

Projet : Getenv.

SAE 501 – VCOD : ANALYSE ET CONCEPTION D'UN OUTIL DECISIONNEL

SAE 601 – VCOD : DEVELOPPEMENT ET TEST D'UN OUTIL DECISIONNEL



Thibault VIVIER

UNIVERSITE DE LORRAINE | B.U.T. SCIENCE DES DONNEES – VCOD

Table des matières

Table des matières.....	3
1. Contexte	5
1.1. Contexte théorique	5
1.1.1. Descriptif.....	5
1.1.2. Attendus et objectifs de la SAE.....	5
1.2. Contexte matériel et environnement	6
1.3. Cahier des charges et objectifs	7
2. Analyse.....	8
2.1. Veille technique	8
2.1.1. Webinaire DataGrandEst.....	8
2.1.2. Conférences Devovx	12
2.1.3. Recherches supplémentaires	14
2.1.4. Synthèse	15
2.2. Analyse	16
2.2.1. Définition du projet.....	16
2.2.2. Solutions pratiques envisagées.....	16
2.3. Chaîne décisionnelle	17
2.3.1. Architecture	17
2.3.2. Cas d'utilisations pratiques.....	19
3. Conception, développements et tests.....	21
3.1. Gestion du projet	21
3.1.1. Méthodologie	21
3.1.2. Outils utilisés.....	22
3.2. Stations de collecte et capteurs	22
3.2.1. Le Raspberry PI	22
3.2.2. Les capteurs et émetteurs	23
3.2.3. Architecture logicielle	25
3.2.3. Station de collecte : premiers traitements.....	25
3.3. Traitement et intégration de données	27
3.3.1. Environnement Virtuel	27
3.3.2. Envoi des données en temps réel	27
3.3.3. Traitement des images	30
3.3.4. Intégration en base de données	32
3.4. Tableau de bord	33
3.4.1. Outil utilisé : Power BI.....	33

3.4.2. Valorisation des données	33
4. Pistes d'amélioration	36
4.1. Matérielles	36
4.2. Logicielles	36
4.3. Organisationnelles	37
5. Bilan	38
6. Bibliographie	39
6.1. Référentiel abréviation	39
6.2. Veille technique	39
6.2.1. Webinaire DataGrandEst	39
6.2.2. Conférences Devovx	40
6.2.3. Recherches personnelles	41
6.3. Documentation technique	41

1. Contexte

1.1. Contexte théorique

Le programme national de la formation explicite le contexte théorique nécessaire à la compréhension de ces Situations d'Apprentissage et d'Evaluation (SAE). Après un bref descriptif des SAE, leurs attendus et objectifs sont ensuite détaillés .

1.1.1.Descriptif

Extraits du programme national :

SAE 501 VCOD : Analyse et conception d'un outil décisionnel

L'étudiant est mis en situation d'analyse et conception d'outil décisionnel. À partir de la commande d'un outil formulée par un client pour supporter son activité décisionnelle, il devra analyser les besoins clients, proposer une architecture logicielle, spécifier les fonctionnalités offertes par l'outil et concevoir des interfaces de visualisation des données restituées. Seront également prises en charge la modélisation des données, la création de la structure de stockage associée et la spécification des solutions de collecte et/ou d'intégration de données préconisées.

SAE 601 VCOD : Développement et test d'un outil décisionnel

L'étudiant est mis en situation de développement, de test et de livraison d'un outil décisionnel, selon un cahier des charges imposé. L'étudiant doit savoir travailler avec des approches professionnelles du développement logiciel (méthode agile, gestionnaires de version de code, ...) et se confronter aux exigences d'une gestion de projet devant se solder par la livraison d'un produit logiciel développé à plusieurs, de manière itérative et incrémentale, dont on attend qu'il soit fonctionnel.

1.1.2.Attendus et objectifs de la SAE.

Extrait du programme national :

SAE 501 VCOD : Analyse et conception d'un outil décisionnel

La mise en place d'une solution logicielle passe par une phase nécessaire d'analyse des besoins afin d'identifier les spécifications de l'outil logiciel attendu. En tant que développeur décisionnel, l'étudiant pourra être amené à analyser les besoins BI exprimés par les utilisateurs métiers afin de les accompagner dans la prise de décision et à concevoir une solution décisionnelle à mettre en œuvre. Il devra en outre définir les livrables attendus et les tests à réaliser pour s'assurer que le rendu final réponde aux besoins exprimés.

Les objectifs de cette SAE sont les suivants :

- *Mener l'étudiant à comprendre que la mise en place d'une solution décisionnelle nécessite une réflexion en amont sur l'architecture logicielle.*
- *L'accompagner dans la spécification des fonctionnalités de l'outil, issue de l'analyse des besoins.*
- *Lui faire percevoir que la réflexion quant à l'interface de restitution est nécessaire en amont de la conception.*

Afin de répondre à ces objectifs, nous avons pu être guidé dans nos recherches et travaux préparatoires afin de bien appréhender les diverses problématiques et enjeux de ce projet. Nous avons pu aussi être accompagnés techniquement lors de la découverte de nouveaux outils afin de nous les réapproprier. Nous avons ainsi pu explorer une grande variété de ressources afin de mener les réflexions attendues autour de ce projet.

SAE 601 VCOD : Développement et test d'un outil décisionnel

Après la phase d'analyse des besoins et d'identification des spécifications de l'outil logiciel attendu, le développement d'une solution décisionnelle doit se faire à partir de méthodes de développement adaptées et professionnelles. La validation des programmes, passant par une phase de test et de recette est une démarche fondamentale pour assurer la qualité du programme.

En tant que développeur décisionnel, l'étudiant pourra être amené à concevoir, réaliser et maintenir des applications aidant la production, en collectant et stockant les données, puis tester les différents composants de la solution décisionnelle mise en place, et intégrer et mettre en production le développement.

Les objectifs de cette SAE sont les suivants :

- *Mettre en œuvre la démarche décisionnelle dans sa globalité.*
- *Amener l'étudiant à s'approprier les approches professionnelles du développement logiciel.*

Pour répondre à l'objectif : mettre en œuvre une démarche décisionnelle dans sa globalité, l'étudiant doit concevoir un outil décisionnel en se basant sur l'ensemble du cycle de la donnée. Il doit développer un outil s'appuyant sur la collecte de données, puis déployer des processus de traitement et de stockage ainsi que des outils d'analyse et de valorisation des données.

Afin de répondre au second objectif : s'approprier les approches professionnelles du développement logiciel, l'étudiant doit réaliser ces tâches en utilisant des méthodes de travail professionnelles. Pour cela, il sera amené à utiliser des outils de collaboration et devra mettre en place une démarche de tests garantissant la qualité de son travail et de ses développements.

1.2. Contexte matériel et environnement

Afin de répondre aux objectifs de cette SAE : la mise en place d'une démarche décisionnelle globale en adoptant une démarche professionnelle, les étudiants ont pu bénéficier de cours théoriques et de séances pratiques sur divers outils et méthodes de travail professionnelles. Des outils matériels et logiciels ont été mis à leur disposition pour leur offrir une grande liberté sur leurs choix technologiques. Les étudiants ont aussi été soumis à un cahier des charges défini par leurs enseignants, précisant les problématiques professionnelles auxquels ils sont confrontés ainsi que les attendus pratiques de la SAE.

Dans le cadre de ce projet, des nano ordinateurs Raspberry PI ont été mis à la disposition des étudiants. Confronté à une technologie nouvelle, ils ont dû, dans le cadre de cette SAE, découvrir les outils matériels et logiciels accompagnant ce nouveau matériel afin de répondre à des problématiques professionnelles.

1.3. Cahier des charges et objectifs

Faire le lien entre les objectifs du PN et le contexte matériel → attendus des différents enseignants.

Les objectifs de ce projet définis par le programme national sont les suivants :

- Mettre en œuvre la démarche décisionnelle dans sa globalité.
- Amener l'étudiant à s'approprier les approches professionnelles du développement logiciel.

Afin de répondre à l'objectif : Mettre en œuvre la démarche décisionnelle dans sa globalité, les étudiants doivent réaliser un outil de monitoring basé sur une station de collecte de données. Cette station de collecte de données se doit d'être déployée sur les Raspberry PI via l'utilisation de différents capteurs. Ce matériel est mis à disposition des étudiants dans le cadre de cette SAE. Ces derniers doivent déployer les stations de collecte et assurer le cycle de vie de la donnée afin de pouvoir valoriser les données avec leur outil de monitoring. La mise en place de diverses stations pour assurer la collecte, le traitement, le stockage, l'analyse et la valorisation des données est un point essentiel dans ce projet.

Concernant la réalisation de l'objectif : s'approprier les approches professionnelles du développement logiciel, les étudiants doivent réaliser le travail demandé en utilisant leur savoirs et compétences développés au cours de cette année. Ils ont pu bénéficier de cours sur la mise en place de tests, sur l'utilisation d'outils collaboratifs, et sur des méthodes de travail professionnelles. De plus, les étudiants doivent répondre à des problématiques nouvelles accompagnant la découverte d'un nouveau contexte : l'Internet des Objets. La compréhension et la réappropriation de ce contexte sont essentielles afin de professionnaliser leur approche.

Afin de familiariser les étudiants au contexte professionnel, des rapports détaillant leurs travaux réalisés leur sont demandés. De plus, il leur est demandé de mettre à disposition le code et des différentes solutions implémentées.

Le travail de groupe, l'adaptabilité et l'autonomie sont au cœur de ce projet.

2. Analyse

2.1. Veille technique

Afin de mener une veille technique et de découvrir les problématiques de l'internet des objets (*Internet of Things* - IoT), nous avons été orientés vers des ressources par nos enseignants.

Nous avons ainsi pu nous renseigner sur divers projets et technologies existants autant sur l'internet des objets que sur le monde de la science des données en général. Ce fut intéressant de découvrir de nouveaux outils, méthodes et processus au travers d'interviews et de conférences. J'ai aussi continué cette veille technique tout au long du projet lors de recherche pour l'implémentation d'idées nouvelles dans nos développements.

2.1.1. Webinaire DataGrandEst

Le sujet du webinaire est : « Les capteurs de données en temps réel ». Après une brève introduction par les organisateurs, le webinaire se divise en trois parties qui correspondent chacune à présentation de l'un des intervenants professionnels. Chacun de ces intervenants présente son projet pendant une dizaine de minutes avant de répondre à une série de questions. Les projets des différents intervenants s'inscrivent tous dans le sujet du webinaire, à savoir une exploitation professionnelle des données collectées par des capteurs en temps réel. Chacun présente un projet très différent avec une approche qui lui est propre.

Présentation de DataGrandEst

Pour améliorer la connaissance des territoires, l'Etat et la Région Grand Est ont mis en place une démarche partenariale d'échange de données avec les acteurs publics de l'aménagement du territoire, intitulée « DataGrandEst ». Cette dynamique s'inscrit dans la poursuite de la mise en œuvre de l'infrastructure européenne des données géographiques « INSPIRE » et d'une démarche partenariale *Open Data*. La plate-forme DataGrandEst propose ainsi à ses partenaires et au public des services de recherche, visualisation, téléchargement et transformation de données conformes à INSPIRE et aux principes de l'*Open Data*.

La démarche vise quatre objectifs principaux:

- Piloter: concevoir un nouveau cadre de gouvernance et de confiance de la donnée, en y intégrant notamment les acteurs privés.
- Produire: dynamiser la production de données de référence et enrichir la plateforme de données d'intérêt local.
- Partager: accélérer l'ouverture des données et accompagner les acteurs dans la publication de leurs données
- Valoriser: développer les usages de la donnée, animer la démarche régionale, favoriser les réutilisations.

Projet 1 : StrasApp – Mathias TREFFOT

Présentation

StrasApp est une application développée à destination des habitants de l'Europole de Strasbourg. Le but de cette application est de rendre la vie des citoyens de l'Europole de Strasbourg plus simple. Proposer un suivi en temps réel des activités et des événements présents dans l'Europole.

L'application propose un suivi en temps réel des niveaux de fréquentation de lieux communs (parking, piscines, établissement publiques, ...). Elle propose aussi un service de signalement en temps réel des événements ou défaut urbains présents dans la ville (ex : accidents). D'autres fonctionnalités comme l'agenda ou la possibilité d'avoir un suivi de ses démarches administratives sont intégrées à l'application.

Un système de notifications est mis en place pour gérer les événements en semi temps réel. Ces notifications sont envoyées aux utilisateurs après un signalement ou un événement particulier (ex : pic de pollution ou encore présence d'un accident). Ces notifications sont catégorisées en familles en l'utilisateur peut choisir les familles de notifications qui l'intéresse.

