Mémoire de certification

Conception d'un système d'Objets Connectés
CNCP 1544

Système intelligent pour la certification biologique collaborative de production de plantes aromatiques

CULTYDATA

ALAOUI MOHAMED

Internet Of Things
IOT 2 – 2017/2018

Remerciements

Je tiens tout d'abord à remercier toutes les personnes qui m'ont accompagné durant ces derniers mois afin de réussir cette entreprise de certification à l'Ecole Polytechniques.

Je souhaiterais particulièrement rendre hommage à mon défunt père qui vient de nous quitter ces jours-ci ainsi qu'à mes proches qui m'ont encouragé tout au long de ce parcours.

Par ailleurs, je souhaite à mes collègues de la promotion IOT 2 ainsi qu'à l'équipe pédagogique de l'Ecole Polytechniques Executive Education tout le succès que méritent leurs projets et initiatives.

Résumé

Ce rapport contient un compte rendu détaillé des travaux qui ont été effectués suite au volet présentiel du Programme IOT de l'école Polytechniques Executive Education.

Dans la cadre de l'activité de la société CULTYDATA créée en novembre 2017, ce travail est un prototype d'un système intelligent pour une aide à la certification biologique collaborative de production de plantes aromatiques.

Table des matières

1 Introduction 5
1.1 Enjeu 5
1.2 Structure de ce mémoire 5
2 Description du projet 6
2.1 Caractéristiques des plantes aromatiques utilisées 6
2.1.1 Le Persil 6
2.1.2 La Coriandre 7
2.1.3 Le Basilic 7
2.1.4 La Ciboulette 8
2.2 Notion de certification biologique collaborative dans le contexte de ce mémoire 8
2.3 Environnement d'exploitation 8
2.3.1 Remarque sur l'exploitation 8
2.3.2 Préparation de l'exploitation 9
2.4 Périmètre du « POC » (Proof Of Concept) 9
3 Architecture fonctionnelle 10
3.1 Identification des acteurs de la plateforme 10
3.2 Cas d'utilisation 11
3.3 Cartographie des commandes et des évènements de la plateforme 12
3.3.1 Les cas d'utilisation nominaux 12
3.3.2 Cas d'utilisation de configuration 13
3.3.3 Cas d'utilisation lié à la certification 13
3.4 Schéma d'architecture fonctionnelle 14
3.5 Modèle de certification 14
3.6 Indicateur 15
4 Architecture technique 16
4.1 Généralités techniques 16
4.1.1 Besoins techniques 16
4.1.2 Architecture techniques 17
4.2 Détail du « POC » 18
4.2.1 Capteurs 18
4.2.2 Passerelle ou local Gateway 21
4.2.3 Reverse proxy 23
4.2.4 Middleware haute performance 23
4.2.5 Base de données de série temporelle 25
4.2.6 Outils de visualisation 26
4.2.7 Infrastructure 26
4.3 Problématiques à gérer pour finaliser un « MVP » 27
4.3.1 Blockchain 27
4.3.2 Sécurité 27
4.3.3 Fiabilité et robustesse 28
4.3.4 Gestion de l'énergie 28
4.3.5 Dimensionnement 29
4.3.6 Autres considérations 29
5 Conclusion et élargissement 30
6 Annexes 31
6.1 Luminosité [5] 31
7 Bibliographie 31

1 Introduction

1.1 Enjeu

La certification biologique d'une production agricole demande le respect d'un cahier des charges précis et d'une évaluation permanente.

Je propose dans ce mémoire de présenter un prototype qui permet une certification directe d'une production de plante aromatique à domicile ou dans un lieu à taille humaine.

Dans notre contexte, nous allons considérer une certification simplifiée, basée sur une notion de contrat associé à un élément de surface de terre et à un type de plante aromatique. Le modèle pourra être par la suite affiné pour englober toutes les facettes d'une certification industrielle.

1.2 Structure de ce mémoire

J'aborderais dans un premier temps le domaine concerné et les cas d'utilisation en précisant les types de plantes, la notion de certification collaborative, l'environnement d'exploitation ainsi que le périmètre du « Proof of Concept ».

Je définirai par la suite les fonctions que devraient remplir la plateforme pour réaliser cette auto certification en spécifiant l'architecture fonctionnelle du système et les flux d'informations.

Je traiterai enfin les aspects techniques en décrivant l'architecture applicative et sa projection matérielle en développant les sujets liés aux environnements contraints, à la chaine de valeurs « front to back », à la sécurité et la fiabilité du modèle et du système.

2 Description du projet

2.1 Caractéristiques des plantes aromatiques utilisées

Les semis de plantes aromatiques à cultiver sont les suivants :

Le Persil	La Coriandre	Le Basilic	La Ciboulette
			

Le périmètre de l'étude comprend la germination puis la pousse de la plante jusqu'à la cueillette. Les processus en amont de la germination comme la sélection des semis ainsi que la chaine de production post récolte ne seront pas étudiés dans ce mémoire. Il conviendra par la suite dans le cadre du développement de la société CULTYDATA d'étudier ces chaines afin de compléter le processus de certification.

Dans les sous sections suivantes, je décrirai les caractéristiques idéales permettant une récolte de plante aromatique de qualité. Les informations suivantes sont une consolidation de diverses recherches sur internet [1] [2].
Ces caractéristiques permettront de définir un contrat simplifié de qualité.

2.1.1 Le Persil

La température optimale de croissance est de 15 à 18°C. La levée est très lente, 18 à 30 jours selon les régions, la saison et la fraîcheur du sol. La température optimale du sol lors de la germination est de 24°C. Le persil a de grandes exigences en lumière. Il donne de meilleurs rendements sur les terres profondes, fraîches, argilo-siliceuses ou silico-argileuses, ainsi que sur des terres silico-calcaires riches en humus et en matières nutritives. Il exige un sol bien drainé à ph optimal 6,5. Il faut éviter les sols lourds, froids, très argileux ou pierreux et le fumier pailleux. La distance optimale entre les plants est de 15 à 30 cm.

A partir de ces données, un contrat de qualité peut être spécifié de la manière suivante :

Attributs du contrat	**Valeur**
contract-name	persil-biosmart-contract
optimal-germ-temp	24° +/-3°
optimal-growth-temp	15°-18° +/-1°
optimal-externalization-duration	18-30 +/-2j
optimal-humidity	60-70%
optimal-acidity	6.5 PH +/-0.2
optimal-luminosity	50 000 à 100 000 lux
optimal-distance	15-30 cm (10-15 plantes/m2)

2.1.2 La Coriandre

La coriandre est surtout influencée par la zone géographique de culture. Dans un froid extrême et une courte saison de végétation exceptionnellement en Norvège et en Sibérie, le rendement en huile volatile de coriandre est supérieur à celui de plusieurs pays dans le centre et le sud de l'Europe. Les meilleurs rendements ont été obtenus dans les étés frais et assez humides. Les différentes informations sur internet permettent d'initier son contrat de qualité de la manière suivante :

Attributs du contrat	**Valeur**
contract-name	persil-biosmart-contract
optimal-germ-temp	24° +/-3°
optimal-growth-temp	15°-18° +/-1°
optimal-externalization-duration	18-30 +/-2j
optimal-humidity	60-70%
optimal-acidity	6.5 PH +/-0.2
optimal-luminosity	50 000 à 100 000 lux
optimal-distance	15-30 cm (10-15 plantes/m2)

2.1.3 Le Basilic

Les graines de basilic ont besoin d'une température entre 20 et 25 °C pour déclencher la germination. Le basilic pousse mieux dans des conditions de température très chaude et en plein soleil. Toutefois, une meilleure qualité de feuilles est obtenue avec un minimum d'ombre. Avec des températures quotidiennes supérieures à 27 °C, les plantes doivent être ventilées ou recouvertes d'un filet d'ombrage d'environ 20 %, surtout au cours des fortes épisodes de chaleur si les rayonnements solaires sont important.
Les différentes informations sur internet permettent d'initier son contrat de qualité de la manière suivante :

Attributs du contrat	**Valeur**
contract-name	basilic-biosmart-contract
optimal-germ-temp	22° +/-1°
optimal-growth-temp	20°-25°
optimal-externalization-duration	5-6 semaines
optimal-humidity	60-70%
optimal-acidity	5 - 6.5 PH +/-0.2
optimal-luminosity	25 000 à 75 000 lux
optimal-distance	15-25 cm

2.1.4 La Ciboulette

La ciboulette est une plante rustique, de saison froide, résistante au gel et à la sécheresse.
Les différentes informations sur internet permettent d'initier son contrat de qualité de la manière suivante :

Attributs du contrat	**Valeur**
contract-name	ciboulette-biosmart-contract
optimal-germ-temp	22° +/-1°
optimal-growth-temp	20°-25°
optimal-externalization-duration	5-6 semaines
optimal-humidity	60-70%
optimal-acidity	5 - 6.5 PH +/-0.2
optimal-luminosity	25 000 à 75 000 lux
optimal-distance	15-25 cm

2.2 Notion de certification biologique collaborative dans le contexte de ce mémoire

Les contrats définis précédemment ne garantissent évidemment pas une qualité biologique absolue. Ils permettent par ailleurs de définir un point de départ pour construire une plateforme ayant l'ambition de monitorer la culture de ces plantes aromatiques.
Les différents attributs mentionnés ne sont pas exhaustifs. Pour des améliorations futures, certains indicateurs informant sur la qualité biologique du sol peuvent être ajoutés.

L'idée de la plateforme est de définir un modèle simple permettant d'évaluer l'état de la culture durant son cycle de vie qui, dans notre cas, englobe la germination et la croissance jusqu'à la récolte. Cette évaluation permettra la délivrance d'un certificat de production mentionnant la « distance » par rapport au contrat idéal établi précédemment. Je définirai cette notion de distance par la suite.

