



Master1 en Traitement du Signal et des Images

Mention Ingénierie des Objets Intelligents

Rapport technique

Mise en œuvre d'une plateforme de Cybersécurité pour IoT



ThingsBoard

Réalisé par Nour RAJIOUB, Ourida ACHEROUFENE, Joëlle SAIDEH, Alaa YAKHNI

Encadré par M. Noël RICHARD et Marc PARENTHOEN

Année universitaire 2021-2022



Table des matières

Table des matières	2
Tables des figures.....	5
I. Introduction technique.....	8
II. Contextualisation du projet	9
1. Les éléments de résolution	9
2. Procédure de résolution	10
3. Présentation de Thingsboard	11
III. Installation de Thingsboard.....	12
1. Installation sous Windows	12
a) Installation Java 11 (OpenJDK).....	12
b) Installation du service Thingsboard	18
c) Configuration de la base de données Thingsboard	19
d) Implémentation de la file d'attente du serveur en mémoire.....	25
e) Choisissez le service de file d'attente ThingsBoard.....	32
f) Implémentation du service de file d'attente Kafka.....	33
2. Installation sous Linux	35
a) Installez Java 11 (OpenJDK)	35
b) Installation du service ThingsBoard	36
c) Configurer la base de données ThingsBoard	36
d) Configuration de ThingsBoard.....	37
e) Installation de Kafka	38
f) Configurer le fichier ZooKeeper Systemd Unit	38
g) Configurer le fichier Kafka Systemd Unit	39
h) Configuration de ThingsBoard.....	40
g) [Facultatif] Mise à jour de la mémoire pour les machines lentes (1 Go de RAM)	40

h) Exécutez le script d'installation.....	41
i) Démarrer le service ThingsBoard	41
IV. Les bases de Thingsboard	42
1. Fonctionnalités de Thingsboard Community	42
4. Utilisation de base des fonctionnalités les de base de ThingsBoard :	42
5. Exemple d'utilisation de la plateforme :	43
a) Ajouter un nouveau dispositif :.....	43
b) Etablissement de la connexion du dispositif :.....	44
c) Création du tableau de bord :	45
d) L'ajout d'un alias associé à l'entité :	45
e) Ajout du widget dans le tableau de bord :.....	46
f) Attribution de l'appareil au client.....	47
a) Création d'un utilisateur client.....	47
V. Exemple d'utilisation avec Esp32 :	50
1. Environnement de développement et carte ESP32	50
2. Télémétrie :	52
3. Tableau de bord	52
a) Tableau de bord pour un capteur.....	52
b) Tableau de bord avec plusieurs capteurs	53
g) Utilisation de plusieurs cartes :	54
Code utilisé.....	55
Sensors.h	55
VI. Sécurité.....	66
VII. Annexes	70
1) Installation de l'environnement Espressif IDE.....	70
a) Installation de JAVA.....	70
b) Installation d'ESP IDF v4.4.1	70

c)	Installation de Espressif IDE.....	71
d)	Paramétrage de Espressif IDE	71
VIII.	Références	75

Tables des figures

Figure 1: Schéma explicatif du projet	9
Figure 2: schéma explicatif du fonctionnement de la plateforme	10
Figure 3: Fenêtre d'installation d'openJDK 11.....	13
Figure 4: Archive de build	13
Figure 5: sélection de JDK convenant à notre système d'exploitation.	13
Figure 6: Exécution d'open JDK.....	14
Figure 7: JDK avec hotspot.....	14
Figure 8: JDBC 42.....	15
Figure 9: Accéder au fichier d'exécution.....	15
Figure 10: Création du répertoire pour le fichier d'exécution.	16
Figure 11: Emplacement du fichier PostgresSQL.....	16
Figure 12: création de la nouvelle variable d'environnement.	17
Figure 13: création de la nouvelle variable.	17
Figure 14: Insertion des valeurs de création de la variable.	18
Figure 15: Vérification de l'installation de java.....	18
Figure 16: Dossier de Thingsboard	19
Figure 17: Postgresql version 11.16.....	20
Figure 18: Lancement de l'exécuteur	20
Figure 19: Étape 1 installation de PostgreSQL	21
Figure 20: Insertion du mot de passe par défaut	21
Figure 21: Étape 2 installations de PostgreSQL.....	22
Figure 22: Installation des logiciels complémentaires avec stack Builder.....	22
Figure 23: L'ajout des outils à la base de données.....	23
Figure 24: Lancement de PgAdmin4.	23
Figure 25:création de la base de données	24
Figure 26: Crédit de database	24
Figure 27: Base de données créée	25
Figure 28: Gestion du port de Postgres	26
Figure 29: Installation de Thingsboard sur Windows Power Shell.....	27
Figure 30: Connexion à Thingsboard.....	28
Figure 31: Accès à la plateforme.....	29
Figure 32: Ajout d'un Tenant	29

Figure 33: Tenant créé.....	29
Figure 34: Création d'administrateurs.....	30
Figure 35: champs d'utilisateur à renseigner.....	30
Figure 36: Accès à la plateforme admin.....	31
Figure 37: Blocs de ThingsBoard.....	31
Figure 38: Lien pour installer kafka.....	33
Figure 39: Kafka .tgz.....	34
Figure 40: Installation sous PowerShell.....	34
Figure 41: zookeeper.properties	35
Figure 42: Dispositifs	43
Figure 43: Ajout de dispositif.....	44
Figure 44: détail du dispositif.....	44
Figure 46: Télémétrie	Erreur ! Signet non défini.
Figure 47: Cr éation d'un tableau de bord	45
Figure 48: Dispositif cr éé	46
Figure 49: Configuration de la source des donn ées.	46
Figure 50: R esultat	Erreur ! Signet non d éfini.
Figure 51: Client.....	47
Figure 52: Affectation d'un client.....	47
Figure 53: D étail de l'utilisateur	48
Figure 54: Lien de g én eration	48
Figure 55: Mot de passe de l'utilisateur.....	49
Figure 56: Plateforme à disposition du client.....	49
Figure 57: Carte X-NUCLEO-IKS01A3.....	50
Figure 58: Programme du protocole HTTP.....	51
Figure 59: Télémétrie	52
Figure 60: Tableau de bord de la température.....	53
Figure 61: Tableau de bord avec deux cartes ESP et capteurs	53
Figure 62: r écupération le jeton et les donn ées avec wireshark	67
Figure 63 : risque de certificat auto-signé	69
Figure 64 : ThingsBoard en https	69
Figure 65: Java Download.....	70
Figure 66: Installer ESP32	71
Figure 67: installer un software.....	71

Figure 68: Création d'un nouveau software.....	72
Figure 69: Dossier de esp-idfv4.3.2	73
Figure 70 : ESP-IDF Tools.....	73

I. Introduction technique

Depuis quelques années, la question de la cybersécurité est de plus en plus abordée, cela est dû à l'embarquement de l'internet des objets dans notre usage quotidien accentuant le risque d'être confronté à d'éventuelles attaques.

Au cours de notre formation qui se porte sur la connexion des objets, la cybersécurité fait partie des éléments primordiaux à prendre en compte lors de la réalisation de divers projets entrant dans ce contexte. Nous avons été encore plus suscités par cette question de cybersécurité depuis un évènement récent qui s'agit d'un piratage d'un casino à partir d'un thermomètre connecté installé dans un aquarium, cette intrusion nous a mené à se poser les bonnes questions sur ce processus de connecter tout ce qui nous entoure et prendre conscience des mauvaises surprises qui peuvent être cachées derrière la connexion d'un simple objet que ce soit à petite échelle au niveau des foyers (connexion d'une cafetière par exemple), ou sur grande échelle (informatiques des entreprises) malgré la différence de l'impact en cas de faille de sécurité, l'empêchement d'une éventuelle faille reste nécessaire.

Pour mieux comprendre l'arrivée de ce genre d'incidents, nous sommes retournés vers les rapports réalisés par la société Darktrace spécialisée dans la cybersécurité, explique que le thermomètre a été utilisé par les pirates comme un point d'accès à la base de données, la négligence de la sécurisation d'une donnée issue d'un simple appareil peut donc être la source d'une grande faille. Cela dit, la surface d'attaque serait plus étendue en multipliant ces objets connectés, alors que ces derniers ne sont pas souvent couverts de moyens de défenses traditionnelles.

Notre projet sera un point de départ pour pouvoir tester la partie sécurité, en créant une maquette permettant la collecte des données sur un réseau.

