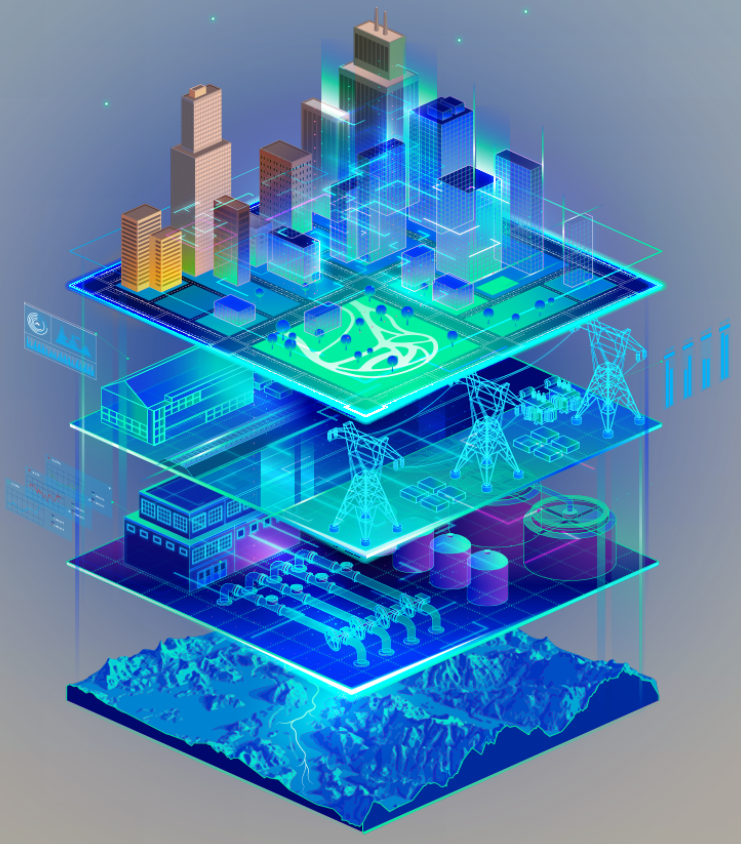
Université de Paris

Rapport de projet

JUMEAU NUMÉRIQUE DANS L'INDUSTRIE DU FUTUR

L2G1-2021



Rapport de projet

PROJET JUMEAU NUMÉRIQUE

Identification du document

Référence du document : L2G1.R
Version du document : 1.1
Date du document : 14/05/2022
Auteurs : Wattrelos Tigran Boka Ricardo Abou Assaf Mahyr-Florian Daumont-Ouk Ilan'

Université de Paris

Sommaire

REMERCIEMENTS

Nous tenons tout d’abord à remercier Mr Nassime El Fartass pour nous avoir encadré tout au long de notre projet, pour les aides à nos problèmes et les explications apportées. Ainsi qu’à l’UFR Paris Descartes et à son responsable Mr David Janiszek, pour nous avoir donné l'opportunité d’étendre nos compétences en informatique (Grafana, InfluxDB, Machine Virtuelle).

Grâce à ce module, nous avons pu apprendre à travailler en groupe, améliorer notre communication afin de se répartir les tâches et donc à avancer plus vite et de manière efficace.

Contexte

Ce projet est réalisé dans le cadre de l'UE "Projet de programmation" de l'UFR de mathématiques-informatique de la faculté Université Paris Cité. Quatres étudiants sont sous la tutelle d'un encadrant auquel ils doivent rendre compte de l'avancement de leur projet chaque semaine. Ils doivent également rendre différents documents afin de faciliter la compréhension du projet.
Lors d'une soutenance orale, les étudiants doivent présenter à la fin des 12 semaines de préparation le projet devant un jury.