Le contrat idéal pourra être ajusté par apprentissage après une évaluation qualitative et collaborative fournie par les consommateurs des plantes récoltés.

2.3 Environnement d'exploitation

Dans cette partie, je propose de décrire l'environnement d'exploitation et sa préparation afin de permettre le bon fonctionnement de la plateforme.

2.3.1 Remarque sur l'exploitation

Afin de simplifier nos cas d'utilisation, l'exploitation est considérée à taille humaine. Elle permet la culture de plantes aromatiques pour une consommation de proximité.

Je considère par ailleurs qu'un capteur est capable de fournir des indicateurs pour un type de plantes aromatiques et pour une unité de production. Une unité de production est un espace sol/air homogène en termes d'indicateurs. Ainsi, je peux identifier une unité de production par son capteur.
Dans le contexte de ce mémoire, l'unité de production correspond à un bac en polyester pour la phase de germination. Elle est associée à un grand pot ou à une surface de jardin pour la phase de croissance.

Chaque unité de production aura une carte d'identité descriptive associée à un QR code. Ce dernier permet aussi d'identifier le capteur.

Ainsi, pour récapituler, la structure analytique de l'environnement d'exploitation est la suivante :

- Une plateforme est associée à un ensemble d'exploitation
- Une exploitation possède une ou plusieurs passerelles IOT vers internet
- Une exploitation possède un ensemble d'unité de production
- Une unité de production est associée à un capteur

2.3.2 Préparation de l'exploitation

Les éléments utilisés lors de la phase de germination sont les suivants :

- Un bac de polyester (respect d'un range de température)
- Une vitre « effet de serre »
- Un plastique noir régulateur de luminosité pour le sol

Les éléments utilisés lors de la phase de croissance sont les suivants :

- 2/3 de terreau 1/3 de terre du jardin
- Couche drainante en bas du pot : bille d'argile ou cailloux

La politique d'arrosage pourra être traitée automatiquement via un système prédictif de l'humidité et de la température.

L'environnement d'exploitation IOT sera décrit dans la partie technique de ce mémoire.

2.4 Périmètre du « POC » (Proof Of Concept)

Les contraintes de temps ne permettent pas d'élaborer un « MVP » (Minimum Viable Product) complet. J'ai donc opté pour l'élaboration d'un « POC » d'une partie de la colonne vertébrale de la plateforme qui est composé des éléments suivants :

- Un ensemble de capteurs (2 dans notre cas)
- Une passerelle de localité (une par exploitation demandant une connexion internet)
- Un micro service en reverse proxy internet, frontal des composants du serveur
- Un middleware performant permettant une montée en charge
- Un ensemble de micro services pour l'exploitation des évènements du middleware
- Un système de stockage de series temporelles

- Une interface graphique pour visualiser les séries temporelles

Afin de compléter le « POC » en « MVP », il faudrait ajouter :

- Une blockchain (en cours de développement)
- Une base de données et une interface d'administration pour la gestion des droits et l'évaluation qualitative des récoltes
- Un système d'intelligence artificielle pour la prédiction de séries temporelles pour l'asservissement de l'arrosage
- Un système d'intelligence artificielle pour ajuster la définition des contrats selon l'évaluation qualitative des récoltes
- Un environnement de déploiement industriel en continue notamment pour la partie serveur

Par ailleurs, pour une industrialisation de la plateforme, il sera nécessaire de traiter et d'implémenter les fonctionnalités liées à la sécurité et la fiabilité du modèle et du système ainsi que d'opérer les optimisations nécessaires pour tenir compte des caractéristiques d'un environnement contraint du fait de :

- La gestion de la mémoire
- La capacité de traitement
- La gestion de l'énergie au niveau des capteurs et de la passerelle de localité

Certains éléments concernant ces sujets seront précisés par la suite.

3 Architecture fonctionnelle

3.1 Identification des acteurs de la plateforme

Différents acteurs entrent en jeu dans les processus liés à la certification d'une unité de production, en voici une liste qui pourrait être enrichie par la suite :

Acteurs	**Rôle**
Operateurs	• Mettre en place l'exploitation • Gérer les tâches manuelles liées à l'exploitation • Configuration et maintenance de la plateforme
Capteurs	• Relever des indicateurs • Action mécanique sur une unité de production tel que l'arrosage par exemple
Passerelles de localité (Local Gateway)	• Broker multi-unité de production • Construction du Payload
Servers	• Système distribué pour la gestion de la certification biologique collaborative

Ces acteurs sont responsables de plusieurs tâches afin de pouvoir délivrer in-fine une certification. La réalisation d'une tâche peut être la conséquence d'une commande « COM » ou d'un événement « EV ». La nuance réside principalement dans le fait que la commande est voulue par un acteur donné alors qu'un évènement est quelque chose qui s'est passée au sein de la plateforme et qui peut entrainer d'autre évènements.

3.2 Cas d'utilisation

Les différents cas seront numérotés afin de pouvoir les référencer par la suite. Je ne serai pas exhaustif sur les cas d'utilisation, je me concentrerais sur les principaux.

Reference du cas d'utilisation	**Nom**	**Description**
001-UC	onboard-production-unit	L'enregistrement d'une unité de production permet de définir la carte d'identité d'une unité de production en précisant : • La localisation • Le type de plante • La superficie • Le type de sol • Les caractéristiques du capteur • Les jalons temporels
002-UC	init-smart-contract	Permet de configurer les instances de contrat associées à chaque unité de production
003-UC	**configure-connected-object**	**Permet de configurer les paramètres du capteur afin d'assurer et d'optimiser la relevée des indicateurs (activation, désactivation, tick de relevé…)**
004-UC	compute-smart-contract	Permet d'enrichir le contrat par les évènements liés à l'unité de production
005-UC	**persist-indicator**	**Permet de stocker un indicateur dans une structure adaptée**
006-UC	**view-indicator**	**Permet de visualiser les indicateurs dans une interface utilisateur**
007-UC	evaluate-production-unit	Permet d'évaluer la production par les consommateurs
008-UC	update-smart-contract	Permet de mettre à jour le contrat initial à l'aide des évaluations des consommateurs
009-UC	compute-certificate	Permet le calcul du certificat qui sera fournis aux consommateurs avec la carte d'identité de l'unité de production
010-UC	predict-indicateur	Permet le « Forecast » d'une serie temporelle pour une aide à la décision

012-UC	activate-water	Permet un arrosage asservi
012-UC	view-ar	Permet de visualiser en direct l'exploitation via une réalité augmentée

Le « POC » concerne les cas d'utilisation suivants :

- **003-UC-configure-connected-object**
- **005-UC-persist-indicator**
- **006-UC-view-indicator**

Dans les sections suivantes, je traiterais principalement ces cas d'utilisations. Je me permettrai par ailleurs de mentionner certains points liés aux autres cas.

Aussi, par soucis de simplicité et en cohérence avec le « POC », les indicateurs qui seront mentionnés sont la température (T) et l'humidité (H).

3.3 Cartographie des commandes et des évènements de la plateforme

3.3.1 Les cas d'utilisation nominaux

Considérons par exemple les cas **005-UC-persist-indicator** et **006-UC-view-indicator.**
Les commandes seront suffixées par COM et les événements par EV.

Tempo	**Description des commandes et des évènements**	**User Interface**	**Operateur**	**Capteur**	**Local Gateway**	**Server-reverse-proxy**	**Server-middleware**	**Server-storage**
t1	L'opérateur active le capteur par une commande de déploiement ou de configuration		Activate.sensor.COM					
t2	Le capteur effectue les différents relevés et envoi un payload initial en sans fil			T.H.sent.EV				
t3	La passerelle reçoit le payload				T.H.recieved.EV			
t4	La passerelle expose le payload à internet				T.H.pushed.EV			
t5	Le payload est récupéré par le frontal du server					T.H.consumed.EV		
t6	Le payload est enrichi puis envoyé à un middleware haute performance						T.H.pushed.EV	
t7	Le payload est persisté dans une base de données							T.H.consumed-stored.EV
t8	Le payload est pushé dans une interface pour pouvoir le visualiser	T.H.viewed.EV						

Cette méthodologie pourrait être utilisée pour identifier toute les commandes et évènements du système.
Cette cartographie, servira de base pour définir un vérificateur de modèle pour rendre la plateforme fiable et plus robuste.