II. Contextualisation du projet

Dans le cadre de notre formation, notre objectif principal serait de réaliser une maquette pouvant gérer plusieurs objets connectés. L'enjeu serait donc d'avoir un environnement qui permet de collecter un ensemble de données transmises sur un réseau wifi local tout en donnant la possibilité d'accéder aux données tout en assurant leur sécurisation.

Nous pouvons résumer notre maquette par le schéma suivant :

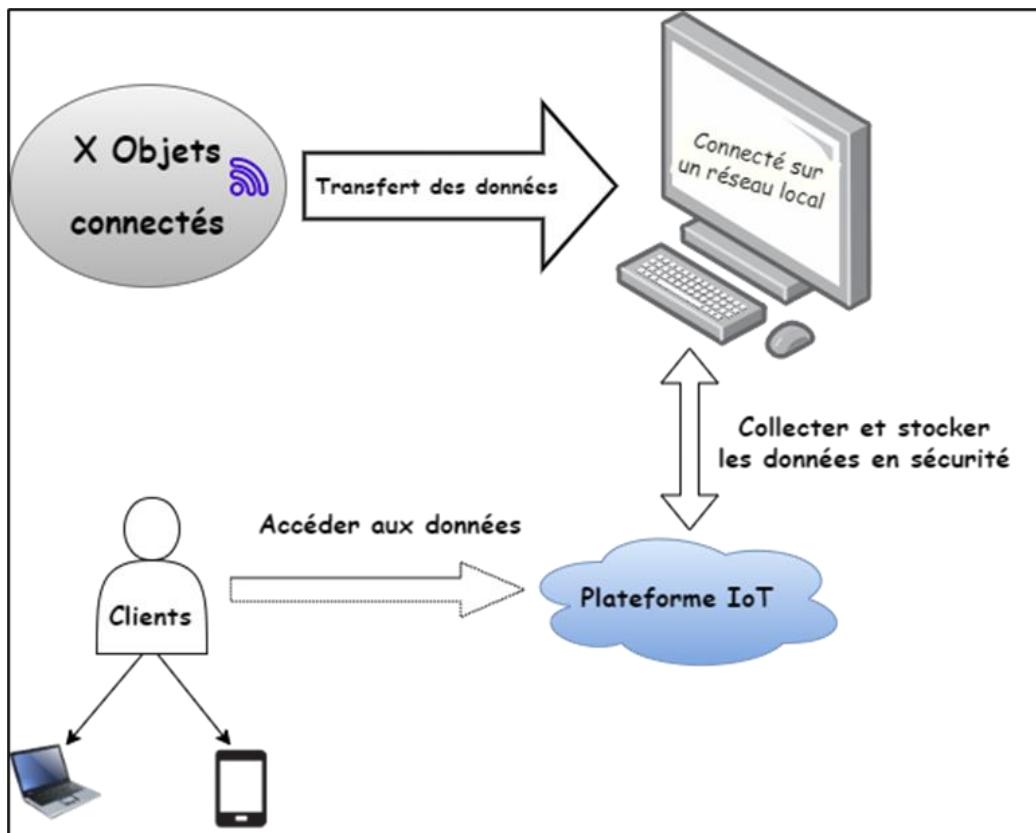


Figure 1: Schéma explicatif du projet

1. Les éléments de résolution

Avant de procéder à la réalisation, nous avons défini les éléments à utiliser, à commencer par l'outil qui va nous servir pour la récolte des données et la partie logicielle qui permettra l'envoi des données.

- **La plateforme choisie :** Nous avons opté pour la plateforme « ThingsBoard » par rapport à ses fonctionnalités et la possibilité de l'utiliser en local.

- **Environnement matériel :** Notre objet connecté est constitué à partir de la carte ESP32 et shield XNucléo qui contient les capteurs d'où seront envoyées nos données.
- **Environnement de développement logiciel :** L'envoi des données sera fait en utilisant l'environnement de développement Espressif avec le langage

2. Procédure de résolution

Nous devons créer une plateforme IoT où l'appareil donc le capteur puisse se connecter à un réseau local en commun avec l'hôte comme indiqué dans la figure suivante :

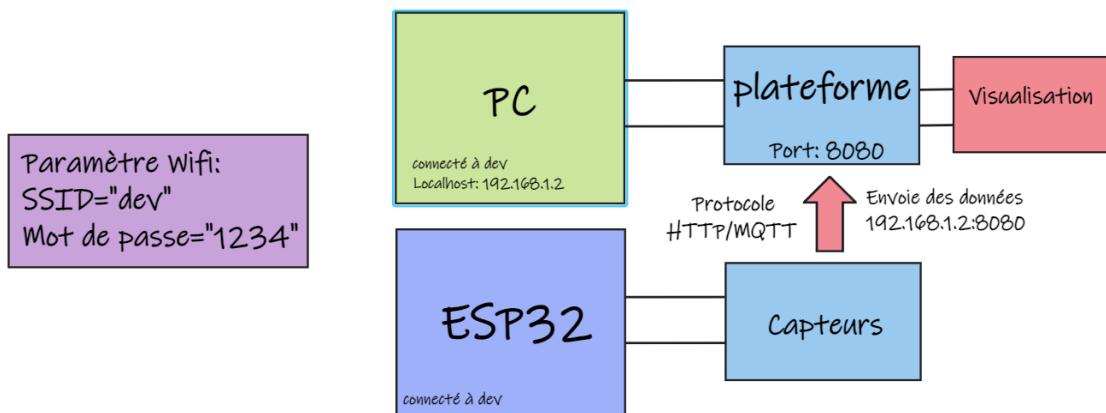


Figure 2: schéma explicatif du fonctionnement de la plateforme.

Tout d'abord, le PC et l'ESP32 se connectent à une même WiFi pour qu'ils puissent établir une communication entre eux. Le PC obtient alors une nouvelle adresse locale appelée « localhost ».

La plateforme IoT installée sur le PC utilise l'adresse locale et un port qui lui est assigné.

Le capteur de l'ESP32 délivrant la température en temps réel par exemple va procéder à un envoi de donnée vers l'adresse locale du PC et le port utilisé par la plateforme avec un protocole comme HTTP.

Pour procéder à la réalisation, il faut disposer des outils précités, en commençant par l'installation de la plateforme et l'environnement de développement, une fois ces deux éléments sont disponibles, il faut passer à la configuration de la plateforme afin de pouvoir l'utiliser.

3. Présentation de Thingsboard

Une Plateforme open source, qui permet le développement et la gestion de projets IOT. Elle fournit une solution cloud ou sur site IoT prête à l'emploi tout en donnant une infrastructure côté serveur.

Caractéristiques :

- Provisionnez les appareils, les actifs et les clients, et définissez les relations entre eux.
- Collectez et visualisez les données des appareils et des ressources.
- Analysez la télémétrie entrante et déclenchez des alarmes avec un traitement d'évènements complexe.
- Contrôlez vos appareils à l'aide d'appels de procédure à distance (RPC).
- Créez des flux de travail basés sur un évènement du cycle de vie de l'appareil, un évènement d'API REST, une demande RPC, etc.
- Concevez des tableaux de bord dynamiques et réactifs et présentez la télémétrie et les informations sur les appareils ou les actifs à vos clients.
- Activez les fonctionnalités spécifiques aux cas d'utilisation à l'aide de chaines de règles personnalisables.
- Poussez les données de l'appareil vers d'autres systèmes.

III. Installation de Thingsboard

Nous avons effectué l'installation de la plateforme sur les deux systèmes d'exploitation Windows et Linux, avant d'y procéder, il faut disposer d'au moins 2G de RAM sur la machine pour faire fonctionner la plateforme avec la base de données.

1. Installation sous Windows

a) Installation Java 11 (OpenJDK)

Le service ThingsBoard s'exécute avec la version 11 de Java. Donc nous allons suivre les étapes suivantes pour installer OpenJDK.

OpenJDK est une version open source du kit de développement Java, et le kit est une ressource clé pour les applications utilisant la plateforme java.

Ouvrir lien pour installer Open JDK : <https://adoptopenjdk.net/index.html>

-Il faut cliquer sur la rubrique « Autres plate-formes » pour accéder à la fenêtre fournissant tous les fichiers d'exécution :

Binaires OpenJDK prédéfinis gratuitement!

