الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research
المدرسة الوطنية العليا للتكنولوجيات المتقدمة
National Higher School of Advanced Technologies

Département Génie Électrique et Informatique Industrielle

Mémoire de fin d'étude en vue de l'obtention du diplôme

Master

-Filière-

Télécommunication

-Spécialité -

Système de télécommunications et réseaux

- Thème -

Conception et développement d'une application web pour la mise à jour à distance d'une passerelle Wifi-NRF24 et la gestion d'une application IoT

Réalisé par

BEGHAMI Akram

SEBAA Hicham

Les membres de Jury :

BOUTERFAS Malika	Président	MCA ENSTA
ABBAD Leila	Examinateur	MAA ENSTA
HABANI Lamia	Examinateur	MAA ENSTA
SAADI Hadjer	Encadrante	MCA ENSTICP
BELLOULA Abdelmalek	Co-Encadrant	R&D Direction innovation Cevital
ZAOUI Abdelhalim	Co-Encadrant	Prof ENSTA

Alger, le 27 / 06 / 24

Année universitaire 2023 – 2024

Conception et développement d'une application web pour la mise à jour à distance d'une passerelle Wifi-NRF24 et la gestion d'une application IoT

SEBAA HICHAM
Département de Génie Electrique et Informatique Industrielle
École Nationale Supérieure des Technologies Avancées
Dergana, Alger
h_sebaa@ensta.edu.dz

BEGHAMI AKRAM
Département de Génie Electrique et Informatique Industrielle
École Nationale Supérieure des Technologies Avancées
Dergana, Alger
a_beghami@ensta.edu.dz

***Abstract*— Cet article présente la conception et le développement d'une application web pour la mise à jour à distance d'une passerelle WiFi-NRF24 utilisant un ESP32 et la gestion d'une application IoT. Le système utilise Python et Flask pour faciliter les mises à jour OTA (Over-The-Air) et la gestion des appareils, offrant ainsi une solution robuste pour la maintenance et la gestion des objets IoT.**

Keywords— IoT, OTA, Passerelle, NRF24.

I. INTRODUCTION

Dans un monde de plus en plus connecté, l'Internet des objets, ou en anglais Internet of Things (IoT) joue un rôle crucial dans la transformation des processus industriels et la gestion des données en temps réel. Cet article présente la conception et le développement d'une application web innovante permettant la mise à jour à distance d'une passerelle Wifi-NRF24 et la gestion d'une application IoT. L'objectif principal est de simplifier la maintenance et la gestion des appareils IoT tout en garantissant une communication fiable et sécurisée. Nous détaillerons les différentes étapes de la conception, les choix technologiques, le développement, l'intégration et la mise en œuvre de cette solution. Le système utilise Python et Flask pour faciliter les mises à jour avec la technologie OTA (Over-The-Air) et la gestion des appareils, offrant ainsi une solution robuste pour la maintenance et la gestion des objets IoT.

II. CONTEXTE ET DEFINITION

A. *Définition et importance de l'IoT*

Un système IoT est un ensemble de dispositifs physiques connectés, munis de capteurs, de logiciels et d'autres technologies qui facilitent la collecte et le partage de données en internet entre plusieurs dispositifs une variété d'objets, tels que des appareils domestiques, des capteurs industriels, des dispositifs portables, et bien plus encore. L'IoT est crucial car il peut transformer les processus et les systèmes en fournissant une connectivité et une intelligence accrues. Il améliore l'efficacité des opérations, facilite la prise de décision en temps réel, et crée de nouvelles possibilités commerciales en créant des produits et services novateurs. En outre, il occupe une place essentielle dans la gestion des infrastructures urbaines, contribuant ainsi à améliorer la qualité de vie grâce à l'automatisation et à la gestion intelligente des ressources [1], [2].

B. *Passerelle IoT*

Dans les réseaux IoT, une passerelle IoT est importante, en tant qu'intermédiaire entre les appareils connectés, tels que les capteurs et les actionneurs, et les systèmes backend, tels que les serveurs cloud [3]. Sa fonction est essentielle pour recueillir, traiter, filtrer et transmettre les données entre les dispositifs IoT et les infrastructures réseau [4]. La passerelle IoT offre également des caractéristiques de sécurité, de gestion de périphériques et de mise à jour en temps réel, ce qui permet une gestion centralisée et sécurisée des dispositifs IoT déployés sur le terrain.

La passerelle IoT développée pour ce projet occupe une position centrale dans la collecte et la transmission des informations dans un contexte industriel. Cette structure est conçue pour intégrer de manière transparente différents protocoles de communication et garantir une interopérabilité efficace entre différents dispositifs IoT. Voici les principales composantes de son architecture :

- **Nœuds de Capteurs :** Disposés à des emplacements stratégiques, ces nœuds collectent des données environnementales et les transmettent à la passerelle.
- **Passerelle Centrale :** Agissant comme un hub, elle gère les communications entre les nœuds de capteurs et le cloud. La passerelle utilise des technologies RF (Radio Fréquence) pour assurer une transmission de données rapide et fiable.
- **Cloud :** Les données collectées par la passerelle sont publiées dans le cloud pour permettre une surveillance et une analyse en temps réel.

