Prototype d'une application pour réfrigérateur intelligent SOFA – SMART OPERATIONAL FRIDGE APP MTI840

Amna SNENE
Baptiste VIERA
École de Technologie Supérieure

12 avril 2023

Table des matières

1 Introduction **3**

2 Définitions **3**

2.1 Informatique en périphérie : edge computing . . . 3

2.2 AWS IoT Greengrass . . . 3

2.3 Paradigme Pub/Sub . . . 3

3 Analyse des besoins **3**

3.1 Adaptation des besoins . . . 4

4 Conception **5**

4.1 Architecture matérielle . . . 5

4.2 Outils et services utilisés . . . 5

4.3 Architecture logicielle générale . . . 5

4.4 Architecture détaillée de la couche edge . . . 6

4.5 Architecture Cloud . . . 7

4.6 Interface graphique . . . 8

5 Conclusion **9**

6 Bibliographie **9**

7 Annexes **10**

7.1 Code source 10

7.2 Captures d'écran AWS 10

Table des figures

1 Diagramme d'activité sur l'identification des aliments. 4

2 Diagramme de la lecture de la date de préemption 4

3 Diagramme d'activité sur le suivi de la température. 4

4 Diagramme d'activité sur l'identification des aliments et la lecture de la date de péremption réalisés. 4

5 Architecture matérielle 5

6 Architecture logicielle générale. 6

7 Architecture Edge. 7

8 Architecture Cloud. 8

9 Interface graphique. 8

10 Capture d'AWS de la règle de routage des messages. 10

11 Capture d'AWS SNS pour envoyer une alerte par mail lorsque la température dépasse le seuil fixé. 10

1 Introduction

De nos jours, la surconsommation est devenue une caractéristique de notre société, ce qui peut rendre la tâche des courses alimentaires particulièrement stressante pour de nombreux consommateurs. Pour résoudre ce problème, nous avons développé S.O.F.A (Smart Operational Fridge App), une application destinée à alléger la charge mentale liée aux courses alimentaires. Grâce à cette application, les utilisateurs peuvent accéder facilement au contenu de leur réfrigérateur depuis n'importe quel endroit, obtenir des recettes adaptées à leurs besoins et limiter le gaspillage en étant informés de la date de péremption des aliments. En outre, S.O.F.A envoie des alertes en cas de température anormalement élevée dans le réfrigérateur, permettant ainsi aux utilisateurs de réagir rapidement et d'éviter tout gaspillage éventuel.

2 Définitions

2.1 Informatique en périphérie : edge computing

Avec l'évolution rapide des applications IoT et l'émergence de le 5G, le modèle actuel de calcul centralisé en nuage (cloud) ne répond plus aux besoins des applications de nouvelle génération. En effet, ces applications sont très exigeantes en termes de latence et traitement des données massives.
Dans ce contexte, la technologie informatique en périphérie (Edge Computing) voit le jour. Il s'agit d'étendre les fonctionnalités du cloud sur les périphéries de réseau près des utilisateurs finaux.

2.2 AWS IoT Greengrass

AWS IoT Greengrass est un runtime de périphérie et un service cloud open source pour l'Internet des objets (IoT) qui aide à créer, déployer et gérer des applications IoT sur des appareils peu puissants. AWS IoT Greengrass permet aux appareils IoT de collecter et d'analyser des données plus près de l'endroit où elles sont générées, de réagir de manière autonome aux événements locaux et de communiquer en toute sécurité avec d'autres appareils sur le réseau local.
Les appareils Greengrass peuvent également communiquer en toute sécurité avec AWS IoT Core et exporter des données IoT vers le nuage AWS.

2.3 Paradigme Pub/Sub

Publish/Subscrib est un paradigme, très populaire, utilisé pour la communication asynchrone orientée interlogiciel (MOM). Il est adapté aux spécifications des applications IoT puisqu'il permet le découplage des ses différentes entités.
Publish-Subscribe permet de publier des messages à partir des diffuseurs pour des abonnés qui s'intéressent à certaines catégories ou classes de messages.

3 Analyse des besoins

Dans cette section, nous identifions les scénarios d'utilisation que l'application offre. Le premier scénario est relatif à l'identification des aliments présents dans le frigo et la génération de recette à partir de ces derniers. Ceci est représenté par le diagramme d'activité illustré dans la figure 1.