Java™ est le premier langage et plateforme de programmation au monde. AdoptOpenJDK utilise Infrastructure, construire et test scripts pour produire des binaires prédéfinis à partir de OpenJDK™ bibliothèques de classes et un choix de OpenJDK ou la Eclipse OpenJ9 VM. Tous les binaires et scripts AdoptOpenJDK sont licence open source et disponible gratuitement.

Télécharger pour Windows x64

1. Choisissez une version 2. Choisissez une machine virtuelle [Aidez-moi à choisir](#)

OpenJDK 8 (LTS) Point chaud
 OpenJDK 11 (LTS) OpenJ9
 OpenJDK 16 (Latest)

adoptium.net
AdoptOpenJDK has moved...

AdoptOpenJDK has moved, the blue download button will take you to the new location.
You can still get AdoptOpenJDK binaries by clicking one of the buttons below.

Autres plates-formes

Archives des versions et versions nocturnes

Figure 3: Fenêtre d'installation d'openJDK 11.

-Ensuite, sur cette page choisir « Archives de build »



Figure 4: Archive de build

-Installer OpenJDK (Windows x64) pour notre cas.

jdk-11.0.15+10			
22 April 2022			
Téléchargez les statistiques			
Linux x64	Installateur	Binaire	SHA256
	Indisponible	JDK (193 MB)	Checksum
	Indisponible	JRE (42 MB)	Checksum
Windows x64	JDK	JDK (196 MB)	Checksum
	JRE	JRE (42 MB)	Checksum
macOS x64	JDK	JDK (186 MB)	Checksum
	JRE	JRE (38 MB)	Checksum
Linux aarch64	Indisponible	JDK (189 MB)	Checksum
	Indisponible	JRE (40 MB)	Checksum
Linux s390x	Indisponible	JDK (167 MB)	Checksum
	Indisponible	JRE (36 MB)	Checksum
Alpine x64	Indisponible	JDK (193 MB)	Checksum
	Indisponible	JRE (42 MB)	Checksum
Linux ppc64le	Indisponible	JDK (175 MB)	Checksum
	Indisponible	JRE (38 MB)	Checksum
Linux arm32	Indisponible	JDK (181 MB)	Checksum
	Indisponible	JRE (40 MB)	Checksum
Windows x86	JDK	JDK (176 MB)	Checksum
	JRE	JRE (37 MB)	Checksum

Figure 5: sélection de JDK convenant à notre système d'exploitation.

-Exécutons d'OpenJDK11U-jdk_x64_windows_hotspot_11.0.15_10 (2).msi

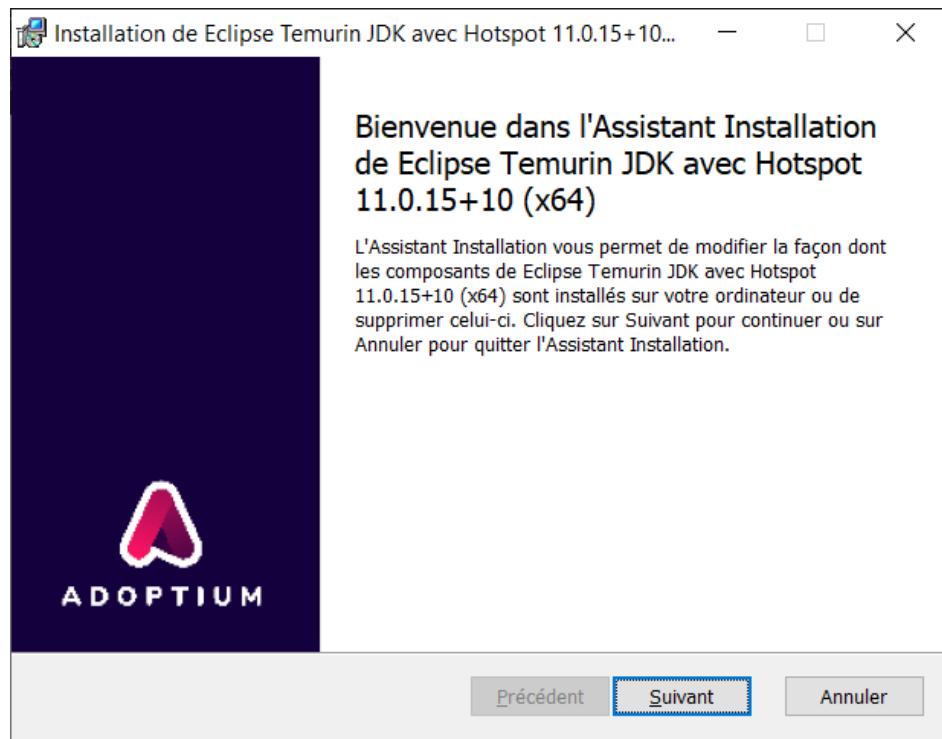


Figure 6: Exécution d'open JDK.

-Une installation d'Éclipse Temurin sera requise, choisir installation sur le disque local dans « Ajouter un Path » :

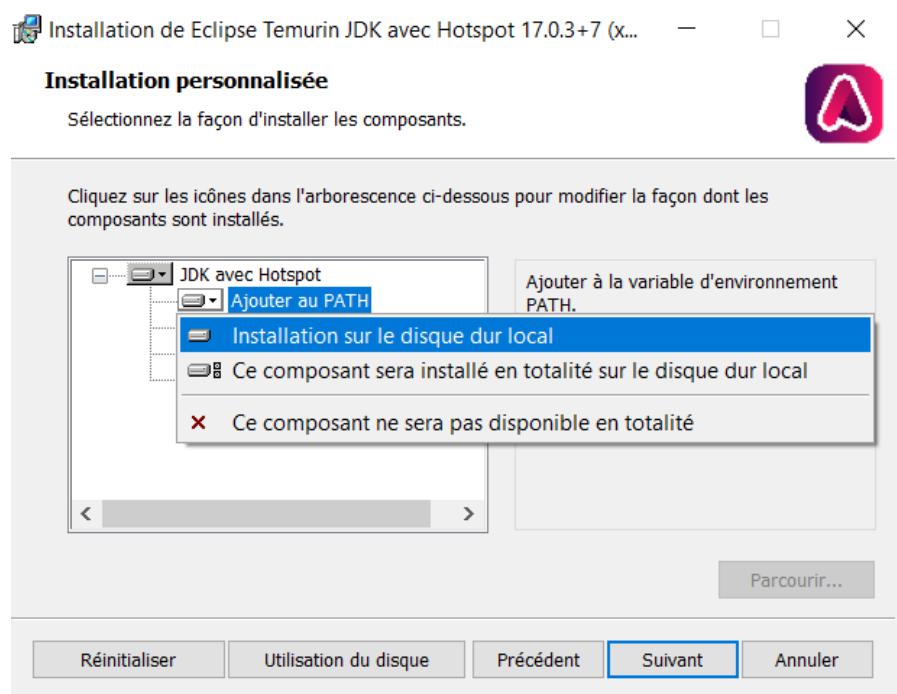


Figure 7: JDK avec hotspot

- Il faut par la suite installer PostgreSQL avec laquelle la plate-forme fonctionnera en allant sur le lien suivant : <https://jdbc.postgresql.org/download.html> et téléchargez 43.3.5 JDBC 42.

Version	JDBC 4.0	JDBC 4.1	JDBC 4.2	Source
42.3.5			42.3.5 JDBC 42	42.3.5 JDBC Source
42.3.4			42.3.4 JDBC 42	42.3.4 JDBC Source
42.3.3			42.3.3 JDBC 42	42.3.3 JDBC Source
42.3.2			42.3.2 JDBC 42	42.3.2 JDBC Source
42.2.25	42.2.25 JDBC 4	42.2.25 JDBC 41	42.2.25 JDBC 42	42.2.25 JDBC Source
42.3.1			42.3.1 JDBC 42	42.3.1 JDBC Source
42.3.0			42.3.0 JDBC 42	42.3.0 JDBC Source
42.2.24	42.2.24 JDBC 4	42.2.24 JDBC 41	42.2.24 JDBC 42	42.2.24 JDBC Source
42.2.23	42.2.23 JDBC 4	42.2.23 JDBC 41	42.2.23 JDBC 42	42.2.23 JDBC Source

Figure 8: JDBC 42

- Une fois l'installation terminée, il faut placer le fichier « postgres-42.3.5 » dans le bon endroit :

- Entrer dans le fichier d'exécution Eclipse Adoptium qui se trouve dans le disque C /programmes.

