Multi-tasking Embedded Vehicular Platform based on I2C (Master/Multi-Slaves) and MQTT (Cloud/Node) Communication Protocols - Real Time Operating System -

BADRY Zakaria
Filiere : Genie informatique
Kenitra
zakaria.badry@uit.ac.ma

CHAGRI ANASS
Filiere : Genie informatique
Kenitra
anass.chagri@uit.ac.ma

MEZIANE ZERHOUNI HASSAN
Filiere : Genie informatique
Kenitra
hassan.mezianezerhouni@uit.ac.ma

hamza mouhaouire
Filiere : Genie informatique
Kenitra
hamza.mouhaouire@uit.ac.ma

I. INTRODUCTION

Avec l'essor des véhicules intelligents et connectés, il devient essentiel de concevoir des plateformes embarquées capables de gérer efficacement les données et les actions liées à la surveillance et au contrôle du véhicule. Le présent rapport décrit la conception et l'implémentation d'une plateforme véhiculaire embarquée multitâche basée sur les protocoles de communication I2C (Maître/Multi-Esclaves) et MQTT (Nuage/Noeud). Cette plateforme est composée d'un maître, un Raspberry Pi, et trois esclaves Arduino, chacun responsable de tâches bien spécifiques .

II. Cryptage et Décryptage :

La plateforme se distingue par l'emploi de méthodes sophistiquées de chiffrement et de déchiffrement. Elle préserve la confidentialité des données transmises en utilisant les protocoles I2C et MQTT, offrant ainsi une sécurité accrue. Les informations délicates transmises entre les composants internes du véhicule et les noeuds externes sont sécurisées grâce à des algorithmes de cryptage solides, garantissant l'accès exclusif aux utilisateurs autorisés.

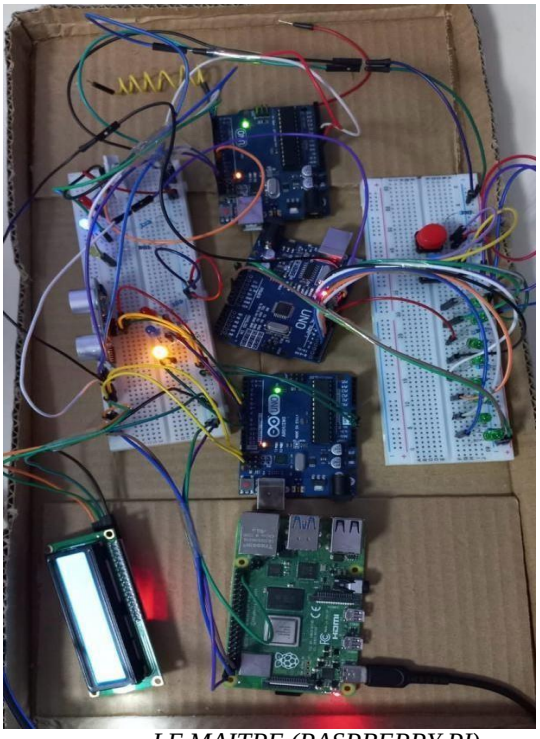
III. OBJECTIF

L'objectif principal de ce projet est de créer une plateforme capable de collecter et de traiter des données provenant de différents capteurs embarqués . Les capteurs sont capables de fournir des informations pertinentes , son environnement, . La plateforme est chargée de recevoir, de stocker, et de traiter ces données de manière efficace .

IV. L'ARCHITECTURE DE LA PLATFORME

La structure fondamentale de cette plateforme repose sur une hiérarchie maître-esclave, où le Raspberry Pi assume le rôle de maître tandis que trois Arduino jouent le rôle d'esclaves. Les échanges d'informations entre le maître et les esclaves sont orchestrés au moyen du protocole I2C, garantissant une communication efficace et bidirectionnelle. Parallèlement, la connectivité entre le nuage (cloud) et le nœud (la plateforme Raspberry Pi et ses esclaves Arduino) est établie grâce au protocole MQTT, permettant une communication fluide et instantanée, enrichissant ainsi la plateforme de fonctionnalités de surveillance et de contrôle avancées.