Introduction

C'est en 1969 lors du lancement d'Apollo 11, que le premier digital twin est programmé. Il permettait de visualiser de manière informatique les données récoltées de la fusée, d'être capable de la contrôler à distance, voire même d'anticiper ses futurs déplacements ou pannes grâce aux données. Récemment, les plus grosses entreprises se sont intéressées au Digital Twin, afin de mieux prévoir les changements ou erreurs, et donc d'avoir un moindre impact sur le coût d'un objet produit en masse.

De nombreuses zones géographiques sont concernées par le digital twin comme l'Europe, l'Asie, les Etats-Unis..
Les industries automobiles, médicales, ou de l'électrique sont par exemple sollicitées à l'utiliser, dans le but d'améliorer les processus, de coordonner différents corps de métier, ce qui permet d'avoir confiance en l'équipement industriel. Tout ceci vise à augmenter le chiffre d'affaires, par la prévention et la gestion de pannes potentielles.
Les sociétés utilisant et proposant des digital twins sont Microsoft, Siemens, IBM, Dell, etc.

Objectif

L'objectif de ce projet est de créer le jumeau numérique d'un objet connecté qui collecte des données. Nous utilisons comme objet connecté la carte mère Raspberry Pi Zero, dotée de 3 capteurs : pression / humidité / température.
L'objectif est de récolter les données perçues par ces capteurs, de les envoyer via un broker MQTT vers un serveur pour alimenter une base de données temporelles. Cette dernière enverra ses valeurs au tableau de bord lui-même connecté au serveur.

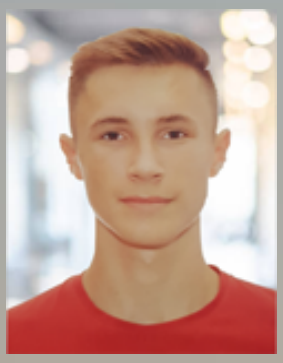
Gestion du projet

La gestion de projet permet de planifier au mieux le bon déroulement de ce dernier, de la manière la plus efficace possible.

1. L'équipe

Pour ce projet encadré par Nassime EL FARTASS architecte informatique chez *ATOS*, l'équipe L2G1 est composée de 4 étudiants en 2ème année de Licence informatique à l'Université Paris Cité:

Tigran WATTRELOS



Ricardo BOKA

Mahyr-Florian ABOU ASSAF

Ilan' DAUMONT-OUK

2. La planification

Le projet Digital Twin s'étend sur une durée de 12 semaines, incluant des réunions hebdomadaires avec notre encadrant pour structurer notre avancement, mais aussi le rendu régulier de documents, tels que le cahier des charges, le cahier de recettes, l'étude de marché, le plan de tests, l'architecture logicielle et des manuels pour le client.