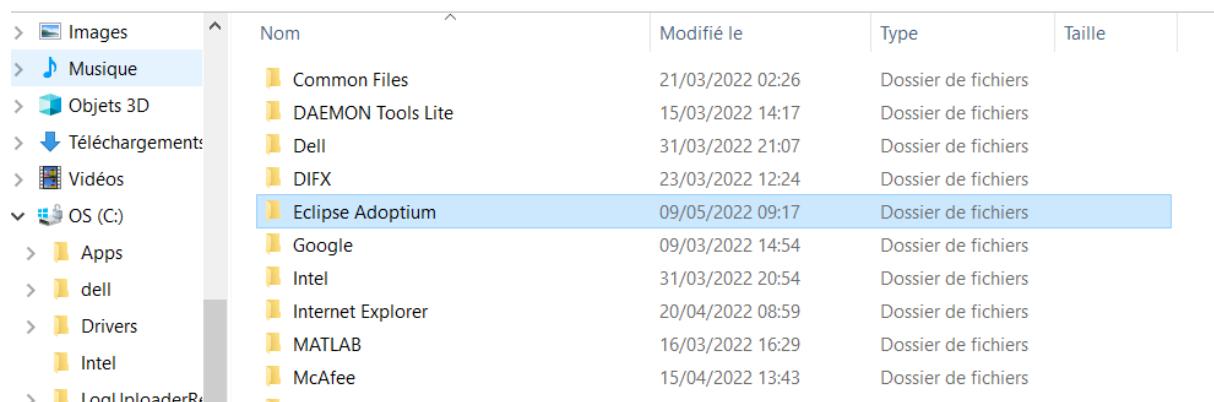


Figure 9: Accéder au fichier d'exécution

En entrant dans le répertoire, si le dossier nommé « jre » n'existe pas, il faut le créer avec les fichiers lui appartenant tel que : jre/lib/ext :

Nom	Modifié le	Type	Taille
bin	09/05/2022 09:17	Dossier de fichiers	
conf	09/05/2022 09:17	Dossier de fichiers	
include	09/05/2022 09:17	Dossier de fichiers	
jmods	09/05/2022 09:17	Dossier de fichiers	
legal	09/05/2022 09:17	Dossier de fichiers	
lib	09/05/2022 09:17	Dossier de fichiers	
Nouveau dossier	09/05/2022 09:29	Dossier de fichiers	
NOTICE	19/04/2022 22:51	Fichier	3 Ko
release	19/04/2022 22:51	Fichier	2 Ko

Figure 10: Création du répertoire pour le fichier d'exécution.

- Le fichier d'exécution sera placé ainsi dans le bon répertoire :

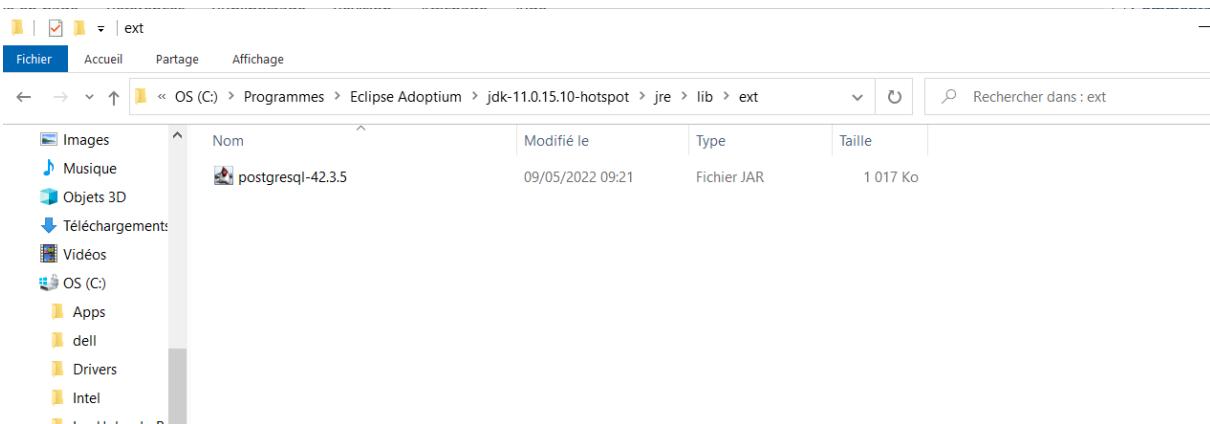


Figure 11: Emplacement du fichier PostgreSQL.

- Il faut désormais ajouter une variable globale à notre fichier installé qu'on nomme : CLASSPATH ayant comme variable le répertoire d'emplacement de « PostgreSQL » .
- Pour créer ajouter la variable : clique droit sur mon ordinateur > paramètres avancés>variables d'environnement > Nouvelle.

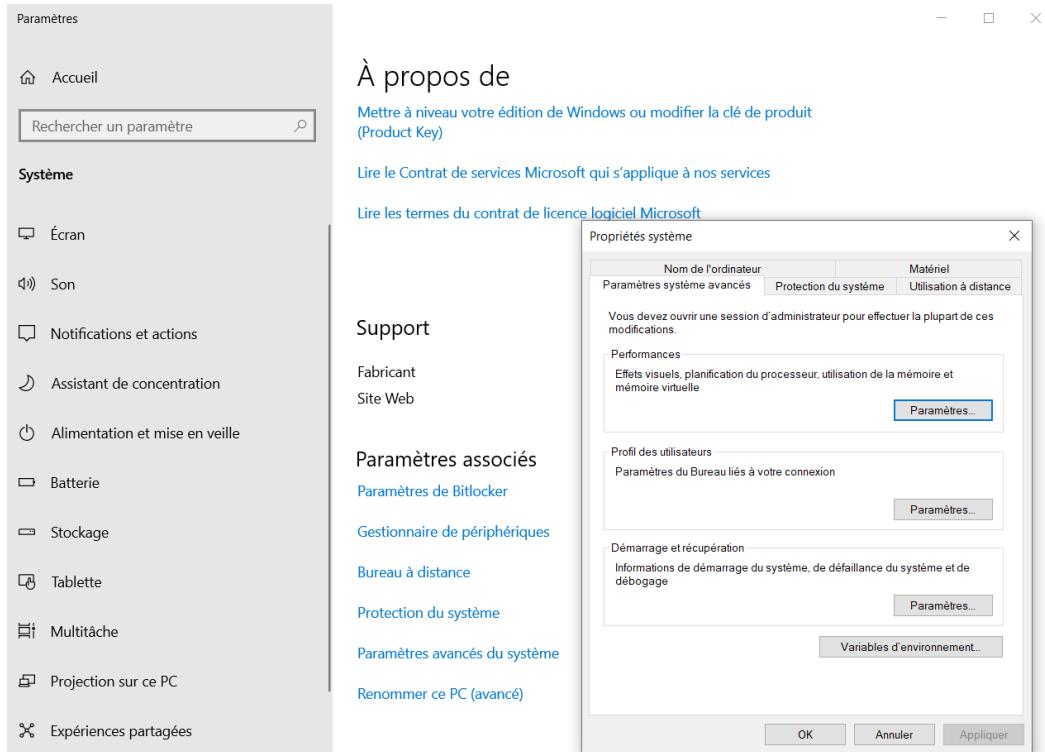


Figure 12: création de la nouvelle variable d'environnement.

