

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE  
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique  
Université des Sciences et de la Technologie HOUARI BOUMEDIENE  
Faculté Informatique



THÈSE DE LICENSE

Présentée pour l'obtention du grade de **licenciée**

**Département** : Informatique

**Spécialité** : Génie des Télécommunications et Réseaux

**Promoteur** : BENMERAR TARIK Zakaria

**Présenté par** : Hamza Lagab, Abdallah Daheur

**Thème**

**Architecture réseaux pour le suivi et controle d'une maison  
intelligente à distance**

Soutenue publiquement, le **04/06/2024**, devant le jury composé de :

M. ZAFONE YOUCEF Professeur à l'USTHB, Président  
Mme. CHAOUI KHADIDJA Professeure à l'USTHB, Membre

# dedication

À mes chers parents, Votre amour inconditionnel, votre soutien indéfectible et vos encouragements constants ont été les piliers essentiels qui ont guidé chacune de mes étapes académiques. À travers ce projet de fin d'études, je tiens à vous exprimer toute ma reconnaissance pour votre dévouement et votre sacrifice. Votre influence positive et vos valeurs m'ont inspiré à persévérer et à atteindre mes objectifs. C'est avec une profonde gratitude que je vous dédie ce travail, en espérant qu'il puisse témoigner de la fierté que je ressens d'être votre enfant.

À mon grand-père bien-aimé, bien que tu ne sois plus physiquement parmi nous, ton héritage de sagesse, de force et de persévérance continue de m'inspirer chaque jour. Tu as été un modèle de courage et de détermination, et chaque succès que j'atteins est dédié à ta mémoire. Ce travail porte la marque de ton influence sur ma vie, et je suis reconnaissant pour les valeurs que tu m'as transmises.

À vous tous, je dédie ce projet avec amour et reconnaissance, en sachant que chaque accomplissement est le fruit de votre amour, de votre soutien et de vos enseignements.

*Daheur Abdallah, Lagab Hamza*

# Remerciement

Nous tenons à exprimer notre profonde gratitude envers toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet.

Nous adressons nos plus sincères remerciements à notre encadrant, Monsieur BENME-RAR TARIK Zakaria, pour ses conseils avisés, son soutien constant et ses orientations précieuses qui ont grandement enrichi notre travail.

Nous tenons également à remercier chaleureusement tous les enseignants de l'USTHB qui ont partagé leur expertise et leur savoir-faire, contribuant ainsi à notre formation académique et professionnelle.

Enfin, nos remerciements vont à toutes les personnes qui ont apporté leur soutien, leurs encouragements et leur expertise tout au long de ce projet.

## **Abstract**

Information and telecommunication technologies have revolutionized all aspects of daily life. In particular, homes have become smarter with the integration of IoT (Internet of Things).

In this project, we focus on the specific case of a home made smart through IoT integration. By integrating both local network technologies like BLE (Bluetooth Low Energy) and wide-area network technologies such as MQTT or Web Sockets, we enable users to monitor and control their homes remotely.

**Keywords:** MQTT, Web Sockets, BLE, Embedded systems

## **Résumé**

Les technologies d'information et de télécommunication ont permis de révolutionner tous les aspects de la vie quotidienne. En particulier, les maisons ont été rendues intelligentes avec l'intégration en particulier de l'IoT (Internet of Things).

Dans ce projet, on s'intéresse au cas particulier de la maison qu'on l'a rendue intelligente avec l'intégration de l'IoT. En intégrant à la fois des technologies de réseaux locaux comme BLE (Bluetooth Low Energy) et des technologies WAN comme MQTT ou Web Sockets, on permet à l'utilisateur de suivre et de contrôler sa maison à distance.

**Mots Clés :** MQTT, Web Sockets, BLE, Systèmes Embarqués

# Table des matières

Table des figures	i
Table des figures	ii
Nomenclature	iii
Nomenclature	iv
Introduction Générale	1
<b>1 Généralité sur les systèmes IoT</b>	<b>2</b>
1.1 Introduction à l’Internet Of Things . . . . .	2
1.1.1 Définition IoT . . . . .	2
1.1.2 Définition maison intelligente . . . . .	2
1.1.3 Principaux Dispositifs IoT pour les Maisons Intelligentes . . . . .	2
1.1.4 Dispositifs de contrôle et d’automatisation . . . . .	4
1.2 Protocoles de Communication pour l’IoT . . . . .	4
1.2.1 Protocole BLE . . . . .	4
1.2.2 Protocole MQTT . . . . .	6
1.2.3 Protocole Web Sockets . . . . .	8
1.3 Systèmes Embarqués et Intégration IoT . . . . .	9
1.3.1 Qu’est-ce qu’un système embarqué ? . . . . .	9
1.3.2 Le rôle des systèmes embarqués dans l’IoT . . . . .	9
1.4 Conclusion . . . . .	10
<b>2 Architecture du Système et Méthodologie de la Solution</b>	<b>11</b>
2.1 Introduction de chapitre . . . . .	11
2.2 Description de la Solution . . . . .	11
2.3 Interface de l’utilisateur . . . . .	12
2.3.1 Application web . . . . .	12

2.3.2	Gestion des données : . . . . .	13
2.4	Protocoles de messagerie et transport . . . . .	15
2.4.1	Protocole de Messagerie . . . . .	15
2.4.2	Protocole de Transport . . . . .	19
2.4.3	Intégrité, Authentification et Confidentialité . . . . .	20
2.5	Contrôleur de la maison . . . . .	21
2.5.1	Compatibilité des dispositifs BLE avec le Contrôleur . . . . .	22
2.5.2	Interface du contrôleur et authentification . . . . .	25
2.6	Schéma séquentiel . . . . .	25
2.7	Conclusion . . . . .	27
<b>3</b>	<b>Réalisation et Mise en Œuvre de la Solution</b>	<b>28</b>
3.1	Introduction . . . . .	28
3.2	Logiciels et Application utilisés . . . . .	28
3.3	Technologies et bibliothèque utilisés . . . . .	29
3.4	Matériels utilisés . . . . .	29
3.5	Dispositifs utilisés . . . . .	30
3.5.1	Premier Dispositif : . . . . .	30
3.5.2	deuxième dispositif . . . . .	31
3.6	Scripts et configuration utilisés . . . . .	31
3.6.1	Fichier configuration MQTT et Docker . . . . .	31
3.6.2	Script pour l'Interaction avec le protocole MQTT . . . . .	34
3.6.3	Script pour l'Interaction avec les Dispositifs BLE . . . . .	37
3.7	Test et Résultat . . . . .	39
3.7.1	Évaluation du script Node.js avec @abandonware/noble . . . . .	39
3.7.2	Évaluation du script Node.js avec @MQTT.js et l'interface web : . .	40
3.8	Conclusion . . . . .	41
	<b>Conclusion Générale</b>	<b>42</b>
	<b>Bibliographie</b>	<b>43</b>
	<b>A Déploiement et installation du système d'exploitation :</b>	<b>A</b>
	<b>B Configuration Node.js :</b>	<b>C</b>

# Table des figures

1.1	Relation capteurs actionneurs . . . . .	3
1.2	flux de travail du protocole BLE . . . . .	6
1.3	Architecture MQTT de Publication / Abonnement . . . . .	8
1.4	Caractère générique hash . . . . .	8
1.5	Diagramme d'une connexion WebSocket . . . . .	9
2.1	Diagramme de flux de transport d'un message . . . . .	12
2.2	Schéma de modèle relationnel . . . . .	13
2.3	Fichier de journaux du MQTT . . . . .	16
2.4	Architecture des sujets MQTT . . . . .	16
2.5	Diagramme de séquence pour LWT . . . . .	18
2.6	Diagramme de séquence pour QOS,RETAIN . . . . .	18
2.7	Diagramme de protocole de transport . . . . .	19
2.8	MQTT Encryption, Authentification et Authorisation organigramme . . .	21
2.9	Diagramme pour le transport des données . . . . .	22
2.10	Connexion et Déconnexion entre Périphérique et Contrôleur . . . . .	23
2.11	Lecture de Données du Périphérique par le Contrôleur . . . . .	23
2.12	Écriture de Données du Contrôleur vers un Périphérique . . . . .	24
2.13	Notification de Changement de Caractéristique d'un Périphérique au Contrôleur . . . . .	24
2.14	Organigramme de authentification et l'usage de l'interface du contrôleur .	25
2.15	Diagramme d'activité de déroulement de system . . . . .	26
2.16	Diagramme séquentiel de déroulement de system . . . . .	27
3.1	Serveur de noms . . . . .	30
3.2	Profil GATT . . . . .	30
3.3	Local Name . . . . .	30
3.4	Serveur de noms . . . . .	31
3.5	Profil GATT . . . . .	31

3.6	Local Name . . . . .	31
3.7	Fichier de configuration MQTT . . . . .	32
3.8	Fichier de securité dynamique MQTT . . . . .	33
3.9	Fichier de docker compose . . . . .	33
3.10	Résultat d'exécution du script de connexion MQTT dans le fichier journal	34
3.11	Script de connexion MQTT . . . . .	35
3.12	Suit de script de connexion MQTT . . . . .	35
3.13	Script pour les fonctions de création . . . . .	36
3.14	Script pour les fonctions de mise à jour . . . . .	36
3.15	Script pour les fonctions de suppression . . . . .	37
3.16	Code en arrière-plan BLE . . . . .	37
3.17	Suite de code en arrière-plan pour BLE . . . . .	38
3.18	Suite de code en arrière-plan pour BLE . . . . .	38
3.19	Suite de code en arrière-plan pour BLE . . . . .	38
3.20	Résultat après le Balayage . . . . .	39
3.21	Appareils Connectés . . . . .	39
3.22	Détection des Changements de Valeurs . . . . .	39
3.23	Page settings dans le tableau de bord de l'utilisateur . . . . .	40
3.24	Page devices dans le tableau de bord de l'utilisateur . . . . .	40
3.25	La table des utilisateurs après la création du compte . . . . .	40
3.26	La table Devices après l'appairage avec les appareils . . . . .	41
3.27	La table des contrôleurs après l'appairage de l'ID du contrôleur avec le compte utilisateur . . . . .	41
A.1	Raspberry pi Imager . . . . .	B
A.2	Bureau Raspberry pi 4 . . . . .	B
B.1	Fichier package.json . . . . .	C
B.2	Suite du fichier package.json . . . . .	D



# Nomenclature

## Acronymes / Abréviations

ACL	Access Control List (Liste de contrôle d'accès)
AFH	Adaptive Frequency Hopping (Saut de Fréquence Adaptatif)
API	Application Programming Interface ( Interface de programmation d'application)
BLE	Bluetooth Low Energy (Bluetooth À Faible Consommation d'Énergie)
CDN	content delivery network (réseau de diffusion de contenu)
CRUD	Create Read Update Delete (Créer Lire Mettre à jour Supprimer)
DDoS	Distributed Denial-of-Service (dénî de service distribué)
EDL	Eclipse Distribution License (Licence de Distribution Eclipse)
EPL	Eclipse Public License (Licence Publique Eclipse)
HTTP	Hypertext Transfer Protocol
IDE	integrated development environment (environnement de développement intégré)
IoT	Internet Of Things (Internet Des Objets)
IP	Internet Protocol (Plateforme en tant que service)
JSON	JavaScript Object Notation (Notation Objet JavaScript)
LWT	Last Will and Testament (Dernière volonté et testament)
MIM	Man-In-the-Middle (l'homme au milieu)
MQTT	Message Queuing Telemetry Transport (Télémétrie Par Message Queuing)
NPM	Node package manager (Gestionnaire de paquets Node)
OS	Operating System (Système d'exploitation)

PaaS	Platform as a service (Plateforme en tant que service)
PKI	Public Key Infrastructure (Infrastructure à Clés Publiques)
QoS	Quality of Service levels (Niveaux de qualité de service)
SQL	Structured query language (Langage de requête structuré)
SSL	Secure Sockets Layer (Couche de sockets sécurisée)
TCP	Transmission Control Protocol (Protocole de Contrôle de Transmission)
TLS	Transport Layer Security (Sécurité de la couche de transport)
UUID	Universally Unique Identifier (Identifiant Universellement Unique)
WAN	Wide Area Network (Réseau À Grande Distance)
WEB	World Wide Web (Toile Mondiale)
WS	Web Sockets (Sockets Web)
XML	Extensible Markup Language (langage de balisage extensible)

# Introduction Générale

L'Internet des Objets Ou en anglais The Internet of Things (IOT) a révolutionné nos foyers en les transformant en environnements connectés et intelligents. Grâce à l'IoT, nos appareils interagissent, communiquent et automatisent divers aspects de notre vie quotidienne.

Dans ce document, nous explorerons en détail le rôle crucial de l'IoT dans les maisons intelligentes, en mettant l'accent sur les dispositifs, les protocoles de communication, les systèmes embarqués et les défis de sécurité. En comprenant ces éléments clés, nous serons mieux à même d'apprécier les avantages et les opportunités offerts par l'IoT dans nos foyers, tout en reconnaissant les défis qui doivent être surmontés pour garantir une utilisation sûre et efficace de cette technologie.

Ce document est structuré en trois chapitres distincts : Le premier chapitre, consacré à l'état de l'art, établit les fondements conceptuels de l'Internet des Objets (IoT) et des maisons intelligentes en définissant les termes clés et en explorant les dispositifs et protocoles essentiels. Le deuxième chapitre, axé sur la conception, se penche sur l'architecture et l'interface utilisateur des maisons intelligentes en examinant les protocoles de communication, les contrôleurs domestiques, ainsi que les schémas séquentiels. Enfin, le troisième chapitre, dédié à l'implémentation, aborde de manière pratique la mise en œuvre le système IoT, détaillant les logiciels, les technologies, le matériel, les dispositifs utilisés, ainsi que les tests et les résultats obtenus.

# Chapitre 1

## Généralité sur les systèmes IoT

### 1.1 Introduction à l'Internet Of Things

#### 1.1.1 Définition IoT

Internet Of Things (IoT) est un réseau d'objets physiques qui sont intégrés à des logiciels, des composants électroniques, un réseau de communication et des capteurs qui permettent à ces objets de collecter et d'échanger des données.

L'objectif de l'IoT est d'étendre la connectivité Internet à des objets de la vie courante (une montre, un lave-linge, mobilier urbain, compteur de gaz... etc.). [1]

#### 1.1.2 Définition maison intelligente

Une maison intelligente, est une maison qui intègre des systèmes d'automatisation avancés (technologies informatiques) pour fournir aux habitants un contrôle sur les fonctions de la maison.

Par exemple, une maison intelligente peut contrôler les opérations d'éclairage, de température, de multimédia, ainsi que de nombreuses autres fonctions. Aussi une maison intelligente est une résidence équipée de dispositifs technologiques qui observent et collectent les informations sur le résident afin de lui offrir de manière proactive, des services adaptés à ses besoins. [2]

#### 1.1.3 Principaux Dispositifs IoT pour les Maisons Intelligentes

Les dispositifs IoT (Internet Of Things) pour les maisons intelligentes sont des composants technologiques qui interagissent entre eux pour automatiser, surveiller et contrôler divers aspects de la vie domestique. Voici une liste des principaux dispositifs IdO utilisés

dans les maisons intelligentes :

### 1. Les capteurs :

Un capteur est un dispositif transformant une grandeur physique à une autre grandeur utilisable, tel qu'une tension électrique ou une intensité ; On fait souvent la confusion entre capteur et transducteur : le capteur est au minimum constitué d'un transducteur. [2]

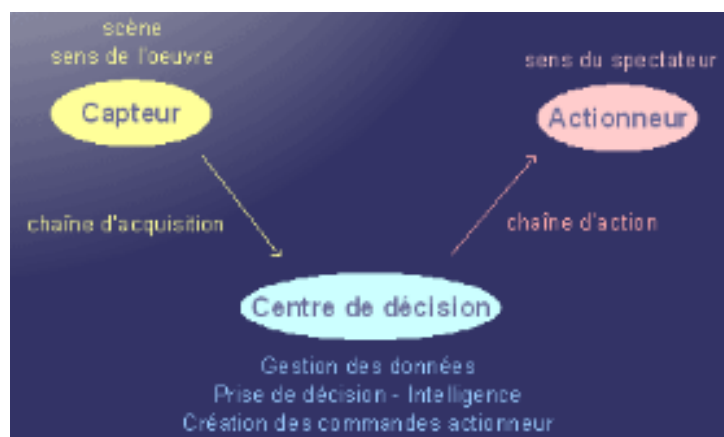
Le capteur est souvent le premier dispositif de la chaîne d'acquisition, il s'agit d'une simple interface entre un processus physique et une information manipulable.