FIGURE 1 – Diagramme d'activité sur l'identification des aliments.

Le deuxième scénario est relatif à la lecture de la date de péremption des aliments. La description détaillée est illustrée par le diagramme d'activité figure 2.

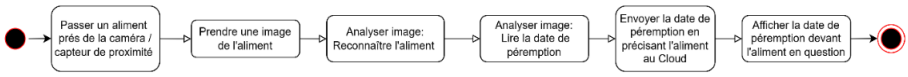

FIGURE 2 – Diagramme de la lecture de la date de préemption

Le dernier scénario, présenté dans la figure 3, est relatif à la température .

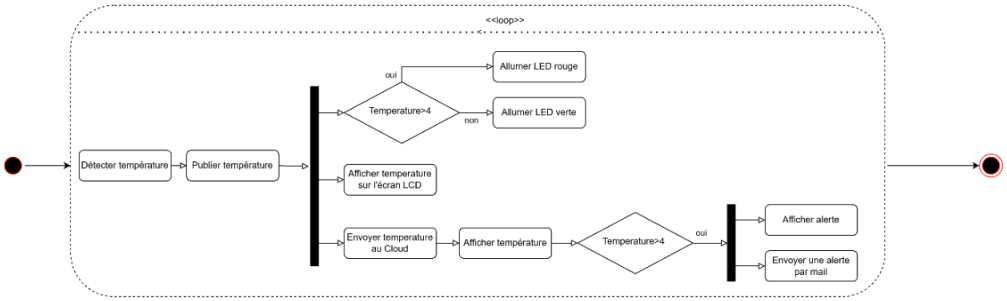

FIGURE 3 – Diagramme d'activité sur le suivi de la température.

3.1 Adaptation des besoins

En fonction des contraintes matérielles et financières, nous avons adapté les deux premiers scénarios d'utilisation pour créer un seul scénario réalisable illustré dans la figure 4.

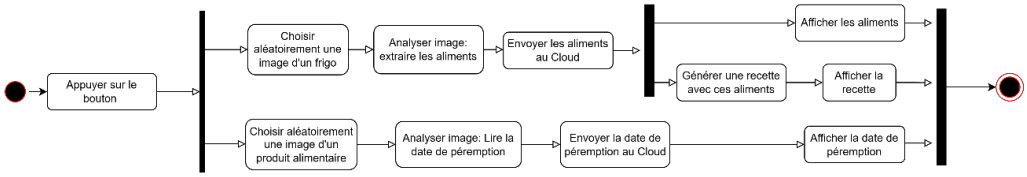

FIGURE 4 – Diagramme d'activité sur l'identification des aliments et la lecture de la date de péremption réalisés.

Le scénario lié à la température a été implémenté tel quel, sans modification par rapport au cas réel présenté ci-dessus, sur la figure3.

4 Conception

4.1 Architecture matérielle

En fonction des moyens matériels et financiers à notre disposition, S.O.F.A se compose des éléments suivants, comme montre la figure5 :

- Le Raspberry PI joue le rôle d'appareil Edge.
- La LED est verte lorsque la température est en-dessous du seuil et devient rouge lorsque la température est au-dessus du seuil.
- Le bouton permet de simuler l'ouverture de la porte du réfrigérateur et déclenche l'envoie de l'image du contenu du réfrigérateur pour l'analyse.
- L'écran permet d'afficher la température.
- Le capteur de température permet de mesurer les variations de température (à noter que pour faciliter nos tests nous avons fixé le seuil à 25°C et non 4°C.)

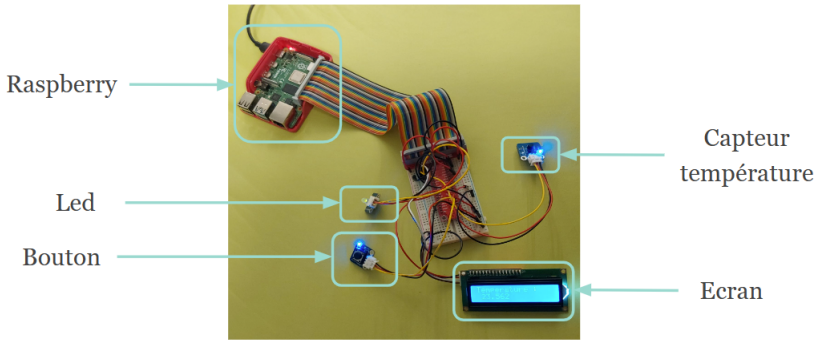

FIGURE 5 – Architecture matérielle

4.2 Outils et services utilisés

Pour mettre en place S.O.F.A, nous avons utilisé les services d'Amazon AWS suivants :