Si une personne doit aller se faire inscrire à la mairie sur les listes électorales de la ville car elle vient d'emménager récemment, elle peut consulter sur l'application l'état du trafic routier et connaître le temps nécessaire pour aller à la mairie tout en ayant connaissance de la présence ou non d'accidents sur la route grâce au système de notifications. Elle peut aussi, avec l'application, connaître le niveau de fréquentation de la mairie et anticiper l'attente à laquelle elle devra faire face si elle décide d'y aller. Si cette personne se décide à ne pas aller à la mairie aujourd'hui, l'application lui rappellera la date des prochaines élections afin qu'elle puisse s'organiser en conséquence pour passer à la mairie un autre jour.

Aspect technique

La collecte des données se fait par beaucoup de sources différentes. Elles viennent soit de sources officielles (données de la collectivité de Strasbourg), soit de l'*Open Data*, soit de sources externes (entreprises, association comme Atmo Grand Est).

La multiplicité de ces sources permet de croiser les données et d'offrir une panoplie de fonctionnalités et de services très différents.

Les données sont récupérées avec des capteurs (ex : passage d'un usager dans un tripode) puis elles sont converties en flux JSON avant d'être collectées par StrasApp avec une API et elle sont enfin affichées dans l'application.

Cette application ne produit pas de données en *Open Data*, elle ne fait que réadapter et centraliser les données de diverses sources pour les mettre au service de l'utilisateur.

Remarque : Le code est propriétaire malgré l'utilisation de données en accès libre (provenant notamment de <https://data.strasbourg.eu/pages/accueil/>)

Projet 2 : Atmo GrandEst – Éric HERBERT

Présentation

Atmo GrandEst est une association à but non lucratif agréée par le Ministère de l'environnement. Leurs missions sont diverses, il s'agit principalement de surveiller les niveaux de pollution et d'assurer un suivi de la qualité de l'air à l'aide de données collectées avec des capteurs. Mais Atmo GrandEst sert aussi de référent en matière de pollution et accompagne les entreprises et les collectivités dans la mise en place de projet de dépollution. Enfin, ils sont aussi un acteur de la recherche dans le domaine de la pollution (recherche de solutions nouvelles pour faire face aux enjeux environnementaux en matière de pollution).

Voici les atouts développés par Atmo GrandEst :

- Mesures : Atmo GrandEst collecte un grand volume de données avec leurs stations de collectes de données (un total de 98 stations au moment du webinaire). Ces stations sont équipées de divers capteurs leur permettant de mesurer et de prélever un grand nombre d'échantillons (176 préleveurs et analyseurs). Les analyses effectuées concernent près de 31 polluants différents.
- Inventaire : Atmo GrandEst propose un inventaire du suivi de la qualité de l'air avec un travail d'archivage des données historiques depuis 1990 en matière de pollution. Ce suivi historique se fait selon 72 critères et se base principalement sur l'échelle de l'IRIS (Ilots Regroupés pour l'Information Statistique).
- Modélisation : Atmo GrandEst propose un outil de modélisation qui offre un suivi de la qualité de l'air à un niveau inter-régional. Cet outil de modélisation est retrouvable sur le site d'Atmo Grand Est. Il se présente sous forme de carte interactive permet d'avoir un suivi précis du niveau de qualité de l'air. Il est possible d'obtenir des indicateurs plus précis sur les villes du Grand Est (Metz, Strasbourg, Nancy, ...)

Aspect technique

Au vu des moyens déployés pour la collecte des données (nombre de stations de collecte), Atmo GrandEst gère de grand volume de données sur des échelles géographiques et temporelles très différentes. D'un point de vue géographique, les données peuvent être collectées à l'échelle d'un quartier jusqu'à un niveau régional voir transfrontalier (Allemagne, Suisse ou encore Luxembourg). Sur la dimension temporelle, l'association propose des données historisées sur 30 ans, les données des observations quotidiennes mais elle propose aussi des modèles prévisionnels en se projetant dans un futur proche. Atmo GrandEst utilise des données avec une granularité de l'ordre des 15 min voir 30 min pour mener ses analyses.

Pour la collecte de ses données, Atmo GrandEst utilise des micros capteurs. Dans la vidéo, Éric Herbert détaille le fonctionnement basique d'un micro capteur. Ces micro capteurs sont composés d'un élément sensible qui va réagir en fonction de l'information qu'il reçoit (ex : présence d'un polluant dans l'échantillon d'air collecté). Cet élément sensible va être associé à une partie électronique qui va transformer l'information perçue en signal électrique. Cette association est appelée capteur. Enfin, quand plusieurs capteurs sont installés ensemble, on parle de système capteur.

Ces capteurs peuvent être déployés dans différents contextes : sur du mobilier urbain, à l'intérieur d'un bâtiment, au-dessus d'un véhicule, ...

Atmo GrandEst est très concerné par la mise en place, l'utilisation mais aussi le conseil et la réglementation autour des capteurs car ils sont autant utilisateurs (suivi de la qualité de l'air, et recherche), que conseillers sur l'utilisation, la mise en place des capteurs ainsi que sur les différents enjeux accompagnant cette technologie.

Atmo GrandEst a mis en place des procédures de test et de validation des données avec des mesures de références et des tests réalisés sur des atmosphères simulées en laboratoire. La mise en place de ces tests est très importante car ils permettent d'avoir des données fiables et qualitatives. De plus, étant données que l'association est un fournisseur de données en *Open Data*, beaucoup d'autres organismes dépendent de la qualité de leurs données (ex : StrasApp)

Les données récupérées par les capteurs sont envoyées sur un *Cloud* où Atmo GrandEst va aller chercher les données pour les intégrer à sa propre base de données à l'aide d'une API. Ces données vont ensuite être utilisées pour le développement de graphiques et visualisations que les fournisseurs (propriétaire du capteur – ce n'est pas toujours Atmo GrandEst) et Atmo GrandEst vont pouvoir mettre en avant sur leur plateforme respective.

Projet 3 : VigieCrues – Fabrice HERY

Présentation

VigieCrues est un service public gérée par le Ministère de la transition écologique. Il a pour principal mission d'assurer une surveillance et un suivi du niveau des cours d'eau pour prévenir les futures inondations. Cette surveillance a pour but d'assurer une meilleure prévention de la population, d'une plus grande efficacité des services d'intervention (ex : pompiers) et de permettre à chaque citoyen de se tenir informé sur les risques de crues sur diverses échelles géographiques (du département jusqu'à l'ensemble du territoire)

La collecte se fait par des stations de capteurs (2 à 3 capteurs par station – 5000 stations sur le territoire français) qui mesurent la hauteur d'eau des fleuves et affluents. Ces stations envoient leurs données sur une base de données placée sur un serveur (AQUAREEL) avant d'être transférées sur une base de données centralisée et protégée. Cette base de données (HYDRO 3) sert aux différentes applications comme VigieCrues. L'échelle temporelle des données peut varier en fonction des régions de leur vulnérabilité. Par exemple, les régions à fort régime torrentiel mesurent des données toutes les 5 min alors que d'autres régions vont envoyer des données à une fréquence de l'ordre de 30 min.

VigieCrues est composé de deux services principaux : l'unité d'hydrométrie et le service de prévision des crues.

La mission principale de l'unité d'hydrométrie est de corriger les données quotidiennes récupérées avec les capteurs afin d'avoir une information la plus sûre possible.

Le service de prévision des crues a développé un outil à partir des données collectées afin d'avoir un aperçu de l'état des différents affluents de France en quasi temps réel, en plus de leurs prévisions. Cet outil propose des cartes interactives et des modèles prévisionnels sur les crues afin de les anticiper et mieux pouvoir prendre des mesures adaptées en fonction de la gravité des crues et de leur localisation. Le service VigieCrues propose une catégorisation des cours d'eau en temps réels en fonction des risques de crues encourus.

2.1.2. Conférences Devovx

Devovx est une conférence réunissant des professionnels du développement dans divers domaines. Créé en Belgique en 2001 sous le nom de JavaPolis, la conférence changera de nom en 2008 pour devenir Devovx. Sa version française : Devovx France, verra le jour en 2011. Lors de cette conférence, de nombreux ateliers et présentations ont lieux, visant à présenter les nouvelles innovations dans le domaine du développement et de l'informatique. La plupart des vidéos présentées ci-dessous viennent de conférences Devovx France.

Conférence 1 : Kit de survie pour l'IoT façon DIY - Laurent HUET

Résumé

La présentation a pour but de sensibiliser le public sur la facilité de construction d'un réseau IoT. Le présentateur, Laurent HUET, aborde les différents points clés d'un réseau électronique en commençant notamment par la Raspberry PI et d'autres microcontrôleurs équivalents. Il parle ensuite des coûts des différents composants et de leur rôle électronique (ex : les résistances permettent d'éviter une surtension des autres composants). Il insiste sur la facilité de mise en place et montre ainsi que l'IoT, loin des laboratoires ou de demander des gros moyens financiers, peut être réalisé très facilement à une échelle locale sans un gros budget.

Ressenti

J'ai choisi cette présentation car il me semblait intéressant de découvrir les divers composants d'un réseau électronique basé sur Raspberry PI (ou équivalent) car cela s'inscrit dans le cadre du projet. De plus, la vidéo m'a permis de comprendre que la mise en place de ces circuits est loin d'être inaccessible et que nous pouvions tout à fait être en mesure de mettre un système similaire en place après quelques heures de travail. Enfin cette vidéo est une bonne introduction à l'aspect pratique et technique de l'IoT.

Conférence 2 : Premiers pas avec un microcontrôleur et Google Cloud IoT Core - Gautier MECHLING

Résumé

Cette présentation Devovx France nous permet d'avoir un aperçu des bases sur les microcontrôleurs et la récupération des données des capteurs. Gautier MECHLING commence sa présentation en introduisant les microcontrôleurs, leurs différents composants et en expliquant leur principale fonction : faire tourner du code en continu sur un processeur indépendant. Après cela, il montre comment « flasher » un microcontrôleur (= intégration de code) en live. Il explique ensuite une partie de connectique et d'électronique en créant un capteur pour son circuit et récupérant les données dudit capteur à l'aide du microcontrôleur. Enfin il envoie ces données sur un Cloud et s'en sert pour créer des visualisations en temps réel.

Ressenti

J'ai décidé de choisir cette présentation car elle me semblait essentielle pour comprendre les bases de l'IoT et des microcontrôleurs. En effet, notre projet concerne la récupération de données à partir de capteurs et de microcontrôleurs, cette vidéo semblait tout à fait adaptée à cette problématique. De plus, le présentateur appuie tous ses points théoriques avec un exemple pratique bien expliqué. Ce qui m'a le plus attiré dans cette présentation était la mise en place des différentes connections (physiques ou numériques) entre les composants (microcontrôleur, capteurs, Cloud) ainsi que le cheminement de la donnée dans ce circuit.

Conférence 3 : La sécurité dans l'IoT difficultés, failles et contre-mesures - A. DUQUE

Résumé

Cette présentation Devovx nous permet d'avoir un aperçu des concepts de la sécurité informatique appliquée à l'IoT. La présentation débute par une brève introduction de l'IoT et la sécurité, notamment en caractérisant des données sécurisées comme étant disponibles, intègres, et avec une confidentialité garantie. Alexis DUQUE introduit ensuite les dangers auxquels est confronté l'IoT, comme l'environnement des objets connectés, la grande variété des appareils dans un réseau IoT et l'étendue du champ d'attaque de ces appareils (physiques et numériques) entre autres. Il insiste sur le fait que les utilisateurs et les entreprises sont encore trop peu sensibilisés à ces problématiques dues à la nouveauté de l'IoT. Il détaille ensuite toutes les formes les plus courantes d'attaques informatiques sur des objets connectés en donnant des exemples. Il cite notamment des attaques d'escalade des privilèges, d'injection de code et d'attaque par Bluetooth (*Blue Born*). Il insiste notamment sur le fait que c'est le rôle des développeurs d'être conscient des problématiques de sécurité dans l'IoT et que parfois, il suffirait d'être aussi prudent que sur le développement des applications web pour éviter certaines vulnérabilités.

Ressenti

J'ai décidé de choisir cette présentation car les problématiques de sécurité me semblaient pertinentes bien que cela ne relève pas forcément de notre domaine. Etre sensible aux problématiques de sécurité est quelque chose d'important quel que soit le domaine d'application afin éviter des problèmes, parfois mineurs, qui peuvent souvent avoir de grosses conséquences pour les individus et/ou les entreprises. De plus, nous vivons dans un monde où l'IoT est omniprésent, et connaître les failles ou des moyens de se prémunir des attaques, par l'adoption de bonnes pratiques, est toujours utile que cela soit en tant que développeur ou en tant qu'utilisateur.