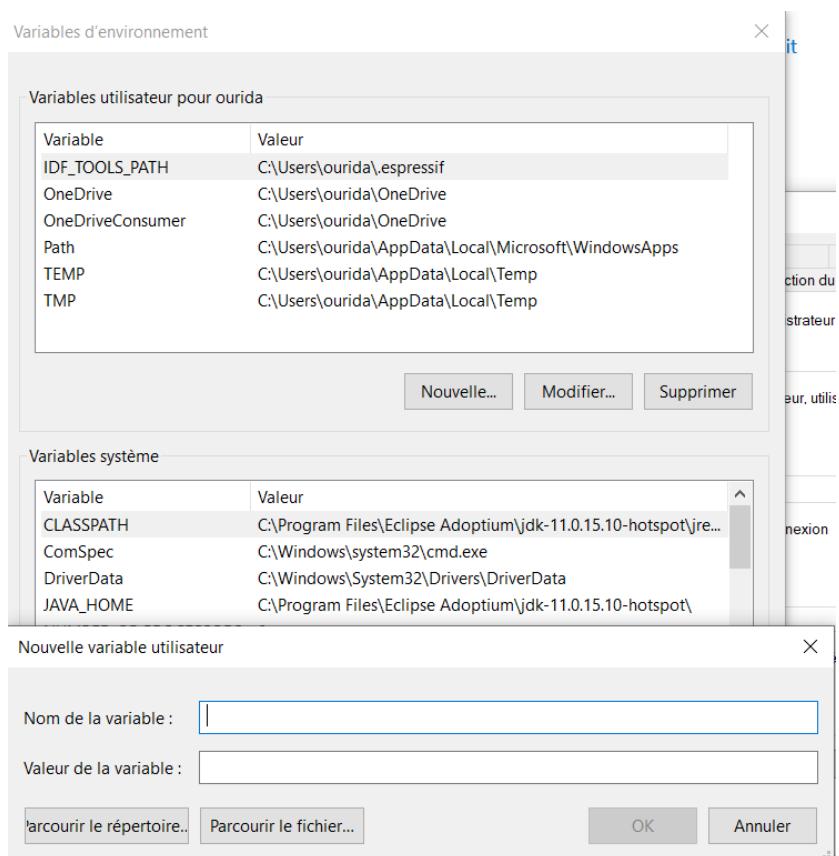


Figure 13: création de la nouvelle variable.

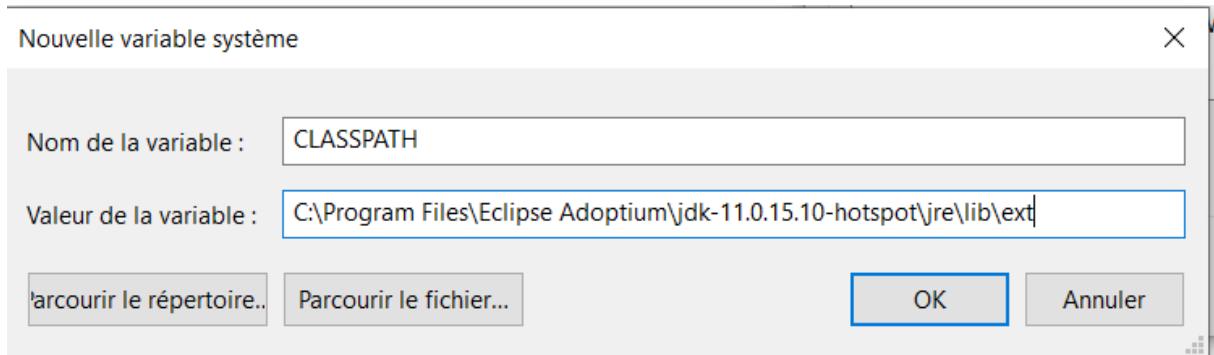
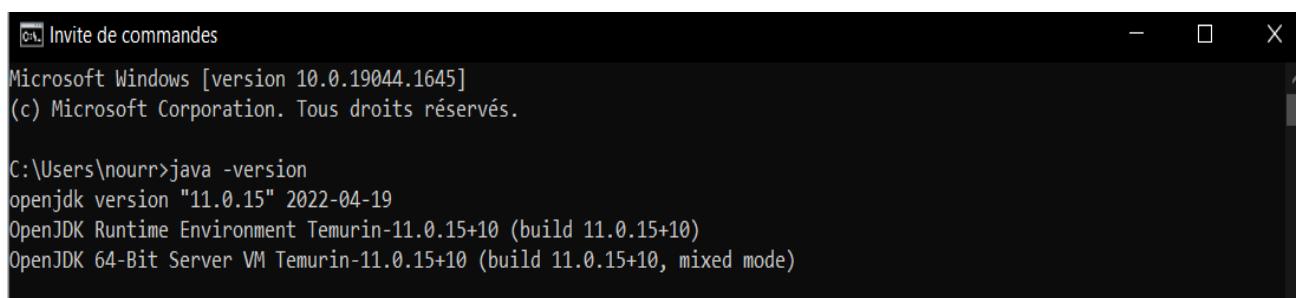


Figure 14: Insertion des valeurs de création de la variable.

- Vérification de l'installation sur un terminal en exécutant la commande java -version :



```
Microsoft Windows [version 10.0.19044.1645]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\nourr>java -version
openjdk version "11.0.15" 2022-04-19
OpenJDK Runtime Environment Temurin-11.0.15+10 (build 11.0.15+10)
OpenJDK 64-Bit Server VM Temurin-11.0.15+10 (build 11.0.15+10, mixed mode)
```

Figure 15: Vérification de l'installation de java.

b) Installation du service Thingsboard

-Il faut commencer par le téléchargement du package et l'extraire sur le lien suivant :

<https://github.com/thingsboard/thingsboard/releases/download/v3.3.4.1/thingsboard-windows-3.3.4.1.zip>

-Décompresser puis déplacer le dossier Thingsboard vers cet emplacement :

C:\Program Files (x86).

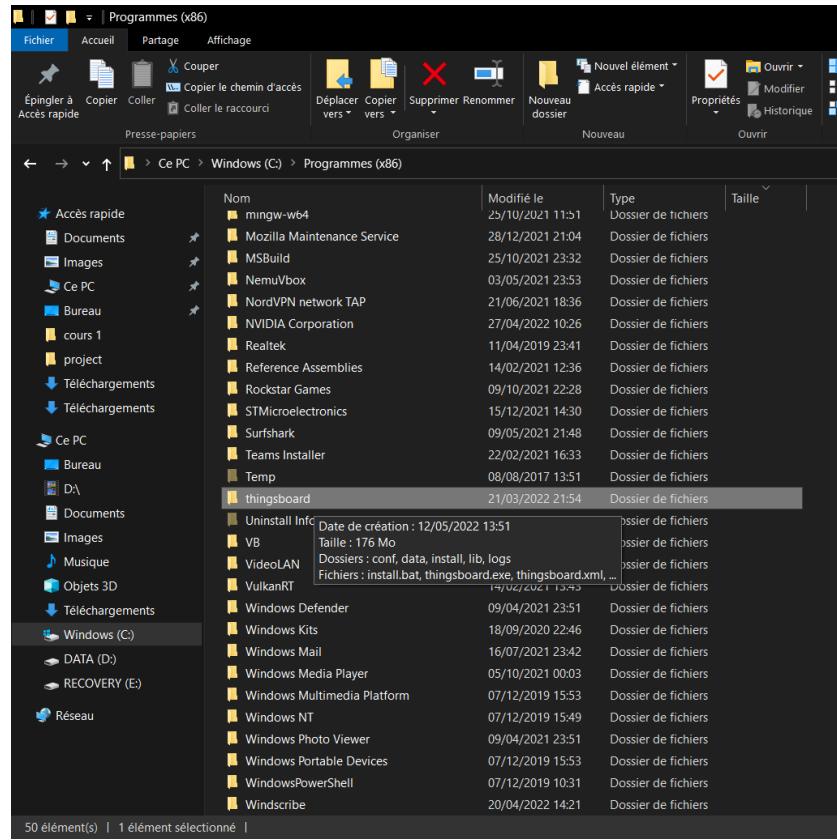


Figure 16: Dossier de Thingsboard

c) Configuration de la base de données Thingsboard

Nous allons choisir pour notre application d'utiliser la base de données PostgreSQL pour une transmission d'environ 5000 messages par seconde, ce qui est largement suffisant pour nous.

Il faut aller sur le lien suivant,

<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads#windows>

Nous avons pris la version 11.16 de Postgres, le choix d'une version plus récente ne posera problème.

