui envoie régulièrement des mesures à la gateway via le protocole réseau LoRaWAN. Cette gateway transmet ensuite par ethernet les mesures au cluster de cinq Raspberry Pi. Ce cluster a pour rôle de décoder, de sauvegarder, de traiter les données et d'alerter l'utilisateur en cas de mesures exceptionnelles. Chaque élément va ici être observé de manière plus détaillée.

Le **nœud SoLo** est un boîtier étanche contenant plusieurs capteurs (*figure 3*), il permet de prendre des mesures de façon intelligente et de transmettre ces mesures grâce au protocole réseau LoRaWAN. Le noeud contient les capteurs suivants :

- capteur de température et d'humidité *HumiTemp* ;
- capteur de luminosité *Lumi* ;
- accéléromètre *Accelero* ;
- capteur de pression *Press*.

L'équipe précédente a également fait le choix d'y ajouter un pluviomètre.



FIGURE 3 – Nœud SoLo

Le nœud SoLo est configurable grâce à des fichiers au format JSON. Ces fichiers permettent de configurer la fréquence d'envoi, la fréquence de prise de mesures et le réseau. Mais ils permettent aussi de configurer les capteurs, ce qui est très intéressant dans le cas des scénarios, qui dépendent directement des données récoltées par ces capteurs. En effet, il est possible de définir ce que l'on appelle des alarmes, qui établissent des seuils particuliers pour lesquels la fréquence de mesure est modifiée.

La **gateway** sert simplement de point relais entre le nœud et le cluster. Après configuration, elle récupère les données du nœud et les transmet donc au cluster.

Le **cluster** utilisé est composé de **cinq Raspberry Pi** connectés entre eux grâce à un switch ethernet. C'est sur ce cluster que sont déployées toutes les applications de Mycélium permettant la sauvegarde et le traitement des données. Les technologies permettant la configuration du cluster et le déploiement des applications de Mycélium sur ce dernier ont été abordées dans le précédent rapport de pré-étude et spécifications [6], elles ne seront donc pas redéveloppées dans celui-ci.

3.1.2 Architecture matérielle de Mycélium 2.0

L'architecture matérielle de Mycélium 2.0 était illustrée en vert dans la *figure 2* présentée précédemment. Mycélium 2.0 a pour objectifs principaux d'étendre le système mis en place par Mycélium 1.0 et d'en améliorer les performances.

Ajout de capteurs Afin d'étendre le système actuel, l'ajout de nouveaux capteurs a été un choix d'appart évident.

Le premier capteur ajouté a pour but de rendre compte de la biodiversité de la faune de la Croix Verte, en partant de l'idée particulière de confirmer la présence d'un renard sur la zone. Ce capteur est constituée d'une caméra reliée à un Raspberry Pi. Lorsqu'un mouvement est détecté, la caméra prend une photo qui est envoyée au cluster via le protocole MQTT [7]. MQTT est un protocole de messagerie publish-subscribe basé sur le protocole TCP/IP, plusieurs exemples viendront approfondir cette définition tout au long du rapport. Le cluster est quant à lui chargé de détecter et d'identifier un potentiel animal sur la photo, dans le cas échéant il envoie une notification à l'utilisateur final. Ce premier capteur et les traitements associés ont fait l'objet d'un projet d'ouverture en IoT nommé FoxyFind [8] auquel plusieurs membres de Mycélium 2.0 ont participé.

Le second capteur constitue en réalité un ensemble de capteurs fournis par l'OSUR : un capteur multi-paramètre et un débitmètre. Cet ensemble de capteurs fournit de nombreuses informations sur les cours d'eau de la Croix Verte comme le débit de la rivière, la conductivité de l'eau, son pH... Ces mesures sont envoyées à un serveur interne à l'OSUR, c'est en faisant des requêtes à ce serveur que l'on accèdera à ces données.

Le fonctionnement de ces nouveaux capteurs et leur intégration sera expliquée de manière plus détaillée dans la seconde partie du rapport.

Scalabilité de Mycélium Il a été observé que le cluster de Raspberry Pi était trop souvent surchargé, ainsi dans un souci d'amélioration des performances, il est à présent souhaitable de déplacer une partie des traitements dans le Cloud. A cet effet nous avons ouvert un VPS (Virtual Private Server ou serveur privé virtuel) à la DSI (Direction du Système d'Information) de l'INSA. Dans le cas où le cluster est trop chargé, il fait appel au serveur virtuel qui le décharge d'une partie de ses traitements. La communication entre le serveur et le cluster se fait avec MQTT. La scalabilité de Mycélium grâce à celle du Cloud sera détaillée dans la seconde partie du rapport.

3.2 Architecture logicielle

L'architecture logicielle de Mycélium est illustrée dans la *figure 4* ci-dessous, où les capteurs propres à Mycélium 2.0 (capteurs de l'OSUR et FoxyFind) sont une fois de plus représentés en vert, et ceux appartenant à Mycélium 1.0 (Nœud SoLo relié à la gateway) en orange.