### 2. Les Actionneurs :

Dans une machine ou un système de commande, qu'il soit semi-automatique ou automatique, un actionneur représente l'élément de la partie opérative qui, lorsqu'il reçoit une instruction de la partie commande, généralement via un capteur ou un pré-actionneur, transforme l'énergie qui lui est fournie en un travail utile pour accomplir des tâches programmées au sein d'un système automatisé. En d'autres termes, un actionneur est l'élément qui fournit la force nécessaire à l'exécution d'un travail spécifique dicté par une unité de commande. [2]

### 3. Relation capteurs actionneurs :

Les capteurs jouent le rôle d'informateurs-traducteurs ils détectent les changements dans l'environnement de la partie opérative et les transforment en informations interprétables (souvent sous forme de grandeurs électriques) par la partie commande. Ces informations sont utilisées pour contrôler les actionneurs, qui convertissent une forme d'énergie en une autre afin de produire l'énergie souhaitée, généralement de nature mécanique [2]. Cette relation est illustrée dans la Figure. 1.1 ci-dessous.



**Figure 1.1:** Relation capteurs actionneurs [3]

Des capteurs et actionneurs, on passe aux dispositifs de contrôle et d'automatisation, essentiels aux maisons intelligentes, pour une gestion efficace du domicile.

### 1.1.4 Dispositifs de contrôle et d'automatisation

Le concentrateur de maison intelligente est crucial pour contrôler à distance les dispositifs tels que les lumières et les thermostats, centralisant ainsi les applications dans une interface unique. Cela améliore l'efficacité de la maison intelligente en offrant un contrôle simplifié et global des différents aspects domestiques. Voici quelques exemples de dispositifs couramment utilisés pour contrôler et automatiser divers aspects de la maison :

- **Systèmes d'éclairage intelligents**
- **Thermostats intelligents**
- **Caméras de sécurité et systèmes de sécurité intelligents**
- **Serrures de porte intelligentes et ouvre-portes de garage intelligents**

## 1.2 Protocoles de Communication pour l'IoT

L'IoT repose sur une connectivité entre appareils intelligents et systèmes centraux, utilisant divers protocoles de communication adaptés à différents cas d'utilisation.

### 1.2.1 Protocole BLE

BLE est également connu sous le nom de Bluetooth intelligent, ce qui en fait un protocole important pour les applications IoT. Il est conçu et amélioré pour une courte portée, une faible bande passante et une faible latence pour les applications IoT. Les avantages du BLE par rapport au Bluetooth classique incluent une consommation d'énergie plus faible, un temps de configuration réduit et la prise en charge d'une topologie de réseau en étoile avec un nombre illimité de nœuds. [4]

- **Comment fonctionne le Bluetooth Low Energy ?**

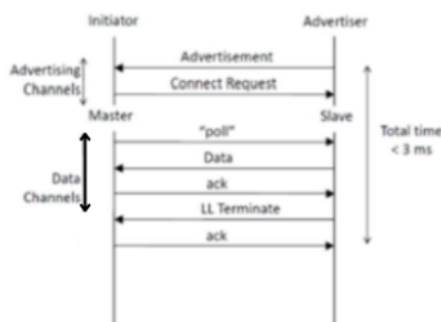
Le protocole de communication Bluetooth Low Energy repose sur un certain nombre de couches essentielles à l'échange de données dont : Generic Access Profile (GAP), Generic Attribute Protocol (GATT) et Attribute Protocol (ATT) et la couche Link Layer (LL) 1.2. [5]

1. **GAP** : Le Generic Access Profile (GAP) du Bluetooth définit les rôles des composants maîtres et esclaves, ainsi que les méthodes de signalement et la découverte du réseau. Il propose deux méthodes de signalement : l'envoi de messages de signalement et la réponse au scan du réseau. Ces messages, de format identique, comportent 31 octets de données. Seuls les messages de signalement sont obligatoires et sont envoyés à intervalles réguliers, avec une consommation d'énergie variant selon la période d'envoi.
2. **GATT** : Le Generic Attribute Profile (GATT) du Bluetooth Low Energy (BLE) facilite les échanges bidirectionnels de données entre maîtres et esclaves. Il organise les données en services, chacun avec des caractéristiques spécifiques. Les modes de fonctionnement varient entre le GAP et le GATT, ce dernier ayant des modes client et serveur. Les périphériques esclaves sont les serveurs GATT, tandis que les maîtres sont les clients GATT. Les connexions sont initiées par les clients qui peuvent demander une liste des services fournis par le serveur lors de chaque nouvelle connexion.
3. **ATT** : Le protocole d'attribut définit la communication entre deux dispositifs jouant respectivement les rôles de serveur et de client, sur un canal L2CAP (Logical Link Control Adaptation Protocol) dédié. Le protocole Attribut définit deux rôles :
  - Serveur : Le dispositif qui stocke les données sous forme d'un ou plusieurs attributs.
  - Client : Le dispositif qui collecte les informations pour un ou plusieurs serveurs.

Le client peut accéder aux attributs du serveur en envoyant des requêtes, qui déclenchent des messages de réponse du serveur. Un serveur peut envoyer à un client des messages contenant des attributs comme les notifications (Notify), lecture (Read), écriture (Write), indications (Indicate).

4. **LL** : La couche de liaison Bluetooth Low Energy (BLE) gère les opérations fondamentales telles que la publicité, le scanning et l'établissement de connexions. La publicité permet aux dispositifs de signaler leur présence et de permettre l'établissement de connexions. Le scanning permet à un dispositif d'écouter

les annonces entrantes pour découvrir et se connecter à des périphériques, ou simplement recevoir des données diffusées. L'établissement de connexion assure une transmission fiable des données grâce à des mécanismes tels que les accusés de réception et les retransmissions, ainsi que l'utilisation du saut de fréquence adaptatif (AFH) pour une connexion physique fiable. Les connexions BLE supportent également le cryptage des données pour assurer la confidentialité.



**Figure 1.2:** Etablissement de la connexion, transmission d'un paquet et fin de la connexion [6]

## 1.2.2 Protocole MQTT

MQTT est une norme OASIS pour la connectivité IoT. C'est un protocole de messagerie extrêmement simple et léger, basé sur un modèle de publication/abonnement au sujets, conçu pour les dispositifs contraints et les réseaux à faible bande passante, à latence élevée ou peu fiables. Les principes de conception visent à minimiser la bande passante du réseau et les exigences en termes de ressources des appareils tout en tentant également d'assurer la fiabilité et un certain degré d'assurance de la livraison. Ces principes se révèlent également idéaux pour le monde de l'Internet of things (IoT) des appareils connectés. [7]

- **Les composants MQTT :** MQTT implémente le modèle de publication/abonnement au sujets en définissant les clients et les agents comme ci-dessous :

1. **Client MQTT :** Un client MQTT est tout appareil, du serveur au microcontrôleur, qui exécute une bibliothèque MQTT. Si le client envoie des messages, il agit comme un éditeur, et s'il reçoit des messages, il agit comme un récepteur. Fondamentalement, tout appareil qui communique à l'aide de MQTT sur un réseau peut être appelé un appareil client MQTT.



2. **Agent MQTT (broker)** : L'agent MQTT est le système dorsal qui coordonne les messages entre les différents clients. Les responsabilités de l'agent comprennent la réception et le filtrage des messages, l'identification des clients abonnés à chaque message et l'envoi des messages à ces derniers [8]. Il est également responsable d'autres tâches telles que :

- Autoriser et authentifier les clients MQTT.
- Transmettre des messages à d'autres systèmes pour une analyse plus approfondie.
- Traiter les messages et les sessions clients manqués.
- Lorsqu'un appareil publie une nouvelle valeur sur un sujet, le courtier la publie sur le sujet spécifié pour minimiser les paquets envoyés.

— **Comment fonctionne MQTT :**

1. **Connexion MQTT** : Les clients et les agents commencent à communiquer en utilisant une connexion MQTT. Les clients initient la connexion en envoyant un message CONNECT à l'agent MQTT. L'agent confirme qu'une connexion a été établie en répondant par un message CONNACK. Les clients ne se connectent jamais entre eux, uniquement avec l'agent. [9]

2. **MQTT sujets :**

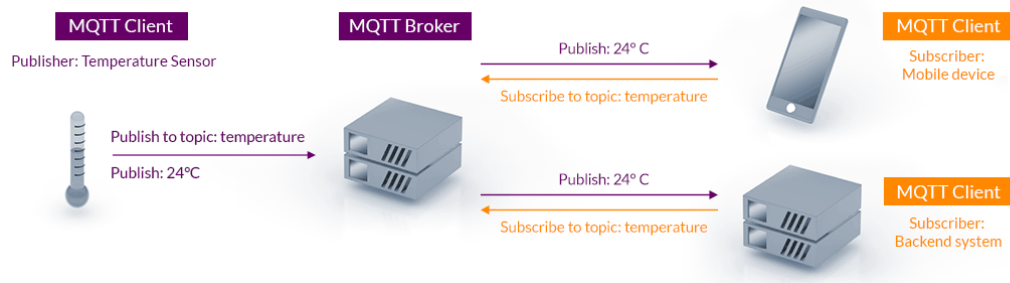
En MQTT, le terme "sujet" fait référence à une chaîne UTF-8 qui filtre les messages pour un client connecté. Un sujet se compose d'un ou plusieurs niveaux séparés par une barre oblique (séparateur de niveau de sujet). [10] Par exemple :

- Maison (niveau)
- Maison/PremierEtage (niveau/niveau)
- Maison/PremierEtage/ChambreA/Temperature (niveau/niveau...)

3. **Publication/Abonnement au topic :**

L'émetteur (Éditeur) et le récepteur (Abonné) communiquent via des sujets (topics). Le récepteur s'abonne à un sujet, chaque fois qu'une valeur (JSON/XML/TEXT) est publiée sur ce sujet, le client la recevra. Les deux parties sont découplées l'une de l'autre, la connexion entre elles est gérée par le courtier MQTT. Le

courtier MQTT filtre tous les messages entrants et les distribue correctement aux abonnés. [11] 1.3



**Figure 1.3:** Architecture MQTT de Publication / Abonnement [12]

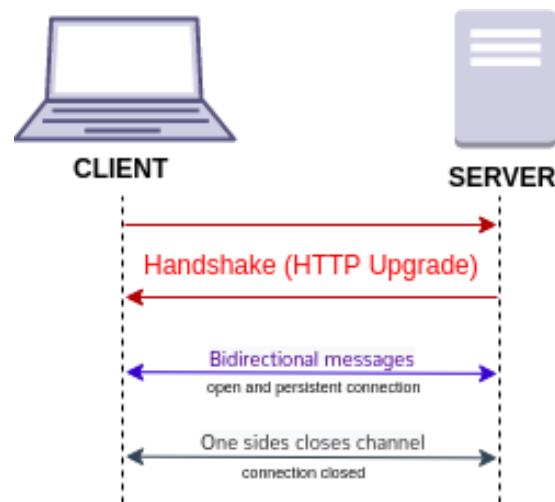
Les sujets ont la possibilité d'utiliser des caractères génériques, tels que #, le # couvre plusieurs niveaux de sujets [10] 1.4.



**Figure 1.4:** Caractère générique hash [10]

### 1.2.3 Protocole Web Sockets

Le protocole WebSocket permet une communication bidirectionnelle entre un client exécutant du code non fiable dans un environnement contrôlé et un hôte distant qui a opté pour les communications provenant de ce code. Le modèle de sécurité utilisé à cet effet est le modèle de sécurité basé sur l'origine, couramment utilisé par les navigateurs web. Le protocole se compose d'une poignée de main d'ouverture suivie d'un encodage de base des messages, superposé sur TCP. L'objectif de cette technologie est de fournir un mécanisme pour les applications basées sur le navigateur qui nécessitent une communication bidirectionnelle avec des serveurs ne dépendant pas de l'ouverture de plusieurs connexions HTTP (par exemple, en utilisant XMLHttpRequest ou des <iframe> et le polling long) 1.5. [13]



**Figure 1.5:** Diagramme décrivant une connexion utilisant WebSocket [14]

## 1.3 Systèmes Embarqués et Intégration IoT

### 1.3.1 Qu'est-ce qu'un système embarqué ?

Un système embarqué est un sous-système électronique spécifique à une application utilisé dans un système plus large tel qu'un appareil, un instrument ou un véhicule. Un système embarqué est généralement composé de logiciels (appelés logiciels embarqués) et d'une plate-forme matérielle. L'évolution des technologies permet l'intégration de plates-formes complexes dans une seule puce (appelée système-sur-puce, SoC) comprenant un ou plusieurs sous-systèmes de processeur pour exécuter des logiciels et une interconnexion sophistiquée en plus de sous-systèmes matériels spécifiques. [15]

### 1.3.2 Le rôle des systèmes embarqués dans l'IoT

La synergie entre l'IoT et les systèmes embarqués est évidente dans leur interdépendance. L'IoT s'appuie sur les systèmes embarqués pour fournir les bases matérielles et la puissance de calcul nécessaires pour traiter et analyser les données. En retour, les systèmes embarqués bénéficient de la connectivité et des capacités de partage de données de l'IoT pour améliorer leur fonctionnalité. [16]

- **Intégration matérielle** : Les systèmes embarqués servent de socle matériel pour les dispositifs IoT, intégrant des microcontrôleurs, des capteurs et des actionneurs pour permettre la communication et le traitement des données. Cette intégration permet la création de dispositifs intelligents et connectés avec des capacités amé-

liorées.

- **Traitement en temps réel** : Les capacités de traitement en temps réel des systèmes embarqués sont cruciales pour les applications IoT qui nécessitent des réponses immédiates. Que ce soit pour ajuster la température d'un thermostat intelligent ou contrôler la vitesse d'un véhicule autonome, les systèmes embarqués garantissent l'exécution opportune et précise des tâches.
- **Calcul en périphérie** : Les systèmes embarqués jouent un rôle pivot dans la mise en œuvre du calcul en périphérie, un paradigme qui implique le traitement des données plus près de la source plutôt que de s'appuyer uniquement sur les ressources cloud. Cela permet non seulement de réduire la latence, mais aussi d'alléger la charge sur la bande passante du réseau.

## 1.4 Conclusion

En conclusion, l'Internet of Things (IoT) est devenu un élément incontournable des environnements domestiques, offrant une multitude de dispositifs et de fonctionnalités pour rendre nos maisons plus intelligentes et plus efficaces. Depuis la définition de l'IoT jusqu'aux protocoles de communication et aux systèmes embarqués, chaque aspect de ce domaine a contribué à façonner l'avenir des maisons intelligentes. Cependant, la sécurité reste un enjeu critique. En adoptant des approches professionnelles et en mettant en œuvre des mesures de sécurité robustes, nous pouvons garantir l'intégrité, l'authentification et la confidentialité. Ces derniers points seront discutés dans le prochain chapitre, assurant ainsi un environnement domestique intelligent, sûr et fonctionnel pour les utilisateurs.