- AWS Greengrass : Permet de déployer des applications IoT sur des appareils périphériques et de les exécuter localement.
- Amazon IoT Core : Permet de connecter des appareils IoT à AWS, de collecter et de stocker des données, et de gérer les appareils à distance.
- Amazon SNS : Permet d'envoyer des notifications push, des SMS et des e-mails
- Amzon EC2 : Permet de lancer des instances de machines virtuelles dans le cloud d'AWS pour exécuter des applications et des charges de travail.

Nous avons également utilisé les framework suivants :

- PyTorchet Open-CV : Permet de créer et d'utiliser des modèles de vision par ordinateur
- Roboflow : Permet de créer, d'annoter et d'entraîner et d'utiliser des modèles de vision par ordinateur.
- Bulma : Permet de fournir des composants d'interface utilisateur (CSS) prêts à l'emploi pour la création de sites web.
- Nginx : Permet d'offrir un serveur web léger, rapide et hautement extensible qui peut être utilisé pour servir du contenu web statique ou dynamique.
- Flask : Permet de créer des API Rest en python.

4.3 Architecture logicielle générale

Pour le développement de notre application S.O.F.A, nous avons opté pour une architecture deux tiers Cloud – Edge représentée dans la figure 6.

La couche Edge est composée par un serveur Edge exécutant AWS IoT Greeangrass et un ensemble de capteurs. La couche Cloud est composée par un ensemble de service AWS (AWS IoT Core, AWS SNS et EC2) et par un serveur Roboflow.
La couche Edge communique avec le service AWS IoT Core à travers le protocole MQTT et avec EC2 et Roboflow en HTTP.

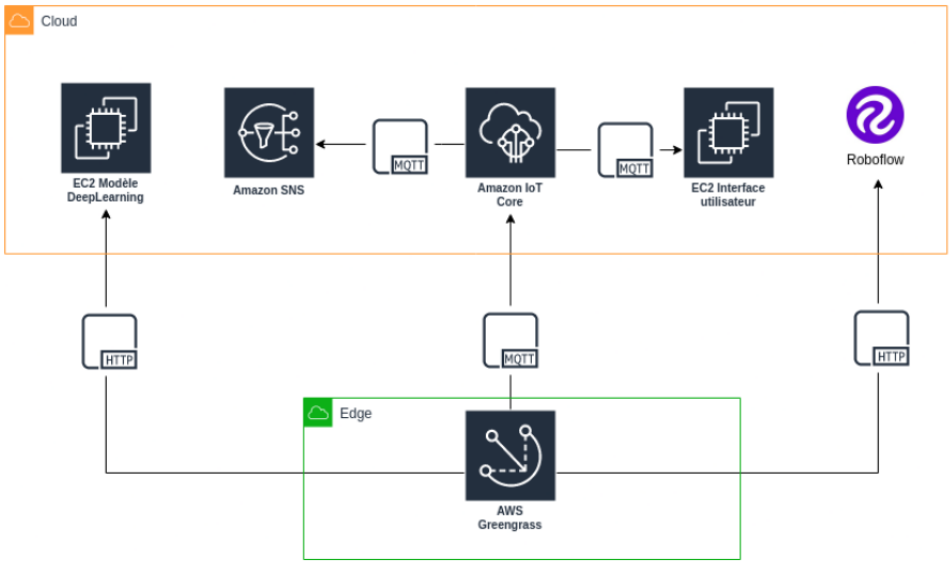

FIGURE 6 – Architecture logicielle générale.

4.4 Architecture détaillée de la couche edge

Un composant Greengrass est un module logiciel qui est déployé et fonctionne sur un appareil Greengrass Core. Tous les logiciels développés et déployés avec AWS IoT Greengrass sont modélisés en tant que composants. AWS IoT Greengrass fournit des composants publics préconstruits et offre la possibilité de développer des composants personnalisés.
Dans le cadre de notre projet, nous avons développé 7 modules logiciels qui communiquent entre eux à travers le module inter-process communication (IPC) suivant le modèle publish/subscribe, comme montre la figure 7.