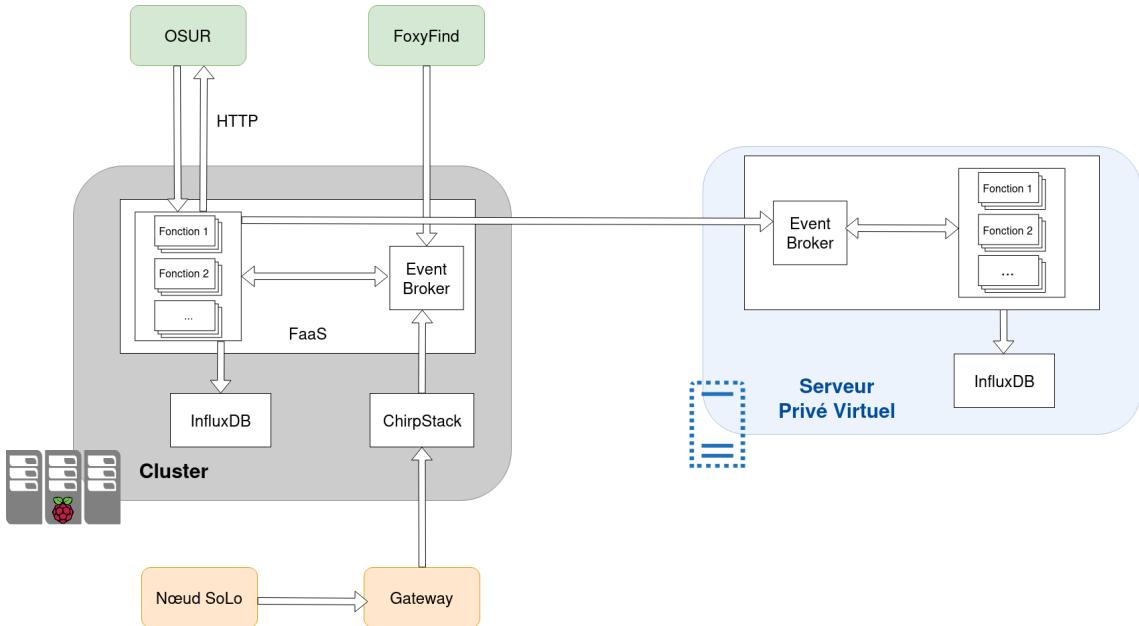


FIGURE 4 – Architecture logicielle de Mycélium

Mycélium Core, qui est constitué de tous les éléments logiciels nécessaires au fonctionnement du cluster, est représenté par le bloc cluster en gris sur la *figure 4*.

Un **miroir partiel de Mycélium Core** est déployé sur le serveur privé virtuel, représenté dans l'encadré bleu de la *figure 4*, de manière à ce que le serveur puisse effectuer les mêmes traitements que le cluster.

Concernant la pile logicielle déjà mise en place dans Mycélium Core 1.0, y figure tout d'abord ChirpStack [9]. Il s'agit d'un ensemble de logiciels open-source permettant la gestion et notamment la configuration de réseaux et d'appareils utilisant le protocole LoRaWAN. Cela est permis par une interface web permettant la gestion et l'intégration d'outils pouvant être utilisés avec LoRaWAN. Ainsi, grâce à ChirpStack, les données transitant par la gateway peuvent être utilisées au sein de Mycélium Core. ChirpStack publie les données reçues de la gateway sur un broker MQTT nommé "Event Broker". Plus précisément, il les publie dans un topic nommé auquel des clients MQTT (ici des fonctions) peuvent s'abonner pour recevoir les messages du topic. Un topic MQTT est une chaîne utilisée par le broker pour identifier et filtrer les messages de chaque client connecté. Le traitement des données est rendu possible grâce à une organisation du code en microservices serverless selon le principe de Function as a Service. En d'autres termes, chaque microservice est une fonction qui remplit une unique tâche. Une fonction permet de s'abonner au topic de l'Event Broker sur lequel ChirpStack a publié afin de récupérer les messages, les autres

fonctions permettent de décoder les messages, d'enregistrer les mesures dans une base de données InfluxDB [10], de repérer les scénarios et d'envoyer des notifications. Les fonctions peuvent s'appeler directement ou se servir de l'Event Broker pour communiquer. OpenFaaS[11] sera utilisé afin d'implémenter cette architecture.

Alors que la première partie de ce rapport se concentrat sur l'architecture globale de Mycélium, sa seconde partie détaillera plus spécifiquement le fonctionnement et l'intégration des nouveaux capteurs, l'organisation des fonctions FaaS, le transfert de charge au cluster et les nouveaux scénarios.

4 Ajouts de Mycélium 2.0

Cette seconde partie du rapport va exposer plus en détail la conception des ajouts de Mycélium 2.0. La première sous-partie présentera comment fonctionnent les nouveaux capteurs, leur interfaçage et leur intégration avec le système existant. Il sera d'abord vu que l'architecture de Mycélium 1.0 n'est pas adaptée à l'ajout de capteurs et par conséquent les ajustements qui y sont apportés seront exposés. Ensuite seront présentés le développement et l'intégration du projet FoxyFind, et enfin l'intégration des données des capteurs de l'OSUR. La seconde sous-partie permettra de détailler notre solution pour déplacer une partie du traitement dans le Cloud. La dernière sous-partie présentera pour finir les nouveaux scénarios mis en place, ainsi que le développement et l'intégration de ceux-ci en tant que fonctions FaaS.

4.1 Les nouveaux capteurs

Le choix d'ajouter de nouveaux capteurs vise à mieux comprendre la zone de la Croix Verte. Ce choix entraîne de nouveaux flux de données qui vont devoir être gérés par Mycélium Core. Comme énoncé précédemment, le premier flux provient du projet FoxyFind qui a pour but d'étudier la biodiversité de la Croix Verte, il s'agit d'une caméra détectant le mouvement et d'un traitement de reconnaissance d'animaux. Le second flux provient des capteurs fournis par l'OSUR, qui fournissent des informations sur les cours d'eau de la Croix Verte. Ce flux est particulièrement intéressant car l'un des objectifs de la renaturation de la Croix Verte est de faire remonter des cours d'eau sous-terrains sur trois points d'eau en surface. La section 4.1.1 va présenter les modifications apportées à l'architecture logicielle de Mycélium 1.0 afin de permettre l'ajout de ces nouveaux capteurs. La section 4.1.2 va se concentrer sur le projet FoxyFind. Enfin la section 4.1.3 va s'attarder sur les capteurs fournis par l'OSUR et l'intégration de ces derniers.