# Chapitre 2

## Architecture du Système et Méthodologie de la Solution

### 2.1 Introduction de chapitre

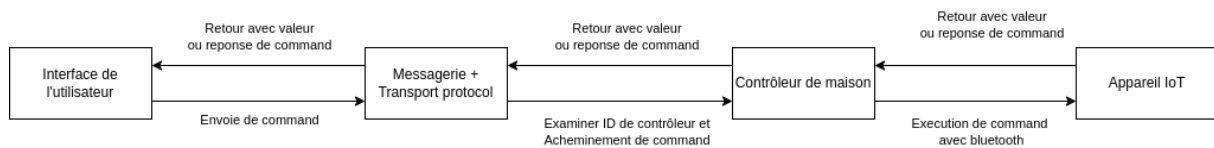
Dans ce chapitre, nous explorerons plus en détail la terminologie nécessaire à la réalisation de ce projet. Nous aborderons l'architecture de l'application web, le protocole de transport de données, le protocole de messagerie et le system embarqué.

### 2.2 Description de la Solution

Pour parvenir à une solution de gestion intelligente de la maison à distance, nous devons mettre en œuvre certaines architectures clés et interconnecter ces architectures. Les architectures seront expliquées en détail dans les prochaines sections, mais il est utile d'avoir une vue d'ensemble globale au préalable [2.1](#). Les trois principales architectures à prendre en compte sont :

1. **Interface de l'utilisateur** : L'interface à laquelle l'utilisateur peut accéder lui permet de gérer sa maison en envoyant des commandes à exécuter, telles que l'arrêt ou l'allumage, et de surveiller sa maison en recevant les valeurs de l'état des appareils.
2. **Protocoles de messagerie et transport** : les protocoles qui achemine les commandes et données entre machine de client, serveur web et le bon contrôleur.
3. **Contrôleur de maison** : typiquement un system embarqué placé a la maison il communique avec les équipements de la maison par BLE en récupérant l'etat de

l'appareil IoT ou la reponse de commmand executé auprès d'eux et les envoie via le protocole de transport à l'interface web.



**Figure 2.1:** Diagramme de flux pour le transport d'un messages

## 2.3 Interface de l'utilisateur

Pour que le concept de contrôle à distance soit appliqué, nous avons besoin d'une interface permettant à l'utilisateur d'accéder et de gérer sa maison intelligente. Il existe plusieurs technologies d'interface qui peuvent être utilisées. Les applications web sont accessibles via des navigateurs web, qui sont des programmes intégrés dans la plupart des systèmes d'exploitation, sans besoin d'installer des outils supplémentaires ce qui améliore l'expérience utilisateur. L'application Web est censée satisfaire tous les besoins requis pour que l'utilisateur surveille et contrôle pleinement sa maison intelligente.

### 2.3.1 Application web

L'Application web comprend les interfaces suivantes :

#### 1. Pages de l'interface d'accueil :

- **Accueil** : Elle offre une vue d'ensemble de notre solution pour maison intelligente, avec des sections expliquant le fonctionnement de notre solution et comment utiliser l'application web.
- **À propos** : Elle offre une vue détaillée de notre solution pour maison intelligente, expliquant les points forts de notre solution ainsi que des questions fréquemment posées avec leurs réponses.
- **Emplacements** : Elle fournit une vue indiquant l'emplacement de notre siège sur la carte algérienne, permettant ainsi aux utilisateurs de connaître l'emplacement du siège des développeurs le plus proche pour des raisons de maintenance.
- **Connexion et inscription** : Ils fournissent une interface pour créer un compte ou se connecter à un compte existant par crée une session dans le serveur web.

## 2. Pages de l'interface de Tableau de bord :

- **Tableau de bord** : Fournit une vue globale du nombre de contrôleurs de maison associés, du nombre d'équipements connectés à ces contrôleurs, ainsi que du temps de disponibilité des appareils en ligne, exprimé en heures et minutes.
- **Appareils** : Sur cette page, l'utilisateur peut gérer ses contrôleurs associés, ainsi que les appareils associés à chaque contrôleur.
- **Paramètres** : Cette page donne à l'utilisateur la possibilité de modifier les paramètres du compte tels que l'image de profil, l'adresse e-mail du compte et le mot de passe du compte ainsi que le ID de contrôleur

Pour le stockage des données des utilisateurs notre solution repose sur la relation de données entre les contrôleurs, les appareils et les comptes utilisateurs. Pour atteindre le stockage des données basé sur les relations, nous pouvons utiliser un langage de programmation de back-end et un modèle relationnel, qui est une solution efficace et satisfait nos besoins.

### 2.3.2 Gestion des données :

Avec des données relationnelles, il est évident que nous devrions appliquer le modèle relationnel d'entités en utilisant le modèle de données suivant :

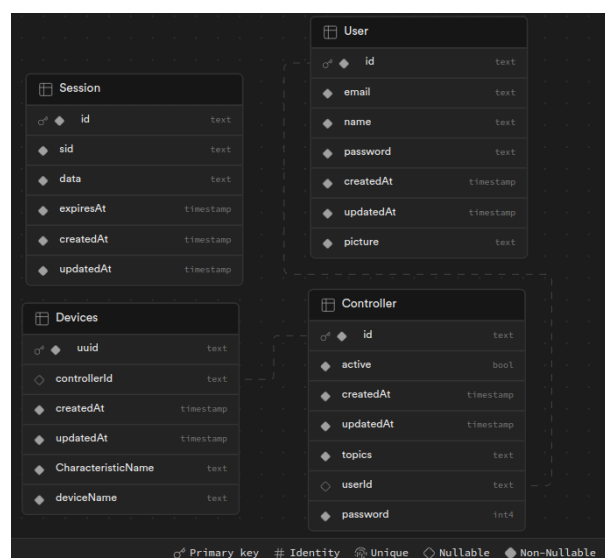


Figure 2.2: Schéma de modèle relationnel

Relation	Attribut	Contraintes	Rôle
User	id  email name password picture createdAt  updatedAt	clé primaire, pas null, unique  pas null, unique	Identifiant unique de l'utilisateur.  Adresse e-mail de l'utilisateur. Nom de l'utilisateur. Mot de passe de l'utilisateur. Photo de profil de l'utilisateur. Date de création du compte de l'utilisateur.  Date de dernière modification du compte de l'utilisateur.
Session	id  sid data expiresAt createdAt updatedAt	clé primaire, pas null, unique  pas null, unique	Identifiant unique de tuple.  Identifiant unique de session. Informations de session. Date d'expiration de session. Date de création du session. Date de dernière modification du session.
Controller	id  password  active  topics  userId  createdAt updatedAt	clé primaire, pas null, unique      clé étrangère, pas null, unique	Identifiant unique de contrôleur.  Mot de passe de l'interface de contrôleur. Représente l'état du contrôleur si il est en ligne. Listes de sujets abonné par le contrôleur <a href="#">1.3</a> ID utilisateur propriétaire de ce contrôleur. Date de création du contrôleur. Date de dernière modification du contrôleur.
Devices	uuid  CharacteristicName  DeviceName controllerId  createdAt updatedAt	clé primaire, pas null, unique  pas null  pas null clé étrangère, pas null, unique	Identifiant unique de Caractéristique l'appareil. Nom de caractéristique de l'appareil. Nom de l'appareil. Identifiant du contrôleur associé à cet appareil Date de création de l'appareil. Date de dernière modification de l'appareil.

Tableau 2.1: Tableau du modèle relationnel



Après avoir terminé la gestion de nos données, nous aurons besoin de transporter ces données utilisateur ainsi que les données des appareils IoT de l'utilisateur à travers les réseaux, tout en maintenant le transport des données en temps réel pour pouvoir appliquer un modèle d'événement-réaction.

## 2.4 Protocoles de messagerie et transport

Les protocoles de messagerie et de transport sont la partie principale du projet. Ils doivent permettre, en utilisant un langage de programmation, de rendre les commandes utilisateur et les données transportables, de créer une architecture organisée des appareils IoT, de transporter les données et l'état des appareils IoT, et de transporter les commandes utilisateur vers un appareil IoT spécifique, tout en maintenant le transport en temps réel et en appliquant l'authentification et l'intégrité des données.

### 2.4.1 Protocole de Messagerie

Le protocole de messagerie joue le rôle de bureau de poste. Son rôle est de recevoir les messages du point A, d'appliquer des filtres et des règles de QoS aux messages en fonction du type, de la source et du nom de la destination, et de les envoyer au point B via un protocole de transport. De plus, comme le bureau de poste, il est responsable de la création des enveloppes et de leur scellement en chiffrant les données.

MQTT nous fournit une solution utilisant une architecture de publication/abonnement tout en étant optimale et rapide en minimisant les paquets TCP envoyés. Il nous aide également à créer une architecture des appareils dynamique et organisée par des sujets [1.3](#). Utilisant ces fonctionnalités puissantes pour concevoir notre maison intelligente par :

#### 1. Choix de l'agent (Broker) MQTT :

Le choix d'un agent est une tâche cruciale. Il existe de nombreux agents, chacun présentant des avantages et des faiblesses. Notre agent doit offrir les fonctionnalités suivantes :

- Utilise le modèle de publication/abonnement.
- Support du protocole TLS/SSL pour le chiffrement des communications.
- Support des niveaux de qualité de service (QoS), Cela définit le niveau de garantie de livraison pour un message spécifique.

## 2. Choix D'architecture des sujets :

Avec les sujets servant de tunnel pour le passage de nos messages, nous utiliserons cette fonctionnalité tout en maintenant la conception des sujets dynamique et évolutive. On propose le design suivant :

ID de controlleur\Type\Appareil\charge utile (Payload JSON)

Le Niveau \Type est utilisé pour distinguer entre un sujet de valeur ou un sujet de commande ou un sujet de Dernière volonté et testament, noté par "vlr" pour valeur, "cmd" pour commande et "lwt" pour Dernière volonté et testament. Par exemple :

```
id1\cmd\Nom appareil\{"Power" : "On"}
id1\vlr\Nom appareil\{"Temperature" : "28"}
id1\lwt\Nom appareil\{"Crashed" : True}
```

Concevoir les sujets de cette manière présente des avantages, tels qu'une meilleure le filtrage et journalisation des messages et un débogage plus efficace 2.3, une scalabilité pour l'ajout de nouveaux controlleur et une réduction du calcul des sujets. Par placer le nom de l'appareil au début du sujet suivi de la filtration applique le principe de Diviser pour régner (Divide-and-conquer) 2.4.

```
3117 2024-03-23T20:35:55: Received PUBLISH from raspberry-5247 (d0, q0, r0, m0, 'rasberry-5247/vlr/thermostat', ... (265 bytes))
3118 2024-03-23T20:35:55: Sending PUBLISH from hamza (d0, q0, r0, m0, 'rasberry-5247/cmd/lampe', ... (101 bytes))
3119 2024-03-23T20:36:48: Received PUBLISH from raspberry-5247 (d0, q0, r0, m0, 'rasberry-5247/cmd/lampe', ... (263 bytes))
```

Figure 2.3: Fichier de journaux du MQTT

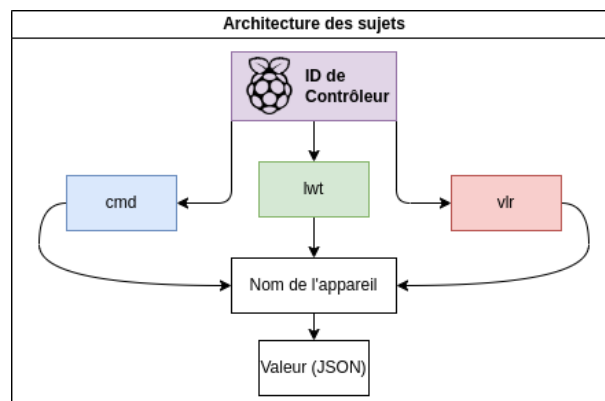


Figure 2.4: Architecture des sujets MQTT

### 3. Options de connexion et messages :

MQTT nous fournit un ensemble d'options par rapport au messages et sujets. Voici quelques options clés que nous devrions considérer dans une architecture IoT :

- **LWT** : LWT nous donne la main d'envoyer un message sur un sujet spécifique lorsqu'une déconnexion non planifiée se produit, en publiant un message bien formé sur un sujet spécifique à cet événement. Cela nous permet de définir une logique de détection automatique et de réaction lorsque une déconnexion non souhaitée se produit [2.5](#).
- **QoS** : Définissez le niveau à 0 pour les sujets de valeurs des appareils afin d'éviter l'utilisation de paquets ACK à chaque mise à jour des valeurs, réduisant ainsi le nombre de paquets TCP envoyés. Utilisez le niveau 1 pour les messages de commande et LWT en raison de la fréquence d'exécution réduite [2.6](#).
- **Messages conservés (retain messages)** : En définissant "retain" sur true, les nouveau abonné du sujet recevra la dernière valeur publiée sur celui-ci. Chaque nouveau souscripteur puisse obtenir la dernière valeur sans avoir besoin de l'envoyer à nouveau dans le sujet pour tous les souscripteurs, ce qui minimise les paquets TCP envoyés [2.6](#).

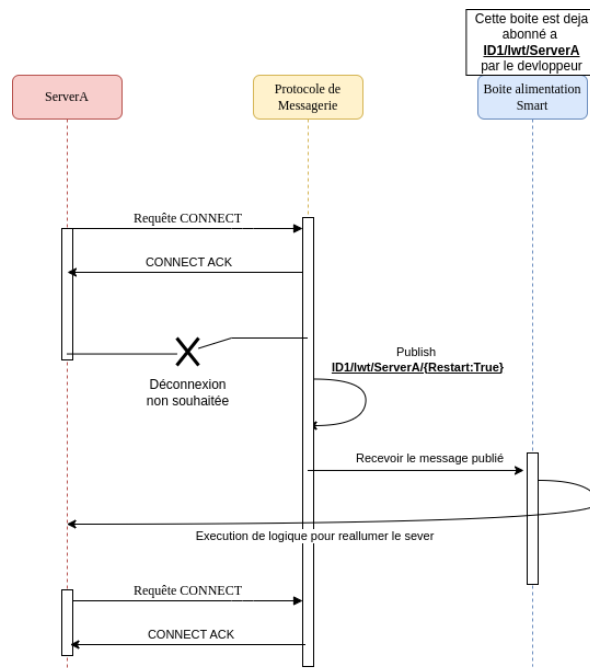


Figure 2.5: Diagramme de séquence pour LWT

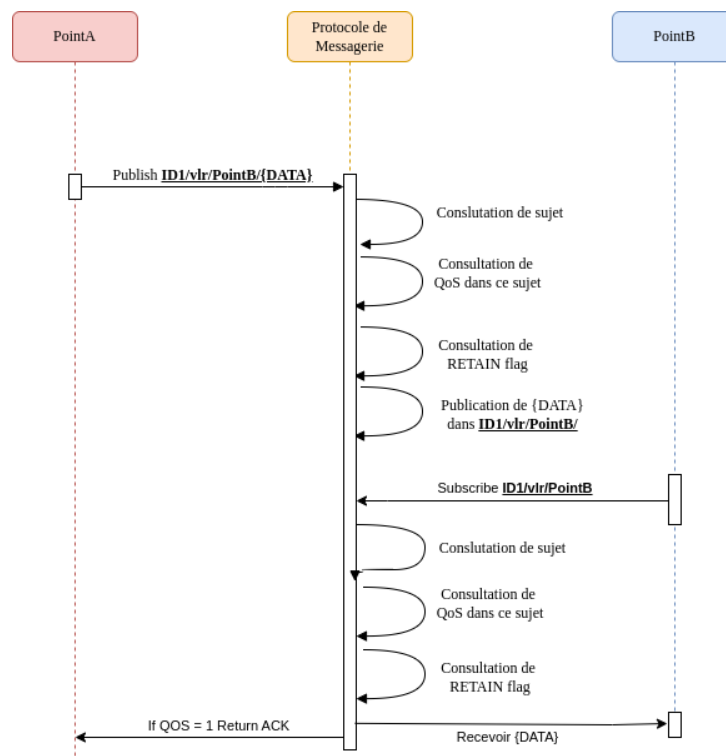


Figure 2.6: Diagramme de séquence pour QOS,RETAIN

Après avoir réussi à mettre en place une architecture domestique et un système de communication, cette communication n'est pas encore en temps réel. MQTT fonctionne sur

une communication unidirectionnelle et n'utilise pas de communication bidirectionnelle, ce qui peut entraîner un suivi lent des valeurs dans le navigateur en cas de latence élevée.