Conférence 4 : Le Big Data, de la récolte jusqu'à la production à grande échelle ! - Alexis SLAWNY & Mathias KLUBA

Résumé

Cette présentation sur le BigData commence par une brève introduction au concept et à ses domaines d'application (marketing, surveillance, santé, aéronautique, etc...). Ensuite les présentateurs détaillent comment fonctionne le BigData ou plutôt ses problématiques fondamentales, à savoir le traitement d'un grand volume de données. Ils expliquent les différentes solutions existantes : scalabilité verticale (augmentation des performances d'une machine) et horizontale (division des traitements de données sur plusieurs machines – cf. Hyper convergence). Ils donnent ensuite un exemple pour expliquer ces concepts fondamentaux. Après cela, ils présentent les différentes technologies utilisées dans le BigData : Hadoop, Spark, NoSQL, et leur fonctionnement basique respectif. Ils insistent beaucoup sur la scalabilité horizontale et sur la répartition de l'exécution des traitements sur plusieurs machines (ressemble au *multi-processing* Python vu en cours, étendu sur plusieurs machines). Enfin, les présentateurs appuient leur propos avec divers exemples de traitement de données en Open Source (meilleure bière de Paris et algorithme de choix de la meilleure route commerciale dans un jeu vidéo).

Ressenti

Cette présentation m'a grandement intéressé car elle permet de revoir les fondamentaux de Big Data et elle fait écho à des concepts vus en cours (*multi-processing*). De plus, elle est aussi intéressante dans le cadre de ce projet car au vu des données à traiter (très variées et volumineuses), avoir une connaissance basique du BigData semble essentiel. Enfin le dernier aspect qui m'a intéressé est le fait que les applications BigData présentées ne soient pas restreintes à un seul domaine d'application mais semblent bel et bien utilisables dans de nombreux projets.

2.1.3. Recherches supplémentaires

En parallèle de la veille technique que nous avons menée, orientée par nos enseignants, nous avons pu faire des recherches personnelles complémentaires. Si une grande partie de ces recherches en autonomie se sont portées sur des aspects techniques (utilisation d'outils, de logiciels ou de librairie de programmation), nous avons aussi pu nous intéresser aux différents projets et solutions existant.

Recherche 1 : Flux Vision – Orange

Flux vision est un outil développé par l'opérateur français Orange. Il s'agit d'un outil visant à étudier les flux humains à partir de données captées par les opérateurs et de données GPS. Le but est de fournir un service qui fournit un ensemble d'indicateurs clés à partir de l'analyse des fréquentations selon les besoins du client. L'idée est de fournir un ensemble d'informations très précis sur le type de flux humain en étudiant les profils issus des données captées mais aussi en garantissant une précision géographique maximale par l'utilisation de données GPS. Cet outil est utilisé par de nombreuses industries comme le tourisme ou le retail. L'entreprise a mis un accent très prononcé sur le respect des réglementations et les contraintes légales liées aux données imposées par la Commission Nationale de l'Informatique et des Libertés (CNIL) et le Règlement Général de la Protection des Données (RGPD). Afin d'atteindre des objectifs de fiabilité de la données, Orange pondère les données récupérées par ses parts de marchés locales. En effet, s'ils sont le leader sur le marché des opérateurs mobiles, ils ne possèdent pas le monopole.

Recherche 2 : Chasing Your Tail With a Raspberry Pi – Matt EDMONDSON

Ce projet a été présenté lors de la conférence Black Hat de 2022. Black Hat est une conférence sur la sécurité informatique, réunissant les acteurs du domaine autour de présentations et réunion. Matt EDMONDSON, a présenté lors de cette conférence un outil portatif, développé sur Raspberry PI permettant de savoir si l'on est suivi. Cet outil scan les appareils environnant afin d'enregistrer leurs adresses MAC. En comparant les diverses adresses MAC (Media Access Control) enregistrées au cours du temps, il peut ainsi savoir s'il est ou non suivi. Cet outil s'adresse avant tout à un public concerné par les questions de filatures ou de confidentialité, mais il peut être intéressant dans notre cas d'utiliser par exemple, un système de scan des appareils pour réaliser des mesures de fréquentation. L'outil se sert de Python et du package Kismet, développé notamment pour scanner les appareils environnant connectés au Wifi et au Bluetooth.

Recherche 3 : Horizon Europe

Horizon Europe est un programme d'innovation lancé par l'Union Européenne (UE) visant à maintenir l'UE en tant que l'un des leaders mondiaux de la recherche et de l'innovation. L'un des objectifs de ce programme est de renforcer la base scientifique et technologique de l'UE. Dans le cadre de ce programme, l'UE a adopté *l'European Data Strategy*, qui a pour but de faire de l'Europe l'un des leaders mondiaux sur le sujet de la donnée. Il s'agit de mettre en place un cadre législatif encadrant le domaine et de mettre en place un marché unique de la donnée, international, qui bénéficierait à tous les acteurs. De plus, dans cette idée de recherche et d'innovation, l'UE a adopté une politique européenne en matière d'Internet des Objets (IoT – Internet of Things) afin d'accompagner les acteurs du domaine vers le monde de demain.

Recherche 4 : Industrie 4.0

L'industrie 4.0 est un terme utilisé pour définir le secteur de l'industrie après la quatrième révolution industrielle. Les trois premières révolutions industrielles désignent les changements majeurs de l'industrie ayant eu lieu à l'émergence et la démocratisation de :

- La machine à vapeur (XVIII).
- Le pétrole et l'électricité (XXe siècle).
- L'électronique et les technologies de l'information (XXIe siècle).

La quatrième révolution industrielle s'apparente à une transformation numérique du terrain. Ce changement a pour d'offrir une productivité et une flexibilité accrues. Avec l'ère de la personnalisation des produits et de l'instantané, l'industrie doit pouvoir fournir un service répondant aux besoins du client de façon efficace, en temps réel. L'émergence de nouvelles technologies dans les domaines de l'IoT, du BigData, du *Cloud Computing* et de l'Intelligence Artificielle accompagne cette nouvelle révolution en offrant des solutions techniques aux nouvelles problématiques de l'industrie.

Un exemple d'entreprise ayant mis en pratique l'industrie 4.0 est Volkswagen :

Ils ont investi près de 4 milliards d'euros dans des projets de digitalisation, et ont ainsi pu moderniser leurs chaînes de production de véhicules afin d'offrir la possibilité de produire des véhicules personnalisés en plus de leurs véhicules traditionnels produits en série. Le véritable progrès se situe notamment dans le fait que quel que soit le véhicule produit, leur chaîne de production reste interrompue. De plus, ils ont su mettre en place des stations de monitoring afin de centraliser les informations sur la chaîne de production sur un tableau de bord unique.

D'autres entreprises comme Siemens, Toyota ou Bosch sont aussi des acteurs majeurs de l'industrie 4.0.

2.1.4. Synthèse

Faire une synthèse des recherches réalisées. Faire un lien clair avec la SAE.

L'ensemble de ces recherches et de cette veille technique réalisée m'a permis de bien comprendre les enjeux professionnels liés à l'IoT et au Big Data. En effet, les nouvelles problématiques professionnelles liées à une société de services, de produits personnalisés, et de réponse quasi-instantanée aux besoins accélère l'innovation technologique et le déploiement de solutions nouvelles. Aucun domaine professionnel n'est épargné, qu'il s'agisse

de services publics, de villes intelligentes, d'industries, de tourisme, de santé, de sécurité ou encore de banques et assurances, tous ont maintenant besoin de s'adapter et d'utiliser les solutions en terme d'IoT ou de Big Data.

D'un point de vue technique, cette veille technique m'a permis d'appréhender la grande diversité de solutions existantes. Il existe autant de solutions qu'il y a de problématiques. Les choix technologiques à entreprendre sont dépendants des besoins liés au projet. L'identification des besoins techniques à partir de problématiques professionnelles est le principal enjeu afin d'utiliser les outils appropriés et de pouvoir répondre au mieux à ces problématiques.

2.2. Analyse

2.2.1. Définition du projet

Afin de respecter le cahier des charges et de répondre à l'objectif principal de cette SAE : réaliser un outil de monitoring, nous avons choisi de développer un outil permettant de monitorer les salles de l'I.U.T. à l'aide des Raspberry Pi. Pour ce faire, nous avons pour objectif de mettre en place une station de collecte de données, équipées de capteur, puis des stations de traitement de ces données, avant de paramétrer une ou plusieurs stations de monitoring, équipées de tableau de bord permettant de valoriser nos données. L'idée étant de pouvoir fournir des indicateurs clés sur des problématiques précises en fonction des événements ou des périodes de la journée (incendie, nuit, fréquentation des salles, ...).

2.2.2. Solutions pratiques envisagées

Plus concrètement, il pourrait être intéressant de mettre en place des mesures et calculs de fréquentation des salles de l'I.U.T. en fonction des données captées par le Raspberry Pi. L'analyse de la température, du taux de CO₂, du bruit, du détecteur de mouvement et d'une photo de la pièce nous permettrait par exemple de calculer le nombre de personnes dans la pièce. Bien que certaines données ne permettent que de faire des approximations sur le nombre de personnes présentes, leur croisement pourrait permettre de gagner en précision. Cela pourrait servir à la mise en place de panneau d'information à destination des étudiants en identifiant les points de passages ou les salles les plus utilisées pour améliorer la transmission de l'information. Un autre usage possible serait de guider la prise de décision dans les travaux de rénovation : faciliter l'accessibilité des salles très fréquentées ou repenser l'aménagement des locaux moins utilisés.

Un autre objectif intéressant serait de repérer des événements anormaux se déroulant dans les salles de l'I.U.T. à partir des données, par exemple une augmentation anormale de la température sur une durée prolongée pourrait être significative d'un incendie comme la présence d'une personne en pleine nuit dans une salle pourrait être synonyme d'intrusion. Le croisement de plusieurs types de données rend d'autant plus intéressant leur analyse afin d'identifier ces différents événements. Un incendie sera, par exemple, non seulement repérable par une augmentation permanente de la température, mais aussi par une augmentation significative du taux de CO₂ et par la présence d'un bruit de fond assez marqué (crépitements des flammes). Un autre exemple serait la mise en place d'une alerte automatique pour l'aération des pièces : quand le taux d'humidité et le taux CO₂ deviennent trop importants, il faut aérer la pièce. Cela pourrait aider dans la prise de décision de mesures de remédiation

(en identifiant les causes potentielles d'évènements indésirables) ou d'action immédiate (ouverture de la fenêtre).

Enfin, il pourrait tout à fait être possible, à partir des Raspberry PI de mettre en place un réseau de caméra de surveillance. En captant le flux vidéo fourni par la caméra de différentes stations, il serait possible de monter un système de vidéosurveillance. Il serait théoriquement possible, à partir de ces différents flux vidéo d'automatiser la gestion des absences des étudiants. En effet, la démocratisation des modèles de reconnaissance faciale permet maintenant d'identifier de façon toujours plus précise les personnes présentes sur un flux photo ou vidéo. En enregistrant l'heure à laquelle un visage est identifié par le système, il est possible de notifier un retard ou une absence (cela reste conditionné à la fiabilité du modèle, à sa précision et à sa capacité à détecter tous les visages sur une photo d'ensemble). Malgré les nombreuses contraintes, cela reste théoriquement possible. De plus nous serions confrontés à des problématiques légales (cf. Règlement Général de Protection des Données – RGPD), dû au recueil des accords individuels, à la collecte, au traitement, à l'analyse et au stockage de données personnelles. Un outil destiné aux étudiants permettant d'être informé de leurs diverses absences, et de leurs conséquences pourrait être développé.

2.3. Chaîne décisionnelle

2.3.1. Architecture

Pour mettre en œuvre une démarche décisionnelle globale, il a fallu définir les différents maillons de notre chaîne décisionnelle. Cette chaîne décisionnelle repose sur les différentes étapes du cycle de la donnée : la collecte, le traitement, le stockage, l'analyse et la valorisation des données. En partant de ce principe, nous devons ensuite détailler le rôle de nos différentes stations, ainsi que leurs interactions avec le reste du système. Plus simplement, il a fallu définir l'architecture de notre chaîne décisionnelle.