4.1.1 Architecture induite par les nouveaux capteurs

Aux données envoyées par le nœud SoLo via le protocole LoRaWAN s'ajoutent les données envoyées par la caméra de FoxyFind via Wi-Fi et les données des capteurs de l'OSUR récupérées sur leur serveur via des requêtes HTTP. Or l'architecture actuelle est centrée sur un seul et unique flux provenant du nœud SoLo. En effet, en analysant la *figure 5* sur la page suivante, on remarque que c'est la fonction *ChirpStack Handler* qui est chargée de décoder les données provenant de ChirpStack puis de republier ces données décodées sur le broker pour que la suite des traitements puisse avoir lieu, pour enregistrer les données par exemple. Or si de nouveaux capteurs étaient ajoutés, il faudrait que chaque fonction soit capable de traiter tous les différents types de capteurs. Cela entraînerait une trop grande complexité dans les fonctions et une maintenance trop compliquée.

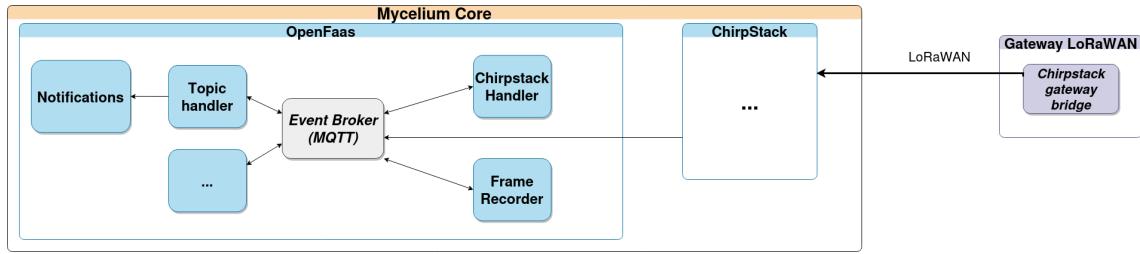


FIGURE 5 – Architecture du traitement serverless de Mycélium 1.0

Il a donc fallu repenser l'architecture de Mycélium Core afin que le système soit plus à même d'accueillir de nouveaux flux. L'architecture choisie est détaillée ci-après en *figure 6*. Chaque capteur ou nœud de capteurs envoie ses données via MQTT dans un topic qui lui est associé. Chaque topic a quant-à lui une fonction OpenFaas *Handler* propre permettant de décoder les données du topic. On dit de ces fonctions qu'elles sont "event driven", c'est à dire déclenchées par un évènement, en l'occurrence l'arrivée d'un message dans le topic. Chacun des handlers republie les données décodées dans un topic qui lui est propre, la fonction *Frames Recorder* souscrit à tous ces topics de données décodées pour les enregistrer en base de données.

Les fonctions *Scénario* peuvent s'appuyer sur les données d'un ou plusieurs capteurs, et peuvent appeler la fonction *Notifications* en cas de mesures exceptionnelles.

D'autres fonctions sont appelées à intervalles de temps réguliers, à la différence des fonctions "event driven". Ces fonctions peuvent consulter la base de données et faire appel à la fonction *Notifications* ou bien effectuer des appels HTTP, c'est par exemple le cas des fonctions *Regular Function* et *Handler Server*. Il faut noter que ce schéma simplifie les difficultés liées au réseau : certains capteurs, comme le nœud SoLo qui passe par la gateway puis par ChirpStack, ne peuvent pas directement envoyer les données via MQTT.

