الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

Filière Télécommunication
Spécialité Réseaux & Télécoms

présenté par

Daouadji Manel

&

El aichi Asma

Plateforme IOT générique adaptable

Proposé par : Mr Kabir Yacine

Année Universitaire 2019-2020

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force, la patience et la santé malgré tous les évènements du covid-19, pour accomplir ce Modeste travail.

Nous voudrions remercier, notre promoteur de mémoire M. KABIR YACINE pour ça patience, et ces judicieux conseils, qui ont contribué à alimenter notre réflexion.

Nous adressons nos sincères remerciements à tous les professeurs intervenants, et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidés nos réflexions.

Nous remercions nos très chères parentes, qui ont toujours été là pour nous, nous remercions nos frères et nos sœurs, pour leurs encouragements.

Enfin nous remercions nos amies et nos cousines et cousins, leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

A toutes ces intervenants, nous présentons nos remerciements, nos respects et notre gratitude.

Manel & Asma

Nous dédions ce travail qui n'aura jamais pu voir le jour sans les soutiens indéfectibles de nos chers parents qui ne cessent de nous donner l'amour nécessaire pour que nous puissions arriver à ce que nous somme aujourd'hui. Que dieux vous protège et que la réussite soit toujours à notre portée pour que nous puissions vous combler de bonheur.

Manel & Asma

ملخص: تتيح تقنية إنترنت الأشياء الجديدة إمكانية توصيل مليارات الاجهزة عن بُعد. هذه التكنولوجيا هي مزيج من هندسة شبكات إنترنت الأشياء والتطبيقات والأجهزة والبرامج. لإعداد بنية إنترنت الأشياء، من الضروري مراعاة جوانب الأجهزة (البطاقات الإلكترونية) والبرامج (النظام الأساسي، البروتوكول). الهدف من هذا المشروع هو إنشاء منصة عامة للتحكم عن بعد لإنترنت الأشياء. نتمنى، مع النتائج الواعدة التي تم الحصول عليها، أن نساهم في تحفيز الروح المعنوية لإعطاء المزيد من الاهتمام لهذا المجال في الجزائر.

الكلمات المفتاحية: إنترنت الأشياء، شبكات المعلومات المستعرضة.

Résumé : La nouvelle technologie de l’internet des objets permet de connecter des milliards des objets à distance. Cette technologie est une combinaison de l’architecture de réseau IdO, d’applications, des matériels et des logiciels. Pour mettre en place une architecture IoT, il faut considérer les deux aspects matériel (cartes électroniques) et logiciel (plateforme, protocole). Le but de ce projet consiste à créer une plateforme générique de contrôle à distance de l'IdO. Nous espérons avec les résultats prometteurs obtenus, contribuer à stimuler les esprits pour donner plus d’intérêts à ce domaine en Algérie.

Mots clés : Internet des objets, réseaux d'information transversaux.

Abstract: New Internet of Things technology enables billions of things to be connected remotely. This technology is a combination of IoT network architecture, applications, hardware, and software. To set up an IoT architecture, you have to consider both hardware (electronic cards) and software (platform, protocol) aspects. The goal of this project is to create a generic IoT remote control platform. We hope, with the promising results obtained, to help stimulate the spirits to give more interest to this field in Algeria.

Keywords : Internet of things, transversal information networks.

Listes des acronymes et abréviations

Act/Cap	Actionneur/Capteur
API	Application Programming Interface
ARM	Acorn Risc Machine
ASGI	Asynchronous Server Gateway Interface
BDD	Base de Données
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
CSI	Camera Serial Interface
CSS	Cascading Style Sheets
DAL	Data Access Layer
DSI	Display Serial Interface
DVI	Digital Visual Interface
EDGE	Enhanced Data Rates for GSM Evolution
GPO	General Purpose Outlet
GPRS	General Packet Radio Service
HDMI	High-Definition Multimedia Interface
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifiant
IDE	Integrated Development Environment

IdO Internet des Objets

IEEE Institute of Electrical and Electronics Engineering

IoT Internet of Things

IP Internet Protocol

IPSO Internet Protocole for Smart Objects

ITU International Telecommunication Union

I2C Inter-Integrated Circuit

JTAG Joint Test Action Group

LAN Local Area Network

LED Light Emitting Diode

LoRa Long Range

LTE Long Term Evolution

MCU Micro-Controller Unit

MQTT Message Queuing Telemetry Transport

MVC Modèle-Vue-Contrôleur

MVT Modèle-Vue-Template

M2M Machine to Machine

NFC Near Field Communication

NFS Network File System

PHP Hypertext Preprocessor

PMOD Priority Mail Open and Distribute

RAM Random Access Memory

REST Representational State Transfer

RF Radio Fréquence

RFID Radio Frequency Identification

SD Secure Digital

SDHC Secure Digital High Capacity

SGBD Systèmes de Gestion de Base de Données

SPI Serial Peripheral Interface

SQL Sigle de Structured Query Language

SRAM Static Random Access Memory

Sub/Pub Subscriber/Publisher

TCP Transmission Control Protocol

UART Universal Asynchronous Receiver-Transmitter

UDP User Datagram Protocol

UIT Union International des Télécommunication

URI Uniform Resource Identifier

URL Uniform Resource Locator

USB Universal Serial Bus

UUID Universally Unique Identifier

VoIP Voice Over Internet Protocol

WAMP Windows, Apache, MySQL, and PHP

WAN Wide Area Network

WIFI Wireless Fidelity

WLAN Wireless Local Area Network

WSN Wireless Sensors Network

WSGI Web Server Gateway Interface

XML Extensible Markup Language

XMPP Extensible Messaging and Presence Protocol

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

En raison de la relation importante entre les humains et les machines en informatique, de nombreuses technologies deviennent de plus en plus capables de relier entre l'environnement, les machines et les personnes.

L'utilisation des technologies sans fil telles que l'Internet des objets (IoT) a un impact énorme sur le monde, non seulement connectant des milliards de personnes mais aussi des milliards d'objets.

L'Internet des Objets est le réseau d'objets physiques embarqués avec l'électronique, les logiciels, les capteurs et la connectivité réseau. L'Iot permet de détecter et de contrôler des objets à distance sur l'infrastructure réseau existante, créant des opportunités d'intégration plus directe entre le monde physique et les systèmes informatiques.

Grace à ses potentialités et son aspect ubiquitaire, la montée en puissance de l'IoT peut s'observer dans plusieurs domaines des plus personnelles aux plus industrielles. Ceci a conduit à des avancées énormes, meilleur gestion d'énergie, traçabilité des produits, amélioration du suivi de la santé, simplification des tâches quotidiennes, etc.

L'Internet des Objets aspire à répondre aux besoins de l'utilisateur qui recherche le bon service au bon moment et au bon endroit, et c'est en développant de nouvelles applications qu'il sera possible d'exploiter cette technologie pour le bien-être de l'homme aussi bien sur le plan individuel (confort et santé) et sur le plan de la société (gestion des villes, industrie, santé, ...).

Dans ce travail, nous allons nous intéresser à cette nouvelle tendance des objets connectés dans le but d'améliorer leur utilisation. Pour cela nous allons créer une plateforme générique pour la gestion des réseaux IoT.

Le but est d'aider les utilisateurs (personnes, entreprises, industries) à mieux gérer à distance leurs objets connectés.

Ce mémoire est organisé en trois chapitres, le premier chapitre sera consacré à la présentation de l'Internet des Objets, ainsi nous allons expliquer la technologie IoT, le principe de son utilisation, nous allons également exposer quelques applications existantes et d'autre envisageables pour l'IoT, tout en évoquant les avantages et les inconvénients, ainsi que le matériel utilisé dans le monde de l'IoT.

Dans le chapitre 2, nous présentons la conception détaillée de notre système, nous expliquons l'implémentation de la plateforme générique crée pour l'Internet des objets tout en exposant différentes applications possible de notre système.

Dans le dernier chapitre, nous allons exposer les différentes applications de notre plateforme et présenter les résultats des expérimentations de la connectivité entre les objets connectés (Act/Cap) et notre plateforme.

Chapitre 1 Généralités sur l'Internet des Objets

L'Internet des objets (IoT) est le réseau d'objets physiques, d'appareils, de véhicules, de bâtiments et d'autres éléments intégrés à l'électronique, aux logiciels, aux capteurs et à la connectivité réseau. L'Internet des objets permet de détecter et de contrôler les objets à distance sur l'infrastructure réseau existante, créant des opportunités d'intégration plus directe entre le monde physique et les systèmes informatiques.

Dans ce chapitre nous allons donner des généralités sur l'internet des objets, le début, le développement, l'architecture de l'IoT, les domaines d'applications et les défis de l'IoT, nous donnerons aussi un aperçu sur la sécurité dans l'Internet des objets, et nous terminons avec une brève description du matériel IoT.

1.1 Historique sur l'Internet des Objets

L'IoT est un système utilisant diverses technologies, allant de l'Internet à la communication sans fil des systèmes micro-électromécaniques, jusqu'aux systèmes embarqués.

Le principe de contrôle des objets tel que les équipements électriques et électroniques à distance via internet existe depuis le début des années 1990, lorsque John Romkey a créé le premier appareil avec Internet : un grille-pain qu'il pouvait allumer et éteindre via Internet.

Le terme « internet of things » a été proclamé pour la première fois par le Britannique Kevin Ashton employé de Procter & Gamble (P&G), dans une présentation qu'il a faite en 1999, reliant la nouvelle idée de la RFID (Radio Frequency Identification), grâce à cette présentation, se diffuse l'idée de machines capables de communiquer entre elles.

Le groupe RFID définit l'IoT comme « le réseau mondial d'objets interconnectés d'adresse unique, basé sur des protocoles de communication standard. [1]

Le premier objet connecté est commercialisé en 2003 par la firme Violet. C'est la lampe DAL, équipée de 9 LEDs qui s'allument en fonction des évènements. Deux ans plus tard, en 2007 c'est l'apparition des smartphones. Enfin, en 2008 c'était la création des adresses IPSO, adresses IP des objets connectés qui leur permettent d'interagir entre eux. [2]

1.2 Définition de l'Internet des Objets

Selon l'union internationale des télécommunications (ITU), l'IoT est « une infrastructure mondiale au service de la société de l'information ». De son côté, l'IEEE définit l'IoT comme un « réseau d'éléments chacun muni de capteurs qui sont connectés à Internet ».

L'Internet des objets caractérise des objets physiques connectés ayant leur propre identité numérique et capables de communiquer les uns avec les autres. Ce réseau crée en quelque sorte une passerelle entre le monde physique et le monde virtuel. [3]

L'IoT peut être réalisé selon trois paradigmes : orienté vers internet (middleware), orienté vers les objets (capteurs) et orienté vers la sémantique (apprentissage). [2]

1.2.1 Objets connectés

Les « objets » sont des participants actifs dans un système IOT, ils peuvent interagir et communiquer entre eux et avec leurs environnements, échanger des données et des informations détectées ou collectées à partir de leurs environnements, tout en réagissant de manière autonome aux événements réels/physiques.

Un objet connecté est la plupart du temps simplement composé ;

- D'un capteur.
- D'une petite mémoire permettant de stocker des données.
- D'un processeur permettant de comparer des valeurs.

- D’une antenne de communication (Bluetooth, wifi, ou autre) lui permettant ainsi d’envoyer les informations du capteur pour visualisation et de recevoir les informations de l’utilisateur. [4]

1.2.2 Avantage des Objets connectés

- Possibilité de stocker et de traiter de nombreuses informations.
- Taille réduite de l’équipement nécessitant moins de composants.
- Coût réduit à la production de l’équipement.
- Ils permettent de contrôler l’état de l’environnement. [4]

1.2.3 Objets physiques et virtuels

- Les objets physiques : ils font partie du monde qui nous entoure, le monde physique. Ces objets peuvent être à la fois commandés, détectés et connectés. Les différents robots ainsi que tous les biens et équipements électriques sont des exemples d’objets physiques.
- Les objets virtuels : ils font partie du monde de l’information. C’est-à-dire que nous pouvons les stocker, les traiter et y accéder. Les contenus multimédias ou différents logiciels sont des exemples d’objets virtuels. [5]

1.3 Composants de l’Internet des Objets

Le tableau suivant représente les composants importants de l’IoT :

Composants IoT	Descriptions
Objets physiques	• Un objet connecté est un objet physique équipé de capteurs ou d’une puce qui lui permettent de transcender son usage initial pour proposer de nouveaux services. Il s’agit d’un matériel électronique capable de communiquer avec un ordinateur, un

	smartphone ou une tablette via un réseau sans fil (Wi-Fi, Bluetooth, réseaux de téléphonie mobile, réseau radio à longue portée, etc...), qui le relie à internet ou à un réseau local.
Capteurs	Ils sont installés sur les objets connectés, ils sont plus ou moins intelligents, selon qu'ils intègrent ou non eux-mêmes des algorithmes d'analyse de données. Les capteurs connus sont : capteurs de température, capteurs de pression, détecteurs d'intensité lumineuse, etc...
Utilisateurs	Exemple : Les humains peuvent contrôler l'environnement via des applications mobiles.
Prestations de service	Concerne les plateformes de services à qui sont destinées les données, exemple : Services Cloud.
Plateformes	Elle est considérée comme un type d'intergiciel utilisé pour connecter les composants IoT (objets, personnes, services, etc.) à l'environnement l'IoT.
Réseaux	Les composants IoT sont liés entre eux par des réseaux, utilisant diverses technologies, normes et protocoles sans fil et filaires.

Tableau 1.1 : Les composants de base d'un système IoT [6]

1.4 Caractéristiques fondamentales de l'Internet des Objets

Le paragraphe ci-dessous, indique les caractéristiques fondamentales de l'Internet des Objets ainsi que leurs exigences.

1.4.1 Interconnectivité

Tous les objets présents dans l'Internet des Objets peuvent être connectés à l'infrastructure mondiale de l'information et de la communication. Parmi les formes de connectivités les plus connues entre l'IoT et internet, nous retrouvons l'Ethernet. [7] Cependant, tous les appareils peuvent se connecter via une large variété de mode de connexion et de technologies, qu'elles soient avec (Ethernet) ou sans fil. Nous pouvons retrouver dans les options sans fil les technologies : Bluetooth, EDGE, GPRS, LTE, NFC, RFID, WLAN et ZigBee, etc. [8]

1.4.2 Services reliés aux objets

L'IoT est capable de fournir des services qui sont liés aux objets tout en tenant compte des contraintes telles que la protection de la vie privée ainsi que la cohérence sémantique entre les objets physiques. [8]

1.4.3 Hétérogénéité

Étant donné qu'ils sont basés sur des plates-formes matérielles ou des réseaux différents, les appareils reliés à l'IoT sont hétérogènes. Ils peuvent ainsi interagir avec d'autres appareils et dispositifs ou des plates-formes de service via d'autres réseaux. [8]

1.4.4 Changements dynamiques

L'état des dispositifs (par exemple connecté/déconnecté) change de manière dynamique tout comme le contexte dans lequel ces dispositifs fonctionnent qu'il soit relié au cadre spatio-temporel, ou également dans le cadre de la vitesse ou encore de la localisation. [8]

1.4.5 Très grande échelle

Dans le futur, l'ensemble des dispositifs sera au moins dix fois plus nombreux qu'à l'heure actuelle comme le précise l'UIT. Ils devront donc être à la fois gérés et être capables de communiquer entre eux. [7]

1.5 Architecture générale de l'IoT

Nous considérons en particulier la visualisation de l'IoT dans un modèle de référence développé par Cisco. Ce modèle a été développé pour fournir des définitions et des descriptions claires qui peuvent être appliquées avec précision aux éléments et aux fonctions des systèmes et des applications IoT. [9]

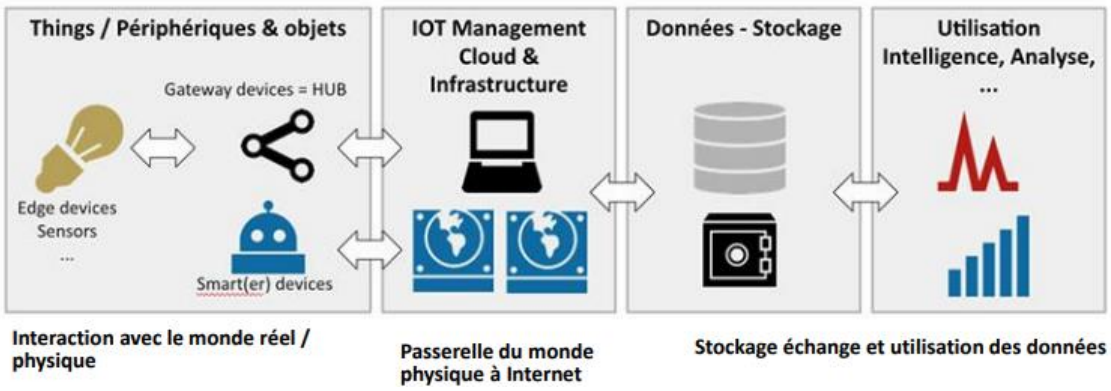


Figure 1.1 : Architecture générale de l'Internet des Objets [9]

1.6 Couches de l'Internet des Objets

L'architecture orientée aux services a été appliquée comme technologie habilitante qui divisait l'IoT en quatre couches [10] :

a Couche de détection

Cette couche est intégrée avec le hardware existant (capteurs, actionneurs, etc...) et sert à détecter, contrôler et collecter les informations en temps réel, interconnecte le monde physique et numérique.

b Couche de mise en réseau et passerelle

Cette couche fournit un soutien réseau de base et le transfert de données par un réseau sans ou avec fil, et prend en charge les exigences de communication pour la latence, la bande passante ou la sécurité.

c Couche de service

Cette couche crée et gère les services et fournit des services pour satisfaire les besoins des utilisateurs.

d Couche d'interface

Cette couche fournit des méthodes d'interaction aux utilisateurs et aux autres applications pour divers secteurs comme les transports, les soins de santé, l'agriculture, les chaînes d'approvisionnement, etc.

1.6.1 Niveaux de l'Internet des Objets

Le modèle de référence IoT comprend sept niveaux, chaque niveau est défini avec une terminologie qui peut être standardisée pour créer un cadre de référence. Le modèle de référence IoT ne se limite pas à la portée ou à la localisation de ses composants, dont lequel il décrit comment les tâches à chaque niveau doivent être traitées afin de permettre une grande évolutivité et assurer un soutien.

La figure présentée en dessous illustre le modèle de référence IoT et ses niveaux. Il est important de noter que dans l'IoT, les données circulent dans les deux sens. [11]

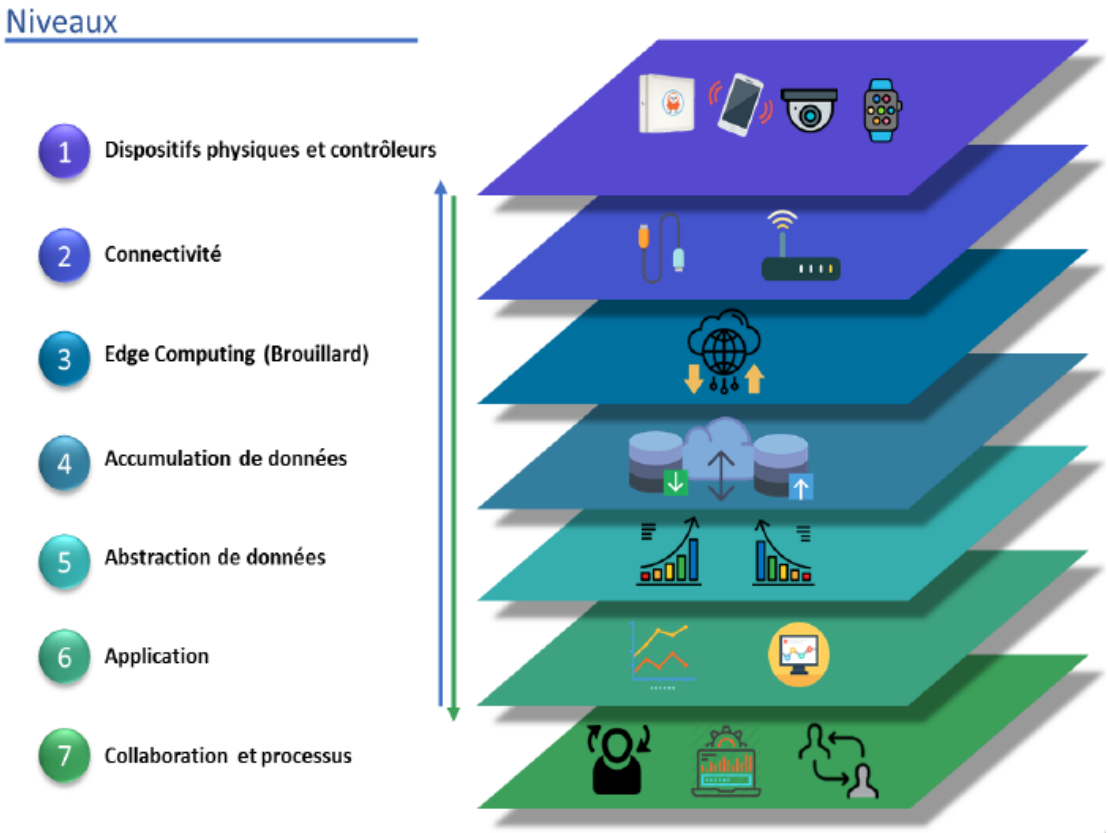


Figure 1.2 : Modèle de référence de L'IoT (CISCO, 2014) [11]

Ces couches de l'IoT basés sur le modèle de (CISCO, 2014) ont été présentées comme suit [11] :

a Niveau 1 : Dispositifs physiques et contrôleurs

Périphériques physiques et contrôleurs pouvant contrôler plusieurs périphériques. Ce sont les « objets » de l'Internet des Objets, et ils comprennent une large gamme d'appareils de terminaux qui envoient et reçoivent des informations.

b Niveau 2 : Connectivité

Les communications et la connectivité sont concentrées au niveau 2. La fonction la plus importante de ce niveau est la transmission fiable et opportune de l'information. Cela comprend les transmissions entre les appareils (niveau 1) et le réseau, entre le réseau et le traitement des informations de faible niveau se trouvant au niveau 3.

c Niveau 3 : Edge Computing

Les fonctions de niveau 3 sont représentées par la conversion des flux de données du réseau en informations appropriées pour le stockage et le traitement de niveau 4. Cela signifie que les activités de niveau 3 se concentrent sur l'analyse sur place des données, et leur transformation à haut volume.

Le traitement de niveau 3 peut couvrir de nombreux exemples, tels que :

- Évaluation : évalue des données pour déterminer si elles doivent être traitées à un niveau supérieur.
- Mise en forme : reformate des données pour un traitement cohérent de haut niveau.
- Expansion/décodage : manipule de données cryptiques avec un contexte supplémentaire (comme l'origine).
- Distillation/Réduction : réduit et/ou synthétise des données afin de minimiser l'impact des données et du trafic sur le réseau et les systèmes de traitement de niveau supérieur.
- Évaluation : détermine si les données représentent un seuil ou une alerte ; cela peut inclure la redirection de données vers des destinations supplémentaires.

d Niveau 4 : Accumulation de données

Les systèmes de réseau sont conçus pour déplacer les données en toute sécurité, les données sont en mouvement. Le modèle est piloté par les événements.

Les applications supposent généralement que les données sont «au repos » ou immuables en mémoire ou sur disque. Au niveau 4 (Accumulation de données), les données en mouvement sont converties en données au repos.

Au fur et à mesure que le niveau 4 capture les données et les met en veille, il devient possible d'utiliser ces données par des applications en temps non réel. Les applications accèdent aux données en cas de besoin.

e Niveau 5 : Abstraction de données

Les fonctions d'abstraction des données de niveau 5 se concentrent sur le rendu des données et leur stockage de manière à permettre le développement d'applications plus simples et améliorées. Avec plusieurs périphériques générant des données, il existe de nombreuses raisons qui expliquent pourquoi ces données ne peuvent pas se retrouver dans le même magasin de données :

- Il peut y avoir trop de données à mettre dans un endroit.
- Le transfert de données vers une base de données peut consommer beaucoup de puissance de traitement, de sorte que sa récupération doit être séparée du processus de génération de données.
- Les appareils peuvent être géographiquement séparés et le traitement est optimisé localement.
- Les niveaux 3 et 4 peuvent séparer les « flux de données brutes continues » des « données représentant un événement ».
- Le stockage de données pour le streaming de données peut être un système de données volumineux. Le stockage des données d'événement peut être un système de gestion de base de données relationnelle avec des temps de requête plus rapides.