Une chaîne décisionnelle simple (cf. *Figure 1*) est composée de 4 stations principales :

- La station de collecte : implémentée sur le Raspberry PI, elle permet de récupérer les données des capteurs et de les envoyer à la prochaine station.
- La station de traitement : elle applique des calculs et analyses sur les données collectées. C'est aussi sur cette station que nous pouvons insérer les données en base de données.
- La station de stockage : il s'agit d'une base de données contenant les données captées et/ou leurs dérivations.
- La station de monitoring : permet d'avoir un suivi et des indicateurs clés sur les données présentes dans la base de données. Réservée à l'utilisateur, c'est le dernier maillon de la chaîne et l'objectif final de ce projet.

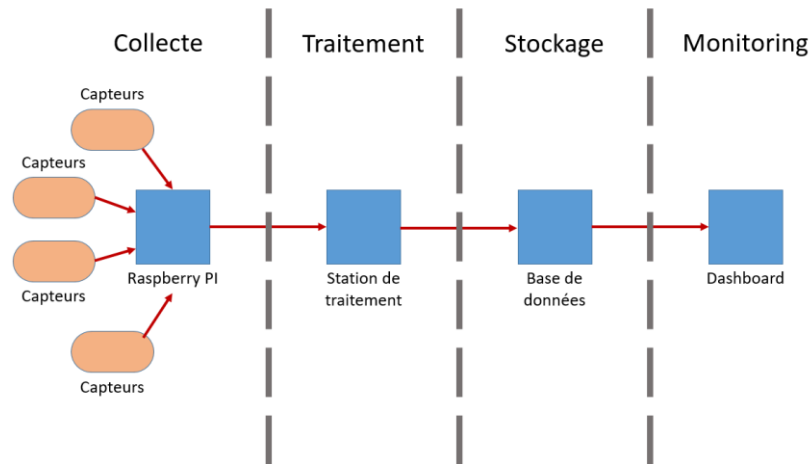


Figure 1

Alors que l'organisation de ces stations peut être différente, cet ordre applicatif permet de simplifier l'organisation et le déploiement pratique du système. Bien que dans le schéma ci-dessus (Figure 1), les stations relatives à chaque tâche soient uniques, il est possible d'imaginer de multiples stations de collecte, de traitement, de stockage et de monitoring. Ces stations peuvent être associées à des tâches très différentes ou bien réaliser des tâches similaires selon les besoins et les problématiques auxquelles nous souhaitons répondre (cf. 2.3.2.).

Sur le schéma ci-dessous (Figure 2), une architecture plus complexe du système est présentée. Dans cette configuration, les stations de traitement sont associées à des tâches distinctes, parfois indépendantes, parfois complémentaires, qui permettent ensuite de répondre à des problématiques différentes, incarnées par leur station de monitoring respective. Il s'agit ici d'un exemple théorique, et nous allons ensuite appliquer un modèle similaire à un cas concret (cf. 2.3.2.).

L'idée est la suivante : nous ne sommes pas obligés de nous limiter à la réalisation d'un unique outil de monitoring, avec un flux linéaire de données. Il est tout à fait possible d'imaginer des traitements multiples, nécessitant plusieurs schémas de base de données indépendants. Il est tout à fait possible d'imaginer divers outils de monitoring, variant selon les besoins, les problématiques mais aussi selon les supports matériels à disposition (ordinateur portable, smartphone, ...).

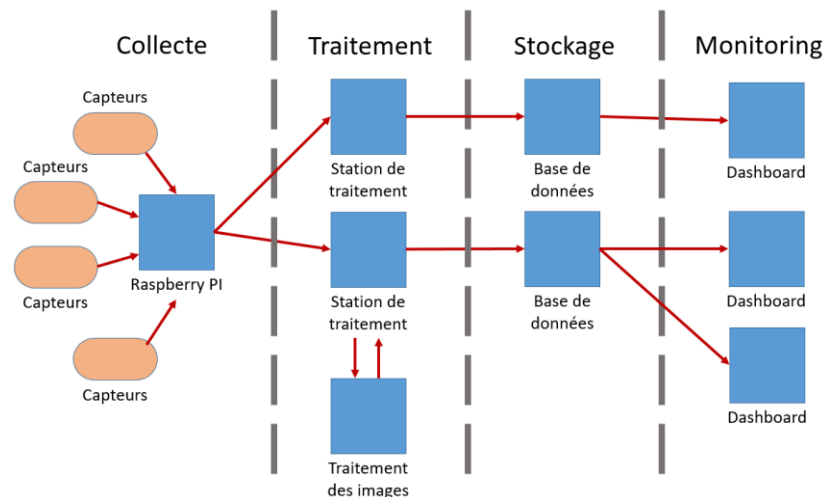


Figure 2

2.3.2. Cas d'utilisations pratiques

Cette architecture théorique peut être appliquée aux cas concrets décrits précédemment (cf. 2.2.2.).

Dans le cadre de l'estimation de la fréquentation dans les salles de l'I.U.T., nos stations de collecte pourraient être l'ensemble des Raspberry PI installés dans les salles. Avec un système d'intégration de données en temps réel, il est possible de récupérer les données collectées sur une station de traitement : un ordinateur ou un serveur indépendant des Raspberry PI. Ces données peuvent être traitées si besoin, par exemple en calculant le nombre de visages repérés sur une photo prise depuis une station de collecte, puis intégrée dans une base de données. La base de données quant à elle peut être déployée sur un serveur, accessible depuis la station de traitement mais aussi depuis les stations et outils de monitoring. Différents supports peuvent ensuite être utilisés afin de monitorer la fréquentation dans les salles de l'I.U.T. Par exemple, le chef de département des salles associées à un groupe de stations de collecte pourrait être informé en temps réel de l'estimation de la fréquentation dans ces salles, en plus d'être informé de l'état de la salle (fenêtre ouverte, lumière allumée, ...). Il pourrait ainsi intervenir pour agir dans ces salles s'il le juge nécessaire. Reposant sur les mêmes informations, un tableau de bord pourrait être mis à disposition des étudiants pour les informer des salles disponibles s'ils souhaitent travailler en autonomie. Un autre outil de monitoring pourrait être par exemple un tableau de bord fournissant des informations et des indicateurs clés à propos de la fréquentation des salles de l'I.U.T. sur une période donnée (mois, trimestre, semestre, ...).

Concernant la détection d'évènement anormaux, incendie ou intrusion par exemple, des alarmes spécialisées seraient bien plus efficaces. Cependant il pourrait être envisageable de déployer un système annexe, informant les pompiers ou les personnes responsables en cas d'évènement anormaux. Si l'une des stations de collecte détecte une intrusion, l'intégration de données en temps réel permet à la station de traitement de déclencher un processus de prévention ou d'alarme associé à l'évènement : envoi d'un message ou appel d'un numéro préenregistré par exemple. De plus, malgré l'alerte, la collecte de données devrait continuer. Cela pourrait ensuite permettre, à l'aide d'un tableau de bord ou autre outil de monitoring, d'identifier les causes ou les sources d'un évènement: l'intrusion a eu lieu par la fenêtre de

telle salle, car un promontoire permet d'y accéder et la fenêtre était restée ouverte. Cela pourrait faciliter la mise en place de mesure de remédiation ou de sécurisation par exemple.

Enfin, dans une problématique de notification des absences et des retards. L'identification des élèves par une station de traitement spécialisée pourrait être reliée à une base de données différentes des cas d'usages précédents. L'un des outils de monitoring associé serait, par exemple, une application destinée aux étudiants pour assurer le suivi de leurs absences et les potentielles pénalités associés à ces dernières.

Grâce à l'architecture détaillée précédemment (cf. 2.3.1.), il serait possible de déployer ces diverses chaines décisionnelles simultanément. Ces diverses chaines décisionnelles reposent toutes sur un ensemble de stations variées. Elles ont pour point commun les stations de collecte de données mais parfois aussi certaines stations de traitement voire une base de données. Ce partage de ressources permet de déployer plusieurs chaines décisionnelles, bénéficiant toutes de leur propre architecture, dans une seule architecture globale permettant de répondre à des problématiques variées.

3. Conception, développements et tests

3.1. Gestion du projet

3.1.1. Méthodologie

Outil de collaboration

Afin de nous répartir le travail et de nous partager nos scripts et diverses réalisations, nous avons mis en place un dépôt GitHub. Voici le lien vers le dépôt en question :

https://github.com/VIV-T/Analyse_Conception_outil_decisionnel

Ce dépôt regroupe tout le travail réalisé dans le cadre de ce projet. Les scripts utilisés sur les stations de collecte, ceux utilisés par la station de traitement, ainsi que les différents tableaux de bord créés. Des rapports et présentations sont aussi disponibles sur ce dépôt.

Tests

En matière de tests, j'ai pu développer des scripts Python permettant de tester des fonctionnalités distinctes de notre code. En effet, les scripts de tests permettent de s'assurer du bon fonctionnement du :

- Transfert de données via MQTT et le package Python : paho-mqtt
- Transfert de données via Kafka et le package Python : kafka-Python. (Script fonctionnel si un serveur Kafka est déployé et renseigné dans le script)
- Traitement des images avec DeepFace.
- Scan Bluetooth des appareils environnants avec le package Python : Pybluez.
- Collecte des données de la station de collecte avec les différents capteurs.
- Calcul de moyennes sur les valeurs numériques collectées sur une durée déterminée avec les *Dataframes* Pandas.
- Automatisation de la capture d'une photo avec la caméra installée sur le Raspberry PI.

Ces différents scripts de tests sont trouvable dans le dépôt GitHub du projet. Le but de ces tests est de nous assurer du bon fonctionnement de chacun des modules de code indépendamment du reste du code.

Un autre type de test implémenté est la création de données factices. En effet, afin de construire des prototypes de tableau de bord, il nous fallait des données des stations de collecte. Cependant à ce stade du projet, nous n'avions pas encore implémenté nos scripts de transfert et d'intégration de données. Il nous a donc fallu simuler les données captées avec des scripts Python. Ces scripts nous ont permis de générer des données factices mais réalistes, à partir de paramètres définis par l'utilisateur (granularité des données, période de génération, ...). L'idée fut de fournir des données diverses, permettant d'observer l'apparition de phénomènes et d'évènement à partir de notre tableau de bord (ex : ouverture de fenêtre, salle occupée, ...).

3.1.2. Outils utilisés

Station de collecte :

- Raspberry PI 4, capteurs et émetteurs Grove.
- Packages Python : Pandas, Time, Pybluez ainsi que les scripts développés par notre enseignant pour les interactions avec les capteurs et émetteurs connectés au Raspberry PI.

Intégration de données :

- MQTT (Python : paho-mqtt) : transfert de données de la station de collecte à la station de traitement.
- Kafka (Python : kafka-Python)
- Python: pour intégrer les données récupérées et traitées par la station de traitement dans notre base de données.

Traitement :

- DeepFace (Python) : traitement des images.
- Pillow (Python) : traitement des images.
- Pandas (Python) : utilisation de *Dataframes* en Python.

Base de données :

- PostgreSQL

Tableaux de bord :

- Power BI

3.2. Stations de collecte et capteurs

Dans le cadre de ce projet, l'I.U.T. nous a fourni des Raspberry PI et un ensemble de capteurs.

3.2.1. Le Raspberry PI

Un Raspberry PI est un nano ordinateur. Un nano ordinateur est une ordinateur d'une taille semblable à celle d'une carte de crédit. La puissance d'un nano ordinateur est bien moindre comparée à celle d'un ordinateur classique. Les nano ordinateurs sont souvent utilisés pour mettre en place des systèmes spécialisés comme la collecte de données par exemple.

Il existe de multiples usages au Raspberry PI. En voici quelques exemples :

- Rétro-gaming.
- Automatisation de système et domotique (gestion du chauffage, ...).
- Robotique.
- Hébergement d'un serveur (léger).

Bien que les ressources d'un Raspberry PI soient limitées, il reste cependant un ordinateur. Son faible coût et sa petite taille en font un objet d'intérêt dans bien des domaines, notamment dans l'IoT. Il est possible de faire un comparatif entre un ordinateur « classique » et le Raspberry PI 4 qui nous a été fourni (cf. Tableau 1).

	Raspberry PI 4	Ordinateur classique moyen
Nombre de cœurs (du processeur)	4	4 – 16
Fréquence d'horloge	1.5 GHz	2.5 – 5 GHz
RAM	2 – 8 GB	8 – 64 GB
Taille	56,5mm x 85,6mm x 11mm	300mm x 200mm x 100mm
Poids	46 g	5 – 10 Kg

Tableau 1

Le Raspberry PI 4 possède un processeur ARM Cortex A72 avec 4 cœurs cadencé à 1.5Ghz. En terme de mémoire vive (Rapid Answer Memory – RAM) il existe plusieurs modèles allant du model B1 avec 1 Go de RAM au model B8 avec 8 Go tous en DDR4. Les modèles sont tous équipé d'une carte graphique VideoCore VI.