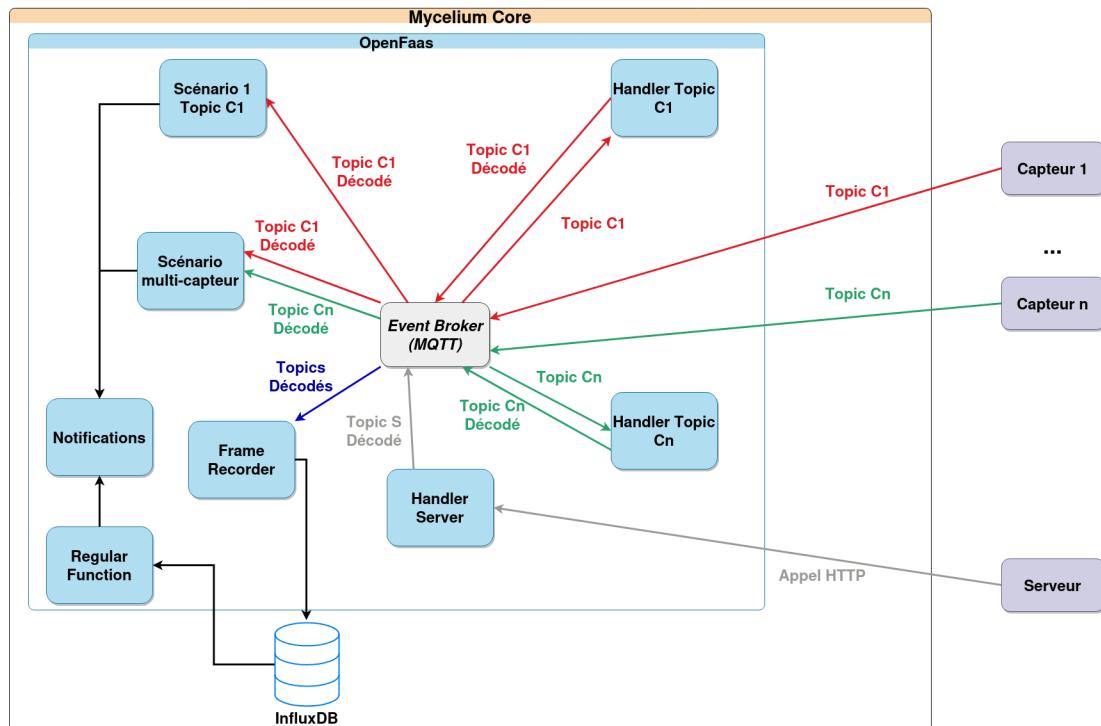


FIGURE 6 – Architecture du traitement serverless de Mycélium 2.0

4.1.2 FoxyFind

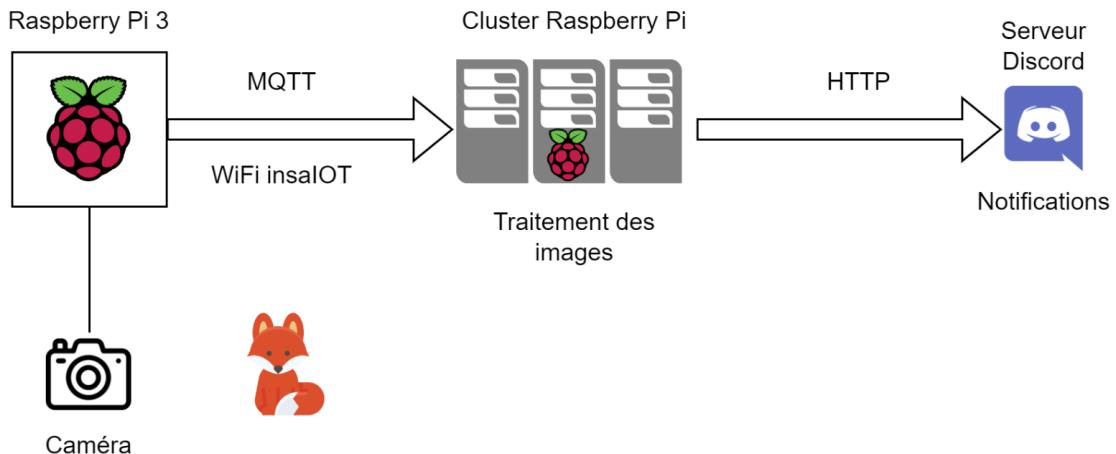


FIGURE 7 – Architecture du projet FoxyFind

Le projet FoxyFind est un projet développé par une équipe du module d'ouverture IoT. Il a pour but d'intégrer à Mycélium des capteurs vidéos afin de détecter et de reconnaître la faune indigène à la Croix Verte et plus particulièrement la détection d'un éventuel renard sur la zone. La *figure 7* ci-dessus présente l'architecture du projet FoxyFind. A noter que dans le cadre du module IoT, le traitement des images sur le cluster de Raspberry Pi est remplacé par un traitement sur ordinateur.

Tout d'abord, pour la détection du mouvement, un Raspberry Pi 3 auquel est connectée une caméra est utilisé. Afin de détecter un mouvement, celui-ci filme en continu son environnement ou plus exactement capture en continu l'environnement dans lequel il se trouve. Ainsi, une vidéo n'est qu'une suite d'images. Cela permet alors de détecter un mouvement dans le champ de vision de la caméra. En effet, pour détecter un mouvement, le Raspberry applique un calcul entre l'image actuelle et celle la précédant car une image n'est finalement qu'une matrice de pixels. Ce calcul permet de définir l'erreur quadratique moyenne entre deux images :

$$MSE = (image_{actuelle} - image_{précédente})^2$$

Si l'erreur quadratique moyenne ou MSE (Mean Squared Error) dépasse un certain seuil défini, alors une photo est prise et envoyée au cluster via le protocole MQTT. Une des limites de cette façon de faire est que la détection est très sensible aux conditions météorologiques de l'environnement où le Raspberry est utilisé. Effectivement, s'il se met à pleuvoir ou à venter, le Raspberry détectera un mouvement alors qu'il n'en est rien et prendra une photo. Celle-ci sera alors considérée comme un faux positif. Nous pourrions contourner ce problème avec un anémomètre pour adapter le seuil de détection avec la puissance du vent et de la même manière avec un capteur d'humidité. Mais là encore, il est difficile de définir la fonction qui donne un seuil avec un taux d'humidité et une vitesse du vent :

$$f(humidité, vent) = seuil$$

Dans le matériel mis à notre disposition figure le Sense HAT qui permet de détecter l'humidité et donc de minimiser le problème de la pluie.