Pour ces raisons, le niveau d'abstraction des données doit traiter beaucoup de choses différentes. Ceux-ci incluent :

- Concilier différents formats de données provenant de différentes sources.
- Assurer une sémantique des données cohérente entre les sources.
- Vérifiez que les données sont complètes pour l'application de niveau supérieur.

f Niveau 6 : Application

Le niveau 6 est l'endroit où l'interprétation de l'information se produit. Le logiciel à ce niveau interagit avec le niveau 5 et les données au repos. Les applications varient en fonction des marchés verticaux, de la nature des données de l'appareil et des besoins de l'entreprise. Par exemple, certaines applications se concentreront sur la surveillance

des données de l'appareil. Certains se concentreront sur le contrôle de l'appareil. Certains vont combiner l'appareil et non les données de l'appareil.

g Niveau 7 : Collaboration et processus

Les gens utilisent des applications et des données associées pour leurs besoins spécifiques. Souvent, plusieurs personnes utilisent la même application pour différentes raisons.

Les modèles de distribution, les standards et les études sur l'IoT et ses applications, y compris les villes et communautés intelligentes, fournissent la plateforme de normalisation spécialisée de l'IoT nécessaire à cette convergence vers un ensemble cohérent de normes internationales. Avec la participation des participants dans le domaine des technologies de l'information et de la communication, ce type d'étude aura une incidence sur la promotion du développement de « systèmes de systèmes » très efficaces qui contribueront à réduire la fracture numérique et à rendre le monde plus connecté.

1.7 Fonctionnement de l'Internet des Objets

Les réseaux sans fil fusionnent un large éventail de technologies de l'information qui englobent le matériel, les logiciels système, les réseaux et les méthodologies de programmation. La croissance de l'IoT devrait majoritairement être portée par l'utilisation de technologies sans fil et mobiles. Les technologies de connectivité sans fil sont nombreuses et variées, et l'usage de de ces technologies est souvent avant tout décidé par la portée du réseau envisagé. Certains cas d'usage nécessitent également l'association de technologies sans fil et filaire pour relier les équipements à des réseaux privés étendus ou à internet. [12]

1.7.1 Technologies de l'Internet des Objets

Plusieurs technologiques existent pour interconnecter les objets de l'Iot. Citons quelques exemples.

a RFID (Radio Frequency Identification)

RFID est une technologie sans fil qui est utilisée pour l'identification des objets, elle englobe toutes les technologies qui utilisent des ondes radio pour identifier automatiquement des objets ou des personnes. C'est une technologie qui permet de mémoriser et de récupérer des informations à distance grâce à une étiquette qui émet des ondes radio. L'étiquette contient des informations stockées électroniquement pouvant être lues à distance. [13] [14]

b WSN (Wireless Sensors Network)

WSN est un ensemble de nœuds qui communiquent sans fil et qui sont organisées en un réseau coopératif. Chaque nœud possède une capacité de traitement et peut contenir différent type de mémoire, un émetteur-récepteur RF et une source d'alimentation. Il peut aussi tenir compte des divers capteurs et actionneurs. [13]

c M2M (Machine to Machine)

M2M est l'association des technologies de l'information et de la communication avec des objets intelligents dans le but de donner à ces derniers les moyens d'interagir sans intervention humaine avec le système d'information d'une organisation ou d'une entreprise. [13] [14]

1.7.2 Protocoles de fonctionnement de l'Internet des Objets

L'Internet des Objets utilise plusieurs protocoles pour permettre la communication entre les différents systèmes, pour cela différents groupes ont été créée pour fournir des protocoles afin de faciliter et simplifier les taches des programmeurs d'applications et des fournisseurs de services.

a Protocoles de messagerie

Les protocoles de messagerie s'appuient sur un mécanisme de publication et d'abonnement, où les transferts de données se font de manière asynchrone. Citons deux types de protocoles de messageries [15] :

- **MQTT (Message Queuing Telemetry Transport)**

Est un protocole de messagerie de publication et d'abonnement (publish/subscribe) basé sur le protocole TCP/IP. Un client, appelé publisher, établi dans un premier temps une connexion de type 'publish' avec le serveur MQTT, appelé *broker.* Puis, le publisher transmet les messages au broker sur un canal spécifique, appelé topic. Par la suite, ces messages peuvent être lus par des abonnés, appelés subscribers, qui au préalable ont établi une connexion de type 'subscribe' avec le broker. Ainsi, la transmission et la consommation des messages se font de manière asynchrone.

Le schéma illustré dans la figure I.3, représente le fonctionnement détaillé du protocole MQTT. [15]

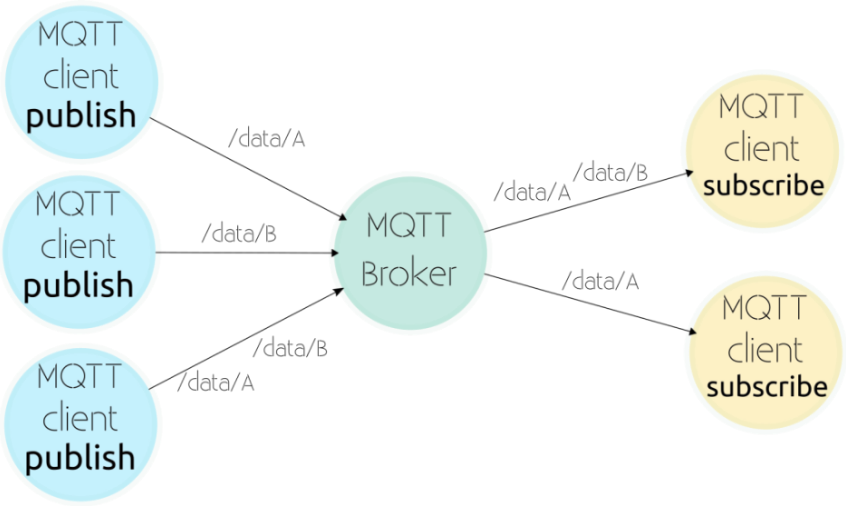


Figure 1.3 : Fonctionnement du protocole MQTT [15]

- **XMPP (Extensible Messaging and Presence Protocol)**

Est à l'origine un protocole de messagerie instantanée utilisé notamment dans les services Jabber et Google Talk. En outre, il permet l'utilisation dans d'autres applications telle que la VoIP. Il permet aux utilisateurs de communiquer entre eux en envoyant des messages instantanés sur internet quel que soit le système d'exploitation qu'ils utilisent. XMPP permet aux applications de messagerie instantanée d'accéder à l'authentification, au contrôle d'accès, à la mesure de la confidentialité, et à la compatibilité avec d'autres protocoles. Son fonctionnement est basé sur une architecture client/serveur où l'échange de données, au format XML, se fait sur le même principe que les messageries électroniques. [16]

b Protocoles de transfert web

- **CoAP (Constrained Application Protocol)**

Est un protocole de la couche application pour les applications IoT. Il définit un protocole de transfert Web basé sur les fonctionnalités HTTP et sur une architecture client/serveur, il est lié à la suite UDP/IPv6.

Le protocole CoAP est utilisé dans les applications de réseaux sociaux et mobiles et élimine l'ambiguïté en utilisant les méthodes HTTP get, post, put et delete. Il permet aux clients et aux serveurs d'exposer et de consommer des services. [13] [17]

- **API REST**

REST, pour (Representational State Transfer), est un protocole qui permet de gérer, identifier et manipuler des ressources (utilisateurs, images, données capteurs, etc.) par l'intermédiaire d'une interface de programmation d'application (ce que l'on appelle en anglais API (Application Programming Interface). Cette interface correspond à un ensemble d'URI accessible via les différentes méthodes (GET, PUT, POST et DELETE) des requêtes HTTP.

API REST est largement utilisé et a fait ses preuves, il est simple à mettre en place et à modifier grâce à une combinaison d'URI et de méthodes. Il reste adapté aux objets connectés qui n'ont pas de contraintes de communication comme par exemple les smartphones. [16]

c Protocoles réseaux

- **Websocket**

Le protocole Websocket permet l'établissement d'un canal de communication full-duplex en une seule connexion TCP entre un client et un serveur, Websocket permet donc d'établir des échanges de messages temps réel idéals pour les remontées d'alertes, des notifications. La figure ci-dessous indique les trois phases de la vie du canal [16];

- La phase de connexion appelé « Handshake » initié par le client.
- La phase d'échange bidirectionnel de messages.
- La phase de clôture du canal initié par l'une des deux parties.

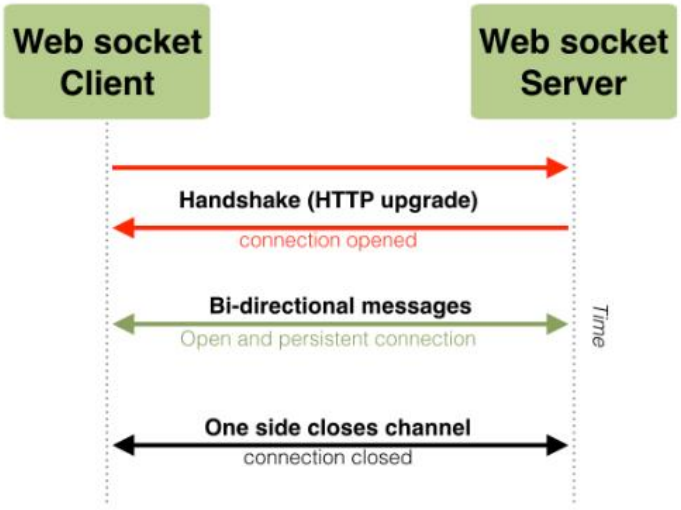


Figure 1.4 : Fonctionnement du protocole Websocket [16]

1.8 Domaines d'application de l'Internet des Objets

Ces derniers temps, le concept d'internet est devenu de plus en plus présent dans notre vie quotidienne, ce qui a entraîné une augmentation de nos besoins d'objets intelligents.

Les applications de l'Internet des Objets touchent pratiquement aujourd'hui tous les domaines. Comme le schéma ci-dessous le montre, on trouve alors l'IoT dans notre vie personnelle quotidienne et également dans les services publics offerts par le gouvernement. [18]

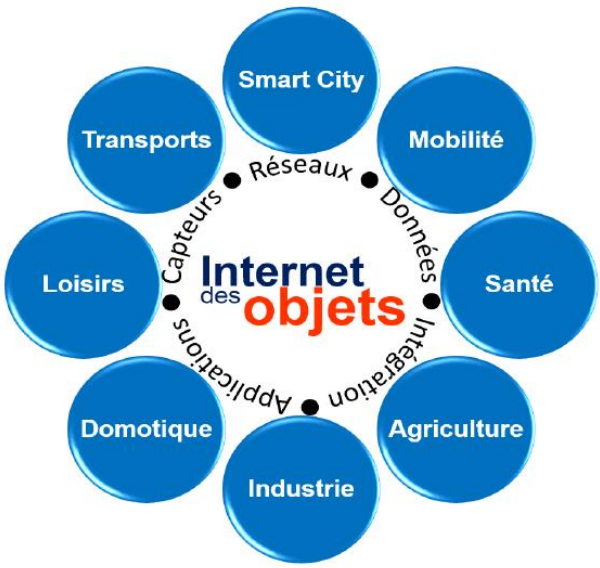


Figure 1.5 : Domaines d'application de l'IoT [18]

1.8.1 Transport et mobilité intelligente

Le développement de transport est l'un des facteurs qui indiquent le bien-être de pays. Des voitures connectées ou autonomes aux systèmes de transport intelligents, l'IoT peut sauver des vies, réduire le trafic et minimiser l'impact des véhicules sur l'environnement. Dans le domaine de l'automobile, l'IoT permet à la voiture de transformer les données en informations exploitables, à la fois dans le véhicule et dans le monde qui l'entoure, ce qui aide à optimiser la gestion du trafic urbain. [19]

1.8.2 Système de santé électronique

L'Internet des Objets trouve tout son intérêt dans le domaine médical. Les équipements et les capteurs sont de plus en plus « intelligents » et génèrent toujours plus de données nécessaires aux équipements médicaux, aux professionnels et profitant ainsi aux patients, en réduisant les coûts et en améliorant leur satisfaction. Les données ainsi collectées facilitent, adaptent, améliorent, anticipent ou réorganisent les soins des patients.

Des appareils intelligents dans les soins de santé sont utilisés pour stocker et gérer les paramètres de soins clés et pour gérer les données sur les maladies.

La figure ci-dessous illustre la conception d'un système intelligent de prise de décision clinique, matérialisé par le stockage des données collectées sur les patients et leur accessibilité universelle, procurerait au médecin un excellent appui durant la phase de traitement. L'Internet des Objets peut donc contribuer au développement du domaine médical. [19]

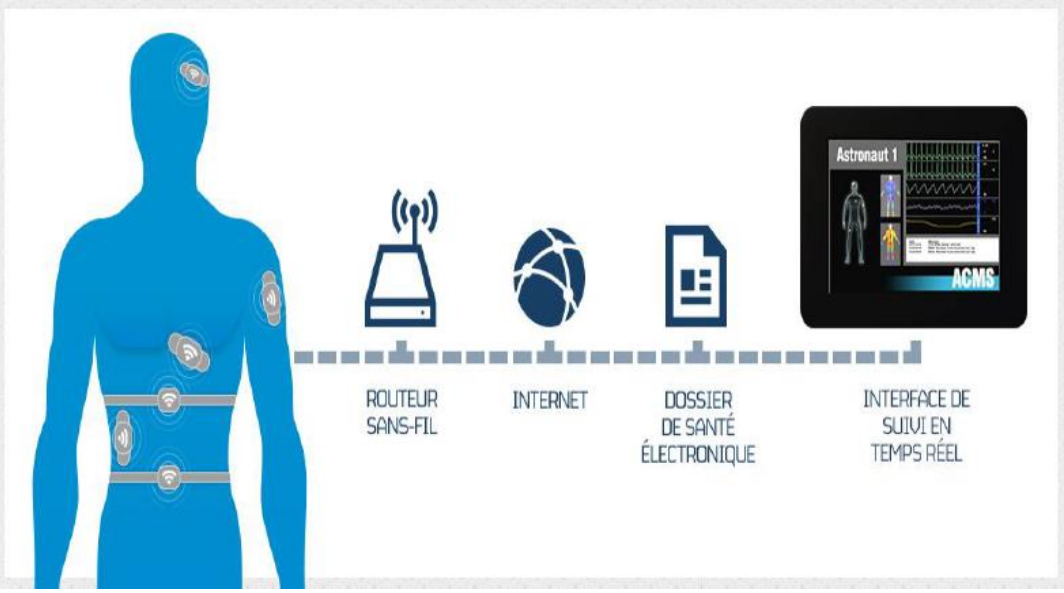


Figure 1.6 : Un système de santé électronique [19]

1.8.3 Internet des Objets dans le domaine de l'industrie

Le déploiement de L'IoT dans l'industrie sera un grand support pour le développement de l'économie et le secteur des services. En effet l'IoT permettra d'assurer un suivi total des produits, de la chaine de production, jusqu'à la chaine logistique et de distribution en supervisant les conditions d'approvisionnements.

L'usine intelligente a ajouté une nouvelle valeur dans la fabrication en intégrant l'intelligence artificielle, l'apprentissage automatique et l'automatisation du travail et la communication M2M avec le processus de fabrication. [20]

Figure 1.7 : L'IoT industriel [20]

1.8.4 Maisons et bâtiments intelligents

Les maisons et les bâtiments peuvent aujourd'hui exploiter de nombreux appareils et objets intelligemment. On trouve plusieurs applications intéressantes de l'IoT dans les maisons intelligentes et les bâtiments tel que : l'éclairage intelligent, le contrôle de l'air et de chauffage central, la gestion de l'énergie et la sécurité. D'autre part, l'internet avec des systèmes de gestion de l'énergie offre aussi la possibilité d'accéder aux systèmes d'information et de contrôler l'énergie d'un bâtiment par un ordinateur portable ou un smartphone placé n'importe où dans le monde, l'objectif de ces bâtiments étant aussi d'améliorer la performance thermique de l'habitat et de consommer moins d'énergie. [19]

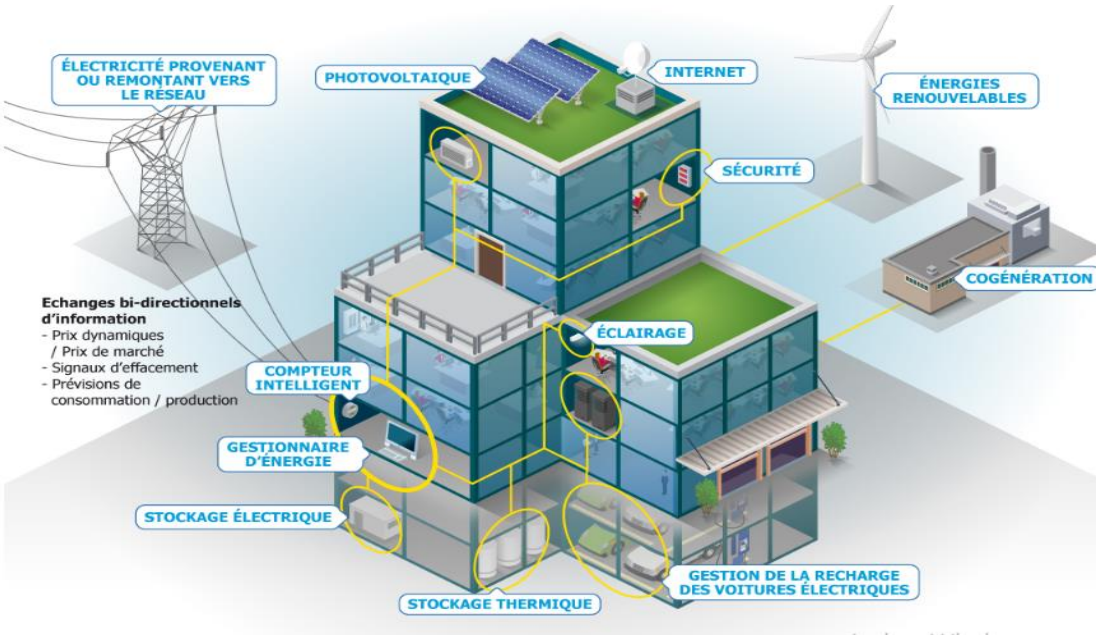


Figure 1.8 : Bâtiment intelligent [21]

1.8.5 Internet des Objets dans le domaine de l'agriculture

L'usage des objets connectés se démocratise dans l'agriculture. L'IoT peut offrir de nombreuses améliorations et des meilleurs outils à l'environnement des cultures, elle sert à optimiser l'eau d'irrigation, elle peut être utilisée pour lutter contre la pollution et améliorer la qualité de l'environnement en général, comme elle peut également interférer dans la surveillance de la croissance des plantes ou encore la prévention des risques météo.

Par exemple, les serres sont connues pour être énergivores, la construction de serres intelligentes permet d'économiser de l'argent et d'atteindre les objectifs agricoles.

Une serre intelligente dotée de la technologie IoT peut ainsi contrôler l'environnement. Des serveurs sur le Cloud peuvent accéder à distance au système informatique de la serre lorsqu'elle est connectée. Le traitement des données et la suppression de la surveillance manuelle permanente permettent alors de réaliser des économies de coûts. [19]

Figure 1.9 : Application dans l'industrie agricole [22]

1.9 Défis de l'Internet des Objets

L'Internet des Objets fait face à de nombreux défis de point de vu sécurité, du nombre colossal d'adresse IP demandé, en plus du coût de la mise en œuvre.

En effet l'adoption de d'IPv6 et son remplacement de l'IPv4 ont été une innovation énorme et cruciale pour l'avenir des communications Internet. IPv6 offre à l'IoT une sécurité renforcée, et permet de fournier un nombre très importants ($3{,}4\times10^{38}$) d'adresse IP, l'IPv6 peut être compressé en petits paquets réseau de seulement quelques dizaines d'octets grâce au 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks). Par une meilleure efficacité énergétique, les appareils à faible alimentation disposent aussi d'une connectivité réseau complète. [23]

Nous pouvons résumer ces défis dans cinq sous sections suivantes [24] :

1.9.1 Découverte automatique

Dans les environnements dynamiques les services appropriés pour les objets doivent être identifié automatiquement. Ceci nécessite des moyens sémantiques appropriés pour décrire leur fonctionnalité afin de les exploiter d'une manière efficace.

1.9.2 Interopérabilité

Chaque type d'objets intelligents dans l'Internet des objets possède différentes capacités de stockage d'information, de traitement et de communication. Différents objets intelligents seraient également soumis à des conditions différentes telles que la disponibilité de l'énergie et les besoins en bande passante de communications. A cet effet, pour faciliter la communication et coopération de ces objets, des normes communes sont requises.

1.9.3 Sécurité et confidentialité

Les aspects de sécurité et de protection d'Internet tel que la confidentialité des communications, l'intégrité des messages ainsi que l'authenticité et la fiabilité de partenaires de communication doivent être assurés dans un environnement IoT. De plus, d'autres exigences selon la particularité du domaine d'application, doivent être aussi considérées comme par exemple dans les transactions commerciales des objets intelligents doivent être protégés des yeux des concurrents.

1.9.4 Tolérance aux pannes

Les objets dans L'IoT sont beaucoup plus dynamiques et mobiles que les ordinateurs, d'où la nécessité d'utiliser des techniques telle que la redondance en plusieurs niveaux pour l'adaptation automatique au changement de conditions afin de traiter le problème de déconnexion.

1.9.5 Auto-Organisation

Les objets intelligents ne devraient pas être gérés comme des ordinateurs qui obligent leurs utilisateurs pour les configurer et les adapter à des situations particulières. Les objets mobiles, qui sont souvent utilisées de façon sporadique, doivent établir

spontanément des liens et pouvoir être organisé et configurer par eux-mêmes en fonction de leur environnement d'exécution.

Ces défis que rencontrent l'IoT, vont lui ouvrir la porte vers le grand but l'Internet of everything (internet de tout connecté), est l'internet multidimensionnel qui combine les champs de IoT et du big data (personnes, processus, données et objets).

1.10 Inconvénients de l'Internet des Objets

Il y a clairement des interrogations sur :

1.10.1 Menaces sur les données et les réseaux

Le manque de surveillance et de protection physique des objets communicants, car aucun système n'est à l'abri d'une cyber-attaque. [25]

1.10.2 Menaces sur la vie privée

Toutes les données récoltées peuvent propager au travers de plusieurs serveurs et Il y'a toujours le risque d'interception qui se pose. [26]

1.10.3 Impact sur nos actions journalières

Nous vivons actuellement dans une société où nos gestes sont de plus en plus liés à notre smartphone, ou tout objet dit connecté et cela implique des formes de dépendance face à cette connectivité. [9]

1.11 Sécurité dans l'Internet des Objets

La sécurité de l'internet des objets vise à assurer l'ensemble des données et des ressources matérielles et logicielles permettant de stocker ou de faire circuler les données d'une façon sécurisée. [27]

1. **La protection de la technologie** concerne la sécurité des données, des communications et des infrastructures réseaux et leurs fonctionnalités.
2. **La protection des personnes** concerne la protection de la vie privée des usagers pour éviter des litiges causés éventuellement par l'Internet des Objets. [28]

3. **La protection des systèmes interconnectés** et hébergeant les objets de l'IoT, concernera la protection des objets eux-mêmes livrés à ces systèmes et les processus qu'ils contrôleront. [29]

1.12 Matériels de l'Internet des Objets

Le hardware IoT est l'ensemble des appareils qui répondent et qui ont la capacité de capturer des données, il comprend une large gamme d'équipages tels que des appareils de routage, des capteurs, des actionneurs … etc. Ces appareils IoT gèrent des tâches et des fonctions clés telles que la génération de l'information, la détection des objets, la collecte de l'information, traitement de l'information, et la sécurité … etc. Il existe autant d'application IoT dans divers secteurs, alors qu'il est impossible de généraliser une architecture matérielle. [30]

1.12.1 Raspberry pi 3

Raspberry Pi 3 est un ordinateur très abordable et minuscule qui peut intégrer un serveur Web entier. Souvent appelé « RasPi », Cet ordinateur de petite taille avec de nombreuses performance, c'est une plate-forme parfaite pour interfacer avec de nombreux appareils. [30]

Les éléments essentiels qui compose la carte Raspberry pi sont : Processeur, puce graphique, mémoire de programme – RAM, et d'autres périphériques en option (diverses interfaces et connecteurs pour les périphériques).