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
14.3	postgresql.org	postgresql.org	postgresql.org	postgresql.org	Not supported
13.7	postgresql.org	postgresql.org	postgresql.org	postgresql.org	Not supported
12.11	postgresql.org	postgresql.org	postgresql.org	postgresql.org	Not supported
11.16	postgresql.org	postgresql.org	postgresql.org	postgresql.org	Not supported
10.21	postgresql.org				
9.6.24*	postgresql.org				

Figure 17: Postgresql version 11.16

Lancer l'exécuteur :

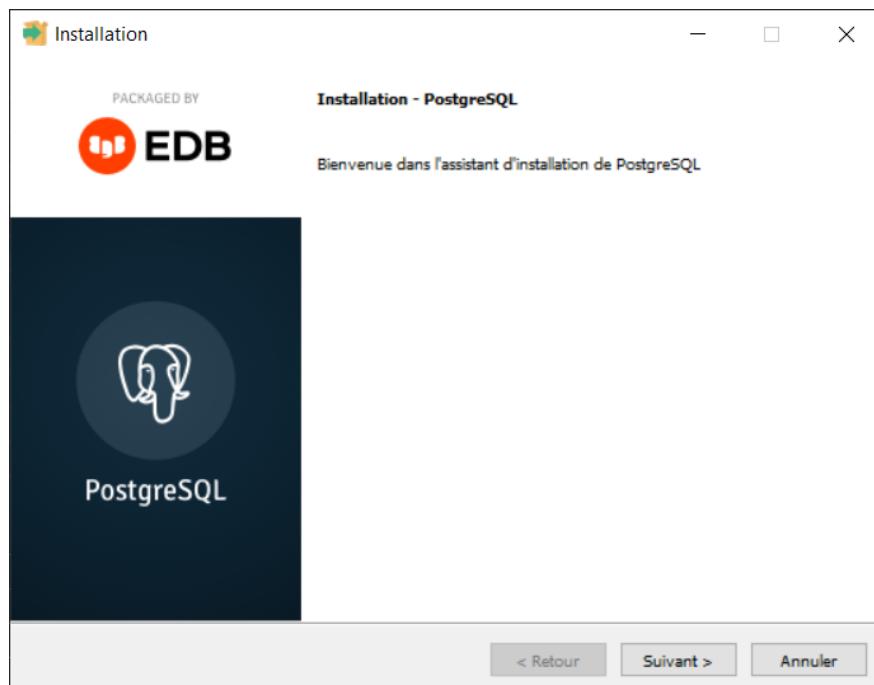


Figure 18: Lancement de l'exécuteur

Dans la figure suivante, nous devons choisir les éléments de gestion de la base de données, en cochant sur les cases qui s'affichent. Dans le cas contraire il faudra les installer à part.

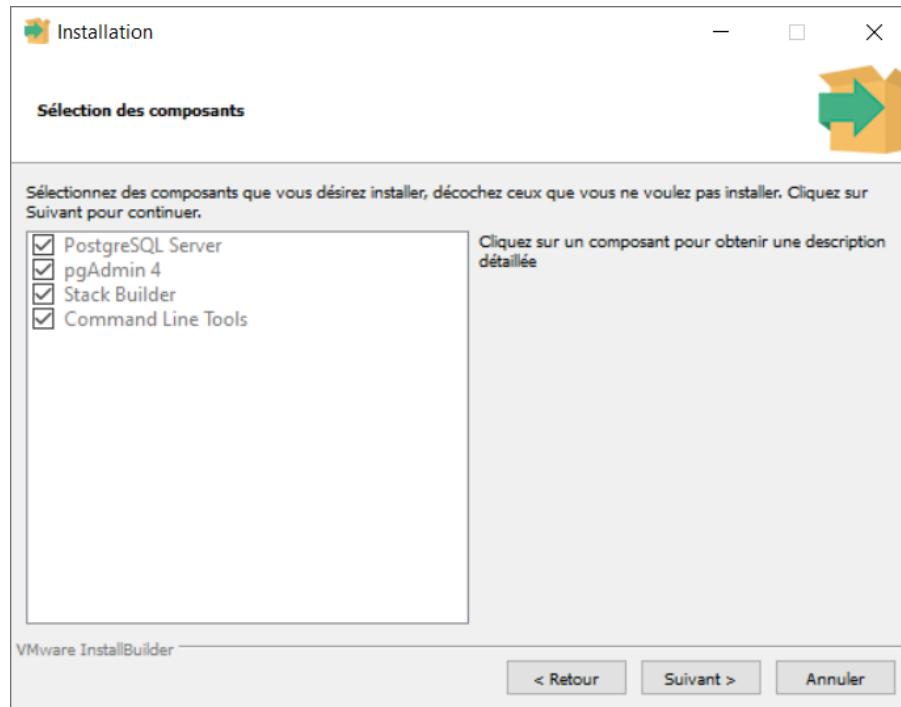


Figure 19: Étape 1 installation de PostgreSQL

Un mot de passe sera demandé pour la création d'un compte super utilisateur et pouvoir manipuler la base de données, mettre « postges » par défaut.

En cas de personnalisation du mot de passe une configuration sera requise après l'installation.

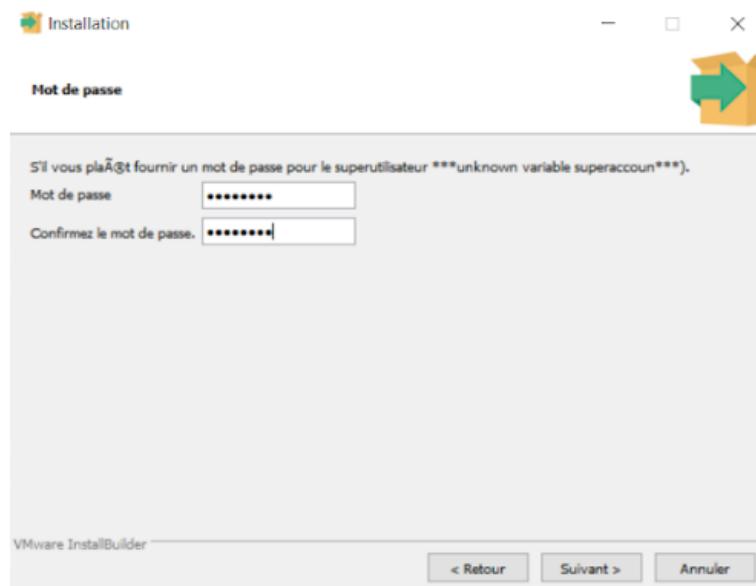


Figure 20: Insertion du mot de passe par défaut

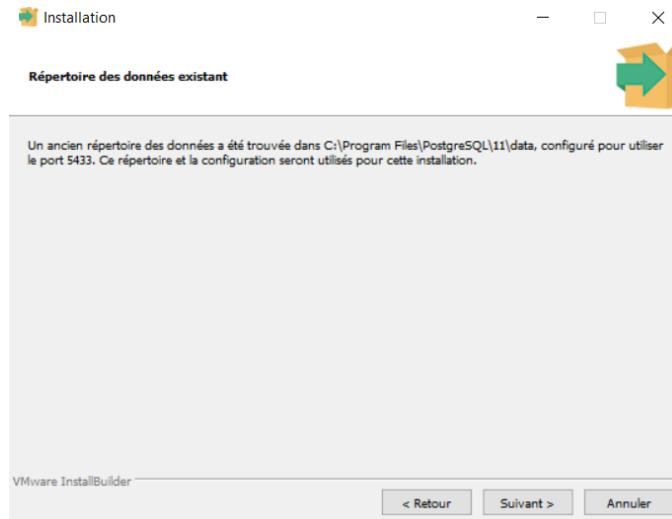


Figure 21: Étape 2 installations de PostgreSQL

Le port 5433 indiqué ci-dessus et sera utile en cas de problème pour l'installation de Thingsboard, donc il faut le noter.

Utiliser un serveur par défaut :

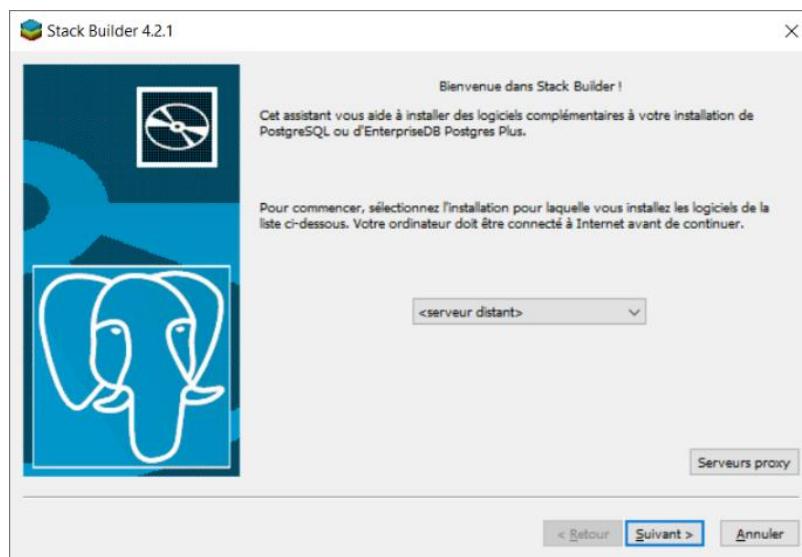


Figure 22: Installation des logiciels complémentaires avec stack Builder

Cette étape sert à ajouter des applications supplémentaires à la base de données.