V.



LE MAITRE (RASPBERRY PI)

Le maître Raspberry Pi a plusieurs fonctions dans la plateforme. Il est responsable de :

- Gérer la communication I2C avec les esclaves Arduino, en leur envoyant des requêtes ou des commandes, et en recevant leurs données.
- Gérer la communication MQTT avec le nuage, en publiant les données des esclaves sur des topics spécifiques,

et en s’abonnant à des topics pour recevoir des instructions ou des alertes du nuage.

- Exécuter le script principal qui coordonne les différentes tâches de la plateforme, en fonction des données reçues et des décisions prises.
- Afficher les données et les actions sur un écran LCD connecté au Raspberry Pi, pour permettre une visualisation locale des données.

Initialisation de l'Application (initApp):

Configure les éléments nécessaires tels qu'un écran LCD, des capteurs I2C, une connexion à un broker MQTT, et un serveur socket pour la réception d'images.
Établit des constantes, incluant les sujets MQTT, les détails du broker, les emplacements des fichiers, etc.
Communication MQTT (mqttCommunicateDataTask):

Envoie des données telles que la température, l'humidité, la distance, la vitesse, les informations de flash et le buffer d'image sur des canaux MQTT dédiés.
Boucle Principale (loopApp):

Exécute plusieurs tâches de manière séquentielle :
Collecte de Données (colletDataTask): Récupère des données depuis des capteurs I2C.
Affichage des Données (displayDataTask): Montre les données sur un écran LCD.
Capture d'Image (captureCameraTask): Capture et publie une image de la caméra en chaîne encodée base64.
Communication MQTT (mqttCommunicateDataTask): Transmet des données via MQTT.
Chiffrement (encryptingDataTask): Crypte et envoie des fichiers spécifiques via MQTT.
Pause (timeSleepTask): Utilise time.sleep pour créer une pause.
Affichage des Données (displayDataTask):

Rassemble et montre des informations telles que température, humidité, distance et flash sur l'écran LCD.
Collecte de Données (colletDataTask):

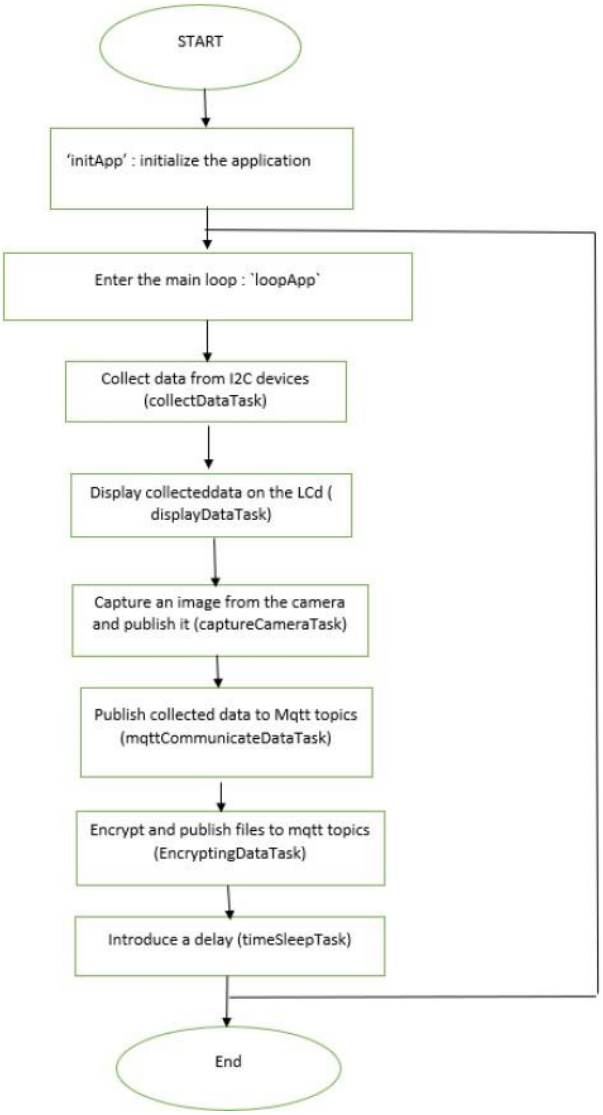
Lit les données de trois capteurs I2C (Flash, DHT et Ultrasonic) et affiche les résultats.
Capture d'Image (captureCameraTask):