Le Raspberry PI nous a été fourni avec un système d'exploitation : Twister OS. Il s'agit d'un système d'exploitation basé sur une distribution Linux : Debian. Twister OS a été conçu pour fonctionner sur les Raspberry PI.

Etant étranger à Linux, il m'a fallu apprendre à manipuler ce nouvel outil et ses particularités, notamment les lignes de commandes. Qu'il s'agisse de la navigation dans l'arborescence de fichiers, de la manipulation de fichiers, de l'exécution de scripts ou d'installation et de mise à jour de packages, j'ai pu réaliser toutes ces tâches à partir de terminal de commande. J'ai aussi pu me familiariser avec le gestionnaire de package du système d'exploitation « apt ».

Voici quelques commandes que j'ai pu utiliser régulièrement lors de mon travail :

- **sudo apt install *nom_package*** : installer un nouveau package.
- **sudo apt update *nom_package*** : mettre à jour un package installé.
- **cd *nom_dossier*** : se déplacer vers un dossier contenu dans le dossier courant
- **cd ..** : se déplacer vers le dossier contenant le dossier courant
- **nano *fichier.py*** : éditer un fichier

3.2.2. Les capteurs et émetteurs

Notre station de collecte est donc composée de notre Raspberry PI et d'un ensemble de capteurs et d'émetteurs « Grove ». Nous avons pu connecter ces composants ensemble utilisant une « Grove Hat », qui est une plaquette que nous rajoutons au Raspberry PI et qui permet de connecter divers autres composants avec de l'analogique ou du numérique.

Voici les capteurs et émetteurs que nous avons utilisés.

Capteurs (Bleu – cf. Figure 3) :

PIR Sensor : Connecté sur un port numérique D18. Capteur infrarouge passif qui détecte la présence de mouvement dans son champ de vision. Il envoie un signal numérique haut ou bas selon qu'un mouvement est détecté (0 ou 1).

DHT Sensor : Connecté sur un port numérique D26. Mesure la température et l'humidité ambiantes et communique ces valeurs via un protocole numérique. Les valeurs mesurées sont en degré Celsius pour la température et en pourcentage pour l'humidité.

Air Quality Sensor : Connecté sur un port I2C. Analyse la qualité de l'air en détectant les composés organiques volatiles (TVoC) et le CO2 ambiant. L'unité de ces mesures est le « ppm » (parties par millions).

Button LED : Connecté sur un port numérique D5. Composant interactif utilisé pour capter des commandes utilisateur. La valeur récupérée permet de connaître l'état du bouton : enfoncé ou relâché (0 ou 1).

Emetteurs (Vert – cf. Figure 3) :

LED Stick RGB : Connecté sur un port PWN (port 10,12) permettant d'adapter la tension en fonction de l'intensité des LED que l'on veut. Une bande de LEDs RGB pouvant afficher des couleurs variables.

Buzzer : Connecté sur un port numérique D22. Émet des signaux sonores pour alerter ou signaler une activité.

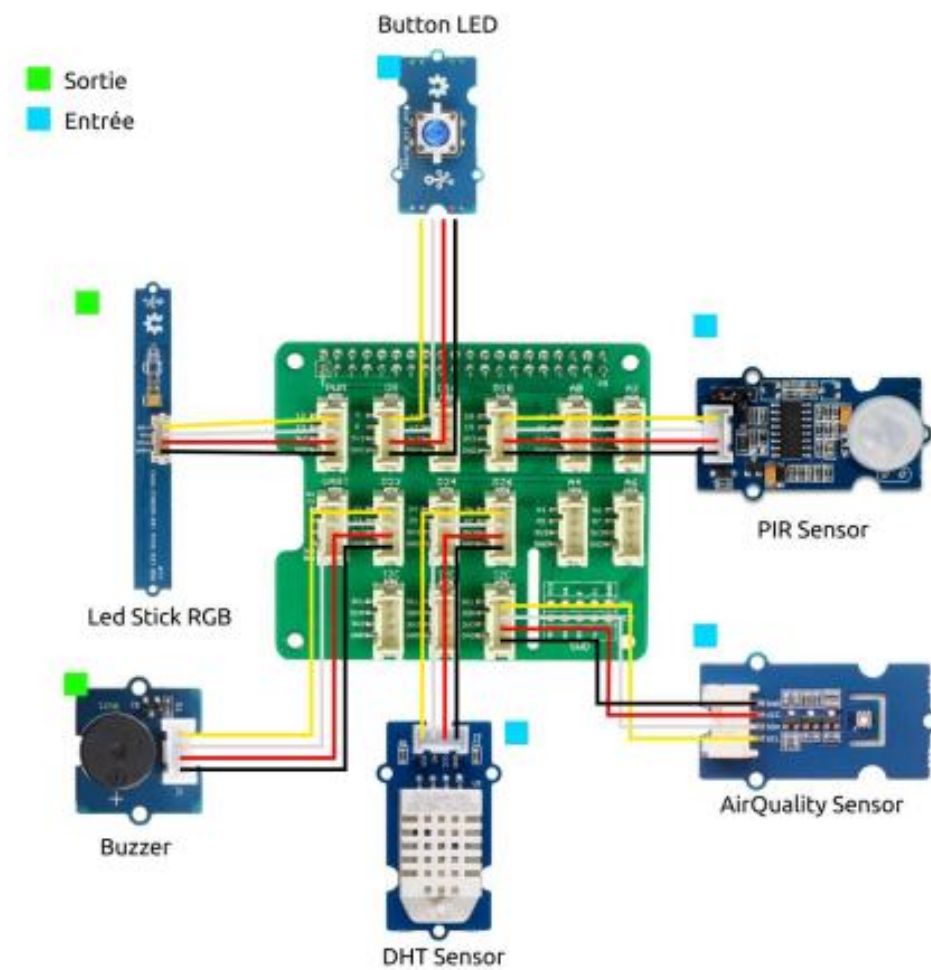


Figure 3

Nous avons eu l'opportunité de rajouter dans la suite de notre projet une caméra pour réaliser des captures d'images de l'environnement du Raspberry PI. Nous avons pu utiliser la caméra suivante : Module Caméra Okdo Raspberry Pi, CSI-2, 1 canal (cf. Figure 4). Les images capturées étaient enregistrées au format « jpg ».



Figure 4

En plus de ces divers modules que nous avons connectés au Raspberry, nous avons aussi pu nous servir du Bluetooth intégré afin d'effectuer des scans pour identifier le nombre d'appareils environnants.

3.2.3. Architecture logicielle

L'architecture logicielle déployée sur le Raspberry PI est relativement simple. Dans un dossier contenant nos scripts, nous avons ajouté deux sous-dossiers : « tests » et « lib ». Dans le dossier « lib », il est possible de retrouver l'intégralité des packages installés et nécessaires au bon fonctionnement de notre code. Dans le dossier « tests » se trouvent tous nos scripts de tests (cf. 3.1.1.).

Sur la station de collecte, seul trois scripts sont véritablement utilisés lors du fonctionnement usuel de la station : « main.py », « gateway.py » et « LastDataDf.py ».

Seul le script « main.py » est appelé lors de l'exécution du programme. Il se sert du script « gateway.py » afin de se connecter aux capteurs, récupérer leurs valeurs, les traiter à l'aide notamment du troisième script « LastDataDf.py », avant de mettre à jour les sorties, comme les LED par exemple, et d'envoyer les données à la station de traitement.

3.2.3. Station de collecte : premiers traitements

Afin de bien comprendre les opérations réalisées par la station de traitement et le rôle des différents scripts implémentés sur la station, voici quelques explications détaillées :

- **gateway.py** : c'est le script principal du système. Il contient le code de la classe Python: Gateway. Cette classe permet, quand elle est instanciée, de se connecter avec du code aux capteurs connectés physiquement au Raspberry PI. En faisant cela, il est possible d'accéder aux valeurs captées dans la méthode de classe « *inputUpdate* ». Cette méthode permet d'accéder à l'état du système au moment de l'appel de la méthode. Ensuite, il est possible de traiter les données récupérées avec la méthode de classe « *inputProcessing* ». Ces traitements peuvent être des calculs ou l'écriture dans un fichier annexe par exemple. Dans notre cas, cette méthode de classe nous permet de calculer des moyennes sur les valeurs captées sur les 20 dernières secondes. Ce traitement va nous servir ensuite pour gérer les outputs. Vient ensuite une troisième

méthode classe, intitulée « graph » qui permet de gérer le graphe d'état. Il s'agit ici de faire le point sur les états du système. Un état peut être assimilé à un stade du système, ces stades vont, dans notre code, modifier le fonctionnement du Raspberry PI et plus particulièrement du traitement des inputs et de leurs rendus. Dans notre code, il y a deux états possibles pour le système :

- état 1 – Fonctionnement habituel : lorsque l'IUT (atelier) est ouvert, l'état du système est l'état 1. On va récupérer les données captées, puis calculer une moyenne sur les données des 20 dernières secondes avant de redéfinir la couleur des LED en fonctions des moyennes calculées (voir méthode « *outputProcessing* » & « *outputUpdate* »). De plus, si les moyennes calculées sont au-dessus de seuil sanitaires ou que les valeurs enregistrées sont significatives d'un danger, une alarme se met en place avec l'activation du clignotement rouge de deux des LED et d'un bip continu du buzzer lors de l'alerte.

- état 2 – Fonctionnement nocturne de la station : quand l'IUT est fermé, c'est dans cet état qu'est le système. Dans cet état, la station sert de détecteur d'intrusion, d'alerte incendie, et d'alerte de fuite de gaz (présence de gaz dangereux pour l'homme). Il s'agit plus de collecter des données sur les horaires des différentes alertes plutôt que de véritablement avertir (là où des alarmes spécialisées seront beaucoup plus efficaces – et accessoirement réglementaires). Ce deuxième état est activable par l'utilisateur via un bouton de la station, s'il n'est pas déclenché manuellement, il se met en place en fonction des horaires de fermeture de l'IUT. De même, il est possible de rebasculer sur le premier état quand ce bouton est appuyé.

On a ensuite les méthodes de classe « *outputProcessing* » et « *outputUpdate* » qui permettent de mettre à jour les outputs de la station (*LEDstick* et *buzzer*) en fonction des traitements et des inputs. La première méthode concerne l'aspect logique et calculatoire, et la seconde permet la mise à jour technique des composants. C'est dans la première méthode que nous écrivons les données calculées dans un fichier annexe avec une granularité de 3 minutes.

Une méthode de classe nommée « alerte » est aussi implémentée. Elle permet de concrétiser les alertes avec les clignotements et les bips du buzzer.

- **main.py** : ce script est celui qui est exécuté lorsque la station est en marche. Il permet d'instancier la classe Gateway, et d'appeler dans une boucle infinie, de manière successive, les méthodes : *inputUpdate*, *inputProcessing*, *graph*, *outputProcessing*, *outputUpdate*. Ce script permet la répétition de la mise à jour de l'état du système et la collecte de données en temps réel.
- **LastDataDf.py** : cette classe permet de faciliter le traitement. Il s'agit ici d'utiliser la bibliothèque Python nommée Pandas. Elle permet d'utiliser l'objet « Dataframe » (tableau de données) au sein de Python et de faciliter les calculs réalisés sur ces objets. Le fait de créer une classe permet de réutiliser le code : pour le calcul des moyennes des 20 dernières secondes – afin de mettre à jour les LED – ou bien sur autre granularité, plus grande (1 ou 3 minutes par exemple) – pour l'écriture dans le fichier .csv annexe.

Note : chacun de ces scripts est lui-même documenté avec de nombreux commentaires afin de faciliter la compréhension et la maintenance du système.

3.3. Traitement et intégration de données

3.3.1. Environnement Virtuel

Pour déployer notre station de traitement, nous avons créé un environnement virtuel Python sur une machine Windows. L'idée est d'isoler notre environnement de développement, notamment la version de Python utilisée et les packages installés sur cet environnement, du reste de nos travaux et projet. La version de Python que nous utilisons sur notre station de collecte est la 3.12.

Pour le traitement de nos images, nous souhaitons utiliser DeepFace, c'est une bibliothèque Python permettant de faire de la reconnaissance faciale. Cependant cette bibliothèque se base en partie sur TensorFlow, une autre bibliothèque Python. Cette dernière n'est pas supportée sur Python 3.13.