3.3.2 Cas d'utilisation de configuration

Concernant le cas **003-UC-configure-connected-object** :

Tempo	Description des commandes et des évènements	User Interface	Operateur	Capteur
t1	L'opérateur active le capteur par une commande de déploiement ou de configuration		Configure.sensor.COM	
t2	Le capteur effectue les différents relevés et envoi un payload initial en sans fil	conf.payload.sent.EV		
t3	La passerelle reçoit le payload			conf.payload.recieved.EV
t4	La passerelle expose le payload à internet			conf.payload.done.EV

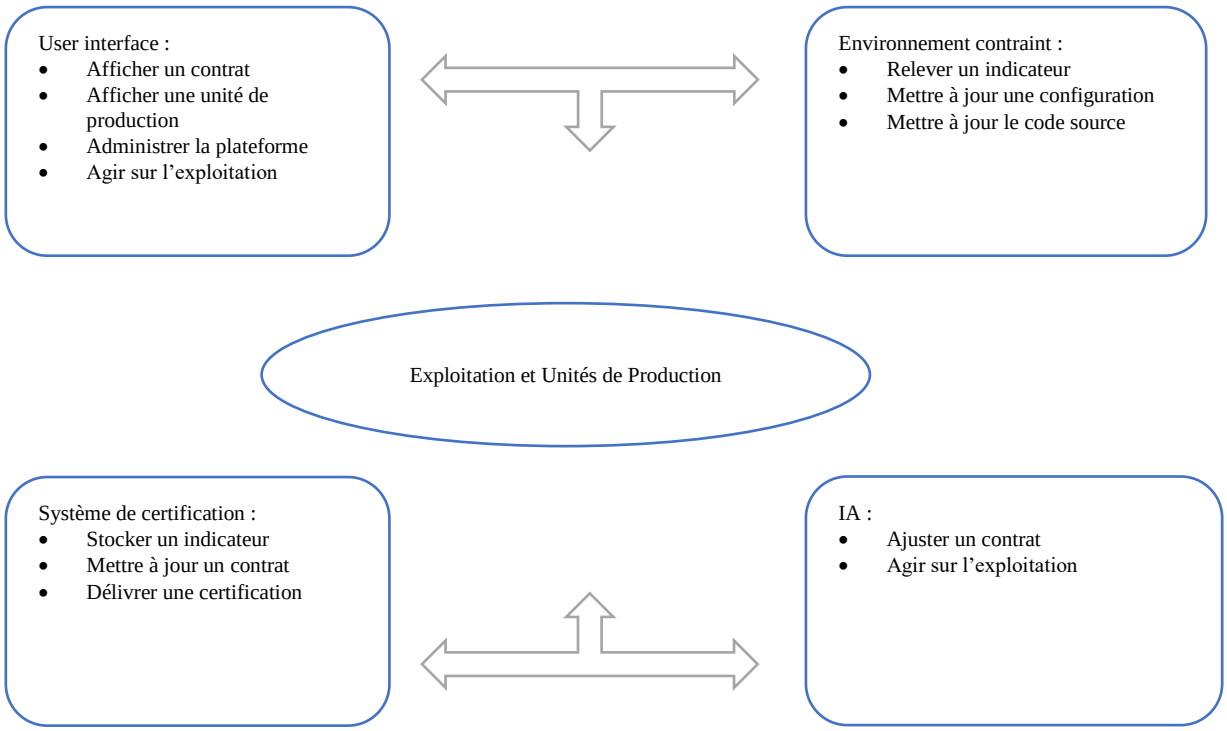
3.3.3 Cas d'utilisation lié à la certification

Afin d'étayer le fonctionnement de la plateforme, je propose de décrire les commandes et évènements liés à la certification et plus précisément au cas **003-UC-compute-smart-contract :**

Tempo	Description des commandes et des évènements	User Interface	Capteur	Local Gateway	Server-reverse-proxy	Server-middleware	Server-blockchain-service
t1	Le capteur effectue les différents relevés et envoi un payload initial en sans fil		T.H.sent.EV				
t2	La passerelle reçoit le payload			T.H.recieved.EV			
t3	La passerelle expose le payload à internet			T.H.pushed.EV			
t4	Le payload est récupéré par le frontal du server				T.H.consumed.EV		
t5	Le payload est enrichi puis envoyé à un middleware haute performance					T.H.pushed.EV	
t6	Le payload est persisté dans une blockchain et met à jour le contrat associé						T.H.consumed-computed.EV
t7	Le contrat reflète l'état de l'unité de production en live	Live.contract.updated.EV					

3.4 Schéma d'architecture fonctionnelle

Le schéma suivant représente les différentes fonctions du système.



Les choix d'implémentation de ces fonctions sont la pierre angulaire de ce mémoire et seront traitées et justifiée lors de la partie liée à l'architecture technique.

3.5 Modèle de certification

Dans un premier temps, le modèle de certification consiste en un score de conformité calculé à partir de l'inverse des cumules des écarts quotidiens constatés par rapport au contrat.
Un score de 10 au niveau de la certification correspond à une conformité parfaite au contrat durant toute la production.

Prenons comme exemple le persil :

persil-biosmart-contract :

- optimal-germ-temp = 21-27
- optimal-growth-temp = 14-19
- optimal-humidity = 60-70

Les relevés montrent une température de germination de 18°C pendant 5h et de 30 pendant 24h puis une température de 21°C en phase de croissance pendant 10h puis une humidité de 55% pendant 48h. Les autres relevés ne présentent pas d'écarts par rapport au contrat.

Ainsi, on peut calculer une pénalité cumulée et lissée sur 24h de la manière suivante :
Pen = [(21-18)*5 + (30-27)*24+(21-19)*10+(60-55)*48]/24 = 347/24 = 14,458

On en déduit un score :
Score = 10-1/347 = ***9,931.***

Ce modèle sera normalisé et revu par apprentissage après plusieurs récoltes, consommations et évaluations d'un certains nombres d'unités de production.

3.6 Indicateur

Les contrats réagissent à différents indicateurs relevés par les capteurs. Les indicateurs qui intéressent les contrats définis au préalable sont les suivants :

- Température (direct) : T
- Humidité (direct) : H
- Luminosité (direct) : L
- PH acidité du sol (direct) : PH
- Image camera (indirect) : I

Dans le « POC », seules la température et l'humidité ont été traitées.

4 Architecture technique

4.1 Généralités techniques

4.1.1 Besoins techniques

Dans cette section, je listerai l'ensemble des besoins techniques avec un descriptif des choix effectués. Dans les sections suivantes, je détaillerai certains de ces choix.

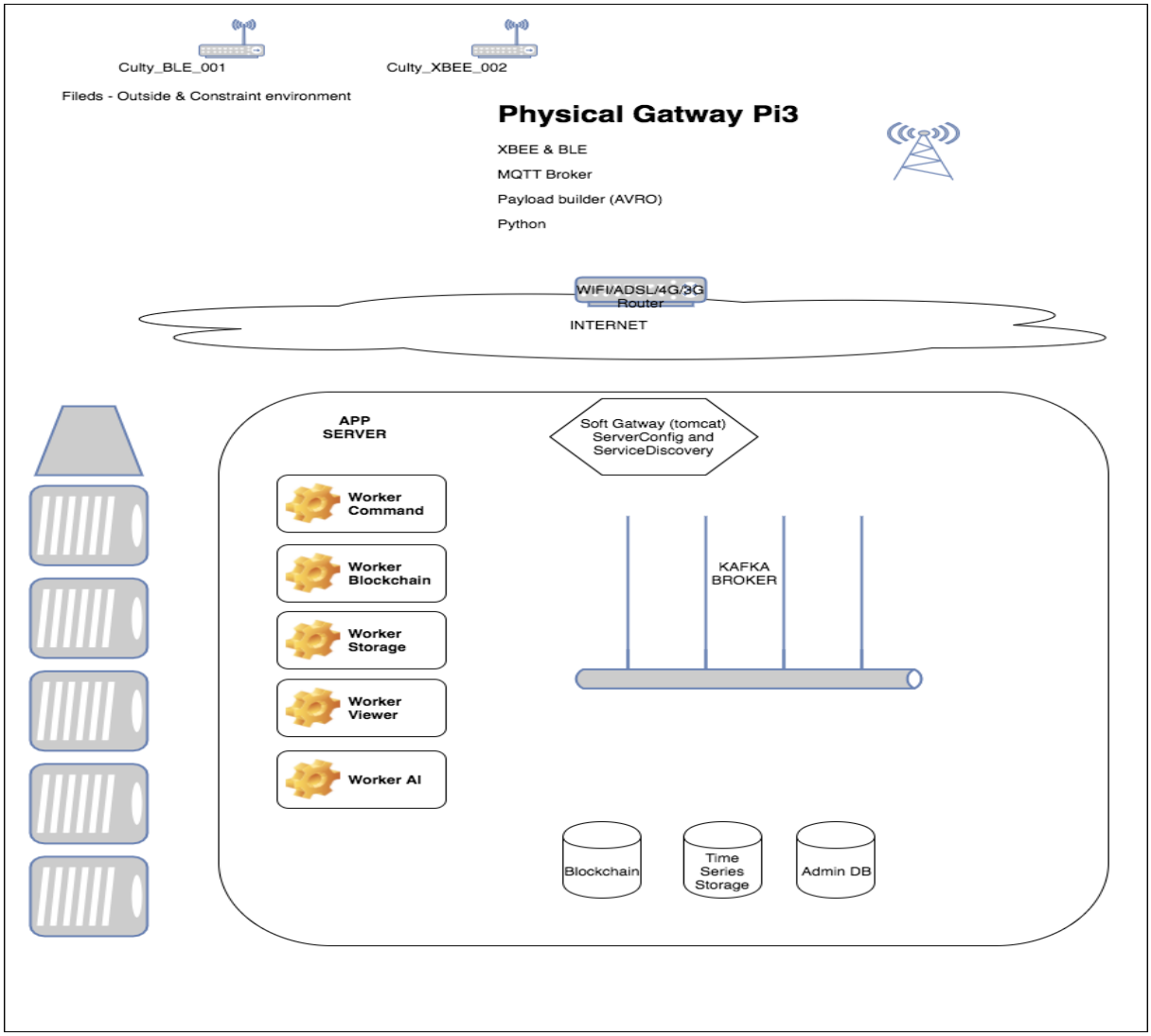
Besoins	**Composant**	**Rôle**	**Technologies**
Prototypes de capteurs	2 Arduino Uno R3 1 avec module BLE 1 avec XBEE PRO S2C Capteurs de températures/humidité DHT11	Relève des indicateurs de température et d'humidité et envoi à période régulière en BLE pour une unité de production et en Zigbee pour une deuxième unité de production	Code C dans l'IDE Arduino avec freertOS pour le multi-thread. Ce choix est essentiellement justifié par la relative facilitée du prototypage. Les protocoles choisis répondent au besoin d'un environnement d'exploitation à taille humaine
Paserelle ou local Gateway	Raspberry Pi 3 modèle B	Réception d'évènements BLE et Zigbee. Heberge un broker MQTT pour exposer les évènements à un système distribué hautes performances	Mini server Python permettant une écoute sur le port série du pi3 et une consommation des évènements via BLE pour ensuite produire des évènements sur des topics MQTT locaux. Broker MQTT jouant le rôle d'IOT middleware pour une gestion performante et sécurisée des évènements issues de capteurs de l'exploitation. Les évènements sont ainsi exposés par le broker à internet.
Reverse proxy server ou API Gateway	CYDBRIDGE : MQTT bridge micro service	Permet une rupture de protocole et une intégration entre la partie purement IOT et la partie serveur d'application	Java Spring Boot avec spring integration. Permet une architecture d'entreprise performante, scalable et ouverte grâce à l'écosystème JAVA
Middlware haute performance	KAFKA message broker	Bus de message event-driven hautement performant	KAFKA et ZOOKEEPER permet une architecture robuste event-driven avec des messages immuables et une gestion des topics flexible avec plusieurs fonctionnalités intéressantes (partition de topics, groupe de consommateur pour la consommation multithread, politique de persistance et de retour sur historique, streaming et windowing pour la gestion des prédictions de série temporelle)
Base de données serie temporelle	CULTYDATA DB CYDBRIDGE : Store micro service	Stockage des séries temporelles	InfluxDB est la base du moment pour les séries temporelles. Elle permet une gestion robuste des problématiques autour des stockages de séries temporelles
Outils de visualisation	CULTYDATA Dashboard	Visualisation des séries temporelles	GRAFANA est un outil de visualisation hautement configurable et facilement intégrable avec divers outils de stockage de données