### 2.4.2 Protocole de Transport

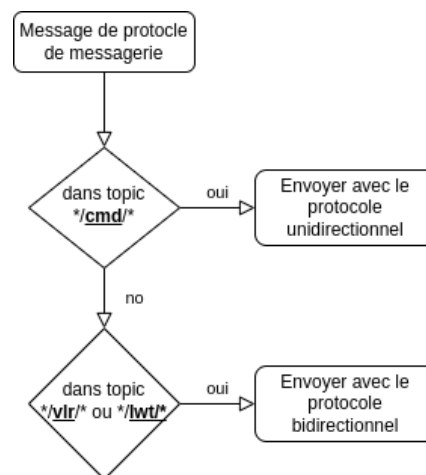
De nombreuses solutions peuvent être mises en œuvre pour résoudre ces problèmes. Pour généraliser, le protocole de transport doit être capable de nous fournir une communication bidirectionnelle. De plus, pour empêcher le système de planter lors de périodes de communication intensive, le protocole de transport doit fonctionner sur plusieurs ports simultanément afin de répartir la charge selon les besoins, en utilisant les ports comme suit :

#### 1. Port de contrôle :

Ce port est configuré pour fonctionner avec MQTT standard. Il est utilisé lorsque des communications non temps réel sont nécessaires, telles que les messages de configuration d'administration.

#### 2. Port de communication :

Ce port est configuré pour exécuter MQTT lorsque des communications en temps réel sont nécessaires, telles que le acheminement des valeurs et l'état des appareils.



**Figure 2.7:** Diagramme de séquence de protocole de transport

Après avoir configuré la communication de manière efficace, tous les objets peuvent publier sans contraintes et tous les messages ne sont pas cryptés, ce qui n'est pas une communication que nous voulons effectuer sur des réseaux non protégés. Nous devons garantir l'intégrité des données et l'authentification.

### 2.4.3 Intégrité, Authentification et Confidentialité

Sans définir de règles d'authentification, de confidentialité ou d'intégrité des données, nous exposons nos communications aux attaques de type MIM et DDoS. Bien que de nombreuses solutions puissent être utilisées pour contrer cela, nous pouvons commencer par les mesures de sécurité de base fournies par défaut par l'agent MQTT :

#### 1. Authentification :

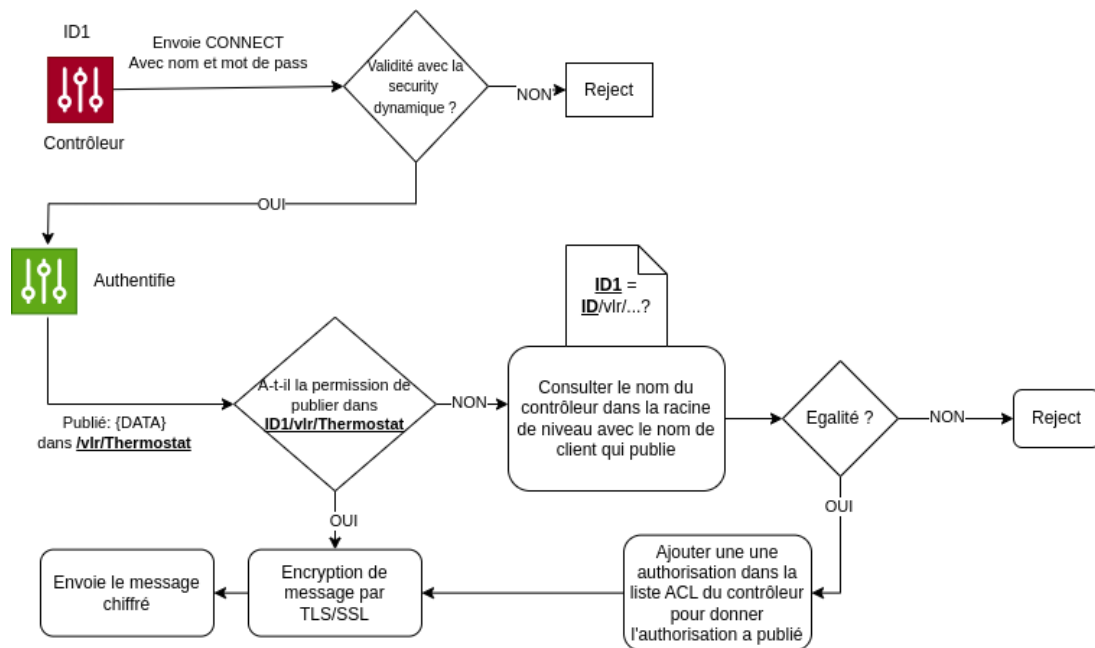
Pour l'authentification des clients qui sont soit des utilisateurs soit des contrôleurs, il existe deux solutions : la liste de noms d'utilisateur et mots de passe ou le plugin de sécurité dynamique. Ils nous permettent tous deux de créer des utilisateurs définis par des identifiants et des mots de passe. La sécurité dynamique nous offre la possibilité de créer/modifier/supprimer des utilisateurs pendant que l'agent fonctionne, sans avoir besoin de le redémarrer. Nous utilisons cette fonctionnalité pour créer.

#### 2. Intégrité des données :

MQTT par défaut ne chiffre pas les données, mais le agent Mosquitto nous offre la possibilité de chiffrer les messages par TLS/SSL sur les ports. En utilisant l'architecture PKI et les certificats, une poignée de main TLS est effectuée au début de la connexion, suivie du chiffrement des données pour chaque message après la poignée de main. Nous pouvons également utiliser le certificat pour l'authentification, mais la sécurité dynamique se combine très bien avec l'autorisation et l'intégrité des données.

#### 3. Confidentialité :

Pour la confidentialité, Mosquitto nous fournit une solution intégrée de configuration de liste de contrôle d'accès (ACL) qui fonctionne avec le plugin de sécurité dynamique. Nous pouvons configurer des règles ACL sur les sujets pour des événements tels que la publication et l'abonnement avec des utilisateurs ou des contrôleurs spécifiques. En utilisant cette solution, nous pouvons établir une logique qui accorde aux contrôleurs et aux utilisateurs les autorisations requises en fonction de l'abonnement et de la publication sur les sujets.

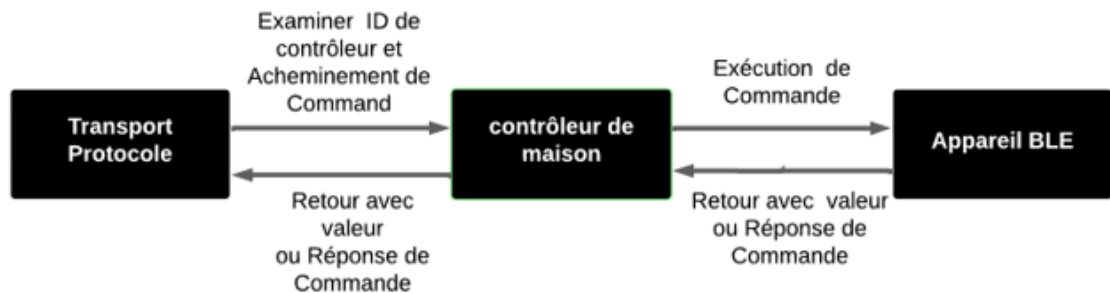


**Figure 2.8:** MQTT Encryption, Authentification et Authorisation organigrammee

Après avoir configuré l'interface et les protocoles, nous tournons maintenant vers la partie électronique de la solution. Nous devons mettre en place un système avec une logique capable de filtrer, détecter et connecter des appareils Bluetooth. Il doit également être capable d'interagir avec les appareils en envoyant des commandes et en récupérant des valeurs. De plus, il doit être en mesure de communiquer en utilisant MQTT pour publier et s'abonner à des sujets.

## 2.5 Contrôleur de la maison

Dans le projet, l'absence d'un contrôleur de maison pose des difficultés importantes, comme la complexité accrue de la communication avec les équipements BLE de la maison intelligente et l'impossibilité de contrôler la maison à distance. Cette situation met en évidence l'importance d'examiner l'intégration d'un contrôleur de maison afin d'améliorer la cohérence et l'efficacité du système global.



**Figure 2.9:** Diagramme pour le transport des données

Ce schéma 2.9 a été élaboré en utilisant cette solution où le contrôleur de maison joue un rôle essentiel en tant que passerelle et point central de contrôle, facilitant ainsi la communication entre les dispositifs domestiques compatibles avec le BLE et le Protocole de Transport. Une fois que les dispositifs Bluetooth Low Energy ont été développés avec leurs caractéristiques et services spécifiques, la prochaine étape consiste à les connecter au contrôleur de la maison.

### 2.5.1 Compatibilité des dispositifs BLE avec le Contrôleur

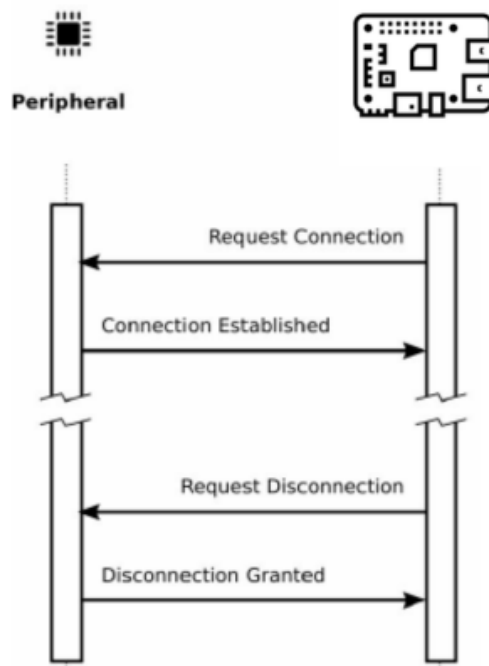
En tant que point central de contrôle, le contrôleur de maison intègre des scripts en arrière-plan qui facilitent l'interaction avec les dispositifs BLE. Ces scripts sont déployés pour effectuer des opérations telles que la connexion, la déconnexion et le balayage (découverte) des appareils. Par la suite, ils sont utilisés pour collecter des informations provenant des dispositifs BLE, définir des valeurs sur ces appareils selon les besoins, et gérer les notifications en cas de modifications importantes dans les dispositifs connectés.

Pour notre projet, nous avons employé les modules ci-dessous :

- **Balayage, connexion et déconnexion :**

La détection des appareils BLE à proximité est assurée par ce module. Il procède à un balayage régulier afin de repérer les périphériques disponibles et de collecter leurs données, comme les adresses MAC et les noms, ensuite le module assure la connexion avec le périphérique choisi 2.10.

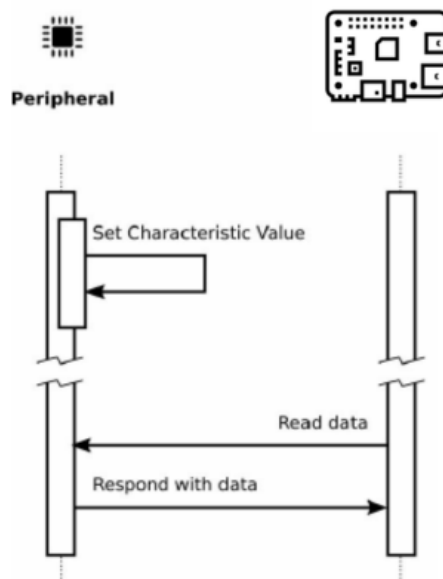




**Figure 2.10:** Connexion et Déconnexion entre Périphérique et Contrôleur

— **La lecture des caractéristiques :**

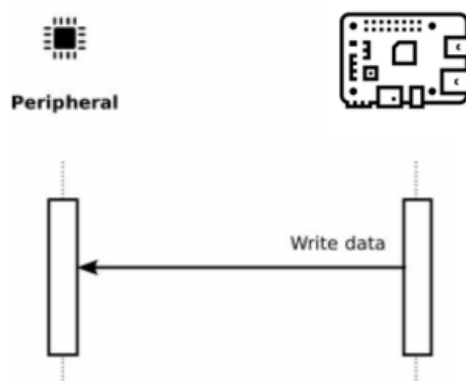
Une fois connecté, ce module permet de consulter les informations du périphérique BLE. Il rassemble des informations spécifiques disponibles en fonction des caractéristiques du périphérique [2.11](#).



**Figure 2.11:** Lecture de Données du Périphérique par le Contrôleur

— **Écriture des valeurs sur les caractéristiques :**

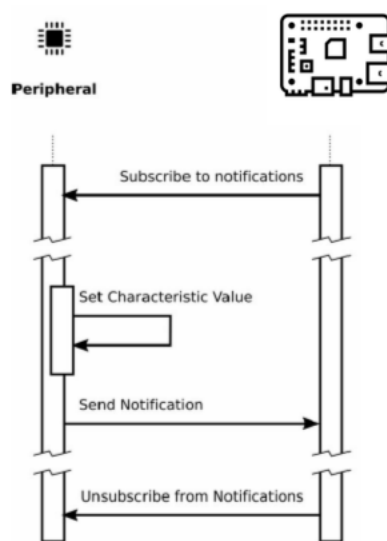
Il est possible d'écrire des informations sur les caractéristiques du périphérique BLE, ce qui permet de configurer des paramètres spécifiques ou de modifier les valeurs existantes de manière appropriée [2.12](#).



**Figure 2.12:** Écriture de Données du Contrôleur vers un Périphérique

— **Gestion des notifications :**

Ce module est responsable de recevoir les notifications envoyées par le dispositif BLE. Il est possible d'ajuster sa configuration pour recevoir des notifications lorsque certaines caractéristiques changent ou lorsque des événements spécifiques surviennent sur le dispositif [2.13](#).

