REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Etude et conception d'une passerelle transparente pour le Cloud Microsoft Azure

---

MENTOURI Chiraz et TISMELANE Rahma

Sous la direction de :

M. Rabah SADOUN Prof. ENP

Mm Widad KARTOUS. ENP

et **Mlle Anfel BEGHOURA. BRANDT**

Présenté et soutenue publiquement le (30/06/2019)

Composition du jury :

Président	**M. CherifLARBES**	**Prof.**	**ENP**
Promoteur	**M. Rabah SADOUN**	**Prof.**	**ENP**
Promotrice	**Mm. Widad KARTOUS**	**Dr.**	**ENP**
Promotrice	**Mlle Anfel BEGHOURA**	**Ingénieur**	**BRANDT**
Examinateur	**M. Mohamed TAGHI**	**Prof.**	**ENP**

ENP 2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en électronique

---

Etude et conception d'une passerelle transparente pour le Cloud Microsoft Azure

---

MENTOURI Chiraz et TISMELANE Rahma

Sous la direction de :

M. Rabah SADOUN Prof. ENP

Mm Widad KARTOUS. ENP

et **Mlle Anfel BEGHOURA. BRANDT**

Présenté et soutenue publiquement le (30/06/2019)

Composition du jury :

Président	**M. Cherif LARBES**	**Prof.**	**ENP**
Promoteur	**M. Rabah SADOUN**	**Prof.**	**ENP**
Promotrice	**Mm Widad KARTOUS**	**Dr.**	**ENP**
Promotrice	**Mlle Anfel BEGHOURA**	**Ingénieur**	**BRANDT**
Examinateur	**M. Mohamed TAGHI**	**Prof.**	**ENP**

ENP 2019

Dédicace

On dédie ce travail à nos chers parents et grands parents,

A nos familles,

A nos frères et soeurs,

A nos amis
Et à tous ceux qui ont fait confiance à Nos Capacités
pendant notre parcours.

Remerciements

Nous souhaitons avant tout exprimer notre reconnaissance et gratitude à notre directeur de thèse ***Pr.Rabah SADOUN*** pour le temps qu'il a consacré pour nous apporter tous les outils et les éléments nécessaires pour la réalisation de notre projet, pour sa patience et surtout ces conseils précieux qui ont toujours contribué à alimenter notre réflexion. Nous remercions aussi Anfel BEGOURA pour nous avoir proposé ce sujet de PFE et nous avoir donné cette opportunitée de travailler sur un projet industriel d'intérêt national.

Nous présentons également toutes nos remerciement à l'entreprise **BRANDT/-Cevital** de nous avoir donné la chance de travailler sur un tel sujet d'actualité.

Nous adressons nos sincères remerciements à tous nos professeurs de l'**Ecole Nationale Polytechnique** de nous avoir accompagné tout au long de notre parcours. Aussi,à tous les intervenants et toutes personne qui par leurs critique ou conseils ont guidé notre chemin vers la réussite .

Nous adressons tous nos respects à l'ensemble des membre de jury ***Pr.Cherif LARBES*** et ***Pr.Mohamed TAGHI***.

Nous remercions en particulier nos Chères parents ainsi qu'à nos Frères et soeurs pour leurs soutiens émotionnel ,aussi pour l'intérêt qu'ils ont toujours porté pour nous .

Nous adressons toutes nos gratitudes à tous nos ami(e)s et à toutes les personnes qui nous ont aidé dans la réalisation de ce travail.

Enfin, les mots les plus simples étant les plus forts, nous adressons toutes nos affections à nos mamans et nos papas. Merci pour avoir fait de nous ce que nous somme aujourd'hui.

ملخص
تمثل "إنترنت الأشياء" و "السحابة" و "التشغيل الآلي للمنزل" اللبنات الأساسية للمنزل الذكي ، ومع تضاعف الأجسام المتصلة ،بدأ
نموذج الحوسبة السحابية في تقديم بعض العيوب ، مما يفرض جيلًا جديدًا التكنولوجيا السحابية وتحليل حيث "الحوسبة " الحافة".
يعالج هذا المشروع تصميم بوابة شفافة جسر شفاف بتقنية الحوسبة على الحاف استنادًا إلى خدمات إنترنت الأشياء التي تقدمها منصة مايكروسوفت أزور.
يتمثل الدور الرئيسي لهذه البوابة في توصيل العديد من الأجهزة وتجميع البيانات وتحليلها وإرسالها بعد ذلك. نظرًا لأن البوابة مثبتة بالقرب من الأجهزة وأجهزة الاستشعار ، فإنها لا ترسل سوى البيانات ذات أهمية إلى السحابة ، مما يقلل من التكافة و الوقت.
الكلمات الرئيسية: إنترنت الأشياء ، السحابة ، أتمتة المنزل ، المنزل الذكي ، الحوسبة الحافة ، منصة مايكروسوفت أزور كلاود.

Abstract
The Internet of Things, the Cloud and the home automation present the building blocks of the smart home. However, as connected objects are multiplying, the cloud computing model is beginning to present some issues. This imposes a new generation of cloud technology and analysis ,where the "Edge Computing".
This project consist of the conception of an Edge transparent gateway based on the IoT services offered by the Microsoft Azure Cloud platform.
The main role of this gateway is to connect many devices, aggregates and analyzes the data, and sends them. Because the gateway is installed near devices and sensors, it sends only relevant data to the cloud, reducing the cost of latency.
Keywords: Internet of Things, Cloud, Home Automation, Smart Home, Edge Computing, Gateway, Microsoft Azure Cloud Platform.

Résumé
l’internet des objets, le Cloud et la domotique présentent les éléments constitutifs de la maison intelligente .Cependant, à l’heure où les objets connectés se multiplient ,le modèle du cloud computing commence à présenter quelque failles .Ce qui impose un nouveau paradigme pour le Cloud à savoir le « Edge Computing ».
Ce projet aborde la conception d’une passerelle transparente de type Edge basée sur les services IoTs offerts par la plateforme Cloud Microsoft Azure .
Le rôle principale de cette Gateway est de connecter de nombreux appareils, agrège et analyse les données, et les envoie par la suite. Comme la passerelle est installée à proximité des appareils et capteurs, elle envoie uniquement des données pertinentes vers le Cloud, réduisant ainsi le coût de la latence .
Mots clés : Internet des objets ,Cloud ,Domotique,Maison intelligente,Edge Computing,Gateway , la plateforme Cloud Microsoft Azure.

Table des matières

Liste des tableaux

Liste des figures

Liste des abréviations

Introduction générale **14**

I État de l'art et concepts 16

1 Home Automation, Internet of Things et Cloud **17**
1.1 Introduction . . . 18
1.2 Définitions . . . 18
1.2.1 domotique . . . 18
1.2.2 Maison connectée . . . 18
1.2.3 Maison intelligente . . . 18
1.3 Origines de la maison intelligente . . . 19
1.4 Maison connectée dans le Monde . . . 20
1.4.1 Aux Etats Unis . . . 20
1.4.2 En Asie . . . 20
1.4.3 En Europe . . . 20
1.5 Travaux dans le contexte du Home Automation : état de l'art . . . 21
1.6 Architecture et services d'une maison connectée . . . 23
1.6.1 Architecture . . . 24
1.6.2 Les services de la maison connectée . . . 25
1.6.2.1 économie d'énergie . . . 25
1.6.2.2 Confort . . . 26
1.6.2.3 Sécurité . . . 26
1.7 Architecture d'une maison intelligente . . . 27
1.7.1 Contribution du Cloud à la maison connectée . . . 27
1.7.2 Internet of Things (IoT) . . . 29
1.7.3 Architecture IoT . . . 29
1.7.4 Technologies et protocoles de communication IoT . . . 30
1.7.5 Services de la maison intelligente . . . 32
1.7.5.1 Sécurité . . . 32

1.7.5.2 Confort . . . 33
1.7.5.3 Economies d'énergie . . . 34
1.8 Maison intelligente : de Cloud Computing au Edge Computing . . . 34
1.9 Conclusion . . . 35

2 Microsoft Azure **37**
2.1 Introduction . . . 38
2.2 Cloud Microsoft Azure . . . 38
2.2.1 Présentation . . . 38
2.2.2 Architecture générale . . . 39
2.2.3 Services Microsoft Azure . . . 40
2.2.3.1 Services de calcul . . . 40
2.2.3.2 Services de données . . . 40
2.2.3.3 Services réseau . . . 41
2.2.3.4 Service IoT . . . 41
2.3 Azure IoT . . . 41
2.3.1 Architecture . . . 42
2.3.1.1 Appareils IoT . . . 43
2.3.1.2 Passerelle Cloud . . . 43
2.3.1.3 Provisionnement des appareils . . . 43
2.3.1.4 Traitement de flux . . . 44
2.3.1.5 Stockage du chemin à chaud . . . 44
2.3.1.6 Stockage de chemin à froid . . . 44
2.3.1.7 Transformation des données . . . 44
2.3.1.8 Intégration des processus métier . . . 45
2.3.1.9 Gestion des utilisateurs . . . 45
2.3.2 Services Azure IoT . . . 45
2.3.2.1 IoT Central . . . 45
2.3.2.2 Accélérateurs de solutions IoT . . . 46
2.3.2.3 Azure IoT Hub . . . 46
2.3.2.4 IoT Edge . . . 46
2.3.2.5 Azure Maps . . . 46
2.4 Azure IoT Hub . . . 46
2.4.1 Architecture . . . 47
2.4.2 La communication . . . 47
2.4.2.1 Priorité du message sur la communication entre le périphérique et le backend . . . 48
2.4.2.2 Priorité du message sur la communication Backend to Device . . . 48
2.4.3 Service "Provisionning" . . . 48
2.5 Azure IoT Device . . . 50
2.5.1 Device . . . 50
2.5.2 Device Twin . . . 51
2.6 Azure IoT edge . . . 52
2.6.1 Architecture Azure IoT Edge . . . 53
2.6.1.1 Conteneur Docker . . . 53
2.6.1.2 Organisation en Module . . . 54

2.6.1.3 Module Twin . . . 54
2.6.1.4 Azure IoT Edge Runtime . . . 55
2.6.2 Routage des message . . . 57
2.6.2.1 Source . . . 57
2.6.2.2 Condition . . . 57
2.6.2.3 Destination . . . 57
2.6.3 Les certificats . . . 57
2.6.3.1 Autorité de certification AC . . . 58
2.6.3.2 Certificat racine CA . . . 59
2.6.3.3 Certificat intermédiaire . . . 59
2.6.3.4 Certificat Device CA . . . 59
2.6.3.5 Certificat de serveur IoT Hub . . . 59
2.7 Service de développement- Azure SDK . . . 59
2.7.1 Device SDK . . . 60
2.7.2 Service SDK . . . 60
2.7.3 Azure IoT Edge SDK . . . 60
2.8 Sécurité . . . 60
2.8.1 Protection des serveurs Windows et Linux . . . 61
2.8.2 Protection des applications Cloud . . . 61
2.8.3 Protection des données . . . 61
2.8.4 Protection d'une solution IoT . . . 61
2.9 Conclusion . . . 62

II Implémentation 63

3 Implémentation 64
3.1 Introduction . . . 65
3.2 Architecture Générale de la solution . . . 65
3.3 Protocoles de communication . . . 67
3.4 Choix du langage de programmation . . . 68
3.5 Le développement multicible . . . 68
3.6 Les opérations préalables à réaliser . . . 68
3.6.1 Provisionnement . . . 68
3.6.1.1 Enregistrement de l'appareil IoT . . . 69
3.6.1.2 Configuration . . . 70
3.6.1.3 Vérification de l'enregistrement de l'appareil . . . 70
3.6.1.4 Monitoring de démarrage du device . . . 71
3.6.2 sécurité . . . 72
3.7 Développement de IoT Edge Gateway . . . 74
3.7.1 Infrastructure de développement . . . 74
3.7.2 Flot de conception . . . 75
3.7.2.1 Module SimulatedTelemetry . . . 77
3.7.2.2 Module ReceiveModule . . . 77
3.7.2.3 Module TraitementModule . . . 78
3.7.2.4 Module TransmitModule . . . 78
3.7.3 Organigramme de fonctionnement général d'un module . . . 78

3.7.4 Explication du code source de chaque type de module 80
3.7.4.1 Module Simulated Telemetry 80
3.7.4.2 Module ReceiveModule 84
3.7.4.3 Module TraitementModule 85
3.7.4.4 Module TransmitModule 87
3.7.4.5 Routage des Messages 88
3.7.5 Construction des images et leurs publications sur le Cloud . . . 88
3.8 Développement de l'application Device 89
3.8.1 Flot de conception 89
3.8.2 Architecture de l'application 92
3.8.3 Explication du code source de chaque type de module 94
3.9 Validation 98

Conclusion générale et perspectives **100**

Bibliographie **102**

Liste des tableaux

1.1 Les différents couches d'une infrastructure IoT 30

Table des figures

1.1 L'architecture de la maison connectée 24
1.2 Le paradigme global du Cloud 27
1.3 L'architecture du Forum mondial de l'IoT 30
1.4 Du Cloud computing au Edge Computing 35

2.1 La plate-forme d'Azure dans le monde 39
2.2 Architecture de la plate-forme Cloud Azure 40
2.3 L'architecturede référence pour une application IoT d'Azure 42
2.4 Traitement du flux de données 42
2.5 Les service d'Azure IoT 45
2.6 L'architecture de IoT Hub 47
2.7 Porvisionnement du Device 49
2.8 L'architecture du device Twin 51
2.9 Présentation du format JSON 52
2.10 Azure IoT Edge 53
2.11 L'architecture d'Azure IoT Edge 53
2.12 chaîne de certificats 58
2.13 Azure Security Center 60

3.1 Architecture générale de la solution. 66
3.2 Provisionnement d'un appareil 69
3.3 Enregistrement d'un appareil 70
3.4 La vérification du l'enregistrement de l'appareil. 71
3.5 Démarrage de l'appareil 72
3.6 Infrastructure de développement 74
3.7 Flot de conception et de déploiement d'un module (Simplifié) 75
3.8 Modèle de l'IoT Edge Device développé 77
3.9 Organigramme de fonctionnement d'un module en mode transmission de message 79
3.10 Organigramme de fonctionnement d'un module en mode réception . . . 80
3.11 Le routage des messages 88
3.12 Flot de conception 91
3.13 Fonctionnement de la communication D2D 93
3.14 Résultat pour Receive Module 98
3.15 Résultat pour le module de traitement 98

3.16 Confirmation que le message envoye du Client est bien recue par la Gateway 99

Liste des abréviations

IoT : Internet of Things

SDK : Software Developement Kit

CNN : Cable New Network

SOA : Service Oriented Architecture

AAL : Ambiante Assistive Living

API : Application Programming Interface

KPI : Key Performance Indicators

MQTT : Message Queuing Telemetry Transport

AMQP : Advanced Message Queuing Protocol

HTTP : Hypertext Transfer Protocol

AI : Intelligence Artificielle

TLS : Transport Layer Security

AC : Authetification Certeficat

DPS : Device Provosionning Service

SVM : Service Maching Learning

SQL : Structured Query Language

CVC : Chauffage Ventilation et Climatisation

JSON : JavaScript Object Notation

PaaS : Platform as a Service

SaaS : Software as a Service

IaaS : Infrastructure as a Service

VLS : Visible Light Communication

RFID : Radio Frequency Identification

UUID : Universely Unique Identifier

DB : UData Base

TPM : Total Productive Maintenance

BLE : Bleutooth Low Energy

SMS : Short Message Service

IWF : IoT World Forum

HSM : Hardware Security Module

Introduction générale

Il n'y a pas longtemps, l'informatique a été appliquée à la création d'habitats du futur (ou maisons intelligentes) afin d'améliorer les conditions de vie domestiques et d'offrir un système de contrôle distant fiable. Une telle maison est définie comme une résidence équipée de technologies de l'électronique, de l'automatique, de l'informatique et des télécommunications permettant d'améliorer le confort, la sécurité, la communication et la gestion d'énergie. La maison intelligente repose sur trois principes, en l'occurrence, la liaison entre les appareils, la communication entre l'utilisateur et les appareils et enfin, l'autonomie. Par ailleurs, il n'est pas judicieux d'évoquer « la maison intelligente » sans évoquer les dernières technologies telles que l'IoT et le Cloud Computing.

En effet, es milliards d'objets connectés génèrent des données en continu nécessitant un stockage et une évaluation en temps réel pour des applications critiques. Cela représente une tâche que les solutions Cloud compting ne pourront à la longue que difficlement accomplir. Ce qui impose un nouveau paradigme pour le Cloud à savoir l' Edge Computing.

Le « Edge Computing » consiste à déporter les fonctionnalitées du Cloud vers la périphérie pour proposer des services près de la source des données (service(s) proche(s) du capteur). Ce qui permet d'éviter la contrainte liée à la nécessité de déploiement de réseaux de forte capacité ; d'où des contraintes technologiques liées à l'infrastructure et des contrainte économique liée à l'usage (côté usager).

Dans ce contexte, nous sommes étions emmenées à réaliser un Projet de Fin d'Études au sein de la société BRANDT pour étudier et développer une solution de passerelle pour des noeuds Iot dans le but de permettre l'agrégation et le prétraitement des données générées avant leurs transmission vers le Cloud.

Pour développer dans le présent mémoire le travail que nous avions accompli, nous l'avons structuré en trois parties.

Dans le premier chapitre, nous avons contextualisé la problématique de notre développement. Aussi, nous aborderons dans un premier temps, les concepts de la domotique, de la maison connectée et de la maison intelligente pour converger vers le concept du edge computing ; concepte qui une solution pour les clouds d'aujourd'hui.

Le deuxième chapitre, quant à lui, se focalisera sur la présentation de la plateforme Cloud de Microsoft Azure en général, et de cloud Azure IoT en particulier. Nous citerons les différents services IoT disponibles. Nous détaillerons esentiellement le service Azure IoT Edge qui représente le cœur de notre solution. Les concepts qui y seront abordés vont être utiles pour l'explication (chapitre trois) de la solution que nous avions développée.

Le troisième chapitre sera dédié à la description du développement de notre solution et sa validation . Une conclusion évaluera le travail que nous avions accompli et en donnera des perpectives.

Première partie

État de l'art et concepts

Chapitre 1

Home Automation, Internet of Things et Cloud

1.1 Introduction

Les premiers développements de la domotique sont apparus dans les années 1980 grâce à la miniaturisation des systèmes électroniques et informatiques. Dès lors, l'industrie a concentré ses expérimentations sur le développement d'automates, d'interfaces et d'outils apportant confort, sécurité et assistance au sein d'un édifice.
La combinaison des technologies de l'Internet des Objets et du Cloud Computing a donné une autre dimension à la maison de demain ce qui est à l'origine de la maison connectée et la maison intelligente.

Ce chapitre est consacré aux concepts de la domotique, maison connectée, maison intelligente ainsi que la contribution de l'IoT et du Cloud Computing au développement des systèmes domestiques.

1.2 Définitions

1.2.1 domotique

Venant du mot latin « domus » signifie maison et le suffixe « tique » pour informatique [1].Le terme « domotique » regroupe, à la fois, l'ensemble des technologies de l'Electronique, de l'Informatique et des Télécommunications qui sont utilisées dans les domiciles pour rendre ceux-ci plus « intelligents » [2].

1.2.2 Maison connectée

C'est une maison équipée avec des systèmes domotiques en donnant la possibilité de contrôler et piloter, sur place ou à distance, tout l'environnement de la maison [1] [2].

1.2.3 Maison intelligente

C'est l'extension de la maison connectée. Autonome, elle possède la capacité d'anticiper le comportement des équipements automatisés [1] [2].

1.3 Origines de la maison intelligente

Bien que l'idée de la domotique existe depuis un certain temps, les maisons connectées n'existent que depuis la minimisation et l'évolution des systèmes électroniques. Nous présentons, dans cette section, les inventions qui ont contribué aux développements des systèmes domotiques.

1901 – 1920 : Bien que les appareils électroménagers ne soient pas ce que nous considérions comme «intelligents», néanmoins ils constituaient un exploit incroyable au début du XXe siècle. Ces réalisations ont commencé avec le premier aspirateur à moteur en 1901. Un aspirateur plus pratique fonctionnant à l'électricité a été inventé en 1907 [3].

1966 – 1967 : ECHO IV ordinateur de cuisine. Bien qu'il n'ait jamais été commercialisé, l'ECHO IV a été le premier appareil intelligent. Cet appareil pourrait calculer des listes d'achats, contrôler la température de la maison et commander d'autres périphériques.

1991 - Gerontechnology Gerontechnology combine la gérontologie et la technologie et facilite la vie des personnes âgées. Dans les années 90, il y avait beaucoup de nouvelles recherches et technologies dans ce domaine.

1998 - Début des années 2000 : La popularité des maisons connectées a commencé à augmenter et de différentes technologies domestiques ont commencé à émerger. Les maisons connectées sont soudainement devenues une option plus abordable, et donc une technologie viable pour les consommateurs.
Les maisons connectées d'aujourd'hui sont davantage axées sur la sécurité et le confort de la vie quotidienne. Elles sont durables et permettent d'assurer que les maisons ne dépensent pas d'énergie inutile. Elles aident également à prévenir des intrus (que nous soyons chez nous ou non).

1.4 Maison connectée dans le Monde

Depuis les années 90, lorsque les capteurs, les objets connectés et les technologies informatiques sont devenus rapidement accessibles, les scientifiques et autres professionnels y ont vu une occasion de développer des systèmes pour une maison connectée. Dans ce qui suit, nous citons quelques réalisations de maison connectée à travers le Monde [4].