Blockchain	CULTYDATA CertiBC CYDBRIDGE : CertiBC micro service	Gestion des biosmart contract et délivrance de la certification d'une unité de production	Etherium privé (ou en consortium). Cette blockchain permet d'implémenter des systèmes distribués sécurisés et des smart contracts

Dans un second temps, l'architecture sera enrichie par les composants suivants :

- Un outil de « service discovery » permettant une gestion dynamique des points d'entrées des différents services
- Un service de configuration facilitant le déploiement coté serveur. Une extension à la partie IOT est envisageable sous certaines conditions.
- Une base de données d'administration et de gestion des droits avec son interface utilisateur web (mobile)
- Un outil de visualisation en réalité augmenté directement au niveau de l'exploitation en utilisant la technologie AR Core.
- Un outil d'intégration continue ou de déploiement continu
- Un outil de gestion de l'infrastructure

4.1.2 Architecture techniques



4.2 Détail du « POC » [3]

4.2.1 Capteurs

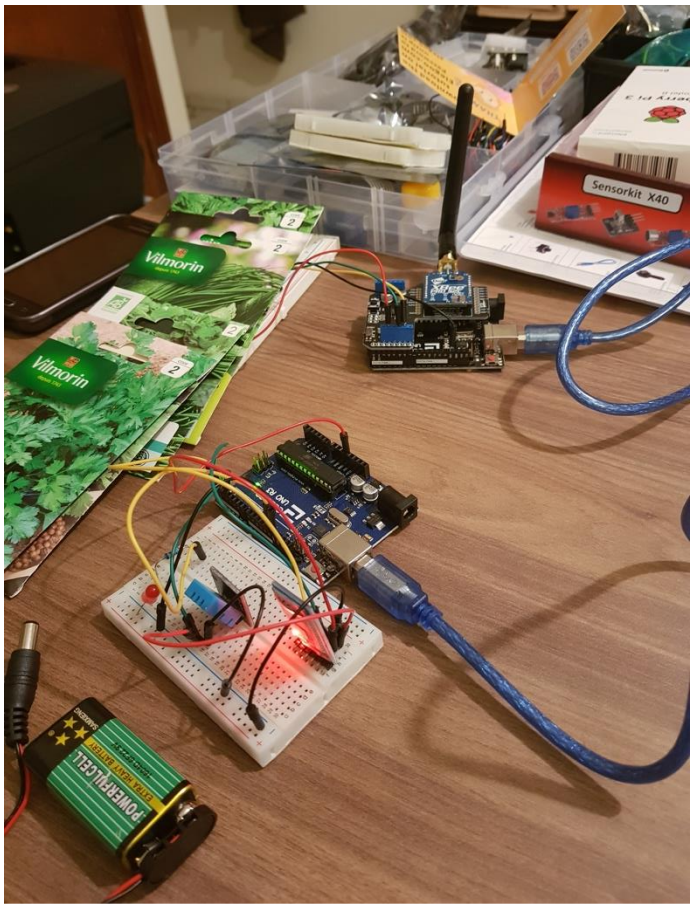


Figure 1-Dispositif avec 2 Arduino, un module BLE et un module Xbee

Les microcontrôleurs utilisés sont des Arduino UNO R3. Le capteur de température et d'humidité est le DHT11.

Les choix des protocoles de communication IOT ont été dictés par l'environnement de l'exploitation. Les unités de production que l'on souhaite certifier s'étalent sur des surfaces à taille humaine. Des protocoles de type « PAN » (Personal Area Network) ou « LAN » (Local Area Network), peu couteux en énergie répondent à notre besoin en termes de capteurs.

Les protocoles « Bluetooth Low Energie » et ZigBee 802.15.4 sont 10 à 100 fois moins gourmands en énergie que le Bluetooth traditionnel. La consommation est comprise entre 10 et 500 mW. Par ailleurs leur portée est suffisante pour nos cas d'utilisation. En effet, ZigBee a une portée de 77 mètres et le BLE de 50 mètres environ.

Par ailleurs, le débit du BLE est de 1 Mbit/s. En cas de traitement applicatif pour relayer un évènement, le débit est réduit à 270 Kbit/s. Le débit Zigbee est quant à lui de 250 Kbit/s. Contrairement au BLE qui a une fonctionnalité « sleep », le Zigbee reste actif entre deux

rafales. Il est possible de relayer un message via un nœud Zigbee si la taille de l'exploitation l'oblige cependant il faudrait l'éviter en cas d'un réseau dense pour éviter les embouteillages.

Lors de la réalisation du « POC », la partie ZigBee n'a pas été finalisée. L'effort a été concentré sur le BLE.

4.2.1.1 BLE

Le module utilisé est le HM-10 avec un chip Texas Instrument. Sans rentrer dans les détails des spécifications du module et du fonctionnement du BLE, il est important de noter que le BLE permet une communication bidirectionnelle via un service UUID qui possède une caractéristique UUID permettant la notification, la lecture ou l'écriture de 20 octets non formatés.

Quand un objet veut envoyer une donnée au module, il écrit dans la caractéristique UUID.
Quand le module veut envoyer une donnée, il envoie une notification à l'objet connecté.
Le module BLE gère sa connexion comme un port serie standard RX/TX.

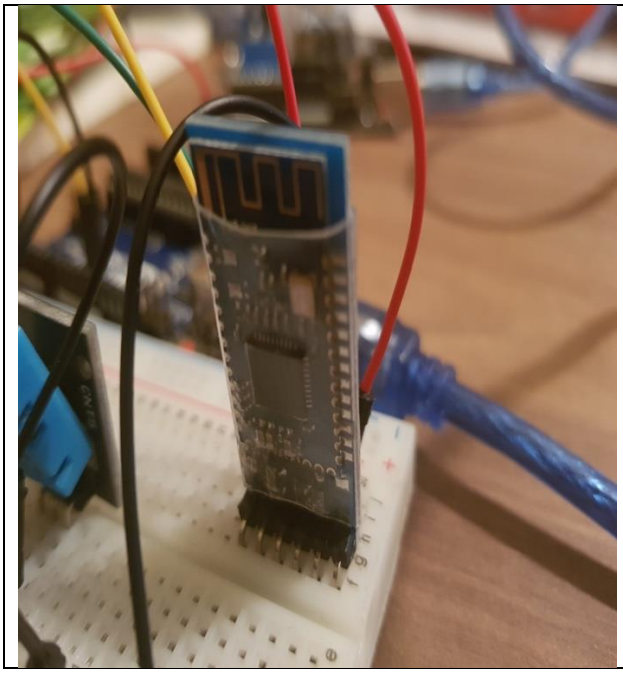


Figure 2-Module BLE

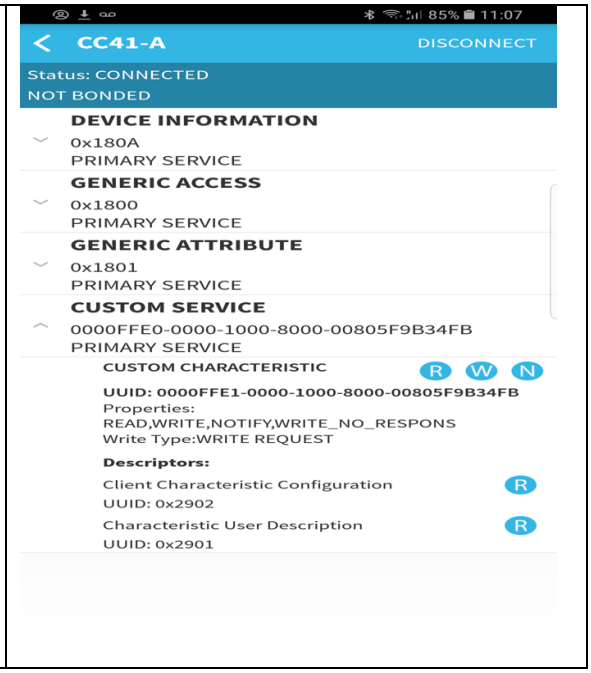


Figure 3-Application Android pour BLE

4.2.1.2 Code Arduino

Le code C permettant la récupération des indicateurs de température et d'humidité est écrit via l'IDE Arduino puis téléversé dans le microcontrôleur : [Lien github]

Les lib DHT11 et freertOS sont utilisées pour gérer les différentes tâches comme la configuration et le relevé des indicateurs. L'utilisation de freertOS permet de préparer aussi l'arrivée de nouveaux indicateurs.