Attend une connexion, reçoit et sauvegarde une photo, et envoie l'image codée en base64 au broker MQTT.
Chiffrement (encryptingDataTask):

Produit une clé, crée un fichier texte, le crypte, et diffuse le fichier crypté et la clé sur des canaux MQTT.
Pause (timeSleepTask):

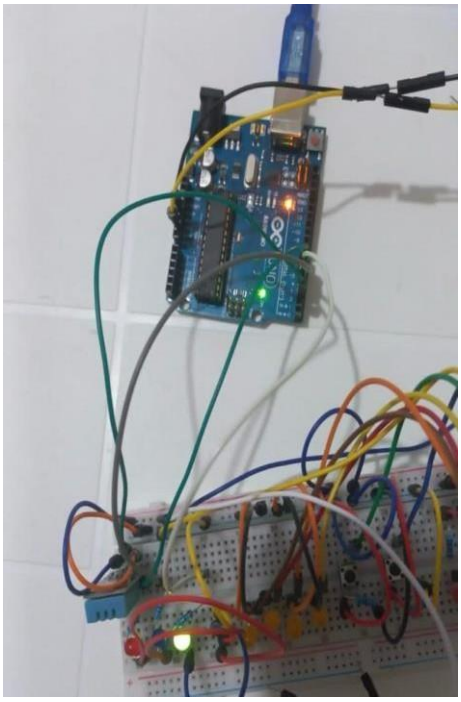
Introduit un temps d'arrêt pour gérer la fréquence de traitement des données.

SCHEMA ALGORITHMIQUE : CODE MASTER

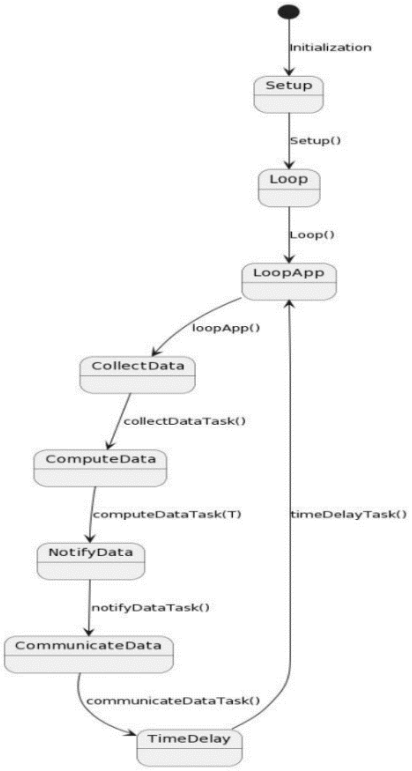


VI. L'ESCLAVE 1

Le premier esclave est équipé d'un capteur DHT11 permettant de mesurer la température et l'humidité. Ces données sont transmises au maître via le protocole I2C.



SCHEMA ALGORITHMIQUE : CODE DHT



V. L'ESCLAVE 2

Le deuxième esclave est équipé d'un capteur ultrasonique pour mesurer la distance et la vitesse du véhicule. Il contrôle également trois lampes de signalisation (bleue pour loin, verte pour moyenement proche , rouge pour tres proche) en fonction de la distance mesurée.