- Température : Ce module lit la valeur de température détectée par le capteur et la publie sur le sujet (topic) 'Température' chaque 4 secondes.
- LED : Ce module exprime son intérêt (subscribe) pour le sujet 'Température'. Quand il reçoit un message de la part de l'PC, il vérifie si la température est au-dessus de la température maximale et il allume la LED en vert sinon en rouge.
- Écran : Ce module exprime son intérêt pour le sujet 'Température'. Quand il reçoit un message de la part de l'PC, il affiche la valeur de la température sur l'écran.
- Bouton : Quand l'utilisateur appuie sur le bouton, ce module publie un message sur le sujet 'Notification'.
- Date de péremption : Ce module exprime son intérêt pour le sujet 'Notification'. Quand il reçoit un message de la part de l'PC, il choisit une image locale aléatoirement et il l'envoie au EC2 contenant l'algorithme deep learning de lecture des dates. Comme réponse, il reçoit la date et il la publie sur le sujet 'Date de péremption '.
- Aliments : Ce module exprime son intérêt pour le sujet 'Notification'. Quand il reçoit un message de la part de l'PC, il choisit une image locale aléatoirement et il l'envoie au Roboflow contenant l'algorithme deep learning de reconnaissance des aliments. Comme réponse, il reçoit la liste des aliments et il la publie sur le sujet 'Aliments'.
- Sync_data_cloud : Ce module exprime son intérêt pour les sujets 'Température', 'Date de péremption ' et 'Aliments'. Il synchronise les données locales avec le cloud en publiant les

messages reçus sur les MQTT Topics de AWS IoT Core.

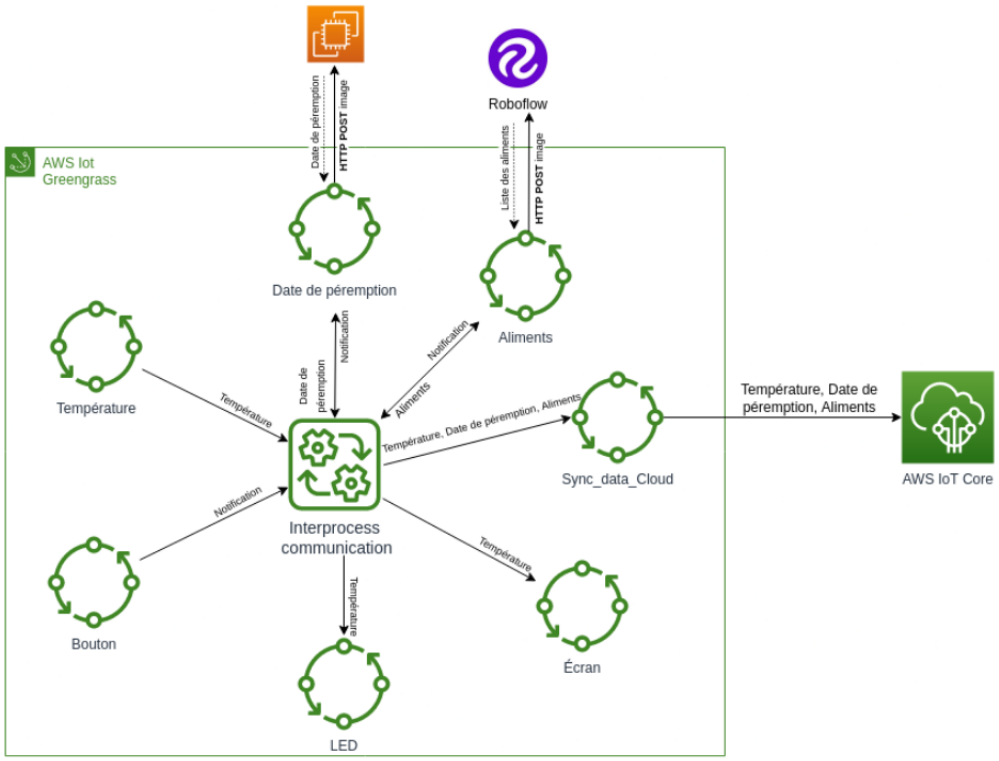

FIGURE 7 – Architecture Edge.