La passerelle Wifi-NRF24 est essentiel pour assurer le bon déroulement du système IoT. Elle assure la collecte et la transmission des données, en recevant les informations des nœuds de capteurs par voie radio et en les transmettant au cloud via une connexion Wi-Fi. La passerelle assure la gestion de la communication en synchronisant les transmissions et en gérant les données entrantes et sortantes, assurant ainsi un transfert fluide et sans défaut. La passerelle intègre des protocoles solides en matière de sécurité afin de préserver les données transmises contre les personnes non autorisées. En outre, sa conception souple et évolutive permet l'intégration de nouveaux nœuds de

capteurs et l'intégration de nouvelles fonctionnalités sans avoir besoin de modifications matérielles importantes, garantissant ainsi une adaptabilité et une évolutivité optimales pour répondre aux besoins changeants des environnements industriels.

La passerelle a été conçue autour du microcontrôleur ESP32, développé par Espressif Systems. L'ESP32 est un microcontrôleur polyvalent et largement utilisé, connu pour son faible coût, sa faible consommation d'énergie et son riche ensemble de fonctionnalités, notamment les capacités Wi-Fi et Bluetooth intégré [5]. Ces caractéristiques font de l'ESP32 un choix idéal pour les applications IoT. Les principales caractéristiques de l'ESP32 sont les suivantes [6]:

- **Processeur double cœur :** Fournit une puissance de traitement accrue pour les applications complexes.
- **Connectivité sans fil :** Prend en charge à la fois le Wi-Fi et le Bluetooth, ce qui permet d'offrir des options de communication polyvalentes.
- **Broches GPIO :** Offre plusieurs broches d'entrée/sortie à usage général pour l'interfaçage avec divers capteurs et actionneurs.
- **Riche ensemble de périphériques :** Comprend la prise en charge matérielle de UART, SPI, I2C, ADC, DAC, etc., permettant l'intégration d'un large éventail de dispositifs externes.

III. TECHNOLOGIE OTA

La technologie Over-the-Air (OTA) permet de mettre à jour le logiciel d'un dispositif à distance via une connexion sans fil. Cette méthode est particulièrement utile dans le domaine de l'IoT où les dispositifs sont souvent déployés dans des environnements difficiles d'accès. Les mises à jour OTA éliminent la nécessité d'une intervention physique pour installer de nouvelles fonctionnalités, corriger des bugs ou améliorer la sécurité [7].

A. Importance et avantages de la technologie OTA

Les mises à jour OTA sont un élément essentiel de la gestion du cycle de vie des appareils IoT. Elles permettent la distribution à distance des mises à jour du micrologiciel sans qu'il soit nécessaire d'accéder physiquement à l'appareil, ce qui est particulièrement avantageux dans les déploiements à grande échelle ou dans les scénarios où les appareils sont situés dans des endroits difficiles d'accès. Les mises à jour OTA offrent plusieurs avantages clés [8]:

- **Gestion à distance :** Les mises à jour OTA permettent aux administrateurs de gérer et de mettre à jour les appareils à partir d'un emplacement centralisé, sans intervention physique.
- **Efficacité des coûts :** Réduction des coûts opérationnels associés aux mises à jour manuelles, tels que les frais de déplacement et les temps d'arrêt.
- **sécurité :** Facilite le déploiement en temps voulu des correctifs et des mises à jour de sécurité afin d'atténuer les vulnérabilités et d'améliorer la sécurité des appareils.
- **Évolutivité :** Prend en charge la gestion d'un grand nombre d'appareils déployés sur différents sites géographiques, en garantissant des performances et des fonctionnalités cohérentes.
- **Expérience de l'utilisateur :** Améliore l'expérience de l'utilisateur en veillant à ce que les appareils soient continuellement mis à jour avec de nouvelles fonctionnalités et améliorations.

Les mises à jour OTA offrent plusieurs avantages significatifs :

- **Mise à jour simplifiée :** Les mises à jour peuvent être installées rapidement et en même temps sur différents appareils, garantissant ainsi la mise à jour de tous les systèmes sans avoir à effectuer des déplacements coûteux et chronophages.
- **Amélioration de la sécurité :** Grâce aux mises à jour régulières, il est possible de corriger rapidement les vulnérabilités de sécurité, ce qui permet de protéger les dispositifs contre les attaques cybernétiques.
- **Évolutivité :** Les dispositifs IoT peuvent bénéficier de la technologie OTA pour intégrer de nouvelles fonctionnalités et améliorer leurs performances sans perturber leur fonctionnement.
- **Réduction des coûts :** Les mises à jour OTA permettent de diminuer les dépenses opérationnelles et d'améliorer l'efficacité des opérations de gestion des dispositifs en éliminant le besoin de maintenance sur site.

B. Processus de mise à jour OTA

En début de processus de mise à jour OTA, le développeur prépare la mise à jour logicielle, puis la télécharge sur un serveur sécurisé pour être disponible pour les dispositifs IoT, qui reçoivent une notification et téléchargent le nouveau logiciel à partir du serveur. Le dispositif vérifie l'intégrité du fichier téléchargé avant l'installation afin de garantir qu'il ne soit pas corrompu. Après validation, la mise à jour est installée et le dispositif redémarre avec le nouveau logiciel et envoie une confirmation au serveur signalant mise à jour réussie 9].