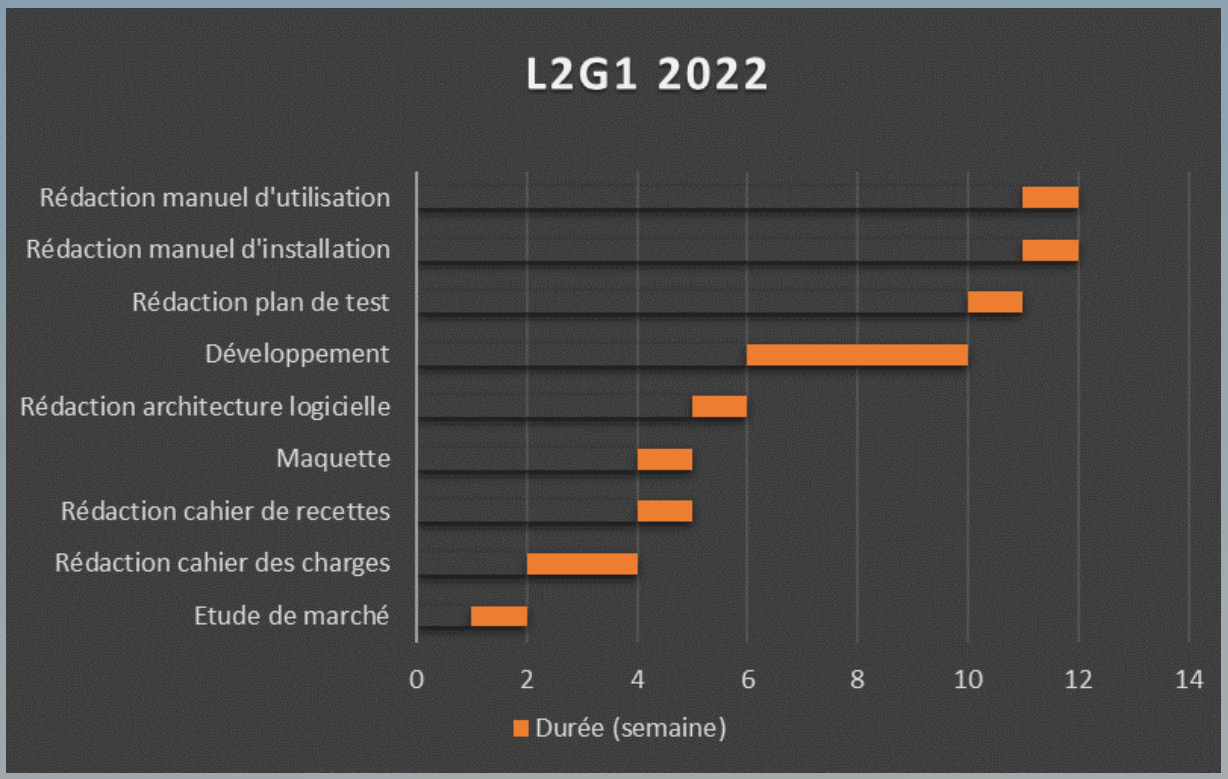


Diagramme de Gantt de la planification du projet

Ce diagramme de Gantt présente les différents documents rendus ainsi que le temps passé à les rédiger. Par la suite, nous vous décrivons les documents que nous avons écrits.

Nous avons passé une bonne partie du temps à faire des recherches liées à cette technologie et à prendre des repères solides sur le marché, ce que cela représente dans le monde. S'en est suivi la rédaction des différentes idées sur notre projet, c'est-à-dire les différentes fonctionnalités que nous trouvons pertinentes après discussion avec l'encadrant, mais également de mettre en œuvre les différents tests qui feront appel aux principaux concepts utilisés. L'architecture logicielle, un

document clé pour le passage de la phase de conceptualisation du projet à l'implémentation de nos idées, celui-ci, comme tous les autres documents, nous ont permis de simplement et rapidement traduire nos concepts en code fonctionnel. Finalement, nous avons proposé des manuels qui contiennent pas à pas les étapes à suivre pour l'installation et à l'utilisation du Digital Twin.

Documents réalisés :

Étude de marché (semaine 1) : permet de mieux connaître sur quels sujets se base le digital twin, de cibler les offres concurrentes et d'évaluer la demande actuelle sur le marché mondial.

Cahier des charges (semaine 2) : permet d'avoir une vue globale sur la façon dont le projet va être mené, avec la description de la demande, les contraintes et l'établissement de notre travail en tant que programmeurs.

Cahier de recettes (semaine 4) : permet de lister les scénarios à vérifier tout au long du projet et lors de la soutenance.

Maquette (semaine 4) : première visualisation du rendu de notre tableau de bord.

L'architecture logicielle (semaine 5) : décrit l'organisation du code et de l'implémentation entre chaque composant du digital twin, que le groupe doit développer.

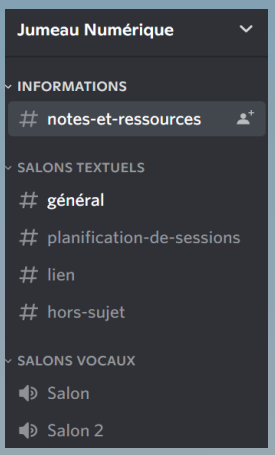
Plan de test (semaine 10) : destiné uniquement aux programmeurs, ce document permet de s'assurer que l'ensemble des composants créés, fonctionnent indépendamment, mais aussi entre eux. Il détaille les objectifs, les ressources et les processus d'un test.