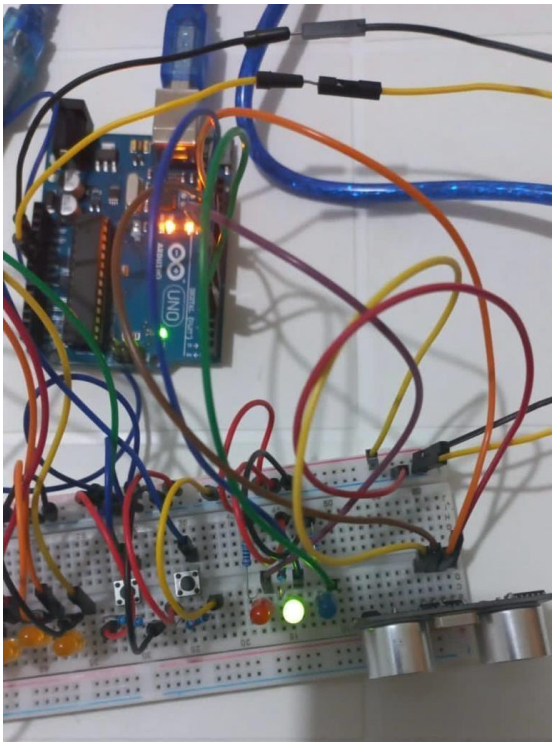
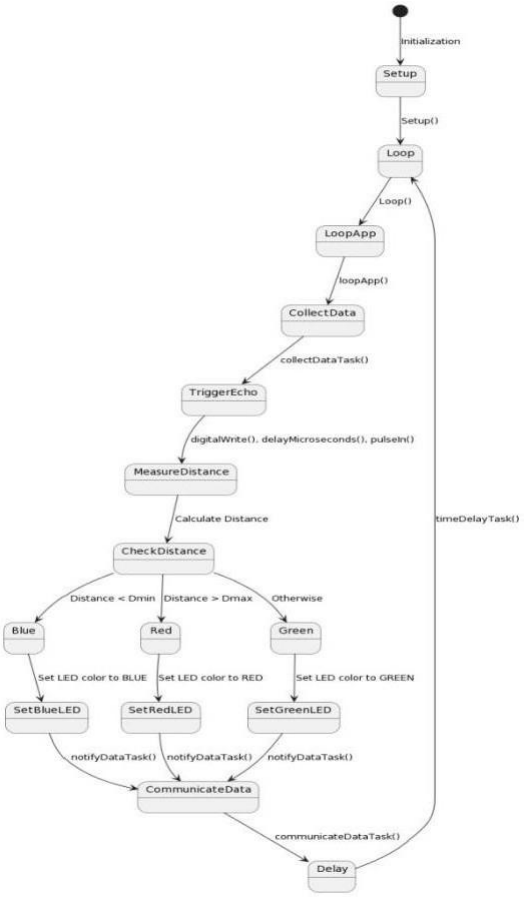