1. **Caractéristiques**

Processeur	ARM QuadCore 1.2GHz
Mémoire	1GB RAM
Connectivité	• 4 Ports USB • Wifi 802.11n & Ethernet 10/100Mb • Bluetooth BLE 4.1
Vidéo et son	• HDMI • Port CSI (Camera)

	• Port DSI (Display)
Accès	40 Pins GPIO (pour interfacer des capteurs)
La carte SD	MicroSD Card

Tableau 1.2 : Caractéristiques de la carte RaspberryPI[30]

2. Rôles

- Gestion et contrôle des nœuds.
- Contrôler des éléments mécaniques : systèmes, lumières, moteurs, etc. [31]

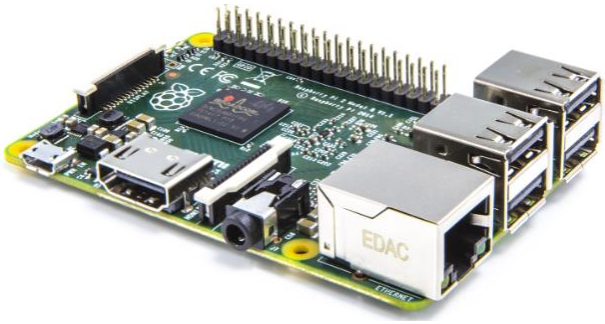


Figure 1.10 : Carte Raspberry Pi 3 [31]

1.12.2 BeagleBoard

BeagleBoard est un ordinateur à carte unique avec un système d'exploitation basé sur Linux qui utilise un processeur ARM, capable d'un traitement plus puissant que RasPi. Les cartes Galileo et Edison du géant de la technologie Intel sont d'autres options, toutes deux idéales pour une production à plus grande échelle. [31]

1. Caractéristiques

Développeur	Texas Instruments et Digi-Key
Processeur	ARM Cortex A8 700 MHz à 1 GHz
Stockage	Carte SD
Mémoire	128 Mo à 512 Mo
Connectivité	USB On-The-Go, DVI-D, PC audio, SDHC, JTAG, HDMI

Tableau 1.3 : Caractéristiques de la carte BeagleBoard [31]

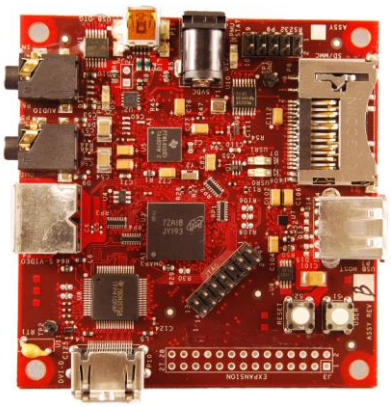


Figure 1.11 : Carte BeagleBoard [31]

1.12.3 Capteurs (sensors)

2. Définition

Le matériel le plus important dans l'IoT pourrait être ses capteurs, ils sont la source des données pour cela, ils sont utilisés pour recueillir des informations sur l'environnement. Ces appareils sont constitués de modules d'énergie, de modules de gestion de l'alimentation, de modules RF et de modules de détection. Les modules RF gèrent les communications via leur traitement du signal, WiFi, ZigBee, Bluetooth, émetteur-récepteur radio.

Les capteurs sont des objets électroniques de taille extrêmement réduite avec des ressources très limitées, autonomes, capable de traiter des informations et de les transmettre, via les ondes radio, à une autre entité (capteurs, unité de traitements...) sur une distance limitée à quelques mètres. [32]

3. Fonctionnement :

Un meilleur terme pour un capteur est un transducteur, un transducteur est un appareil physique qui convertit une forme d'énergie en une autre. Ainsi, dans le cas d'un capteur, le transducteur convertit un phénomène physique en une impulsion électrique qui peut ensuite être interprétée pour déterminer une lecture. [33]

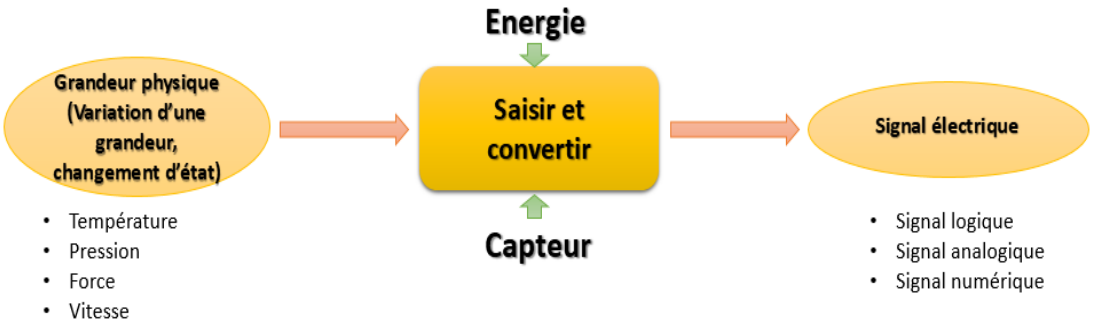


Figure 1.12 : Fonctionnement d'un capteur

4. Rôles

- Récoltent les données de leur environnement.
- Permettent de faire des mesures de l'environnement physique (ex : température, humidité, bruit).
- Transforme l'état d'une grandeur physique observée en une grandeur utilisable. [33]

5. Classification des capteurs

Il existe deux classifications de capteurs :

1) Capteurs actifs et passifs

- Capteurs actifs [34] :

Les capteurs actifs sont le type de capteurs qui produit un signal de sortie à l'aide d'une alimentation d'excitation externe. Les propriétés physiques propres du capteur varient en fonction de l'effet externe appliqué. Par conséquent, il est également appelé capteurs auto-générateurs.

- Capteurs passifs :

Les capteurs passifs sont le type de capteurs qui produit un signal de sortie sans l'aide d'une alimentation d'excitation externe. Ils n'ont pas besoin de stimulus ou de tensions supplémentaires.

Exemple : Thermocouple, qui génère une valeur de tension correspondant à la chaleur appliquée. Il ne nécessite aucune alimentation externe.

2) Capteurs analogiques et numériques

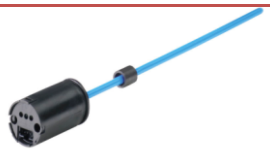
- Capteur analogique :

Un capteur analogique convertit la quantité physique mesurée sous forme analogique (continue dans le temps).

- Capteur numérique :

Un capteur numérique produit une sortie sous forme d'impulsion. Les codeurs sont des exemples de capteurs numériques.

6. Types de capteurs

Capteur de position : Ce sont des capteurs à contact. Ils peuvent être équipe d'un galet, d'une tige souple, d'une bille. L'information donnée par ce type de capteur est de type tout ou rien et peut être électrique ou pneumatique.	 **Capteur de position linéaire**
Capteur à fuite	
Capteur de température : (pyromètre, thermomètre, sonde PT100, thermocouple, thermistance...).	
Capteur de pression : (tube de Bourdon, capsule anéroïde, piézo-électrique, corde vibrante, baromètre, hypsomètre...).	
Capteur de lumière : (photodiode ou phototransistor, capteur photographique, cellule photoélectrique...).	

Capteur de courant : (Capteur de courant à effet Hall …).	 **Capteur de courant linéaire à effet hall**
Capteur de son : (microphone, hydrophone...).	

Tableau 1.4 : Types de capteurs [35]

1.12.4 Actionneurs

1. **Définition**

Un actionneur est un appareil qui convertit l'énergie en mouvement. Il est généralement utilisé pour appliquer une force sur quelque chose, ils fournissent le mouvement à un système de collecte de données, pour récupérer les détails en fonction des mouvements, il peut agir sur l'environnement. [33]

2. **Fonctionnement**

Un actionneur fonctionne dans le sens inverse d'un capteur. Il prend une entrée électrique et la transforme en action physique. [36]

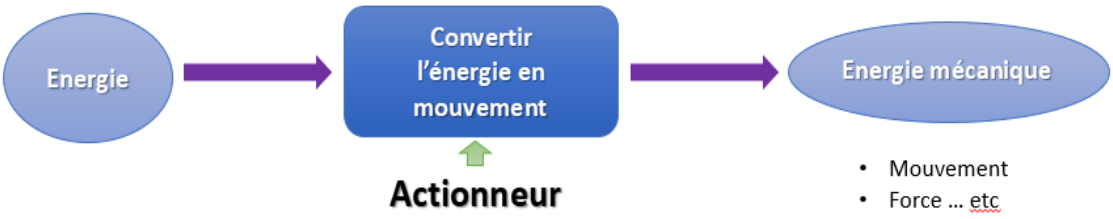


Figure 1.13 : Fonctionnement d'un actionneur

3. Types d’actionneurs

Un servomoteur : est un actionneur rotatif ou un actionneur linéaire qui permet un contrôle précis de la position angulaire ou linéaire, de la vitesse et de l'accélération. Il se compose d'un moteur approprié couplé à un capteur pour le retour de position. Il nécessite également un contrôleur relativement sophistiqué, souvent un module dédié spécialement conçu pour être utilisé avec des servomoteurs.	
Les moteurs pas à pas : sont des moteurs à courant continu qui se déplacent par étapes discrètes. Ils ont plusieurs bobines qui sont organisées en groupes appelés ‘’phases’’. En excitant chaque phase en séquence, le moteur tournera, une étape à la fois.	
Les moteurs à courant continu (CC) : sont des moteurs à rotation continue à deux fils (alimentation et masse). La plupart des moteurs à courant continu fonctionnent à un régime élevé (tours par minute), par exemple les ventilateurs de refroidissement d'ordinateur ou les roues de voiture radiocommandées.	
Un actionneur linéaire : est un actionneur qui crée un mouvement en ligne droite, contrairement au mouvement circulaire d'un moteur électrique conventionnel. Les actionneurs linéaires sont utilisés dans les machines-outils et les machines industrielles, dans les périphériques informatiques tels que les lecteurs de disque et les imprimantes, dans les vannes et les amortisseurs, et dans de nombreux autres endroits où un mouvement linéaire est requis.	

Un relais : est un interrupteur à commande électrique. De nombreux relais utilisent un électro-aimant pour actionner mécaniquement un interrupteur, mais d'autres principes de fonctionnement sont également utilisés, tels que les relais à semi-conducteurs. Les relais sont utilisés là où il est nécessaire de contrôler un circuit par un signal distinct de faible puissance, ou lorsque plusieurs circuits doivent être contrôlés par un seul signal.	
Un solénoïde : est simplement un électroaimant spécialement conçu. Les solénoïdes sont peu coûteux et leur utilisation est principalement limitée aux applications tout ou rien telles que le verrouillage, le verrouillage et le déclenchement. Ils sont fréquemment utilisés dans les appareils électroménagers (par exemple les vannes de machine à laver), l'équipement de bureau (par exemple les photocopieuses), les automobiles (par exemple les verrous de porte et le solénoïde du démarreur), les flippers (par exemple, les pistons et les pare-chocs) et l'automatisation d'usine.	

Tableau 1.5 : Types d'actionneurs [37]

1.12.5 Nœuds

Ce sont des cartes à microcontrôleurs (petits ordinateurs) contenant une unité de traitement programmable et des broches d'entrée/sortie. Les nœuds sont des systèmes embarqués qui assurent la transmission des valeurs mesurées par des capteurs standards au système d'évaluation via un système de transmission, et raccordent les objets connectés à internet. [38]

1. NodeMCU (basé sur un ESP8266)

Module basé sur le microcontrolleur ESP8266 cadencé à 80 MHz et exécutant le firmware open source NodeMCU. Cette carte se programme via l'IDE Arduino et est compatible avec les scripts LUA.

Ce microcontrôleur dispose d'une interface Wifi idéale pour les objets connectés. Des connecteurs latéraux mâles et femelles permettent d'enficher le module sur une plaque de montage rapide.

L'interface sans fil Wifi permet la création de point d'accès sans fil, l'hébergement d'un serveur, la connexion à internet et le partage des données par exemple. [39]

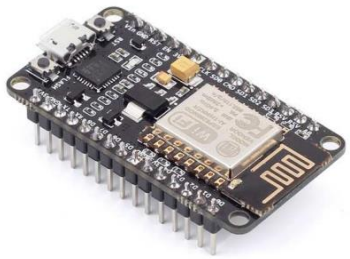


Figure 1.14 : NodeMCU (ESP8266) [39]

Alimentation	• 5 Vcc via micro-USB • 5 à 9 Vcc via broche Vin (régulateur intégré)
Microcontrôleur	ESP8266
Microprocesseur	Tensilica LX106
Fréquence	80 MHz
Mémoire RAM	64 KB
Mémoire Flash	96 KB
Interfaces	I2C, SPI, UART
Interface Wifi	802.11 b/g/n 2,4 GHz
Dimensions	58 x 31 x 12 mm

Tableau 1.6 : Caractéristiques du NodeMCU (ESP8266) [39]

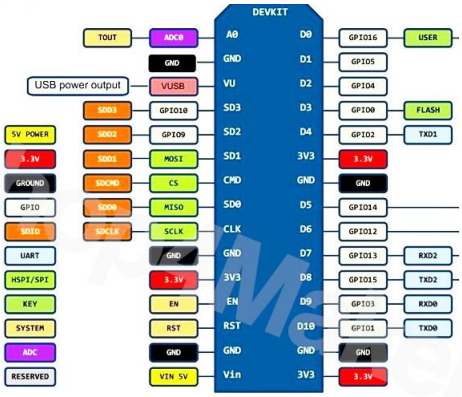


Figure 1.15 : Brochage du NodeMCU (ESP8266) [39]

2. B-L475E-IOT01A

Le kit de découverte B-L475E-IOT01A pour nœud IoT permet aux utilisateurs de développer des applications avec une connexion directe aux serveurs cloud.

Le kit Discovery permet une grande diversité d'applications en exploitant la communication basse consommation, la détection multivoie et les fonctionnalités de la série STM32L4 basées sur le cœur Arm Cortex-M4.

La prise en charge de la connectivité Arduino Uno V3 et PMOD offre des capacités d'extension illimitées avec un large choix de cartes complémentaires spécialisées. [40]

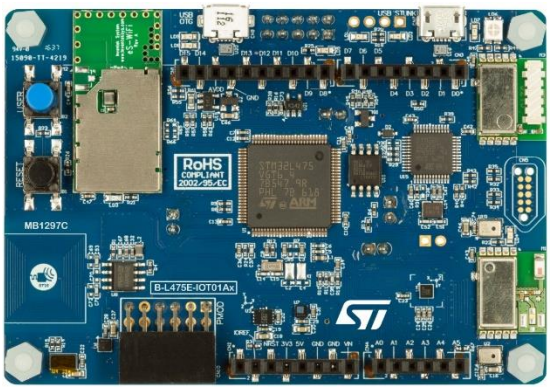


Figure 1. 16 : Nœud B-L475E-IOT01A [40]

3. Caractéristiques [40]

- MCU de la série STM32L4 à très faible consommation d'énergie basés sur le cœur Arm Cortex-M4 avec 1 Mo de mémoire Flash et 128 Ko de SRAM, dans un boîtier LQFP100.
- Mémoire Flash Quad-SPI (Macronix) 64 Mbit.

- ❖ Module Bluetooth V4.1 (SPBTLE-RF).
- ❖ Module RF programmable basse puissance sub-GHz (868 MHz ou 915 MHz) (SPSGRF-868 ou SPSGRF-915).
- ❖ Module Wi-Fi compatible 802.11 b / g / n d'Inventek Systems (ISM43362-M3G-L44).
- ❖ Tag NFC dynamique basé sur M24SR avec son antenne NFC imprimée.
- ❖ Capteur numérique capacitif pour l'humidité relative et la température (HTS221).
- ❖ 2 microphones numériques omnidirectionnels (MP34DT01).
- ❖ Y a 2 fréquences : le model B-L475E-IOT01A1 = 915 MHz, et le model B-L475E-IOT01A2 = 868 MHz.

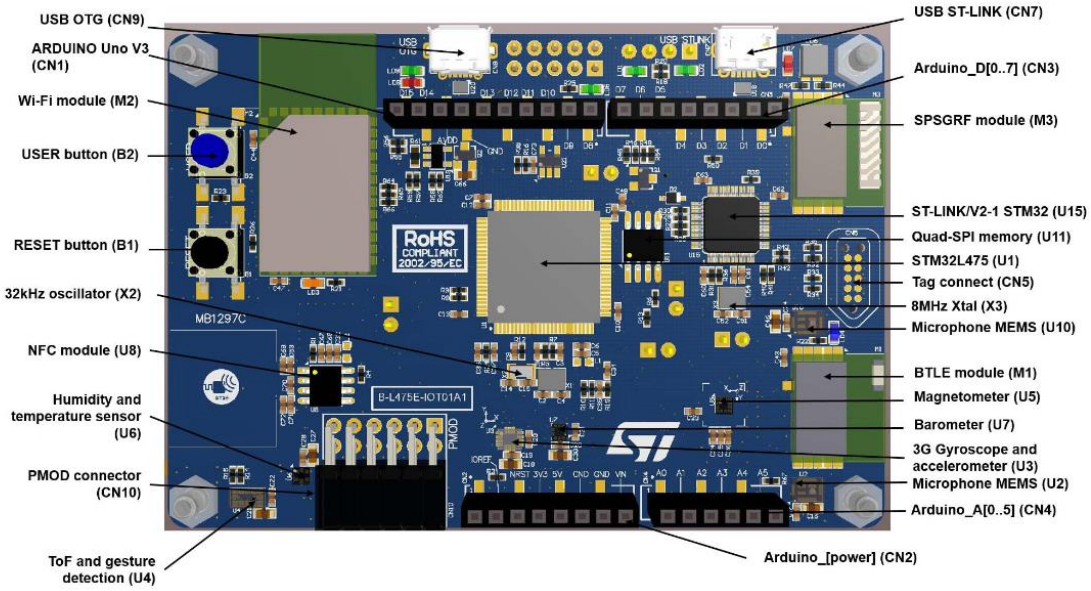


Figure 1.17 : Brochage du Nœud B-L475E-IOT01A[40]

1.13 Conclusion

Dans ce chapitre, nous avons essayé de donner des généralités sur l'Internet des objets. Nous avons donné un bref historique sur l'IoT, sa définition, et ses principaux composants. Puis nous avons cité les caractéristiques fondamentales de l'IoT, et son architecture. Nous avons aussi abordé son fonctionnement en citant ses technologies et quelques protocoles de fonctionnement, et nous avons rappelé brièvement les domaines d'application de l'IoT et ses défis, nous avons présenté par la suite les inconvénients de l'IoT et l'aspect sécurité. Et enfin nous avons terminé par une brève description du matériel utilisé pour la conception d'un système IoT.

Chapitre 2 Conception et fonctionnement de la plateforme

La technologie de l'Internet des Objets permet de contrôler à distance différents éléments que nous utilisons dans la vie quotidienne, ou dans un domaine professionnel. Qui dit réseaux dit architecture, d'où la nécessité de définir une structure d'architecture pour IoT. Des normes ont été mis en place afin de définir un vocabulaire commun, des schémas conceptuels réutilisables et unifier les pratiques dans le secteur.

Pour mettre en place une architecture IoT, nous devons considérer les deux aspects matériel et logiciel.

Dans ce chapitre, nous allons voir en détail les étapes d'élaboration de notre plateforme, nous allons décrire l'architecture de l'IoT physique et logique, ainsi que les langages de programmation utilisés dans notre plateforme.

2.1 Architecture IoT

Le but de notre travail est de réaliser une plateforme générique, fiable, et ergonomique qui s'adapte facilement avec n'importe quel système IoT. L'architecture proposée est modulaire et peut être représenté suivant trois parties :

Software : relative aux applications de la plateforme permettant à l'utilisateur de contrôler son propre réseau IoT.

Hardware : relative au côté physique du réseau incorporant les objets connectés qui vont être contrôlé à distance.

Protocoles et communication : Et enfin le lien entre ces deux parties, ceci concerne les protocoles utilisés.

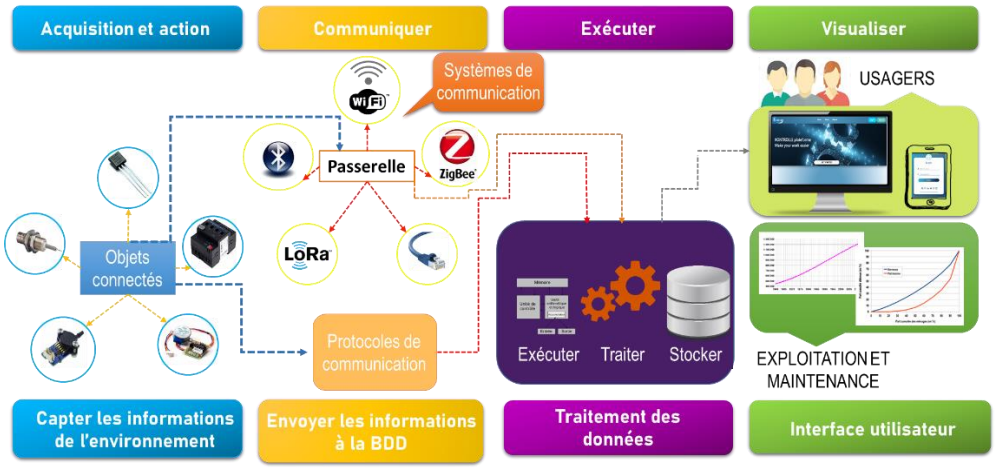


Figure 2.1 : Architecture physique de l'Internet des Objets

L'architecture illustrée ci-dessus, représente le schéma physique de l'Internet des Objets, nous expliquons par la suite les différentes parties de cette architecture.

2.1.1 Capteur et actionneur

Dans cette section nous somme au niveau de l'objet connecté, nous parlons des capteurs qui vont capter des informations de l'environnement (relever une mesure) qui peut être une température, une pression, une géolocalisation..., nous trouvons aussi des actionneurs qui peuvent agir sur l'environnement.

2.1.2 Communication

Dans cette phase, les mesures relevées (données) vont être transporté à la base de données grâce à des cartes (Raspberry pi, BeagleBoard) et des nœuds (NodeMCU …) qui sont connectés entre eux via des protocoles de communication (MQTT, HTTP, CoAP, etc), et à travers des technologies de communication sans fil (Bluetooth, Zigbee, Wifi, …) intégrés, ou bien filaire (Ethernet), afin d'établir la connexion entre le réseau local et le réseau externe.

2.1.3 Exécution

Dans cette partie la plateforme IoT se charge de contrôler les objets connectés, de collecter les données, les stocker, les traiter, les analyser, et les envoyer à des applications.

2.1.4 Visualisation

L'information est rendue utile et lisible pour l'utilisateur, dans cette section la plateforme permet d'offrir aux usagers plusieurs services à travers des applications, dont lesquelles ils peuvent consulter les données collectées, et piloter leurs objets.

Le schéma ci-dessous explique la circulation de données entre le réseau interne et le réseau externe.

Au niveau du réseau local, les capteurs/actionneurs communiquent avec les nœuds via câble. La communication entre les nœuds et la carte raspberry pi est sans fil.

Les données des capteurs transitant par les nœuds sont envoyées au serveur MQTT (mosquitto), ce serveur est installé sur une carte Raspberry. Il est possible de visualiser les données à partir d'un réseau externe via internet. En effet la carte raspberry dispose d'une connexion internet wifi via un routeur installé localement.

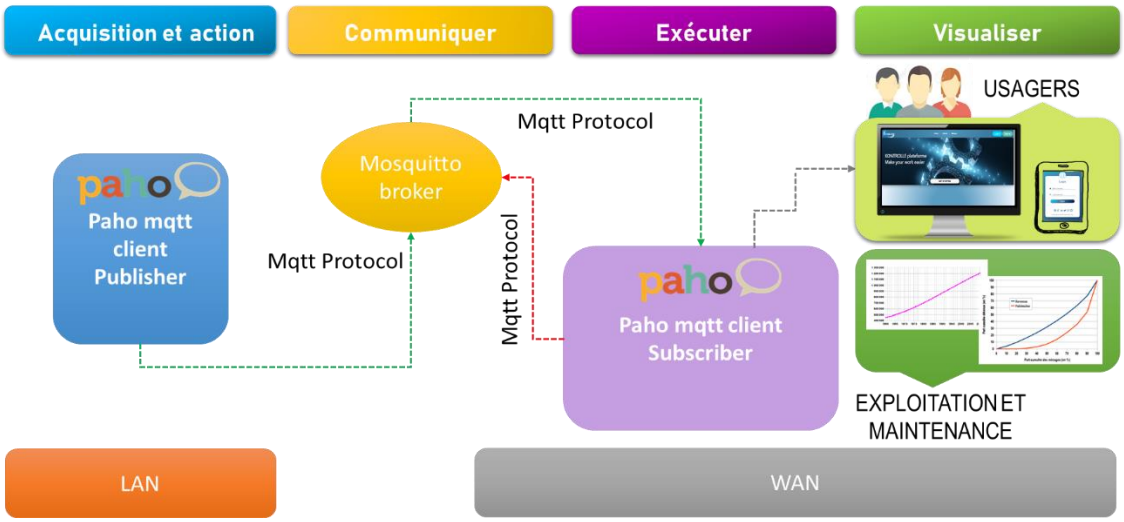


Figure 2.2 : Architecture logique de l'Internet des Objets

2.2 Partie logicielle de la Plateforme

2.2.1 Principe de fonctionnement

Un utilisateur peut s'inscrire en créant un compte sur la plateforme et ainsi être en mesure de créer ses propres applications IoT. Pour chaque application crée, il peut ajouter les capteurs et actionneurs de son système.

Pour chaque capteur/actionneur, l'utilisateur doit souscrire à un topic afin de publier les données en cas de capteurs et recevoir des notifications en cas d'actionneurs.