4.5 Architecture Cloud

L'objectif de S.O.F.A est d'afficher dynamiquement les différentes informations (Température, Aliments, Date de péremption) sur un site web accessible par l'utilisateur final. Pour se faire, nous avons mise en place une architecture cloud qui se compose des éléments suivants :

- EC2 Lecture Date de Péremption : Afin de décharger partiellement l'EC2 Interface Utilisateur, et du fait de ressources financières limitées (nos EC2 sont peu performantes), nous avons mis en place notre modèle de deep learning basé sur la bibliothèque EasyOCR dans un autre EC2. Comme expliqué précédemment, la date sera publiée sur le sujet « Date de péremption»
- EC2 Interface Utilisateur : Cette machine virtuelle contient notre site web développé à l'aide de Flask et Nginx. Elle contient également le modèle de deep learning de génération de la recette en fonction des aliments dans le réfrigérateur. Les aliments, tout comme la température et la date de péremption sont récupérés par l'EC2 qui joue le rôle d'IoT Thing. Cela lui permet de se souscrire au MQTT Topics présents dans l'AWS IoT Core qui contiennent les données envoyées par l'intermédiaire d'AWS IoT Greengrass.
- AWS SNS : Dans les MQTT Topics, un d'entre eux contient la valeur de la température. Nous avons alors appliqué une «AWS IoT Rule» qui déclenche le service AWS SNS si la température est supérieure au seuil fixé. Ce déclenchement entraine l'envoie d'une alerte sous la forme d'un email à l'utilisateur pour l'informer d'une température anormalement élevée.

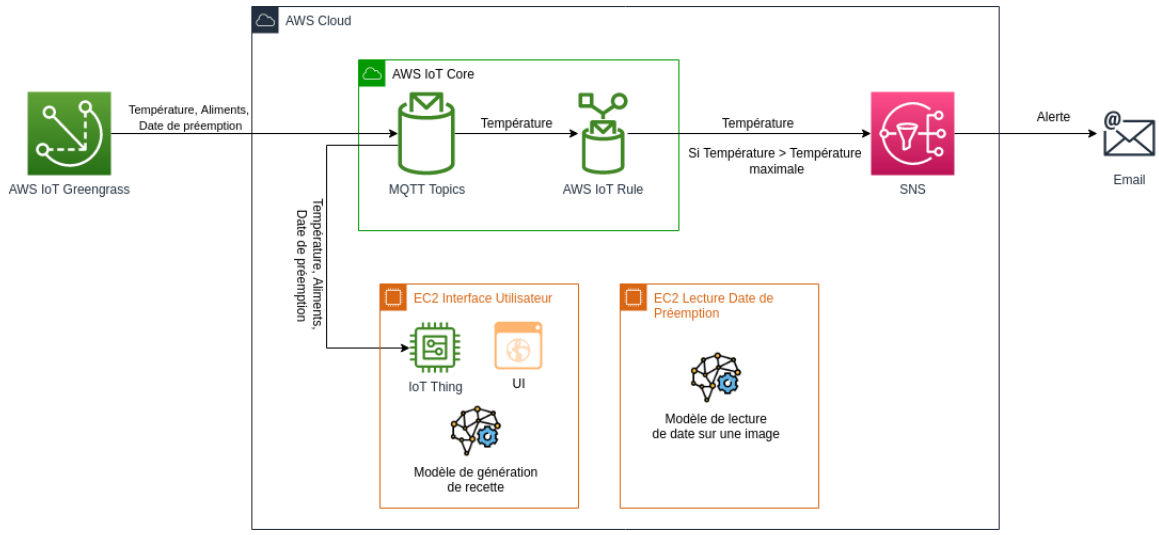

FIGURE 8 – Architecture Cloud.

4.6 Interface graphique

Pour la conception de l'interface utilisateur de notre site web, nous avons utilisé un template HTML/CSS que nous avons personnalisé en utilisant le framework Bulma pour répondre à nos besoins spécifiques, tel que l'affichage d'un message d'alerte en cas de dépassement du seuil de température.