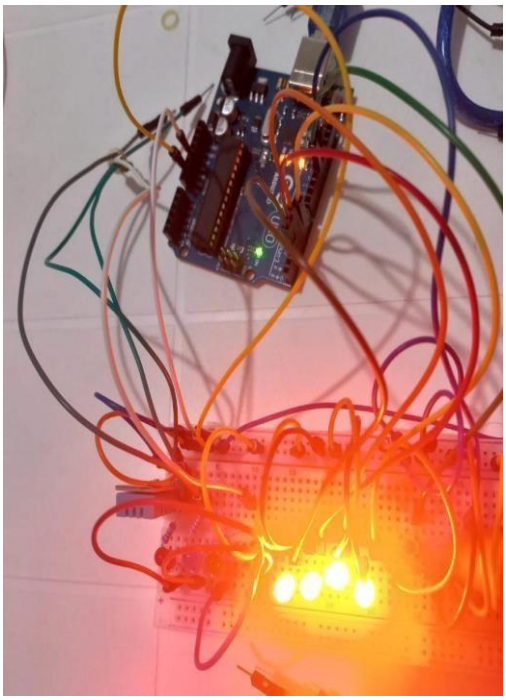
Schéma algorithmique : code ultrasonic



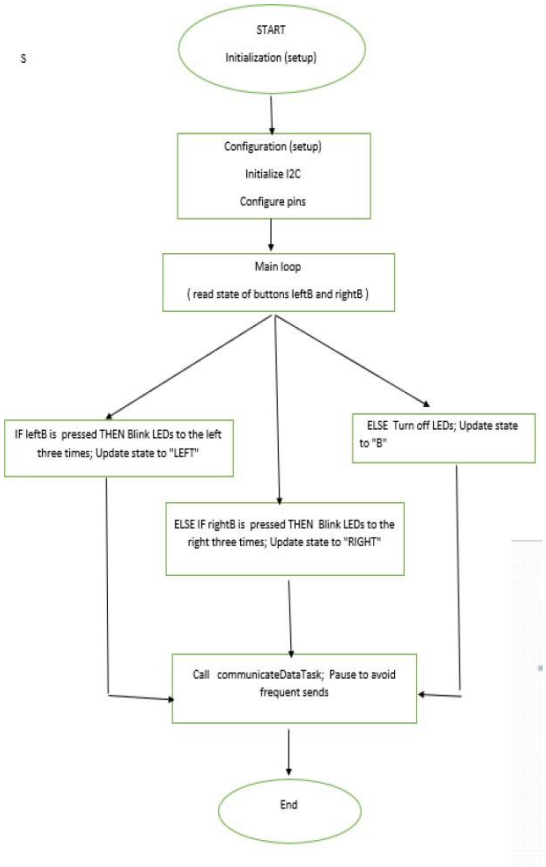
VI. L'ESCLAVE 3 ET CAMERA

- Le troisième esclave assume la responsabilité de la gestion de la direction . Cette direction est

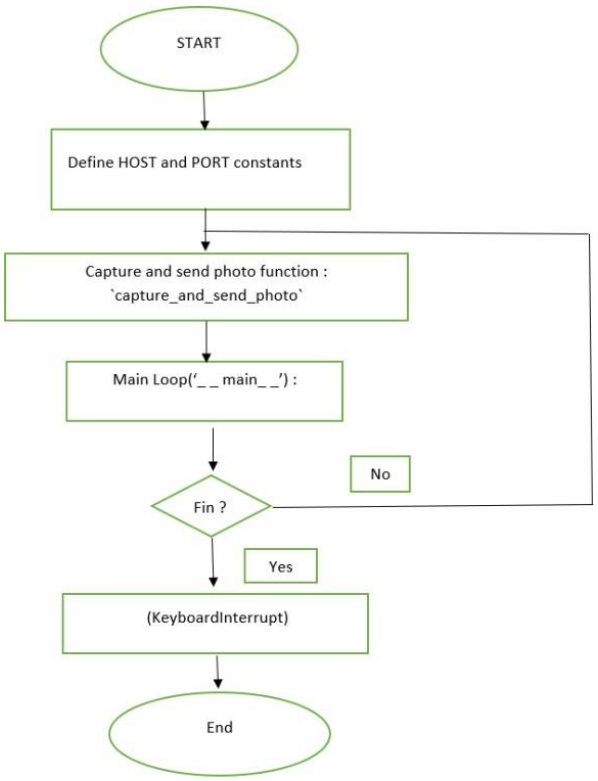
visuellement signalée par des indicateurs lumineux qui défilent de manière dynamique, s'ajustant en temps réel en fonction de la direction sélectionnée par l'utilisateur à l'aide d'un bouton dédié.il y a aussi une camera qui capture en continue des photos et eux aussi sont envoyés au maitre .