Le schéma suivant résume la fonctionnalité de cette plateforme :

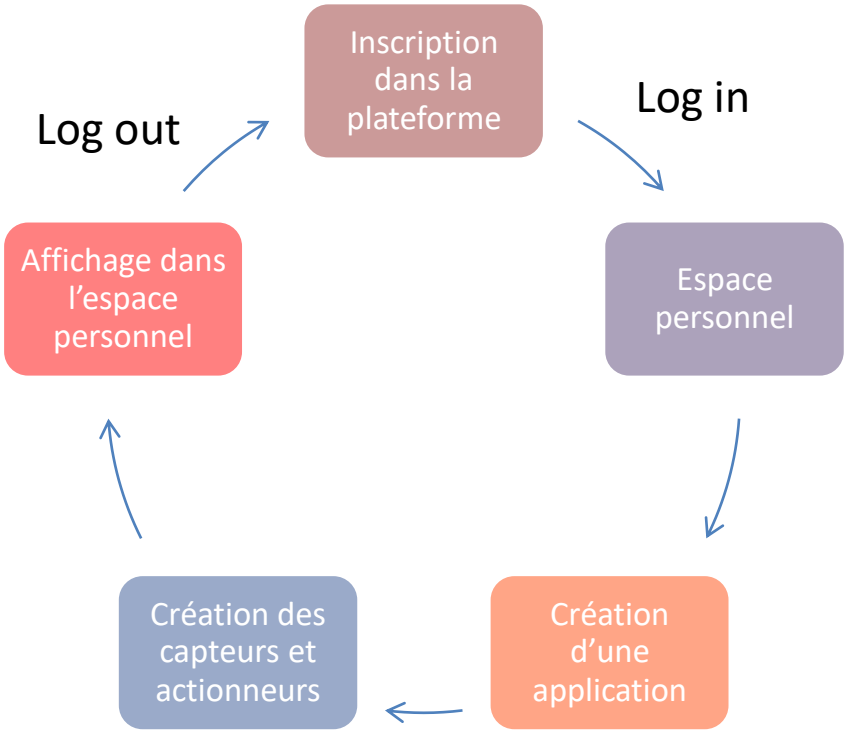


Figure 2.3 : Schéma présentatif de fonctionnement de la plateforme

2.2.2 Logiciels utilisés

Dans notre projet nous avons besoin d'utiliser certains outils logiciels qui nous permettent de réaliser notre interface.

a Langage de programmation python

Ce langage de programmation présente de nombreuses caractéristiques intéressantes [41] :

- Il est multiplateforme, c'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
- Il est gratuit, nous pouvons l'installer sur autant d'ordinateurs que nous voulons.

- C'est un langage de haut niveau, il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété, un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
- Il est orienté objet, c'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc) avec un certain nombre de règles de fonctionnement et d'interactions.
- Il est relativement simple à prendre en main.

Figure 2.4 : Langage python [42]

b Le Framework Django

Nous allons utiliser Django comme un Framework de développement web.

Un Framework est un ensemble d'outils qui simplifie le travail d'un développeur, traduit littéralement de l'anglais. Un Framework est un « cadre de travail », Il apporte les bases communes à la majorité des programmes ou des sites web, celles-ci étant souvent identiques (le fonctionnement d'un espace membres est commun à une très grande majorité de sites web de nos jours), un développeur peut les réutiliser simplement et se concentrer sur les particularités de son projet.

Il s'agit donc d'un ensemble de bibliothèques coordonnées, qui permettent à un développeur d'éviter de réécrire plusieurs fois une même fonctionnalité, et donc d'éviter de réinventer constamment la roue. [43]

- **Avantages d'un Framework**

Un Framework instaure en quelque sorte sa « ligne de conduite ». Tous les développeurs Django codent de façon assez homogène (leurs codes ont le même fonctionnement, les mêmes principes). De ce fait, lorsqu'un développeur rejoint un projet utilisant un Framework qu'il connaît déjà, il comprendra très vite ce projet et pourra se mettre rapidement au travail

Le fait que chaque Framework possède une structure commune pour tous ses projets à une conséquence tout aussi intéressante : en utilisant un Framework, notre code sera le plus souvent déjà organisé, propre et facilement réutilisable par autrui.

Voici d'ailleurs un grand défi des Frameworks : bien que ceux-ci doivent instaurer une structure commune, ils doivent aussi être souples et modulables, afin de pouvoir être utilisés pour une grande variété de projets, du plus banal au plus exotique.

Django est donc un Framework Python destiné au web, ce n'est pas le seul dans sa catégorie, nous pouvons compter d'autres Frameworks Python du même genre comme : **web2py**, **TurboGears**, **Tornado** ou encore **Flask**. Il a cependant le mérite d'être le plus exhaustif, d'automatiser un bon nombre de choses et de disposer d'une très grande communauté. [43]

Figure 2.5 : Framework Django [43]

Aujourd'hui, Django est devenu très populaire et est utilisé par des sociétés du monde entier, telles qu'**Instagram**, **Pinterest**, et même la **NASA**…etc.

Figure 2.6 : Exemple des Applications à base de Django [44]

- **Fonctionnement de Django**

La construction de la plupart des Frameworks ce fait grâce au modèle MVC, et Django s'articulent autour du modèle MVT, que nous introduirons également.

1. Architecture MVC

Lorsque nous parlons de Frameworks qui fournissent une interface graphique à l'utilisateur (soit une page web, comme ici avec Django, soit l'interface d'une application graphique classique, comme celle de traitement de texte par exemple), nous parlons souvent de l'architecture MVC. Il s'agit d'un modèle distinguant plusieurs rôles précis d'une application, qui doivent être accomplis. Comme son nom l'indique, l'architecture (ou « patron ») Modèle-Vue-Contrôleur est composé de trois entités distinctes, chacune ayant son propre rôle à remplir.

Tout d'abord, le modèle représente une information enregistrée quelque part, le plus souvent dans une base de données. Il permet d'accéder à l'information, de la modifier, d'en ajouter une nouvelle, de vérifier que celle-ci correspond bien aux critères (nous parlons d'intégrité de l'information), de la mettre à jour, etc. Il s'agit d'une interface supplémentaire entre le code et la base de données, mais qui simplifie grandement les choses, comme nous le verrons par la suite.

Ensuite la vue qui est, comme son nom l'indique, la visualisation de l'information. C'est la seule chose que l'utilisateur peut voir. Non seulement elle sert à présenter une donnée, mais elle permet aussi de recueillir une éventuelle action de l'utilisateur (un clic

sur un lien, ou la soumission d'un formulaire par exemple). Typiquement, un exemple de vue est une page web, ni plus, ni moins.

Finalement, le contrôleur prend en charge tous les événements de l'utilisateur (accès à une page, soumission d'un formulaire, etc.). Il se charge, en fonction de la requête de l'utilisateur, de récupérer les données voulues dans les modèles. Après un éventuel traitement sur ces données, il transmet ces données à la vue, afin qu'elle s'occupe de les afficher. Lors de l'appel d'une page, c'est le contrôleur qui est chargé en premier, afin de savoir ce qu'il est nécessaire d'afficher. [45]

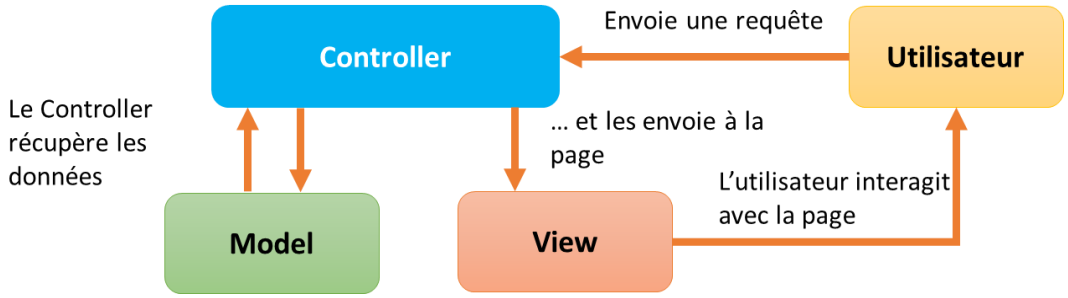


Figure 2.7 : Schéma explicatif de l'architecture de MVC

2. Architecture MVT

L'architecture utilisée par Django diffère légèrement de l'architecture MVC classique. En effet, la « magie » de Django réside dans le fait qu'il gère lui-même la partie contrôleur (gestion des requêtes du client, des droits sur les actions…). Ainsi, nous parlons plutôt de Framework utilisant l'architecture MVT : Modèle-Vue-Template.

Cette architecture reprend les définitions de modèle et de vue que nous avons vues, et en introduit une nouvelle : le Template (voir figure suivante).

Un Template est un fichier HTML, aussi appelé en français « gabarit ». Il sera récupéré par la vue et envoyé au visiteur ; cependant, avant d'être envoyé, il sera analysé et exécuté par le Framework, comme s'il s'agissait d'un fichier avec du code. Django fournit un moteur de Template très utile qui permet, dans le code HTML, d'afficher des variables, d'utiliser des structures conditionnelles (if/else) ou encore des boucles (for), etc.

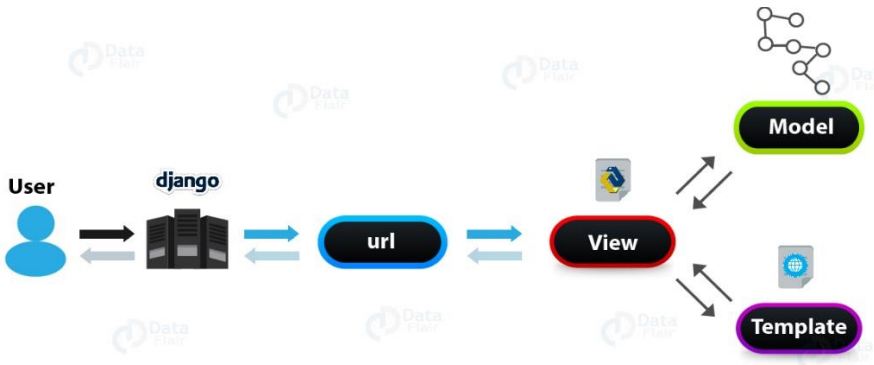


Figure 2.8 : Schéma explicatif de l'architecture de MVC [45]

Concrètement, lorsque l'utilisateur appelle une page d'un site réalisé avec Django, le Framework se charge, via les règles de routage URL définies, d'exécuter la vue correspondante. Cette dernière récupère les données des modèles et génère un rendu HTML à partir du Template et de ces données. Une fois la page générée, l'appel fait chemin arrière, et le serveur renvoie le résultat au navigateur de l'utilisateur.

Nous distinguons les quatre parties qu'un développeur doit gérer :

- ✓ Le routage des requêtes, en fonction de l'URL.
- ✓ La représentation des données dans l'application, avec leur gestion (ajout, édition, suppression…), c'est-à-dire les modèles.
- ✓ L'affichage de ces données et de toute autre information au format HTML, c'est-à-dire les Templates.
- ✓ Enfin le lien entre les deux derniers points : la vue qui récupère les données et génère le Template selon celles-ci.

Le développeur se doit de fournir le modèle, la vue et le Template, une fois cela fait, il suffit juste d'assigner la vue à une URL précise, et la page est accessible.

Si le Template est un fichier HTML classique, un modèle en revanche sera écrit sous la forme d'une classe où chaque attribut de celle-ci correspondra à un champ dans la base de données. Django se chargera ensuite de créer la table correspondante dans la base de données, et de faire la liaison entre la base de données et les objets de notre classe.

Non seulement il n'y a plus besoin d'écrire de requêtes pour interagir avec la base de données, mais en plus le Framework propose la représentation de chaque entrée de la table sous forme d'une instance de la classe qui a été écrite.

Il suffit donc d'accéder aux attributs de la classe pour accéder aux éléments dans la table et pouvoir les modifier, ce qui est très pratique.

Enfin, une vue est une simple fonction, qui prend comme paramètres des informations sur la requête (s'il s'agit d'une requête GET ou POST par exemple), et les paramètres qui ont été donnés dans l'URL.

En résumé Django respecte l'architecture MVT, directement inspirée du très populaire modèle MVC, il gère de façon autonome la réception des requêtes et l'envoi des réponses au client (partie contrôleur), un projet est divisé en plusieurs applications, ayant chacune un ensemble de vues, de modèles et de schémas d'URL, si elles sont bien conçues, ces applications sont réutilisables dans d'autres projets, puisque chaque application est indépendante. [45]

2.3 Réalisation du Framework

Sur la base de l'explication de l'architecture MVT de Django nous allons construire notre plateforme, pour cette dernière nous devons ouvrir un nouveau projet Django dans un environnement virtuel (Venv) et installer nos outils de projets.

Les environnements virtuels sont un mécanisme très souvent utilisé pour la gestion de dépendances et isolation entre projets python développés dans une même machine.
Chaque projet Django, contient des fichiers python fondamentales où se trouvent les paramètres de chaque projet (settings.py et urls.py).

NB : tous les commandes utiliser soit de Django soit de mosquitto broker sont valable pour tous les systèmes d'exploitation (Linux, Windows, Mac os).

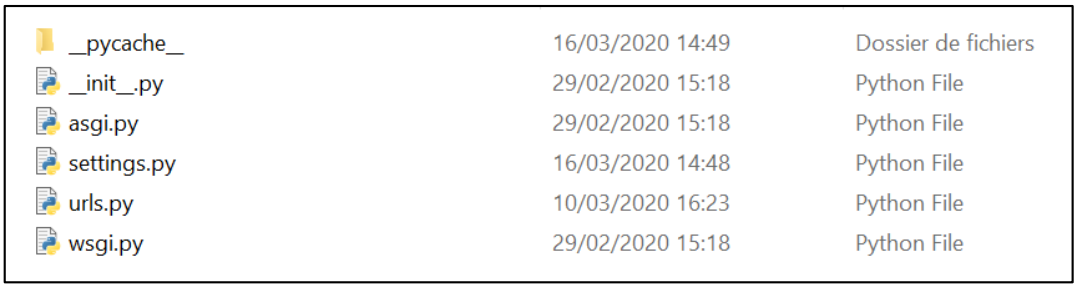
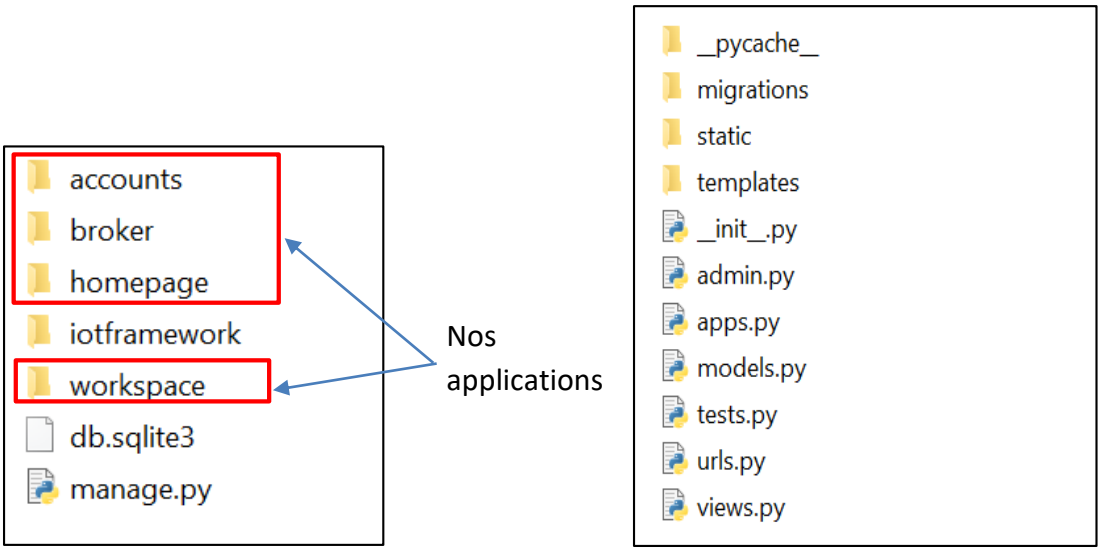


Figure 2.9 : Fichiers fondamentales d'un projet Django

À l'intérieur de chaque projet nous trouvons ce que nous appelons des applications, et nous devons les créer par l'utilisation de l'invite de commande. Nous expliquons quelles sont ces applications ; Par exemple dans notre cas nous avons besoin de plusieurs applications [homepage application, account application, workspace application, Broker application] dans chaque application il existe également des fichiers de base, où nous pouvons ajouter des URLs, et d'autres fichiers python dont lesquelles nous ajoutons des Templates des classes, des fonctions, et le fichier view.py de visualisation.



La figure ci-dessous résume la hiérarchie des fichiers Django de notre projet :

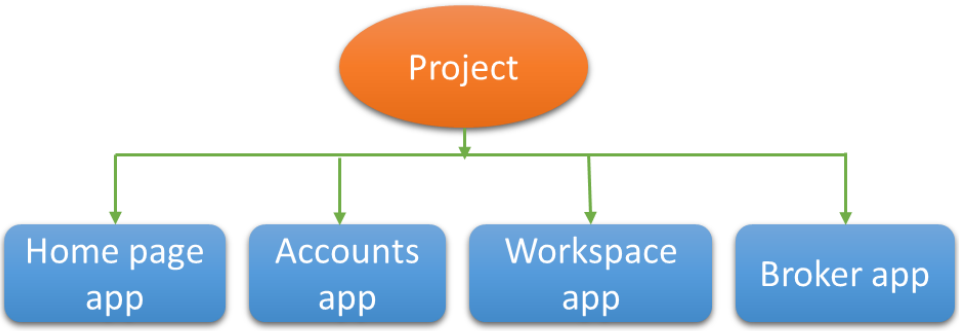


Figure 2.10 : Schéma explicatif des applications du projet

Chaque application a une URL personnalisée et unique, sinon il y aura chevauchement dans des adresses, et elle a le fichier ‘’models.py’’ associé à la base de donnée où nous ajoutons les paramètres de plusieurs valeurs.

Nous expliquons les applications et leurs rôles individuellement :

2.3.1 Application de page d’accueil

C’est la première page affichée lorsque les utilisateurs consultent la plateforme, l’url est reliée directement avec l’adresse IP du site. Suivons les règles de Django nous créons une fonction dans le fichier views.py pour appeler le fichier HTML/CSS représentant notre home page. Dans cette page nous mettons trois directions que l’utilisateur peut prendre. Dans le cas où l’utilisateur visite la plateforme pour la première fois, il va s’inscrire en cliquant sur le bouton Sign-up. Si l’utilisateur est déjà inscrit, il doit alors cliquer sur le bouton log-in pour être dirigé vers son interface personnelle, et ensuite se diriger vers sa page de travail en cliquant sur (Get started).

2.3.2 Application de compte

Cette application est responsable de l’inscription et de la confirmation du compte. Ici nous programmons le système d’authentification d’utilisateur, pour cela chaque utilisateur doit remplir ces informations pour avoir le droit d’accès à la plateforme comme (username, password, email, …ect). Nous utilisons une application **USER** d’authentification qui existe dans la bibliothèque de Django, ensuite nous utilisons un système de formulaire et un fichier HTML pour construire une interface simple et visible par l’utilisateur, la figure ci-dessous exprime la démarche de notre programme.

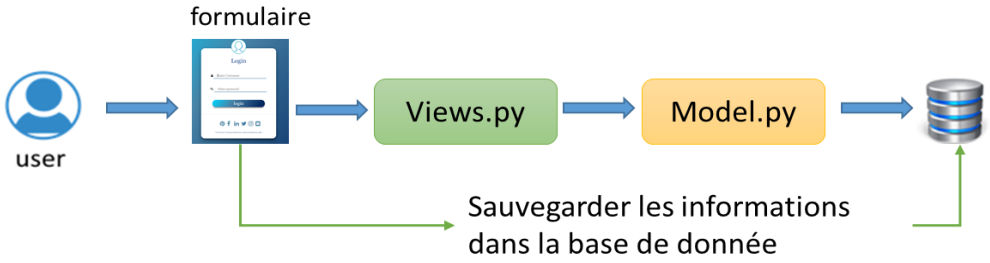


Figure 2.11 : Sauvegarde des informations dans la BDD

Nous faisons la même chose pour le programme d'identification sauf que, dans l'inscription les informations entrées par l'utilisateur vont être sauvegardées dans la base de données, dans ce cas l'utilisateur confirme l'existence de ces informations. La figure ci-dessous explique le fonctionnement du système d'identification.

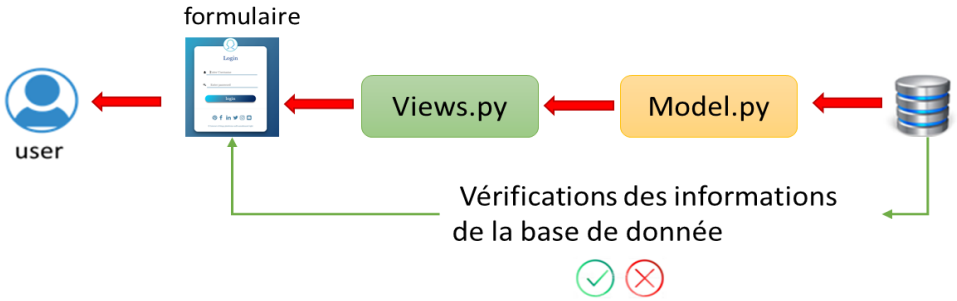


Figure 2.12 : Confirmation des informations de la BDD

2.3.3 Application d'espace de travail

Dans cette application nous programmons l'interface de travail où l'utilisateur commence à créer ses applications. Dans une application l'utilisateur peut ajouter un ensemble de capteurs et d'actionneurs qu'il souhaite contrôler. Ces applications vont être classées dans un tableau, nous disposons d'un bouton qui permet de diriger l'utilisateur vers une autre page de création d'application. Chaque application est identifiée par un ensemble d'informations comme un nom et un identificateur ID unique, ces données vont être enregistrées dans la base de données.

Concernant l'identifiant unique nous utilisons dans notre programme une option de UUID, qui permet de générer un code unique avec 32 caractères (lettre ou chiffre), mais dans notre cas nous utilisons 11 caractères qui est suffisant pour notre projet. Le code

est généré automatiquement lorsque l'utilisateur ajoute un nom à son application. Ensuite il va être dirigé vers une autre page pour ajouter des capteurs et des actionneurs de son application.

Au niveau de cette page, un formulaire permet à l'utilisateur de sélectionner une application parmi celles qui existe dans la BDD, puis d'ajouter des capteurs en les choisissant parmi les types de capteurs fournis. La même opération est faite pour les actionneurs. Les types son standardisés par la communauté des IoT.

L'utilisateur doit sélectionner un topic pour chaque capteur et actionneur afin d'interagir avec le broker MQTT.

Finalement, après enregistrement de ces informations, l'utilisateur est redirigé vers la première page du workspace. A ce niveau, nous remarquons que l'application est définie par un nom et un ID et trois boutons sont fournis, un pour la suppression, l'autre pour la modification et le dernier pour l'accès à la liste des Act/Cap. L'utilisateur a la possibilité d'ajouter d'autres capteurs et actionneurs, de les modifier et de les supprimer. La figure suivante résume le fonctionnement de cette application :

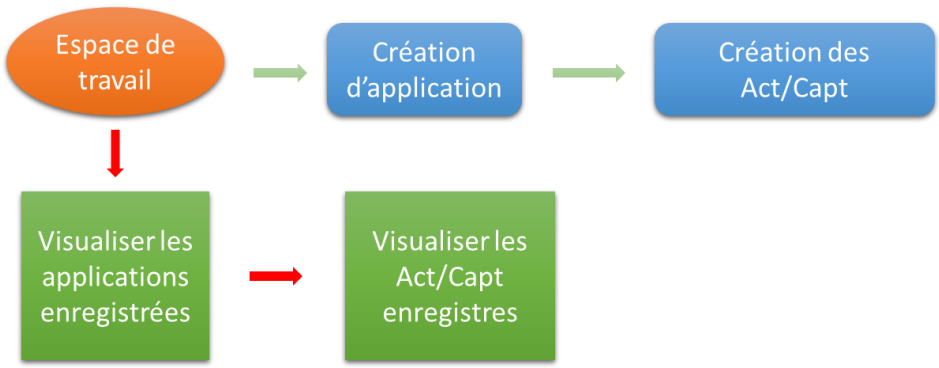


Figure 2.13 : Les étapes de navigation dans la plateforme

2.3.4 Application du broker

Cette application est responsable de gérer les données reçues à partir du Broker MQTT (serveur mosqitto), ceci grâce à une librairie Paho client implémenté dans la plateforme. L'application permet de générer les informations nécessaires pour la

communication avec notre table de la base de données [Id, auto_Id, Created_date, Topic, pub_info, palyoad, State]. Pour visualiser les informations du topic il suffit de cliquer sur détails.