1.4.1 Aux Etats Unis

La maison « Matilda », proposée par l'université de Florida, comporte un système de localisation des habitants basé sur les ultrasons.
« Gator Tech Smart House » est la deuxième génération de maison connectée « Matilda » qui comporte un grand nombre d'équipements «intelligents » tels que boîte aux lettres, portes d'entrée, lits, salle de bain, miroirs, plancher sensitif, etc.

1.4.2 En Asie

La maison connectée « Osaka », issue du projet de Matsuoka, détecte automatiquement les événements inhabituels causés par certaines pathologies ou accidents.
Le projet « Intelligent Sweet Home », en Corée, propose une maison connectée dédiée aux personnes âgées et handicapées incluant notamment un lit robotisé intelligent équipé d'une main connectée, un fauteuil roulant motorisé ainsi qu'un élévateur intelligent permettant de déplacer la personne entre le lit et le fauteuil roulant.

1.4.3 En Europe

La maison connectée de « British Telecom » et « Anchor Trust » en Angleterre surveille à distance l'activité des personnes.
Le projet européen « Handicom », issu de l'initiative AAL-119 « Ambient Assistive Living » du traité européen, élabore une infrastructure ouverte de services dont l'objectif est d'améliorer la qualité de vie des personnes âgées et handicapées dans le monde.
Le projet « PROSAFE » à Toulouse a été développé pour la surveillance continue

des personnes âgées à domicile. L'objectif est d'assurer le maintien des personnes âgées autonomes dans leur logement et d'envoyer automatiquement, sans intervention humaine, un signal d'alarme en cas d'urgence.

1.5 Travaux dans le contexte du Home Automation : état de l'art

Nous citons dans ce qui suit les recherches relatives à la maison connectée : [5] :

Dans l'ouvrage intitulé "A review of Internet of Things for smart home : Challenges and solutions ", une proposition de modèle de gestion des systèmes d'une maison connectée au cloud qui génère sa propre énergie a été présentée. (Biljana vd., 2016)

Dans l'étude, intitulée "Designing and Implementing a Lightweight WSN MAC Protocol for Smart Home Networking Applications", l'architecture du système et la conception GRAFCET (langage / méthode de programmation de puce) d'un périphérique réseau sont présentées pour les applications de maison connectée basées sur l'IoT. Le protocole de connexion WSN / MAC est recommandé dans le modèle (Chen vd.2017).

Dans le travail intitulé "Home Automation and Internet of Things ". Une proposition de système de domotique basée sur Arduino, qui contrôle tous les capteurs de la maison, est présentée. Il est souligné que le modèle nécessite peu de maintenance après l'installation et peut être facilement modifié (Singh vd, 2016).

Un système domotique basé sur les interactions de transducteurs intelligents avec efficacité énergétique / coût de revient a été introduit dans le travail intitulé "Home Automation System Based on Intelligent Transducer Enablers ". Le système comprend des points de conversion intelligents, une passerelle domotique et des points d'accès sans fil. (Albela vd, 2016).

Un modèle de système de surveillance à domicile flexible et peu coûteux basé sur le système d'exploitation Android est proposé dans l'étude intitulée "Internet of Things :

Ubiquitous Home Control and Monitoring System using Android based Smart Phone". Le système peut être géré par des appareils mobiles intelligents. Dans ce modèle, même si le WI-FI est offline, le système est accessible avec la technologie de communication 3G / 4G (Piyare, 2016).

Une étude intitulée "Research and application on the smart home based on component technologies and Internet of Things" présente un système de maison connectée IoT basé sur la SOA (architecture basée sur les services). Dans l'étude, il a été démontré que le support logiciel SOA offrait une proposition logicielle pouvant être contrôlée sur le Web pour répondre rapidement aux besoins et demandes instantanés (Li et Yu, 2011).

Dans le cadre des "Smart Home Mobile RFID based Internet Systems", une architecture de lecteur RFID a été proposée pour les applications et les services de la maison connectée. Le système a été développé comme alternative aux modèles de lecture RFID à forte consommation d'énergie. Dans ce contexte, il est entendu qu'un modèle d'efficacité énergétique élevé est visé. Des exemples d'utilisation de divers services à domicile intelligents tels que les programmes de lavage au travail, la cuisine, les achats et les soins de santé pour personnes âgées sont donnés à titre d'exemple de systèmes utilisant le système RFID (Darianian et Michael, 2008).

L'application "Smart Grid ", nommée "System Design of Internet-of Things for Residential Smart Grid ", fournit une conception de système IoT à grande échelle nécessitant un temps de réponse rapide pour de nombreux utilisateurs à domicile, Viswanath vd, 2016).

En tant qu'alternative à la technologie coûteuse de la maison connectée traditionnelle, la technologie de capteur sans fil ZigBee à haute rentabilité et la proposition de modèle de maison connectée IoT utilisant la technologie Cloud ont été proposées comme "The Design and Application of Low-Cost Smart Home Under the Internet of Things and Cloud Computing Platform " fournie (Wei et Qin, 2013).

L'architecture "WiFi Multi Access Point Smart Home IoT Architecture " offre un concept de réseau de capteurs sans fil spécialement conçu pour les applications domotiques. La nature innovante de ce réseau concerne l'utilisation simultanée de capteurs sélectionnés servant de points d'accès et de clients réseau pour le transfert et le routage d'informations.(Lech, 2016).

Dans l'ouvrage intitulé "The Possibility of Using VLC Data Transfer in the Smart Home", il propose un modèle IoT qui fournit une communication via la méthode VLC (Visible Light Communication) basée sur KNX. Système de gestion (Smart Home-Building Automation System) (Vanus vd, 2016).

Dans le travail intitulé "Smart Home System Based on Internet of Things ", une proposition de modèle IoT basée sur Arduino a été présentée. Elle a été développée pour contrôler différents types de dispositifs intelligents et d'objets IoT dans une maison à partir d'un système central unique. Le système est compatible avec les appareils Android et les ordinateurs personnels avec accès HTML (Verma vd, 2016).

1.6 Architecture et services d'une maison connectée

La maison connectée tient une place de plus en plus importante dans le logement. La maison est dotée de divers capteurs, permettant la connexion via une télécommande universelle ou un Smartphone. Ce qui permet de gérer les programmes des appareils de la maison n'importe où, en fonction des besoins. Elle définit une résidence dotée d'appareils électroménagers, d'éclairage, de chauffage, de climatisation, de téléviseurs, d'ordinateurs, de systèmes de divertissement, d'appareils ménagers de grande taille tels que laveuses, sécheuses, réfrigérateurs, congélateurs, ainsi que de systèmes de sécurité et de caméras capables de communiquer entre eux et d'être contrôlés à distance.

Ces systèmes sont constitués d'interrupteurs et de capteurs connectés à un concentrateur central contrôlé par le résident utilisant un terminal ou une unité

mobile connectée à des services Cloud via Internet.

La maison connectée offre la sécurité, l'efficacité énergétique, un faible coût d'exploitation. Dans la plupart des cas, son infrastructure est suffisamment flexible pour s'intégrer à une large gamme d'appareils de différents fournisseurs et normes.

1.6.1 Architecture

L'architecture de base permet de mesurer les conditions domestiques et de traiter les données instrumentées, en utilisant des capteurs pour mesurer les conditions domestiques et des actionneurs pour la surveillance de dispositifs embarqués à la maison.

Les appareils ménagers se connectent dans une architecture réseau appropriée prédéfinie et à l'aide de protocoles standard. L'architecture de base pour les maisons connectées utilisant l'IoT est illustrée à la figure 1.1.

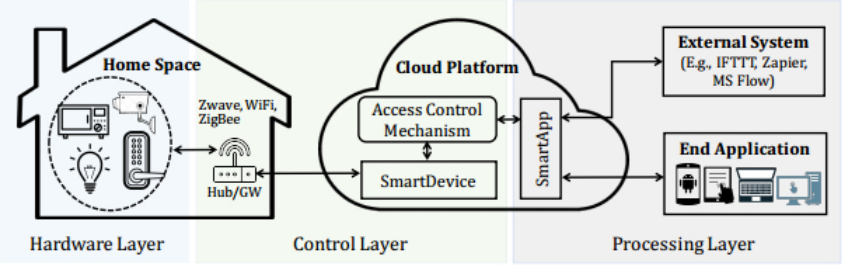

FIGURE 1.1: L'architecture de la maison connectée.

[6]

Nous comptons plusieurs architectures différentes, dont notamment [7] :

1. **Une architecture centralisée**

   Un contrôleur centralisé (Gateway/Cloud) reçoit des informations provenant de multiples capteurs et, une fois traitées, génère les commandes appropriées pour les actionneurs.

2. **Architecture distribuée**

   Toute l'intelligence du système est distribuée par tous les modules. Ces modules

sont des capteurs ou des actionneurs. Habituellement, ce sont des systèmes de câblage de bus ou les réseaux sans fil (Wi-Fi, Bluetooth).

3. **Architecture mixte**

   Ce sont des systèmes avec architecture décentralisée dans le sens qu'ils ont plusieurs appareils capables d'acquérir et de traiter des informations provenant de multiples capteurs et de les transmettre à d'autres appareils distribués par la maison. Par exemple, les systèmes basés sur ZigBee sont complètement sans fil.

   Les appareils connectés (capteurs, actionneurs) génèrent des données liées à l'environnement domestique et les envoient vers la Gateway (passerelle) qui joue le rôle d'un agrégateur de données avant de les acheminer vers la plateforme Cloud où ils seront traitées et stockées.

   Les passerelles connectées de type Edge déportent certaines fonctionnalités de Cloud sur le réseau local, tels que l'analyse, le traitement et le stockage de données.

1.6.2 Les services de la maison connectée

Les services offerts par l'automatisation pour la maison sont regroupés en fonction de cinq aspects ou domaines principaux, à savoir [8] [1] :

1.6.2.1 économie d'énergie

Les économies d'énergie ne sont pas quelque chose de tangible, mais un concept qui peut être atteint de plusieurs façons. Dans de nombreux cas, il n'est pas nécessaire de remplacer les appareils ménagers ou d'autres systèmes qui utilisent moins d'énergie. Le but final est de fournir une gestion efficace de celle-ci.

- CVC et chaudières : la planification et le zonage, être en mesure d'utiliser un thermostat de manière autonome.
  - Ils peuvent allumer ou éteindre la chaudière en utilisant une prise de contrôle, ou en utilisant un téléphone portable, un téléphone fixe, ou via le Wi-Fi et ou Ethernet.
- Contrôle automatique ou manuel des volets et stores électriques :

  - Protection automatique des stores ou des volets du vent, avec le même capteur de vent agissant sur toutes les installations.
  - Protection solaire automatique, par le même capteur solaire agissant sur tous les volets et stores.
  - Utilisation d'un produit ou groupe de produits pour activer ou désactiver le fonctionnement du capteur, avec une télécommande ou un contrôle central.
- Gestion de l'alimentation :
  - Rationalisation des charges électriques : déconnexion de l'équipement non utilisé prioritairement sur la base de la consommation d'énergie dans un temps donné.
  - Frais de gestion, en contournant les performances de certains appareils à des heures de taux réduit.

1.6.2.2 Confort

Le confort implique toutes les actions qui peuvent être menées pour améliorer le confort dans une maison. Ces actions peuvent être à la fois une responsabilité comme un actif ou mixte.

- Eclairage
  - Automatisation on/off à chaque point de lumière dans la maison
  - Règlement de l'éclairage en fonction du niveau de luminosité ambiante
- Automatisation de tous les systèmes en leur offrant un contrôle efficace et une manipulation facile
- Contrôle des périphériques via le réseau

1.6.2.3 Sécurité

Le système de technologie dans la maison connectée se compose d'une gamme de services de sécurité, dont notamment :

- Alarmes d'intrusion utilisées pour détecter ou empêcher la présence d'étrangers dans une maison ou un bâtiment.

— Détecteurs d'incendie (détecteur de chaleur et détecteur de fumée), détecteur de gaz (fuites de gaz), détecteur des fuites d'eau et les inondations, détecteur de la concentration de monoxyde de carbone dans les garages.
— Alerte médicale et soins à distance.
— Accès aux caméras IP.

1.7 Architecture d'une maison intelligente

1.7.1 Contribution du Cloud à la maison connectée

Le Cloud Computing est une plateforme partagée de ressources fournissant une variété de services à différents niveaux, de l'infrastructure de base, aux services d'application les plus sophistiqués facilement attribuées et déployées. En pratique, il gère les ressources de traitement, d'analyse, de stockage et de communication partagées par plusieurs utilisateurs dans un environnement virtuel et isolé hébergé à distance. La figure 1.2 illustre le concept global du Cloud Computing

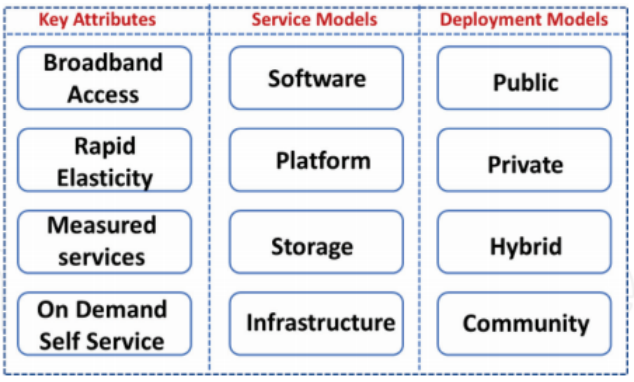

FIGURE 1.2: Le paradigme global du Cloud.

La maison connectée peut tirer parti des vastes ressources et fonctionnalités du Cloud pour compenser ses limitations en matière de stockage, de traitement, de communication, de prise en charge, de sauvegarde et de restauration. En effet, le Cloud peut prendre en charge la gestion des services IoT et l'exécution des applications complémentaires à l'aide des données produites[9].

La maison connectée peut être condensée uniquement sur les fonctions de base et critiques, en minimisant ainsi les ressources de la maison locale et s'appuyant sur les capacités et les ressources du Cloud. Elle se concentrera sur la collecte de données, le traitement de base et la transmission vers le Cloud pour un traitement plus complexe. Quant à la sécurité de la maison connectée, le Cloud peut être soit privé (local) où l'ensemble des ressources est hébergé et géré en interne assurant une haute sécurité, soit public si les ressources sont gérées par un fournisseur et ne sont accessibles que par internet.

La maison connectée et le Cloud ne sont pas simplement une fusion de technologies. Mais plutôt, un équilibre entre le Computing local et central ainsi que l'optimisation de la consommation de ressources. Une tâche peut être exécutée en local sur des objets IoT (une Gateway) dans la maison connectée ou sous-traitée au niveau Cloud. Dans ce cas, l'ensemble des ressources de la maison connectée, tels que les passerelles, les protocoles de communication, les Firmware, le Cloud Computing, les bases de données et les middlewares, est hébergé et géré par la plate-forme Cloud.

Une plate-forme Cloud typique assure la sécurité et l'authentification des périphériques, les courtiers de messages et la mise en file d'attente des messages, l'administration des périphériques, les protocoles, la collecte de données, la visualisation, l'analyse de fonctionnalités, l'intégration avec d'autres services Web et plateformes, l'évolutivité, les APIs pour la circulation de l'information en temps réel et les bibliothèques Open Source pour le développement des solutions.

Selon la demande, le fournisseur du Cloud offre des infrastructures en tant que service (IaaS) qui signifie l'externalisation de l'infrastructure matérielle (réseaux, stockage et serveurs), des plates-formes (PaaS) en tant que service qui signifie l'externalisation non seulement d'infrastructure matérielle mais également d'environnement logiciel (bases de données, couches d'intégration, runtimes) , ou bien des logiciels en tant que service (SaaS) qui permet de disposer à distance d'une solution logicielle.

D'une part, le modèle d'informatique triple, impliquant le Cloud, l'IoT et la maison

connectée, devrait minimiser le coût total des systèmes domotiques, en mettant davantage l'accent sur la réduction de la consommation des ressources à la maison. D'autre part, les fournisseurs des services Cloud devrait permettre aux utilisateurs de la maison connectée de mieux répondre à leurs besoins et de résoudre les problèmes complexes découlant du nouveau modèle de services IoT, maison connectée et Cloud.

1.7.2 Internet of Things (IoT)

Le paradigme de l'Internet des objets (IoT) fait référence aux appareils connectés à Internet. Les appareils sont des objets, tels que les capteurs et les actionneurs, équipés d'une interface de télécommunication, des unités de traitement, de stockage et d'applications.[10] [11].

IoT est un domaine intégrant l'automation, l'embarquée et les nouvelles technologies de communication, tous destinés à améliorer le confort, la sécurité et le bien-être. Ce qui s'applique directement à la maison connectée.

1.7.3 Architecture IoT

Le comité d'architecture du Forum mondial de l'IoT (IWF) a publié un modèle d'architecture IoT de référence en Octobre 2014.

Ce modèle est conçu en sept couches afin que chaque couche fournisse des informations supplémentaires permettant d'établir une terminologie commune :

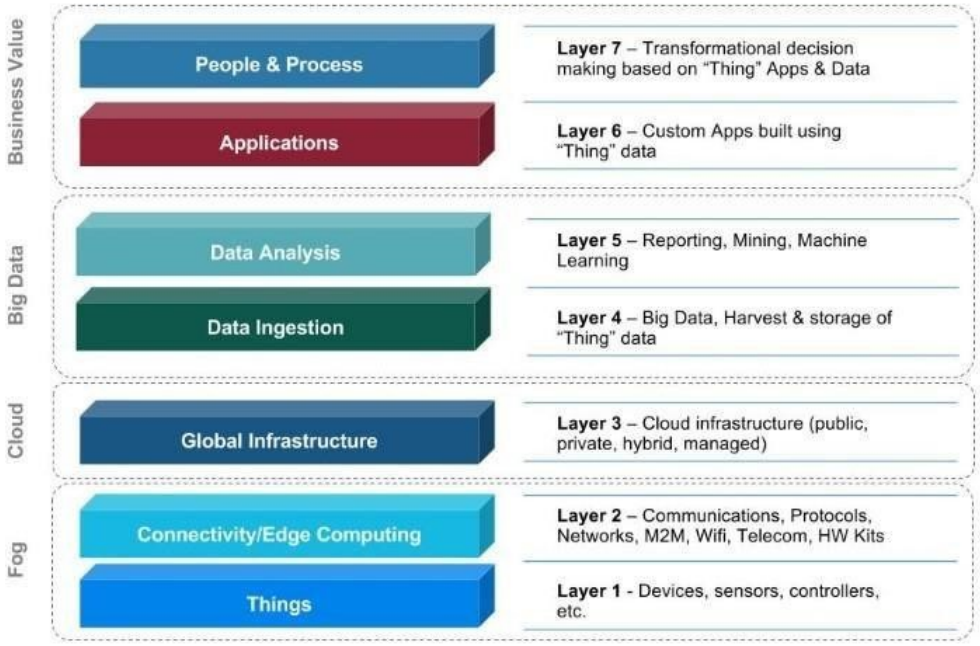

FIGURE 1.3: L'architecture du Forum mondial de l'IoT.

Couche IoT	**Fonctionnement**
Personnes	Cette couche implique le contrôle des appareils IoT par le biais des différentes applications.
Application	Cette couche interprète les données issues des devices IoT. Elle comprend les applications personnalisées qui utilisent réellement les données des objets.
Analyse des données	Applique des algorithmes de traitement et d'analyse sur un volume important de données pour générer des résultats fiables, utiles pour le système IoT.
Ingestion des données	Stockage de données
Couche Infrastructure Cloud	Le Cloud reçoit les données provenant des devices (Gateway /périphériques) et y applique plusieurs types de traitement avant de les orienter vers les bases de données.
Couche de connectivité et Edge Computing	La partie connectivité permet aux appareils de se communiquer entre eux et avec la plateforme Cloud via le réseau filaire ou sans files, l'Edge Computing permet le traitement préliminaire et le filtrage des données en local avant de les transférer au Cloud.
Couche physique	Cette couche inclut les appareils physiques connectés (Capteurs, Actionneurs, Machines)

TABLE 1.1: Les différents couches d'une infrastructure IoT

1.7.4 Technologies et protocoles de communication IoT

Les protocoles de communication jouent un rôle essentiel dans les systèmes IoT offrant la possibilité d'échanger de données dans le réseau. Ces protocoles spécifient

le format de l'échange de données, le codage des données, les schémas d'adressage pour les périphériques et le routage des paquets de la source vers la destination.

Wi-Fi et ZigBee sont considérés comme les technologies de communication les plus adaptées pour les solutions IoT en général, et les maisons connectées en particulier. Malgré la popularité du Wi-Fi, son application est limitée en raison de sa consommation d'énergie, c'est la raison pour laquelle de nombreux fabricants choisissent ZigBee pour développer des dispositifs de domotique sans fil. En effet, ZigBee présente une technologie de transmission plus sécurisée, nécessitant un faible débit de données et une faible consommation d'énergie.

Bluetooth est une autre technologie de transmission mais avec une portée plus faible que le Wi-Fi et ZigBee. La nouvelle version, appelée Bluetooth Smart ou Bluetooth Low Energy (BLE), est très utilisée pour la transmission dans une maison connectée, car elle consomme moins d'énergie et peut être intégrée à des Smartphones et autres appareils mobiles.

Outre les technologies, des protocoles de communication sont utilisés pour connecter les systèmes IoT dans la maison intelligente. MQTT est un protocole de messagerie ouvert qui permet le transfert des messages à partir des périphériques IoT (capteurs, actionneurs, téléphones mobiles ou systèmes embarqués) dans les réseaux à ressources limitées ou à latence élevée lorsque la vitesse du réseau est faible par exemple. La spécification principale de MQTT est qu'il utilise un modèle de publication / abonnement qui consomme très peu de ressources.