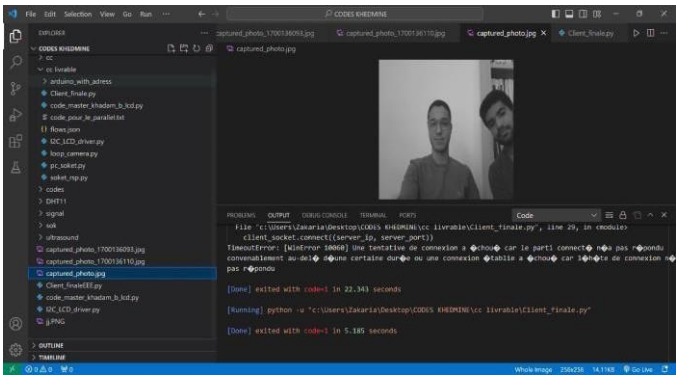


SCHEMA ALGORITHMIQUE : CODE SIGNALISATION



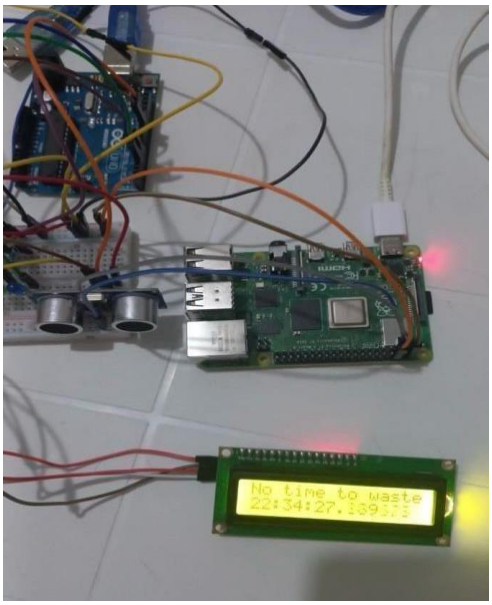
SCHEMA ALGORITHMIQUE : CODE CAM

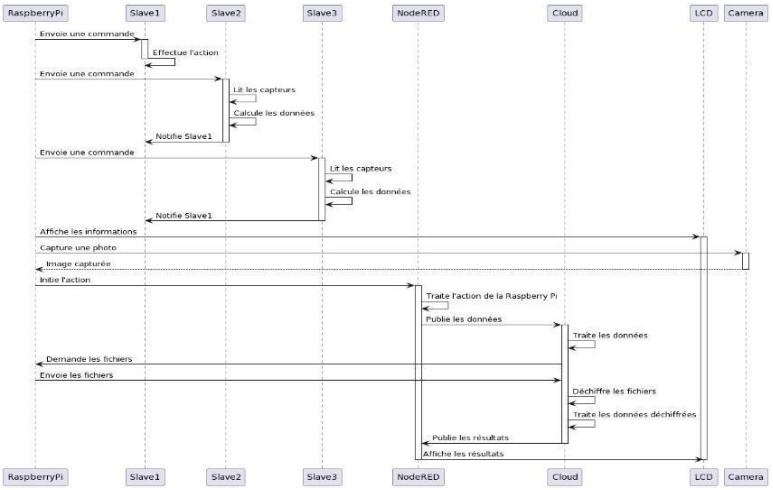




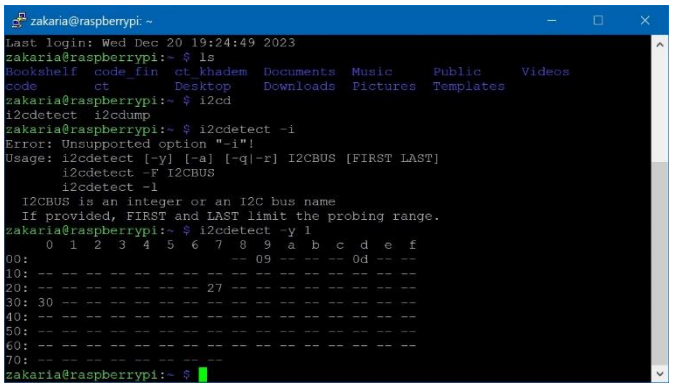
VIII. GESTION DE LA COMMUNICATION I2C

Le maître (Raspberry Pi) communique avec les esclaves (Arduino) via le protocole I2C, assurant une transmission de données bidirectionnelle. Des adresses I2C uniques sont attribuées à chaque esclave pour permettre une communication sélective