Manuel d'installation (semaine 11) : décrit les étapes de déploiement du digital twin pour le mettre en place sur le serveur, la base de données et le dashboard.

Manuel d'utilisation (semaine 11) : décrit l'ensemble des fonctionnalités du digital twin, et des actions à réaliser pour l'utiliser.

3. Outil de gestion

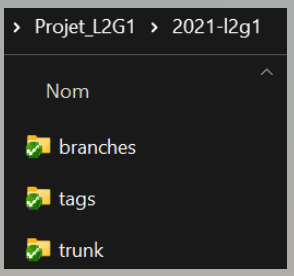
Les principaux outils de gestion de l'organisation, de l'environnement de travail ainsi que l'espace de partage des documents sont les suivants :



Discord : Logiciel de call et de partage des ressources entre étudiants, cette application permet de créer différents salons de discussions en fonction des ressources partagées. Nous avons créé deux salons de discussions vocaux pour la répartition des tâches en groupe de 2 lors de la phase de développement, et 5 salons de discussion textuels.

Forge : Environnement de partage et communication avec les enseignants. Tous les documents tels que les plans, les cahiers et les manuels ont pu être envoyés sur la Forge, de plus les chefs de projet avaient pour consigne de communiquer sur le forum proposé, notamment les comptes-rendus de chaque séance.

SVN : Environnement de partage de la partie informatique, les scripts python, java et les paramètres du dashboard ont pu être communiqués dans un dossier "branches" lieu où le code est envoyé par groupe (selon les groupes et leurs tâches) puis un dossier "trunk" où nous avons pu envoyer la finalité des dossiers avec les scripts bien ordonnancés et le projet fonctionnel.



Oracle VM VirtualBox : Environnement de travail virtuel, toutes les installations, les paramétrages et développements des scripts ont pu être générés sur un système d'exploitation Linux.

4. Organisation interne

En-dehors des réunions avec l'encadrant, l'équipe se réunissait chaque lendemain de réunion pour rédiger le compte-rendu et commencer à se répartir les tâches pour la semaine suivante. De plus, nous nous réunissions deux ou trois fois par semaine entre nous afin de rendre compte de l'avancement de chaque tâche et des possibles difficultés rencontrées.

5. Chef de projet

Toutes les deux semaines, un étudiant est désigné comme chef de projet qui se voit attribuer le rôle d'organisateur au sein de l'équipe et d'envoyer le compte-rendu sur la forge. Celui-ci avait également pour devoir de gérer les communications avec l'encadrant. En fin de projet, cette responsabilité basculait à une semaine par personne afin d'atteindre une répartition équitable des responsabilités sur 12 semaines.

6. Environnement de développement et de production

Pour le développement de notre projet, nous avons utilisé une machine virtuelle Ubuntu 64 et une machine virtuelle Raspbian. Ceci nous a permis de tous travailler sur le même environnement. Nous avons bien entendu testé ce que nous avons développé directement sur le Raspberry. Le projet a pour objectif d'être installé sur une machine linux pour le côté serveur et sur un raspberry pi zero équipé d'un sense hat B.