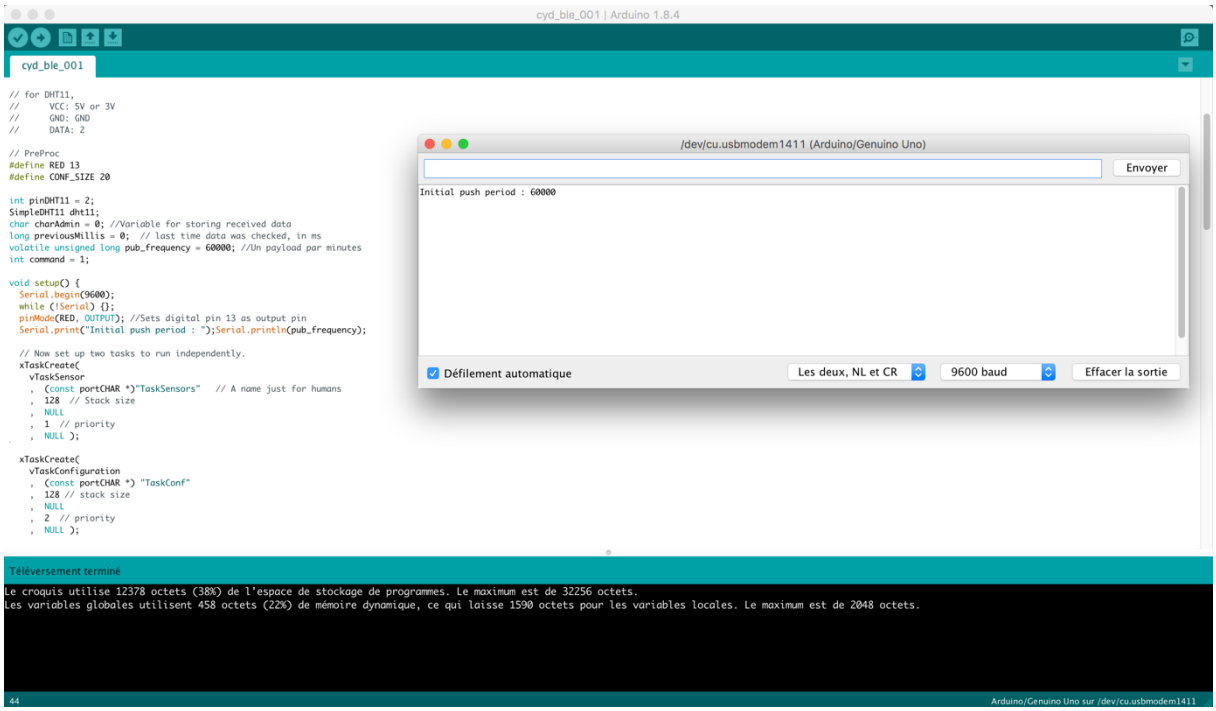


Figure 4-Code Arduino, Log du moniteur série

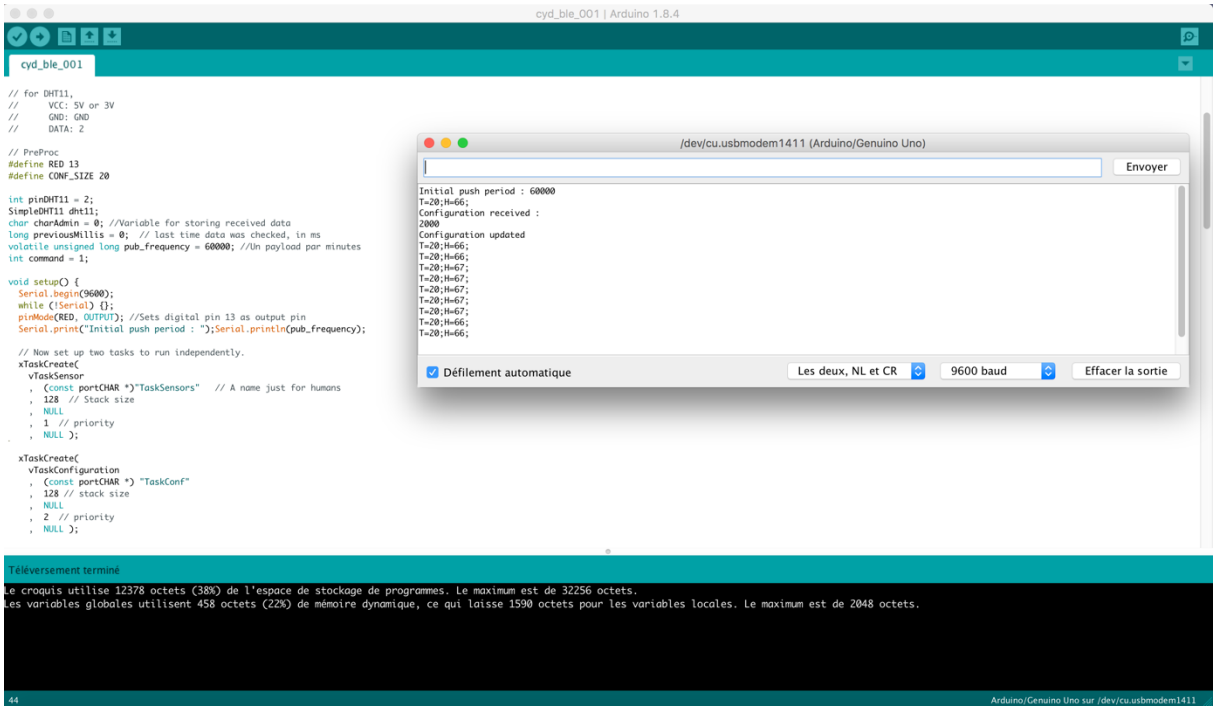


Figure 5-Reconfiguration de la période des relevées des indicateurs

4.2.2 Passerelle ou local Gateway

La passerelle est un Raspberry Pi3 model B ayant toute les connectiques necessaire. Le système d'exploitation installé est un linux Debian.

Figure 6-Raspberry Pi3 utilisé comme une passerelle multi-capteurs

La passerelle héberge un broker MQTT et un mini server python. Elle permet de gérer les messages de multiples capteurs. La librairie « pygatt » est utilisée pour communiquer avec le module BLE : [Lien github]

```
def notifyBle(handle, value):
    print("START1")
    globalPayLoad = str(value)
    tabPayload = globalPayLoad.split(';')
    tempPayload = "temperature,host=192.168.1.3,deviceId=00:15:83:10:FD:8D "+tabPayload[0]
    humPayload = "humidity,host=192.168.1.3,deviceId=00:15:83:10:FD:8D "+tabPayload[1]
    print(tempPayload);
    print(humPayload);
    pubMqqtLinkSession("0000FFE1-0000-1000-8000-00805F9B34FB", "192.168.1.3", tempPayload, humPayload)
    print("END1")

adapter = pygatt.GATTToolBackend()

adapter.start()
device = adapter.connect('00:15:83:10:FD:8D')
device.subscribe("0000FFE1-0000-1000-8000-00805F9B34FB", callback=notifyBle)
```