Voici une liste d'étape et de commandes permettant l'installation et le déploiement de l'environnement virtuel souhaité :

1. Installation de Python 3.12 (compatibilité DeepFace)
2. Check des versions de Python installées : `py -0` (si nécessaire)
3. Se placer dans le dossier où l'on souhaite créer l'environnement virtuel puis taper la commande suivante dans le terminal : **`cd C:/environnements_Python`**
4. Création d'un « venv » (environnement virtuel) Python avec la commande suivante : **`py -3.12 -m venv deepface_env`**
5. Activation de l'environnement virtuel avec cette commande : **`deepface_env\Scripts\activate`**
6. Installation des packages nécessaires avec la commande d'installation : **`pip install ...`**
 - deepface opencv-Python
 - tf-keras (complète l'installation de DeepFace)
 - pillow
 - paho-mqtt
 - psycpg2
7. Désactivation de l'environnement en tapant la ligne de commande : **`deactivate`** (si nécessaire)

3.3.2. Envoi des données en temps réel

Dans le cadre de ce projet, l'intégration de données en temps réel fut la meilleure solution. En effet, il aurait tout à fait été possible d'utiliser des moyens classiques d'intégration de données (planification, ETL, code, ...), mais nous avons opté pour des outils permettant de faire de l'intégration de données en continu. Ce choix est motivé par le fait que nous avons besoin de traiter et d'analyser les données captées directement, sans attendre une intégration quotidienne par exemple. De plus, le domaine de l'IoT est intimement lié à des problématiques d'intégration continue. En effet, l'un des intérêts principaux de l'IoT est justement de pouvoir traiter et analyser les données en continu. C'est pourquoi nous avons opté pour l'utilisation d'outil permettant de faire de l'intégration continue.

Parmi les solutions existantes, deux outils en particulier nous ont été présentés : le système de Queuing MQTT (*Message Queuing Telemetry Transport*) et Kafka.

Système de Queuing : MQTT

MQTT (*Message Queuing Telemetry Transport*) est un protocole de messagerie basé sur le protocole TCP/IP. Il utilise une architecture *publish-subscribe*. Un serveur, appelé *broker* va héberger des « files » de messages, appelés *Topic*. Des « clients » vont pouvoir venir se connecter au *broker* pour échanger des informations avec lui. Les clients peuvent publier un message sur un ou plusieurs *Topic*, ou bien être abonnés à un ou plusieurs *Topic* afin de *consommer* (lire), les messages envoyés sur le *Topic* (cf. Figure 5).

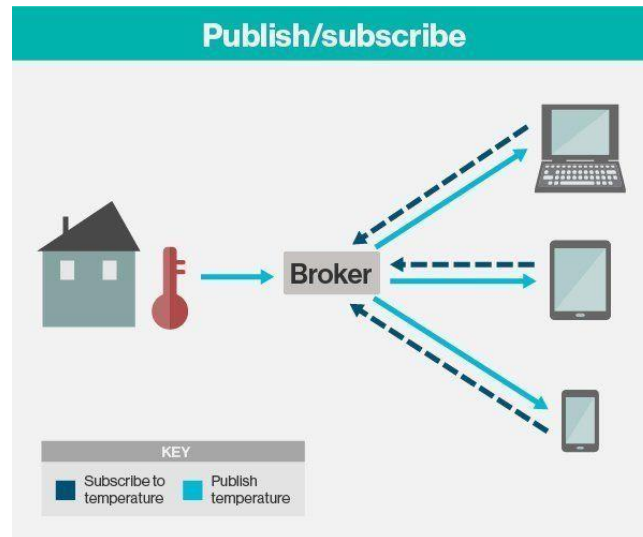


Figure 5

La particularité de ce système est que, une fois que tous les abonnés (*subscribers*) ont consommé le message, celui-ci disparaît du *Topic* et n'est plus récupérable ni lisible par de futurs abonnés.

Ce système est peu énergivore et est idéal pour les appareils avec des ressources limitées comme les nano ordinateurs ou les systèmes embarqués par exemple.

Kafka

Apache Kafka est une solution plus récente en matière d'intégration continue. Le déploiement de Kafka nécessite cependant plus de ressources qu'un système de Queuing classique. Si son fonctionnement ressemble à un système *publish-subscribe*, il y a cependant quelques particularités.

Premièrement, les messages sont persistants, c'est-à-dire qu'ils sont sauvegardés par Kafka et ne sont pas effacés après avoir été consommés par l'ensemble des abonnés. Cette notion de persistance est très importante car elle permet à Kafka d'offrir un nouveau type de solution en terme d'intégration continue, se démarquant ainsi d'un système de *Queuing* classique.

Ensuite, il faut retenir que Kafka, contrairement aux système de *Queuing*, utilise la « scalabilité ». C'est une logique similaire au dicton « diviser pour mieux régner ». Plutôt que de stocker l'ensemble des messages dans un seul et même *Topic*, il est possible de mettre en place un système de « partitions ». Chaque partition va stocker physiquement les messages associés à un groupe d'abonnés du *Topic*, identifiés avec des clés de partitionnement. Le *Topic* Kafka va regrouper tous les messages de toutes les partitions. L'idée derrière ce partitionnement est le gain de performance : en découpant les messages des *Topics* en partitions, il devient plus facile pour Kafka de gérer l'ensemble des messages des *Topics*, car

le système est distribué, il peut donc mobiliser ou répartir des ressources physiques selon ses besoins pour gagner en performances.

Choix et implémentation

Dans le cadre du projet, nous avons choisi de d'abord implémenter MQTT avec le package Python « paho-mqtt ». Ce package permet de mettre en place un *broker MQTT* ainsi que des clients qui vont venir se connecter au *broker* pour publier et consommer les messages.

L'implémentation de MQTT nous a semblé plus simple que Kafka dans un premier temps, et nous a permis de nous familiariser avec l'implémentation et le déploiement d'un système *publish-subscribe* en Python. Cependant, nous avons aussi pu implémenter la solution Apache Kafka avec le package Python « kafka-Python ». Nous avons réussi à développer des scripts de tests fonctionnels mais n'avons pas implémenté Kafka dans notre système par manque de temps.

Note : cette solution technique nous a été présentée tardivement par nos enseignants.

Lors de l'implémentation de ces solutions d'intégration de données, l'un des obstacles rencontré fut la variabilité de l'adresse IP des appareils sur un réseau WIFI. En effet, à chaque fois qu'un appareil se connecte à un réseau WIFI, il se voit attribuer une adresse IP unique et dynamique, qui change à chaque connexion de l'appareil au dit réseau. Pour se connecter aux différents *Topic* présents sur le *broker MQTT*, nous devons utiliser son adresse IP.

Mais si l'adresse IP du *broker* est variable, comment s'assurer de se connecter à la bonne adresse IP avec nos clients MQTT ?

Une solution envisageable serait de faire un scan du réseau afin d'obtenir la liste de toutes les adresses IP, mais il reste cependant un problème : quelle adresse IP correspond au *broker* ?

Afin de nous éviter ces problèmes de scan et d'identification, nous avons choisis une autre solution : rendre l'adresse IP du *broker MQTT* fixe dans le réseau. Il s'agit de faire en sorte que, dès que le *broker* se connecte au réseau, il se voit attribuer tout le temps la même adresse IP fixe, qui lui restera dédiée pour chaque future connexion au réseau. Ainsi, il devient possible de connecter les clients MQTT à cette adresse afin de publier ou de consommer des messages.

Afin de transférer de façon optimale nos données, nous avons choisi de diviser nos *Topics*. En effet, publier tous nos messages sur le même *Topic* aurait apporté beaucoup de confusion et ne nous aurait pas permis de transférer correctement nos données. En effet, le choix de division des *Topics* est intimement lié à la structure de la base de données. La concentration des messages sur un seul et unique *Topic* n'est pas une bonne solution. L'intégration de données dans la base ne serait que plus difficile et confuse. Pour éviter cela, nous avons choisi de diviser nos *Topics* pour suivre la logique suivante : un *Topic* correspond à une table dans notre base de données. Cette logique s'applique avec une station de collecte. Cependant, lorsque nous avons déployés plusieurs stations de collecte, nous avons choisi de suivre la logique suivante : une station publie sur autant de *Topics* qu'il y a de tables dans la base de données et chaque station à un ensemble de *Topics* qui lui sont propres (cf. Figure 6).

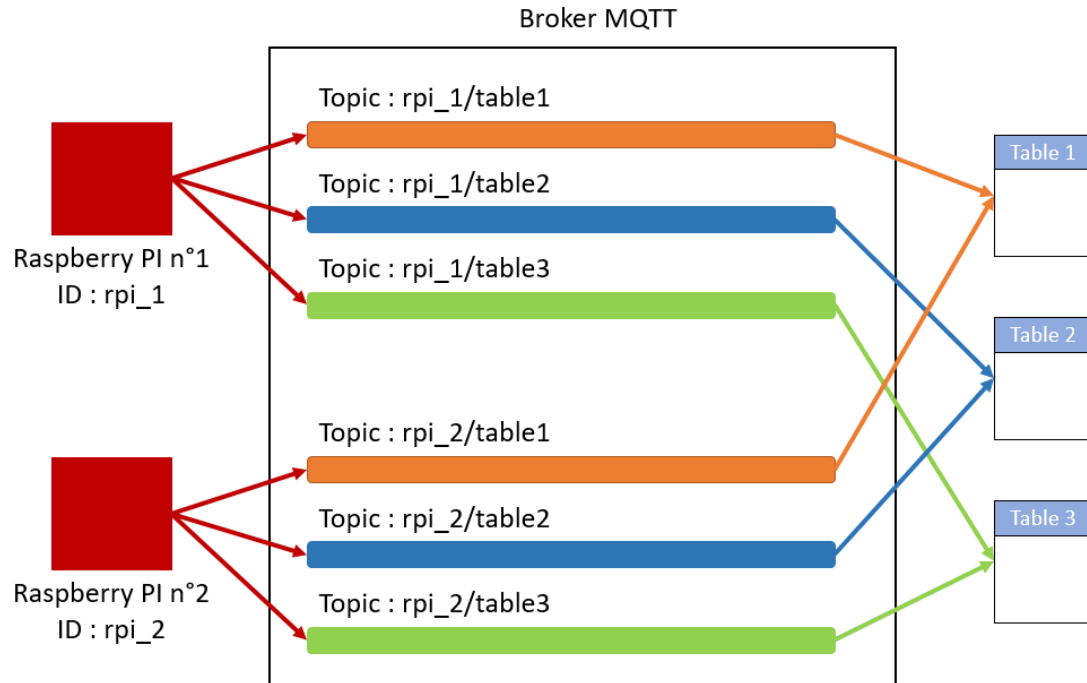


Figure 6

Cette organisation des *Topics* facilite ainsi l'intégration de données et l'exécution de traitements spécifiques, pour les images par exemple.

3.3.3. Traitement des images

Sur le Raspberry PI, l'un des capteurs est une caméra. Nous avons pu implémenter un code pour prendre des photos à intervalle régulier afin de les analyser. L'une des problématiques liées à la gestion des images fut leur transfert. Avec le package Python « paho-mqtt » présenté précédemment, il nous était impossible de transférer directement l'image en *.jpg*. La solution que nous avons implémentée fut de transformer notre image *.jpg* en un format Python : le *bytearray* (littéralement « tableau de bytes », cf. Figure 7). Ce tableau a ensuite pu être envoyé via le système de Queuing, avant d'être récupéré et reconverti en image par la station de traitement grâce au package Python « pillow ».