Un autre protocole de messagerie est AMQP (Advanced Message Queuing Protocol). Ce protocole est orienté vers des scénarios exigeant la performance, la flexibilité, la sécurité et un routage / livraison des messages fiable.

1.7.5 Services de la maison intelligente

L'évolution des technologies IoT et des services offerts par le Cloud ont fait de la maison connectée une maison intelligente, qui est non seulement contrôlée et pilotée à distance, mais aussi une maison autonome dans laquelle les équipements possèdent la capacité de décision en fonction des conditions de l'environnement domestique.

L'IoT contribue à la connexion Internet et à la gestion à distance des appareils mobiles, incorporant une variété de capteurs. Ces capteurs peuvent être connectés à des appareils domestiques, tels que la climatisation, des systèmes d'éclairage et autres dispositifs environnementaux. Ainsi, il intègre l'intelligence dans les appareils domestiques afin de fournir des moyens pour mesurer les conditions domestiques et de surveiller leurs fonctionnement.

Le Cloud Computing fournit une puissance de calcul, un espace de stockage et des applications évolutives pour le développement, la maintenance, l'exécution de services à domicile et l'accès aux périphériques domestiques partout et à tout moment. Les Services de Machine Learning (SVM) et de l'Intelligence Artificielle (IA) offerts par le Cloud permettent de contrôler et d'orchestrer toute la composition avancée de la maison intelligente et les prédictions des événements à domicile.

Les services offerts par la maison intelligente couvrent trois domaines principaux :

1.7.5.1 Sécurité

En matière de sécurité domestique, les systèmes d'alarmes, détecteurs de mouvement ou d'intrusion, interphones et d'autres sont utilisés pour surveiller les individus, prévenir les risques d'accident (incendie, fuite de gaz, etc.) et signaler des pannes (inondation, coupure de courant électrique, etc.).

La sécurité de la maison intelligente inclut également la centralisation de la surveillance et du contrôle de toutes les zones de la maison par la mise en place des passerelles à la périphérie permettant de gérer en local les données générées. Pour

l'intérieur des pièces, des micros ultra sensibles, des caméras invisibles, des détecteurs de fumées assurent aussi une grande sécurité.

1.7.5.2 Confort

Toutes les actions, effectuées manuellement par l'individu, peuvent être automatisées et intégrées dans des scénarios préprogrammés. L'élimination des gestes fastidieux et répétitifs fait gagner beaucoup de temps, économiser l'énergie et tranquilliser l'esprit de l'habitant. Parmi ces scénarios préprogrammés, on peut citer :

- Régulation en fonction de la luminosité extérieure Un capteur de luminosité peut être installé pour piloter l'éclairage en fonction d'un seuil prédéfini ou le réguler de façon continue afin d'obtenir une luminosité constante. Les éclairages s'allument, s'éteignent ou s'ajustent alors en variation pour optimiser les conditions de luminosité.
- Commande d'éclairage Le capteur de présence permet de déclencher automatiquement un éclairage quand nous passons devant un garage, couloir, cave, etc. La minuterie permet d'interrompre un circuit après un laps de temps déterminé.
- Programmation hebdomadaire et quotidienne Les Services de l'Intelligence Artificielle et de Machine Learning permettent de programmer et d'automatiser les périphériques selon les activités quotidiennes de l'habitant. Ce concept est illustré par un exemple pratique :

  « *Nous nous réveillons en douceur. Notre réveil, qui s'est coordonné avec notre agenda, sonne, tandis que les volets s'ouvrent tous seuls, et que le chauffage tourne déjà dans la salle de bain. Au moment du départ, nous prévenons notre maison que nous partons grâce à un geste élémentaire au moyen d'une télécommande (ou avec notre Smartphone). Un programme se lance, les lumières s'éteignent, le chauffage s'arrête, les portes se verrouillent. Avant de rentrer, le soir, nous prévenons notre domicile à distance. Le chauffage sera à la bonne température lorsque nous arriverons, et quand nous rentrerons et sans descendre de notre voiture, nous désactivons l'alarme, ouvrons le portail, éclairons l'allée si nécessaire et ouvrons la porte du garage au moyen d'une télécommande* ».

1.7.5.3 Economies d'énergie

La maison intelligente permet de diminuer la consommation d'énergie grâce aux automatismes et des capteurs, les équipements électriques connectés pilotent la consommation énergétique (chauffage, éclairage, eau, ventilation, etc.), tout en gardant sous contrôle le confort des zones occupées. Les systèmes de régulation permettent de maîtriser la consommation d'électricité, de gérer le chauffage et la production d'eau chaude avec un niveau de confort optimal.

Un détecteur de présence placé dans chaque pièce, par exemple, commande instantanément l'allumage ou l'extinction des éclairages, la mise en route ou l'arrêt du chauffage. Au jardin, par exemple, l'arrosage s'automatise et le détecteur se charge d'allumer les lumières dès la tombée de la nuit et de lancer l'irrigation des plantes. La maison intelligente utilise la programmation domotique via des scénarios qu'on peut déterminer en fonction des besoins spécifiques, évitant les pertes thermiques et les risques d'oubli ou de sécurité.

1.8 Maison intelligente : de Cloud Computing au Edge Computing

Le développement des services Cloud et la réduction des coût du matériel, ces dernières années, ont ouvert une nouvelle ère pour les maisons connectées, d'où l'apparition du concept du Edge Computing, qui étend le Cloud Computing jusqu'à la périphérie du réseau local. Ceci permet le développement de nouvelles applications et services pour l'Internet des objets. Edge Computing est, généralement, implémenté sur des Gateway afin de réduire la charge de travail sur le Cloud, d'une part, et la latence des systèmes IoT d'autre part [12].

Une Gateway de type Edge possède la capacité de traitement, d'analyse et de stockage préliminaire des données et permet de minimiser la dépendance au Cloud. Ce qui réduit les coûts de la solution IoT et augmente les performances en sécurité

et gestion de données et en termes de latence.

Les considérations relatives à l'intégration d'une passerelle peuvent inclure, entre autres, la connectivité dans le Cloud, les protocoles pris en charge, la complexité de la personnalisation et la prise en charge du prototypage.

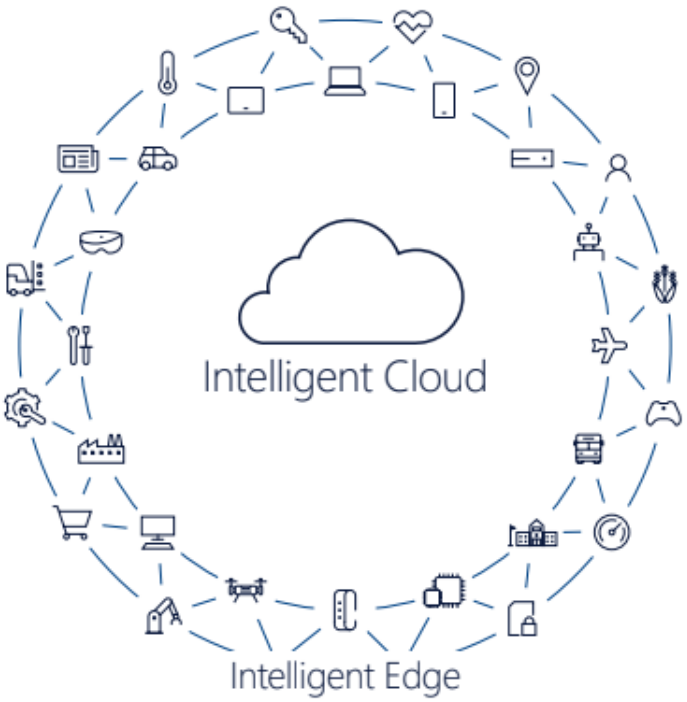

FIGURE 1.4: Du Cloud computing au Edge Computing

[12]

1.9 Conclusion

Dans ce chapitre, nous avons présenté la domotique, ses principales fonctionnalités et son impact sur la vie de l'individu. Nous avons explicité également l'intégration des trois composants de la maison connectée, IoT et le Cloud Computing, la gestion au niveau de la maison intelligente.

Dans le chapitre, nous allons présenter la plate-forme Cloud de Microsoft Azure ainsi que les services d'Internet des objets qui répondent aux défis de la maison intelligente.

Chapitre 2

Microsoft Azure

2.1 Introduction

Peu à peu, cette technologie de Cloud Computing remplace les infrastructures sur site dans les entreprises de toutes les industries. Son principal avantage est de permettre aux utilisateurs d'accéder à des ressources informatiques sans avoir à investir dans des Data Center ou à gérer des serveurs.
Toutefois qu'une application fonctionne dans le Cloud, utilise les services fournis par le Cloud, ou les deux, une sorte de plate-forme d'application est requise. Dans l'ensemble, une plate-forme d'application fournit des services accessibles aux développeurs pour créer des applications ou stocker des données.

Dans ce chapitre, nous allons présenter la plate-forme Cloud de Microsoft Azure et les services qu'elle offre.

2.2 Cloud Microsoft Azure

2.2.1 Présentation

Lancée en 2010, Azure est la plate-forme Cloud public gérée par Microsoft [12] (Plate-forme as a Service et Infrastructure as a Service) constituée d'une gamme de services intégrés (analyse, calcul, base de données, fonctions mobiles, mise en réseau, stockage et Web) que les entreprises utilisent sans acheter ni approvisionner du matériel physique permettant ainsi de créer et de gérer de manière optimale des solutions personnalisées de bout en bout, en faisant appel aux technologies les plus récentes.

FIGURE 2.1: La plate-forme d'Azure dans le monde

[12]

Cette plateforme est installée sur 52 pays [12], offrant une large accessibilité et une meilleure scalabilité à travers le monde.

2.2.2 Architecture générale

Cloud, Microsoft Azure permet de profiter de ressources de Cloud Computing à la demande en regroupant différents services Cloud. Habituellement, une solution Azure ne contient pas qu'un seul service Azure. Plusieurs services Azure sont utilisés par une solution. La liaison entre les différents services d'azure est illustrée à la figure 2.2.

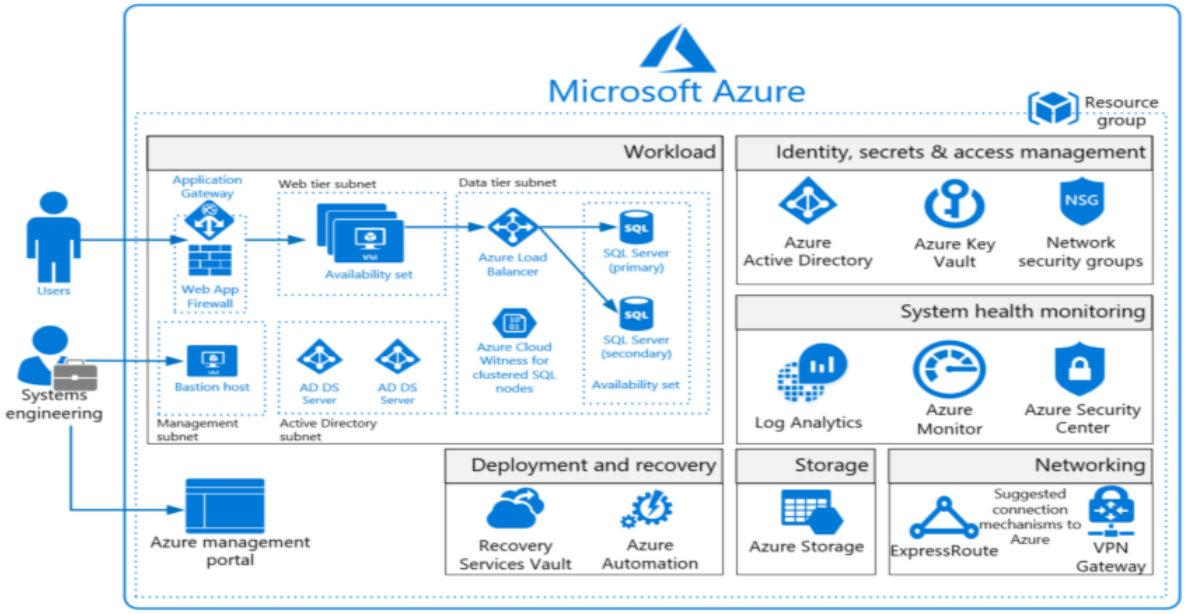

FIGURE 2.2: Architecture de la plate-forme Cloud Azure

2.2.3 Services Microsoft Azure

paragraph Azure offre accès à plus de 200 services entre les fonctionnalités IaaS et PaaS, qui se complètent et s'intègrent aux systèmes locaux. Nous citons, dans ce qui suit, les principaux [13] :

2.2.3.1 Services de calcul

Ce service comprend les Machines Virtuelles Azure, les applications mobiles pour créer et héberger l'application principale pour n'importe quelle application mobile, les applications Web pour créer et déployer rapidement des applications Web critiques à l'échelle, App services pour créer rapidement des applications Cloud performantes.

2.2.3.2 Services de données

Ceci inclut Microsoft Azure Storage tel que la base de données Azure SQL, Azure Cosmos DB, Stockage Table (NoSQL) et Base de données Azure pour MySQL.

2.2.3.3 Services réseau

Ceci inclut les fonctionnalités Azure telles Traffic Manager pour router le trafic entrant assurant une haute disponibilité, Azure DNS pour héberger les domaines DNS dans Azure et Virtual Network pour la mise en service des réseaux privés avec la possibilité de connexion à des bases de données locales.

2.2.3.4 Service IoT

C eci inclut deux parties principales :

- Azure IoT Hub permettant de connecter, surveiller et gérer des milliards de ressources IoT.
- Azure IoT Edge offrant la possibilité de déployer l'intelligence du Cloud sur des appareils en local.

2.3 Azure IoT

Microsoft Azure IoT donne la possibilité de développer des solutions IoT de bout en bout, en se basant sur les services Cloud d'Azure et d'autre services IoT. Azure IoT met à disposition des développeurs une gamme complète de SDK facile à utiliser, disponible en plusieurs langages de programmations tel que c, Python, c++ pour développer des applications personnalisées .

Pour connecter les appareils IoT, Azure IoT utilise des protocoles de communication IoT tels que HTTPS, AMQP et MQTT.

Microsoft Azure IoT offrent les fonctionnalités suivantes :

- La réception des données à partir des appareils IoT
- L'analyse du flux de données généré par le système IoT
- Le stockage et l'exploitation d'un volume important de données
- La visualisation des données
- La gestion et la surveillance des appareils IoT

Ces solutions présentent des implémentations de base des modèles de solutions IoT courants, qui permettent de réduire le temps de développement des applications.

2.3.1 Architecture

L'architecture de référence, pour les applications IoT recommandée par Azure, utilise des composants PaaS (Platform as a Service) comme illustrée dans la figure 2.3.

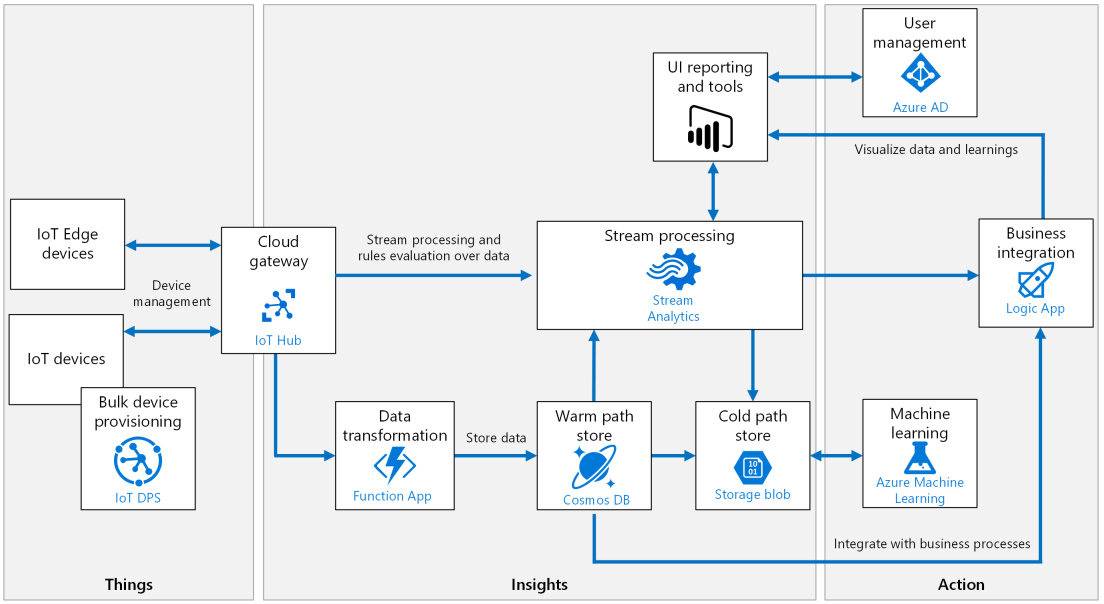

FIGURE 2.3: L'architecturede référence pour une application IoT d'Azure.

De plus, le flux passe par différents blocs avant d'être analysé. La figure 2.4 nous montre l'enchaînement de traitement du flux.

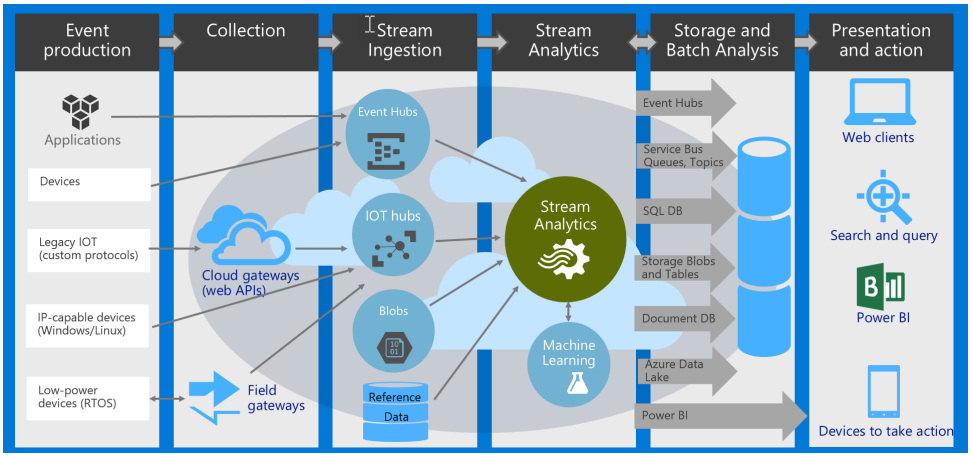

FIGURE 2.4: Traitement du flux de données.

Les applications IoT peuvent être décrites comme des éléments (appareils) envoyant des données qui génèrent des insights. Ces insights génèrent des actions permettant d'améliorer une activité ou un processus. Un four (l'élément) envoyant des données de température en est un exemple. Ces données sont utilisées pour déterminer s'il fonctionne comme prévu (l'insight). L'insight est utilisé pour hiérarchiser de manière proactive la planification de la maintenance par exemple (l'action).

Cette architecture est constituée des composants suivants :

2.3.1.1 Appareils IoT

Les appareils peuvent s'inscrire de manière sécurisée au Cloud et peuvent se connecter au Cloud pour envoyer et recevoir des données. Certains appareils peuvent être des appareils de périphérie qui effectuent un traitement de données sur l'appareil lui-même ou dans une passerelle locale. Azure IoT Edge est la solution proposée pour le traitement en périphérie.

2.3.1.2 Passerelle Cloud

Une passerelle Cloud fournit un Hub Cloud pour que les appareils se connectent de manière sécurisée au Cloud et envoient des données. Elle s'occupe aussi de la gestion des appareils et des fonctionnalités notamment la commande et le contrôle des appareils. Quant à la passerelle Cloud, Azure propose IoT Hub. Il s'agit d'un service Cloud hébergé qui reçoit les événements des appareils en faisant office de répartiteur de messages entre les appareils et les services back-end. IoT Hub fournit une connectivité sécurisée, l'ingestion des événements, la communication bidirectionnelle et la gestion des appareils.

2.3.1.3 Provisionnement des appareils

Pour l'inscription et la connexion de grands ensembles d'appareils, Azure recommande d'utiliser le service IoT Hub Device Provisioning (DPS). Le Service DPS permet d'attribuer des appareils à des points de terminaison Azure IoT Hub spécifiques à grande échelle et de les inscrire auprès de ces derniers.

2.3.1.4 Traitement de flux

Le traitement de flux analyse de grands flux d'enregistrements de données et évalue des règles. Pour ce traitement, Azure recommande Azure Stream Analytics. Stream Analytics peut exécuter une analyse complexe à grande échelle à l'aide de fonctions de fenêtrage temporel, d'agrégations de flux de données et de jonctions de sources de données externes.

La Machine Learning permet l'exécution d'algorithmes prédictifs sur des données de télémétrie historiques, afin de prendre en charge des scénarios tels que la maintenance prédictive. Pour le Machine Learning, le service Azure Machine Learning est recommandé.

2.3.1.5 Stockage du chemin à chaud

Contient les données qui doivent être disponibles immédiatement sur l'appareil pour la création de rapports et la visualisation. Pour le stockage du chemin à chaud, Cosmos DB est une base de données multi-modèle distribuée à l'échelle mondiale.

2.3.1.6 Stockage de chemin à froid

Contient les données qui sont conservées à plus long terme et qui sont utilisées pour le traitement par lots. Pour le stockage de chemin à froid, les données peuvent être archivées dans le Stockage Blob à moindre coût et pour une durée indéterminée. Ces données sont facilement accessibles pour le traitement par lots.

2.3.1.7 Transformation des données

Manipule ou agrège le flux de données de télémétrie. Parmi les exemples figurent la transformation de protocole, comme la conversion de données binaires au format JSON ou la combinaison de points de données.