Figure 7-Partie du code du mini server python hebergé par la passerelle

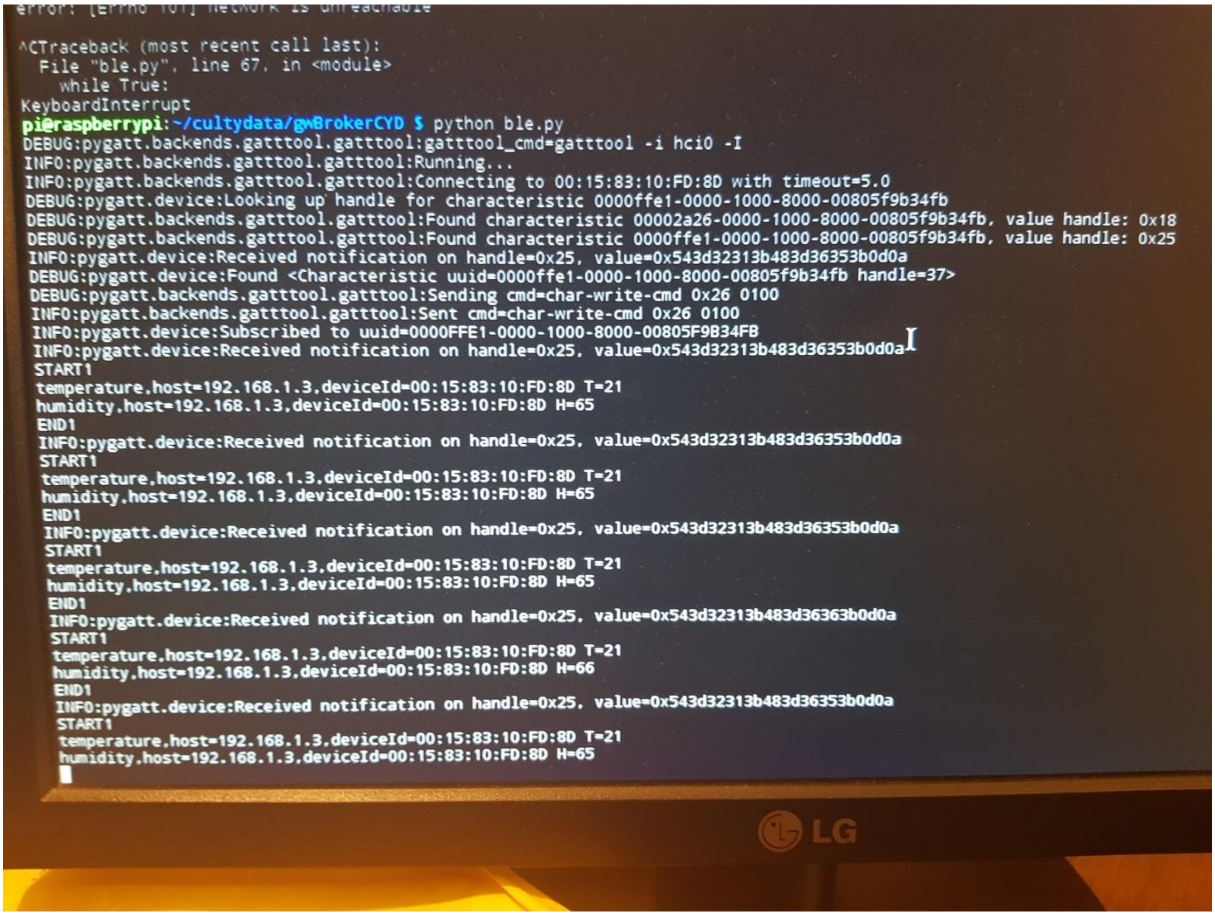


Figure 8-Log du mini-server python traçant les indicateurs envoyés par le module BLE

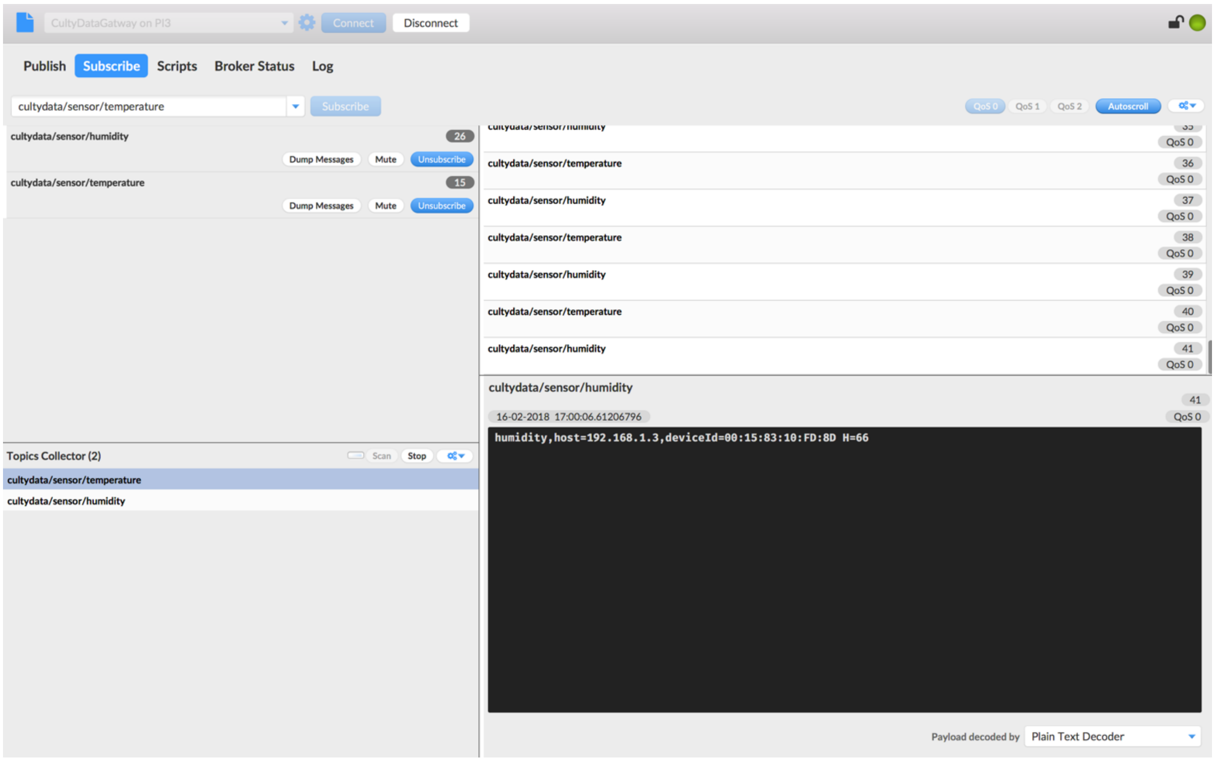


Figure 9-Vérification du bon fonctionnement des topics MQTT

4.2.3 Reverse proxy

Le « reverse proxy » est un serveur web « tomcat » embarqué dans un jar « **cydbridge** » réalisé à l'aide de la technologie JAVA Spring. L'ensemble des micro services JAVA utilisés par le système sont embarqués dans ce composant quitte à les sortir si nécessaire.

Le « POC » contient un bridge MQTT vers Kafka permettant la consommation d'un évènement MQTT et la production d'un « Payload » enrichi sur des topics Kafka : [Lien github]

```
2018-02-16 16:07:12.868  INFO 25921 --- [           main] o.a.kafka.common.utils.AppInfoParser     : Kafka version : 0.10.2.0
2018-02-16 16:07:12.868  INFO 25921 --- [           main] o.a.kafka.common.utils.AppInfoParser     : Kafka commitId : 576d93a8dc0cf421
2018-02-16 16:07:12.872  INFO 25921 --- [           main] o.s.i.endpoint.EventDrivenConsumer       : Adding {logging-channel-adapter:_org.springframework.integration.errorLogger} as a
subscriber to the 'errorChannel' channel
2018-02-16 16:07:12.872  INFO 25921 --- [           main] o.s.i.channel.PublishSubscribeChannel    : Channel 'application.errorChannel' has 1 subscriber(s).
2018-02-16 16:07:12.872  INFO 25921 --- [           main] o.s.i.endpoint.EventDrivenConsumer       : started _org.springframework.integration.errorLogger
2018-02-16 16:07:12.872  INFO 25921 --- [           main] o.s.c.support.DefaultLifecycleProcessor  : Starting beans in phase 1073741823
2018-02-16 16:07:12.949  INFO 25921 --- [           main] w.i.MqttPahoMessageDrivenChannelAdapter  : started inbound
2018-02-16 16:07:12.966  INFO 25921 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator  : Discovered coordinator 192.168.1.4:9092 (id: 2147483647 rack: null) for group test-
consumer-group.
2018-02-16 16:07:12.970  INFO 25921 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator  : Revoking previously assigned partitions [] for group test-consumer-group
2018-02-16 16:07:12.971  INFO 25921 --- [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer    : partitions revoked: []
2018-02-16 16:07:12.971  INFO 25921 --- [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator  : (Re-)joining group test-consumer-group
2018-02-16 16:07:13.018  INFO 25921 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
2018-02-16 16:07:13.026  INFO 25921 --- [           main] com.cultydata.CydbridgeApplication       : Started CydbridgeApplication in 15.154 seconds (JVM running for 18.123)
```

Figure 10-Bridge MQTT vers KAFKA en zone démilitarisée

4.2.4 Middleware haute performance

Le bus permettant la gestion des évènements provenant de multiple exploitation est un broker Kafka. La haute disponibilité est gérée par un superviseur « Zookeeper ».

Le message poussé dans le topic Kafka a pour format (« Line Protocole ») :

temperature,host=192.168.1.3,deviceId=00:15:83:10:FD:8D T=21 1518797405265000000

```
temperature,host=192.168.1.3,deviceId=00:15:83:10:FD:8D T=21 1518797405265000000
2018-02-16 16:10:05.270  INFO 25921 --- [ntainer#0-0-C-1] okhttp3.OkHttpClient                     : --> POST
http://192.168.99.100:8086/write?u=admin&p=admin&db=CULTYDATA&rp=autogen&precision=n&consistency=one http/1.1 (80-byte body)
```