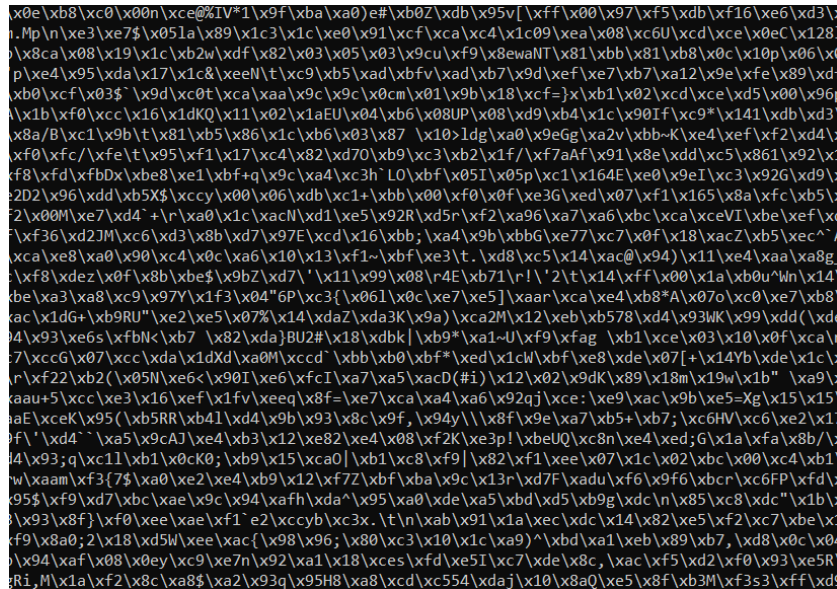


Figure 7

Afin d'analyser les images, nous avons utilisé le package Python spécialisé dans la reconnaissance faciale : DeepFace. Nous avons pu utiliser l'outil DeepFace dans le cadre d'un projet précédent de reconnaissance faciale. S'il est possible de faire de la reconnaissance faciale, il est aussi possible de simplement compter le nombre de visages sur une image (cf. Figure 8), c'est cette fonctionnalité en particulier que nous avons utilisé. Notre idée principale est d'utiliser DeepFace pour compter le nombre de visages présents sur une photographie de la pièce où est installée la station de collecte. Le nombre de visages identifiés permet d'avoir une estimation de la fréquentation et viens apporter une information complémentaire pour l'estimation de la fréquentation.



Figure 8

3.3.4. Intégration en base de données

Comme pour le traitement des images, nous avons pu récupérer l'ensemble des données transférées depuis la station de collecte vers la station de traitement avec la consommation de messages des différents *Topics MQTT*. Si le traitement des images s'est bien fait dans la station de traitement, le reste des données a pu être traité en amont directement dans la station de collecte (cf. 3.2.3.). Après avoir récupéré l'ensemble des données et traité les images, il nous fallait donc intégrer les données dans une base de données afin de les rendre disponibles pour nos tableaux de bord.

Nous avons choisi de passer par une base de données PostgreSQL pour stocker nos données. Ce choix a principalement été motivé par le fait que nous n'avions jamais utilisé cet outil en cours bien que nous connaissions déjà des outils similaires (MariaDB et MySQL). J'ai pensé qu'il serait intéressant d'apprendre à manipuler un outil largement utilisé sur le marché du travail (PostgreSQL est l'un des SGBD (Système de Gestion de Base de Données) les plus présents dans les offres d'emplois en Science des Données). Enfin, choisir un SGBD SQL est une solution optimale pour la création de tableau de bord en temps réel avec Power BI, là où l'utilisation d'une base de données NoSQL peut amener quelques complications.

Pour nous connecter à cette base de données depuis Python, nous avons pu utiliser le package : Psycopg2.

Cependant, l'utilisation d'une base de données NoSQL a aussi ses avantages. En effet, passer par MongoDB, une base de données NoSQL orientée document est aussi une solution viable en fonction de notre problématique. Dans le contexte de l'IoT, nous sommes souvent confrontés à des situations de BigData, dû à un volume et une variété très importante de données. Or l'utilisation d'une base de données MongoDB devient ici pertinent notamment grâce à l'une de ses caractéristiques : elle est distribuée, c'est-à-dire que la base de données est hébergée sur un *cluster* (un groupe) de machines. Les bases de données SQL peuvent aussi offrir cette caractéristique. Mais Mongo DB est une base de données orientée document, c'est-à-dire qu'elle permet une plus grande flexibilité sur la structure des données. Cela est un avantage non négligeable dans notre projet car nous pourrions imaginer rajouter des capteurs sur notre station de collecte ou implémenter de nouveaux traitements, amenant à l'ajout de champs dans nos tables de données. Cette modification serait permise par Mongo DB dû à sa flexibilité mais impossible avec un SGBD SQL comme PostgreSQL. L'inconvénient d'une base de données Mongo DB est l'impossibilité de créer des tableaux de bord en temps réel avec Power BI, notre outil de visualisation. Cependant, il reste possible, soit d'utiliser de nouveaux outils de visualisations, soit de se limiter à l'utilisation de cette base de données NoSQL pour répondre à des problématiques sur le long terme, ne nécessitant pas la mise à jours en temps réel du tableau de bord.

Difficultés rencontrées

Au cours du développement, nous avons rencontré un problème : l'hébergement de base de données, sur un serveur, est payant. Pour contourner ce problème, l'I.U.T. nous a proposé un serveur sur lequel il est possible d'héberger nos bases de données, à condition d'être connecté sur le réseau de l'université. Le problème devient alors une contrainte technique : notre transfert de données se fait sur un réseau privé, indépendant du réseau de l'université, nous ne pouvons donc pas accéder au services d'hébergement de base de données de l'I.U.T. même en passant par un VPN. Pour surpasser cette contrainte et pour pouvoir continuer notre travail, nous avons choisi de déployer notre base de données en local et de réaliser nos tableaux de bord en local. Mais cela reste une solution limitante, notamment si nous souhaitons déployer notre projet sur un ensemble de stations de monitoring indépendantes.

Dans un contexte professionnel, il serait possible pour une entreprise d'héberger elle-même une base de données sur un serveur de son réseau local, ou de payer pour héberger sa base sur un serveur externe.

3.4. Tableau de bord

3.4.1. Outil utilisé : Power BI

Afin de créer nos tableaux de bord, nous avons choisi de passer par l'outil Power BI. Les raisons qui ont motivé ce choix sont :

- La découverte d'un nouvel outil : bien que nous avons reçu des cours sur l'outil similaire Tableau, nous n'avons jamais utilisé Power BI dans un projet.
- La possibilité de créer des tableaux de bord sur divers supports. En effet, selon les problématiques, les tableaux de bord peuvent être créés pour un ordinateur, une tablette, ou un smartphone par exemple. Cette flexibilité sur le format est très importante pour la réalisation de notre projet.
- La mise à jour en temps réel des tableaux de bord à partir de la base de données. Power BI permet la mise à jours de nos tableaux de bord à l'intégration de nouvelles données dans notre base de données, ce qui est essentiel si nous souhaitons faire du monitoring en temps réel.

3.4.2. Valorisation des données

Afin de répondre à diverses problématiques professionnelles, nous avons envisagés plusieurs tableaux de bord. La construction de ces tableaux de bord a nécessité une étape de réflexion préalable très importante. En effet, il est facile pour un technicien de collecter des données, les traiter et les stocker, cependant réussir à leur donner du contexte avec des analyses pertinentes est beaucoup plus difficile. Nous avons donc dû mener une réflexion sur la contextualisation et la valorisation de nos données avant d'aboutir à la création de nos tableaux de bord.

Un autre élément important est l'accès à notre/nos bases de données. En effet, tous ces tableaux de bord ont besoin d'accéder à des données pour afficher leurs indicateurs. Si des visualisations restent dans l'infrastructure comme le premier et le troisième tableau de bord, d'autre sont extérieures et nécessitent un accès à la base de données en réseau. Etant donné que l'I.U.T. et plus généralement l'université, ont une politique particulière en terme de réseau et d'accès aux services, il faut faire attention à ces problématiques si les données doivent être rendues accessibles depuis l'extérieur, par exemple via une application.

Tableau de bord 1

Le premier tableau de bord que nous avons voulu réaliser est inspiré des appareils installés dans certaines salles de classe pendant la pandémie pour répondre à un besoin sanitaire. Le but est de surveiller, en temps réel, la température, le taux d'humidité, le taux de CO2 et de particules dans les salles afin de répondre à une problématique sanitaire : l'ouverture des fenêtres. En effet, aérer une salle permet de rétablir des seuils « normaux », notamment pour le taux de CO2 et de particule afin de limiter la propagation de virus mais aussi pour rendre l'environnement de travail plus sain.

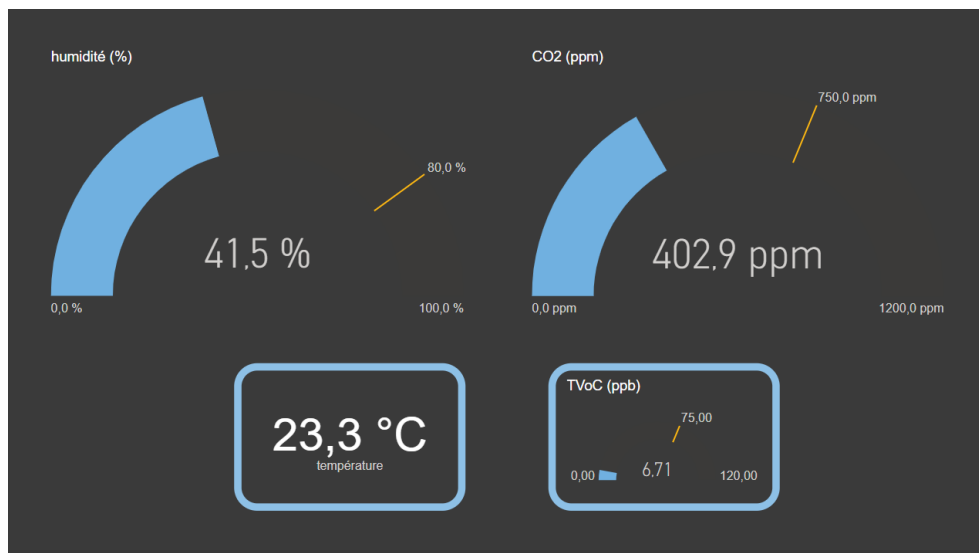


Figure 9 – Premier tableau de bord

Pour améliorer ce tableau de bord, il pourrait être intéressant de connecter notre tableau de bord aux données captées en temps réel, depuis la station de collecte et ne pas se servir de la base de données contenant les données transférées toutes les minutes. Une autre solution pourrait être de réduire la granularité des données, en passant de 1 minute à 30 secondes par exemple.

Tableau de bord 2

Le second tableau de bord que nous avons imaginé permettrait de suivre le niveau de fréquentation des salles dans lesquelles sont installées les stations de collectes. L'idée serait de fournir aux étudiants, une application ou une extension de l'application de l'université leur permettant de connaître le niveau de fréquentation estimé dans chacune des salles. L'objectif serait de permettre aux étudiants d'identifier les salles disponibles afin de trouver rapidement un espace de travail pour travailler en autonomie.

The mobile app interface shows a list of empty rooms and a table of occupancy rates.

Salles vides

Salle

Taux de fréquentation

Salle	Fréquentation	Personne
rpi_3	Fréquentation faible	1,00
rpi_4	Fréquentation faible	1,00

Figure 10 – Aperçu mobile

Pour ce tableau de bord, Power BI n'est peut-être pas l'outil le plus pertinent. En effet, Power BI n'est pas un outil de développement et ne permettrait donc pas de créer une application ou une extension. Utiliser des outils de développement serait plus pertinent.

Tableau de bord 3

Le dernier tableau de bord que nous avons imaginé permettrait d'avoir un suivi sur le long terme de nombreux indicateurs. Il ne serait cependant réalisable qu'avec le déploiement de multiples stations de collecte dans une infrastructure. Il s'agirait de fournir, sur le long terme, un suivi de tous les indicateurs possibles, captés et calculés, afin de répondre à un ensemble de problématiques professionnelles. En effet, ce tableau de bord pourrait servir à identifier et suivre les événements : détection d'intrusion, départ d'incendie, ... et potentiellement d'identifier des causes de ces événements. Une autre application possible serait la mesure du niveau de fréquentation des salles de l'infrastructure afin de repérer les points d'intérêts, les salles les plus prisées par les étudiants par exemple. Une autre application serait de guider des travaux de rénovation. En fournissant des données, sur le long terme, sur l'ensemble de l'infrastructure, il serait possible d'identifier des situations anormales comme une fuite ou une mauvaise isolation (chute de température anormale ou augmentation anormale du taux d'humidité par exemple).