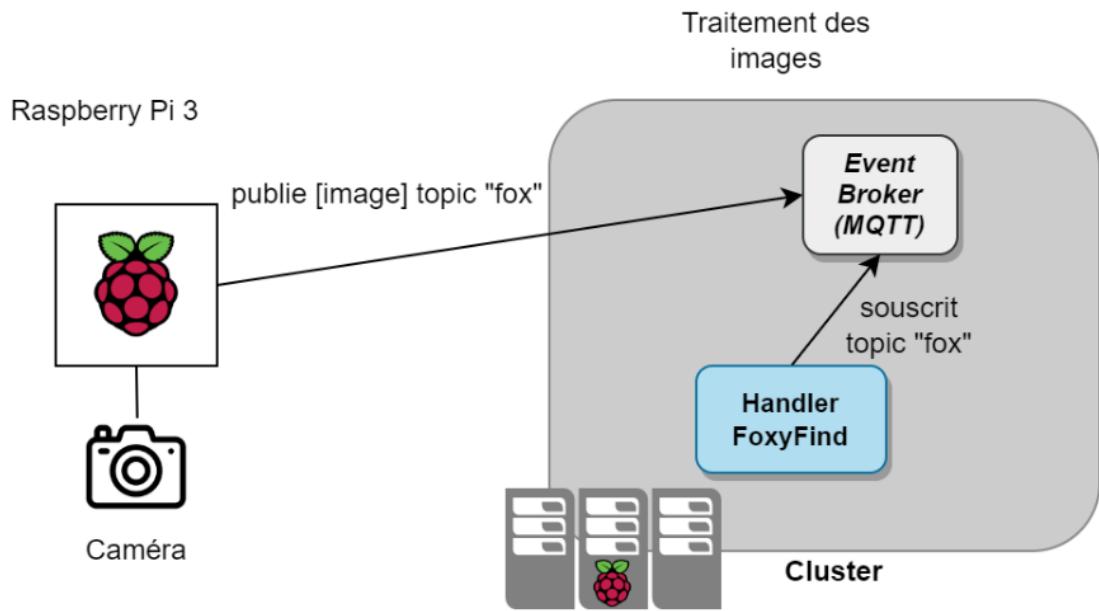


FIGURE 8 – Détail du transfert via le protocole MQTT

La *figure 8* présente la transmission des données via le protocole MQTT jusqu'à l'Event Broker situé sur le cluster de Raspberry Pi. Chaque fois qu'une photo est enregistrée sur le Raspberry, cette dernière est publiée dans un topic et envoyée au cluster de Raspberry Pi.

Le handler topic souscrit au même topic sur l'Event Broker utilisé que pour l'envoi des images. Ainsi, une connexion directe s'établit entre le cluster et le Raspberry Pi dédié à la capture d'images avec l'Event Broker agissant comme pont entre les deux. Lorsqu'une photo arrive sur le cluster, celle-ci déclenche une fonction OpenFaaS qui permet son traitement par le cluster. Une fois que la fonction a été traitée par le cluster, elle peut alors mener à deux comportements de la part du cluster :

- Un animal est détecté : une notification est envoyée via le protocole HTTP à un serveur Discord avec l'image et le nom de l'animal détecté ;
- Rien n'est détecté ou quelque chose est détecté mais il ne s'agit pas d'une détection significative : afin d'éviter l'encombrement de la capacité de stockage du cluster, la photo est supprimée.

4.1.3 Capteurs de l'OSUR

Mycélium 2.0 comme Mycélium 1.0 sont des projets réalisés en collaboration avec et pour l'OSUR. Pour une meilleure analyse de flux de données de la Croix Verte et dans le cadre du projet TERRA FORMA, l'équipe de l'OSUR avec laquelle nous travaillons a fait de nouveaux achats par rapport à l'an dernier. Ils ont notamment investi dans de nouveaux capteurs, qu'ils vont utiliser, mais mettent également à disposition de notre équipe :

- Un **débitmètre** : relevant des données sur le débit du cours d'eau, celles-ci sont directement transmises au serveur de l'OSUR, qui pourra ensuite les retransmettre à notre équipe pour être traitées ;

- Un **nœud multi-paramètre** : solution commerciale installée dans l'eau, il sert à mesurer sa conductivité, sa température, et sa pression. L'objectif de l'installation de ce nœud spécifique est de comparer les résultats de cette solution commerciale aux capteurs de l'OSUR ;
- **Plusieurs nœuds SoLo** : disposant de quatre nœuds SoLo, ils nous proposent également d'augmenter la taille du réseau de capteurs en fonction de nos envies et besoins dans le projet. Il peut en effet s'avérer intéressant d'avoir deux nœuds SoLo identiques mais servant à récupérer des données différentes. En effet, des données différentes fournies par des capteurs identiques situés à des endroits éloignés peuvent permettre d'obtenir des données plus précises en les moyennant. Des données variant trop d'un capteur à l'autre pourraient indiquer une anomalie ou un dysfonctionnement de l'un d'entre eux, permettant ainsi d'éviter certaines erreurs.

Il est donc intéressant pour nous dans le cadre de Mycélium 2.0 d'avoir accès à ces différents capteurs, et particulièrement aux données qu'ils fournissent. De plus, un plus grand nombre de capteurs permet une meilleure analyse et compréhension des données. Avec un **lien descendant** établi entre les capteurs et le cluster, certains capteurs pourraient réagir différemment en fonction du comportement analysé par un autre. Il s'avère que Ghislain NOUVEL de l'IETR travaille sur ce sujet avec des étudiants en Master 2 d'Ingénierie Durable des Bâtiments Communicants Intelligents à l'Université de Rennes 1 sur des nœud THr (température, humidité, luminosité) distribués par la société NKE [12]. Ils se penchent d'une part sur le reparamétrage de ce nœud, afin d'établir entre autres le lien descendant énoncé ci-dessus. D'autre part, ils s'intéressent au Firmware Update Over the Air (FUOTA), un processus permettant de mettre à jour le logiciel interne d'un dispositif sans avoir à le connecter à un ordinateur. Cela pourrait correspondre à la mise à jour du firmware de certains capteurs en passant directement par le réseau LoRaWAN. L'établissement de ce lien descendant et cette fonctionnalité de mise à jour pourraient à terme être utiles à Mycélium pour une meilleure maintenabilité du système.