Pour la communication entre la passerelle et le serveur pour effectuer la mise à jour à distance, plusieurs protocoles peuvent être utilisés, comme MQTT, CoAP, et HTTP/HTTPS. Nous avons choisi le protocole HTTPS pour la mise à jour OTA en raison de sa sécurité, sa fiabilité, et sa compatibilité. HTTPS assure la confidentialité et l'intégrité des données grâce au chiffrement SSL/TLS et permet la vérification de l'authenticité du serveur, réduisant ainsi les risques de « man-in-the-middle attacks » . Ce protocole est largement adopté, compatible avec de nombreux dispositifs IoT, et facilite l'intégration avec d'autres services web et API, garantissant une communication robuste et fiable pour les mises à jour OTA.

IV. CONCEPTION D'UNE APPLICATION WEB POUR LA MISE A JOUR A DISTANCE

L'objectif de l'application web est de permettre la mise à jour à distance "OTA" et la gestion des appareils IoT à distance de manière sécurisée et efficace. Nous avons choisi le framework Flask pour le développement de cette application en raison de sa

simplicité, de sa flexibilité et de sa légèreté, ce qui le rend idéal pour les projets IoT.

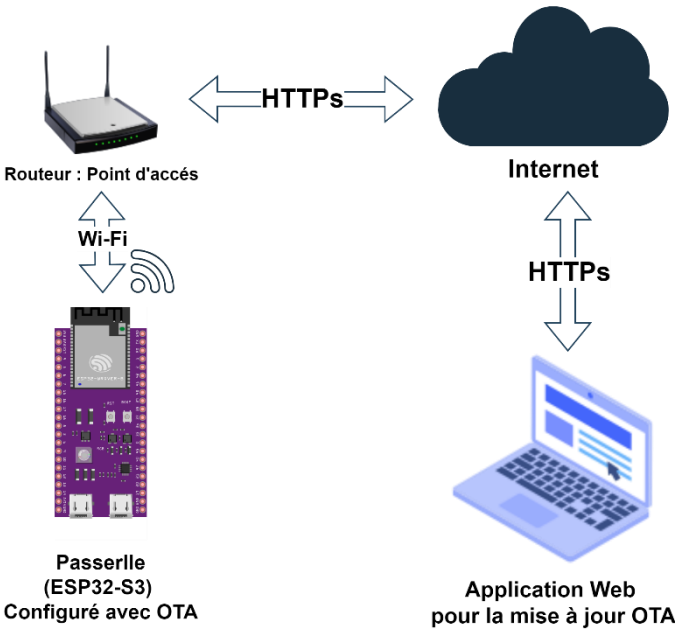

Fig. 1. Schéma de connexion entre la passerelle et l'application web pour la mise à jour à distance

A. *Langages et Frameworks Utilisés*

Pour la conception de l'interface utilisateur, nous avons opté pour HTML et CSS, qui garantissent une structure robuste et un style attrayant. Le Framework Flask, un choix populaire pour les applications web, python en raison de sa légèreté, a été utilisé pour développer l'aspect backend de notre application.

B. *Configuration de l'Application Flask*

Le processus d'installation d'une application Flask comprend la création d'un environnement virtuel, l'installation des dépendances nécessaires, et la configuration de la structure du projet.

1) *Création de l'Environnement Virtuel*

```
➢ Python3 -m venv env
➢ source env/bin/activate
```

2) *Installation des Dépendances*

```
➢ pip install flask
➢ pip install flask_sqlalchemy
➢ pip install flask_login
```

3) *Structure du Projet*

```
├─ app/
│   ├── static/
│   ├── templates/
│   ├── uploads/
│   ├── __init__.py
│   ├── routes.py
│   └── models.py
├── venv/
├── run.py
├── config.py
└── requirements.txt
```

Notre projet est structuré de manière à séparer clairement les différentes parties de l'application, facilitant ainsi la maintenance et l'extension future. Au niveau le plus élevé, nous trouvons le dossier (app/), qui regroupe l'essentiel de l'application Flask. À l'intérieur, les ressources statiques (static/), les templates HTML (templates/), et un dossier pour les fichiers de firmware téléchargés (uploads/) sont soigneusement organisés. Le dossier (venv/) héberge l'environnement virtuel Python, isolant les dépendances du projet. Le script d'entrée run.py permet de démarrer l'application, et config.py gère les paramètres de configuration. Enfin, requirements.txt liste les paquets Python requis, assurant une reproduction facile de l'environnement de développement.