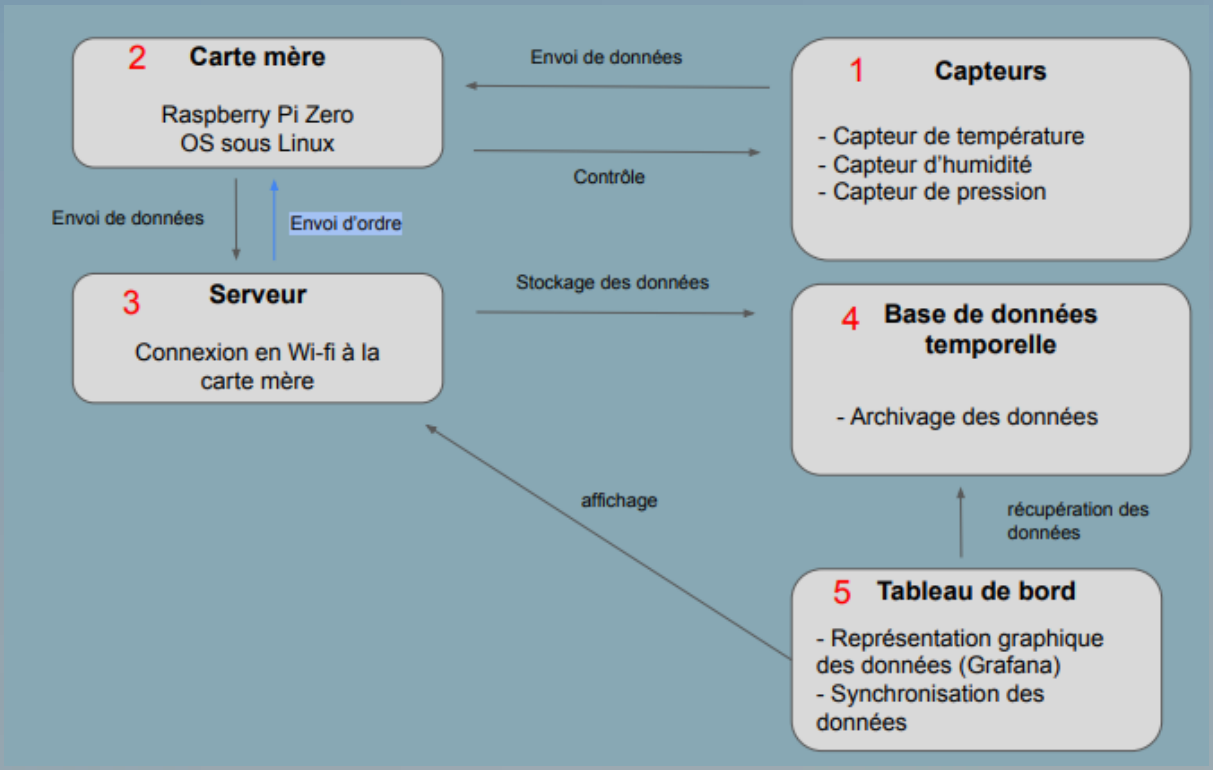
Développement

1. Lexique

- **Python** : Langage de programmation interprété, favorisant la programmation impérative structurée, fonctionnelle et orientée objet.
- **Java** : Langage de programmation orienté objet.
- **InfluxDB** : Base de données orientée séries temporelles, permettant de stocker des données sur la durée.
- **Grafana** : Logiciel libre qui permet de visualiser des données depuis une base de données temporelles par exemple.
- **MQTT** : Protocole open source de messagerie assurant des communications entre des appareils.
- **Alerting** : Notification sur des changements importants des valeurs du tableau de bord
- **NTP** : Service permettant de récupérer l'heure actuelle grâce au réseau internet
- **Dashboard** : équivalent de "tableau de bord"
- **Digital Twin** : équivalent de "jumeau numérique"

2. Architecture logicielle

2.1 Explication de l'architecture logicielle



On peut séparer cette architecture en deux grandes catégories : le côté Raspberry et le côté machine client. Du côté Raspberry, les capteurs (1) envoient des données de pression, de température et d'humidité vers la carte mère (2). Via une connexion internet, celle-ci envoie les différentes données, sa position et l'heure d'envoi des métriques au côté machine client, plus exactement au serveur (3). Ensuite, ces données sont archivées dans une base de données (4).

Finalement, le tableau de bord va chercher les valeurs de température, de pression et d'humidité dans la base de données et va les afficher sur des graphiques et des jauges.