- Détection i2c



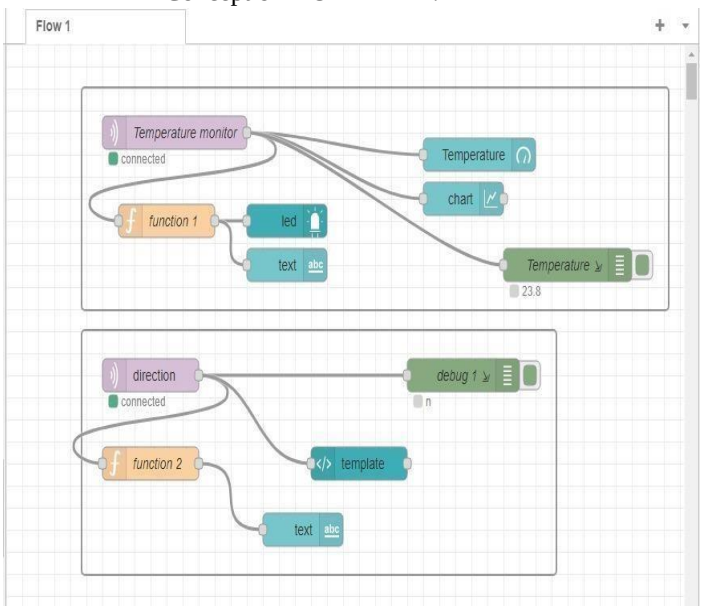
IX. Diagramme de séquence - handshake –

Le Raspberry Pi envoie une requête d'authentification au point d'accès Wi-Fi (AP) du cloud, qui répond par un message d'acceptation.
L'AP envoie au Raspberry Pi un nonce (un nombre aléatoire utilisé une seule fois) et des informations sur le groupe de chiffrement utilisé.
Le Raspberry Pi génère une clé de session (Pairwise Transient Key, ou PTK) à partir du nonce, de son adresse MAC, de l'adresse MAC de l'AP et de la clé pré-partagée (Pre-Shared Key, ou PSK) du réseau. Il envoie ensuite à l'AP un nonce et un message d'authentification (Message Integrity Code, ou MIC) calculés à partir de la PTK.
L'AP vérifie le MIC et génère la même PTK que le Raspberry Pi. Il envoie ensuite au Raspberry Pi un message d'authentification confirmant la réussite du handshake.

X. MISE EN OEUVRE LOGICIELLE (INTEGRATION DU PROTOCOLE MQTT , SOCKET ET DE NODE-RED)

La plateforme utilise MQTT ou dans notre cas Socket pour la communication Cloud-Maitre. Le nuage envoie des ordres au maître via le broker MQTT, lui demandant de lui fournir les données des esclaves. Le maître reçoit les ordres , et les exécute en demandant les données aux esclaves via I2C, il collecte les données des esclaves en simultané, les compresse en un seul fichier, et le crypte . Le maître utilise ensuite un socket pour envoyer le fichier crypté au cloud via le broker MQTT. Le nuage reçoit le fichier crypté , le décrypte , et le décompresse pour extraire les données. Puis le cloud communique les données à NODE-RED pour la création d'un tableau de bord interactif affichant les données de manière conviviale.

- Conception NODE-RED :