C. *Création du point final de mise à jour a distance*

Un point final dans Flask pour gérer les mises à jour OTA pour l'ESP32 est créé, avec des extraits de code fournis.

```python
@app.route('/', methods=['GET', 'POST'])
def login():
    ...

    return render_template('login.html')


@app.route('/home', methods=['GET', 'POST'])
def upload_file():
    ...

    return render_template('upload.html', message="")


@app.route('/post-data', methods=['POST'])
def post_data():
    ...

    return jsonify({"status": "failed", "error": str(e)}), 400


@app.route('/version', methods=['GET'])
def get_version():
    return send_from_directory(app.config['UPLOAD_FOLDER'], 'version.json')


@app.route('/firmware', methods=['GET'])
def get_firmware():
    return send_from_directory(app.config['UPLOAD_FOLDER'], 'code.bin')
```

Fig. 2. Extrait de code pour la création du point final de mise à jour a distance

D. Gestion des Téléchargements de Fichiers

La gestion des téléchargements et du stockage des fichiers de firmware est essentielle pour le processus OTA. Voici un extrait de code pour gérer cette fonctionnalité :

```
UPLOAD_FOLDER = 'uploads'  # Define the folder where uploaded files will be stored

# Route for uploading files
@app.route('/home', methods=['GET', 'POST'])
def upload_file():

    ...

    if request.method == 'POST':  # Handle POST request

        ...


        if file and allowed_file(file.filename):  # Check if the file type is allowed
            filename = 'code.bin'  # Define the filename to save as
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)  # Define the full file path
            file.save(filepath)  # Save the uploaded file to the specified path
            return render_template('upload.html', message="File successfully uploaded")  # Render upload.html with a success message
    return render_template('upload.html', message="")  # Render upload.html without any message if no POST method or invalid file


# Route for downloading firmware
@app.route('/firmware', methods=['GET'])
def get_firmware():
    return send_from_directory(app.config['UPLOAD_FOLDER'], 'code.bin')  # Send the 'code.bin' file from the UPLOAD_FOLDER directory
```

Fig. 3. Extrait de code pour la gestion de téléchargements de fichiers

Définitions des routes : Routes implémentées (routes.py) pour gérer diverses fonctionnalités :

- **/ :** Gère la connexion de l'utilisateur. Valide les identifiants et établit une session lors d'une connexion réussie.
- **/home :** Nécessite une connexion. Gère les téléchargements de fichiers (POST) et affiche l'interface de téléchargement (GET).
- **/post-data :** Reçoit des données JSON pour les informations actuelles sur le micrologiciel et les enregistre dans « version.json ».
- **/version :** Fournit les informations actuelles sur le micrologiciel stockées dans « version.json ».
- **/firmware :** Fournit « code.bin » pour les mises à jour du micrologiciel des dispositifs ESP32.

V. INTERFACE UTILISATEUR POUR LA MISE A JOUR A DISTANCE

Notre application OTA comprend deux interfaces principales, la première interface est la page de connexion, qui offre une expérience utilisateur intuitive et sécurisée. Les utilisateurs peuvent s'authentifier en toute sécurité à l'aide de champs dédiés pour le nom d'utilisateur et le mot de passe. La sécurité des données est assurée par des protocoles de cryptage robustes. La figure 4 illustré l'interface conçue.

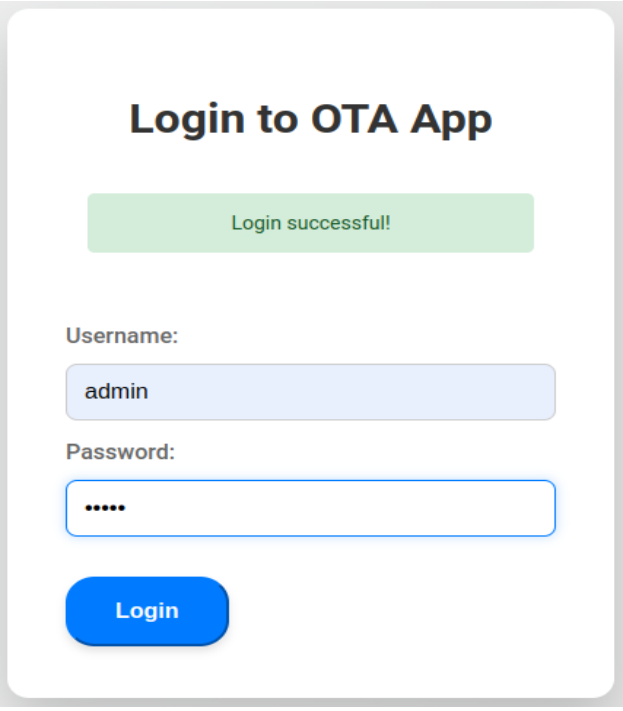

Fig. 4. Interface utilisateur pour la connexion à l'application OTA

La seconde interface est dédiée à la gestion du micro logiciel (Firmware Management). Elle se compose de deux composants distincts :

- **Affichage d'informations sur le firmware** : Ce composant présente de manière interactive et personnalisable des informations complètes sur le firmware actuellement installé sur notre passerelle. Cela inclut les numéros de version, les dates de publication et toutes les notes ou mises à jour pertinentes.
- **Téléchargement des images du micrologiciel :** Nous avons intégré des formulaires dynamiques grâce à Jinja2 pour créer des modèles HTML dynamiques, permettant une présentation interactive et personnalisable des informations. En complément, nous avons utilisé Flask pour gérer efficacement le téléchargement de fichiers de firmware et la gestion des entrées utilisateur de manière sécurisée et fiable. Cette combinaison de technologies garantit une expérience utilisateur robuste, facile à utiliser, tout en assurant la sécurité et l'intégrité des données manipulées par l'application. La figure 5 illustre l'interface conçue.