2.2 Application de l’architecture logicielle

Du côté du Raspberry Pi Zero, avec une machine virtuelle sous Linux, Raspbian/Debian est utilisé comme système d’exploitation car il est optimisé pour le matériel de la carte mère.

Le code python est installé sur le raspberry et permet la récupération des données des trois capteurs, de température de pression et d’humidité. Il permet aussi au raspberry d’établir une connexion sur un broker MQTT pour transmettre les données au serveur accompagné de l’heure de récupération et de sa position exacte. Cette heure est récupérée grâce au protocole NTP. La position du raspberry est obtenue à l’aide de l’adresse ip de celui-ci.

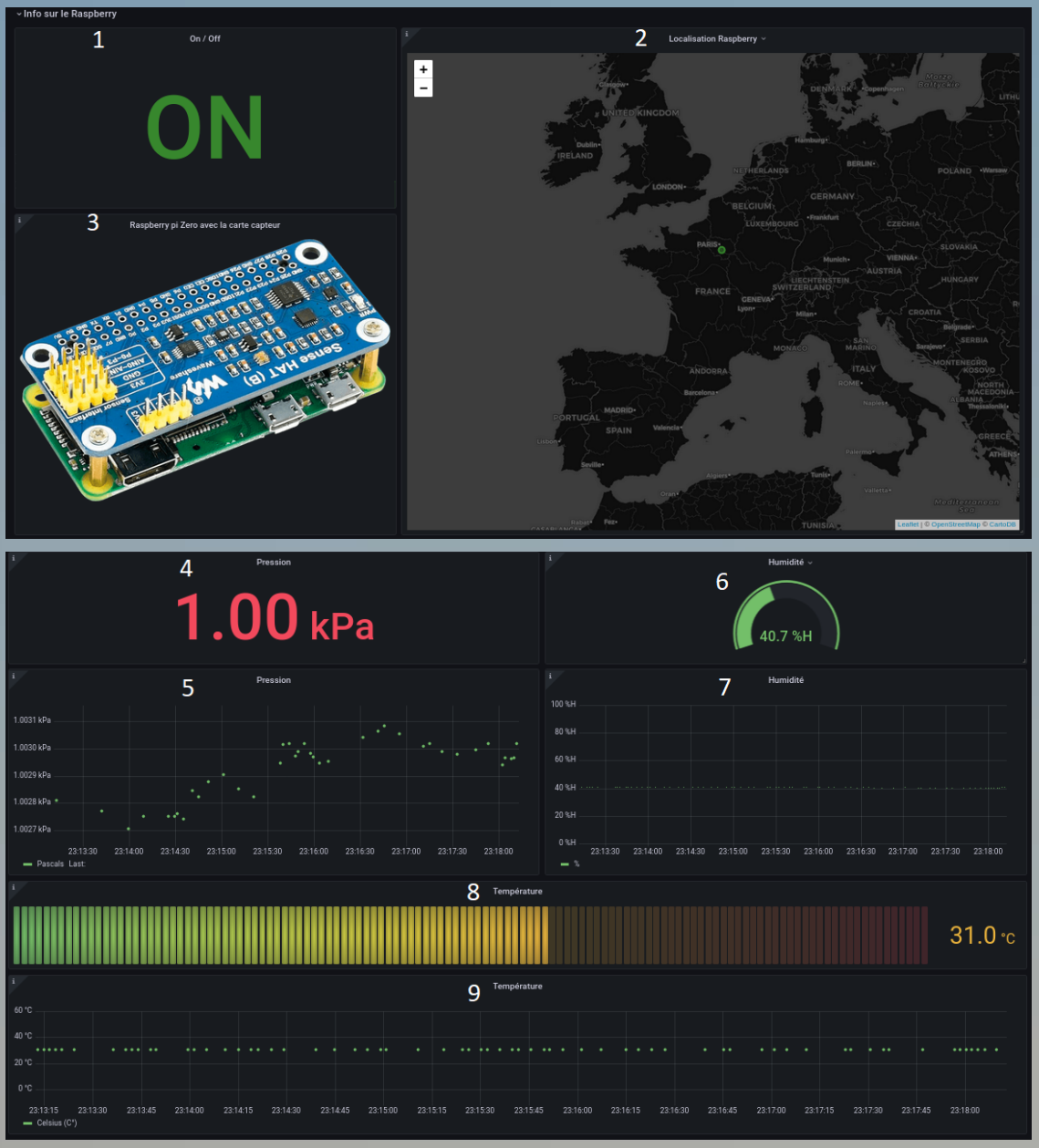
Le code Java quant à lui, se charge de récupérer les valeurs via le MQTT, et de les envoyer à InfluxDB qui les stockera, en se connectant au port 8086.