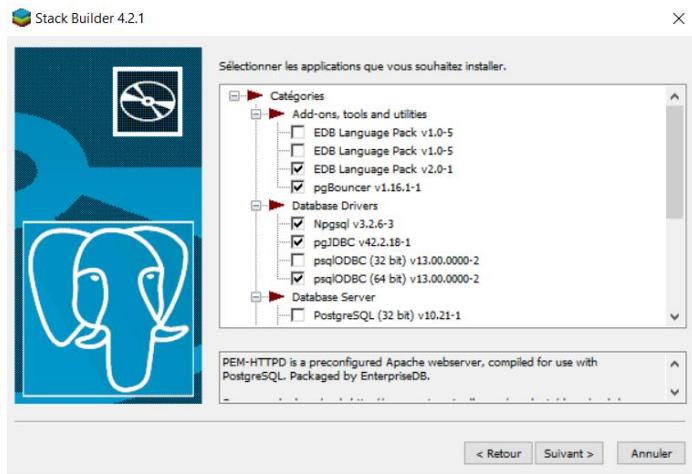


Figure 23: L'ajout des outils à la base de données.

Pour procéder à la création de la base de données, il faut ouvrir pgAdmin, et insérer le mot de passe inséré durant l'installation.

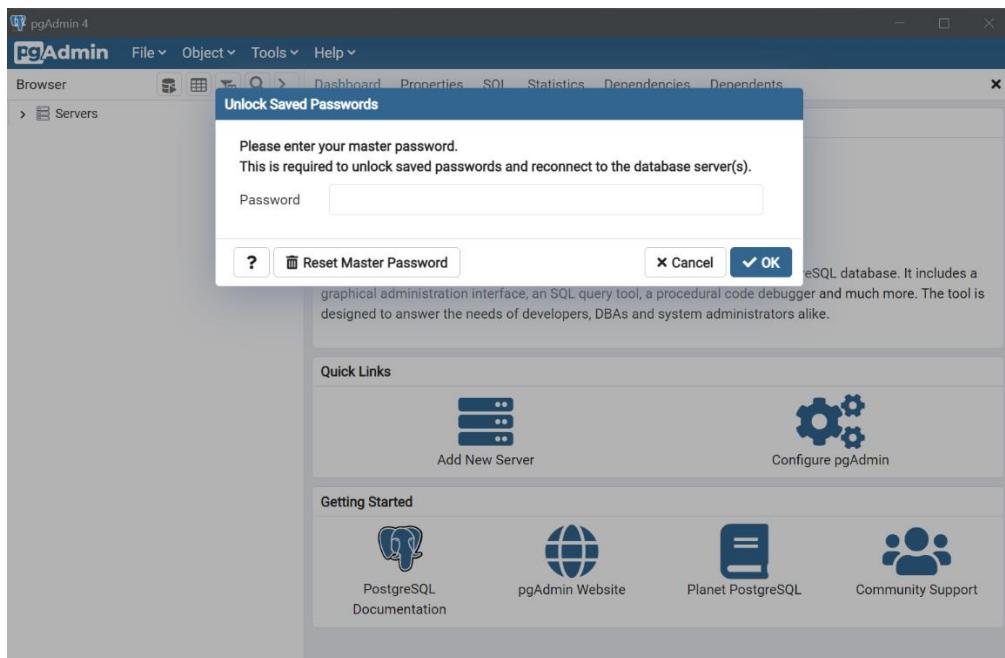


Figure 24: Lancement de PgAdmin4.

Il faut aller dans serveurs puis cliquez droit sur **Databases->Create->Database**

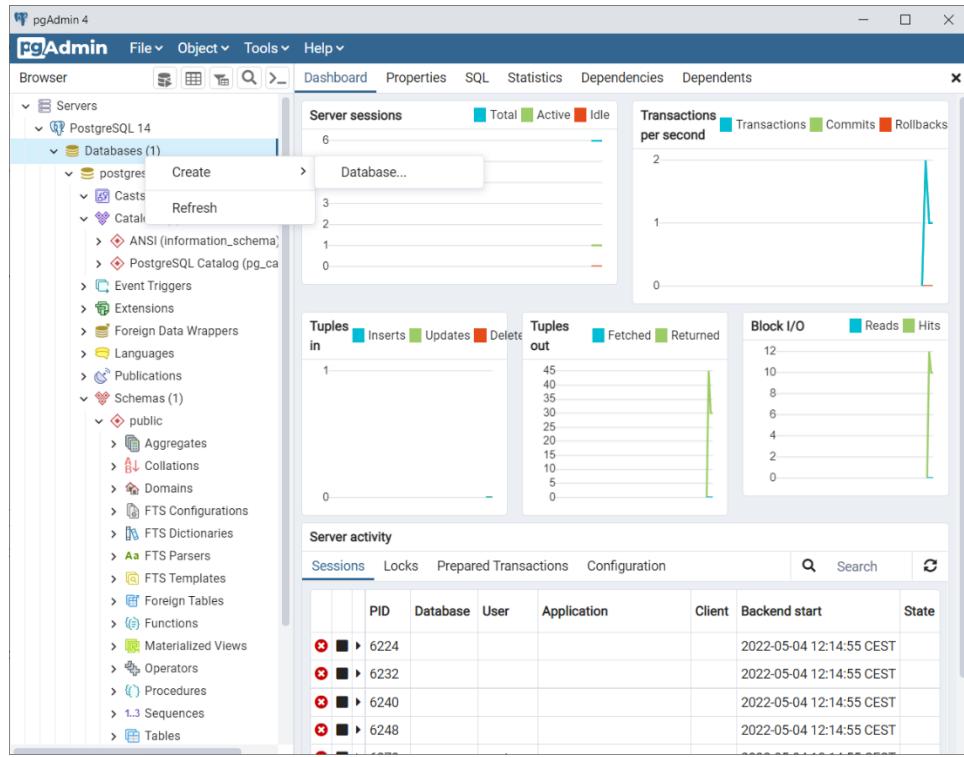


Figure 25: création de la base de données

La figure suivante nous montre la création de notre base de données nommée « thingsboard ». Il faut impérativement la nommer ainsi pour la première utilisation de la plateforme, dans le cas contraire la base de données ne sera pas reconnue et l'installation ne pourra pas s'effectuer.

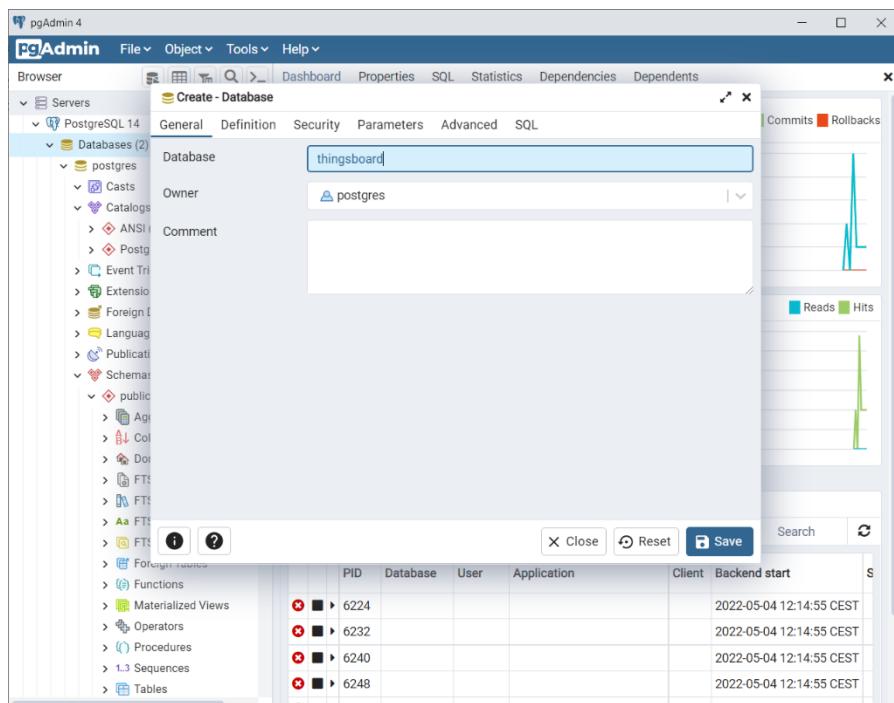


Figure 26: Création de database

Voilà notre base de données :