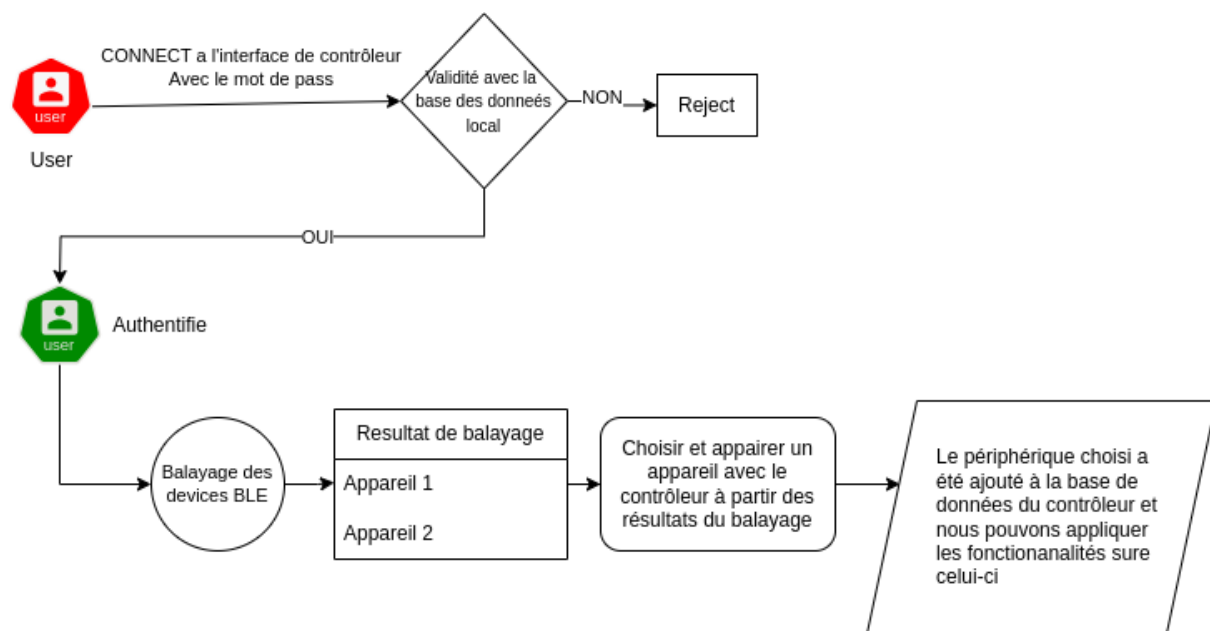


**Figure 2.13:** Notification de Changement de Caractéristique d'un Périphérique au Contrôleur

Ces fonctionnalités nous ont aidés à interagir et à contrôler nos appareils intelligents à partir d'un contrôleur centralisé. Cependant, un logiciel malveillant injecté dans le contrôleur pourrait donner à un pirate la possibilité de contrôler tous les appareils et d'exploiter ces fonctionnalités à leur avantage.

### 2.5.2 Interface du contrôleur et authentification

En tant que mesure de sécurité, un tableau de bord du contrôleur avec authentification est nécessaire. De nombreuses méthodes d'authentification peuvent être utilisées comme solution, telles que l'utilisation d'un mot de passe, cette approche incluent la mise en place d'un tableau de bord du contrôleur et l'appariage des appareils avec le contrôleur depuis le tableau de bord lui-même, plutôt que depuis l'interface utilisateur sur le site Web. Cela applique l'authentification en garantissant que la personne appariant les appareils est bien le propriétaire du contrôleur [2.14](#).



**Figure 2.14:** Organigramme de authentification et l'usage de l'interface du contrôleur

## 2.6 Schéma séquentiel

Pour avoir une idée globale de notre conception, nous devons la représenter sur un schéma, en combinant les schémas et les idées précédents que nous avons explorés, et en connectant les éléments clés, nous pouvons avoir une vue d'ensemble de la solution :

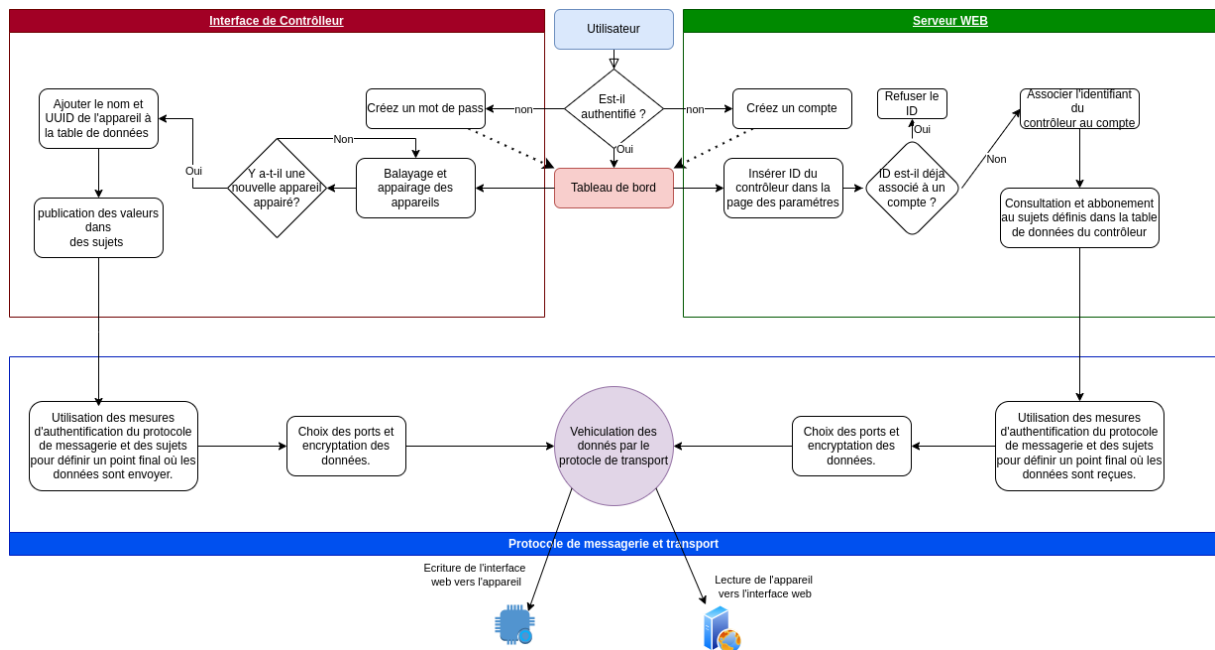


Figure 2.15: Diagramme d'activité de déroulement de system

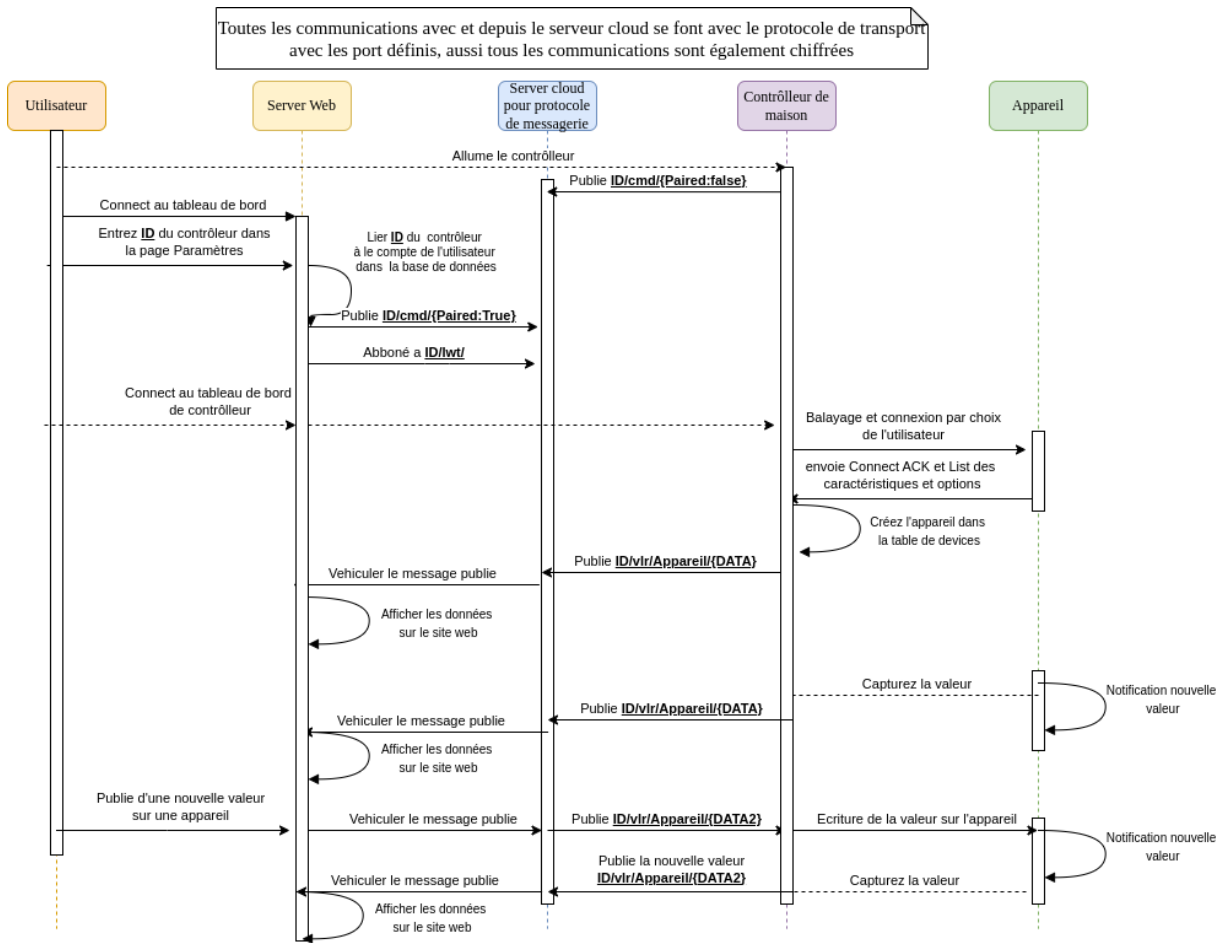


Figure 2.16: Diagramme séquentiel de déroulement de system

## 2.7 Conclusion

Dans ce chapitre, nous avons examiné en détail les composants essentiels de notre solution. Nous avons exploré les subtilités de l'interface utilisateur et ses fonctionnalités, défini le fonctionnement des protocoles et leur intégration avec le système embarqué et l'interface. De plus, nous avons interconnecté ces éléments pour former une solution complète garantissant à la fois l'intégrité et l'authentification. Nous avons mis en place des mécanismes de sauvegarde tels que les bases de données, des outils de gestion des défaillances comme LWT, et intégré des fonctionnalités de qualité de service. De plus, nous avons examiné les meilleures pratiques pour résoudre les problèmes et présenté des stratégies pour atténuer les pires scénarios. Dans le chapitre suivant, nous explorerons la mise en œuvre pratique de ces concepts, faisant passer notre solution de l'exploration théorique à une étape expérimentale.

# Chapitre 3

## Réalisation et Mise en Œuvre de la Solution

### 3.1 Introduction

Dans ce chapitre, nous présenterons les techniques, solutions et configurations utilisées pour atteindre notre objectif et appliquer les concepts discutés, présenterons les logiciels, technologies, matériels, et scripts utilisés. Nous ne parlerons pas de l'installation des bibliothèques, de la gestion des câbles et de l'installation des logiciels dans ce chapitre, Ces informations seront disponibles dans l'annexe.

### 3.2 Logiciels et Application utilisés

- **Visual Studio Code** : Est un éditeur de code source développé par Microsoft pour Windows, Linux, macOS et les navigateurs web.
- **nRF Connect** : nRF Connect est un outil puissant et polyvalent qui nous permet de créer et de simuler des dispositifs BLE depuis notre téléphone avec toutes les caractéristiques et réglages souhaités.
- **Postman** : Postman est une application qui permet de tester les API web, nous donnant la possibilité de tester les requêtes POST, GET, DELETE et PUT vers un point de terminaison donné.
- **Docker** : Docker utilise la virtualisation au niveau du système d'exploitation pour distribuer des logiciels dans des packages appelés conteneurs. Il nous aide à placer le protocole MQTT dans un environnement isolé dédié uniquement au protocole MQTT, et également à gérer les ports, les adresses IP et les protocoles de transport.
- **Rpi-Imager** : Un outil convivial pour créer des supports bootables pour les ap-

pareils Raspberry Pi, cela nous aide à installer le système d'exploitation Raspbian sur la carte SD. Pour voir comment installer le OS raspbian voir l'annexe.

### 3.3 Technologies et bibliothèque utilisés

- **Javascript** : est un langage de programmation et une technologie centrale du Web.
- **Node.js** : Node.js est un environnement d'exécution JavaScript open-source multiplateforme. Cette technologie nous aide à créer toute la logique backend pour les bases de données, les protocoles et les interactions avec le système d'exploitation.
- **@Express.js** : Est un framework d'application web back-end pour construire des API RESTful avec Node.js. Cela nous aide à créer des pages web dynamiques qui changent de contenu en fonction des conditions désirées.
- **@MQTT.js** : Est une bibliothèque client MQTT largement utilisée pour JavaScript, conçue pour fonctionner à la fois dans les environnements navigateur et Node.js. Cette bibliothèque nous aide à utiliser toutes les fonctionnalités MQTT.
- **Postgresql** : système de gestion de base de données relationnelle open-source mettant l'accent sur l'extensibilité et la conformité SQL, il nous aide à mettre en place le modèle relationnel.
- **Eclipse Mosquitto** : Est un agent MQTT open source (sous licence EPL/EDL). Le agent est le moteur sous le capot MQTT, Mosquitto satisfait tous nos besoins et peut appliquer toutes les fonctionnalités MQTT nécessaires.
- **WebSockets** : C'est notre protocole de transport, en raison de ses excellentes fonctionnalités bidirectionnelles avec le web, est un choix parfait pour afficher des données en temps réel sur le web.
- **@abandonware/Noble** : La bibliothèque est conçue pour Node.js et est utilisée pour détecter et contrôler les périphériques BLE présents dans notre environnement.

### 3.4 Matériels utilisés

- **Carte SD** : Ceci est une carte de stockage, placée sur le Raspberry Pi. Son principal usage est de contenir le stockage du système et les données de la base de données locale.
- **RaspberryPi** : Nous avons choisi le Raspberry Pi 4 Computer Model B 8 GB RAM comme base et contrôleur pour notre projet de maison intelligente grâce à sa compatibilité avec divers systèmes d'exploitation, à son prix abordable, à sa

polyvalence, ainsi qu'à son intégration native du Wi-Fi et du Bluetooth.

- **Telephone Android** : Nous avons utilisé deux téléphones Android pour simuler deux dispositifs BLE à l'aide du programme nRF Connect.

## 3.5 Dispositifs utilisés

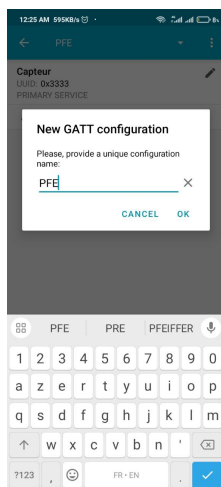
Notre dispositifs utilisés sont des appareils virtuels installés avec le programme nRF sur nos téléphones mobiles.