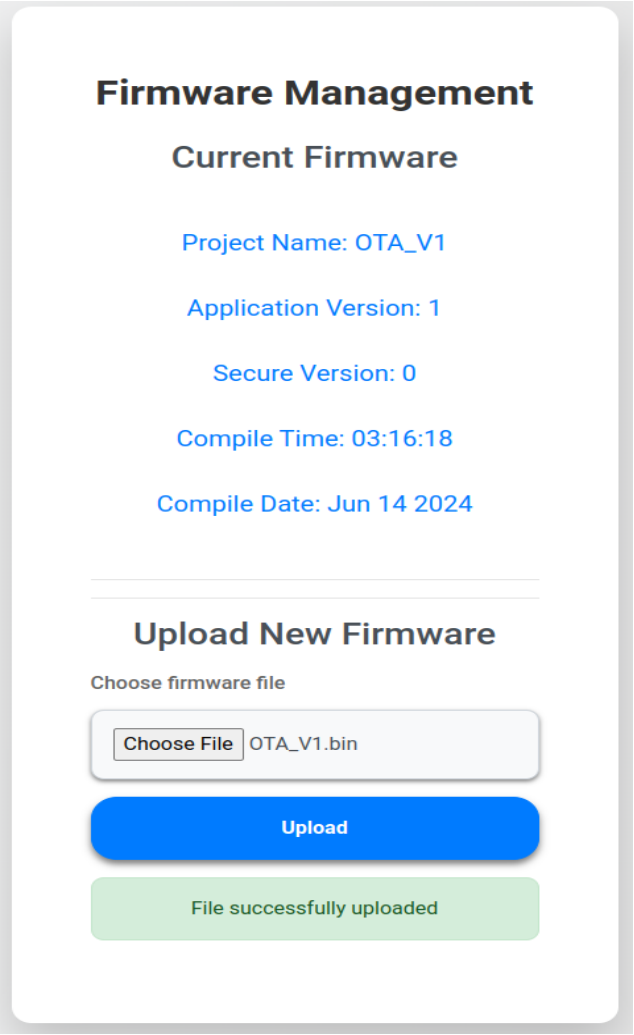

Fig. 5. Interface utilisateur pour la mise à jour a distance

VI. Interaction entre la passerelle et le serveur Flask

Les administrateurs chargent les fichiers de micrologiciel via l'interface web, que Flask stocke de manière sécurisée sur le serveur.

Du côté de la passerelle, le système vérifie toutes les 10 secondes la disponibilité de mises à jour en envoyant des requêtes à l'endpoint du serveur Flask (/firmware). Le serveur répond avec les métadonnées détaillant la dernière version disponible du micrologiciel.

À la réception de ces métadonnées, la passerelle les compare avec sa version actuelle du micrologiciel. Si la version disponible est plus récente, le serveur Flask initie une transmission sécurisée du fichier de micrologiciel vers la passerelle via HTTPS.

Une fois reçu, la passerelle applique la mise à jour en suivant sa logique de mise à jour OTA. Cela inclut la vérification de l'intégrité du micrologiciel, l'application de signatures cryptographiques (si applicable), ainsi que la mise en place d'un mécanisme de retour en arrière en cas d'échec de la mise à jour.

Ce processus garantit que les dispositifs Gateway sont constamment à jour avec les dernières améliorations de fonctionnalité et de sécurité, assurant ainsi une performance optimale et une gestion efficace des mises à jour logicielles. la figure 6 illustre l'organigramme de fonctionnement de la tâche OTA dans la passerelle.

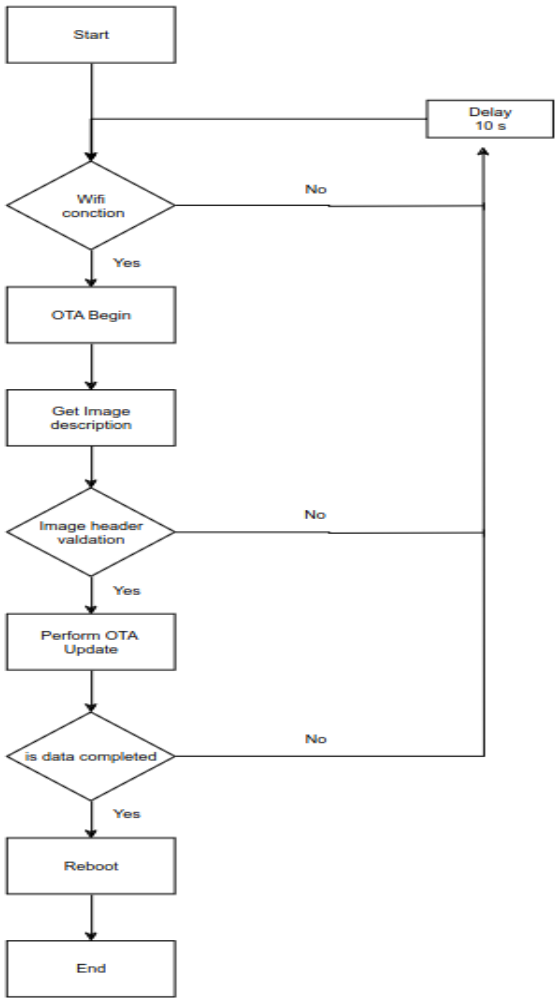

Fig. 6. Organigramme de fonctionnement de la tâche OTA

VII. Application web pour la gestion et la surveillance d'une passerelle IoT

Dans cette section, nous présentons l'application web développée pour la gestion IoT de notre système. Cette application permet de surveiller divers paramètres des nœuds de capteurs et de visualiser les statistiques en temps réel. Elle offre une interface utilisateur intuitive pour la surveillance et la gestion des dispositifs IoT, en utilisant le protocole de communication MQTT pour une transmission fiable et efficace des données. L'application est construite sur la plateforme Node-RED, qui facilite le développement de flux de données et l'intégration de multiples services grâce à son interface de programmation visuelle. Cette approche permet non seulement de simplifier la gestion des nœuds de capteurs mais aussi d'assurer une communication sécurisée et stable au sein du réseau IoT.