Figure 11-Payload dans les topics KAFKA

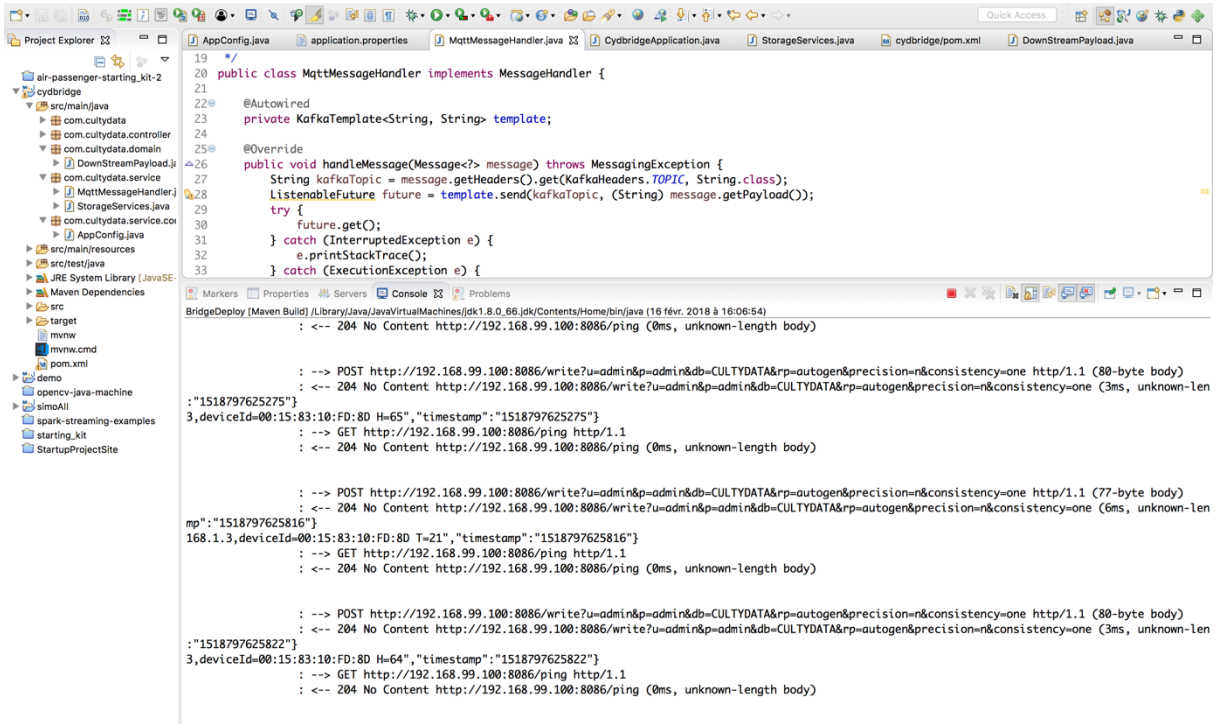


Figure 12-Code des handlers MQTT vers KAFKA

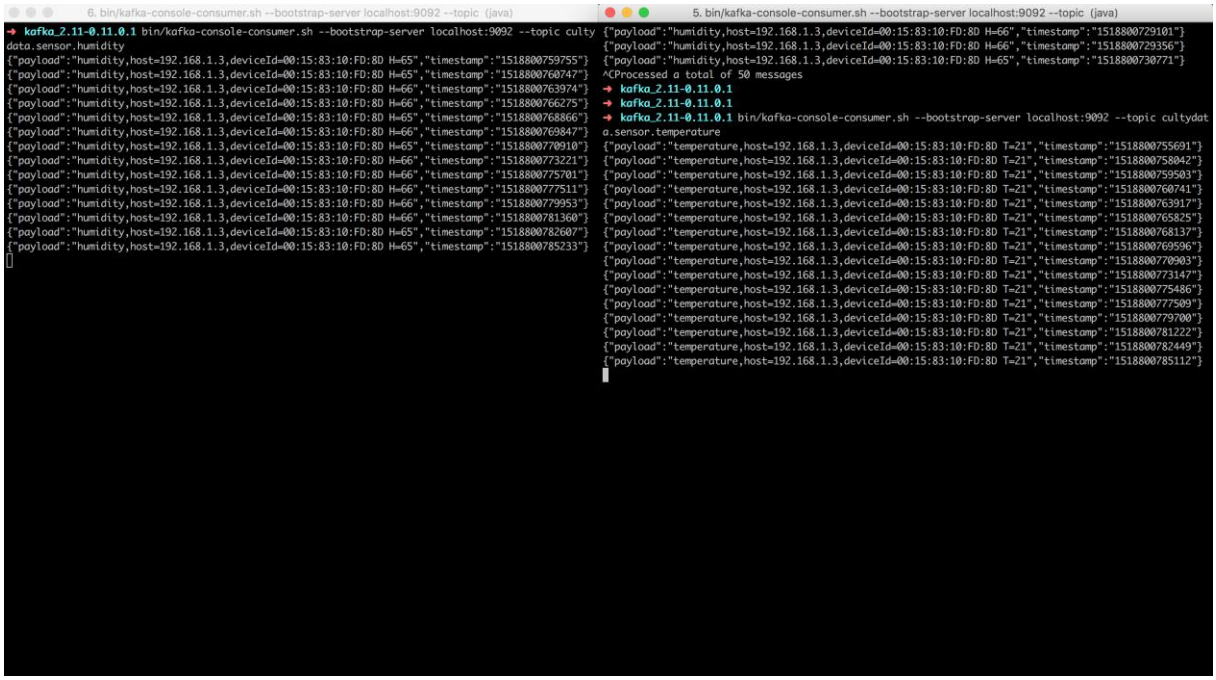


Figure 13-Events contenus dans les topics KAFKA

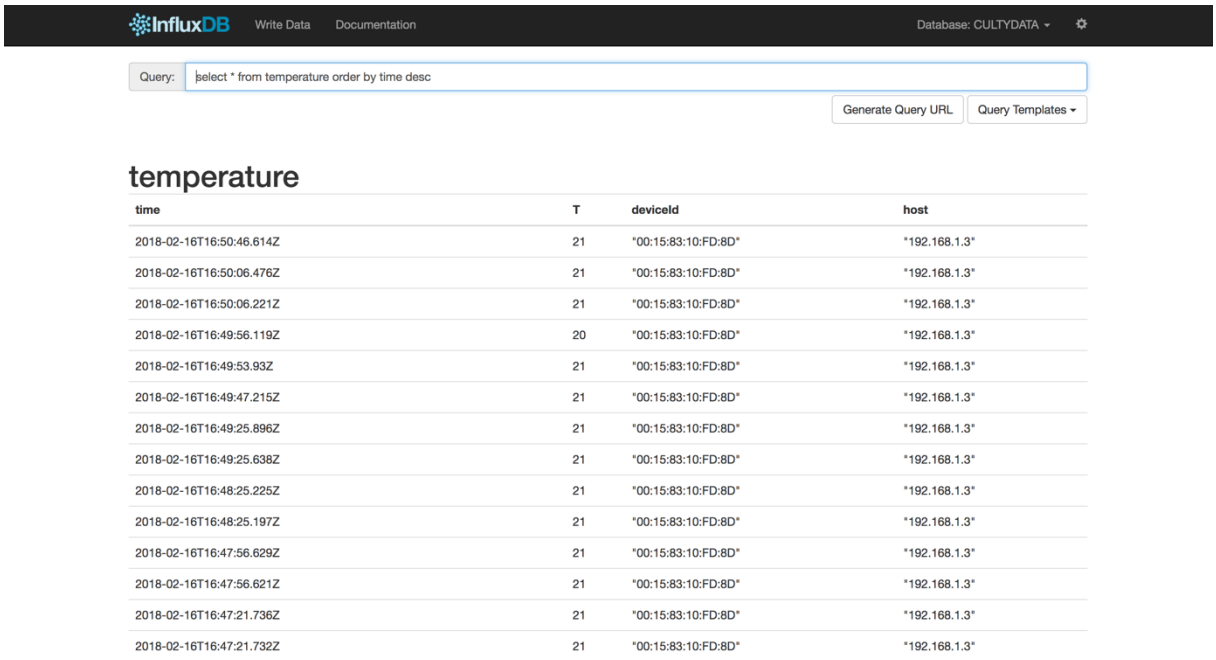
4.2.5 Base de données de série temporelle

Les évènements Kafka suivent un format « Line Protocole » permettant une intégration directe avec la base de données de séries temporelles InfluxDB : [Lien github]

Un micro service permet de consommer un évènement du topic Kafka pour l'enregistrer dans une instance de la base de données InfluxDB.

```
package com.cultydata.service;

import org.influxdb.InfluxDB;
import org.influxdb.dto.Pong;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.kafka.support.KafkaHeaders;
import org.springframework.messaging.handler.annotation.Header;
import org.springframework.stereotype.Service;

import com.cultydata.domain.DownStreamPayload;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

@Service
public class StorageServices {

    @Autowired
    private InfluxDB influxDB;

    final GsonBuilder builder = new GsonBuilder();
    final Gson gson = builder.create();

    @KafkaListener(topics = {"cultydata.sensor.temperature" , "cultydata.sensor.humidity"})
    public void onReceiving(String message, @Header(KafkaHeaders.OFFSET) Integer offset,
                            @Header(KafkaHeaders.RECEIVED_PARTITION_ID) int partition,
                            @Header(KafkaHeaders.RECEIVED_TOPIC) String topic) {

        System.out.println("From KFKA : " + topic + "/" + partition + "/" + offset + ":" +  message);

        Pong response = this.influxDB.ping();
        if (response.getVersion().equalsIgnoreCase("unknown")) {
            System.out.println("Error pinging server.");
        }
        else {
            System.out.println(response.getVersion());
        }

        DownStreamPayload dsp = gson.fromJson(message, DownStreamPayload.class);
        System.out.println(dsp.toLineProtocol());

        influxDB.write(dsp.toLineProtocol());

    }

}
```

Figure 14-Stockage dans une base de données de séries temporelles



temperature

time	T	deviceId	host
2018-02-16T16:50:46.614Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:50:06.476Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:50:06.221Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:49:56.119Z	20	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:49:53.93Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:49:47.215Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:49:25.896Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:49:25.638Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:48:25.225Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:48:25.197Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:47:56.629Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:47:56.621Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:47:21.736Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"
2018-02-16T16:47:21.732Z	21	"00:15:83:10:FD:8D"	"192.168.1.3"

Figure 15-Contenu de la base CULTYDATA dans InfluxDB

4.2.6 Outils de visualisation

L'outils Grafana permet la visualisation des séries temporelles.

Figure 16-Visualisation des séries temporelles via Grafana

4.2.7 Infrastructure

Les composants du « POC » ont été déployés sous différentes formes.
En VM linux Virtual Box pour Grafana, en Docker pour InfluxDB, en machine physique windows ou Mac pour le broker Kafka et les micro services. Le choix reste ouvert et sera étudié au moment venu.