### 3.5.1 Premier Dispositif :

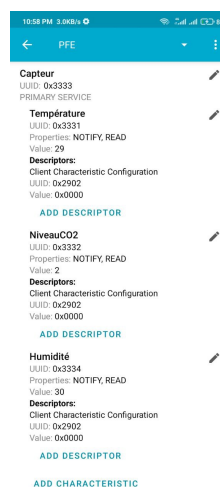
Cet appareil multifonctionnel, mesurant la température, l'humidité et le niveau de CO2 à la maison. Le nom local de cet appareil est "Capteur multifonction" 3.3, défini pour faciliter son identification lors des connexions BLE. Le serveur de cet appareil a été nommé PFE 3.1, et son profil GATT 3.2, incluant un service et des caractéristiques, a été rédigé et organisé selon le tableau suivant :

	Nom	UUID	Propriétés
Profile Gatt	PFE	X	X
Service 1	Capteur	0x3333	X
Caractéristique 1	Température	0x3331	Notify,Read
Caractéristique 2	NiveauCO2	0x3332	Notify,Read
Caractéristique 3	Humidité	0x3334	Notify,Read

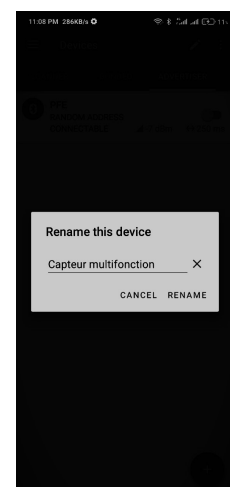
**Tableau 3.1:** UUID du service, nom, pour le premier dispositif, ainsi que l'UUID et les propriétés des caractéristiques



**Figure 3.1:** Serveur de noms



**Figure 3.2:** Profil GATT



**Figure 3.3:** Local Name

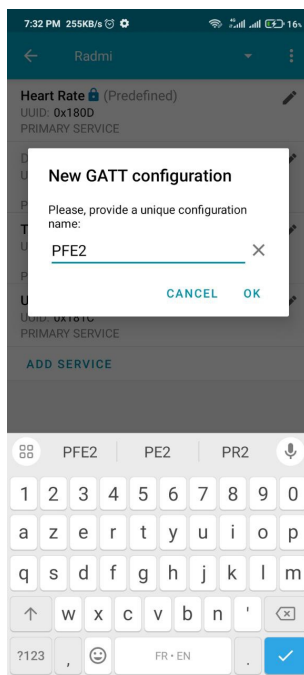


### 3.5.2 deuxième dispositif

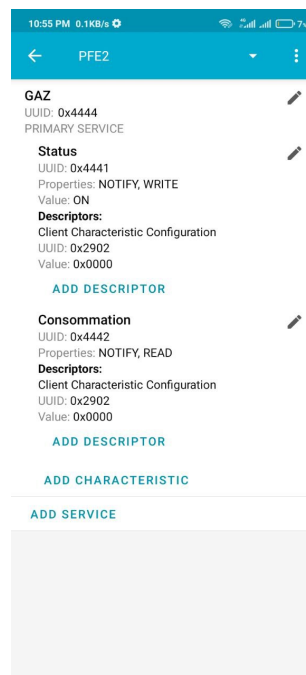
Cet appareil, dédié à surveiller le gaz dans la maison, a été équipé. Le nom local de cet appareil est 'Capteur GAZ' 3.6, défini pour faciliter son identification lors des connexions BLE. Le serveur de cet appareil a été nommé PFE2 3.4, et son profil GATT 3.5, incluant un service et des caractéristiques, a été rédigé et organisé selon le tableau suivant :

	Nom	UUID	Propriétés
Profile Gatt	PFE2	X	X
Service 1	GAZ	0x4444	X
Caractéristique 1	Statut	0x4441	Notify,Write
Caractéristique 2	Consommation	0x4442	Notify,Read

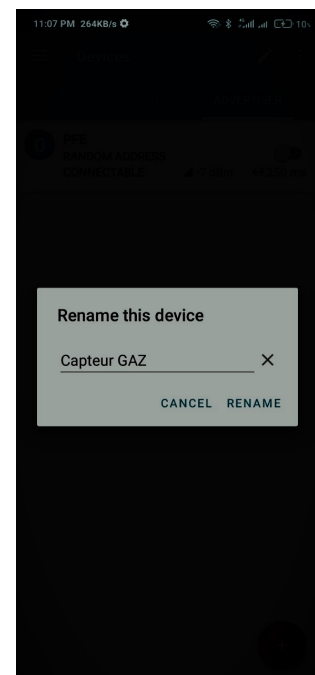
**Tableau 3.2:** UUID du service, nom, pour le deuxième dispositif, ainsi que l'UUID et les propriétés des caractéristiques



**Figure 3.4:** Serveur de noms



**Figure 3.5:** Profil GATT



**Figure 3.6:** Local Name

## 3.6 Scripts et configuration utilisés

### 3.6.1 Fichier configuration MQTT et Docker

Le fichier de configuration MQTT (Figure. 3.7) est le cerveau du protocole. Dans ce fichier, nous définissons toutes les règles de notre protocole, telles que les méthodes

d'authentification et de chiffrement, ainsi que les paramètres des fichiers journaux. En revanche, dans le fichier de sécurité dynamique (Figure. 3.10), nous trouvons tous les utilisateurs qui sont créés avec leurs rôles, et chaque rôle ayant une liste de contrôle d'accès (ACL) qui définit ce qu'un client peut faire avec ce rôle.

```
1  per_listener_settings false
2  allow_anonymous false
3  allow_zero_length_clientid false
4  check_retain_source true
5
6  plugin /usr/lib/mosquitto_dynamic_security.so
7  plugin_opt_config_file /mosquitto/dynamic-security.json
8
9  persistence true
10 persistence_location /mosquitto/data/
11 log_type all
12 log_timestamp true
13 log_timestamp_format %Y-%m-%dT%H:%M:%S
14 log_dest file /mosquitto/log/mosquitto.log
15 log_dest stdout
16 connection_messages true
17 autosave_interval 1800
18 autosave_on_changes true
19
20 listener 8883 0.0.0.0
21 protocol websockets
22 socket_domain ipv4
23 cafile /mosquitto/ca_certificates/ca.crt
24 keyfile /mosquitto/cert/server.key
25 certfile /mosquitto/cert/server.crt
26 require_certificate false
27 tls_version tlsv1.2
28
29 listener 8884 0.0.0.0
30 protocol mqtt
31 socket_domain ipv4
32 cafile /mosquitto/ca_certificates/ca.crt
33 keyfile /mosquitto/cert/server.key
34 certfile /mosquitto/cert/server.crt
35 require_certificate false
36 tls_version tlsv1.2
```

Figure 3.7: Fichier de configuration MQTT

```

"clients": [{
  "username": "hamza",
  "textname": "Dynsec admin user",
  "roles": [{
    "rolename": "admin"
  }, {
    "rolename": "temperature"
  }],
  "password": "mz+cqtD8UZh0HbcjTWqybIF+pTMFno5b5Xh0hE33SZp9hkV26N3wgoPelJeQo8dduYI3qdsdhdWFM8ajQipSyuQ==",
  "salt": "z1JC5pTS9aGGzcbA",
  "iterations": 101
}],
"roles": [{
  "rolename": "temperature",
  "acls": [{
    "acltype": "publishClientSend",
    "topic": "temperature/#",
    "priority": 0,
    "allow": true
  }, {
    "acltype": "publishClientReceive",
    "topic": "temperature/#",
    "priority": 0,
    "allow": true
  }, {
    "acltype": "subscribePattern",
    "topic": "temperature/#",
    "priority": 0,
    "allow": true
  }, {
    "acltype": "unsubscribePattern",
    "topic": "temperature/#",
    "priority": 0,
    "allow": true
  }
]}]

```

Figure 3.8: Fichier de sécurité dynamique MQTT

Tout comme le fichier de configuration MQTT, nous avons une configuration Docker (Figure. 3.14) qui représente le moteur du système exécutant MQTT. Avec cette configuration, nous définissons les adresses IP, les ports disponibles et les fichiers partagés de notre système vers le conteneur.

```

version: "3.8"

services:
  mosquitto:
    image: eclipse-mosquitto:2
    ports:
      - 8883:8883
      - 8884:8884
    volumes:
      - /etc/mosquitto/config:/mosquitto/config
      - /etc/mosquitto/data:/mosquitto/data
      - /etc/mosquitto/log:/mosquitto/log
      - /etc/mosquitto/cert:/mosquitto/cert
      - /etc/mosquitto/ca_certificates:/mosquitto/ca_certificates
      - /etc/mosquitto/dynamic-security.json:/mosquitto/dynamic-security.json
    restart: unless-stopped

networks:
  mosquitto:
    driver: bridge
  ipam:
    config:
      - subnet: 192.168.1.0/24

```

Figure 3.9: Fichier de docker compose

### 3.6.2 Script pour l'Interaction avec le protocole MQTT

Pour démontrer un flux de script de base, le travail initial par défaut du script est de souscrire le client qui exécute le script au sujet de sécurité dynamique pour vérifier les règles ACL, et de souscrire au client LWT pour vérifier s'il y a des valeurs conservées afin d'exécuter une logique basée sur cela.

```

4591 2024-05-20T09:33:13: Opening websockets listen socket on port 8883.
4592 2024-05-20T09:33:13: Opening ipv4 listen socket on port 8884.
4593 2024-05-20T09:33:13: mosquitto version 2.0.18 running
4594 2024-05-20T13:42:08: Client connection from 159.223.106.63 failed: error:1402542E:SSL routines:ACCEPT_SR_CLNT_HELLO:tlsv1 alert protocol version.
4595 2024-05-20T19:42:30: mosquitto version 2.0.18 terminating
4596 2024-05-20T19:42:30: Saving in-memory database to /mosquitto/data//mosquitto.db.
4597 2024-05-20T19:42:32: mosquitto version 2.0.18 starting
4598 2024-05-20T19:42:32: Config loaded from /mosquitto/config/mosquitto.conf.
4599 2024-05-20T19:42:32: Loading plugin: /usr/lib/mosquitto_dynamic_security.so
4600 2024-05-20T19:42:32: Opening websockets listen socket on port 8883.
4601 2024-05-20T19:42:32: Opening ipv4 listen socket on port 8884.
4602 2024-05-20T19:42:32: mosquitto version 2.0.18 running
4603 2024-05-20T19:44:53: New client connected from 105.98.28.233:38770 as Admin (p2, c0, k60, u'admin').
4604 2024-05-20T19:44:53: Will message specified (35 bytes) (r0, q0).
4605 2024-05-20T19:44:53: disconnect/Admin
4606 2024-05-20T19:44:53: Sending CONNACK to Admin (0, 0)
4607 2024-05-20T19:44:53: Received SUBSCRIBE from Admin
4608 2024-05-20T19:44:53: $CONTROL/dynamic-security/# (QoS 0)
4609 2024-05-20T19:44:53: Admin 0 $CONTROL/dynamic-security/#
4610 2024-05-20T19:44:53: disconnect/# (QoS 0)
4611 2024-05-20T19:44:53: Admin 0 disconnect/#
4612 2024-05-20T19:44:53: Sending SUBACK to Admin
4613 2024-05-20T19:44:57: Client Admin closed its connection.
4614 2024-05-20T19:56:11: mosquitto version 2.0.18 terminating
4615 2024-05-20T19:56:11: Saving in-memory database to /mosquitto/data//mosquitto.db.
4616 2024-05-20T19:56:18: mosquitto version 2.0.18 starting
4617 2024-05-20T19:56:18: Config loaded from /mosquitto/config/mosquitto.conf.
4618 2024-05-20T19:56:18: Loading plugin: /usr/lib/mosquitto_dynamic_security.so
4619 2024-05-20T19:56:18: Opening websockets listen socket on port 8883.
4620 2024-05-20T19:56:18: Opening ipv4 listen socket on port 8884.
4621 2024-05-20T19:56:18: mosquitto version 2.0.18 running
4622 2024-05-20T19:57:11: New client connected from 105.98.28.233:50720 as Admin (p2, c0, k60, u'admin').
4623 2024-05-20T19:57:11: Will message specified (35 bytes) (r0, q0).
4624 2024-05-20T19:57:11: Admin/lwt
4625 2024-05-20T19:57:11: Sending CONNACK to Admin (1, 0)
4626 2024-05-20T19:57:11: Received SUBSCRIBE from Admin
4627 2024-05-20T19:57:11: $CONTROL/dynamic-security/# (QoS 0)
4628 2024-05-20T19:57:11: Admin 0 $CONTROL/dynamic-security/#
4629 2024-05-20T19:57:11: Admin/lwt (QoS 0)
4630 2024-05-20T19:57:11: Admin 0 Admin/lwt
4631 2024-05-20T19:57:11: Sending SUBACK to Admin
4632 2024-05-20T19:57:14: Client Admin closed its connection.

```

Figure 3.10: Résultat d'exécution du script de connexion MQTT dans le fichier journal

Le script a quatre événements principaux :

- À la connexion : lorsque un client établit une communication avec MQTT.
- À la déconnexion : lorsque un client termine la communication avec MQTT.
- À la réception d'un message : lorsque un client reçoit un message via le protocole MQTT.
- À la survenue d'une erreur : lorsque une erreur se produit, comme un sujet inexistant ou lorsque le client n'a pas les permissions nécessaires.

```

// Import CA cert
const ca = fs.readFileSync(process.env.CERT_PATH).toString();

const router = express.Router();

// MQTT broker connection options
const options = {
  clientId: process.env.CLIENT_ID,
  protocolId: 'MQTT',
  protocolVersion: 4,
  keepalive: 60,
  reconnectPeriod: 5 * 1000,
  connectTimeout: 6 * 1000,
  username: process.env.BROKER_USERNAME,
  password: process.env.BROKER_PASSWORD,
  clean: false,
  will: {
    topic: `${process.env.CLIENT_ID}/lwt`,
    payload: `${process.env.CLIENT_ID} disconnected without a reason`,
    qos: 0,
    retain: false,
  },
  ca,
};

// MQTT broker URL
const brokerUrl = process.env.BROKER_URL;

// Create MQTT client instance
const client = mqtt.connect(brokerUrl, options);

client.on('reconnect', () => {
  console.log('Reconnecting...');
});
client.on('message', (topic, message) => {
  console.log(`Received message on topic ${topic}: ${message.toString()}`);
});
client.on('close', () => {
  console.log('Disconnecting...');
});
client.on('error', (err) => {
  console.error('MQTT client error:', err);
});

```

Figure 3.11: Script de connexion MQTT

```

// Event handlers
client.on('connect', () => {
  console.log('Connected to MQTT broker');
  const publishOptions = {
    qos: 1,
    retain: true,
    dup: false,
  };
  const subscribeOptions = {
    qos: 0,
  };
  client.subscribe(
    [process.env.CONTROL_TOPIC, `${process.env.CLIENT_ID}/lwt`],
    subscribeOptions,
    (err, granted) => {
      if (err) {
        console.error('Error subscribing to topic:', err);
      } else if (granted) {
        console.log(`Subscribed to: ${JSON.stringify(granted)}`);
      }
    },
  );
});

// Middleware to attach mqtt-Client PublishOptions and subscribeOptions to the request object
router.use((req, res, next) => {
  req.client = client;
  req.publishOptions = publishOptions;
  req.subscribeOptions = subscribeOptions;
  next();
});

router.use('/admin', [getRoutes, deleteRoutes, postRoutes, putRoutes]);
});

module.exports = router;

```

Figure 3.12: Suit de script de connexion MQTT

En plus de cela, nous avons créé un module regroupant toutes les fonctions fournies par l'API MQTT pour faciliter l'interaction avec la sécurité dynamique et la gestion des clients et des permissions.

```
const express = require('express');
const adminController = require('../controllers/admin-controller.js');
const deviceController = require('../controllers/device-controller.js');

const router = express.Router();

// Client POST endpoints
router.post('/client/get/record', adminController.get_client_by_name);
router.post('/client/add/record', adminController.post_client);
router.post('/client/enable/record', adminController.post_enable_client);
router.post('/client/add/role', adminController.post_client_role);
router.post('/client/add/group', adminController.post_client_group);

// Group POST endpoints
router.post('/group/get/record', adminController.get_group_by_name);
router.post('/group/add/record', adminController.post_group);
router.post('/group/add/role', adminController.post_group_role);

// Role POST endpoints
router.post('/role/get/record', adminController.get_role_by_name);
router.post('/role/add/record', adminController.post_role);
router.post('/role/add/acl', adminController.post_acl_role);

// Controller POST endpoints
router.post('/controller/post/record', deviceController.post_controller);

module.exports = router;
```