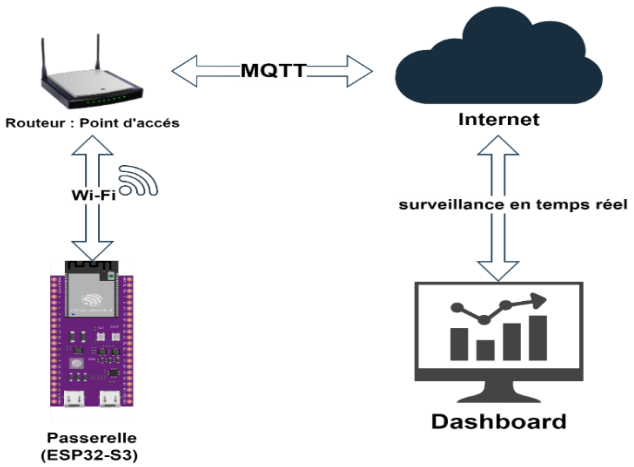

Fig. 7. Schéma du connexion entre la passerelle et l'application de gestion

A. *Protocole MQTT*

Le Message Queuing Telemetry Transport (MQTT) est un protocole de messagerie léger conçu pour les dispositifs contraints en termes de bande passante et de ressources, souvent utilisé dans l'IoT. MQTT fonctionne selon une architecture de type publish/subscribe, où les clients peuvent publier des messages sur des sujets spécifiques et s'abonner à ces sujets pour recevoir les messages pertinents [10]. Voici quelques-unes des principales caractéristiques de MQTT [10] :

1) Légèreté : MQTT est conçu pour minimiser la surcharge réseau, ce qui le rend idéal pour les environnements où la bande passante est limitée. Les messages MQTT sont compacts et utilisent un format binaire efficace, réduisant ainsi la quantité de données transmises. Cela fait de MQTT un choix judicieux pour les applications IoT qui nécessitent une communication fréquente avec des appareils à ressources limitées, comme les capteurs et les actionneurs.

2) Simplicité : La simplicité du protocole MQTT facilite son implémentation sur des dispositifs à ressources limitées. Sa conception légère et son faible nombre de fonctionnalités le rendent facile à apprendre et à intégrer dans les systèmes embarqués.

3) Fiabilité : MQTT offre trois niveaux de qualité de service (QoS) pour garantir la livraison des messages selon les besoins de l'application *:*

- **QoS 0 :** Au plus une fois (envoi sans garantie) : Ce niveau offre la meilleure performance mais ne garantit pas la livraison des messages. Il est adapté aux applications où la perte de données occasionnelle est acceptable.
- **QoS 1 :** Au moins une fois (message garanti d'être livré au moins une fois) : Ce niveau garantit que chaque message est livré au moins une fois. Il est adapté aux applications où la perte de données est possible mais doit être minimisée.
- **QoS 2 :** Exactement une fois (message garanti d'être livré une fois et une seule) : Ce niveau offre la garantie la plus élevée de livraison des messages, assurant qu'aucun message n'est ni perdu ni dupliqué. Il est adapté aux applications critiques où la perte de données est inacceptable.

4) Evolutivité : Grâce à son architecture publish-subscribe, MQTT permet une communication efficace entre un grand nombre de dispositifs. Un seul broker MQTT peut gérer des millions de clients simultanément, ce qui le rend idéal pour les applications IoT à grande échelle.

5) Sécurité : MQTT peut utiliser SSL/TLS pour sécuriser les communications, garantissant ainsi la confidentialité et l'intégrité des données échangées. Cela fait de MQTT un choix approprié pour les applications IoT qui nécessitent un niveau élevé de sécurité, telles que celles impliquant des données sensibles ou des transactions financières.

B. *MQTT broker*

Un MQTT broker est un serveur qui agit comme intermédiaire entre les clients MQTT (Message Queuing Telemetry Transport). Il reçoit des messages des clients et les route vers d'autres clients qui se sont abonnés aux sujets correspondants. Essentiellement, un MQTT broker permet la communication entre les appareils ou les applications utilisant le protocole MQTT, facilitant ainsi l'échange de données efficace et léger dans les scénarios de l'IoT et autres échanges de données en temps réel [10].