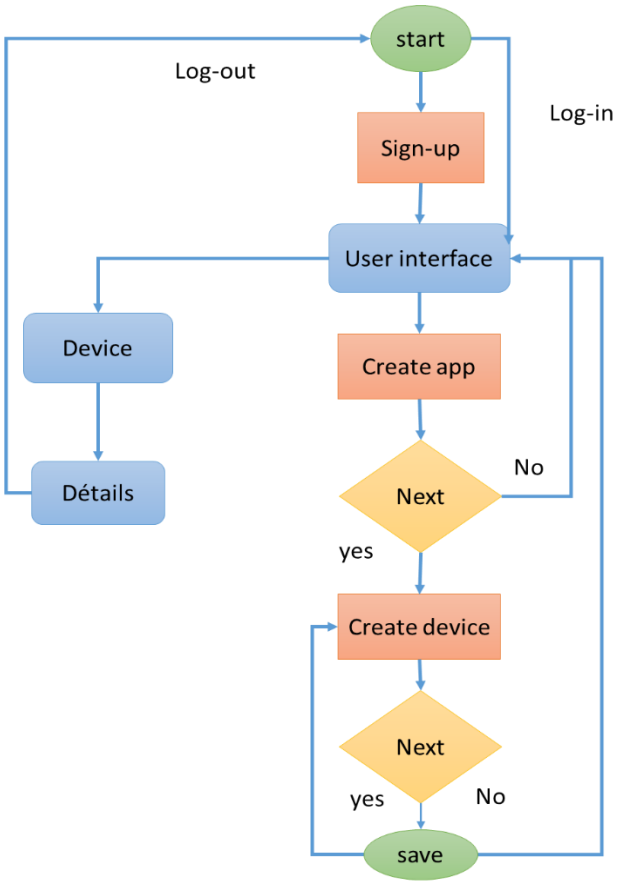


Figure 2.14 : Organigramme de fonctionnement de la plateforme

2.4 Base de données

Une base de données est une collection de données stockées dans des fichiers et accessibles à la demande pour plusieurs utilisateurs et des besoins divers.

L'utilisateur dispose de moyens très élaborés pour effectuer un large éventail d'opérations : créations de nouveaux fichiers, consultation, ajout, modification ou suppression de données, calculs et éditions de résultats, etc…

Les bases de données permettent d'enregistrer, de stocker, de ranger des données de façon organisée et hiérarchisée. SQL est le langage qui permet de manipuler les bases de données. Les SGBD (Systèmes de Gestion de Base de Données) sont les programmes qui nous permettent de gérer nos données directement sans utiliser de script PHP, les

plus connus sont : MySQL : libre et gratuit, c'est probablement le SGBD le plus connu. Nous avons aussi Wamp qui est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement les bases de données. [46] [47] [48]

Dans notre programme, nous utiliserons la base de données sqlit3 qui est intégrée de base avec Django car nous allons tester le réseau en local, et comme les données du broker ont un volume important, nous utilisons WAMP pour l'archivage des données capteurs.

D'après la définition de la BDD, nous illustrons des exemples de la sauvegarde des données dans notre BDD. Les tableaux ci-dessous représentent respectivement, les informations d'inscription de l'utilisateur [username, password], les applications de l'utilisateur [Application name, ID], les Act/Cap [application, sensor name, sensor type, actuator name, actuator type, sub_topic] et les données de broker [ID, auto_Id, Topic, info, payload, created_date, State].

Id	Username	Password	Password confirmation
1	User 01	Sqfdsfg5688thf	Sqfdsfg5688thf

Tableau 2.1 : Données d'inscription dans la BDD

Id	Application Name	UUID
1	App1	2y57uirc9ol

Tableau 2.2 : Données de création d'application dans la BDD

Id	Application	Sensor name	Sensor type	Actuator name	Actuator type	Sub_topic
1	App1	Seneor001	Temperature	Actuator001	Motors	Labo/Temperature

Tableau 2.3 : Données de création des Act/Cap dans la BDD

Device_id	Auto_id	Topic	Pub _info	payload	Created_date	State
921	12	Labo/temperature	18	Topic : labo/temperature , info : 18	03/09/2020	1

Tableau II. 1 : Données de client MQTT

Le schéma ci-dessous montre l'hiérarchique de notre BDD :

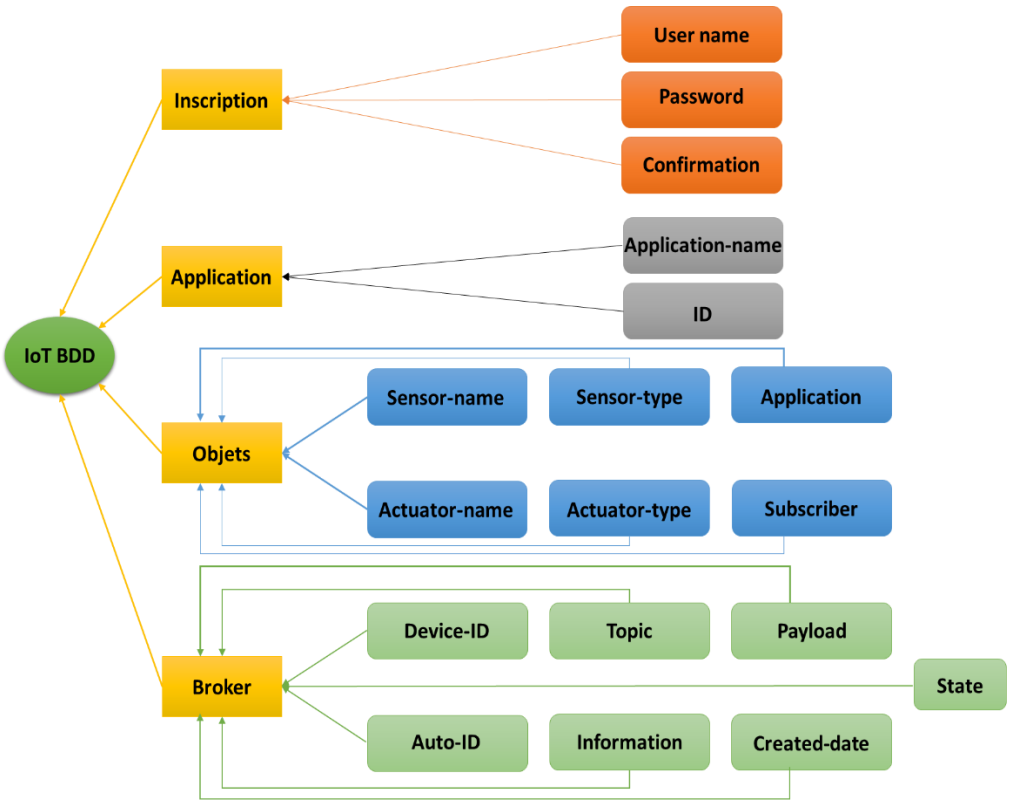


Figure 2.15 : Hiérarchie de la BDD de la plateforme

2.5 Traitement de données en arrière-plan

Dans notre projet, les données circulent entre les objets et la plateforme grâce au protocole MQTT, pour cela nous allons installer le broker Mosquitto qui prend en charge la circulation des données entre les clients (sub/pub). Les clients du broker sont générés par Paho MQTT Client (présenté dans 2.5.3).

Étant donné que notre plate-forme est basée sur Django et que Django ne fonctionne qu'avec le protocole HTTP, nous devons trouver une solution pour gérer les données MQTT, donc Django fournit l'application Channels qui prend en charge d'autres protocoles et services.

2.5.1 MQTT (Message Queuing Telemetry Transport)

MQTT est le protocole de communication de messagerie basé sur TCP, pour faire communiquer des machines. Il permet concrètement aux appareils d'envoyer des informations sur un sujet donné à un serveur qui fonctionne comme un broker (un courtier en français) de messages. Le broker pousse ces informations vers les clients qui se sont précédemment abonnés. Pour l'utilisateur, un sujet ressemble à un chemin hiérarchique. Les clients peuvent s'abonner à un niveau spécifique de la hiérarchie d'un sujet ou à plusieurs niveaux. [49]

a Principe de fonctionnement

MQTT fonctionne selon le modèle "Publish/Subscribe", ce modèle dissocie plusieurs éléments essentiels qui sont [50] :

- **Broker (courtier) :** est le serveur qui distribue les informations aux clients intéressés connectés au serveur. MQTT Broker est responsable de la réception de tous les messages, du filtrage, de la prise de décision et de l'envoi des messages aux clients abonnés. Il existe plusieurs Broker que nous pouvons utiliser, dans notre projet nous utilisons le broker Mosquitto.
- **Client :** est l'appareil qui se connecte au broker pour envoyer ou recevoir des informations.
- **Publish :** est le processus qu'un appareil effectue pour envoyer son message au broker.
- **Subscribe :** là où un appareil récupère un message du broker.
- **Topic :** un sujet, qui est l'endroit où un appareil souhaite placer ou récupérer un message.
- **Message :** le message qui correspond aux données qu'un appareil reçoit «lors de l'abonnement» d'un sujet ou envoyé «lors de la publication» à un sujet.

Plusieurs clients se connectent à un serveur unique (broker), les clients publient les informations qui sont envoyer par un canal (topic), ce dernier peut avoir une hiérarchie qui permet de sélectionner finement les informations que nous désirons. Ces messages peuvent être lus par les abonnées (les clients qui vont souscrire dans les mêmes topics).

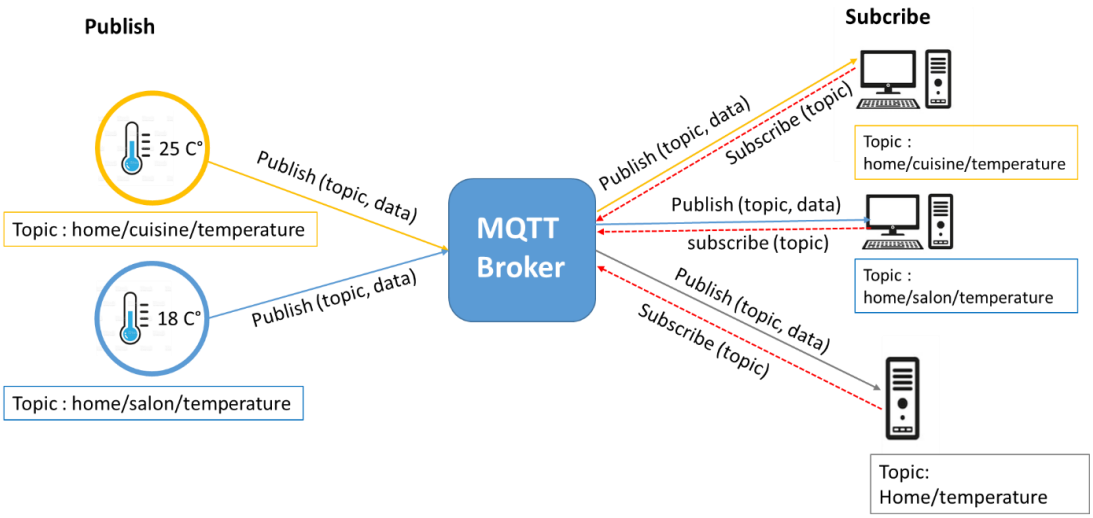


Figure 2.16 : Principe de fonctionnement du MQTT

b Avantages du MQTT

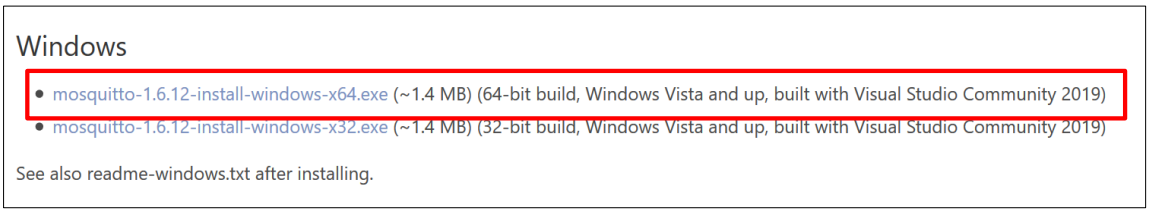
- ✓ Protocole ouvert, simple, léger et facile à mettre en œuvre.
- ✓ Idéal pour la transmission de données en utilisant une très faible bande passante.
- ✓ Adapté au réseau sans fil.
- ✓ Faible consommation énergétique.
- ✓ Très rapide.

2.5.2 MQTT Broker " Mosqquitto"

Mosquitto est un broker de messages open source qui implémente le protocole MQTT. Il est léger et peut être utilisé sur tous les appareils, depuis une carte simple à faible consommation comme Arduino, ESP8266, jusqu'aux ordinateurs et serveurs complets mais aussi sur presque tous les systèmes d'exploitation (macOS, Windows, Linux…). [51]

- **Installation**

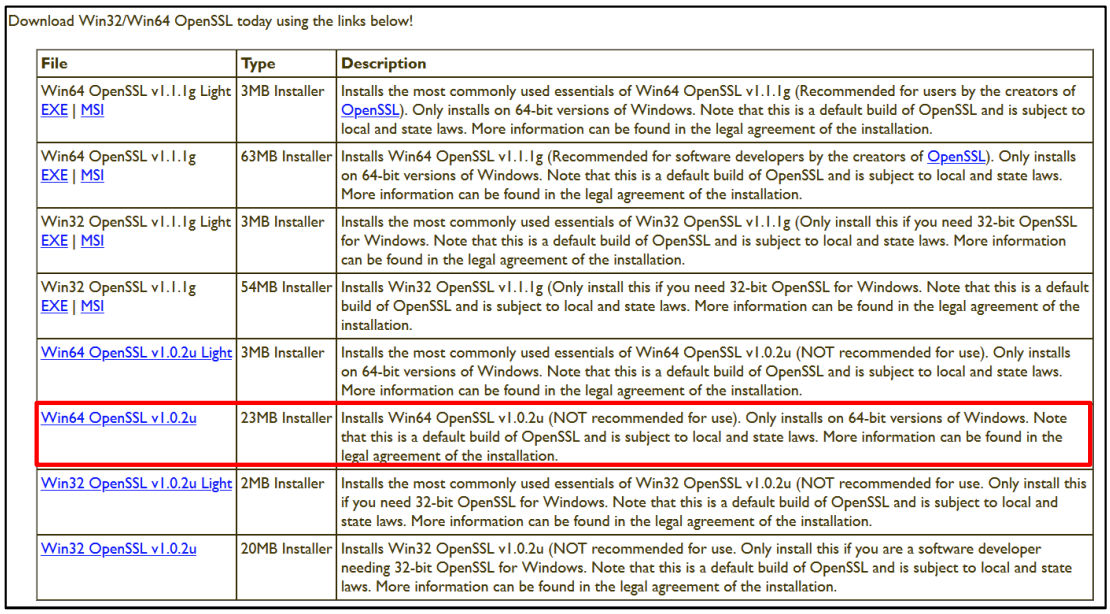
Étape 1 : Tout d'abord nous allons télécharger le paquet d'installation mosqitto qui se trouve dans son propre site https://mosquitto.org/download/, ainsi que le paquet Openssl de http://slproweb.com/products/Win32Ope**.**



Windows

- mosquitto-1.6.12-install-windows-x64.exe (~1.4 MB) (64-bit build, Windows Vista and up, built with Visual Studio Community 2019)
- mosquitto-1.6.12-install-windows-x32.exe (~1.4 MB) (32-bit build, Windows Vista and up, built with Visual Studio Community 2019)

See also readme-windows.txt after installing.



Download Win32/Win64 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v1.1.1g Light EXE | MSI	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.1g (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.1g EXE | MSI	63MB Installer	Installs Win64 OpenSSL v1.1.1g (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1g Light EXE | MSI	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.1g (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1g EXE | MSI	54MB Installer	Installs Win32 OpenSSL v1.1.1g (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.0.2u Light	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.0.2u (NOT recommended for use). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.0.2u	23MB Installer	Installs Win64 OpenSSL v1.0.2u (NOT recommended for use). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.0.2u Light	2MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.0.2u (NOT recommended for use. Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.0.2u	20MB Installer	Installs Win32 OpenSSL v1.0.2u (NOT recommended for use. Only install this if you are a software developer needing 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

Étape 2 : Après l'exécution des deux fichiers téléchargés, nous allons copier le fichier openssl64\bin vers C:\Program Files(x86)\mosquitto, ensuite nous allons copier les deux fichiers **libeay64.dll** et **ssleay64.dll** vers C:\Program Files(x86)\mosquitto.

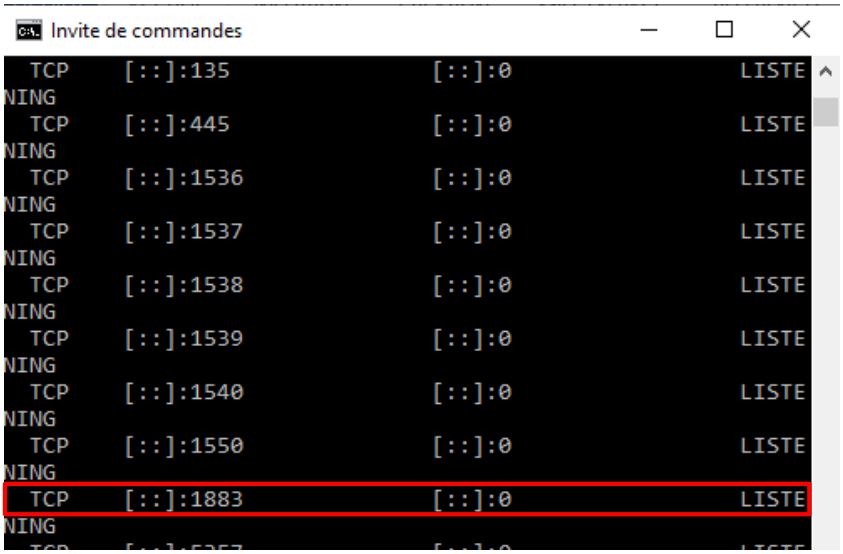
Étape 3 : Nous avons besoin d'un autre fichier pathreadVC2.dll à télécharger à partir de ftp://sources.redhat.com/pub/pthreads-win32/dll-latest/dll/x86/, puis nous allons le transférer vers C:\Program Files(x86)\mosquitto.

Index of ftp://sources.redhat.com/pub/pthreads-win32/dll-latest/dll/x86/

Up to higher level directory

Name	Size	Last Modified
File: md5.sum	1 KB	2/5/2015 1:00:00 AM
File: pthreadGC2.dll	118 KB	5/27/2012 1:00:00 AM
File: pthreadGCE2.dll	120 KB	5/27/2012 1:00:00 AM
File: pthreadVC2.dll	55 KB	5/27/2012 1:00:00 AM
File: pthreadVCE2.dll	61 KB	5/27/2012 1:00:00 AM
File: pthreadVSE2.dll	56 KB	5/27/2012 1:00:00 AM
File: sha512.sum	1 KB	2/5/2015 1:00:00 AM

Étape 4 : pour tester le broker mosquitto, nous allons ouvrir trois invites de commandes. Dans la 1ère fenêtre nous exécutons le broker mosquitto. Par la commande « netstat » nous remarquons que le port 1883 est ouvert.



Dans la 2ème fenêtre nous allons tester le client mosquitto publisher par le fichier mosquitto_pub et la commande –t 'home/temperature' qui signifie le topic, et la commande –m '24' qui signifie le message, le publieur va envoyer un message ''24'' sur le topic ''home/temperature''.

```
F:\Mosquitto>mosquitto_pub -t 'home/temperature' -m '24'
```

La 3ème fenêtre où nous allons souscrire au même topic 'home/temperature' pour recevoir le message publié.

```
F:\Mosquitto>mosquitto_sub -v -t 'home/temperature'
'home/temperature' '24'
```

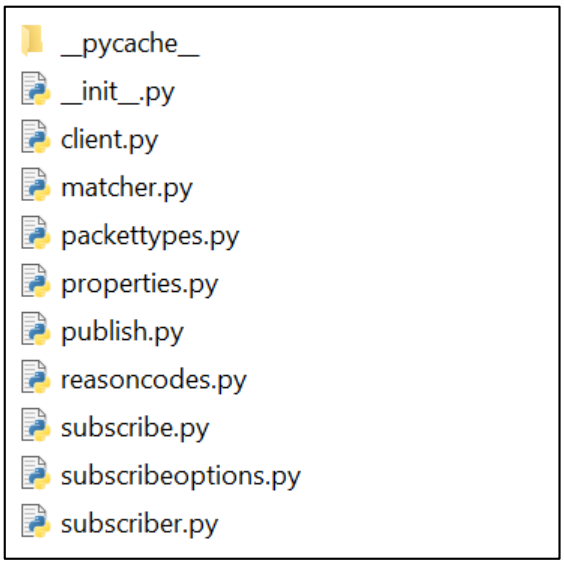
2.5.3 MQTT client Paho

Eclipse Paho est une implémentation open-source du client MQTT, disponible dans divers langages de programmation actuellement, elle contient des implémentations en C, Java, JavaScript, Python (contribution du projet mosquitto). La bibliothèque implémente une classe client qui peut être utilisée pour ajouter la prise en charge de MQTT à notre programme Python, les utilisateurs peuvent avec Paho envoyé des messages d'abonnement et de publication à un broker MQTT tel que le broker mosquitto. Il fournit également des fonctions d'aide pour rendre la publication de messages uniques et extrêmement facile. [52]

- **Installation**

Nous allons installer Paho Client dans notre projet par le paquet d'installation pip de python, en utilisant la commande **''pip install paho-mqtt''**.

La figure suivante montre les fichiers fondamentaux de Paho Client.



Afin de tester notre réseau IoT en local, nous allons créer de faux capteurs par un script python qui va générer des valeurs de façon aléatoires et les envoyer vers le broker mosquitto installé dans notre machine. Ces valeurs vont être renvoyé vers MQTT client ''subscriber''.

Le schéma illustré en dessous explique le fonctionnement de l'architecture Sub/Pub de notre réseau IoT.

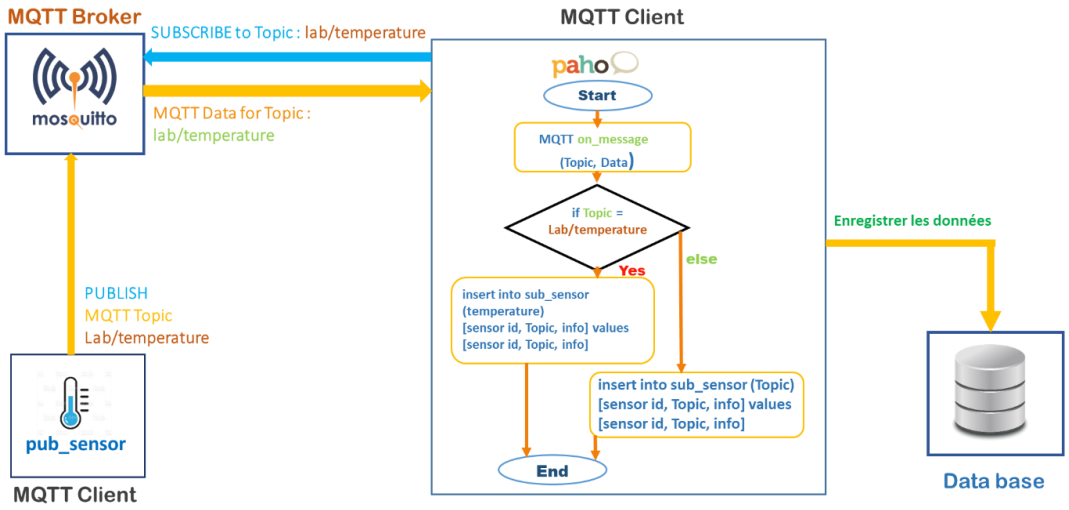


Figure 2.17 : Schéma Sub/Pub du réseau IoT

2.6 Channels

2.6.1 Introduction

Channels permet au Django de pouvoir utiliser du code asynchrone sous et via le cœur synchrone de Django, permettant aux projets Django de gérer non seulement HTTP, mais aussi les protocoles qui nécessitent des connexions de longue durée aussi WebSockets, MQTT, chatbots, radio amateur, etc. Il le fait tout en préservant la nature synchrone et facile à utiliser de Django, nous permettant de choisir comment écrire notre code synchrone dans un style comme les vues Django, entièrement asynchrone ou un mélange des deux. En plus de cela, il fournit des intégrations avec le système d'authentification de Django, le système de session, etc, ce qui facilite plus que jamais l'extension de notre Projet HTTP uniquement vers d'autres protocoles.