Nos données, une fois envoyées vers InfluxDB, seront transmises à Grafana, afin de les visualiser sur notre dashboard.

Grafana est un dashboard qui récupère les données d’InfluxDB, et affiche les composants suivants : la localisation de la carte mère sur la carte du monde ; le voyant On/Off indiquant si la carte mère est allumée ou éteinte, avec un délai de réponse de seulement quelques secondes ; les valeurs des capteurs de pression, d’humidité et de température sous forme de graphe pour les anciennes valeurs, mais aussi une jauge montrant ce que capte actuellement chaque capteur.

3. Affichage pour l’utilisateur

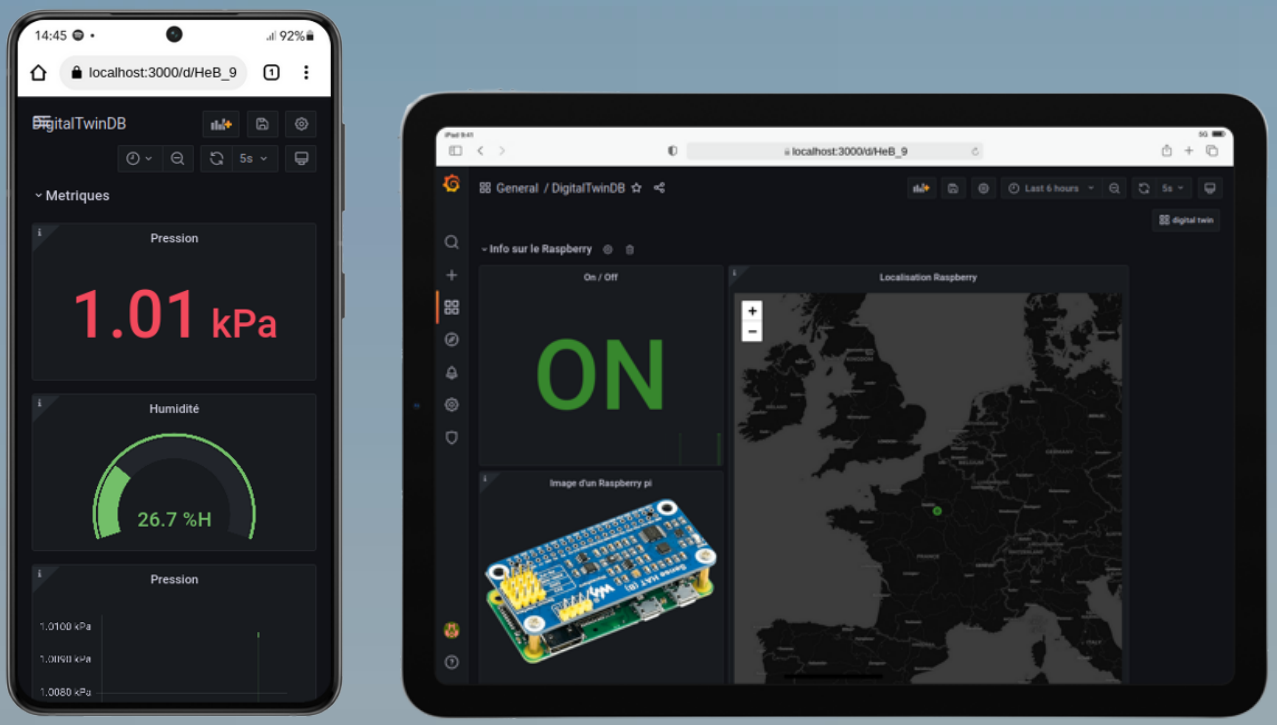
Le tableau de bord se présente comme ci-dessous :