1) Configuration du MQTT broker : Cette section décrit comment nous avons configuré un environnement sécurisé pour notre courtier MQTT déployé. Nous avons mis en œuvre les protocoles Secure Sockets Layer (SSL) et Transport Layer Security (TLS) pour assurer la protection des données lors de leur transmission.

```
...
# Mosquitto broker configuration for secure
communication

listener 8883
# Mosquitto listens for connections on port
8883 (standard for secure connections)

cafile/etc/mosquitto/certs/ca.crt
# Path to the Certificate Authority (CA)
certificate file

certfile/etc/mosquitto/certs/mosquitto.crt
# Path to the server certificate file

keyfile/etc/mosquitto/certs/mosquitto.key
#Path to the server private key file

require_certificate false
# Currently configured to allow unencrypted
connections for testing purposes (modify to
true for mandatory encryption)
...
```

C. *Node-Red*

L'interface graphique de Node-RED est une plateforme open source de développement visuel qui permet de concevoir des

applications et des flux de données. La plateforme repose sur le langage "Node.js", ce qui lui permet de fonctionner aisément sur divers systèmes d'exploitation. Elle peut être accédée à toute adresse IP locale.

Il adopte une méthode de "glisser-déposer" pour générer des nœuds et les relier pour créer un flux de données. Chaque nœud correspond à un rôle ou à une action précise, comme la lecture de données provenant d'une source, la transformation des données, leur envoi à une destination ou l'exécution d'une opération particulière. Il est possible de fusionner ces nœuds afin de créer des flux de données complexes et des applications dédiées à l'internet des objets.

Pour la conception de notre interface Node-RED, nous avons créé plusieurs nœuds pour surveiller différents paramètres de nos capteurs. Les figures ci-dessous illustrent les différentes parties de la conception.

1) Nœud de Surveillance des Paramètres : Ce nœud est configuré pour surveiller et afficher en temps réel des paramètres essentiels tels que la température, l'humidité, la tension, et les valeurs RGB. comme est illustré dans la figure

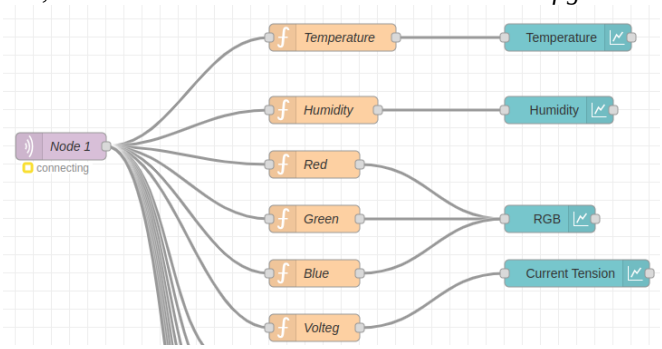

Fig. 8. Flux sur Node-Red pour publication (pour la surveillance des paramètres !)

2) Nœuds de Contrôle : Ces nœuds offrent des fonctionnalités de contrôle, permettant des actions spécifiques comme l'allumage de la LED et l'ajustement de la puissance de transmission.comme est illustré dans la figure.

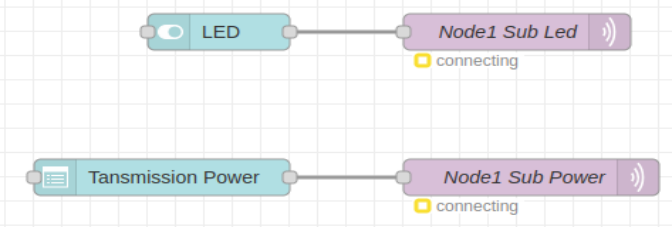

Fig. 9. Flux sur Node-Red pour subscription (pour le contrôle de paramètres)

D. Interface graphique

L'interface graphique de notre système de surveillance est divisée en trois pages distinctes, une page pour la passerelle et deux pages pour les nœuds de capteur. Chaque page offre une vue d'ensemble des informations clés liées à l'élément correspondant.

La page de la passerelle fournit un aperçu complet des performances et de la connectivité de la passerelle, comme est illustré dans la figures 10-12. Elle est structurée en trois principales sections :

1. Sur la partie gauche de la page, une section intitulée Caractéristiques de la Passerelle (Geteway Characteristic) liste les spécifications techniques et les paramètres de configuration de la passerelle. Ces informations permettent de comprendre les capacités et les paramètres de fonctionnement de la passerelle utilisée dans notre système.
2. La section centrale intitulée "Globle Connectivity" (Connectivité Globale) affiche un graphique représentant le "Total Successful Rate" (Taux de Réussite Total) des connexions. Ce graphique est essentiel pour évaluer la fiabilité globale de la passerelle en termes de pourcentage de connexions réussies sur l'ensemble du réseau. Il offre une visualisation en temps réel, ce qui permet de suivre les performances instantanément.
3. Sur la partie droite de la page, la section "Nodes Connectivity" (Connectivité des Nœuds) présente deux graphiques distincts :
   - “Successful Rate Of Connection From Node 1" : Ce graphique montre le taux de réussite des connexions spécifiques au premier nœud de capteur. Une ligne verte indique visuellement les connexions réussies au fil du temps.
   - “Successful Rate Of Connection From Node 2" : De manière similaire, ce graphique affiche le taux de réussite des connexions pour le deuxième nœud de capteur, représenté par une ligne rouge.

1) Ces graphiques permettent de comparer et de vérifier les performances de chaque nœud de capteur individuellement, fournissant des informations précises sur la fiabilité de la communication entre la passerelle et chaque nœud.