2.3.1.8 Intégration des processus métier

Effectue des actions basées sur les insights provenant des données de l'appareil. Cela peut inclure le stockage de messages d'information, le déclenchement d'alarmes, l'envoi d'e-mails ou de SMS, ou l'intégration à CRM. Azure recommande d'utiliser Azure Logic Apps pour l'intégration des processus métier.

2.3.1.9 Gestion des utilisateurs

Permet de limiter les utilisateurs ou groupes qui peuvent effectuer des actions sur des appareils, comme la mise à niveau de microprogrammes. Elle définit également les fonctionnalités pour les utilisateurs dans les applications. Azure recommande d'utiliser Azure Active Directory pour authentifier et autoriser les utilisateurs.

2.3.2 Services Azure IoT

Azure IoT permet d'exploiter divers services Cloud en fonction des besoins. La figure 2.5 illustre les différents services Azure IoT [14] :

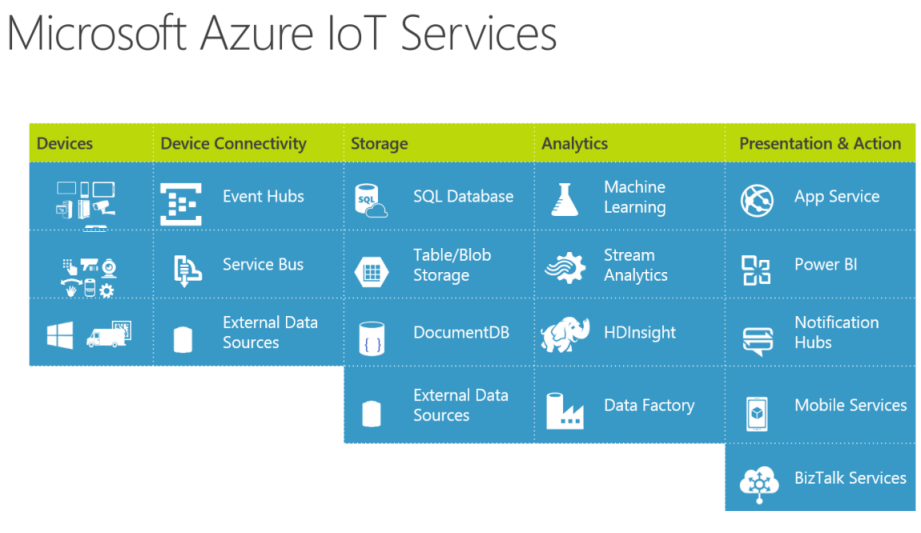

FIGURE 2.5: Les service d'Azure IoT. [14]

2.3.2.1 IoT Central

Il s'agit d'une solution SaaS qui permet de connecter, surveiller et gérer des appareils IoT. Ce service est destiné aux solutions simples qui ne nécessitent pas une

personnalisation approfondie du service.

2.3.2.2 Accélérateurs de solutions IoT

Il s'agit d'un ensemble de solutions PaaS utilisées pour accélérer le développement d'une solution IoT. Ce sont des solutions à personnaliser entièrement selon les besoins.

2.3.2.3 Azure IoT Hub

Ce n'est qu'un service qui assure une communication sécurisée et bidirectionnelle avec des millions de périphériques IoT.

2.3.2.4 IoT Edge

Ce service permet de déporter la capacité de traitement du Cloud sur les appareils de périphérie afin de réduire la dépendance au Cloud.

2.3.2.5 Azure Maps

Ce service fournit des informations géographiques aux applications Web et mobiles. Il existe un ensemble complet d'API Rest ainsi qu'un contrôle JavaScript basé sur le Web qui peut être utilisé pour créer des applications flexibles.

2.4 Azure IoT Hub

IoT Hub est un service managé, hébergé dans le Cloud, qui agit en tant que concentrateur de messages central pour la communication bidirectionnelle entre l'application IoT et les appareils qu'il gère. On utilise Azure IoT Hub pour générer des solutions IoT avec des communications fiables et sécurisées entre des millions d'appareils IoT et une solution backend hébergée dans le Cloud. On peut connecter quasiment n'importe quel appareil à IoT Hub.

2.4.1 Architecture

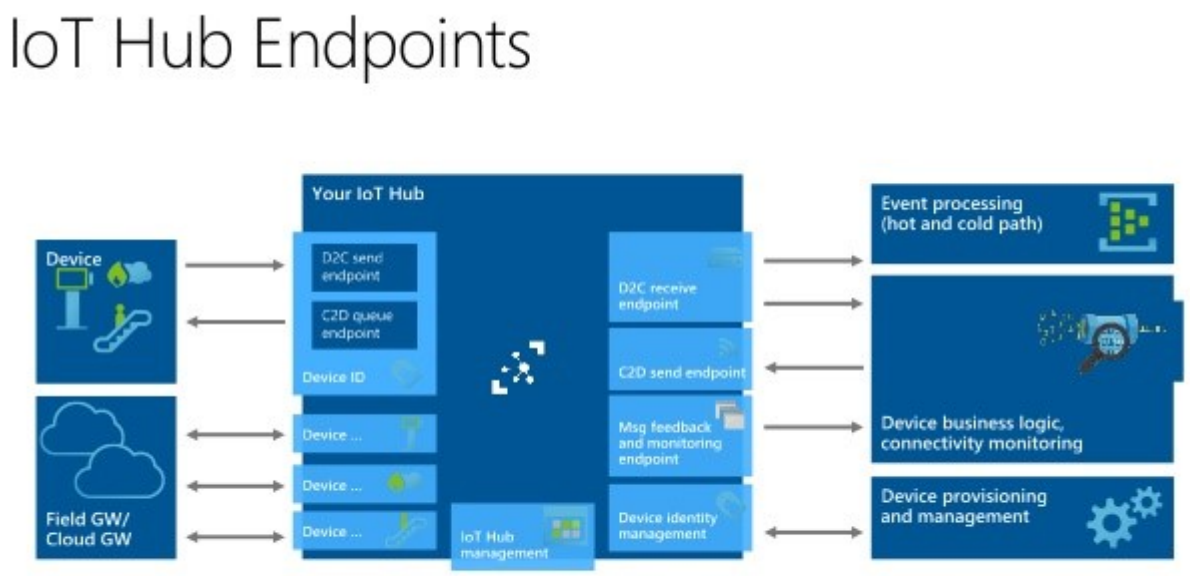

FIGURE 2.6: L'architecture de IoT Hub.

Les principales fonctionnalités d'IoT Hub se résument comme suit :

- Établir une connexion bidirectionnelle entre périphériques-Cloud.
- Authentifier les appareils et sécuriser les communications par certificats.
- Enregistrer des périphériques en utilisant le service de provision.
- Envoyer et recevoir plusieurs messages par jour (télémétrie) au/du Cloud.
- Utiliser les protocoles de communication IoT MQTT, AMQP et HTTP.
- Analyser par IA, stocker et traiter des données.
- Offrir des SDK open-source en multi-langages.

2.4.2 La communication

Il y a deux flux principaux pour lesquels des messages de priorité peuvent être requis :

— Communication Device To Backend (Device2Cloud - D2C)

— Communication Backend vers le périphérique (Cloud2Device - C2D)

2.4.2.1 Priorité du message sur la communication entre le périphérique et le backend

Le cas d'utilisation n'est pas complexe. Nous devons être en mesure d'envoyer des messages à l'appareil à notre backend avec des priorités différentes. Par exemple, lorsqu'une alerte est déclenchée. Même si Azure IoT est rapide, nous devons pouvoir consommer les messages aussi vite que possible.

La fonctionnalité la plus récente et la plus puissante,disponible pour ce scénario, est d'utiliser les règles de routage. Nous pouvons définir une règle de routage qui examine la bruyère du message et redirige le message vers Service Bus Topic, Service Bus Queue or Event Hub.

De cette façon, nous pouvons rediriger des messages avec des priorités différentes vers des consommateurs différents qui peuvent gérer. Tous les messages de priorité normale ou inférieure iraient à l'intérieur d'Azure IoT Hub, où l'on définit les règles de routage. Pour les messages critiques, on les ajoute à Device Twin.

2.4.2.2 Priorité du message sur la communication Backend to Device

Dans ce cas, nous devons envoyer des messages avec une priorité différente de Backend à Device. En ce moment, il n'y a pas de soutien pour une telle exigence. Device Twin peut être utilisé pour pousser les données avec une priorité plus élevée à l'appareil. Le principal avantage est que l'appareil recevra le message presque en temps réel, néanmoins la taille et la quantité des messages de priorité supérieure sont limitées.

2.4.3 Service "Provisionning"

Le provisionnement a plusieurs significations selon le secteur dans lequel le terme est utilisé. Dans le contexte d'IoT, la configuration est un processus en deux parties :

1. La première partie établit la connexion initiale entre l'appareil et la solution IoT en enregistrant l'appareil.

2. La deuxième partie applique la configuration appropriée au périphérique en fonction des exigences spécifiques de la solution pour laquelle il a été enregistré.

Une fois ces deux étapes terminées, nous pouvons dire que le périphérique a été entièrement provisionné.
Le service de provisionnement de périphérique présente de nombreuses fonctionnalités, ce qui le rend idéal pour le provisionnement de périphériques :

— Prise en charge sécurisée des attestations pour les identités basées sur X.509 et TPM.
— Liste d'inscriptions contenant l'enregistrement complet des périphériques. Cette liste contient des informations sur la configuration souhaitée du périphérique une fois qu'il est enregistré et peut être mise à jour à tout moment.
— Enregistrement des diagnostics pour s'assurer que tout fonctionne correctement.
— Prise en charge Multi-Hub pour permettre au service de provisionnement de périphériques d'attribuer des périphériques à plusieurs Hubs IoT. Ce service peut communiquer avec les concentrateurs de plusieurs abonnements Azure.
— Prise en charge inter-régions pour permettre au service de provision de périphériques d'attribuer des périphériques aux concentrateurs IoT d'autres régions.

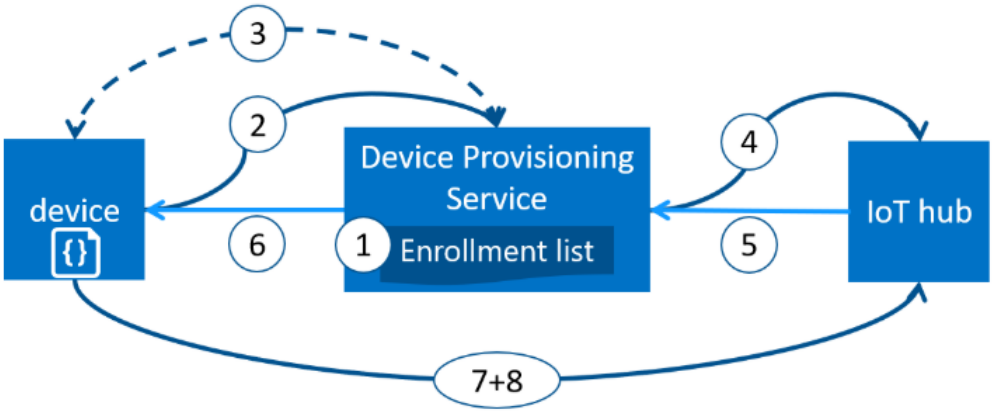

FIGURE 2.7: Porvisionnement du Device.

La figure .7 schématise le Provionnement du Device :

Phase 1 : L'utilisateur ajoute les informations d'enregistrement du périphérique à la liste d'inscription du portail Azure.

Phase 2 : Le périphérique contacte le point de terminaison du service de provisioning. Les informations d'identification seront transmises au service de provisioning pour prouver son identité.

Phase 3 : Le service de provisionnement valide l'identité du périphérique en validant l'ID et la clé d'enregistrement par rapport à l'entrée de la liste d'inscription.

Phase 4 : Le service de provisionnement enregistre le périphérique auprès d'un concentrateur IoT et renseigne l'état de jumelage souhaité du périphérique.

Phase 5 : Le Hub IoT renvoie les informations d'ID de périphérique au service de provisionnement.

Phase 6 : Le service de provisionnement renvoie les informations de connexion du concentrateur IoT au périphérique. L'appareil peut maintenant commencer à envoyer des données directement au Hub IoT.

Phase 7 : L'appareil se connecte au hub IoT.

Phase 8 : Le périphérique obtient l'état souhaité de son jumeau de périphérique dans le concentrateur IoT.

2.5 Azure IoT Device

2.5.1 Device

Les Devices peuvent s'inscrire de manière sécurisée au Cloud et peuvent se connecter au Cloud pour envoyer et recevoir des données. Chaque Device a son jumeau au niveau du Cloud « Device Twin ». Certains appareils ont la capacité du traitement sur l'appareil lui-même, on les appelle Edge Device.

2.5.2 Device Twin

Il s'agit d'un ensemble de propriétés spécifiques à chaque appareil. Device Twin contient trois types de structures dont les deux derniers sont similaires, mais ayant des rôles différents.

1. Tags : Informations spécifiques à l'appareil résidant sur le Hub IoT Azure et n'atteignant pas l'appareil
2. Desired Properties : Propriétés définies sur Azure IoT Hub (par Backend) et livrées à l'appareil
3. Reported Properties : Propriétés définies par Device et livrées à Azure IoT Hub (au Backend)

Les Tags sont utilisés lorsqu'on stocke des informations relatives à un périphérique sur le Backend sans les envoyer au périphérique. En revanche, Desired/Reported Properties sont envoyées/reçues, de l'appareil. Des propriétés telles que la configuration de l'appareil, l'état actuel ou différents compteurs peuvent être trouvées à l'intérieur de ceux-ci.

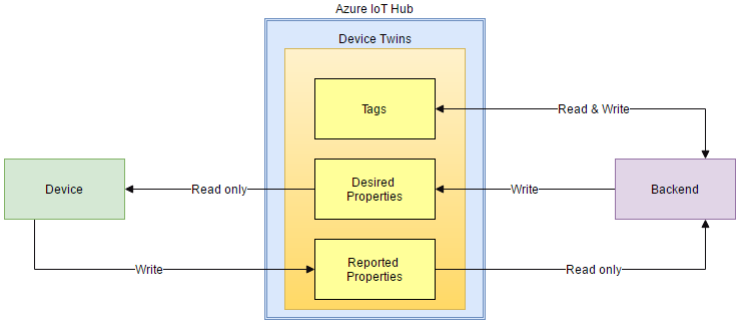

FIGURE 2.8: L'architecture du device Twin.

Toutes les informations relatives à Device Twin sont stockées au format JSON. Les Tags, les Desired Properties et les Reported Properties sont stockés séparément dans le même JSON, comme différents nœuds.

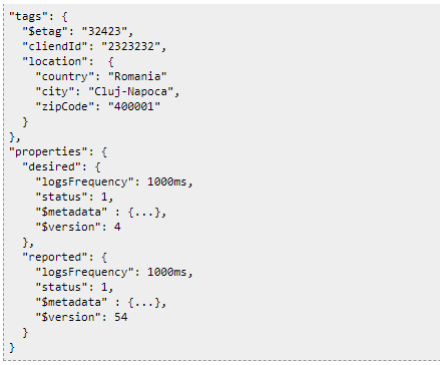

FIGURE 2.9: Présentation du format JSON.

Nous pouvons voir Device Twin comme un clone de notre Device dans le Cloud. Nous pouvons savoir à tout moment quel est l'état actuel de l'appareil et quelle configuration a été reçue par l'appareil et ce qui ne l'a pas été.

2.6 Azure IoT edge

Azure IoT Edge est un ajout récent à Azure IoT. Ce qui était initialement conçu comme une passerelle IoT est devenue une plate-forme d'Edge Computing à part entière. Il est placé entre la couche de périphériques et le Cloud public et possède des fonctionnalités Cloud pour gérer des périphériques en local [15].

Azure IoT Edge gère de manière transparente l'interaction entre les périphériques et le Cloud public. Il expose une fonctionnalité partielle d'IoT Hub pour l'authentification et la communication entre les périphériques locaux. Ceci est utile pour exécuter des applications IoT en mode hors connexion sans communiquer directement avec le Cloud public.

La plate-forme peut être déployée sur les systèmes d'exploitation Linux (ARM et x64) et Microsoft Windows. Selon les exigences de l'environnement d'exécution, Cette flexibilité en fait l'une des plates-formes les plus puissantes.

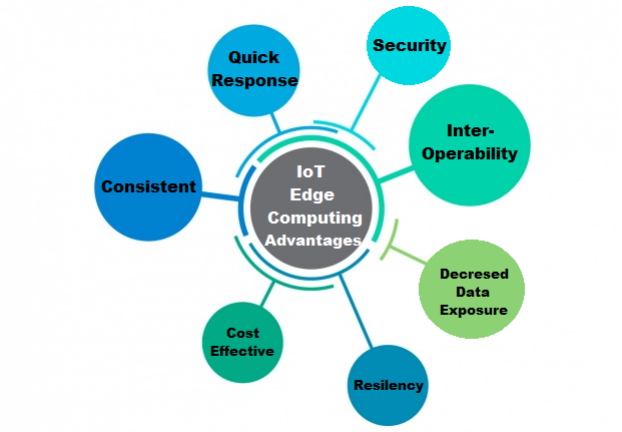

FIGURE 2.10: Azure IoT Edge.

2.6.1 Architecture Azure IoT Edge

Azure IoT Edge est conçu pour être extrêmement modulaire et extensible. Il s'appuie sur Moby, une chaîne d'outils open source alimentée par docker. La plate-forme Edge peut être facilement installée sur le périphérique avec quelques commandes.

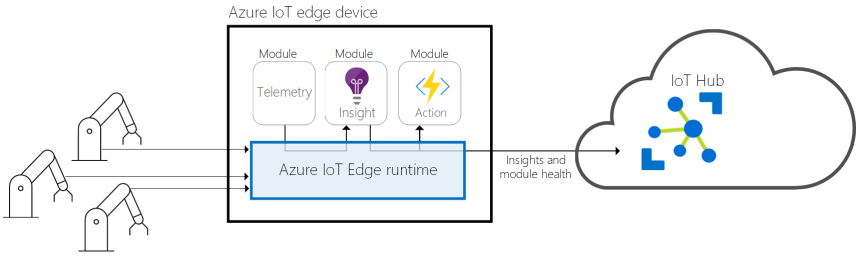

FIGURE 2.11: L'architecture d'Azure IoT Edge.

Nous expliquons, ci-dessous, le rôle de chaque composant de l'architecture d'Azure IoT Edge.

2.6.1.1 Conteneur Docker

Docker est un outil conçu pour faciliter la création, le déploiement et l'exécution d'applications à l'aide de conteneurs. Ces derniers permettent à un développeur de conditionner une application avec toutes les pièces dont il a besoin, telles que des bibliothèques et autres dépendances, et de l'expédier dans un package unique. Ainsi, grâce au conteneur, le développeur peut être assuré que l'application s'exécutera sur

toute autre machine Linux, quels que soient les paramètres personnalisés de cette machine susceptible de différer de la machine utilisée pour l'écriture et le test du code.
Une image de conteneur Docker est un package logiciel léger, autonome et exécutable, qui inclut tout ce qui est nécessaire à l'exécution d'une application : code, exécution, outils système, bibliothèques système et paramètres.
Les images de conteneur deviennent des conteneurs à l'exécution et dans le cas des conteneurs Docker, les images deviennent des conteneurs lorsqu'elles s'exécutent sur Docker Engine. Disponible pour les applications Linux et Windows, le logiciel conteneurisé fonctionnera toujours de la même manière, quelle que soit l'infrastructure. Les conteneurs isolent les logiciels de leur environnement et garantissent son fonctionnement uniforme, malgré les différences, par exemple, entre le développement et la mise en scène.

2.6.1.2 Organisation en Module

Ce sont des conteneurs Docker déployés sur un périphérique Edge et comprennent les logiciels définissant l'application à implémenter. Ces applications peuvent être des services Azure ou bien personnalisées. Les modules sont construits à partir de fichiers Docker et envoyés vers un registre de conteneur public ou privé. Les modules peuvent communiquer entre eux via une interface bien définie établie par le runtime.

2.6.1.3 Module Twin

Les modules étant une représentation directe des périphériques. Chaque périphérique est représenté par un module au niveau de la Gateway. Ils ont un jumeau (Module Twin) défini comme un document JSON. Le jumelage permet de lire la dernière configuration connue et également de définir les propriétés souhaitées sur le périphérique réel. Les jumeaux contiennent des métadonnées supplémentaires utiles pour interroger et filtrer les périphériques en fonction d'attributs tels que l'emplacement, le modèle, la marque et le numéro de série.

2.6.1.4 Azure IoT Edge Runtime

Le moteur d'exécution Azure IoT Edge active la logique personnalisée et Cloud sur les périphériques IoT Edge. Il repose sur le périphérique IoT Edge et effectue des opérations de gestion et de communication.
IoT Edge Runtime possède la communication des deux côtés de la chaîne, en l'occurrence, les périphériques et le Cloud. Il est installé en tant que binaire sur le système d'exploitation cible. IoT Edge Runtime s'exécute en tant que démon au sein du système d'exploitation et s'interface avec Moby pour gérer le cycle de vie des conteneurs déployés en tant que modules.
Le Runtime remplit plusieurs fonctions :

- Installer et mettre à jour les workloads sur le périphérique.
- Maintenir les normes de sécurité Azure IoT Edge sur le périphérique.
- Assurer que les modules IoT Edge sont toujours en cours d'exécution.
- Signaler l'état du module au cloud pour une surveillance à distance.
- Faciliter la communication entre les périphériques en aval et les périphériques IoT Edge.
- Faciliter la communication entre les modules sur le périphérique IoT Edge.
- Faciliter la communication entre le périphérique IoT Edge et le cloud.