Il regroupe également cette architecture événementielle avec des couches de canaux, un système qui nous permet de communiquer facilement entre les processus et séparer notre projet en différents processus. [53]

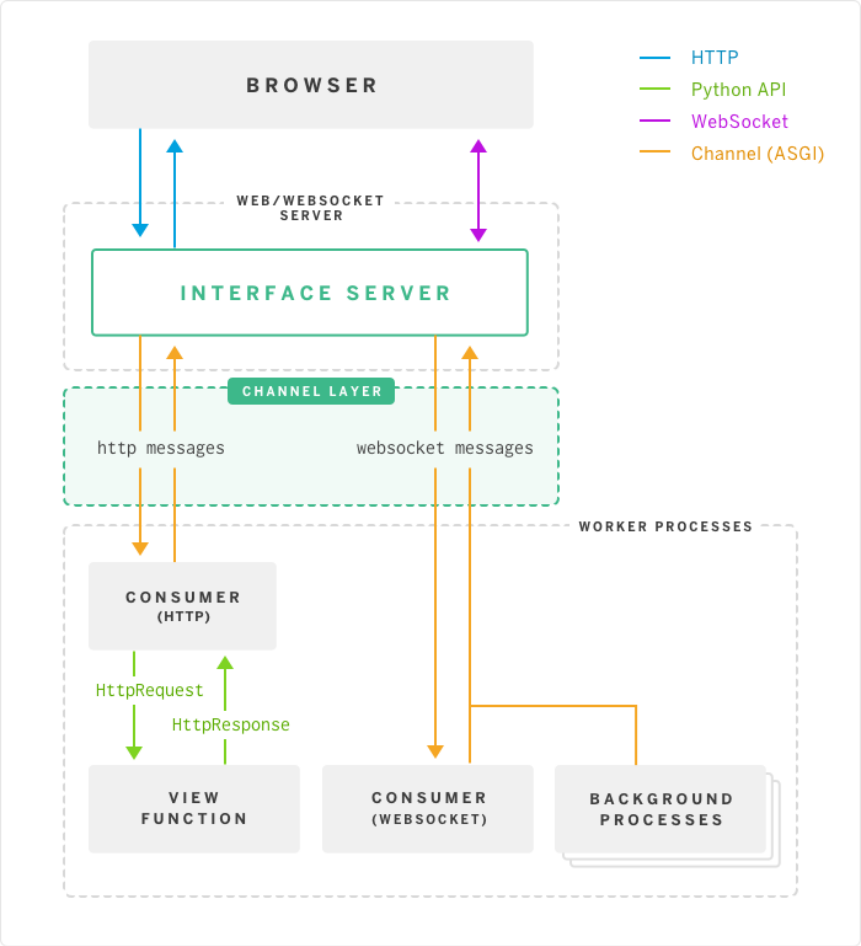


Figure 2.18 : Schéma explicatif de rôle des Channels dans Django plateforme [53]

2.6.2 Installation

Pour intégrer Channels dans notre plateforme nous allons avoir besoin d'établir les configurations suivantes :

1. Installer Channels par la commande, il suffit de lancer cette commande et l'installation se fait automatiquement :

   python -m pip install -U channels

2. Il faut ajouter cette application avec les autres applications de projet dans le fichier settings.py :

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    #my app :
    'workspace',
    'homepage',
    'accounts',
    'broker',
    'channels',
]
```

3. Créer un fichier routing.py et ajouter les commandes de routage des fichiers Channels, et dans ce fichier nous lançons le programme de consumers.py.

```
from channels.routing import ProtocolTypeRouter, ChannelNameRouter
from broker.consumers import MqttConsumer

application = ProtocolTypeRouter({
        "channel": ChannelNameRouter({
          "mqtt.sub": MqttConsumer
    }),
})
```

4. Appliquer des modifications sur le fichier asgi.py et ajouter l'application asgi dans le fichier setting.py :

```
import os
import django
from django.core.asgi import get_asgi_application
from channels.layers import get_channel_layer

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'iotframework.settings')
django.setup()

application = get_asgi_application()
channel_layer = get_channel_layer()
```

```
WSGI_APPLICATION = 'iotframework.wsgi.application'
ASGI_APPLICATION = "iotframework.routing.application"
```

La plate-forme principale de déploiement Django est WSGI, le standard Python pour les serveurs et applications Web.

La commande de gestion startproject de Django définit par défaut une configuration WSGI, que nous pouvons ensuite adapter aux besoins de notre projet, cette configuration est alors utilisable par tout serveur d'applications se conformant au standard WSGI.

En plus de WSGI, Django peut aussi être déployé avec ASGI, le standard Python émergent pour les serveurs et applications asynchrones.

Après avoir intégré Channels dans notre plateforme, cette dernière fonctionne maintenant avec asgi_application. Les serveurs ASGI acceptent généralement le chemin d'accès à l'objet d'application comme Channels pour la plupart des projets Django.

ASGI est le nom de la spécification de serveur asynchrone sur laquelle les canaux sont construits. Comme WSGI, c'est conçu pour nous permettre de choisir entre différents serveurs et frameworks plutôt que d'être verrouillé dans les canaux.

5. Configurer ''Channels layers'' dans le fichier settings :

```
CHANNEL_LAYERS = {
    "default": {
        "BACKEND": "channels_redis.core.RedisChannelLayer",
        "CONFIG": {
            "hosts": [("localhost", 6379)],
        },
    },
}
```

Ensuite, il suffit de créer le fichier **consumers.py** et rédiger le programme nécessaire qui sert à gérer les données MQTT.

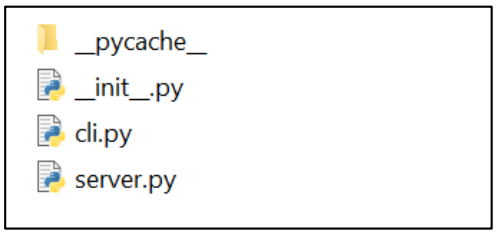
Asgimqtt est l'interface MQTT pour asgi, nous devons installer cette dernière là où se trouve le client MQTT ''l'abonnée'' qui récupère les données du broker mosquitto, pour l'installer, il suffit d'exécuter la commande suivante :

Py pip install asgimqtt

Pour lancer ce dernier il nous faut exécuter la commande :

asgimqtt --host localhost --port 1883 Notreprojet.asgi:channel_layer

Les paquets installés sont les suivantes :



- **Publieur**

Avec un script Python nous allons programmer un capteur virtuel qui va générer des valeurs aléatoires proche des valeurs réelles de l'environnement, et le connecter avec mosquitto broker, ensuite envoyer ces valeurs à ce dernier (Annexe 1).

Nous avons eu des difficultés avec l'utilisation de l'interface ASGI MQTT pour récupérer les données du broker.

L'interface ASGI MQTT n'arrive pas à s'abonner dans un topic afin de récupérer les données. Peut-être qu'il existe un conflit de version des paquets dans Django pour réaliser cette tâche.

Compte tenu du temps qui nous est imparti pour la finalisation du mémoire, Nous avons préféré trouver une autre solution. En effet nous avons proposé pour cela une implémentation de Paho client MQTT en passant par les Websocket (en utilisant Javascript).

2.7 MQTT avec JavaScript

Le client JavaScript Paho est une bibliothèque client basée sur un navigateur MQTT. Pour se connecter et s'abonner à MQTT à l'aide de JavaScript, MQTT doit être configuré pour fonctionner sur des Websockets.

Mosquitto n'active pas les Websockets par défaut, alors nous avons activé le port du Websocket approprié 9001.

```
listener 1883
protocol mqtt

listener 9001
protocol websockets
```

- **Abonné**

Pour créer un client MQTT, il faut télécharger le fichier **paho_mqtt.js** qui contient toutes les fonctions et outils nécessaires permettant d'implémenter le client Paho en code JavaScript, il suffit donc d'appeler ce dernier dans notre client. Ensuite, nous codons trois fonctions, la première pour communiquer avec le broker Mosquitto via le protocole Websocket sur le port 9001, afin qu'il puisse s'abonner à un topic. La deuxième fonction est la déconnexion, et la dernière concerne les messages reçus du broker, où l'utilisateur peut traiter les données, les mettre à jour, les sauvegarder et les visualiser.

Pour l'archivage des données reçues ; nous utilisons le paquet **jquery (jquery -3.5.1.js),** ce paquet permet une communication interactive avec la base de donnée (Mysql) grâce à **Ajax**. Ajax fait office d'intermédiaire entre les messages JavaScript et le fichier de sauvegarde **". PHP"** et de façon asynchrone, le script de l'abonné se trouve dans l'annexe 2.

Figure 2.19 : Sub/Pub entre Websocket et MQTT

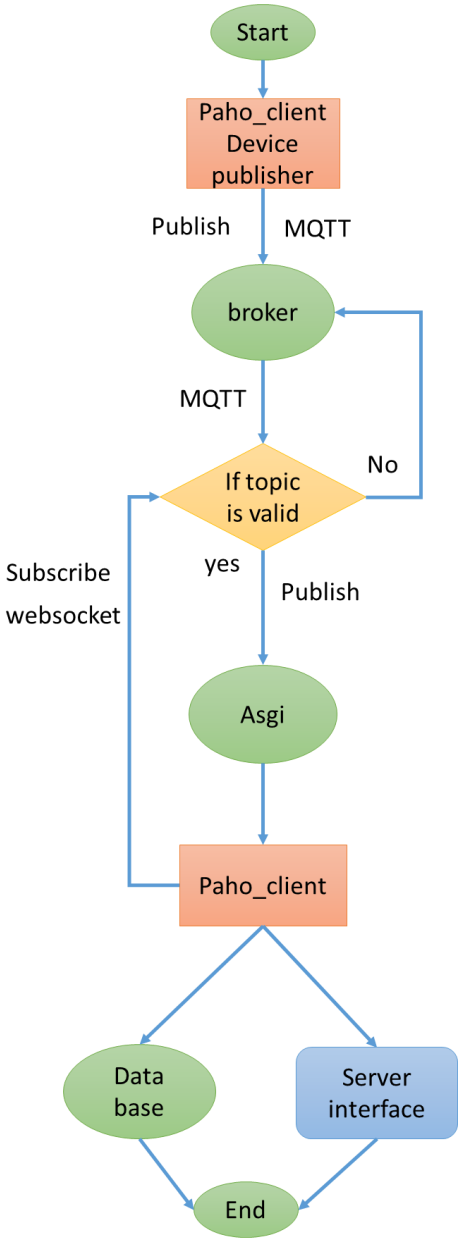


Figure 2.20 : Organigramme de circulation des données MQTT

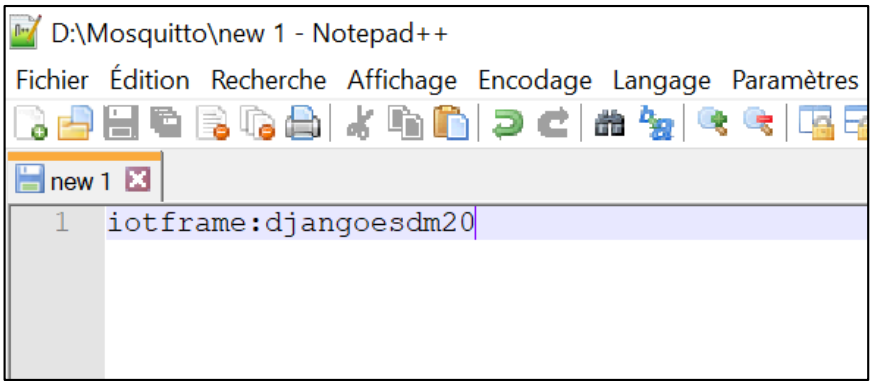
2.8 Système Sécurité de la plateforme

Dans ce projet nous utilisons deux systèmes de sécurité, le premier concerne la protection de l'interface des utilisateurs dans la plateforme avec le système d'authentification (sign-up and log in). Et l'autre pour protéger les données MQTT du broker par le système d'authentification de broker mosquitto.

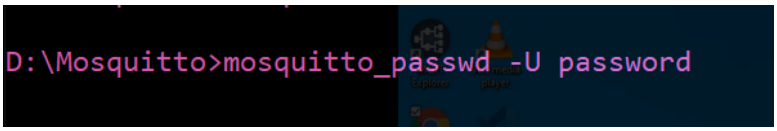
Les étapes de configuration sont les suivantes :

1. Créer un fichier texte et écrire les données dans cet ordre

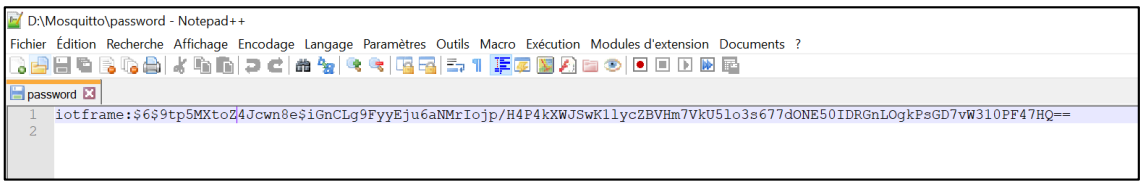
Username : password, puis enregistrer le fichier dans le même emplacement où se trouve mosquitto_passwd dans le paquet de l'installation de mosquitto.



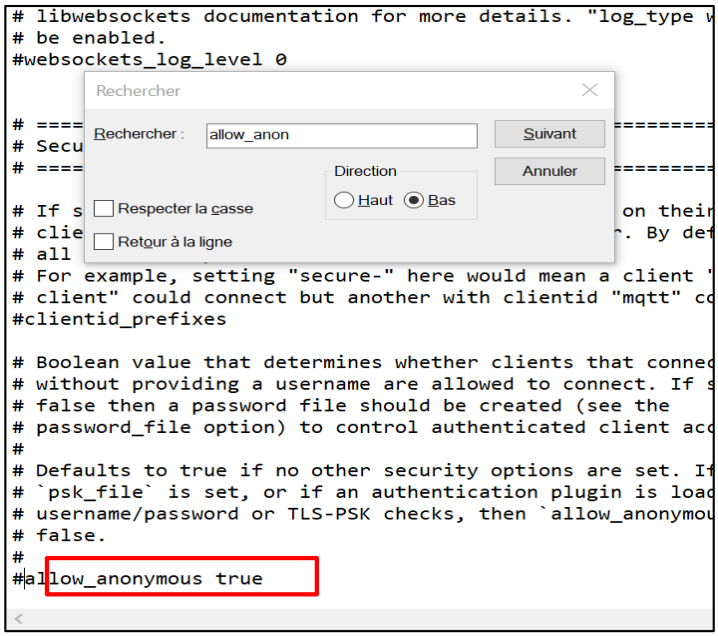
2. Dans une fenêtre de l'invite commande, nous lançons la commande suivante :



Cette commande va encoder le mot de passe de notre ficher password.



3. Modifier le fichier mosquitto.conf, pour la règle « allow_anonymous » qui autorise l'accès sans mot de passe comme suit :




```
# Defaults to true if no other security options are set. If `password_file` or
# `psk_file` is set, or if an authentication plugin is loaded which implements
# username/password or TLS-PSK checks, then `allow_anonymous` defaults to
# false.
#
allow_anonymous false
```

Il faut ajouter le chemin de notre fichier password là où se trouve password_file.

```
# Control access to the broker using a password file. This file can be
# generated using the mosquitto_passwd utility. If TLS support is not compiled
# into mosquitto (it is recommended that TLS support should be included) then
# plain text passwords are used, in which case the file should be a text file
# with lines in the format:
# username:password
# The password (and colon) may be omitted if desired, although this
# offers very little in the way of security.
#
# See the TLS client require_certificate and use_identity_as_username options
# for alternative authentication options. If an auth_plugin is used as well as
# password_file, the auth_plugin check will be made first.
#password file
```

```
# See the TLS client require_certificate and use_identity_as_username options
# for alternative authentication options. If an auth_plugin is used as well as
# password_file, the auth_plugin check will be made first.
password_file D:\Mosquitto\password

# Access may also be controlled using a pre-shared-key file. This requires
```

Pour la publication sur mosquitto il faut lançer mosquitto_pub comme suit :

```
D:\Mosquitto>mosquitto_pub -h localhost -p 1883 -u iotfram -P password -t Labo/temperature -m 18

D:\Mosquitto>
```

Aucune erreur ou difficulté de connexion n'a été trouvée, le message est publié avec succé.

2.9 Conclusion

Dans ce chapitre nous avons présenté l'architecture physique et logique de notre réseau IoT. Nous avons donné une description détaillée de son architecture et son fonctionnement. Nous avons ensuite abordé la partie logicielle de la plateforme tout en expliquant les étapes d'élaboration pas à pas.

Et nous avons aussi présenté les outils et langages de programmation utilisés dans la réalisation de la plateforme.

Nous avons présenté le serveur Mosquitto broker et les clients Paho MQTT utilisé dans notre projet, ainsi que le protocole MQTT responsable de l'envoie des données entre les deux.

Nous avons terminé le chapitre par la présentation de la solution responsable du traitement et de récupération des données MQTT reçues à partir broker MQTT.

Chapitre 3 Simulation de la plateforme et résultats

Dans le présent chapitre nous détaillons les tâches que nous avons pu réaliser ainsi que les différents tests de validation. Nous allons introduire tout le système en état de fonctionnement et la description et l'illustration des interfaces en précisant leurs rôles.

Et en dernier, nous discuterons les conditions d'application du système et les problèmes rencontrés.

3.1 Interface d'administrateur

C'est une interface super utilisateur intégrée à Django, nous la créons avec la commande suivante, et lui donnons un nom d'utilisateur et un mot de passe :

Python manage.py create superuser

Cette interface permet à l'administrateur de visionner toutes les données enregistrées dans la base de données sqlite3 ainsi que la possibilité de modifier, supprimer ou ajouter des paramètres, la figure ci-dessous présente cette interface.

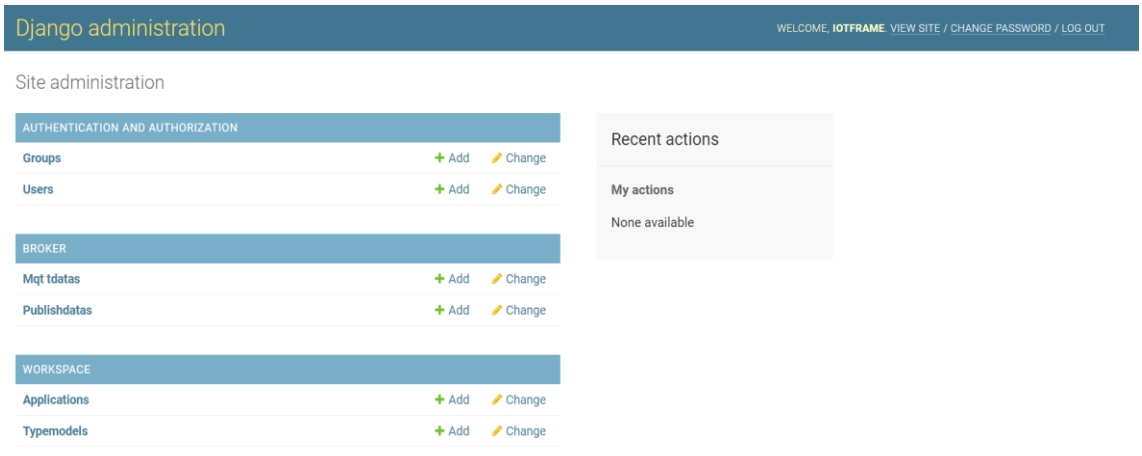


Figure 3.1 : Interface administrateur

3.2 Interfaces de la plateforme

Dans ce qui suit, nous présentons les différentes interfaces que nous avons réussie à mettre en place. Nous pouvons accéder au serveur Django en local avec l'adresse 127.0.0.1:8000. Tous d'abord il faut exécuter la commande : **python manage.py runserver** dans le terminal pour lancer la plateforme.

La plateforme que nous avons réalisée est divisée en plusieurs pages qui sont : la page d'accueil, page de connexion, page de création d'applications et page de visualisation.

3.2.1 Page d'accueil

La figure ci-dessous illustre la page d'accueil contenant le logo de notre plateforme ainsi que quelques informations relatives avec le système IoT et ses fonctionnalités.

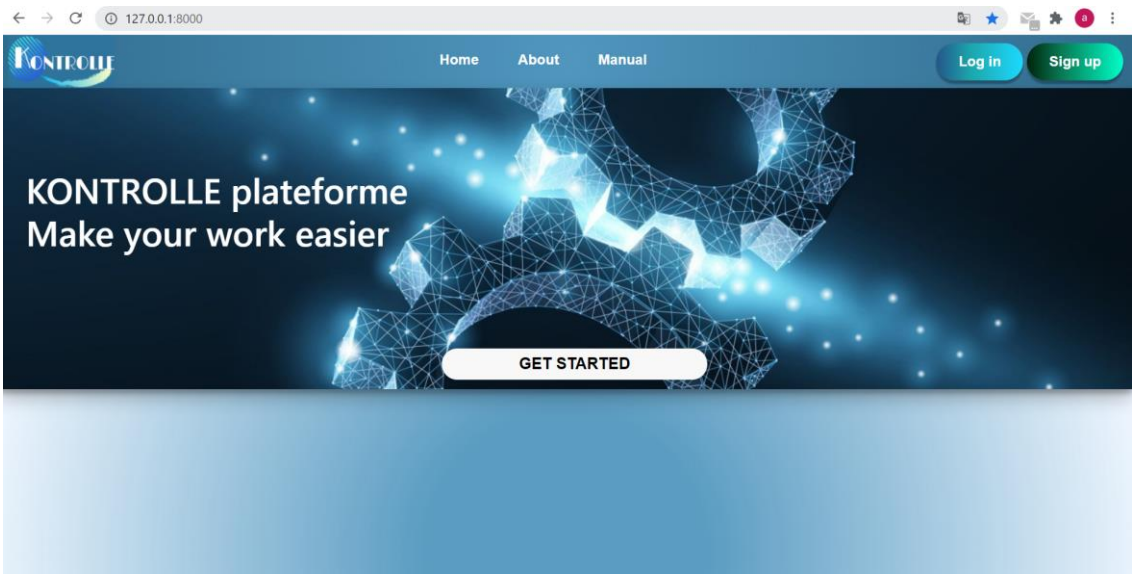


Figure 3.2 : Page d'accueil

3.2.2 Page de connexion

La figure 3.3 représente l'interface de connexion, chaque utilisateur qui utilise cette plateforme pour la première fois doit créer un compte. Pour cela il doit remplir un formulaire d'inscription qui permet de réserver un espace personnalisé et unique pour lui. S'il ne possède pas un compte il sera obligé à s'inscrire.

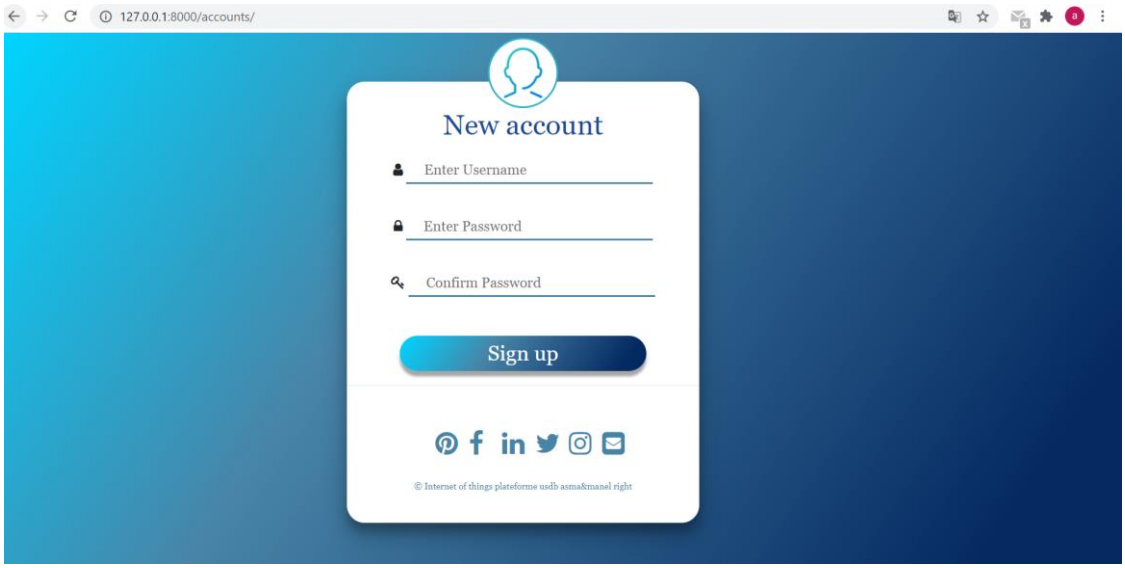


Figure 3.3 : Page de connexion

3.2.3 Page de création d'application

Après inscription, l'utilisateur va être dirigé vers une page de travail où il va ajouter des applications, visualiser les différentes informations, et contrôler son propre réseau.

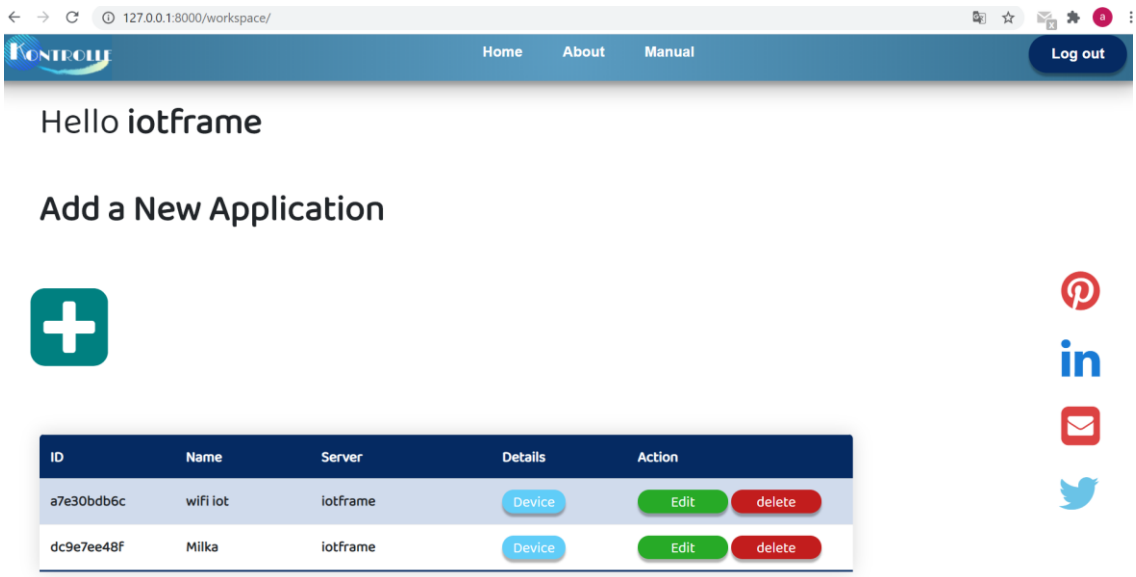


Figure 3.4 : Interface de visualisation d'application

Après cette étape, l'utilisateur va créer une application représentée par un réseau qui est formé de plusieurs actionneurs et capteurs, le choix des capteurs et actionneurs dépend de leurs applications, et des différents paramètres à contrôler. La création de

l'application est suivie par un formulaire qui doit être remplie par l'utilisateur (nom et identifiant), l'ID va être unique pour chaque application est généré automatiquement.

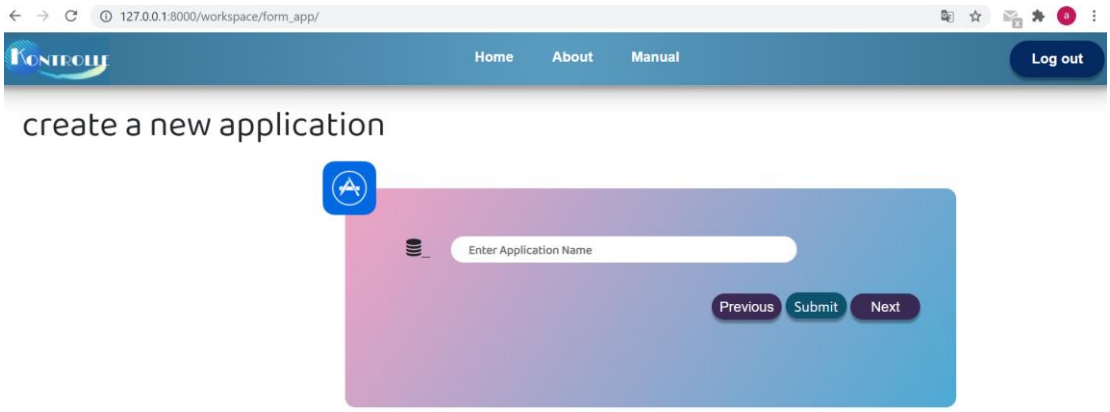


Figure 3.5 : Page de création d'application

Dans la dernière étape, l'utilisateur va créer les capteurs et les actionneurs selon les besoins de l'application et souscrire à un topic. Une fois terminé l'utilisateur va être redirigé vers l'espace personnel où il va trouver d'autres informations concernant les différentes applications déjà crées et à l'intérieur de chaque application, il peut mettre à jour les capteurs et les actionneurs, supprimer ou ajouter d'autres dispositifs et applications. L'utilisateur peut toujours fermer la section par le log out.

Figure 3.6 : Page de configuration

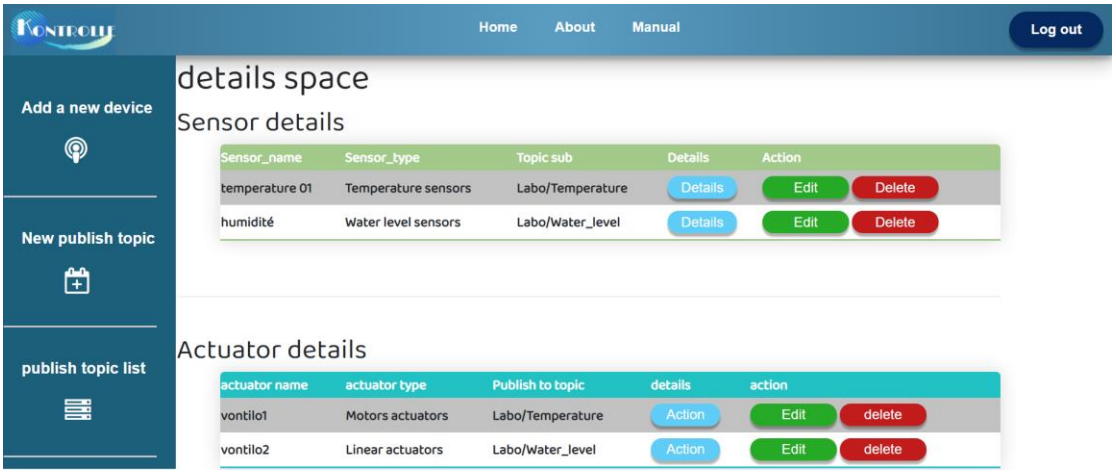


Figure 3.7 : Interface des capteurs et actionneurs

La figure ci-dessus montre l'interface des capteurs et actionneurs. Dans la barre de paramètres à gauche de la page, nous avons trois paramètres que l'utilisateur appliquera. Si l'utilisateur clique sur **Add a new device** il sera redirigé vers l'interface de

création des Act/Cap pour ajouter d'autres dispositifs. La seconde dans **New publish topic**, l'utilisateur peut ajouter un nouveau topic. Dans la dernière l'utilisateur peut afficher la liste des topics existants.

Nous avons également des tableaux de paramètre des Act/Cap, nous pouvons modifier ou supprimer un appareil en cliquant respectivement sur les boutons **Edit/delete**. Le bouton **Details** nous amènera à l'interface de visualisation des données.

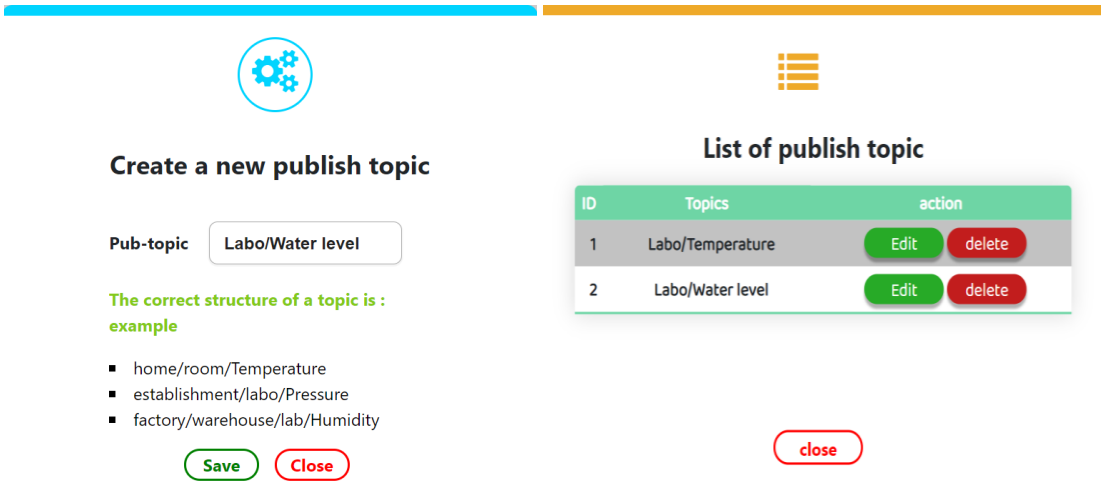


Figure 3.8 : Configuration des topics

3.3 Interface de visualisation

Selon notre architecture réseau au chapitre deux, nous avons effectué des tests sur un réseau local où nous avons installé le broker Mosquitto dans notre hôte et construit des capteurs virtuels avec des fichiers python qui génèrent des valeurs aléatoires.

Au début nous lançons le broker mosquitto par la commande suivante :

Mosquitto –v –c mosquitto.conf