Légende :

1 - Indique l’état du raspberry pi
2 - Carte indiquant la localisation du Raspberry
3 - Image d’un Raspberry Pi Zero accompagné d’un Sense hat B
4 - Pression en temps réel
5 - Histogramme de la pression dans le temps
6 - Humidité en temps réel
7 - Histogramme de la température dans le temps
8 - Température en temps réel
9 - Histogramme de la température dans le temps

4. Web responsive



Le site Grafana est capable de s'adapter à l'appareil sur lequel on l'utilise, ce qui le rend beaucoup plus ergonomique et donc assure une meilleure visibilité des données : il est web responsive. Il s'adapte aussi bien sur téléphone que sur tablette peu importe le système d'exploitation. Le tableau de bord peut donc être consulté sur PC comme sur téléphone.

5. Les difficultés rencontrées et les solutions

L'un des premiers obstacles rencontrés a été la connexion au capteur Sense Hat, nous avions utilisé des librairies compatibles avec Sense Hat A et non avec Sense Hat B. Pour récupérer les données des capteurs nous avons changé de piste et avons opté pour installer d'autres paquets compatibles avec Sense Hat B.

Lors de la configuration de la connexion au broker de messages MQTT, nous avons dû installer plusieurs paquets côté serveur pour recevoir les données relevées par le raspberry. Nous avons dû avec beaucoup de précautions choisir les paquets avec un risque de défaut de version, de classes manquantes dans le paquet, de modification du paquet d'origine ou encore d'incompatibilités se présentant entre notre code et les imports de paquets dans notre fichier java. Après avoir choisi correctement les paquets java à importer dans notre code, des conflits pouvaient encore avoir lieu au niveau des répertoires dossier pour l'import des paquets au sein du code, il a fallu réorganiser l'emplacement des paquets au sein de répertoires qui leur sont communs pour éviter ce type de conflit.

Il y a également eu un choix à faire entre les 2 bases de données Prometheus et InfluxDB. Nous avions d'abord opté pour Prometheus, cependant la complexité de son installation et de sa configuration se fera ressentir très vite. Dès les premières difficultés rencontrées, nous avions décidé de créer deux équipes de deux étudiants, l'une travaillant sur Prometheus, l'autre sur InfluxDB afin de tester leur mode d'exécution puis d'effectuer des comparatifs entre ces dernières, tout en optimisant au mieux notre temps de travail durant la phase de développement. Premièrement, dans le cadre de notre projet, nous devions créer des métriques personnalisables, avec Prometheus il a fallu créer des points de terminaison avec des exportateurs de métriques (utilisation dans ce cas précis de PushGateway) ce qui nécessite des installations et configurations supplémentaires. Les difficultés étaient présentes jusqu'au code, en l'occurrence ici PushGateway où l'envoi de métriques dépendait exclusivement des classes nécessaires au fonctionnement de

l'exportateur et qu'il fallait de plus retrouver sur internet ; une analyse approfondie des méthodes de la classe a dû également être mise en place pour comprendre le déroulement et les différentes manières d'envoi de métriques (jauge, collection, registre…). Un autre point qui fait défaut est au niveau de la persistance des données qui dépendra à nouveau d'autres exportateurs (proposant par ailleurs parmi nos recherches InfluxDB comme solution de stockage de données pour Prometheus). Tous ces points nous ont mis à l'évidence que dans le cadre de notre projet, nous devions nous rabattre sur InfluxDB.