On constate que les responsabilités du Runtime IoT Edge se divisent en deux catégories, à savoir, la communication et la gestion des modules. Ces deux rôles sont exécutés par deux composants faisant partie du Runtime IoT Edge. IoT Edge Hub est responsable de la communication, tandis que IoT Edge Agent déploie et surveille les modules. Edge Hub et Edge Agent sont des modules, comme tout autre module s'exécutant sur un périphérique IoT Edge.
Chaque périphérique IoT Edge exécute au moins deux modules : *edgeAgent* et *edgeHub*, qui font partie du Runtime IoT Edge. Le périphérique IoT Edge peut exécuter plusieurs modules supplémentaires pour un nombre quelconque de processus :

1. Azure IoT Edge Agent IoT Edge Agent est le premier composant de IoT Edge Runtime qui s'exécute en tant que conteneur. Il démarre chaque fois

que le périphérique Edge est mis sous tension. IoT Edge Agent est chargé de télécharger le manifeste de déploiement depuis le Cloud et de maintenir l'état de configuration souhaité du périphérique de périphérie.

Il extrait toutes les images de conteneur des registres et les exécute en fonction de la configuration prédéfinie. Sa responsabilité principale consiste à s'assurer que l'état et la configuration des conteneurs correspondent à la définition d'origine associée au périphérique.

Si un nouveau module est ajouté au manifeste via le portail Azure IoT, l'agent extrait l'image dès qu'il détecte la modification. De même, il ferme les conteneurs quand ils ne font plus partie du manifeste.Azure IoT Edge Agent gère l'interaction entre le Cloud et le moteur d'exécution local afin de conserver l'état souhaité.

2. Azure IoT Edge Hub 'est le deuxième composant du Runtime Azure IoT Edge, qui imite le Hub IoT dans le Cloud public. Il fournit essentiellement des fonctionnalités hors ligne d'un Hub IoT en exposant des services d'authentification et de communication aux périphériques. Un module représentant un périphérique dispose d'une logique pour s'authentifier auprès du Hub local. De même, il peut envoyer des données de télémétrie au Cloud qui les transmettra aux composants en amont, qui sont d'autres modules définis dans le cadre du manifeste.

   Edge Hub expose la même API que son homologue de Cloud public. Cette conception réduit les efforts requis pour authentifier les périphériques sur le Edge. Puisqu'il met en cache les informations d'identification après l'authentification du moteur d'exécution lors de la négociation avec IoT Hub dans le Cloud.

   Edge Hub agit en tant que courtier de communication facilitant la communication de périphériques locaux. Il prend en charge les protocoles standards de l'IoT Hub, notamment AMQP, MQTT et HTTP. Cependant, HTTP n'est pas disponible dans la version actuelle d'Edge Hub.

2.6.2 Routage des message

Le Hub IoT Edge gère la communication entre les modules, le Hub IoT et tous les périphériques IoT. Par conséquent, le module *edgeHubTwin* contient une propriété souhaitée, appelée routes, qui indique comment les messages sont transmis au sein d'un déploiement. Plusieurs itinéraires peuvent être déclarés dans le même déploiement. Les itinéraires sont déclarés dans les propriétés souhaitées de *edgeHub* avec la syntaxe suivante :

"route1" : "FROM ¡source¿ WHERE ¡condition¿ INTO ¡sink¿"

2.6.2.1 Source

La source spécifie d'où proviennent les messages. IoT Edge peut acheminer des messages à partir de modules ou de périphériques.

2.6.2.2 Condition

La condition est facultative dans une déclaration de route. On souhaite transmettre tous les messages de la source au récepteur :On peut également utiliser le langage de requête IoT Hub pour filtrer certains messages ou types de message répondant à la condition.

2.6.2.3 Destination

Le récepteur définit où les messages sont envoyés. Seuls les modules et IoT Hub peuvent recevoir des messages. Les messages ne peuvent pas être acheminés vers d'autres appareils. Il n'y a pas d'options génériques dans la propriété sink.

2.6.3 Les certificats

Les certificats IoT Edge sont utilisés pour les modules et les périphériques IoT afin de vérifier l'identité et la légitimité du module d'exécution du Hub IoT Edge auquel ils sont connectés. Ces vérifications permettent une connexion sécurisée TLS (Transport Layer Security) entre le Runtime, les modules et les dispositifs IoT.
Comme le IoT Hub lui-même, IoT Edge nécessite une connexion sécurisée et cryptée à partir des dispositifs IoT et des modules IoT Edge. Pour établir une connexion TLS

sécurisée, le module IoT Edge Hub présente une chaîne de certificats de serveur aux clients qui se connectent afin de leur permettre de vérifier son identité.
La figure 2.12 illustre l'utilisation des certificats par IoT Edge. Il peut y avoir zéro, un ou plusieurs certificats de signatures intermédiaires entre le certificat racine de l'AC et le certificat de l'AC du périphérique, selon le nombre d'entités impliquées.

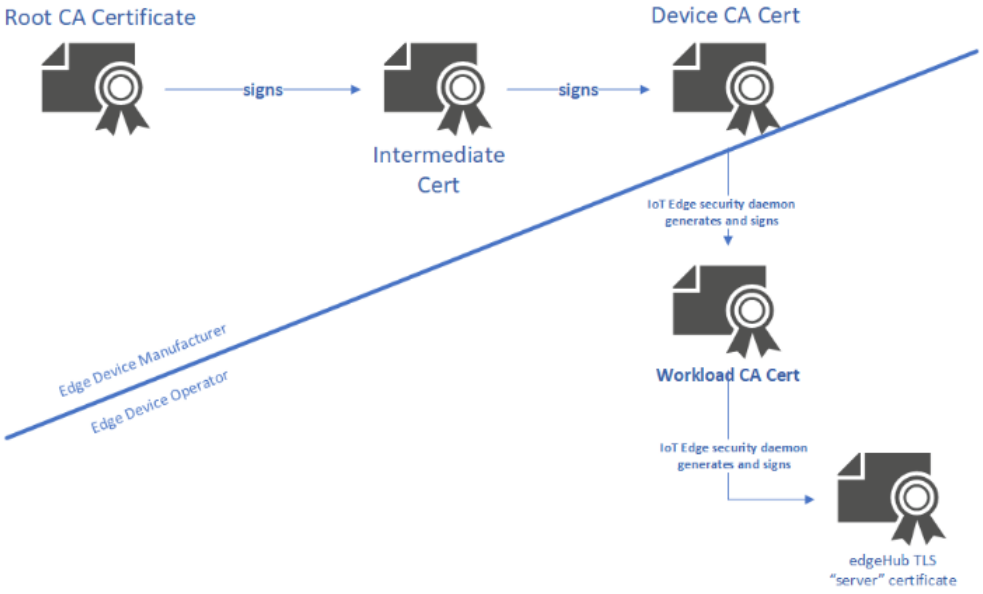

FIGURE 2.12: chaîne de certificats.

2.6.3.1 Autorité de certification AC

L'Autorité de Certification « AC » est une entité qui émet des certificats numériques. Une autorité de certification agit en tant que tiers de confiance entre le propriétaire et le destinataire du certificat.
Un certificat numérique certifie la propriété d'une clé publique par le destinataire du certificat. La chaîne de confiance du certificat fonctionne en émettant initialement un certificat racine, qui est la base de la confiance dans tous les certificats émis par l'autorité. Ensuite, le propriétaire peut utiliser le certificat racine pour émettre des certificats intermédiaires supplémentaires.

2.6.3.2 Certificat racine CA

Un certificat d'AC racine est la racine de la confiance de l'ensemble du processus. Dans un processus de fabrication typique pour créer des dispositifs sécurisés, les certificats d'AC racine sont rarement utilisés directement, principalement, en raison du risque de fuite ou d'exposition. Le certificat racine de l'AC crée et signe numériquement un ou plusieurs certificats intermédiaires de l'AC.

2.6.3.3 Certificat intermédiaire

Le certificat de l'AC de l'appareil est généré et signé par le certificat intermédiaire final de l'AC dans le processus. Ce certificat est installé sur le périphérique IoT Edge lui-même, de préférence dans un stockage sécurisé tel qu'un module de sécurité matériel (HSM). En outre, un certificat CA de périphérique identifie de manière unique un périphérique IoT Edge.

2.6.3.4 Certificat Device CA

Ce certificat est généré et signé par le « certificat CA de l'appareil ». Ce certificat, qui n'est autre qu'un certificat de signature intermédiaire, est utilisé pour générer et signer tout autre certificat utilisé par le Runtime IoT Edge.

2.6.3.5 Certificat de serveur IoT Hub

Le certificat de serveur de concentrateur IoT Edge est le certificat présenté aux périphériques leaf et aux modules pour vérification d'identité lors de l'établissement de la connexion TLS requise par IoT Edge. Ce certificat présente la chaîne complète des certificats de signature utilisés pour le générer jusqu'au certificat racine de l'AC, auquel le périphérique IdO feuille doit se fier.

2.7 Service de développement- Azure SDK

Pour simplifier et accélérer le développement des solutions personnalisées répondant aux exigences de l'utilisateur, Azure offre des Kits Software de développement SDK

permettant de développer des applications sur les devices IoT ainsi que sur la Gateway IoT Edge, les kits de développement sont les suivant :

2.7.1 Device SDK

Elles permettent de développer des applications qui s'exécutent sur les appareils IoT et implémentent plusieurs fonctionnalités tel que l'envoi de la télémétrie au Cloud.

2.7.2 Service SDK

Elles permettent de gérer IoT Hub et envoyer des messages aux périphériques IoT.

2.7.3 Azure IoT Edge SDK

Elles permettent de configurer des passerelles dédiées aux périphériques qui n'utilisent pas un des protocoles supportés. De plus, ces passerelles ont la possibilité de traiter, analyser les messages.

2.8 Sécurité

Microsoft Azure utilise différents contrôles physiques d'infrastructure et opérationnels pour sécuriser Azure. Cependant, il est nécessaire d'exécuter des actions supplémentaires pour protéger nos charges de travail. Pour cela, on active Azure Security Center pour renforcer rapidement l'état de sécurité et se protéger contre les menaces.

FIGURE 2.13: Azure Security Center.

Azure Security Center offre une protection complète grâce à la possibilité de détecter les menaces mais aussi de se protéger. En utilisant le Machine Learning pour traiter des milliards de signaux dans l'ensemble des services et systèmes Microsoft, Azure Security Center alerte en cas de menace sur l'environnement, par exemple, des attaques de force brute RDP (Remote Desktop Protocol) ou des attaques par injection de code SQL.

2.8.1 Protection des serveurs Windows et Linux

Azure Security Center contribue à protéger les serveurs et clients Windows avec la protection avancée contre les menaces Windows Defender et les serveurs Linux avec l'analyse des comportements.

2.8.2 Protection des applications Cloud

Azure Security Centre protège les applications exécutées sur Azure App Service en signalant les comportements susceptibles de traverser les pare-feu d'applications web. Il aide également à protéger d'autres services Cloud tels que les groupes identiques de machines virtuelles et les conteneurs.

2.8.3 Protection des données

Les avancées en matière de Big Data et de Machine Learning permettent à Azure Security Center de détecter les accès aux bases de données et modèles de requêtes anormaux, les attaques par injection de code SQL, et d'autres menaces ciblant les bases de données SQL dans Azure. De plus, on peut bloquer les menaces sur le stockage Azure, notamment les accès à partir d'un emplacement inhabituel, les accès anonymes inhabituels, l'extraction de données inhabituelle ou les suppressions inattendues.

2.8.4 Protection d'une solution IoT

Azure IoT a été conçu pour la sécurité, en simplifiant la complexité de la solution de sécurité IoT grâce à une protection intégrée à chaque étape du déploiement, y

compris les services et les appareils Cloud, et en minimisant les faiblesses de sécurité, où qu'elles soient. De plus, on a la capacité d'optimiser les paramètres de sécurité et améliorer le degré de sécurisation à l'aide de recommandations exploitables dans l'ensemble des machines virtuelles, réseaux, applications et données.

2.9 Conclusion

Dans ce chapitre, nous avons abordé la plate-forme Microsoft Azure et les concepts de base d'une solution Azure IoT Edge. Nous avons présenté la manière dont les technologies existantes telles que Docker, Moby et les conteneurs sont utilisées pour prendre en charge le concept des modules Edge. Le moteur d'exécution Runtime se compose de deux modules spéciaux, edge Agent et Edge Hub ayant chacun un certain ensemble de tâches à leur charge.
Nous avons examiné la manière dont les jumeaux sont utilisés pour la gestion des périphériques, mais plus spécifiquement dans IoT Edge. Nous avons vu comment le routage au niveau de périphérique Edge permet aux modules découplés de communiquer entre eux. Dans le troisième chapitre, nous allons définir notre environnement de développement afin de concevoir notre solution avec les outils adéquats.

Deuxième partie

Implémentation

Chapitre 3

Implémentation

3.1 Introduction

Comme nous l'avons cité précédemment, le nombre important d'appareils connectés engendre des contraintes sur le Cloud. En amont, son extensibilité n'est pas infinie en terme de ressources. En aval et pour le client, un surcoût peut être pénalisant. De plus, le volume considérable de données à traiter peut peser sur les temps des opérations et provoquant ainsi une réduction de la qualité de service . En traitant les données au plus près du capteur, nous pouvons bénéficier d'une latence réduite et d'une qualité de service améliorée tout en réduisant les coûts économique pour l'usager du Cloud.

Pour donner une réponse à cette problématique qui nous est assignée, nous aborderons dans ce présent chapitre la présentation de l'architecture générale de la solution que nous avions adoptée. Le coeur de cette solution s'articule autour du développement multicibles d'une passerelle transparente de type Edge tenant compte des concepts et des outils proposés par la plate-forme Cloud Microsoft Azure. Nous expliciterons les étapes de son développement, de son déploiement et de sa validation. Un scénario de mise en oeuvre sera exploité pour valider notre développement. Une application sur un device y sera développée pour valider la solution globale.

3.2 Architecture Générale de la solution

L'architecture générale de la solution à développer s'articule sur trois niveaux à savoir les noeuds d'extrémité, la passerelle transparente et le Cloud. Notre développement se concentrera essentiellement sur le développement de la partie passerelle. Il est clair que pour valider la solution préconisée, le développement d'une application d'extrémité sera impératif. La mise en oeuvre globale sera subordonnée à la maîtrise des services Cloud IoT Azure.

De base, le rôle de la passerelle est double ; permettre des communication D2C et C2D avec des protocoles qui peuvent être différents en amont et en aval de cette celle ci. Des fonctions traditionnelles aux passerelles peuvent y être ajoutées telles que la

mise en cache, le filtrage, le mandatement, ... Des applications spécifiques peuvent y être envisagées.

L'idée de la solution consistera en l'exploitation des concepts introduits par le Cloud Azure pour développer cette passerelle. Dans ce contexte, le device Edge se présente comme un noeud d'extrémité (device) pouvant être doté de ressources lui permettant des traitements dits "lourds" ; donc bien adaptés à l'implémentation d'une passerelle.

La figure 3.1 illustre l'architecture prototype de la solution que nous avions mise en oeuvre.

L'architecture (software) en modules conjuguée aux mécanismes de routage de messages ainsi que l'utilisation de conteneurs s'avère judicieuse pour implémenter toute fonctionnalité (y compris spécifique) de toute passerelle. Par leurs rôles dédiées à un "device edge", les deux modules "edgeHub" et "edgeAgent" seront les deux conteneurs obligatoirement présentes dans toute solution.

Comme on le remarque sur cette même figure, trois communications sont envisagées C2D, D2C et D2D. Les deux premières sont intrinsèquement dépendantes du Cloud. La dernière a été un développement spécifique que nous avions introduit. Il pourra servir aisément pour l'implémentation d'une fonction proxy comme permettre une fonctionnalité "offline" par rapport au Cloud.

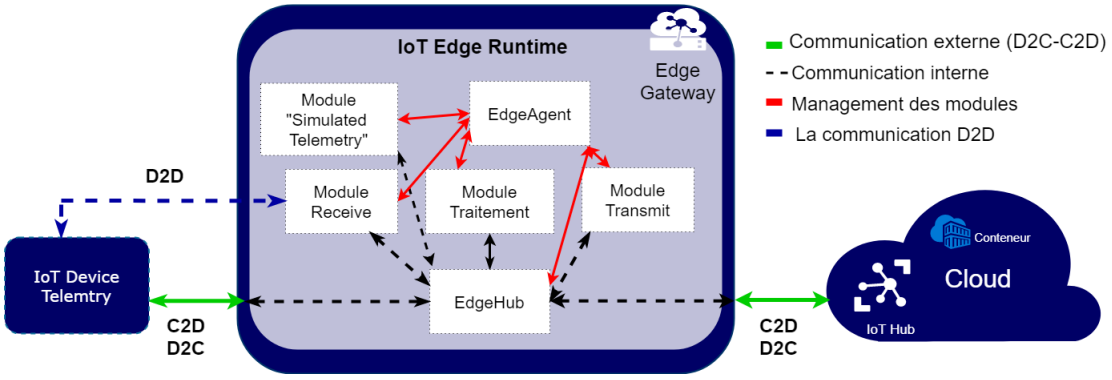

FIGURE 3.1: Architecture générale de la solution.

Nous aurons donc à développer deux types d'application ; celles qui seront portées par le IoT device Edge et celles qui le seront par l'IoT device.

A titre illustratif et pour la bonne compréhension de notre développement, nous nous sommes imposées les noms de domaine suivants :

- iotdevice.mondomaine.io adresse IP : 192.168.1.2
- iotdeviceedge.mondomaine.io adresse IP : 192.168.1.1
- iotedgehub.mondomaine.microsoft.com adresse IP : fournisseur des services Cloud

3.3 Protocoles de communication

De base et comme introduit dans le chapitre précédent, Microsoft Azure offre le choix entre trois protocoles de communications dédiés à l'IoT, à savoir, MQTT(s), AMQP(s) et HTTP(s) tous donc portés par la pile protocolaire DoD. Le choix de l'un ou l'autre des protocoles se fera suivant un paramètre qui sera porté par le fichier de configuration du device.

MQTT, AMQP sont utilisés pour la transmission de données bidirectionnelles Device to Cloud/Cloud to Device en passant par l'IoT Hub aussi bien pour les Iot Device Edge que pour les Iot Device.

Des protocoles issus des bus industriels (modbus, Can, ...) peuvent être envisagés pour l'intercommunication entre les noeuds d'extrémités tout en faisant appel au api adéquates.

Indépendamment des protocoles de transmissions utilisés et sachant l'utilisation plus que probable dans le contexte d'usage de notre projet de la pile protocolaire DoD, nous avons opté pour le protocole HTTP(S) pour l'implémentation de la liaison D2D.

3.4 Choix du langage de programmation

Pour le développement de solutions, Azure offre une gamme complète de kit de développement software et donne aux développeurs la possibilité de choisir parmi une multitude de langages de programmation ; on cite Python, C ,C/C++, etc. Dans notre cas, il nous été demandé d'utiliser le langage C/C++. Nous nous intéressons donc au kit azure IoT C SDK

3.5 Le développement multicible

Pour le device edge et sachant la technologie de conteneurisation qui est le fondement du fonctionnement des modules, le développement multicible devient aisée pour ce type de noeud (ou device). Cette opportunité a été saisie dans notre cas pour restreindre tous nos développement sur des conteneurs portés par des machines virtuelles. Dès que la solution est validée, il devient aisé de "pousser" la solution sur la cible en prenant soins d'y installer au préalable le "runtime" Iot Edge Device.

Concernant le IoT Device, un travail préalable de portabilité de l'API C IoT Azure Client est un impératif.

3.6 Les opérations préalables à réaliser

3.6.1 Provisionnement

Approvisionner un appareil signifie lui attribuer une identité afin qu'il puisse communiquer avec IoT Hub. Cette identité réside dans le Cloud et est attribuée sous forme d'une chaîne de connexion unique, associant ainsi un périphérique physique à une identité de périphérique. Ce procédé comprend les trois étapes suivantes :

- Enregistrement de l'appareil
- Démarrage de l'appareil
- Vérification de l'enregistrement de l'appareil

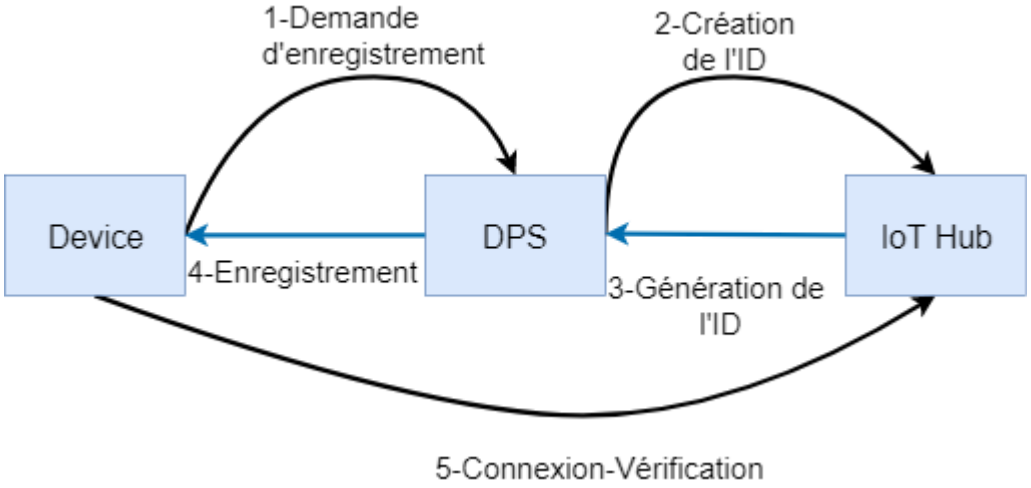

FIGURE 3.2: Provisionnement d'un appareil

3.6.1.1 Enregistrement de l'appareil IoT

Les étapes de création de l'identité du Device se résument comme suit :

Dans le menu de navigation du hub IoT, nous ouvrons « **appareils IoT** », puis nous sélectionnons « **ajouter** » pour inscrire un nouvel appareil dans notre IoT Hub. Nous avons inscrit deux appareils IoT ,la machine Ubuntu et la carte ESP32 ,leurs avons attribuee les **ID Ubuntu-234** et **ESP32-user** ,puis nous sélectionnons « **Enregistrer** » pour valider .Cette action permet de créer une identité d'appareil au niveau de l'IoT Hub.