Concernant l'intégration de ces nouveaux capteurs au reste du projet, une nouvelle fonction OpenFaaS sera appelée régulièrement, et non plus déclenchée par l'arrivée de données comme dans Mycélium 1.0. La **fonction** aura pour objectif de réaliser des **appels HTTP vers le serveur de l'OSUR**, afin de récupérer les données récoltées par leurs capteurs, puis de transmettre ces données aux fonctions d'analyse. L'exécution régulière de la fonction est permise grâce au *cron-connector* d'OpenFaaS, qui permet de définir le moment où la fonction sera exécutée à l'aide d'une expression régulière. L'objectif ici est de récupérer les données du serveur de l'OSUR avec la même fréquence que celles du nœud SoLo, c'est-à-dire toutes les heures. L'expression régulière utilisée est alors la suivante : "0 * * * *", construite selon la structure "minutes heures jourDuMois mois jourDeLaSemaine", avec un astérisque dans un champ représentant toutes les valeurs possibles pour ce champ.

4.2 Scalabilité de Mycélium

La gateway LoRa est branchée sur le cluster, or si le nœud principal du cluster lâche, tout le système cessera de fonctionner. Se pose alors un problème de **maintenabilité**. Une solution pourrait être de définir deux Raspberries maîtres et trois workers (contre un maître et quatre workers actuellement), afin que si un des nœuds principaux cesse de fonctionner, le système soit maintenu en état de marche. Cependant, cela n'a pas été réalisé l'an dernier car il est compliqué de mettre un tel fonctionnement en place dans

notre situation. En effet, 50% de la ressource en CPU est déjà utilisée lorsque le cluster ne fait aucun traitement. Cette ressource est utilisée par les simples installations minimales nécessaires au fonctionnement du cluster dans la configuration souhaitée pour le projet. Re-balancer le travail effectué par quatre workers sur seulement trois limiterait d'autant plus les ressources restantes. Une solution pour augmenter les ressources disponibles est donc la **scalabilité du Cloud** : c'est la capacité de modifier la quantité de ressources informatiques selon les besoins pour justement répondre à l'évolution de la demande.

Afin d'augmenter les ressources du projet en passant par la scalabilité du Cloud, nous avons fait le choix d'obtenir un **VPS**. Un VPS est une machine virtuelle, créée sur un serveur physique et employant ses ressources pour offrir à ses utilisateurs les mêmes fonctionnalités qu'un serveur dédié. Nous avons donc rapidement fait une demande d'accès à un VPS auprès de la DSI de l'INSA Rennes. Doté d'un cœur, de 2 Go de RAM, et de 30 Go d'espace de stockage, ce VPS permettra de combler le besoin en ressources de Mycélium 2.0. Grâce à la scalabilité du Cloud, on peut alors permettre la scalabilité de Mycélium, en ajoutant un complément de ressources évitant la surcharge du cluster.

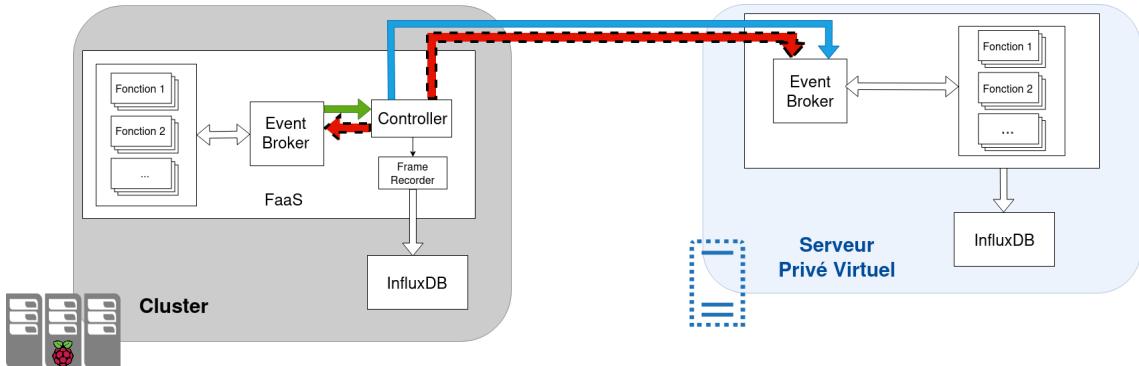


FIGURE 9 – Détail du transfert de charge entre le cluster et le serveur

La figure 9 ci-dessus illustre comment le transfert de charge a lieu. Sont déployées sur le cluster et sur le VPS : les mêmes fonctions Faas (à l'exception du *Controller*), ainsi qu'une base de données InfluxDB, afin qu'ils soient tous les deux capables d'effectuer les mêmes traitements. L'élément central de ce transfert de charge entre le cluster et le serveur est la fonction *Controller*, c'est par cette fonction que transitent toutes les données à traiter. Le *Controller* s'abonne à tous les topics de l'Event Broker du cluster (flèche verte), il fait appel à la fonction *Frame Recorder* du cluster afin d'enregistrer les données en local. Ensuite, il estime la charge actuelle du cluster : si cette dernière dépasse les 70% d'utilisation de la ressource en CPU, alors le *Controller* publie les données sur le topic *traitement* de l'Event Broker du VPS (flèche rouge allant vers le VPS) ; sinon il publie les données sur le topic *enregistrement* de l'Event Broker du VPS (flèche bleue) et il transmet les données aux fonctions FaaS du cluster (flèche rouge restant dans le cluster). Lorsque le VPS reçoit des données sur le topic *enregistrement*, il fait seulement appel à sa fonction *Frame Recorder*. Autrement, lorsqu'il reçoit des données sur le topic *traitement*, il fait appel à *Frame Recorder* et transmet les données aux fonctions de traitement. De cette façon, les deux bases de données contiennent normalement les mêmes données. Le choix a néanmoins été fait de ne pas vérifier que les deux bases de données contiennent effectivement les mêmes données car la perte de quelques messages ne risque pas de compromettre le système.