Figure 3.13: Script pour les fonctions de création

```
const express = require('express');
const adminController = require('../controllers/admin-controller.js');
const deviceController = require('../controllers/device-controller.js');

const router = express.Router();

// Client PUT endpoints
router.put('/client/update/clientId', adminController.put_clientId);
router.put('/client/update/password', adminController.put_client_password);
router.put('/client/update/record', adminController.put_client);

// Group PUT endpoints
router.put('/group/update/record', adminController.put_group);

// Role PUT endpoints
router.put('/group/update/record', adminController.put_role);

// Controller PUT endpoints
router.put('/controller/update/topic', deviceController.put_controller_topic);

module.exports = router;
```

Figure 3.14: Script pour les fonctions de mise à jour

```

const express = require('express');
const adminController = require('../controllers/admin-controller.js');
const deviceController = require('../controllers/device-controller.js');

const router = express.Router();

// Client DELETE endpoints
router.post('/client/delete/record', adminController.delete_client);
router.post('/client/delete/role', adminController.delete_client_role);
router.post('/client/delete/group', adminController.delete_client_group);

// Group DELETE endpoints
router.post('/group/delete/record', adminController.delete_group);
router.post('/group/delete/role', adminController.delete_group_role);

// Role DELETE endpoints
router.post('/role/delete/record', adminController.delete_role);
router.post('/role/delete/acl', adminController.delete_role_acl);

// Controller DELETE endpoints
router.post('/controller/delete/record', deviceController.delete_controller);
router.post(
  '/controller/delete/topic',
  deviceController.delete_controller_topic,
);

module.exports = router;

```

Figure 3.15: Script pour les fonctions de suppression

### 3.6.3 Script pour l'Interaction avec les Dispositifs BLE

Comme mentionné précédemment, un programme en arrière-plan, développé en Node.js, est installé afin d'interagir avec les appareils BLE. Examinons ce code attentivement.

```

var noble = require('@abandonware/noble');
console.log('Scanning for BLE devices...');

var devicesToConnect = [];

noble.on('stateChange', function(state) {
  if (state === 'poweredOn') {
    console.log('Bluetooth is active, starting scan...');
    noble.startScanning();
    setTimeout(function() {
      console.log('Scan finished.');
```

```

      noble.stopScanning();
    }, 4000);
  } else {
    console.log('Bluetooth is not active');
    noble.stopScanning();
  }
});

noble.on('discover', function(peripheral) {
  if (peripheral.advertisement.localName !== undefined) {
    console.log('Peripheral found:');
    console.log('  Local Name: ' + peripheral.advertisement.localName);
    console.log('  UUID: ' + peripheral.uuid);
    console.log('  MAC Address: ' + peripheral.address);
    console.log('  Service UUIDs: ' + peripheral.advertisement.serviceUuids);
    console.log();

    devicesToConnect.push({
      peripheral: peripheral,
      isConnected: false
    });
  }
});

```

Figure 3.16: Code en arrière-plan BLE

```
function connectToDevice(deviceInfo) {
  var selectedDevice = deviceInfo.peripheral;
  selectedDevice.connect(function(error) {
    if (error) {
      console.error('Error connecting to peripheral:', error);
      return;
    }
  });
  console.log('Connected to peripheral:', selectedDevice.uuid, 'Local Name:', selectedDevice.advertisement.localName);
  deviceInfo.isConnected = true;
  discoverServicesAndCharacteristics(selectedDevice);
};
```

**Figure 3.17:** Suite de code en arrière-plan pour BLE

```
function discoverServicesAndCharacteristics(selectedDevice) {
  selectedDevice.discoverAllServicesAndCharacteristics(function(error, services, characteristics) {
    if (error) {
      console.error('Error discovering services and characteristics:', error);
      selectedDevice.disconnect();
      return;
    }

    console.log('Services and characteristics discovered for device:', selectedDevice.uuid);

    // Loop through characteristics
    characteristics.forEach(function(characteristic) {
      // Check if the characteristic supports notifications
      if (characteristic.properties.includes('notify')) {
        var newValue = Buffer.from('OFF', 'utf-8');
        writeCharacteristic(characteristic, newValue);
        // Subscribe to notifications
        characteristic.subscribe(function(error) {
          if (error) {
            console.error('Error subscribing to characteristic:', error);
          } else {
            console.log('Subscribed to characteristic:', characteristic.uuid);
          }
        });
      }

      // Listen for characteristic value changes
      characteristic.on('data', function(data, isNotification) {
        if (isNotification) {
          console.log('Characteristic value changed for', characteristic.uuid + ':', data.toString('utf-8'));
          // If it's a notification, read the new value
          readCharacteristic(characteristic);
        }
      });
    });
  });
};
```

**Figure 3.18:** Suite de code en arrière-plan pour BLE

```
function readCharacteristic(characteristic) {
  characteristic.read(function(error, data) {
    if (error) {
      console.error('Error reading characteristic:', error);
    } else if (data) {
      try {
        const value = data.toString('utf-8');
        console.log('Read characteristic value for', characteristic.uuid + ':', value);
      } catch (e) {
        console.error('Error converting data to string:', e);
      }
    } else {
      console.log('No data received for characteristic:', characteristic.uuid);
    }
  });
}

function writeCharacteristic(characteristic, value) {
  characteristic.write(value, true, function(error) {
    if (error) {
      console.error('Error writing to characteristic:', error);
    } else {
      console.log('Write successful');
    }
  });
}

}
```

**Figure 3.19:** Suite de code en arrière-plan pour BLE



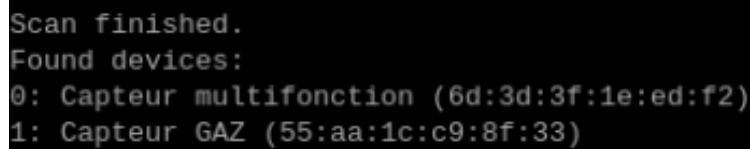
## 3.7 Test et Résultat

### 3.7.1 Évaluation du script Node.js avec @abandonware/noble

Nous examinons dans cette partie le script en arrière-plan développé en Node.js, utilisant la bibliothèque @abandonware/noble, pour vérifier s'il satisfait aux attentes et remplit correctement ses fonctions.

#### 3.7.1.1 Balayage des appareils

Le script effectue le balayage et affiche les appareils détectés [3.20](#).



```
Scan finished.  
Found devices:  
0: Capteur multifonction (6d:3d:3f:1e:ed:f2)  
1: Capteur GAZ (55:aa:1c:c9:8f:33)
```

Figure 3.20: Résultat après le Balayage

#### 3.7.1.2 Connexion vers les appareils

Le script applique la connexion aux appareils de manière appropriée [3.21](#).

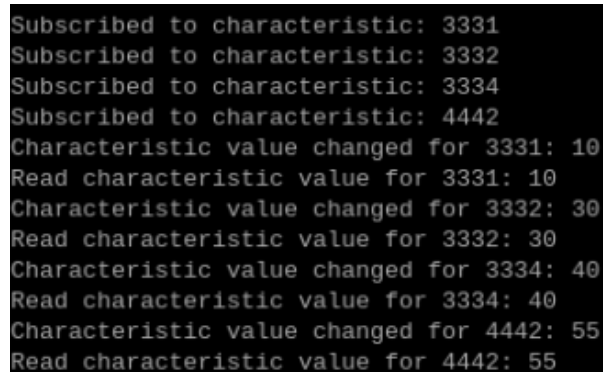


```
Connected to peripheral: 6d3d3f1eedef2 Local Name: Capteur multifonction  
Connected to peripheral: 55aa1cc98f33 Local Name: Capteur GAZ
```

Figure 3.21: Appareils Connectés

#### 3.7.1.3 Abonnement, Notification et Lecture

Le script prend en charge de manière efficace les abonnements, les notifications et les lectures [3.22](#).



```
Subscribed to characteristic: 3331  
Subscribed to characteristic: 3332  
Subscribed to characteristic: 3334  
Subscribed to characteristic: 4442  
Characteristic value changed for 3331: 10  
Read characteristic value for 3331: 10  
Characteristic value changed for 3332: 30  
Read characteristic value for 3332: 30  
Characteristic value changed for 3334: 40  
Read characteristic value for 3334: 40  
Characteristic value changed for 4442: 55  
Read characteristic value for 4442: 55
```

Figure 3.22: Détection des Changements de Valeurs

### 3.7.2 Évaluation du script Node.js avec @MQTT.js et l'interface web :

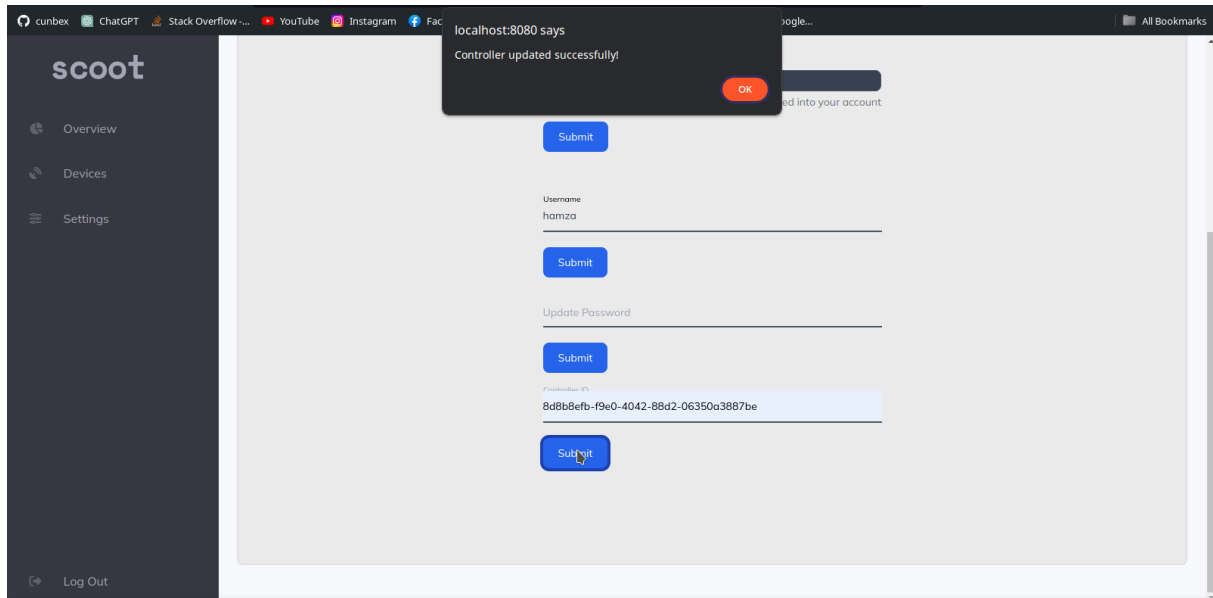


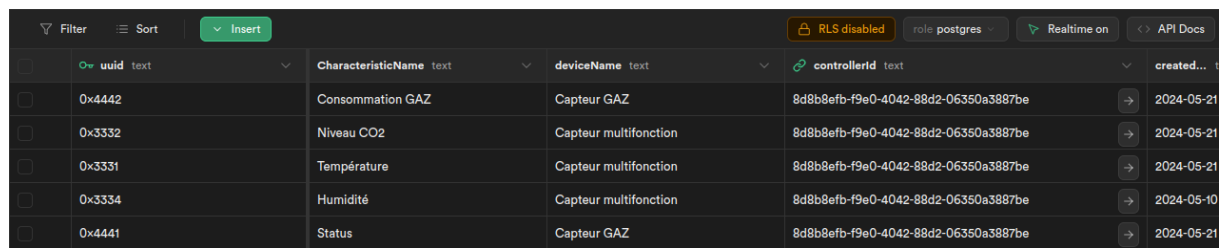
Figure 3.23: Page settings dans le tableau de bord de l'utilisateur

Device Name	Characteristic Name	Characteristic ID	Controller ID	Updated At	Option
Capteur GAZ	Consommation GAZ	0x4442	05a6bc5b-09e3-4303-93bd-3b02b2b854e4	2024-05-21 01:37:24.002	55 WH
Capteur GAZ	Status	0x4441	05a6bc5b-09e3-4303-93bd-3b02b2b854e4	2024-05-21 01:37:24.002	On
Capteur multifonction	Humidité	0x3334	05a6bc5b-09e3-4303-93bd-3b02b2b854e4	2024-05-21 01:37:24.002	40%
Capteur multifonction	Niveau CO2	0x3332	05a6bc5b-09e3-4303-93bd-3b02b2b854e4	2024-05-21 01:22:24.002	30 PPM
Capteur multifonction	Température	0x3331	05a6bc5b-09e3-4303-93bd-3b02b2b854e4	2024-05-21 17:47:24.002	10°C

Figure 3.24: Page devices dans le tableau de bord de l'utilisateur

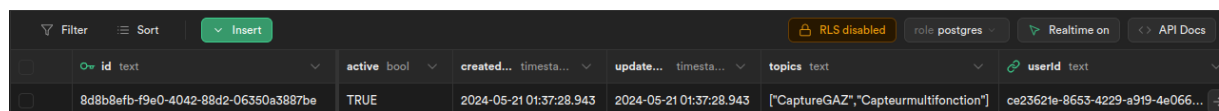
id	email	name	password	created...	update...
ce23621e-8653-4229-a919-4e0661b570fa	hamza@gmail.com	hamza	\$2b\$10\$JCEBxGyqdrOFdxvz0.sUaedK3S	2024-03-23 01:59:18.756	2024-

Figure 3.25: La table des utilisateurs après la création du compte



	uuid	CharacteristicName	deviceName	controllerid	created...
	0x4442	Consommation GAZ	Capteur GAZ	8d8b8efb-f9e0-4042-88d2-06350a3887be	2024-05-21
	0x3332	Niveau CO2	Capteur multifonction	8d8b8efb-f9e0-4042-88d2-06350a3887be	2024-05-21
	0x3331	Température	Capteur multifonction	8d8b8efb-f9e0-4042-88d2-06350a3887be	2024-05-21
	0x3334	Humidité	Capteur multifonction	8d8b8efb-f9e0-4042-88d2-06350a3887be	2024-05-10
	0x4441	Status	Capteur GAZ	8d8b8efb-f9e0-4042-88d2-06350a3887be	2024-05-21

Figure 3.26: La table Devices après l'appairage avec les appareils



	id	active	created...	update...	topics	userid
	8d8b8efb-f9e0-4042-88d2-06350a3887be	TRUE	2024-05-21 01:37:28.943	2024-05-21 01:37:28.943	["CaptureGAZ","Capteurmultifonction"]	ce23621e-8653-4229-a919-4e066...

Figure 3.27: La table des contrôleurs après l'appairage de l'ID du contrôleur avec le compte utilisateur

### 3.8 Conclusion

Dans ce chapitre, nous étudions en profondeur les logiciels, applications, technologies, bibliothèques, équipements et scripts qui sont utilisés pour mettre en œuvre notre projet de maison connectée.

Nous avons aussi étudié l'utilisation de nRF Connect afin de simuler des appareils BLE sur des téléphones Android, ce qui a enrichi notre expérience en créant deux appareils différents, chacun offrant des fonctionnalités spécifiques.

En outre, les scripts créés pour interagir avec le protocole MQTT et les dispositifs BLE ont été minutieusement élaborés afin de garantir une communication fiable entre les divers éléments de notre système. Grâce à cette intégration harmonieuse, une maison intelligente fonctionnelle et adaptable a été créée, mettant en évidence les possibilités et les bénéfices de l'Internet des objets (IoT) dans le domaine de la résidence.