Pour la Gateway ,nous avons inscrit un appareil de type IoT Edge correspondant à l'Identifiant **IoTEdgeGateway**.

Une fois l'appareil créé, nous ouvrons ce dernier à partir de la liste figurant dans le volet « **Appareils IoT** ». nous copions les « **Chaîne de connexion – clé primaire** » que nous utiliserons par la suite pour configurer les périphériques physique.

FIGURE 3.3: Enregistrement d'un appareil

3.6.1.2 Configuration

Configurer un appareil signifie que nous associons la chaîne de connexion de l'identité au niveau du IoT Hub à l'appareil physique.
La configuration de la Gateway ce fait au niveau de fichier config.yamel, en ajoutant la chaine de connection (l'identifiant de cette appareil au niveau du Cloud). La figure ci après illustre cette opération.

Nous choisissons le protocole de communications pour la transmission des données. Enfin, nous validons cette configuration par l'établissement de la connexion entre Device et IoT Hub.
Pour chaque Device, nous allons expliciter la configuration pour le Deice Edge et pour le Device dans la partie y afférente.

3.6.1.3 Vérification de l'enregistrement de l'appareil

Une fois l'inscription réussie, le service Device Provisioning envoie l'URI du Hub IoT, l'ID de l'appareil et la clé chiffrée. Une fois la connexion au Hub établie,

l'appareil doit apparaître dans l'Explorateur « **Appareils IoT** » du Hub IoT.

L'application cliente IoT sur l'appareil peut alors se connecter au Hub Il suffit, pour cela, de configurer la chaîne de connexion dans l'application implémentée au niveau du périphérique physique.

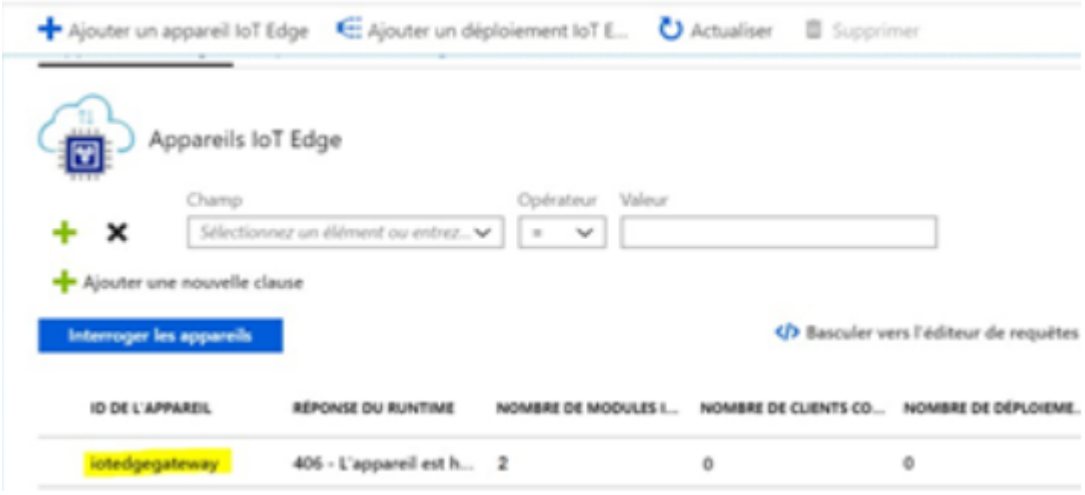

FIGURE 3.4: La vérification du l'enregistrement de l'appareil.

3.6.1.4 Monitoring de démarrage du device

Étant donné que l'appareil IoT est maintenant inscrit avec une instance de service Device Provisioning, l'appareil peut désormais démarrer et être appelé par le service d'approvisionnement pour être reconnu à l'aide du mécanisme de certification. Une fois l'appareil reconnu par le service d'approvisionnement, il est affecté à un IoT Hub.

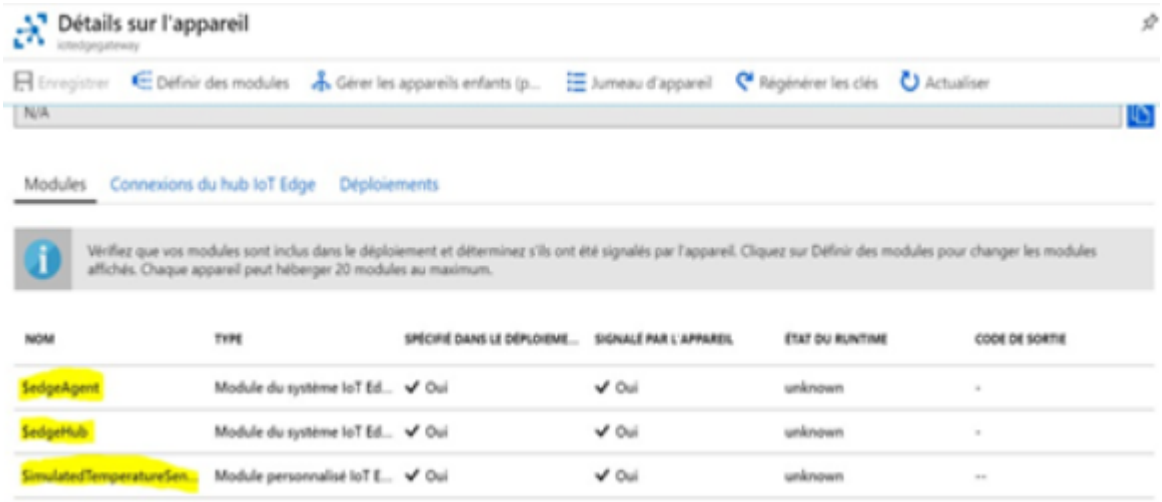

FIGURE 3.5: Démarrage de l'appareil

3.6.2 sécurité

Azure IoT Hub accorde l'accès à l'appareil en vérifiant un jeton par rapport aux stratégies d'accès partagées et aux informations d'identification de sécurité du registre d'identité et cela, via les informations d'identification de sécurité spécifié par la chaîne de connexion de l'appareil.

Comme pour IoT hub,IoT edge offre des communications sécurisées par certificats auto signés ou non pour vérifier si un device donnée IoT possède le privilège de communication avec la Gateway .
Une communication sécurisée entre les appareils est assurée par une infrastructure à clé publique (PKI) locale ou d'un fournisseur de service.Pour qu'un appareil en aval se connecte à la passerelle, il doit avoir une confirmation d'identification des deux côtés (appareil en aval et la passerelle).

La Gateway Edge utilise pour l'identification des appareil IoT un certificat X.509 et une série de certificats signées par une autorité de certification. Ces certificats seront présentés lors d'une ouverture de connexion et seront vérifiés pour établir la connexion.Ces certificats sont définis dans le fichier de configuration config.yam du device Edge (figure ci-dessous).
Comme pour l'IoT Hub , IoT Edge nécessite une connexion sécurisée et chiffrée à

partir de périphériques IoT en aval (ou feuille) . Pour établir une connexion TLS sécurisée, le module EdgeHub présente une chaîne de certificats aux clients afin qu'ils puissent vérifier son identité. Les certificats IoT Edge sont utilisés en aval pour les périphériques IoT afin de vérifier leurs identité et légitimité. Ces vérifications permettent une connexion sécurisée TLS (Transport Layer Security) entre le runtime et les périphériques IoT.

3.7 Développement de IoT Edge Gateway

3.7.1 Infrastructure de développement

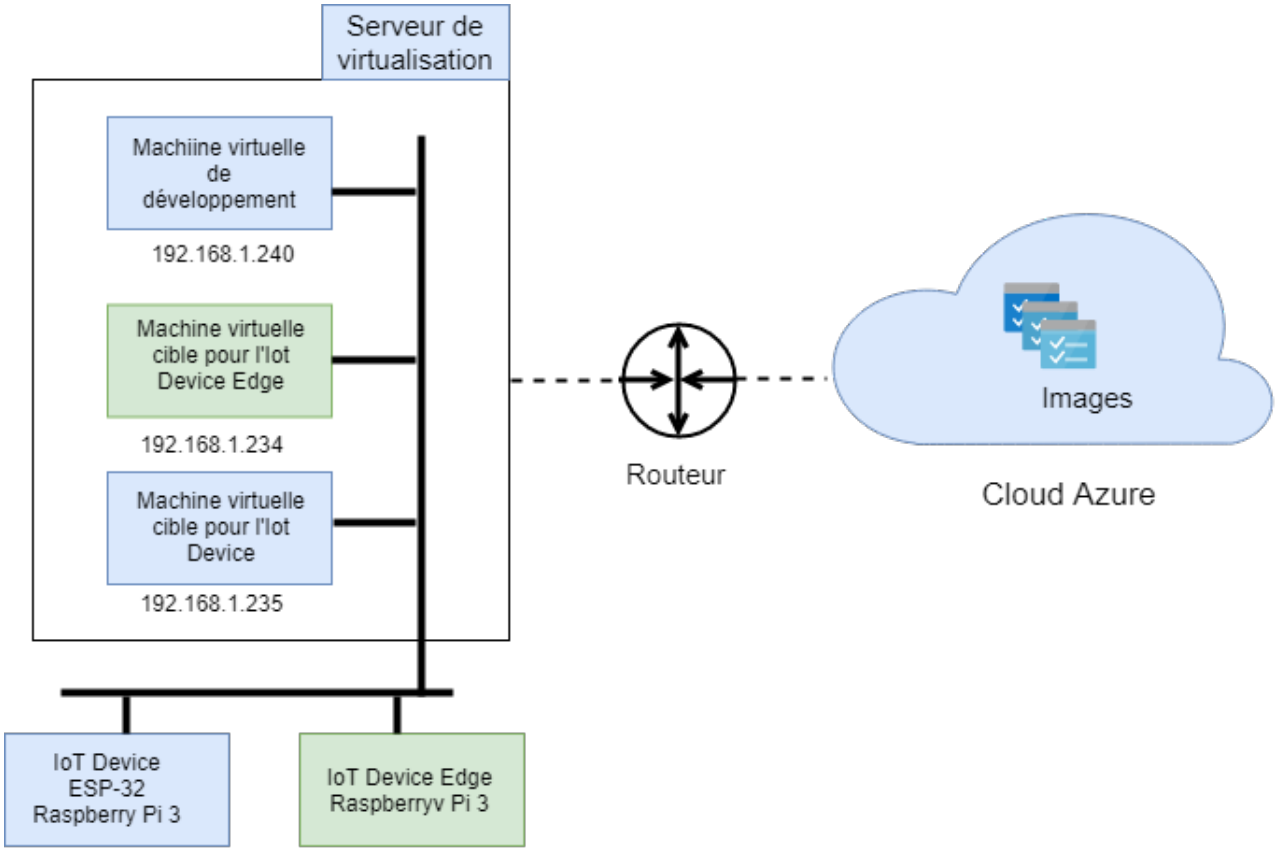

FIGURE 3.6: Infrastructure de développement

3.7.2 Flot de conception

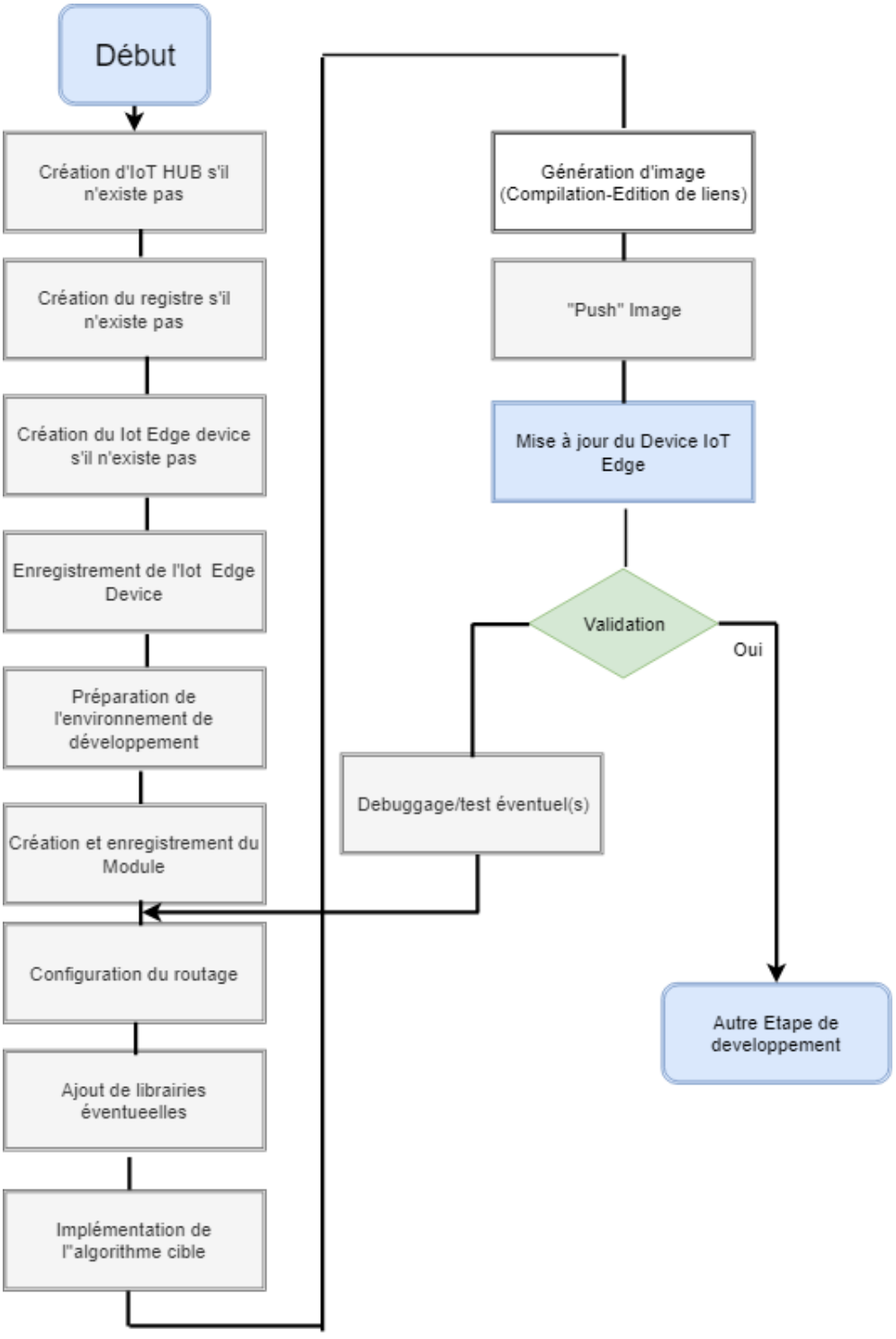

FIGURE 3.7: Flot de conception et de déploiement d'un module (Simplifié)

Pour nous permettre d'avoir une approche de développement optimale (et donc efficace), nous avons opté pour la conception et la réalisation d'un IoT Edge Device

générique. La figure ci après explicite son architecture.

En plus des deux modules de base obligatoires (EdgeAgent et EdgeHub), nous avons pensé à développer :

- un module “Simulated Telemetry” propre. Il pourra être utilisé comme validateur de développement via des capteurs virtuels ou complémentement dédié à l'interfaçage de capteurs réels.
- un module dédié à la transmission D2C
- un module réception D2C. Il pourra être étendu pour implémenter une communication D2D pour des bus industriels ou DoD sachant la pile protocolaire y afférente.
- un module de traitement. Tout traitement peut être envisagé y compris IA.

Cette construction pourra être utilisée pour implémenter n'importe quel scénario possible. Son fonctionnement sera le validateur de notre compréhension aux principes de fonctionnement et de développement de notre solution.

Le module EdgeHub sera configuré pour assurer le routage inter modules y compris les communications en amont (device) qu'en aval (IoTHub) par rapport à notre gateway.

La figure 3.2 ci-dessous illustre l'architecture modulaire que nous avions implémentée.

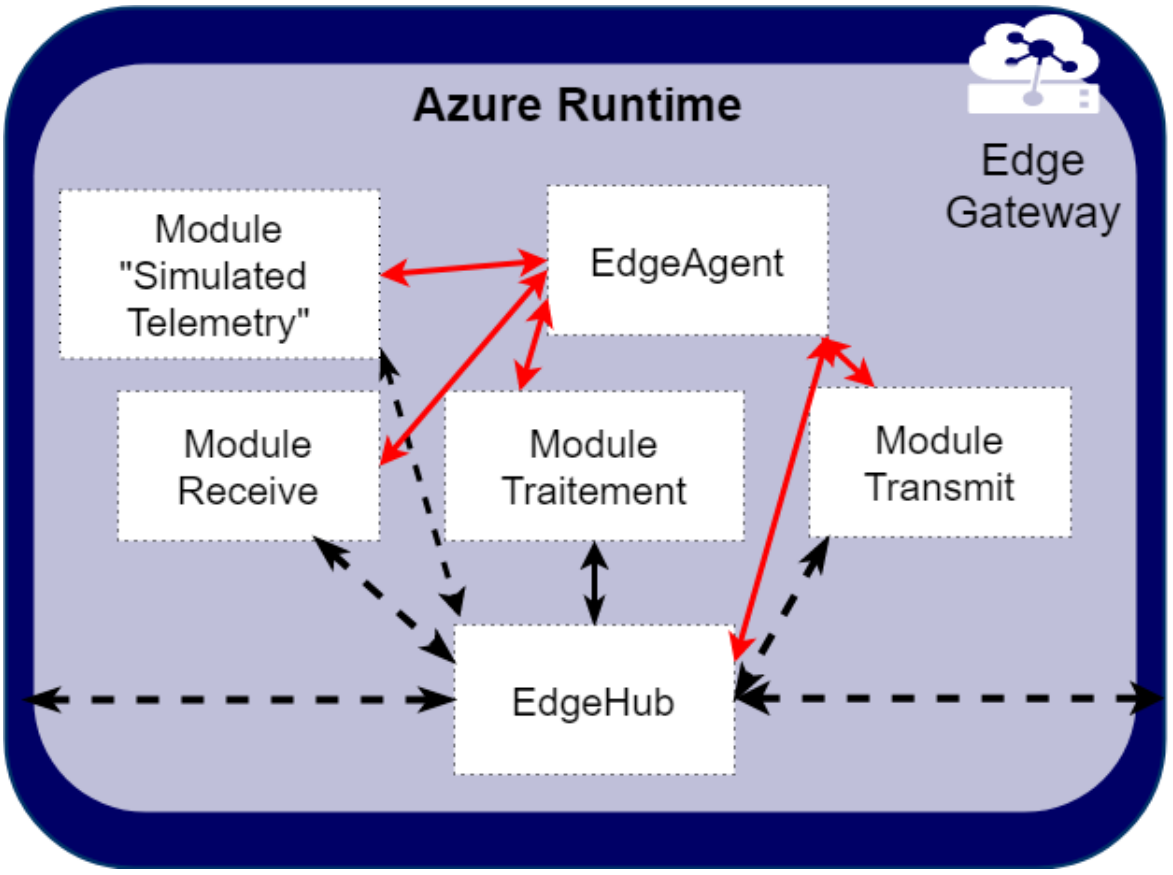

FIGURE 3.8: Modèle de l'IoT Edge Device développé

3.7.2.1 Module SimulatedTelemetry

Dans ce module nous avons développé une application qui génère des données télémétriques de n'importe quel modèle de données capteur(s). Il peut être pensé pour s'interfacer avec de vrais capteurs réels ouvrant la voie à son utilisation comme un IoT Edge device (sans qu'il soit passerelle). Dans notre cas, nous l'avons développé pour générer des données telles que température et humidité en format Json comme illustrée sur la figure ci-dessous. Nous avions donc reproduit le fonctionnement intégrale de "TempSensor", module "fermé" généré par la suite de développement Azure IoT Edge.

3.7.2.2 Module ReceiveModule

Ce module reçoit tous les messages au niveau du Edge, à partir des Devices ou des modules, et les oriente vers le leurs cibles. Il a été aussi réfléchi pour s'interfacer, en fonctionnement de croisière, directement avec tout device indépendamment du Cloud (D2D).

3.7.2.3 Module TraitementModule

Ce module active la fonction du traitement au niveau de la passerelle. Ce traitement pourra être générique (Filtrage, stockage, mandatement, ...) ou spécifique (IA par exemple) à une application donnée.

3.7.2.4 Module TransmitModule

Ce module reçoit les messages en provenance de tout module et le retransmet à l'IoT Hub et inversement. C'est donc la première porte d'entrée (ou de sortie) au niveau du réseau local par rapport cloud.

3.7.3 Organigramme de fonctionnement général d'un module

Les modules que nous avions introduits précédemment ont en commun les étapes de fonctionnement suivantes :

- Initialisation du client
- Attente d'un message
- Génération et transmission d'un message
- Traitement(s) éventuel(s)
- Libération du client dans le cas de la fin de la tâche qui lui est dévolue.