4.3 Nouveaux scénarios

Afin de mieux comprendre la zone de la Croix Verte dans son environnement, nous avons souhaité implémenter de nouveaux scénarios. Le premier scénario est le scénario *orage*, il est intéressant car le phénomène d'orage s'accentue et pourrait continuer de le faire dans les années à venir dû au réchauffement climatique. Le second scénario est le scénario *écart aux normales saisonnières*, il permet d'avertir l'utilisateur dans le cas où les mesures de température de la journée sortent des normales observées à la même période de l'année. Ce scénario permettra de mettre en évidence des épisodes de températures extrêmes qui sont de plus en plus fréquents.

4.3.1 Scénario *orage*

Afin de mesurer les épisodes orageux sur la Croix Verte, un scénario *orage* va être implémenté. Source de vigilance, un orage est un "phénomène atmosphérique, caractérisé par une série d'éclairs et de coups de tonnerre souvent accompagné par un ensemble de phénomènes violents : rafales de vent et précipitations intenses, il se forme lorsque l'atmosphère est instable, avec de l'air chaud près du sol et froid en altitude" selon Météo-France [13]. Causes d'éventuels dégâts et inondations, et de plus en plus fréquents, les orages sont donc grandement étudiés afin d'être prévenus.

Ce scénario est intéressant dans le cadre de notre projet car il va faire appel à plusieurs capteurs implantés sur la zone de la Croix Verte et à plusieurs paramètres différents. Ainsi, presque tout le système d'observation de la zone sera mis à contribution. Tout d'abord, un orage peut être mesuré par le volume de précipitation sur une période réduite sur le secteur à étudier. Pour faire de telles mesures, un pluviomètre connecté au Nœud SoLo sera utilisé et transmettra des données par LoRaWAN. De plus, les cours d'eau s'agitent avec le vent et les précipitations, le débitmètre installé par l'OSUR permet de surveiller l'état des plans d'eau. Les fortes pluies lavent également de nombreuses surfaces, déversant des déchets dans les cours d'eau : cela peut altérer le pH de l'eau, ce qui peut être analysé grâce au capteur multi-paramètre. Ce même capteur mesure également la pression de l'eau, permettant ainsi de contrôler la pression atmosphérique, qui augmente anormalement lors d'épisodes orageux. Enfin, les capteurs de température et d'humidité seront eux aussi sollicités pour comparer les mesures obtenues avec celles normalement observées sur la zone. Si un fort taux d'humidité couplé à une hausse de température assez soudaine sont observés sur la zone, ce changement abrupt dans le relevé sera lui aussi pris en compte pour définir ce scénario.

Le scénario *orage* sera implémenté grâce une fonction OpenFaaS : il s'agit d'un événement qui regroupe plusieurs capteurs. En effet, nécessitant les données de plusieurs capteurs, la fonction OpenFaaS devra traiter des données provenant de différentes sources, à des intervalles de temps pouvant varier, et sous différents formats qu'il faudra interpréter. L'exécution de la fonction sera alors lancée par l'arrivée de plusieurs messages sur l'Event Broker. Ainsi, la notification d'alerte sera déclenchée une fois que plusieurs seuils seront dépassés : humidité, pression, température... On pourra définir un niveau de vigilance associé à l'orage en fonction de la taille des écarts mesurés. De cette manière, la fonction utilisera des composants similaires à ceux des fonctions associées aux scénarios humidité et températures anormales.

Les orages étant souvent accompagnés de vents conséquents, il pourrait être très intéressant à l'avenir d'ajouter un anémomètre au réseau de capteurs disposés sur la Croix Verte. Cela apporterait des informations supplémentaires quant-à l'orage, mais permettrait aussi de donner naissance à un nouveau scénario : le scénario *tempête*. En effet, les épisodes de tempêtes étant fréquents en Bretagne et pouvant être ravageurs, il paraît important de les étudier eux aussi.

4.3.2 Scénario *écart aux normales saisonnières*

Dans un contexte d'urgence climatique, il nous est paru important d'ajouter aux scénarios existants un nouveau scénario rendant compte des changements climatiques. Ainsi nous est venue l'idée du scénario d'*écart aux normales saisonnières*. Les normales saisonnières font partie de ce que l'on appelle des normales climatiques. Selon Météo-France, elles "permettent de caractériser le climat sur un lieu donné, pour une période donnée. Il s'agit d'un produit statistique de référence pour l'étude du climat." Les normales saisonnières portent sur les mesures des 20 dernières années et sont actualisées tous les 10 ans.

De nouveau, ce scénario prend la forme d'une fonction OpenFaaS, cependant cette fois-ci l'exécution de la fonction n'est pas déclenchée par l'arrivée d'un message sur l'Event Broker. En effet la fonction est appelée une fois par jour grâce au *cron-connector* d'OpenFaaS avec l'expression régulière "`0 0 * * *`", ce qui signifie que la fonction sera appelée chaque jour à minuit. La fonction doit donc aller chercher la moyenne de la température de la journée qui vient de s'écouler. Cela se fait très facilement grâce à InfluxDB, dont le fonctionnement est basé sur le principe des séries temporelles, c'est-à-dire que toutes les mesures sont datées. Ainsi, en Python la requête suivante permet d'obtenir le traitement souhaité : on y remarque en ligne 2 l'intervalle de données sélectionné et en ligne 5 la fonction `mean()` qui fait la moyenne des mesures.

```

1  tables = query_api.query('from(bucket: "NormSais/autogen")
2      |> range(start: -1d, stop: now())
3      |> filter(fn: (r) => r._measurement == "temperature")
4      |> filter(fn: (r) => r._field == "value")
5      |> mean()')

```

Il ne reste plus qu'à vérifier que la moyenne obtenue se situe dans l'intervalle normal du mois courant, dans le cas contraire on appelle la fonction *Notifications*.