Il serait judicieux par la suite, pour des raisons de sécurité, de séparer les micro services dans des composants indépendants afin d'effectuer une isolation réseau entre les différentes parties :

- La partie exposée à internet (la zone démilitarisée constituée du bridge MQTT vers KAFKA)
- La partie « LAN » hébergeant le middleware
- Les nœuds de la blockchain, les bases de données et les micro-services non frontaux

Un écosystème de gestion des déploiements multi-cloud hybride tel que Rancher/Jenkins/Ansible peut être utile à l'intégration continue ainsi qu'à l'industrialisation des mises à jour des « softs ».

4.3 Problématiques à gérer pour finaliser un « MVP »

Les points suivants n'ont pas été mis en pratique lors du « POC ». Voici cependant quelques éléments permettant de préparer la réalisation d'un « MVP » à partir du « POC ».

4.3.1 Blockchain

Afin de gérer la certification d'une manière sécurisée, la dynamique permettant la délivrance d'une certification après la récolte sera consignée dans une blockchain privée sous forme de « Smart Contract » réagissant à des événements liés à l'évolution de l'unité de production.

Une blockchain Etherium permettra la mise en pratique de ce principe [4] [5].

Pour ce faire, un micro service traitera les « events » Kafka afin de faire vivre les contrats stockés dans la blockchain Etherium.

4.3.2 Sécurité

La sécurisation des différents flux de données ainsi que la gestion des autorisations dans la plateforme sont des sujets cruciaux.

La communication entre les capteurs et la passerelle se fait en BLE. Une implémentation TLS au-dessus du BLE devra être envisagée. Je prévois d'étudier cette possibilité en analysant les API TLS embarquées tel que « mbed TLS » [6]. Une solution hybride utilisant cette technologie pour le partage du secret peut être complétée par un chiffrement symétrique fait maison pour le « payload ». Le partage du secret devra être envisagé au premier enregistrement du capteur et revu par une stratégie de mise à jour configurable.

La communication avec le broker ne pose pas de problème car en effet, la technologie MQTT permet une communication TLS nativement.

Le frontal coté serveur peut fournir aussi toutes les API nécessaires pour une transmission sécurisée via KAFKA pour InfluxDB, Grafana, quitte parfois à rajouter un « reverse proxy » TLS/SSL pour notamment InfluxDB et Grafana.

Etherium est quant à lui nativement sécurisé.

4.3.3 Fiabilité et robustesse

La chaine d'évènements permettant la certification peut être modélisée à l'aide d'un outil de « model checking » tel que « Uppaal » [7] afin d'assurer un certain niveau de fiabilité. Une intégration du processus de vérification avec une chaine d'intégration continue est envisageable.

Le code fourni devra être qualifié via des tests unitaires intégrés également à la chaine d'intégration continue.

Les différents déploiements suivront une stratégie de redondance multi-site afin d'assurer une haute disponibilité des différents services. L'utilisation de « reverse proxy », de « service discovery » et d'outils tel que « Zookeeper » assurent un certain niveau de robustesse.

4.3.4 Gestion de l'énergie

Dans cette section, je présenterai une méthodologie pour estimer les besoins énergétiques au niveau des capteurs.

Quant à la passerelle constituée d'un Pi 3 modèle B, le site raspberrypi.org préconise une alimentation d'au moins 700mA. Dans la pratique, j'ai constaté qu'une alimentation délivrant 1A (soit 1000mA) permet de couvrir tous les besoins, y compris avec deux périphériques USB énergivores. Je suis contraint de le brancher sur une alimentation comprise entre 4,5V et 5,7V via un chargeur sur secteur. L'installation de l'exploitation devrait le permettre.

Concernant les capteurs, selon les cas d'utilisation, la méthodologie d'estimation est la suivante (en considérant une consommation maximale d'un module BLE à 500mW) :
On a donc un besoin de puissance de P = 0,5 J/s.
En 1 seconde on a donc dépensé une énergie de : E = 0, 5 J.
Or une application accède à un débit de : D = 270 Kbit/s.
Donc en 1 seconde, une application émet 270 kbits.
L'énergie par bit est donc de :
E(bit) =0,5/ (270*1000) = 1,85 µJ / bit

En considérant des caractères codés sur un octet et des « payloads » en amont de la forme :
T=19;H=65…de 5 caractères par indicateurs et les indicateurs suivants :

- Température : T
- Humidité : H
- Luminosité : L
- PH acidité du sol : PH

J'obtiens 20 caractères, soit un « payload » de 160 bit i.e environ une énergie de transmission de 300 J par payload.

Une pile alcaline avec un ampérage de 500 mAh et une tension nominale de 9,6 volt, donne théoriquement une puissance de 0,5*9,6 W en une heure, soit 4,8 Wh ou 8640J.

Ce qui permettrait d'alimenter le capteur pour environ 30 payloads, soit une autonomie de moins de 3 jours si on effectue des relevés toutes les 2 heures. Lors du « POC », la pile fournit dans le kit Arduino n'a pas tenu une nuit vue des rafales testées à quelques minutes.

Par ailleurs, il faudrait prendre en compte l'ampérage nécessaire pour les relevés. Le DHT11 a besoin de 2,5 mA par relevé.

Une batterie lithium à 1200mAh permettrait de doubler l'autonomie iso fréquence. Il existe aussi des méthodes d'optimisation permettant de réduire le temps d'activité du module BLE en le désactivant selon une stratégie donnée. Cependant, ces stratégies ne sont pas applicables pour le module Xbee.

Dans tous les cas, lors de la phase d'industrialisation, il sera nécessaire de revoir le choix du microcontrôleur et des composants éléctroniques. L'Arduino n'est pertinent que lors de la phase de prototypage. Les choix des composants électroniques seront nécessairement guidés par la problématique énergétique pour atteindre une autonomie de quelques mois ou années.

Dans cette estimation, seuls les indicateurs de type valeur ont été pris en compte. Pour des éventuelles images de l'unité de production, une analyse spécifique doit être effectuée.

4.3.5 Dimensionnement

Le dimensionnement de la plateforme sera effectué en tenant compte des fréquences d'envoi du « Payload », de sa taille et de la politique de rétention envisagée. Les choix d'architecture et des composants de la plateforme ont été fait en gardant en tête la « scalabilité » de la solution. Le choix de KAFKA parmi d'autres n'est pas neutre. En effet il ouvre une possibilité de passage à l'échelle certaine. Les différents micro services étant « Stateless » ne poseront pas de soucis de montée en charge.

Les politiques de rétention des séries temporelles seront définies en tenant compte du cycle de vie d'une unité de production quitte à faire des backups accessibles hors instance de production d'un certain historique et permettre des prédictions et de l'analyse statistique si le besoin se présente.

Il faudrait garder à l'esprit, par ailleurs, qu'une certification collaborative demande une profondeur d'historique pertinente pour pouvoir ajuster les contrats. La blockchain Etherium permet ceci nativement mais demande un investissement en matériel. On sera donc confronté à ce type de problématique.

4.3.6 Autres considérations

La mise en place de capteurs dans un environnement naturel contraint nécessite également d'étudier les problématiques liées à l'étanchéité. Lors de la phase de construction d'un prototype industriel, il faudra concevoir un boîtier résistant aux conditions météorologiques.
L'aide d'un « fablab » lors de cette phase sera nécessaire.

5 Conclusion et élargissement

Lors de ce projet, j'ai tenté de couvrir les différentes problématiques permettant la mise en place d'une plateforme intelligente en environnement contraint permettant une certification collaborative d'une production agricole de plantes aromatiques. L'élargissement de ce concept à une production agricole autre est possible en tenant compte des spécificités de l'unité de production que l'on veut certifier. Certains points, tel que les protocoles utilisés, par exemple, sont évidemment à revoir pour adapter la plateforme.

Par ailleurs, le modèle de certification devra tenir compte de toute la chaine de production afin de fournir un modèle robuste permettant de concurrencer les certifications délivrées par les organismes actuels. La réalisation de ce projet nécessitera des investissements humains conséquents. La startup CULTYDATA a l'ambition de poursuivre ce travail durant l'année 2018 tout en cherchant des partenaires ainsi que du financement.

6 Annexes

6.1 Luminosité [8]

Activité ou lieu concerné	Éclairement moyen
Sensibilité d'une caméra de bas niveau	0,001 lux
Nuit de pleine lune	0,5 lux
Rue de nuit bien éclairée	20 à 70 lux
Local de vie	100 à 200 lux
Appartement bien éclairé	200 à 400 lux
Local de travail	200 à 3 000 lux
Stade de nuit (suivant les différentes catégories (E1,E2,E3,E4,E5))	150 à 1 500 lux
Extérieur par ciel couvert	500 à 25 000 lux
Extérieur en plein soleil	50 000 à 100 000 lux

7 Bibliographie

[1] «https://www.fellah-trade.com,» [En ligne].
[2] «http://www.rustica.fr,» [En ligne].
[3] «https://github.com/malaoui79/CultYDATA,» [En ligne].
[4] «https://bitsonblocks.net/2016/10/02/a-gentle-introduction-to-ethereum/,» [En ligne].
[5] «https://dzone.com/refcardz/getting-started-with-etherium-private-blockchain,» [En ligne].
[6] «https://tls.mbed.org/discussions/generic/mbetls-with-bluetooth,» [En ligne].
[7] «http://www.uppaal.org,» [En ligne].
[8] «http://www.manuel-esteban.com/arduino-capteur-de-luminosite/,» [En ligne].
[9] «https://github.com/malaoui79/CultYDATA,» [En ligne].