La prise en charge des messages en entrée(s) se fait via la gestion des événements. Chaque événement pourra être au préalable rattaché à des fonctions "callback". Tout message reçu ou envoyé sera subordonné à un acquittement.

La réception ou (et) l'émission de message doit être incluse dans chaque module. C'est ses fonctions propres d'entrée et de sortie (sachant que les mécanismes de routage assurés par le module EdgeHub).

Les deux organigrammes suivants illustrent d'une manière séparée le fonctionnement de chaque cas. On s'appuiera sur la bibliothèque "IoTHubClient" pour le développement de toutes ses fonctions.

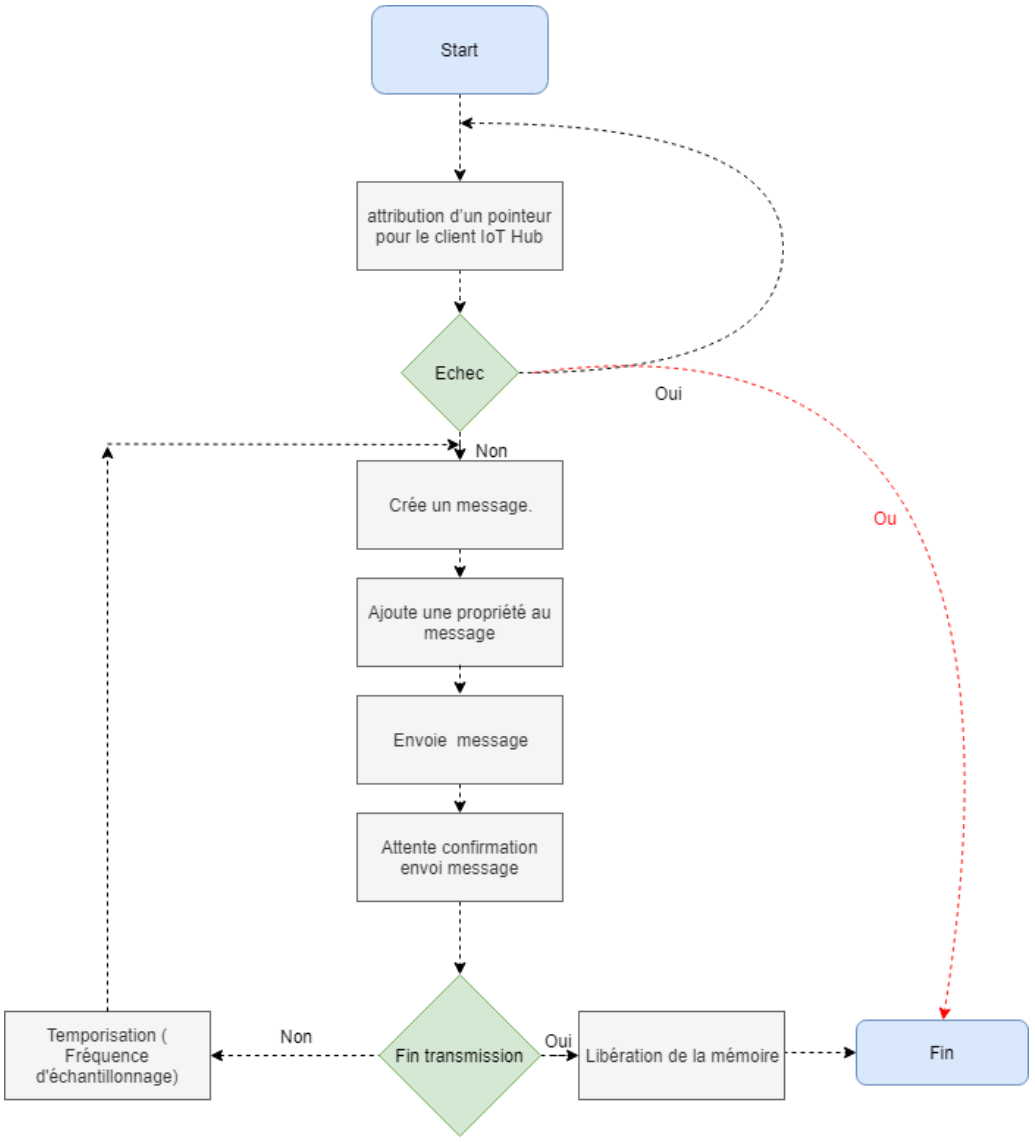

FIGURE 3.9: Organigramme de fonctionnement d'un module en mode transmission de message

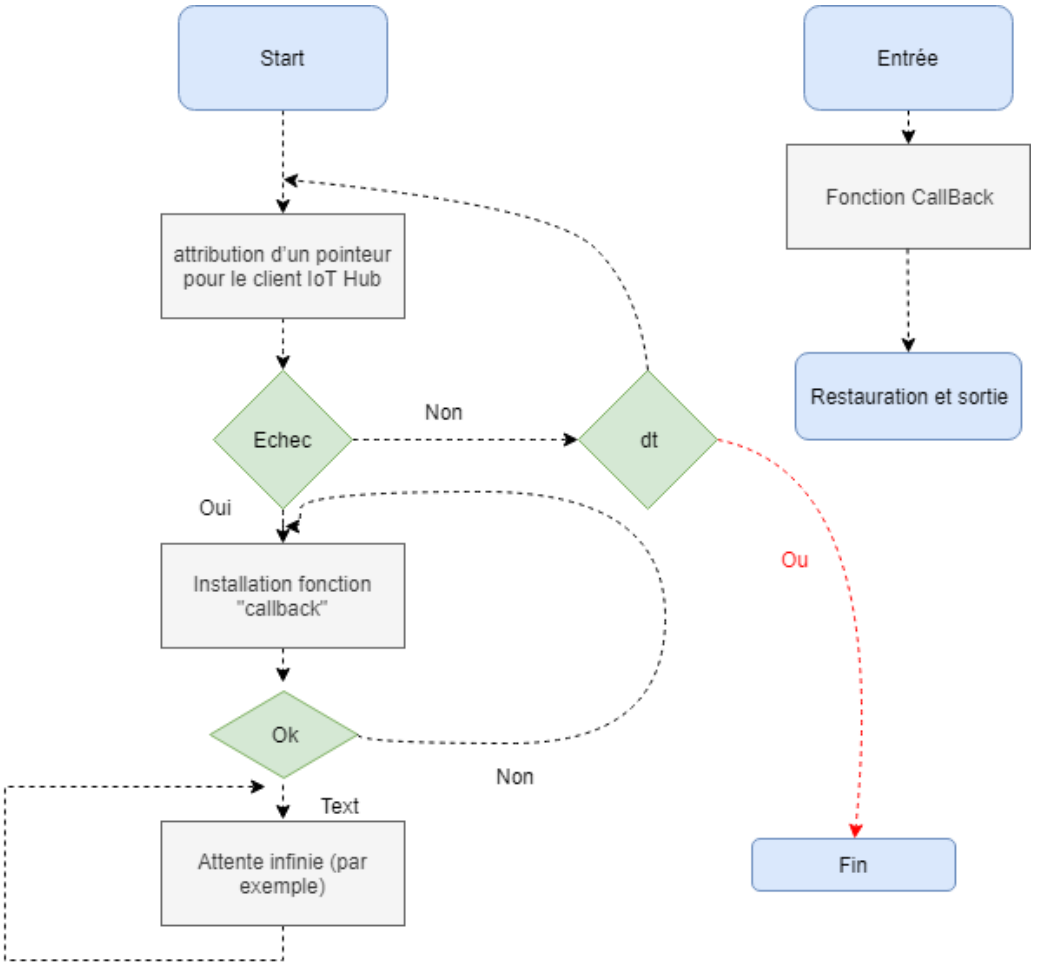

FIGURE 3.10: Organigramme de fonctionnement d'un module en mode réception

La combinaison des organigrammes "émission" et "réception" peut aboutir à des fonctionnements mixtes y compris le traitement à travers les fonctions "callbacks". Chaque module peut être enrichi par des bibliothèques exogènes à l'écosystème microsoft.

3.7.4 Explication du code source de chaque type de module

3.7.4.1 Module Simulated Telemetry

Tout d'abord ,nous commençons par le développement de la fonction qui crée les données JSON. Dans l'exemple de code ci après, on reproduit les données du module générique fermé TempSensor (temperature et humidité). La génération des données JSON s'appuie sur le librairie PARSON. Comme remarqué dans ce code, nous avons

donné des données constantes (34.60,34.61, 34.62,....) pour pouvoir valider le fonctionnement du module.

```
char  *serialization_data(void) {
    JSON_Value *root_value = json_value_init_object();
    JSON_Object *root_object = json_value_get_object(root_value);
    char *serialized_string = NULL;

    json_object_dotset_number(root_object, "machine.temperature", 34.60)
    ;
  json_object_dotset_number(root_object, "machine.pressure", 34.61);
  json_object_dotset_number(root_object, "ambient.temperature", 34.62);
  json_object_dotset_number(root_object, "ambient.humidity", 34.63);
  json_object_dotset_number(root_object, "ambient.timeCreated", 34.64);

  serialized_string = json_serialize_to_string(root_value);

    json_value_free(root_value);

    return serialized_string;

}
```

Après la génération la donnée au format JSON, nous avons développé une fonction qui crée le message à transmettre contenant la donnée brute auquel on a adjoint des propriétés complémentaires telles que le message ID, le COR ID pour l'identification du module et le type de codage de données.

```
IOTHUB_MESSAGE_HANDLE  createMessageTelemetry(unsigned const char*
    myTelemetry) {

            IOTHUB_MESSAGE_HANDLE message_handle;

            // Cr ation de message iothub    partir de la chaine de
    connexion
            message_handle = IoTHubMessage_CreateFromString(myTelemetry)
    ;

```

```
8               // Ajout des propri t s de message
9               (void)IoTHubMessage_SetMessageId(message_handle, "MSG_ID");
10              (void)IoTHubMessage_SetCorrelationId(message_handle, "
    CORE_ID");
11              (void)IoTHubMessage_SetContentTypeSystemProperty(
    message_handle, "application%2Fjson");
12              (void)IoTHubMessage_SetContentEncodingSystemProperty(
    message_handle, "utf-8");
13
14
15              // Destruction de message
16              //IoTHubMessage_Destroy(message_handle);
17
18              return message_handle;
19
20  }
```

Pour envoyer le message nous faisons appel à la fonction InputQueue1Callback qui crée l'instance de message avec les propriétés que nous avons précisé ci-dessus et l'envoie vers la sortie du Module .Cette sortie peut être redirigée vers le Cloud ou vers tout autre module. On peut avoir plusieurs queue de sortie donc plusieurs fonctions callback associées.

```
1  static IOTHUBMESSAGE_DISPOSITION_RESULT InputQueue1Callback(
       IOTHUB_MESSAGE_HANDLE message, void* userContextCallback,
       IOTHUB_MODULE_CLIENT_LL_HANDLE iotHubModuleClientHandle)
2  {
3       unsigned const char* messageBody;
4       IOTHUB_CLIENT_RESULT clientResult;
5       IOTHUBMESSAGE_DISPOSITION_RESULT result;
6
7               messageBody=serialization_data();
8               message =  createMessageTelemetry(messageBody);
9               MESSAGE_INSTANCE *messageInstance = CreateMessageInstance(
    message);
10          if (NULL == messageInstance)
11          {
```

```
12              result = IOTHUBMESSAGE_ABANDONED;
13          }
14          else
15          {
16              printf("Sending message (%zu) to the next Module\n",
    messagesReceivedByInput1Queue);
17                printf("Data: \r\n %s \r\n", messageBody);
18
19              clientResult = IoTHubModuleClient_LL_SendEventToOutputAsync(
    iotHubModuleClientHandle, messageInstance->messageHandle, "output1",
    SendConfirmationCallback, (void *)messageInstance);
20              if (clientResult != IOTHUB_CLIENT_OK)
21              {
22                  IoTHubMessage_Destroy(messageInstance->messageHandle);
23                  free(messageInstance);
24                  printf("IoTHubModuleClient_LL_SendEventToOutputAsync
    failed on sending msg#=%zu, err=%d\n", messagesReceivedByInput1Queue,
     clientResult);
25                  result = IOTHUBMESSAGE_ABANDONED;
26              }
27              else
28              {
29                  result = IOTHUBMESSAGE_ACCEPTED;
30              }
31          }
32              IoTHubModuleClient_LL_DoWork(iotHubModuleClientHandle);
33      messagesReceivedByInput1Queue++;
34      return result;
35  }
```

A chaque fois qu'un message est envoyé une fonction de rappel est appelée pour la confirmation de l'envoie du message.

```
1  static void SendConfirmationCallback(IOTHUB_CLIENT_CONFIRMATION_RESULT
       result, void* userContextCallback)
2  {
3
```

```
4     MESSAGE_INSTANCE* messageInstance = (MESSAGE_INSTANCE*)
      userContextCallback;
5
6     printf("Confirmation[%zu] received for message with result = %d\r\n"
      , messageInstance->messageTrackingId, result);
7     IoTHubMessage_Destroy(messageInstance->messageHandle);
8     free(messageInstance);
9 }
```

3.7.4.2 Module ReceiveModule

Ce module effectue une opération de filtrage en fonction de la source de message.En effet, ReceivModule reçoit les messages de toutes les destinations (Devices et Modules) vers la Gateway et sélectionne uniquement ceux venant des device et les dirige vers un autre module (TraitmentModule) .Ce type de filtrage est effectué dans la partie routage des message que nous pouvons modifier dans le fichier déploiement-template .

Après que la connexion du module est initialisée ,une boucle infinie est lancée en attendant la réception des messages :

```
1 void iothub_module()
2 {
3     IOTHUB_MODULE_CLIENT_LL_HANDLE iotHubModuleClientHandle;
4
5     srand((unsigned int)time(NULL));
6
7     if ((iotHubModuleClientHandle = InitializeConnection()) != NULL)
8     {
9   IOTHUB_MESSAGE_HANDLE message;
10    void* userContextCallback;
11          while (true)
12
13          {
14              InputQueue1Callback(message,userContextCallback,
      iotHubModuleClientHandle);
15              ThreadAPI_Sleep(500);
```

```
        }
    }

    DeInitializeConnection(iotHubModuleClientHandle);
}

int main(void)
{
    iothub_module();
    return 0;
}
```

Dès qu'un message arrive ,une autre fonction de rappel (*SetupCallbacksForModule*) est invoquée ,et qui à son tour appelle la fonction (*InputQueue1Callback*) pour l'envoie de message suivant le routage impose dans le déploiement template.

CODE

3.7.4.3 Module TraitementModule

D ans cette partie nous avons créé une fonction qui après la réception des messages en format json ,fait appel la librairie PARSON qui nous permet l'extraction des données formatées en json. A titre illustratif et dans un contexte de validation du module seulement, cette fonction incrémente la valeur d'une variable choisie arbitrairement .

```
printf("Received Message [%zu]\r\n Data: [%s]\r\n",
            messagesReceivedByInput1Queue, messageBody);


    JSON_Value *root_value = json_parse_string(messageBody);
    JSON_Object *root_object = json_value_get_object(root_value);
    double temperature;
    if (json_object_dotget_value(root_object, "machine.temperature") !=
   NULL )
    {
        temperature = json_object_dotget_number(root_object, "machine.
   temperature") +1 ;
```

```
11      //json_object_dotset_number(root_object, "machine.temperature",
    temperature );
12
13      printf("la valeur de temperature est augmentee a : %f \r\n",
    temperature);
14      MESSAGE_INSTANCE *messageInstance = CreateMessageInstance(
    message);
15      if (NULL == messageInstance)
16      {
17          result = IOTHUBMESSAGE_ABANDONED;
18      }
19      else
20      {
21          printf("Sending message (%zu) to the Cloud \n",
    messagesReceivedByInput1Queue);
22
23          clientResult = IoTHubModuleClient_LL_SendEventToOutputAsync(
    iotHubModuleClientHandle, messageInstance->messageHandle, "output1",
    SendConfirmationCallback, (void *)messageInstance);
24          if (clientResult != IOTHUB_CLIENT_OK)
25          {
26              IoTHubMessage_Destroy(messageInstance->messageHandle);
27              free(messageInstance);
28              printf("IoTHubModuleClient_LL_SendEventToOutputAsync
    failed on sending msg#=%zu, err=%d\n", messagesReceivedByInput1Queue,
     clientResult);
29              result = IOTHUBMESSAGE_ABANDONED;
30          }
31          else
32          {
33              result = IOTHUBMESSAGE_ACCEPTED;
34          }
35      }
36  }
37   else
38  {
39      printf("Not sending message (%zu) to the next stage in pipeline
    .\r\n", messagesReceivedByInput1Queue);
```

```
40          result = IOTHUBMESSAGE_ACCEPTED;
41      }
```

3.7.4.4 Module TransmitModule

Ce module envoie la valeur d'une variable modifiée issue du Traitement Module et l'envois vers le Cloud .

```
1  printf("Received Message Data:: [%s]\r\n",messageBody);
2
3      MESSAGE_INSTANCE *messageInstance = CreateMessageInstance(message);
4      if (NULL == messageInstance)
5      {
6          result = IOTHUBMESSAGE_ABANDONED;
7      }
8      else
9      {
10         printf("Sending message to the Cloud \n");
11
12         clientResult = IoTHubModuleClient_LL_SendEventToOutputAsync(
       iotHubModuleClientHandle, messageInstance->messageHandle, "output1",
       SendConfirmationCallback, (void *)messageInstance);
13         if (clientResult != IOTHUB_CLIENT_OK)
14         {
15             IoTHubMessage_Destroy(messageInstance->messageHandle);
16             free(messageInstance);
17             printf("IoTHubModuleClient_LL_SendEventToOutputAsync failed
       on sending msg#=%zu, err=%d\n", messagesReceivedByInput1Queue,
       clientResult);
18             result = IOTHUBMESSAGE_ABANDONED;
19         }
20         else
21         {
22             result = IOTHUBMESSAGE_ACCEPTED;
23         }
24     }
```

3.7.4.5 Routage des Messages

L e routage des message au niveau de la Gateway est géré par le module edgeHub suivant la configuration du fichier deployment-template de l'application modulaire.

Dans le routage des messages nous devons spécifier la source et la destination ,la source de message peut être à partir d'un device IoT ou d'un module interne de la Gateway ,la destination quand a elle peut etre soi vers le Cloud ou bien vers un autre module .

nous utilisons pour la destination Upstream ,si nous souhaitons envoyer le message vers le Cloud ,ou bien BrokerEndpoint pour l'envoyer vers un autre device.

Aussi nous pouvons ajouter une condition sur les messages pour le filtrage des messages .

Pour répondre au exigences de validation de notre approche de développement, nous avons établie les routes suivante :

FIGURE 3.11: Le routage des messages

3.7.5 Construction des images et leurs publications sur le Cloud

Après avoir développé l'application associée à nos modules, nous faisons recours à gestionnaire local des conteneurs (Docker) pour créer les images de cette application et les stocker au niveau d'un registre de conteneur Cloud.

3.8 Développement de l'application Device

3.8.1 Flot de conception

Avant de commencer le développement de l'application Cliente au niveau du périphérique IoT nous devons valide les opérations préalables cité précédemment.

Par la suite, nous procédons à l'installation des Kit de développement logiciel (SDK) d'appareil Azure IoT (Azure IoT Device SDK). C'est un ensemble de bibliothèques conçu pour simplifier le développement des applications tel que le processus d'envoi et de réception de messages. Plusieurs variantes de ce kit de développement logiciel sont disponibles, or, comme cité ci-dessus nous avons utilisée Azure IoT Device SDK en langages C/C++.Ce dernier est rédigé de façon à optimiser sa portabilité ce qui rend ses bibliothèques adaptées pour fonctionner sur plusieurs plates-formes (Windows / Linux) et appareils.

Azure IoT SDK sont téléchargeables dans le référentiel GitHub.

Pour simuler un écosystème de communication D2D divers, nous avons développé deux application séparées, la première dédiée a l'envoie des donnée télémétriques de la carte électronique Rapberry PI vers la Gateway, la deuxième au niveau de la carte ESP32 qui génère des données de température est d'humidité et les envoie.

L'implémentation de ces deux applications a été réalisé a base des fonctions offertes par les différentes librairies du répertoire « iothubclient », ce répertoire comprend aussi des d'applications types montrant comment utiliser les fonctionnalités du service Microsoft Azure IoT Hub à partir d'un périphérique exécutant du code C permettant ainsi à l'appareil d'interagir avec IoT Hub et la Gateway Edge.

Afin de pouvoir compiler ces applications, IoTHubClient contient l'implémentation de la couche d'API la plus basse de SDK : la bibliothèque IoTHubClient. Cette bibliothèque contient des APIs implémentant des fonctions brutes à exploiter au niveau du Device .

IoTHubClient dépend d'autres bibliothèques Open Source qui implémentent le transport à l'aide de protocoles tels que MQTT et AMQP :

- La bibliothèque Azure uAMQP, qui est une implémentation côté client du protocole AMQP optimisée pour les appareils avec contraintes de ressources.
- La bibliothèque Azure uMQTT , qui est une bibliothèque à usage général implémentant le protocole MQTT et optimisée pour les appareils avec des contraintes de ressources également.