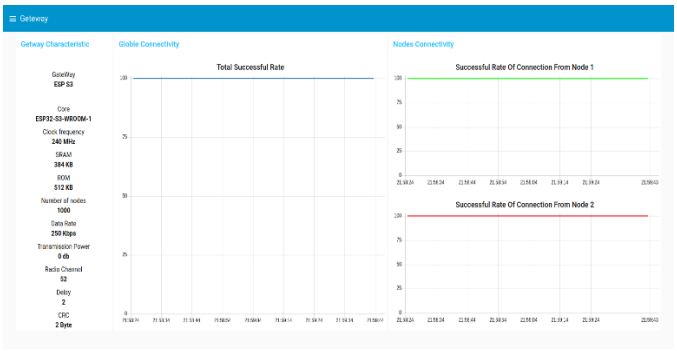

Fig. 10. Page d’acceuil pour la passerelle

Les pages relatives aux nœuds comme est illustré dans la figure 11 et la figure 12 , offrent une vue d'ensemble de l'état, des données et de la configuration de chaque nœud de capteur. Chaque page est divisée en trois sections principales:

- Caractéristiques du nœud, située sur le côté gauche de la page, cette section détaille les caractéristiques matérielles de l'unité de microcontrôleur (MCU) sur laquelle le nœud de capteur est basé. Ces informations sont précieuses pour permettre aux superviseurs de comprendre les capacités et les limites du nœud.
- Données des capteurs en temps réel, occupant la partie centrale de la page, cette section présente des données en temps réel provenant des différents capteurs intégrés au nœud. Les données sont visualisées à l'aide de graphiques afin d'afficher efficacement les courbes au cours du temps. Les données spécifiques du capteur affichées sont la température, l'humidité, la tension et l'intensité lumineuse.
- La section la plus à droite de la page fournit un panneau de commande permettant d'envoyer des ordres au nœud de capteur. Cela permet aux superviseurs d'interagir à distance avec le nœud et de modifier son comportement. Contrôle d'une LED, un interrupteur à bascule ou un bouton pour allumer ou éteindre la LED, et autre pour la commande de la puissance de transmission un menu déroulant pour ajuster la puissance de transmission du nœud.

Fig. 11. Page pour la supervision du premier nœud

Fig. 12. Page pour la supervision du deuxième nœud

VIII. Conclusion

Ce projet a permis de concevoir et de développer une application web efficace pour la mise à jour à distance d'une passerelle Wifi-NRF24 et la gestion d'une application IoT. En exploitant les capacités de l'ESP32 et en utilisant le framework Flask, nous avons réussi à créer une solution robuste qui simplifie la maintenance et améliore la gestion des appareils IoT.

La mise en place des mises à jour OTA via le protocole HTTPS assure la sécurité et la fiabilité de la communication, tandis que l'utilisation de Node-RED pour l'interface graphique permet une surveillance en temps réel des différents paramètres des capteurs. Grâce à cette architecture, la passerelle peut gérer de multiples nœuds de capteurs, garantissant une collecte et une transmission des données optimisées dans des environnements industriels exigeants.

Les résultats obtenus démontrent non seulement l'efficacité de notre passerelle mais ouvrent également la voie à des améliorations futures, notamment en matière d'évolutivité et d'intégration de nouvelles fonctionnalités.

Ce projet illustre ainsi l'importance des passerelles IoT dans la transformation numérique des industries, offrant des solutions innovantes pour améliorer l'efficacité opérationnelle et la gestion des ressources.

References

[1] M. Lekic and G. Gardasevic, "IoT sensor integration to Node-RED platform," in 2018 7th International Symposium INFOTEH-JAHORINA (INFOTEH), 2018.

[2] K. K. Patel, S. M. Patel, and P. Scholar, "Internet of things-IoT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges," Int. J. Eng. Sci. Comput., vol. 6, no. 5, 2016.

[3] S. Seçgin, "Section: Internet of Things (IoT)," in Evolution of Wireless Communication Ecosystems, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2023, pp. 177-189.

[4] L. Juan and A. Acero, "Internet of Things (IoT): The IoT refers to the growing network of devices that are connected to the internet," 2023.

[5] N. Kolban, *Kolban's Book on ESP32*, pp. 60-80, Sep. 2018.

[6] Espressif Systems, "ESP32-S3 Series Datasheet," 2021. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

[7] J. Maria, J. Berriel, and J. Freitag, "Design and Evaluation of a Method for Over-The-Air Firmware Updates for IoT Devices," 2023.

[8] W. Wang, G. Qin, Y. Liang, L. Yang, Y. Wang, and Y. Feng, "An OTA-oriented Protocol for Security Protection," in 2023 3rd International Conference on Frontiers of Electronics, Information and Computation Technologies (ICFEICT), Yangzhou, China, 2023, pp. 129-134, doi: 10.1109/ICFEICT59519.2023.00032.

[9] F. Mahfoudhi, A. Sultania, and J. Famaey, "Over-the-Air Firmware Updates for Constrained NB-IoT Devices," Sensors (Basel, Switzerland), vol. 22, 2022. [Online]. Available: https://doi.org/10.3390/s22197572.

[10] G. C. Hillar, *MQTT Essentials - A Lightweight IoT Protocol*, Packt Publishing, Apr. 2017.