5 Conclusion

Pour conclure, ce rapport donne un aperçu de la conception du projet Mycélium 2.0. A la différence de Mycélium 1.0, Mycélium 2.0 est un projet centré sur l'analyse des données récoltées par le réseau de capteurs, plutôt que sur les capteurs eux-mêmes.

Il se base donc sur la conception de Mycélium 1.0, déroulant en première partie l'architecture globale, divisée d'une part en l'architecture matérielle, et d'autre part l'architecture logicielle. Pour chacune de ces parties, les fonctionnalités propres à Mycélium 1.0 ont été détaillées, en rappelant pour la première les fonctionnalités du noeud SoLo, et pour la seconde celles du cluster, notamment de Mycélium Core. Les ajouts majeurs apportés avec Mycélium 2.0 ont alors été présentés, afin d'être énoncés et mis en relation dans cette architecture globale avec l'existant. Le cœur de Mycélium 1.0 a été mis en fonctionnement à ce stade du projet, permettant l'avancée de Mycélium 2.0.

La seconde partie du rapport se focalise justement sur toutes les nouvelles fonctionnalités apportées par Mycélium 2.0. En termes de conception de Mycélium 2.0, il nous a fallu étudier la scalabilité du projet par le Cloud et mettre en place de nouveaux scénarios, suite à l'addition de plusieurs nouveaux capteurs au réseau existant. Pour la scalabilité, le VPS a été déployé, permettant une augmentation des ressources accessibles. Le module FoxyFind fonctionne et est en phase de test, afin d'être amélioré face aux événements météorologiques. Enfin, les fonctions de traitement décrivant les scénarios d'orage et de d'écart aux normales saisonnières ont été rédigées. Il reste cependant à manipuler ces fonctions pour qu'elles traitent les informations transmises par les nouveaux capteurs, et à déployer le tout en extérieur, face à des conditions réelles.

Ce rapport permet alors de visualiser le trajet des données, depuis leur création dans les différents capteurs disposés sur la Croix Verte, jusqu'à leur stockage et leur traitement. Le système a donc une architecture en microservice afin de contrôler le réseau de capteurs et développer des fonctions de traitement.

Table des figures

1	La Croix Verte	3
2	Architecture matérielle de Mycélium	4
3	Nœud SoLo	5
4	Architecture logicielle de Mycélium	7
5	Architecture du traitement serverless de Mycélium 1.0	9
6	Architecture du traitement serverless de Mycélium 2.0	9
7	Architecture du projet FoxyFind	10
8	Détail du transfert via le protocole MQTT	11
9	Détail du transfert de charge entre le cluster et le serveur	13

Références

- [1] “Site osur.” osur.univ-rennes.fr.
- [2] L. LAHMAR, “TERRA FORMA : un nouveau paradigme pour l’observation des territoires.” www.insu.cnrs.fr.
- [3] G. CHAUVEAU, P. IACONE, F. DABAT, E. JUVÉ, Y. TIAN, V. M. ELOSO RODRIGUES, and H. ZHANG, “RAPPORT FINAL MYCELIUM.”
- [4] “What is a raspberry pi ?.” <https://www.raspberrypi.org>.
- [5] G. CHAUVEAU, F. DABAT, P. IACONE, E. JUVÉ, Y. TIAN, V. M. VELOSO RODRIGUES, and H. ZHANG, “RAPPORT DE CONCEPTION.”
- [6] M. BOIZUMAULT, N. CAROFF, L.-P. JULIEN, T. RAOUL, I. TUMOINE, and G. VILLE, “RAPPORT DE PRÉ-ÉTUDE ET SPECIFICATIONS.”
- [7] “Mqtt documentation.” www.hivemq.co.
- [8] A. BEN MOUSSA, A. COCAN, M. GOUBET, L.-P. JULIEN, E. MOQUET, and I. TUMOINE, “RAPPORT FINAL FOXYFIND.”
- [9] “Chirpstack documentation.” www.chirpstack.io.
- [10] “Influxdb documentation.” www.influxdata.com.
- [11] “Openfaas documentation.” www.openfaas.com.
- [12] “Capteurs thr de la société nke.” capteur-thr-lora-nke.
- [13] “Site météo france.” www.meteofrance.com.

6 Annexe

6.1 Retard par rapport à la planification

Scénario intempérie Dans le rapport de planification, un scénario intempérie est évoqué. Suite à une discussion avec Laurent LONGUEVERGNE et l'ajout de plusieurs capteurs de leur part, il a été décidé de rebaptiser et de transformer le scénario intempérie en scénario orage. Cependant comme ce scénario nécessite des données de capteurs qui sont la propriété de l'OSUR et l'accès à un portail web pour récupérer ces données, à ce jour nous déplorons un retard dans cette tâche.

Intégration des capteurs de l'OSUR L'OSUR met à disposition de notre projet des capteurs afin d'agrandir notre parc de capteurs et de recueillir un plus grand nombre de données. Cependant comme nous ne gérons pas l'installation et la configuration de ces nouveaux capteurs, nous ne pouvons pas prévoir quand l'accès à ceux-ci nous sera accordé. Cela fait donc que nous accusons un retard par rapport à la planification initiale.