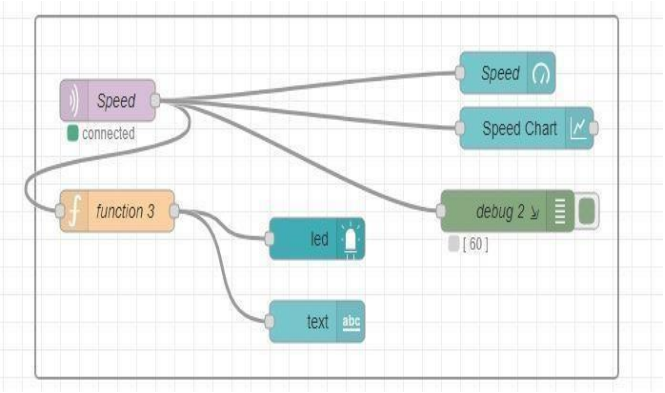
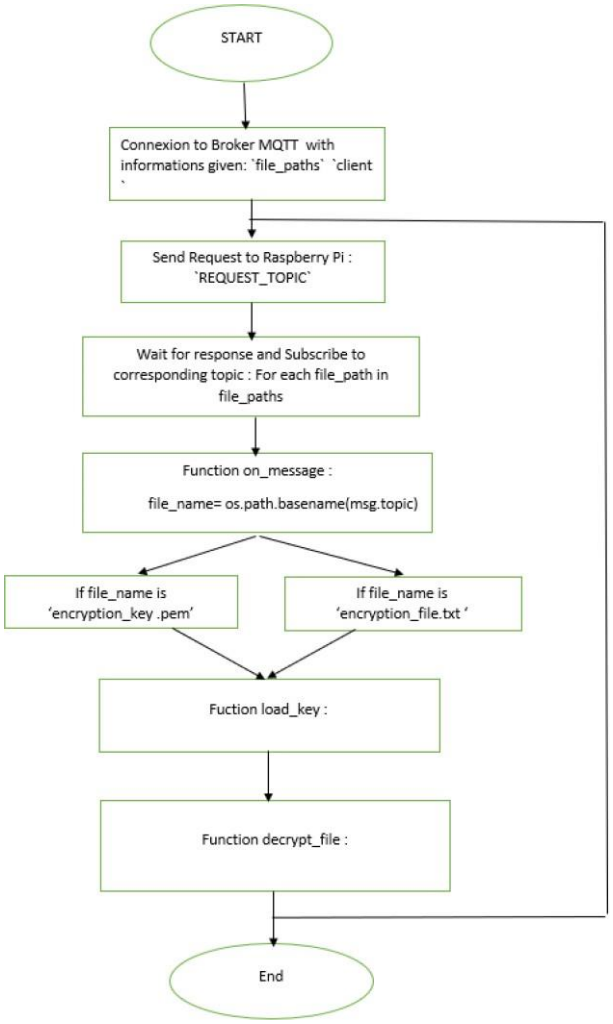


Schéma algorithmique : code cloud



XI. RESULTATS ET TESTS

Les tests effectués ont validé la fiabilité de la plateforme dans des scénarios simulés. L'étalonnage des capteurs, la vérification de la communication I2C, et la réactivité du système ont été confirmés avec succès.

XII. Le temps réel

La plateforme embarquée repose sur une architecture multitâche, offrant la capacité d'exécuter simultanément plusieurs tâches indépendantes. Cette approche permet une gestion optimale des ressources système, favorisant une utilisation efficiente des capacités du véhicule. Grâce à cette architecture, la plateforme peut réagir en temps réel aux événements critiques, assurant une réponse rapide et adaptative aux conditions changeantes. Chaque tâche fonctionne de manière isolée, évitant ainsi les conflits et assurant la stabilité du système global. L'architecture multitâche offre une flexibilité accrue pour traiter diverses opérations en parallèle, contribuant ainsi à une expérience de conduite fluide et sécurisée. Les avantages de cette approche se manifestent notamment dans la gestion efficace des capteurs, la commande des actionneurs et la coordination des différentes fonctionnalités du véhicule. En garantissant une exécution simultanée sans heurts des tâches, cette architecture optimise la réactivité du système, améliorant ainsi la sécurité et les performances globales du véhicule.

XIII. CONCLUSION

La plateforme véhiculaire embarquée multitâche, combinant I2C, MQTT, et Node-RED, offre une solution complète pour la surveillance et le contrôle du véhicule. L'utilisation de Node-RED facilite la visualisation des données, offrant une interface utilisateur intuitive pour une gestion optimale des informations. Cette intégration réussie ouvre la voie à des améliorations futures et à des applications étendues de la plateforme.