```
D:\Mosquitto>mosquitto -v -c mosquitto.conf
1600019980: mosquitto version 1.6.10 starting
1600019980: Config loaded from mosquitto.conf.
1600019980: Opening ipv6 listen socket on port 1883.
1600019980: Opening ipv4 listen socket on port 1883.
1600019980: Opening websockets listen socket on port 9001.
```

Ensuite, nous lançons le publieur qui publiera les messages dans les deux topics (Labo/Temperature et Labo/water_level). La figure suivante montre ces informations.

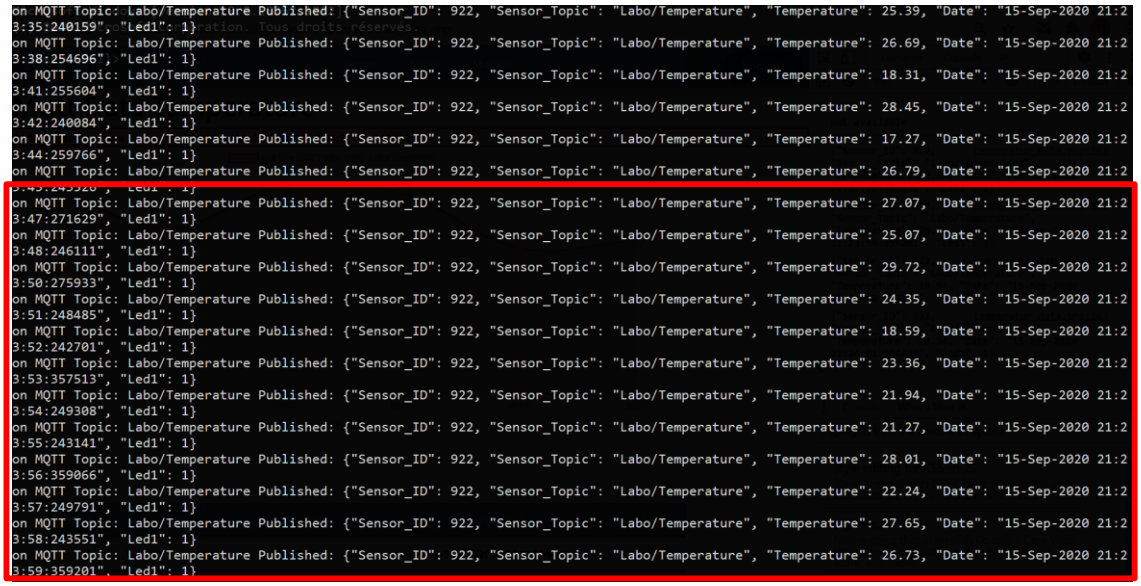


Figure 3.9 : Les données du publieur Labo/Temperature

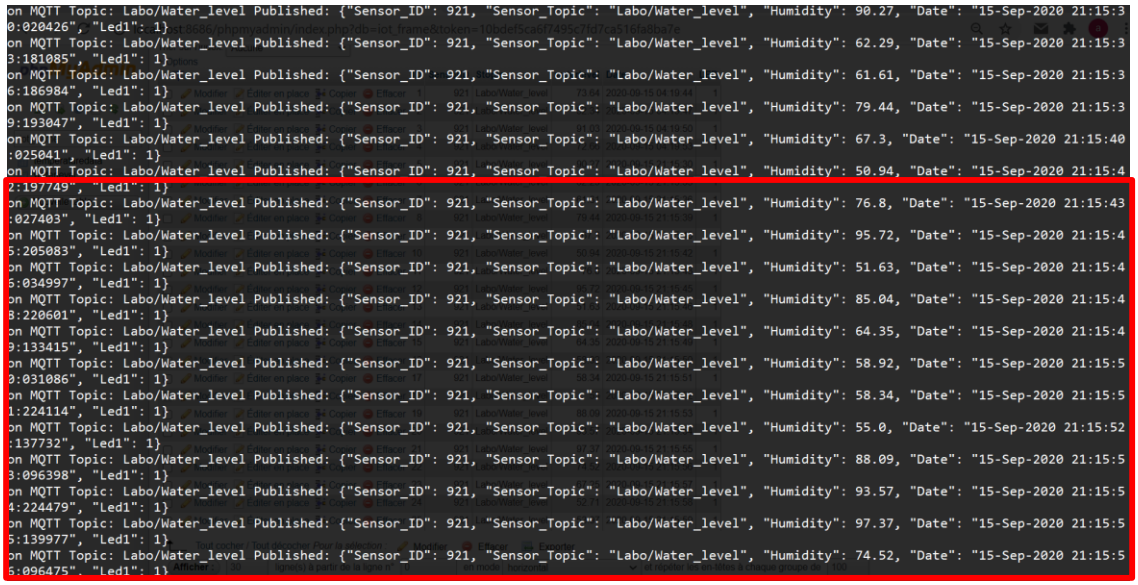


Figure 3.10 : Les données de publieur Labo/Water_Level

Les messages publiés se composent de l'ID du capteur, du topic, des données et de State.

L'utilisateur peut afficher les données du topic pour lequel il s'est précédemment abonné via le bouton Details, si l'utilisateur s'est abonné au topic : Labo /Temperature, l'interface suivante apparaîtra.

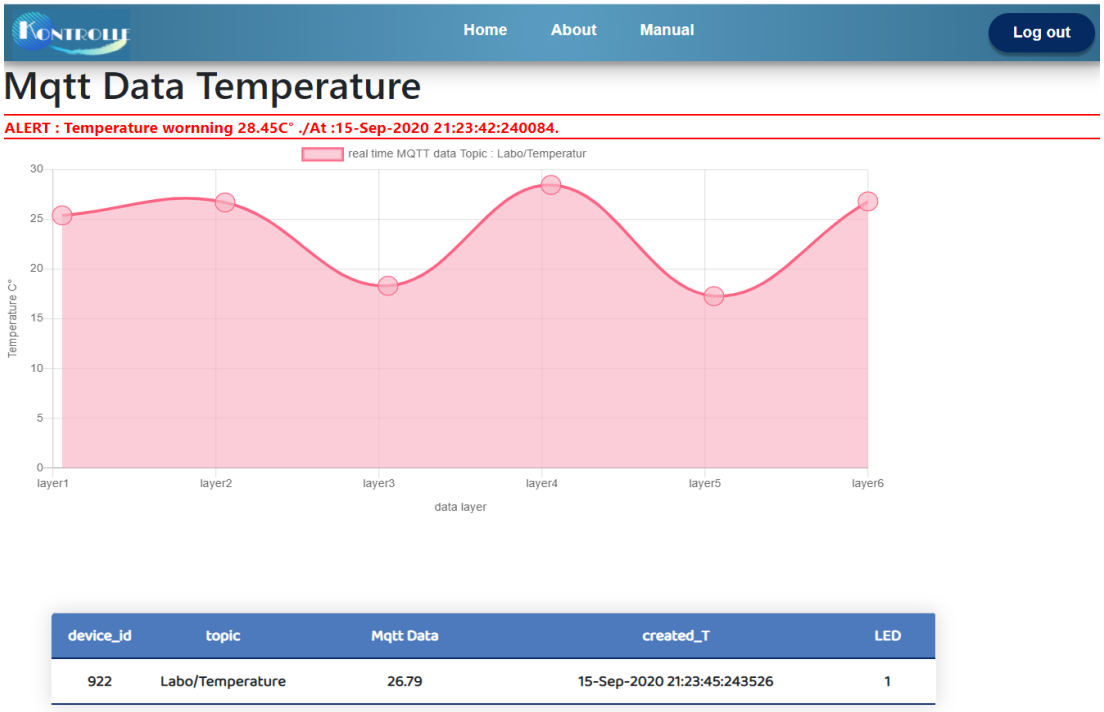


Figure 3.11 :L'interface du topic Labo/Temperature

Si l'utilisateur s'abonne au topic : Lab / water_level, l'interface suivante représentera les résultats comme suit :

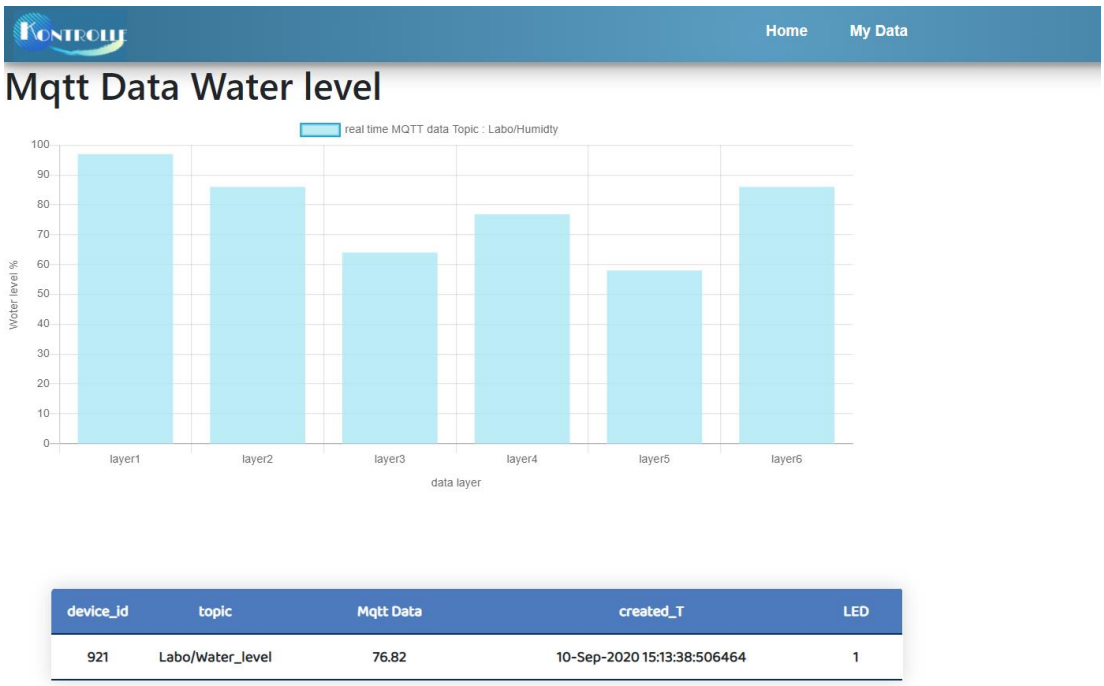


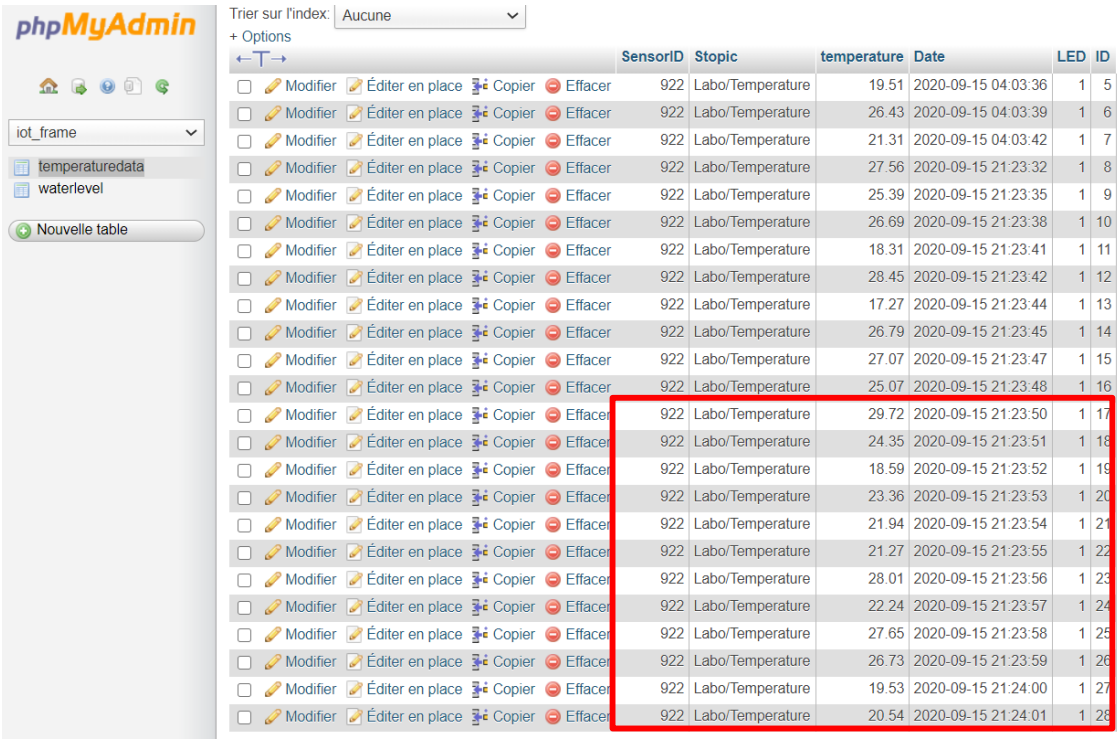
Figure 3.12 : L'interface du topic Labo/water_level

Ces deux interfaces représentent les données captées à partir de l'environnement en temps réel interprétées sous forme de tableau et de graphique, lorsque la température ou le niveau d'eau dépasse la valeur seuil, l'utilisateur reçoit une alerte pour prendre ces précautions. Il peut décider d'activer un actionneur en fonction des valeurs des capteurs, en cliquant sur le bouton d'Action pour allumer l'actionneur, dans ce cas l'utilisateur est un publieur et l'actionneur est un abonné qui reçoit les messages MQTT dans le port ''–p 1883'' sur le topic ''-t Labo/Temperature'', la **figure 3.13** montre le message reçu ''turn it on''.