# Conclusion Générale

Dans le domaine de la maison intelligente, la gestion centralisée et le contrôle à distance sont des stratégies clés pour les maisons du futur. Cette étude vise à fournir une définition générale des applications de l'Internet of things (IoT) dans le contexte de la maison intelligente, en se concentrant particulièrement sur la gestion intelligente des appareils de manière authentifiée et sécurisée.

En adoptant une approche basée sur l'IoT, notre projet vise à optimiser l'efficacité des appareils en collectant des données précises en temps réel à l'aide d'un système embarqué comme contrôleur de la maison. Ces données sont ensuite utilisées pour gérer précisément les appareils, soit en lisant et écrivant les valeurs directement par le client à distance, soit en les utilisant pour d'autres dispositifs IoT afin de créer une maison intelligente basée sur des événements, évitant ainsi le gaspillage et la surconsommation inutile.

L'utilisation de logiciels et d'applications mobiles, combinée à la technologie Bluetooth Low Energy (BLE), facilite la surveillance et la gestion de notre maison intelligente, offrant une solution pratique et accessible. Grâce à ces outils, les utilisateurs peuvent visualiser les données collectées, configurer les données et états des appareils personnalisés, et recevoir des alertes en cas de besoins spécifiques ou de conditions anormales directement sur leurs appareils mobiles.

En mettant l'accent sur des pratiques de maison intelligente efficaces et durables, notre projet contribue à minimiser la consommation d'énergie et à promouvoir une grande accessibilité à nos appareils en un clic. En optimisant l'efficacité de la maison intelligente, nous contribuons à un modèle de maison plus durable et sécurisé.

# Bibliographie

- [1] L. Racim. Système de contrôle et d'acquisition de données à travers les réseaux mobiles GSM/4G. Mémoire du Diplôme de Master en ANNABA : UNIVERSITÉBADJI MOKHTAR- ANNABA., 2022. [Online ; accessed 19-03-2024].
- [2] B. Sabrine. Conception d'un habitat intelligent (smart house). Mémoire de Master en Guelma : Université 08 Mai 1945 de Guelma, 2022. [Online ; accessed 15-03-2024].
- [3] Interface-Z. Interaction temps réel : des capteurs aux actionneurs. <https://www.interface-z.fr/formation/cours/capteurs-et-actionneurs/>, 01/12/2023. [Online ; accessed 20-03-2024].
- [4] A. Kamal A. Shadi, A. Mohammed and A. Mahmood. Internet of Things (IoT) Communication Protocols. [https://www.researchgate.net/profile/Kamal-Alieyan/publication/320614944\\_Internet\\_of\\_Things\\_IoT\\_Communication\\_Protocols\\_Review/links/59f06dfeaca272cdc7ca2a64/Internet-of-Things-IoT-Communication-Protocols-Review.pdf/](https://www.researchgate.net/profile/Kamal-Alieyan/publication/320614944_Internet_of_Things_IoT_Communication_Protocols_Review/links/59f06dfeaca272cdc7ca2a64/Internet-of-Things-IoT-Communication-Protocols-Review.pdf/), 07/2017. [Online ; accessed 12-03-2024].
- [5] M. Abet. Différence entre Bluetooth et Bluetooth Low Energy. <https://elainnovation.com/difference-entre-bluetooth-et-bluetooth-low-energy/>, 01/12/2021. [Online ; accessed 11-03-2024].
- [6] K. Terrachet S. Adrar. Conception et Implémentation d'un capteur BLE connecté à une application de bureau. Mémoire de License GTR USTHB, 07/2022. [Online ; accessed 18-03-2024].
- [7] MQTT. What is MQTT ? <https://mqtt.org/faq/>, 2022. [Online ; accessed 18-03-2024].
- [8] EMQX Team. MQTT Broker : How It Works, Popular Options, and Quickstart. <https://www.emqx.com/en/blog/the-ultimate-guide-to-mqtt-broker-comparison>, 01/01/2024. [Online ; accessed 18-03-2024].

- [9] AWS. Qu'est-ce que le protocole MQTT. <https://aws.amazon.com/fr/what-is/mqtt/#:~:text=MQTT%20is%20a%20standards%2Dbased,constrained%20network%20with%20limited%20bandwidth>. [Online; accessed 23-03-2024].
- [10] HiveMQ. MQTT Topics, Wildcards, Best Practices – MQTT Essentials : Part 5. <https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>, 20/02/2024. [Online; accessed 12-03-2024].
- [11] HiveMQ. MQTT Publish/Subscribe Architecture (Pub/Sub). <https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/>, 06/06/2023. [Online; accessed 15-03-2024].
- [12] MQTT. MQTT Publish / Subscribe Architecture. <https://mqtt.org/>, 2022. [Online; accessed 10-03-2024].
- [13] Internet Engineering Task Force (IETF). International standard rfc6455 : The WebSocket Protocol. <https://datatracker.ietf.org/doc/html/rfc6455#section-1>, 2011. [Online; accessed 11-03-2024].
- [14] WikipediA. WebSocket. <https://en.wikipedia.org/wiki/WebSocket>, 12/02/2024. [Online; accessed 14-03-2024].
- [15] A.A. Jerraya. Long term trends for embedded system design. In *Euromicro Symposium on Digital System Design, 2004. DSD 2004.*, pages 20–26, 2004.
- [16] Indian Institute of Embedded Systems (IIES). What is the Role of IoT in Embedded Systems? <https://iies.in/blog/what-is-the-role-of-iot-in-embedded-systems/>. [Online; accessed 28-03-2024].

# Annexe A

## Déploiement et installation du système d'exploitation :

### A.1 PaaS utilisés :

Les services PaaS délivre une environnement complet de développement et de déploiement dans le cloud, avec des ressources qui vous permettent de livrer tout, des applications simples basées sur le cloud aux applications d'entreprise sophistiquées activées par le cloud.

- **Github** : Est une plateforme pour les développeurs qui leur permet de créer, stocker, gérer et partager leur code. Cela nous aide à faire du contrôle de version, à collaborer sur notre projet et à gérer les bugs et les corrections de manière structurée.
- **Supabase** : Est une infrastructure de base de données open-source construite sur PostgreSQL. Ce service nous aide en fournissant un plan gratuit et illimité pour héberger notre base de données sur le cloud.
- **Render** : Est un cloud unifié pour construire et exécuter toutes vos applications et sites web avec des certificats TLS gratuits, un CDN mondial, des réseaux privés et des déploiements automatiques depuis GitHub. Cela nous aide à héberger notre backend et notre frontend avec un nom de sous-domaine illimité et gratuit.
- **DockerHub** : Comme GitHub, DockerHub est un registre de conteneurs conçu pour les développeurs et les contributeurs open source afin de trouver, utiliser et partager leurs images de conteneurs. Cela nous aide à déployer notre conteneur de protocole de messagerie, à en assurer le contrôle de version et à gérer les bugs de manière structurée.

## A.2 Système d'Exploitation sur Raspberry Pi :

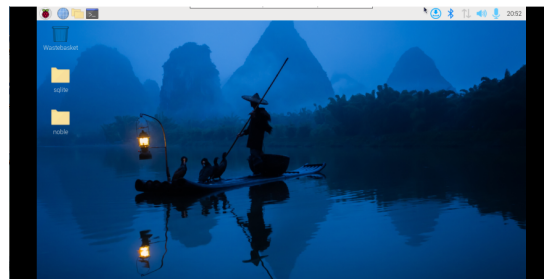
Nous avons procédé à l'installation de Raspbian sur un Raspberry Pi en utilisant Windows en suivant les étapes suivantes :

- Nous avons obtenu l'image du Raspberry Pi OS sur le site officiel de Raspberry Pi.
- Par la suite, nous avons introduit notre carte SD dans notre ordinateur et effectué le formatage.
- Lorsque nous avons démarré Raspberry Pi Imager, nous avons choisi "Système d'exploitation".
- Le modèle de Raspberry Pi que nous avons sélectionné est le "Raspberry Pi OS (64-bit)".



**Figure A.1:** Raspberry pi Imager

- Nous avons sélectionné "Écrire" afin de commencer à écrire l'image sur la carte SD
- Après avoir terminé l'écriture, nous avons retiré la carte SD de manière sécurisée.
- Par la suite, nous l'avons placée dans notre Raspberry Pi et l'avons connectée.
- En respectant les consignes affichées, nous avons achevé la configuration initiale du système d'exploitation Raspberry Pi.



**Figure A.2:** Bureau Raspberry pi 4



# Annexe B

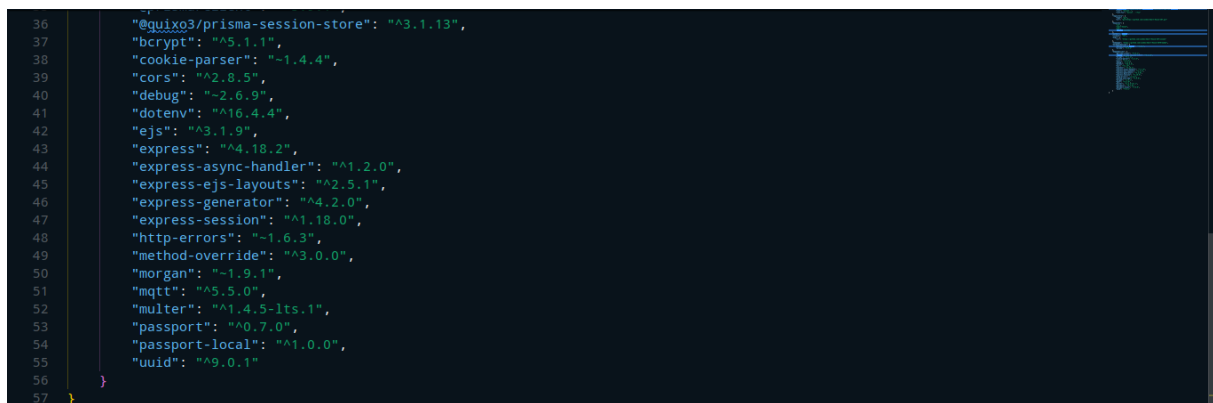
## Configuration Node.js :

### B.1 Fichier Package.json :

Le fichier ‘package.json’ est le cœur de tout projet Node. Il enregistre des métadonnées importantes sur un projet, nécessaires avant la publication sur NPM, et définit également des attributs fonctionnels d’un projet que npm utilise pour installer des dépendances, exécuter des scripts et identifier le point d’entrée de notre package.

```
1 {
2   "name": "pfe",
3   "version": "1.0.0",
4   "description": "license year project",
5   "main": "app.js",
6   "private": true,
7   "scripts": {
8     "start": "node ./bin/www",
9     "devstart": "nodemon ./bin/www",
10    "serverstart": "DEBUG=express-locallibrary-tutorial:* npm run devstart",
11    "lint": "eslint .",
12    "lint:fix": "eslint . --fix"
13  },
14  "repository": {
15    "type": "git",
16    "url": "git+https://github.com/cunbex/Smart-House-IOT.git"
17  },
18  "keywords": [
19    "iot",
20    "smart-house",
21    "web",
22    "embedded-systems"
23  ],
24  "author": "cunbex",
25  "license": "ISC",
26  "bugs": {
27    "url": "https://github.com/cunbex/Smart-House-IOT/issues"
28  },
29  "homepage": "https://github.com/cunbex/Smart-House-IOT#readme",
30  "devDependencies": {
31    "eslint-config-wesbos": "^4.0.1",
32    "prisma": "^5.9.1"
33  },
34  "dependencies": {
35    "@prisma/client": "^5.9.1",
36    "@quixo3/prisma-session-store": "^3.1.13",
```

Figure B.1: Fichier ‘package.json’



```

36   "@quixo3/prisma-session-store": "^3.1.13",
37   "bcrypt": "^5.1.1",
38   "cookie-parser": "~1.4.4",
39   "cors": "^2.8.5",
40   "debug": "~2.6.9",
41   "dotenv": "^16.4.4",
42   "ejs": "^3.1.9",
43   "express": "^4.18.2",
44   "express-async-handler": "^1.2.0",
45   "express-ejs-layouts": "^2.5.1",
46   "express-generator": "^4.2.0",
47   "express-session": "^1.18.0",
48   "http-errors": "~1.6.3",
49   "method-override": "^3.0.0",
50   "morgan": "~1.9.1",
51   "mqtt": "^5.5.0",
52   "multer": "^1.4.5-lts.1",
53   "passport": "^0.7.0",
54   "passport-local": "^1.0.0",
55   "uuid": "^9.0.1"
56 }
57

```

Figure B.2: Suite du fichier 'package.json'

## B.2 Installation et utilisation des bibliothèques :

### — @abandonware/Noble :

```
npm install @abandonware\noble
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const noble = require("@abandonware/noble");
```

### — @prisma/client :

```
npm install @prisma/client
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
import PrismaClient from '@prisma/client';
```

```
const prisma = new PrismaClient();
```

### — @quixo3/prisma-session-store :

```
npm i @quixo3/prisma-session-store
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const expressSession = require('express-session');
```

```
const PrismaSessionStore = require('@quixo3/prisma-session-store');
```

```
const PrismaClient = require('@prisma/client');
```

### — @bcrypt :

```
npm i bcrypt
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const bcrypt = require('bcrypt');
```

— **@cookie-parser :**

npm i cookie-parser

**Pour l'utiliser il faut la déclarer comme suit :**

var cookieParser = require('cookie-parser');

— **@cors :**

npm i cors

**Pour l'utiliser il faut la déclarer comme suit :**

var cors = require('cors');

— **@debug :**

npm i debug

**Pour l'utiliser il faut la déclarer comme suit :**

var debug = require('debug')('http')

, http = require('http')

, name = 'My App';

— **@dotenv :**

npm install dotenv --save

**Pour l'utiliser il faut la déclarer comme suit :**

require('dotenv').config();

— **@Ejs :**

npm install ejs

**Pour l'utiliser il faut la déclarer comme suit :**

app.set('view engine', 'ejs');

— **@Express :**

npm install express --save

**Pour l'utiliser il faut la déclarer comme suit :**

const express = require("express");

const app = express();

— **@Express :**

npm install express --save

**Pour l'utiliser il faut la déclarer comme suit :**

const express = require("express");

```
const app = express();
```

— **@Express-async-handler :**

```
npm i express-async-handler
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const asyncHandler = require('express-async-handler');
```

— **@Express-ejs-layouts :**

```
npm i express-ejs-layouts
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
var expressLayouts = require('express-ejs-layouts');
```

— **@Express-generator :**

```
npm i express-generator
```

**Pour l'utiliser il faut executer cette command en terminal :**

```
npx express-generator
```

— **@Express-session :**

```
npm i express-session
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
var session = require('express-session');
```

— **@Http-errors :**

```
npm i http-errors
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
var createError = require('http-errors');
```

— **@Method-overrides :**

```
npm i method-override
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
var methodOverride = require('method-override');
```

— **@Morgan :**

```
npm i morgan
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
var morgan = require('morgan');
```

— **@Mqtt.js :**

```
npm i mqtt
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const mqtt = require("mqtt");
```

— **@Multer :**

```
npm i multer
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const multer = require('multer');
```

— **@Passport :**

```
npm i passport
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const passport = require('passport');
```

— **@Passport-local :**

```
npm i passport-local
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
const LocalStrategy = require('passport-local').Strategy;
```

— **@Uuid :**

```
npm i uuid
```

**Pour l'utiliser il faut la déclarer comme suit :**

```
import v4 as uuidv4 from 'uuid';
```