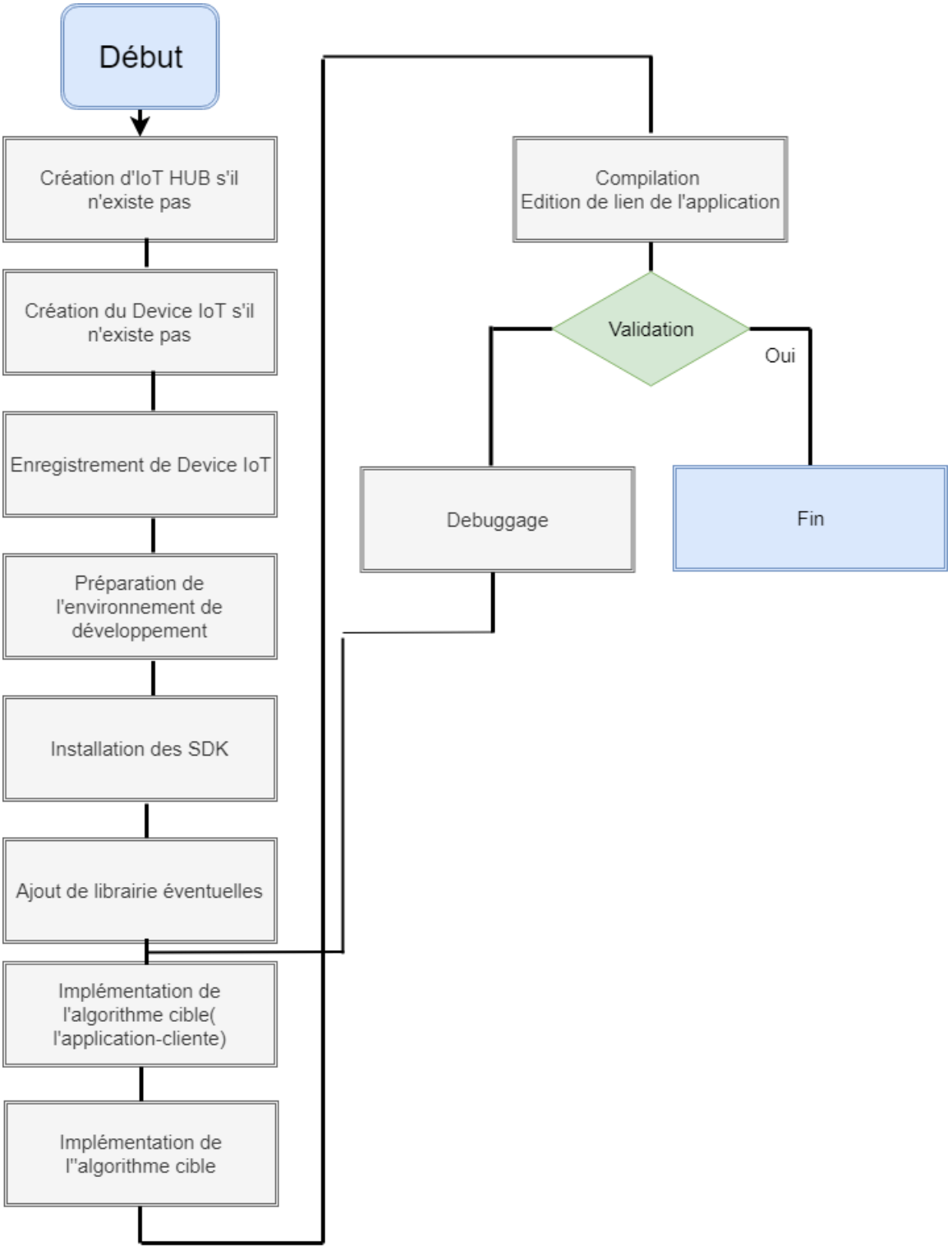

FIGURE 3.12: Flot de conception

3.8.2 Architecture de l'application

Afin d'établir une communication D2D réussie ,les applications implémentées assure les fonctionnements suivants :

— Initialisation du ClientIoTHub

- Génération de message de télémétrie
- Validation du certificat d'authentification
- Transmission de message.

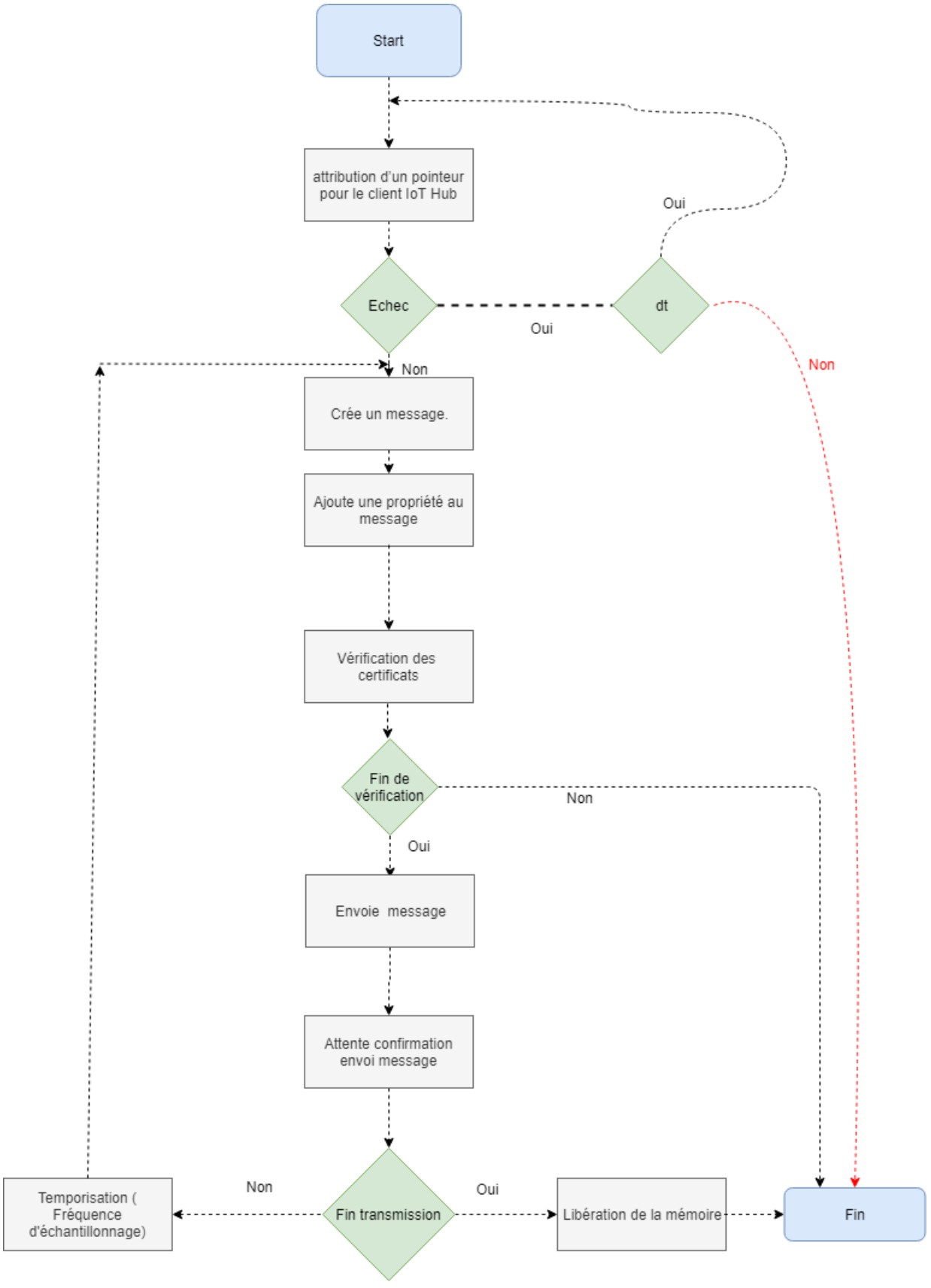

FIGURE 3.13: Fonctionnement de la communication D2D

3.8.3 Explication du code source de chaque type de module

Les codes sources des applications inclut les packages de la bibliothèque IoTHub-Client suivants :

- Microsoft.Azure.C.SharedUtility
- Microsoft.Azure.IoTHub.IoTHubClient

Dans notre implémentation nous avons utilisé le protocole de communication MQTT. Par conséquent nous avons inclut les librairies Microsoft.Azure.umqtt et Microsoft.Azure.IoTHub.MqttTransport dans le code de l'application (il existe des packages équivalents pour les protocoles de communication AMQP et HTTPS) .

Nous commençons notre code par l'initialisation de la chaîne de connexion tel défini sur le portail Azure.Comme la communication est effectuée à travers le Gateway ,nous ajoutant son identifiant IP en bout de la chaîne,

```
static const char* connectionString = "HostName=HubRsaDz.azure-devices.
    net;DeviceId=IoTDevice-235;SharedAccessKey=
    lGK5XvzAsk7Y4WqfZrw7imHeeygFoOaWme/Bp7U19Do=;GatewayyHostName
    =172.28.254.234";
```

Pour authentifie le device et lui attribuer le privilège de communication ,la vérification des certificats est primordiale,ces dernier sont installés également au niveau du Client,et sont appelés à chaque établissement de communication ,

```
static const char* edge_ca_cert_path = "/home/user/CERTDIR/certs/azure-
    iot-test-only.root.ca.cert.pem ";
```

Le code de vérification des certificat :

```
/**
  Lire le fichier de certificat et fournir une cha ne termin e par
    null
  contenant le certificat.
*/
static char *obtain_edge_ca_certificate(void)
{
```

```
7   char *result = NULL;
8   FILE *ca_file;
9
10  ca_file = fopen(edge_ca_cert_path, "r");
11  if (ca_file == NULL)
12  {
13      printf("Error could not open file for reading %s\r\n",
    edge_ca_cert_path);
14
15  else
16
17      size_t file_size;
18
19      (void)fseek(ca_file, 0, SEEK_END);
20      file_size = ftell(ca_file);
21      (void)fseek(ca_file, 0, SEEK_SET);
22      // increment size to hold the null term
23      file_size += 1;
24
25      if (file_size == 0) // check wrap around
26      {
27          printf("File size invalid for %s\r\n", edge_ca_cert_path);
28
29      else
30
31          result = (char*)calloc(file_size, 1);
32          if (result == NULL)
33          {
34              printf("Could not allocate memory to hold the certificate
    \r\n");
35
36          else
37
38              // copy the file into the buffer
39              size_t read_size = fread(result, 1, file_size - 1,
    ca_file);
40              if (read_size != file_size - 1)
41              {
```

```
42                      printf("Error reading file %s\r\n", edge_ca_cert_path
    );
43                      free(result);
44                      result = NULL;
45
46
47
48          (void)fclose(ca_file);
49
50
51      return result;
52  }
```

Choix de protocole de communication :

```
1  // Choix du protocole de communication pour la connexion
2  #ifdef SAMPLE_MQTT
3      protocol = MQTT_Protocol;
4  #endif // SAMPLE_MQTT
```

Après que toutes les configurations en rapport avec la connexion on été definies dans le code source, la communication est initialisée par la génération d'un pointeur sur l'objetIoThubClient .Dès lors ,la fonction de rappel connectionstatuscallback est appelée pour confirmer que le device communique bel est bien avec la gateway.

```
1  static void connection_status_callback(IOTHUB_CLIENT_CONNECTION_STATUS
       result, IOTHUB_CLIENT_CONNECTION_STATUS_REASON reason, void*
       user_context)
2  {
3      (void)reason;
4      (void)user_context;
5
6      if (result == IOTHUB_CLIENT_CONNECTION_AUTHENTICATED)
7      {
8          (void)printf("The device client is connected to IoT Edge\r\n");
9
10     else
11
```

```
12        (void)printf("The device client has been disconnected\r\n");
13
14 }
```

A ce niveau, le message est créé et envoyé vers la Gateway :

```
1  for (size_t index = 0; index < MESSAGE_COUNT; index++)
2        {
3            // Creation de pointeur IoThub a partir de la connexion
     string
4            message_handle = IoTHubMessage_CreateFromString(telemetry_msg
     );
5
6            // Ajouter les propri t s du message
7            (void)IoTHubMessage_SetMessageId(message_handle, "MSG_ID");
8            (void)IoTHubMessage_SetCorrelationId(message_handle, "CORE_ID
     ");
9            (void)IoTHubMessage_SetContentTypeSystemProperty(
     message_handle, "application%2fjson");
10           (void)IoTHubMessage_SetContentEncodingSystemProperty(
     message_handle, "utf-8");
11
12           // Propri t s   d  identification   du device
13           (void)IoTHubMessage_SetProperty(message_handle, "property_key
     ", "property_value");
14
15           (void)printf("Sending message %d to Edge Hub\r\n", (int)(
     index + 1));
16           IoTHubDeviceClient_SendEventAsync(device_handle,
     message_handle, send_confirm_callback, NULL);
17
18           // Suppression des ressource une fois la communication
     cl tur e
19           IoTHubMessage_Destroy(message_handle);
```

La fonction de rappel (callback) est appelée à chaque fois qu'un message arrive au niveau de la terminaison cible (gateway ou cloud).

```
static void send_confirm_callback(IOTHUB_CLIENT_CONFIRMATION_RESULT
    result, void* userContextCallback)
{
    (void)userContextCallback;
    // cette fonction est invokee des que le message arrive
    g_message_count_send_confirmations++;
    (void)printf("Confirmation callback received for message %zu with
    result %s\r\n", g_message_count_send_confirmations, MU_ENUM_TO_STRING
    (IOTHUB_CLIENT_CONFIRMATION_RESULT, result));
}
```

3.9 Validation

Les figures ci-dessous, nous montre les résultats que nous avons obtenu lors de la réalisation de notre projet de fin d'étude :

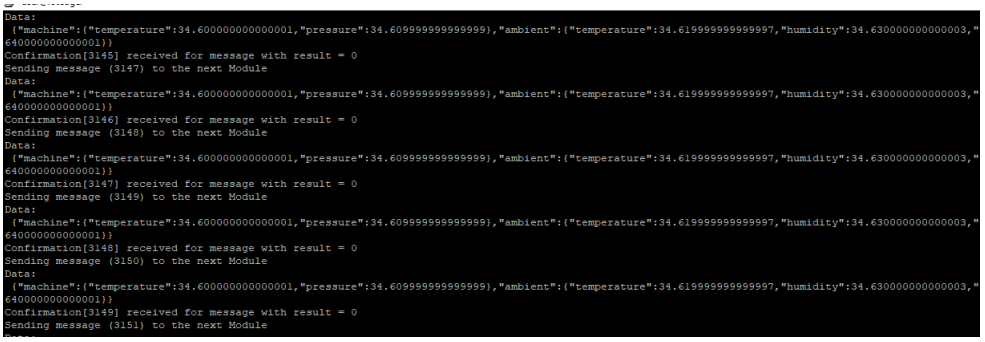

FIGURE 3.14: Résultat pour Receive Module

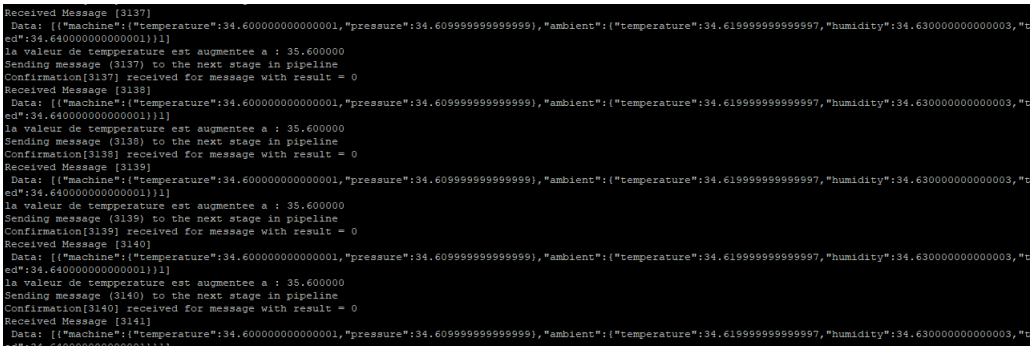

FIGURE 3.15: Résultat pour le module de traitement

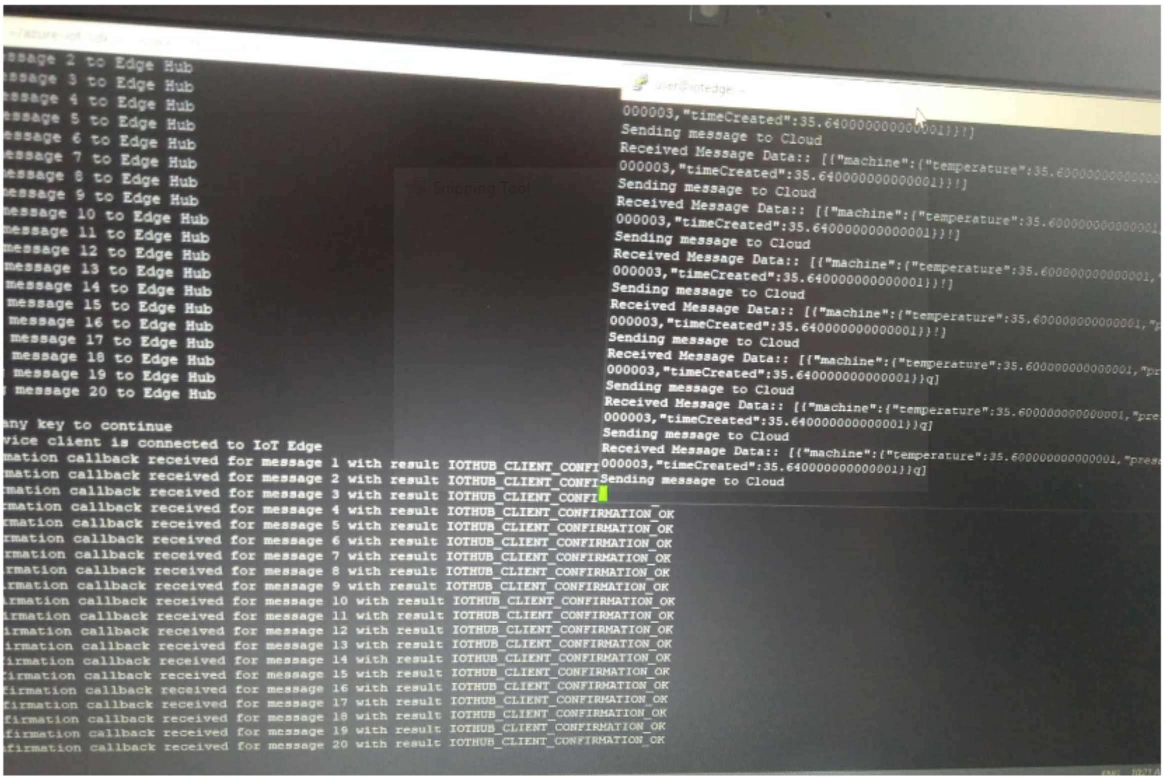

FIGURE 3.16: Confirmation que le message envoye du Client est bien recue par la Gateway

Conclusion générale et perspectives

Notre projet de fin d'étude s'est porté sur la mise en place d'une gateway dans une maison intelligente. L'intérêt d'une telle passerelle réside dans le fait qu'elle est de type Edge,où nous déporterons certains fonctionnalités du Cloud.

Dans notre cas, nous avons choisit la plateforme Microsoft Azure qui intègre le Edge computing dans le service Azure IoT. Nous avons bien constaté les avantages et les bénéfices de Edge computing en terme d'optimisation du temps de réponse, traitement et la réduction des coût.

En ce qui concerne l'implémentation du Edge, nous avons développé des applications qui interagissent avec les périphériques. L'utilisation des Kits de développement software Azure nous a permet de développer plusieurs modules personnalisé afin de répondre aux scénarios du cahier de charges.

De plus, le développement au niveau de la plateforme azure nous a offert le choix entre plusieurs langage de programmation et de système d'exploitation.

A travers ce projet de fin d'études, nous a eu la chance de toucher à des technologies de futur qui y en plein de dévellopement.

Finalement, pour implémenter une passerelle Edge, il faudra établir un cahier de charge bien défini en ce qui concerne le besoin ou les fonctionnalités qu'on doivent avoir sur cette passerelle.

En tant qu'electroniciennes, nous souhaitons en perspectives pousser nos recherches non seulement sur l'architecture software de la gateway pour intégrer plus d'intelligence à l'environnement domestique, mais aussi l'architecture hardware (physique) de la gateway à fin d'étudier ses caractéristiques.

Bibliographie

[1] Dwight Spivey. *home automation for dummies*. 2015.

[2] Enrique Willy. domotique,maison connectée,maison intelligente.

[3] Harvé. The history of smart homes.

[4] Ms.Tel.Benabdallah. *Maison intelligente*. 2015.

[5] Esra BAYIR Tolga KILIÇ. An investigation on internet of things technology (iot) in smart houses. *ResearchGate*, pages 197–204, 2017.

[6] Jingdong Bian Jian Mao, Qixiao Lin. Application of learning algorithms in smart home iot system security. *AIMS*, 2018.

[7] Benjamin Jérôme. Une maison connectée à tous vos désirs.

[8] Futura Michel Berkowicz. Les services de la domotique.

[9] Moeen Azhar Sumaira Kausar Summia Taj, Uniza Asad. Interoperability in iot based smart home : A review. *ResearchGate*, pages 50–52, 2018.

[10] Esther Ososanya et Sasan Haghani Abdelhakim Ahmim, Tam Le. Design and implementation of a home automation system for smart grid applications. *IEEE*, pages 1–2, 2016.

[11] Mrs. Snehal S. Golait Mr. Pranay P. Gaikwad, Mrs. Jyotsna P. Gabhane. A survey based on smart homes system using internet-of-things. *IEEE*, 6, 2015.

[12] *Developing for the intelligent cloud and intelligent edge at Microsoft Connect*. 2018.

[13] L.Bastien. Tout savoir sur la plateforme cloud public, 2018.

[14] Robin Shahan Michael Collier. *Microsoft Azure Essentials*. Microsoft Press, 2016.

[15] David Jensen. Beginnig azure iot edge computing. 2019.