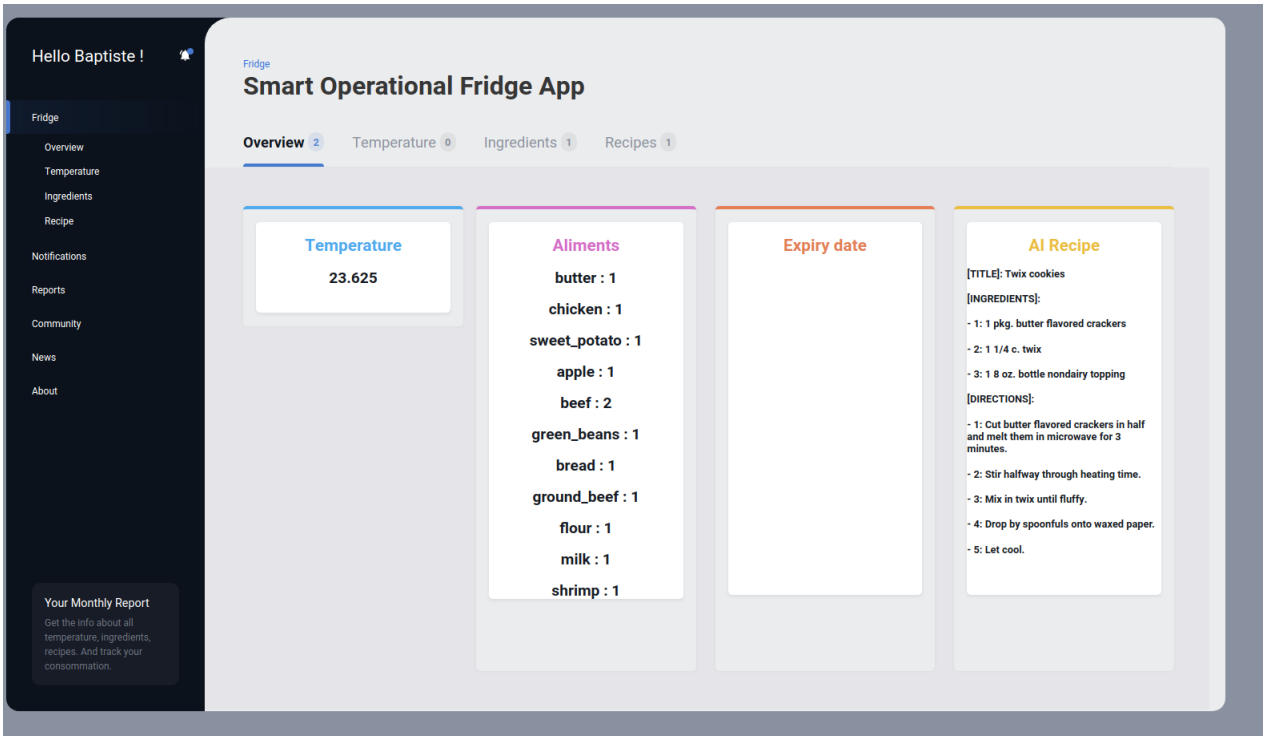

FIGURE 9 – Interface graphique.

5 Conclusion

En résumant, nous avons mis en place une application complète qui assure la récupération de données à partir de nos différents capteurs, leur analyse via une infrastructure edge-cloud et leur présentation sous la forme d'un site web destiné aux utilisateurs finaux. Pour cela, nous avons utilisé différentes technologies telles que les services d'AWS, les bibliothèques Python de deep learning pour la vision par ordinateur et des frameworks pour le développement web.

En ce qui concerne les limites de notre implémentation, nous pouvons mentionner les éléments suivants : nous ne disposons pas de caméra, tous les capteurs sont regroupés sur le même appareil (carte Raspberry) représentant le serveur Edge, nous ne faisons pas appel à une base de données, et enfin nous n'utilisons pas de connexion websocket pour l'interface utilisateur.

En conclusion, nous avons identifié plusieurs perspectives pour le développement futur de notre application. Nous pourrions envisager de créer une version mobile de S.O.F.A, d'ajouter un certificat de sécurité SSL pour renforcer la sécurité de l'application web, d'élargir notre gamme de modèles de machine et de deep learning, et finalement de commercialiser l'application S.O.F.A.