```
D:\Mosquitto>mosquitto_sub -p 1883 -t Labo/Temperature
turn it on
```

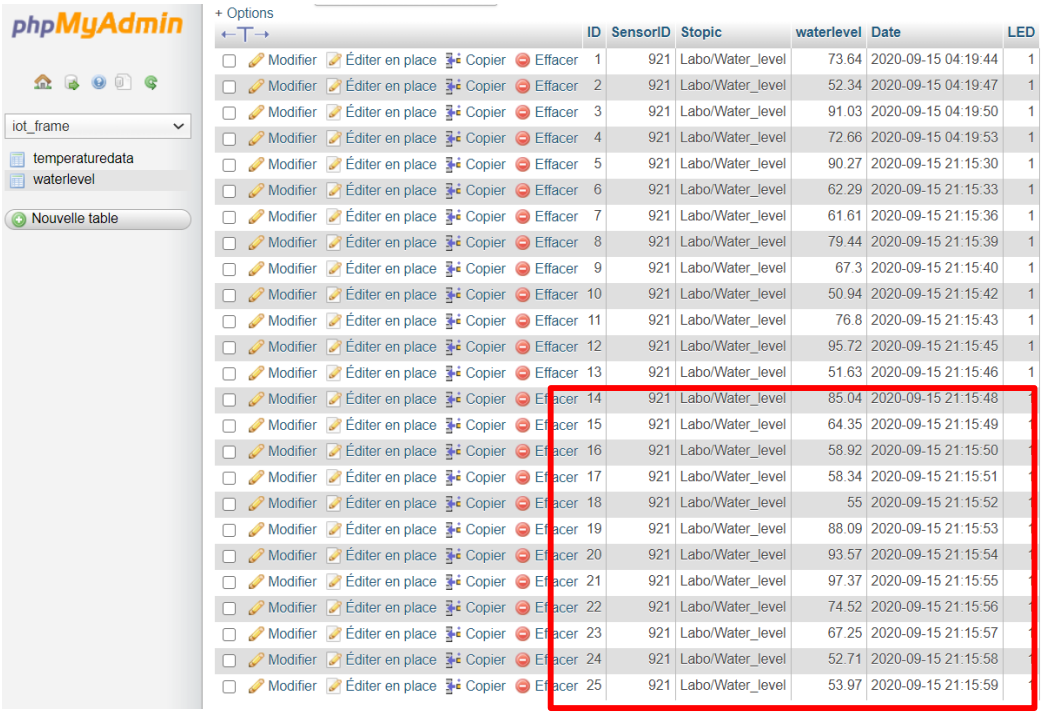
Figure 3.13 : Fenêtre de l'abonné

Dans les figures ci-dessus nous visualisons les deux tableaux de notre base de données, là où nous avons enregistré les données MQTT.



	SensorID	Stopic	temperature	Date	LED	ID
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	19.51	2020-09-15 04:03:36	1	5
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	26.43	2020-09-15 04:03:39	1	6
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	21.31	2020-09-15 04:03:42	1	7
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	27.56	2020-09-15 21:23:32	1	8
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	25.39	2020-09-15 21:23:35	1	9
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	26.69	2020-09-15 21:23:38	1	10
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	18.31	2020-09-15 21:23:41	1	11
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	28.45	2020-09-15 21:23:42	1	12
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	17.27	2020-09-15 21:23:44	1	13
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	26.79	2020-09-15 21:23:45	1	14
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	27.07	2020-09-15 21:23:47	1	15
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	25.07	2020-09-15 21:23:48	1	16
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	29.72	2020-09-15 21:23:50	1	17
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	24.35	2020-09-15 21:23:51	1	18
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	18.59	2020-09-15 21:23:52	1	19
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	23.36	2020-09-15 21:23:53	1	20
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	21.94	2020-09-15 21:23:54	1	21
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	21.27	2020-09-15 21:23:55	1	22
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	28.01	2020-09-15 21:23:56	1	23
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	22.24	2020-09-15 21:23:57	1	24
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	27.65	2020-09-15 21:23:58	1	25
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	26.73	2020-09-15 21:23:59	1	26
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	19.53	2020-09-15 21:24:00	1	27
Modifier Éditer en place Copier Effacer	922	Labo/Temperature	20.54	2020-09-15 21:24:01	1	28

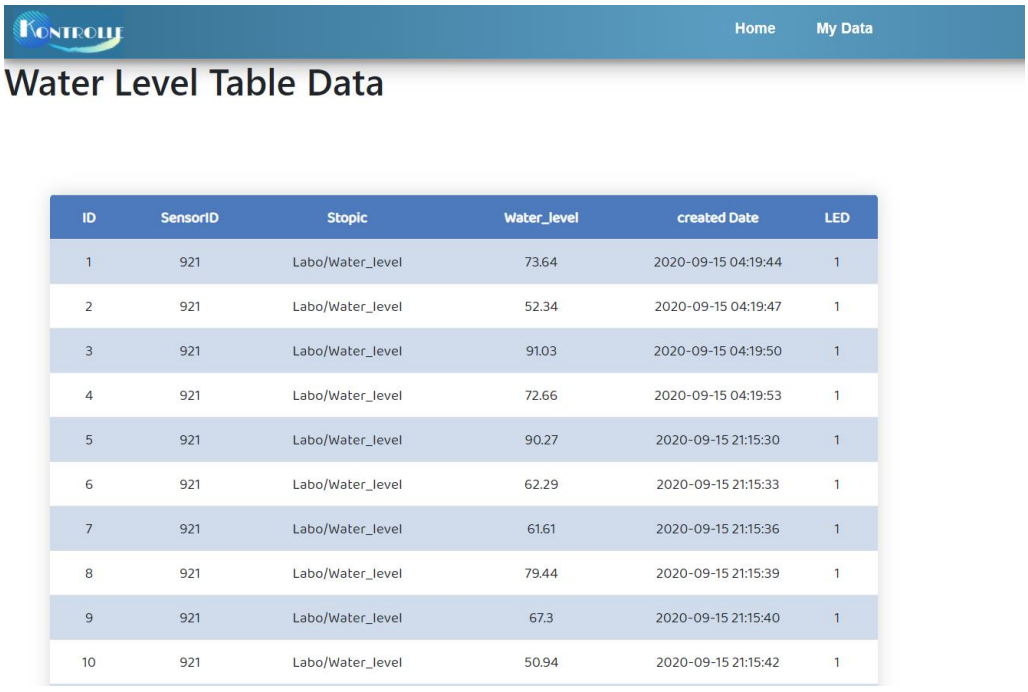
Figure 3.14 : Table BDD Labo/Temperature



	ID	SensorID	Stopic	waterlevel	Date	LED
Modifier Éditer en place Copier Effacer	1	921	Labo/Water_level	73.64	2020-09-15 04:19:44	1
Modifier Éditer en place Copier Effacer	2	921	Labo/Water_level	52.34	2020-09-15 04:19:47	1
Modifier Éditer en place Copier Effacer	3	921	Labo/Water_level	91.03	2020-09-15 04:19:50	1
Modifier Éditer en place Copier Effacer	4	921	Labo/Water_level	72.66	2020-09-15 04:19:53	1
Modifier Éditer en place Copier Effacer	5	921	Labo/Water_level	90.27	2020-09-15 21:15:30	1
Modifier Éditer en place Copier Effacer	6	921	Labo/Water_level	62.29	2020-09-15 21:15:33	1
Modifier Éditer en place Copier Effacer	7	921	Labo/Water_level	61.61	2020-09-15 21:15:36	1
Modifier Éditer en place Copier Effacer	8	921	Labo/Water_level	79.44	2020-09-15 21:15:39	1
Modifier Éditer en place Copier Effacer	9	921	Labo/Water_level	67.3	2020-09-15 21:15:40	1
Modifier Éditer en place Copier Effacer	10	921	Labo/Water_level	50.94	2020-09-15 21:15:42	1
Modifier Éditer en place Copier Effacer	11	921	Labo/Water_level	76.8	2020-09-15 21:15:43	1
Modifier Éditer en place Copier Effacer	12	921	Labo/Water_level	95.72	2020-09-15 21:15:45	1
Modifier Éditer en place Copier Effacer	13	921	Labo/Water_level	51.63	2020-09-15 21:15:46	1
Modifier Éditer en place Copier Effacer	14	921	Labo/Water_level	85.04	2020-09-15 21:15:48	1
Modifier Éditer en place Copier Effacer	15	921	Labo/Water_level	64.35	2020-09-15 21:15:49	1
Modifier Éditer en place Copier Effacer	16	921	Labo/Water_level	58.92	2020-09-15 21:15:50	1
Modifier Éditer en place Copier Effacer	17	921	Labo/Water_level	58.34	2020-09-15 21:15:51	1
Modifier Éditer en place Copier Effacer	18	921	Labo/Water_level	55	2020-09-15 21:15:52	1
Modifier Éditer en place Copier Effacer	19	921	Labo/Water_level	88.09	2020-09-15 21:15:53	1
Modifier Éditer en place Copier Effacer	20	921	Labo/Water_level	93.57	2020-09-15 21:15:54	1
Modifier Éditer en place Copier Effacer	21	921	Labo/Water_level	97.37	2020-09-15 21:15:55	1
Modifier Éditer en place Copier Effacer	22	921	Labo/Water_level	74.52	2020-09-15 21:15:56	1
Modifier Éditer en place Copier Effacer	23	921	Labo/Water_level	67.25	2020-09-15 21:15:57	1
Modifier Éditer en place Copier Effacer	24	921	Labo/Water_level	52.71	2020-09-15 21:15:58	1
Modifier Éditer en place Copier Effacer	25	921	Labo/Water_level	53.97	2020-09-15 21:15:59	1

Figure 3.15 : Table BDD Labo/Water_Level

L'utilisateur peut également récupérer les données de la base de données dans son interface personnelle.



ID	SensorID	Stopic	Water_level	created Date	LED
1	921	Labo/Water_level	73.64	2020-09-15 04:19:44	1
2	921	Labo/Water_level	52.34	2020-09-15 04:19:47	1
3	921	Labo/Water_level	91.03	2020-09-15 04:19:50	1
4	921	Labo/Water_level	72.66	2020-09-15 04:19:53	1
5	921	Labo/Water_level	90.27	2020-09-15 21:15:30	1
6	921	Labo/Water_level	62.29	2020-09-15 21:15:33	1
7	921	Labo/Water_level	61.61	2020-09-15 21:15:36	1
8	921	Labo/Water_level	79.44	2020-09-15 21:15:39	1
9	921	Labo/Water_level	67.3	2020-09-15 21:15:40	1
10	921	Labo/Water_level	50.94	2020-09-15 21:15:42	1

Figure 3.16 : Interface des données de niveau d'eau enregistrées

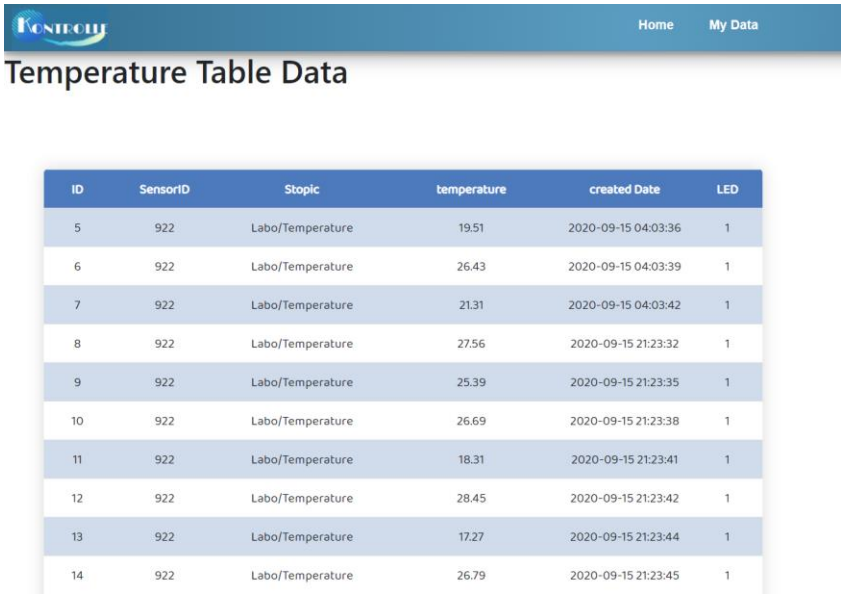


Figure 3.17 : Interface de données de température enregistrées

3.4 Avantage de la plateforme KONTROLLE

La plateforme KONTROLLE offre aux utilisateurs plusieurs avantages et services :

- ✓ Les utilisateurs peuvent surveiller leurs environnements à distance.
- ✓ Adaptable à différent domaines comme l'industrie, le transport ...etc.
- ✓ Facile à utiliser et simple à configurer.
- ✓ Permet de visualiser et d'analyser correctement les données en temps réel pour améliorer la prise de décision basée sur les données.

3.5 Les problèmes rencontrés

Évidemment, nous avons rencontré des problèmes lors de notre etude, d'abord nous avons eu des difficultés à intégrer et à faire fonctionner l'application Channels dans Django. Ce qui nous a poussés à proposer une autre solution basée sur les Websockets,

Nous estimons que les résultats obtenus sont satisfaisants, cependant le travail reste perfectible dans le futur.

3.6 Conclusion

Dans ce chapitre nous avons présenté notre simulation par l'explication de ces principales interfaces. Enfin grâce à cette simulation nous avons pu montrer l'utilité de notre plateforme.

Conclusion générale

L'Internet des Objets est un domaine très vaste, il offre beaucoup de services et peut apporter des solutions à différents problèmes dans tous les secteurs. l'IoT offre également un grand potentiel pour le développement de nouvelles applications utiles aussi bien pour les particuliers que pour l'industrie. Le problème traité dans ce mémoire est comment développer une nouvelle application qui permet d'administrer les objets connectés à distance.

Pour ce faire nous avons créé une plateforme de développement générique adaptable pour l'Internet des Objets à base du framework Django. La plateforme facilite la communication, le transport et l'acheminement des données entre les objets eux-mêmes et d'autres clients (utilisateurs externes).

Cette plateforme a pour but de permettre de contrôler les objets à distance sur un réseau, à travers l'exploitation des objets connectés, créant des opportunités d'intégration plus directe entre le monde physique et les systèmes informatiques, et résultant en une efficacité, une précision de gestion.

Nous avons pu créer cette plateforme ainsi que récupérer les données du broker, mais en raison des événements de cette année, nous n'avons pas pu réaliser un véritable réseau pour tester notre framework. Nous avons donc effectué les tests en local, avec des capteurs virtuels, et un broker installé dans la même machine.

Nous avons pu atteindre le but final de notre projet. En effet, notre plateforme a permis de gérer des objets connectés virtuelles et interagir avec son environnement.

Ce projet nous a donné l'opportunité de synthétiser et d'exploiter nos connaissances et compétences acquis durant notre cursus de formation à l'université.

Nous avons appris à maitriser le langage de programmation python et nous avons eu la chance de tester la plateforme Django et le protocole MQTT, et le plus important c'est d'avoir enrichi nos connaissances à propos des nouvelles technologies de l'Internet des Objets.

Nous attendons avec impatience de voir des améliorations supplémentaires pour rendre ce projet utilisable, dans ce cadre nous proposons quelques perspectives d'avenir :

- Ajouter des systèmes et des protocoles de sécurité plus fiable.
- Ajouter des systèmes de contrôle à distance synchronisé, avec des services et des possibilités offrir par le protocole MQTT.
- Améliorer les services de la plateforme et augmenter la capacité de stockage des données, pour augmenter le nombre des utilisateurs.
- La mise en place de l'application Channels, pour la récupération des données avec une plateforme purement Django.

Annexes

Annexe 1 : Les capteurs virtuels (Publieur), publier des messages sur deux topics Labo/Temperature et Labo/woter_level.

```
import paho.mqtt.client as mqtt
import random, threading, json
from datetime import datetime
import time
#====================================================
# MQTT Settings
MQTT_Broker = "localhost"
MQTT_Port = 1883
Keep_Alive_Interval = 45
MQTT_Topic_Water_level = "Labo/Water_level"
MQTT_Topic_Temperature = "Labo/Temperature"

#====================================================

def on_connect(client, userdata, rc):
    if rc != 0:
        pass
        print ("Unable to connect to MQTT Broker...")
    else:
        print( "Connected with MQTT Broker: " + str(MQTT_Broker))

def on_publish(client, userdata, mid):
    pass

def on_disconnect(client, userdata, rc):
    if rc !=0:
        pass

mqttc = mqtt.Client()
mqttc.connect(MQTT_Broker, int(MQTT_Port), int(Keep_Alive_Interval))
```

```
#mqttc.on_connect = on_connect
#mqttc.on_disconnect = on_disconnect
#mqttc.on_publish = on_publish


while True:
    def publish_To_Topic(topic, message):
        mqttc.publish(topic,message)
        print ( "on MQTT Topic: " + str(topic), "Published: " + str(message) + " " )
        #print ("")


    #======================================================
    # FAKE SENSOR
    # Dummy code used as Fake Sensor to publish some random values
    # to MQTT Broker

    toggle = 0

    def publish_Fake_Sensor_Values_to_MQTT():
        threading.Timer(3.0, publish_Fake_Sensor_Values_to_MQTT).start()
        global toggle
        if toggle == 0:
            Temperature_Fake_Value = float("{0:.2f}".format(random.uniform(17, 30)))


            Temperature_Data = {}
            Temperature_Data['Sensor_ID'] = 922
            Temperature_Data['Sensor_Topic'] = MQTT_Topic_Temperature
            Temperature_Data['Temperature'] = Temperature_Fake_Value
            Temperature_Data['Date'] = (datetime.today()).strftime("%d-%b-%Y %H:%M:%S:%f")
            Temperature_Data['Led1'] =1
            temperature_json_data = json.dumps(Temperature_Data)

            #print ("Publishing fake Temperature Value: " + str(Temperature_Fake_Value) + "...")
            publish_To_Topic (MQTT_Topic_Temperature, temperature_json_data)
            toggle = 1
        else:
            Water_level_Fake_Value = float("{0:.2f}".format(random.uniform(50, 100)))

            Water_level_Data = {}
            Water_level_Data['Sensor_ID'] = 921
            Water_level_Data['Sensor_Topic'] = MQTT_Topic_Water_level
            Water_level_Data['Humidity'] = Water_level_Fake_Value
            Water_level_Data['Date'] = (datetime.today()).strftime("%d-%b-%Y %H:%M:%S:%f")
            Water_level_Data['Led1'] =1
            Water_level_json_data = json.dumps(Water_level_Data)

            #print ("Publishing fake Water_level Value: " + str(Water_level_Fake_Value) + "...")
            publish_To_Topic (MQTT_Topic_Water_level, Water_level_json_data)
            toggle = 0

    publish_Fake_Sensor_Values_to_MQTT()
    time.sleep(10)

```

Annexe 2 : Abonné (Paho MQTT client en JavaScript), exemple de Labo/Temperature.

```
129
130   <script type="text/javascript">
131
132
133       // Create a client instance
134   client = new Paho.MQTT.Client('localhost', Number('9001'), "clientId");
135
136   // set callback handlers
137   client.onConnectionLost = onConnectionLost;
138   client.onMessageArrived = onMessageArrived;
139
140   // connect the client
141   client.connect({onSuccess:onConnect});
142
143
144   // called when the client connects
145   function onConnect() {
146     // Once a connection has been made, make a subscription and send a message.
147     console.log("onConnect");
148     client.subscribe("Labo/Temperature");
149     message = new Paho.MQTT.Message("Hello");
150     message.destinationName = "World";
151     client.send(message);
152   }
```

```
154  // called when the client loses its connection
155  function onConnectionLost(responseObject) {
156    if (responseObject.errorCode !== 0) {
157      console.log("onConnectionLost:"+responseObject.errorMessage);
158    }
159  }
160
161  // called when a message arrives
162  function onMessageArrived(message) {
163    console.log(message.payloadString);
164    var recdata=JSON.parse(message.payloadString);
165    //document.getElementById("data_list").innerHTML = recdata.Sensor_ID+" "+recdata.Temperature+" ";
166    // alert mesage :
167    if (recdata.Temperature > 27){
168       document.getElementById("data_list").innerHTML = "  "+"ALERT : Temperature wornning "+recdata.Temperature+"C°
169    };
170
171    // save data :
172
173    //graphe update :
174    newcaptdata=captdata['data']['datasets'][0]['data'];
175    newcaptdata.shift();
176    newcaptdata.push(recdata.Temperature);
177    captdata['data']['datasets'][0]['data']=newcaptdata;
178    window.myLine.update();
179
```

```
181     //table update :
182     //document.getElementById("tabledata").innerHTML = recdata.Sensor_ID+" "+recdata.Sensor_Topic+" "
183     var row = `<tr> <td>${recdata.Sensor_ID} </td>
184                                 <td>${recdata.Sensor_Topic} </td>
185                                 <td>${recdata.Temperature} </td>
186                                 <td>${recdata.Date} </td>
187                                 <td>${recdata.Led1} </td>
188                             </tr>`;
189       document.getElementById("tabledata").innerHTML = row;
190
191     //save deta in json file:
192     //var text_data=JSON.stringify(recdata);
193     //localStorage.setItem("consoldata", text_data);
194     //console.log(localStorage)
195     //var fs=require('fs');
196     //var text_data=JSON.stringify('recdata', null, 2);
197     //fs.writeFileSync('database.json', text_data, finished);
198
199
200 }
201
```

Bibliographie

[1] Rafael Marques, Kevin Ashton. **That 'Internet of Things' Thing.** (2009).

[2] Cécile Nadai. Focus on the Internet of Things (IoT). 28 aout 2017.

[3] (Christophe Baland 2017), Damien Cauquil, Thomas Gayet, Julia Juvigny, Renaud Lifchitz, Nha-Khanh Nguyen. La sécurité de l'Internet des Objets.Digital Security.

[4] Ms.Eln.Megtit et Dahmane izihome, ''Les objets connectés : avantages et inconvenients'' [En ligne]. Available. http://izihome.fr/objets-connectes-avantages-inconvenients/.

[5] Mémoire - Maxime Dechany International Télécommunication Union. (2014, mai 23).

[6] (Zeinab Kamal elddin, 2017) Zeinab Kamal Aldein Mohammed, Elmustafa Sayed Ali Ahmed. (2017). Internet of Things Applications, Challenges and Related Future Technologies. WSN 67(2) 126-148.

[7] IBM Journal. (2018). 5 Functional Requirements of an IoT Platform. Récupéré sur IBM.

[8] K.Patel, & S.Patel, (2016, mai). Internet of Things-IOT. Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. International Journal of Engineering Science and Computing, 6(5), pp. 6123 - 6131. doi:DOI 10.4010/2016.1482

[9] IOSR Journal of Economics and Finance (IOSR-JEF) e-ISSN : 2321-5933, p-ISSN : 2321-5925.Volume 8, Issue 5 Ver. II (Sep.-Oct .2017), PP 21-35

[10] Xu Li Da, He We, et Li Shangcang. (2014). Internet of Things in Industries. A Survey. IEEE Transactions on Industrial Informatics.

[11] Marques Rafaeal, Armelinni Fabiano. Novembre 2017. Proposta de modelo de referência para utilização da internet das coisas como suporte à estratégia de marketing.

[12] Livre Blanc Préparer la Révolution de l'internet des Objets. 2016.

[13] R.ACHOUR, N.Makhloufi, Authentification dans l'internet des objets. Université A/MIRA de Bejaia, 2017.

[14] Renaud, ''L'internet des objets et ses applications au quotidien'' [En ligne]. Available: https://www.objetconnecte.com/.

[15] 2019/03. Publish/Subscribe Model. https://behrtech.com/blog/mqtt-in-the-iot-architecture/.

[16] 2018/04/16. Internet des objets quels protocoles applicatifs utiliser-1-2. engineering.publicissapient.fr.

[17] S. Feng, J. Cerles, H. Dalmas, T. Do-Khac, and B. Paulin. Sécurité des objets Connectés. Institut national des hautes études de la sécurité et de la justice, 2014.

[18] ''L'IoT au travail aujourd'hui'' [En ligne]. Available. https://www.intel.fr/.

[19] Alok Kulkarni, Sampada Sathe. Healthcare applications of the Internet of Things.2014.

[20] Internet des objets industriel. www.touteleurope.eu.

[21] Bâtiment intelligent. https://www.torondel.net/.

[22] 2018. 5-applications-of-iot-in-agriculture. www.biz4intellia.com.

[23] K. Rose, S. Eldridge. & Chapin. The Internet of Things: An Overview - Understanding the Issues and Challenges of a More Connected World. Récupéré sur Internet Society. 2015 octobre.

[24] Zeinab Kamal Aldein Mohammed, Elmustafa Sayed Ali Ahmed. Internet of Things Applications, Challenges and Related Future Technologies. WSN 67(2) 126-148. (2017).

[25] Y.ait mouhoub, F.Bouchebbah . Propotion d'un modèle de confaince pour l'internet des Objets, Université A/MIRA de Bejaia. 2015.

[26] Y. Challal. Sécurité de l'internet des objets : Vers une Approche Cognitive et Systémique. PHD thesis, Université de technologie de Compiègne. 2012.

[27] Introduction à la sécurité informatique. www.commentcamarche.net

[28] C.Llorens, L.Levier and D.Valois. Tableau de bord de la sécurité réseaux. Eyrolles. 2006

[29] Y. CHALLAL. « Sécurité de l'Internet des Objets : vers une approche cognitive et systémique », HDR, Juin, 2012, UTC.

[30] 2/05/2018. Hardware – Raspberry Pi2. https://data-flair.training/blogs/iot-hardware/.IoT.

[31] Raspberry Pi 3 Model B+. https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[32] Tizzaoui Youva: " Internet des Objets « IoT » Application : Industrie 4.0", UniversitéAbderrahmane Mira de Bejaïa. Pour obtenir le diplômé de Master en Télécommunications, 2017.

[33] IoT Sensors and Actuators. www.postscapes.com.

[34] 2018/11. Types Sensors applications. www.electricaltechnology.org.

[35] 2020. Types of Sensor. www.electrical4u.com.

[36] IoT Hardware and Software. mindmajix.com.

[37] IoT Sensors and Actuators. iotbyhvm.ooo.

[38] Gateways et dispositifs iot. https://trigon.ch.

[39] Module nodemcu esp8266. http://addivoo.com.

[40] Discovery kit for IoT node,multi-channel communication with STM32L4. B-L475E-IOT01A. 2018.

[41] Patrick fuchs, Pierre parlain. Cours de python. Université de Paris, France 2018.

[42] Octobre 2019. Langage Python https://www.lebigdata.fr/python-langage-definition

[42] Mathieu Xhonneux et Maxime Lorant. Développez votre site web avec le Framework Django. 12aout 2019

[43] 2019. Yassine Haddab, Introduction à l'internet des objets.

[44] 05/04/2019. What is the Future Scope of Web Development with Python and Django. https://www.sayonetech.com/blog/future-scope-python-django-web-development/

[45] 02/09/2019. Django Architecture – 3 Major Components of MVC Pattern. https://data-flair.training/blogs/django-architecture/.

[46] Najib Tounsi. Introduction à la base de données.

[47] Les BASES de DONNEES dans WampServer. https://docplayer.fr.

[48] Anthony NIZAC. Wampserver. Installation et présentation. 07/03/2018.

[49] 27/07/ 2016. Internet des Objets. bien comprendre MQTT. https://www.lemagit.fr

[50] 2019. Broker in the IoT system components. https://1sheeld.com/mqtt-protocol/

[51] 2019. Le protocole MQTT. MQTT.org.

[52] 2018. Communicate with A MQTT Broker Using Paho Clients on Dusun Gateways. https://www.dusuniot.com/link-to-a-mqtt-broker-on-dusun-hub

[53] Andrew Godrwin. Channels Documentation Release 2.4.0, 2020