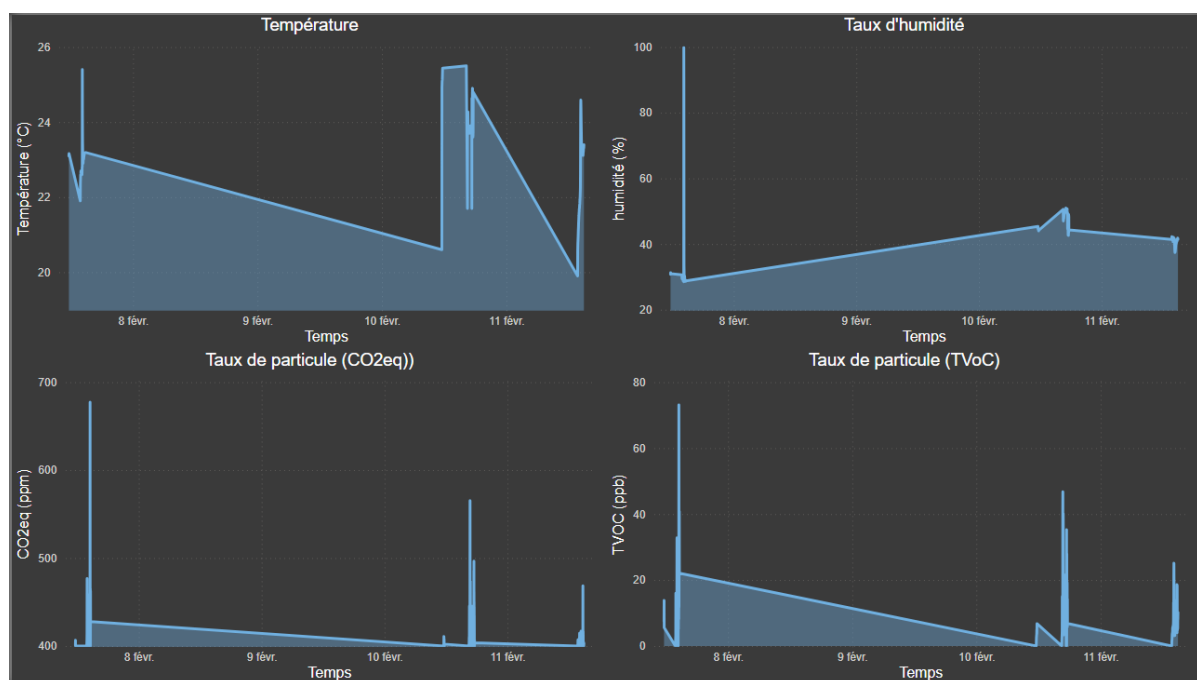


Figure 11 – Prototype de station de supervision

Ce tableau de bord est encore au stade expérimental. Nous avons réussi à déployer notre code sur plusieurs stations de collecte en simultanément pour proposer des tableaux de bord variés. Cependant, la mise en place du système accompagnant ce tableau de bord est plus complexe et nécessite plus de temps. De plus ce tableau de bord repose sur beaucoup de calculs et d'estimations (la fréquentation par exemple) qui se doivent d'être précis, ce qui n'est pas encore le cas (cf. 4.2.).

4. Pistes d'amélioration

4.1. Matérielles

La première piste d'amélioration serait bien entendu la mise en place et le déploiement de notre système dans les salles de l'I.U.T. Nous pourrions envisager de collaborer avec le département GMP (Génie Mécanique et Productique) afin de construire des boîtiers adaptés aux Raspberry PI et aux capteurs, directement installables dans les salles de cours.

4.2. Logicielles

Il existe de nombreuses pistes d'amélioration à notre projet d'un point de vue technique.

La première concerne les scans Bluetooth. Nous avons réussi à implémenter un code fonctionnel permettant de scanner les appareils Bluetooth environnants. Cependant, ce scan ne nous permettait de scanner uniquement les appareils visibles, dit « ouverts » lors du scan. Il faut savoir que la plupart des appareils (smartphones, ordinateurs, écouteurs sans fils, ...) sont automatiquement « fermés », c'est-à-dire invisibles lors d'un scan Bluetooth. Un moyen de contourner ce problème serait d'utiliser le package Python Kismet et s'inspirer du projet réalisé par Matt EDMONDSON (cf. 2.1.3) pour contourner ce problème et implémenter la solution dans notre système.

Une seconde piste d'amélioration serait de mettre en place une base de données Mongo DB pour notre troisième tableau de bord. Une base de données NoSQL, orientée document, serait plus adaptée aux problématiques de ce tableau de bord (cf. 3.3.3.). Cette base de données serait complémentaire aux solutions déjà existantes et implémentées.

Une autre piste d'amélioration serait de passer de MQTT à Kafka pour le transfert des données des stations de collecte aux stations de traitement. L'utilisation de Kafka permettrait de gagner en robustesse, de mieux répondre aux problématiques de Big Data et de permettre l'ajout de stations de traitement à volonté, en offrant la possibilité de consommer l'intégralité des messages envoyés et enregistrés sur le *Topic* Kafka.

Pour améliorer nos estimations, notamment en terme de niveau de fréquentation, il serait intéressant d'implémenter des algorithmes de classification supervisé pour calculer et estimer au mieux les indicateurs ciblés, comme le nombre de personnes présentes dans la pièce. Il serait possible, sur une station de traitement annexe d'implémenter ces algorithmes avec le package Python scikit-learn. Il serait aussi possible d'entraîner un réseau de neurones pour réaliser cette tâche en utilisant, par exemple, l'outil Ollama.

Il aurait aussi été imaginable de maximiser l'utilisation du package Python DeepFace, en implémentant de la reconnaissance faciale afin de gérer les absences et les retards des étudiants avec les images capturées par le Raspberry PI. Mais cela poserait des problèmes légaux, notamment du point de vue du RGPD, des données personnelles CNIL et du consentement des étudiants.

4.3. Organisationnelles

Enfin, d'un point de vu organisationnel, nous aurions pu mieux gérer notre temps. Dû à la nouveauté de ce projet dans l'I.U.T., nous avons suivi les enseignements liés à ces SAE sans avoir la sensation de participer à la mise en place d'un projet global plutôt que de petits projets isolés.

Nous aurions pu aussi nous investir davantage en accédant au matériel et aux outils en autonomie plus tôt dans le projet. Cela nous aurait par exemple permis de mettre en place certaine pistes d'amélioration citées plus tôt.

5. Bilan

Ce projet Getenv. fut globalement une réussite. Nous avons su nous réapproprier les enjeux et problématiques des domaines ciblés dans ce projet, à savoir l'IoT et le Big Data.

La veille technique que j'ai pu mener m'a permis de découvrir de nouveaux acteurs et enjeux mais aussi d'autres canaux d'informations, chaînes YouTube ou médias, que je ne connaissais pas afin de m'informer sur les progrès, les actualités et les dernières innovations.

Si nous avons pu découvrir une panoplie d'outils et de solutions, en cours ou personnellement, nous avons aussi su déployer certaines de ces solutions après avoir mené une réflexion sur les outils à utiliser.

Techniquement, le projet fut très enrichissant. Découvrir les nano ordinateurs et avoir l'opportunité de manipuler et utiliser de nouveaux outils, comme Linux, Kafka, MQTT, PostgreSQL ou encore Power BI, dans un contexte professionnel est une chance dont nous avons su tirer parti.

Il reste cependant des points de frustration, j'aurais personnellement aimé pouvoir finir d'implémenter l'ensemble des idées et des pistes d'améliorations envisagées afin de mener le projet à son aboutissement.

6. Bibliographie

6.1. Référentiel abréviation

SAE : Situation d'apprentissage et d'évaluation

UE : Union Européenne

IoT : *Internet of Things* (Internet des objets)

MQTT : *Message Queuing Transfert Telemetry*

CNIL

RGPD : Règlement Général sur la Protection des Données

6.2. Veille technique

Liens vers les recherches personnelles, les projets existant, les conférences et articles traitant de sujet en liens avec le projet (voir travail de doc préliminaire + Kismet Raspberry PI – vérifier si l'on est suivi)

6.2.1. Webinaire DataGrandEst

Webinaire DataGrandEst : Les capteurs et données en temps réel

<https://www.youtube.com/watch?v=hGrhP5KnqZc>

Data GrandEst :

<https://www.data.gouv.fr/fr/organizations/datagrandest/>

StrasApp :

<https://data.strasbourg.eu/pages/accueil/>

<https://data.strasbourg.eu/pages/accueil/>

Atmo GrandEst :

<https://www.atmo-grandest.eu/>

<https://www.insee.fr/fr/metadonnees/definition/c1523#>

VigieCrues :

<https://www.vigicrues.gouv.fr/>

<https://hydro.eaufrance.fr/>

Les objets connectés :

<https://iotjourney.orange.com/fr-FR/explorer/les-solutions-iot/objet-connecte>

<https://iotjourney.orange.com/fr-FR/explorer/les-solutions-iot/donnees-des-objets>

<https://www.digitsole.com/>

<https://www.supplychaininfo.eu/dossier-optimisation-logistique/comment-objets-connectes-permettent-optimisation-logistique/>

<https://www.aguram.org/aguram/>

<https://smart-city.eco/>

6.2.2. Conférences Devovx

Kit de survie pour l'IoT façon DIY

<https://www.youtube.com/watch?v=IMzgrpLqZmq>

Premiers pas avec un microcontrôleur et Google Cloud IoT Core

<https://www.youtube.com/watch?v=eYDb0X70HGM>

La sécurité dans l'IoT difficultés, failles et contre-mesures

<https://www.youtube.com/watch?v=ThdFcdyqMik>

IOT, timeseries and prediction with Android, Cassandra and Spark by Amira LAKHAL

<https://www.youtube.com/watch?v=1DV9Kdec0-A>

Le Big Data, de la récolte jusqu'à la production à grande échelle !

<https://www.youtube.com/watch?v=uqajB-7ghVM>

Géopolitique de la data

<https://www.youtube.com/watch?v=EOOhYaGGArc>

6.2.3. Recherches personnelles

Flux vision – Orange

<https://www.orange-business.com/fr/solutions/data-intelligence-iot/flux-vision>

<https://www.youtube.com/watch?v=Y0OooJtNYsI>

Projet 'chasing your tail' : Kismet – Matt EDMONDSON

<https://raspberrypi.developpez.com/actu/335858/Un-outil-anti-pistage-alimente-par-Raspberry-Pi-recemment-concu-permet-de-verifier-si-vous-etes-suivi-en-analysant-les-appareils-environnants-l-outil-serait-efficace-et-facilement-reproductible/>

<https://www.blackhat.com/us-22/briefings/schedule/#chasing-your-tail-with-a-raspberry-pi-26930>

Horizon Europe

<https://www.consilium.europa.eu/fr/policies/horizon-europe/#:~:text=%22Horizon%20Europe%22%20est%20le%20nouveau,projets%20associant%2040%20000%20organisations.>

https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-data-strategy_en

<https://digital-strategy.ec.europa.eu/fr/policies/internet-things-policy>

Industrie 4.0

<https://merca.team/entreprises-dominent-industrie-40/>

<https://www.ibm.com/fr-fr/topics/industry-4-0>

<https://www.visiativ.com/actualites/actualites/industrie-4-0-definition-et-mise-en-oeuvre-vers-lusine-de-production-connectee/>

6.3. Documentation technique

Tous les liens vers les sites officiels de documentations des différents packages. Liens vers les différents sites et forums consultés à des fins techniques.

Scan Bluetooth

Pybluez

https://github.com/scivision/pybluez-examples/blob/main/bluetooth_scan.py


https://pybluez.readthedocs.io/en/latest/api/discover_devices.html

Étape 1 : Installer les dépendances nécessaires

Avant d'installer `gattlib`, vous devez vous assurer que vous avez les dépendances requises sur votre Raspberry Pi.

Exécutez ces commandes dans un terminal pour installer les dépendances nécessaires :

bash

 Copier


```
sudo apt-get update
sudo apt-get install libbluetooth-dev
sudo apt-get install libboost-python-dev
sudo apt-get install libglib2.0-dev
```

Étape 2 : Installer `gattlib` via `pip`

Une fois les dépendances installées, vous pouvez essayer d'installer `gattlib` via `pip`. Notez que `gattlib` peut nécessiter une version spécifique de `pip` ou d'autres ajustements sur certaines plateformes, donc essayons d'abord d'installer via `pip`.

Exécutez cette commande pour installer `gattlib` :

bash

 Copier

```
pip3 install gattlib
```

Installation de `gattlib` (pour détecter les appareils bluetooth ble) :

<https://stackoverflow.com/questions/43721463/cant-install-gattlib-on-raspberry-pi-error-stray-xxx-in-program-on-usr-inc>

Packages Python :

Pandas :

https://pandas.pydata.org/docs/user_guide/index.html

Paho-mqtt :

<https://www.hivemq.com/blog/mqtt-client-library-paho-Python/>

Kafka Python :

<https://kafka-Python.readthedocs.io/en/master/>

Pillow :

<https://pypi.org/project/pillow/>

Time :

<https://docs.Python.org/3/library/time.html>

DeepFace :

<https://pypi.org/project/deepface/>

Psycopg2 :

<https://pypi.org/project/psycopg2/>