6 Bibliographie

[1] *Amazon Web Services (AWS) IoT Core*. Consulté le 8 avril 2023 url :https://aws.amazon.com/fr/iot-core/

[2] *Amazon Web Services (AWS) Greengrass*. Consulté le 8 avril 2023 . url : https://aws.amazon.com/fr/greengrass/

[3] *AWSomeIoT. "AWS IoT GreengrassV2*. Consulté le 25 mars 2021. url :https://www.youtube.com/watch?v=rLswsgz77I0

[4] *Roboflow* Consulté le 8 avril 202 url :https://roboflow.com/

[5] *EasyOCR*.Consulté le 8 avril 2023. url : https://github.com/JaidedAI/EasyOCR

[6] *Bulma*.Consulté le 8 avril 2023 url :https://bulma.io/

[7] *Nginx*. Consulté le 8 avril 2023 url : https://www.nginx.com/

[8] *Flask* Consulté le 8 avril 2023. url : https://flask.palletsprojects.com/en/2.2.x/

[9] *OpenCV*. Consulté le 8 avril 2023. url : https://opencv.org/.

[10] *Chef-Transformer*.Consulté le 9 avril 2023. url : https://github.com/chef-transformer/chef-transformer.

[11] *EasyOCR*. Consulté le 9 avril 2023. url :https://github.com/JaidedAI/EasyOCR

7 Annexes

7.1 Code source

AWS Greengrass Components. url :https://github.com/AmnaSnene/MTI840-AWS-IoT-Grenngrass-Components

Interface Web avec l'algorithme deep learning de génération de recette. url :https://github.com/AmnaSnene/MTI840-frontend

7.2 Captures d'écran AWS

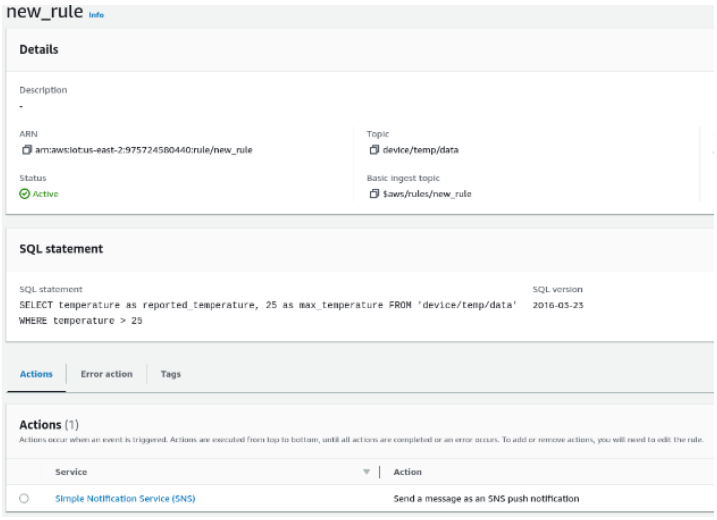

FIGURE 10 – Capture d'AWS de la règle de routage des messages.

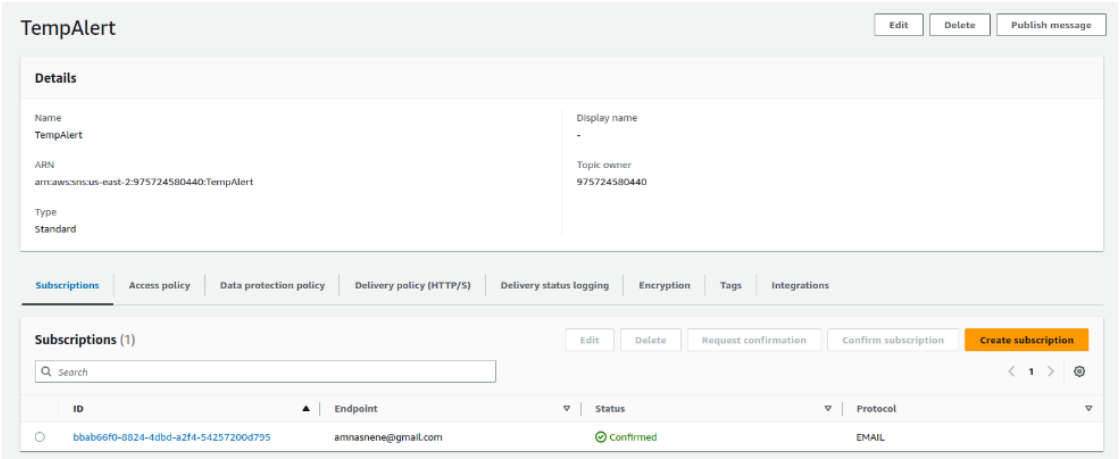

FIGURE 11 – Capture d'AWS SNS pour envoyer une alerte par mail lorsque la température dépasse le seuil fixé.