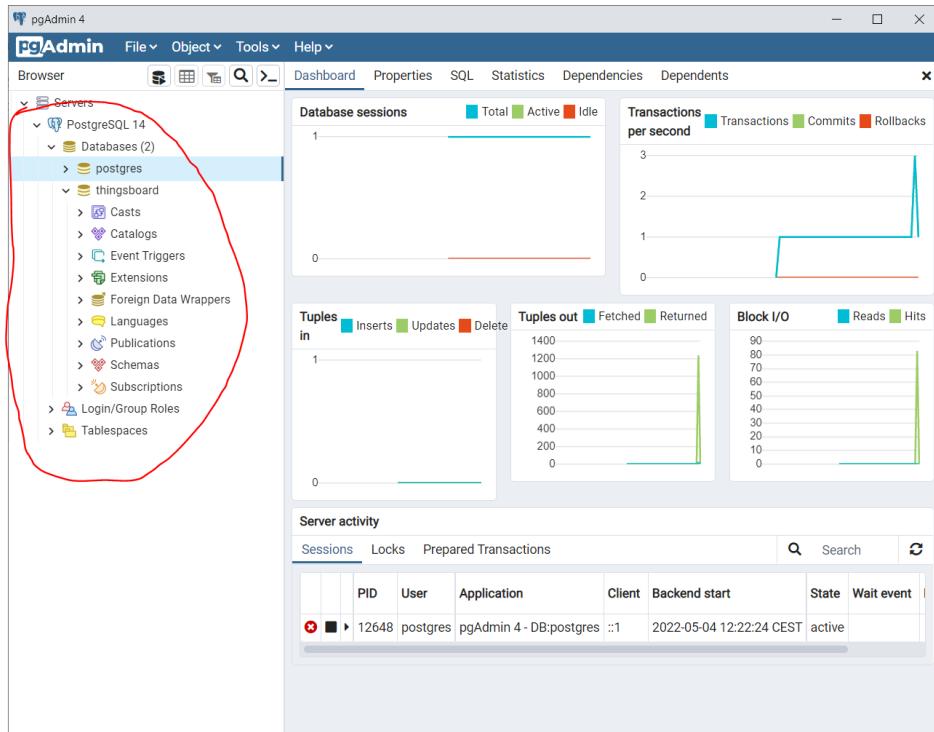


Figure 27: Base de données créée

d) Implémentation de la file d'attente du serveur en mémoire

Il faut exécuter un bloc-notes en mode administrateur afin de pouvoir apporter des modifications au fichier d'installation si nécessaire.

Allez dans le répertoire suivant : C:\Program Files (x86)\thingsboard\conf

Ouvrez le fichier **thingsboard.yml** dans un bloc-notes.

Il faut vérifier si le bon numéro de port est affecté, il faut qu'il soit le même que celui affecté lors de l'installation.

```

thingsboard.yml - Bloc-notes
Fichier Edition Format Affichage Aide
strategy:
  content:
    enabled: "true"

spring.servlet.multipart.max-file-size: "50MB"
spring.servlet.multipart.max-request-size: "50MB"

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation: "true"
spring.jpa.properties.hibernate.order_by.default_null_ordering: "${SPRING_JPA_PROPERTIES_HIBERNATE_ORDER_BY_DEFAULT_NULL_ORDERING:last}"

# SQL DAO Configuration
spring:
  data:
    jpa:
      repositories:
        enabled: "true"
      properties:
        javax.persistence.query.timeout: "${JAVAX_PERSISTENCE_QUERY_TIMEOUT:30000}"
        open-in-view: "false"
      hibernate:
        ddl-auto: "none"
      database-platform: "${SPRING_JPA_DATABASE_PLATFORM:org.hibernate.dialect.PostgreSQLDialect}"
  datasource:
    driverClassName: "${SPRING_DRIVER_CLASS_NAME:org.postgresql.Driver}"
    url: "${SPRING_DATASOURCE_URL:jdbc:postgresql://localhost:5432/thingsboard}"
    username: "${SPRING_DATASOURCE_USERNAME:postgres}"
    password: "${SPRING_DATASOURCE_PASSWORD:postgres}"
    hikari:
      maximumPoolSize: "${SPRING_DATASOURCE_MAXIMUM_POOL_SIZE:16}"

# Audit log parameters
audit-log:
  # Enable/disable audit log functionality.
  enabled: "${AUDIT_LOG_ENABLED:true}"
  # Specify partitioning size for audit log by tenant id storage. Example MINUTES, HOURS, DAYS, MONTHS
  by_tenant_partitioning: "${AUDIT_LOG_BY_TENANT_PARTITIONING:MONTHS}"
  # Number of days as history period if startTime and endTime are not specified
  default_query_period: "${AUDIT_LOG_DEFAULT_QUERY_PERIOD:30}"
  # Logging levels per each entity type.
  # Allowed values: OFF (disable), W (log write operations), RW (log read and write operations)
  logging-level:
    <          >
Ln 529, Col 67   100%   Windows (CRLF)   UTF-8

```

Figure 28: Gestion du port de Postgres

-Aller sur Windows Power Shell en mode administrateur puis exéutez les commandes suivantes :

Set-Location 'C:\Program Files (x86)'

Cd thingsboard

./install.bat

```
PS C:\> Set 'C:\Program Files (x86)\thingsboard'
PS C:\> Set-Location 'C:\Program Files (x86)'
PS C:\Program Files (x86)> cd thingsboard
PS C:\Program Files (x86)\thingsboard> ./install.bat
Detecting Java version installed.
CurrentVersion 110
Java 11 found!
Installing thingsboard ...
=====
:: ThingsBoard ::      (v3.3.4.1)
=====

Starting ThingsBoard Installation...
Installing DataBase schema for entities...
Installing SQL DataBase schema part: schema-entities.sql
Installing SQL DataBase schema indexes part: schema-entities-idx.sql
Installing SQL DataBase schema PostgreSQL specific indexes part: schema-entities-idx-psql-addon.sql
Installing DataBase schema for timeseries...
Installing SQL DataBase schema part: schema-ts-psql.sql
Successfully executed query: CREATE TABLE IF NOT EXISTS ts_kv_indefinite PARTITION OF ts_kv DEFAULT;
Loading system data...
Installation finished successfully!
2022-05-10 10:59:13,398 INFO - Starting ServiceWrapper in the CLI mode
2022-05-10 10:59:15,651 INFO - Completed. Exit code is 0
ThingsBoard installed successfully!
PS C:\Program Files (x86)\thingsboard> ./install.bat
Detecting Java version installed.
CurrentVersion 110
Java 11 found!
Installing thingsboard ...
=====
:: ThingsBoard ::      (v3.3.4.1)
=====
```

Figure 29: Installation de Thingsboard sur Windows Power Shell

Un problème peut survenir à cette étape par rapport au numéro de port affecté au serveur, par défaut c'est le port 8080 en cas de problème avec ce port, il faut utiliser les commandes suivantes afin de pouvoir le changer :

npx kill-port 8080

netstat -ano | findstr :8080

taskill 11572

netstat -ano | findstr :8080 , puis refaire les commandes suivantes :

Set-Location 'C:\Program Files (x86)'

Cd thingsboard

./install.bat

-Le premier lancement de thingsboard s'effectue sur un terminal en utilisant la commande suivante : `net start thingsboard`

-Une fois le premier lancement est fait, il faut allersur le lien suivant :
<http://localhost:8080/login>

Le numéro de port indiqué dans le lien est celui qui correspond au serveur de ThingsBoard, au cas de son changement lors de l'installation, il faut le changer sur le lien aussi.

Pour se connecter à l'interface de thingsboard, la documentation nous propose 3 adresses mail par défaut qui sont les suivants :

La connexion avec les premières coordonnées nous permet d'avoir un accès à la plateforme en tant qu'administrateur du système pouvant créer les administrateurs en tant que tenants.

System Administrator: sysadmin@thingsboard.org / sysadmin (ce sont les logs pour le localhost)

Nous arrivons à nous connecter qu'avec System Administrator et donc nous n'avons pas tous les outillages et blocs nécessaires pour démarrer notre configuration :

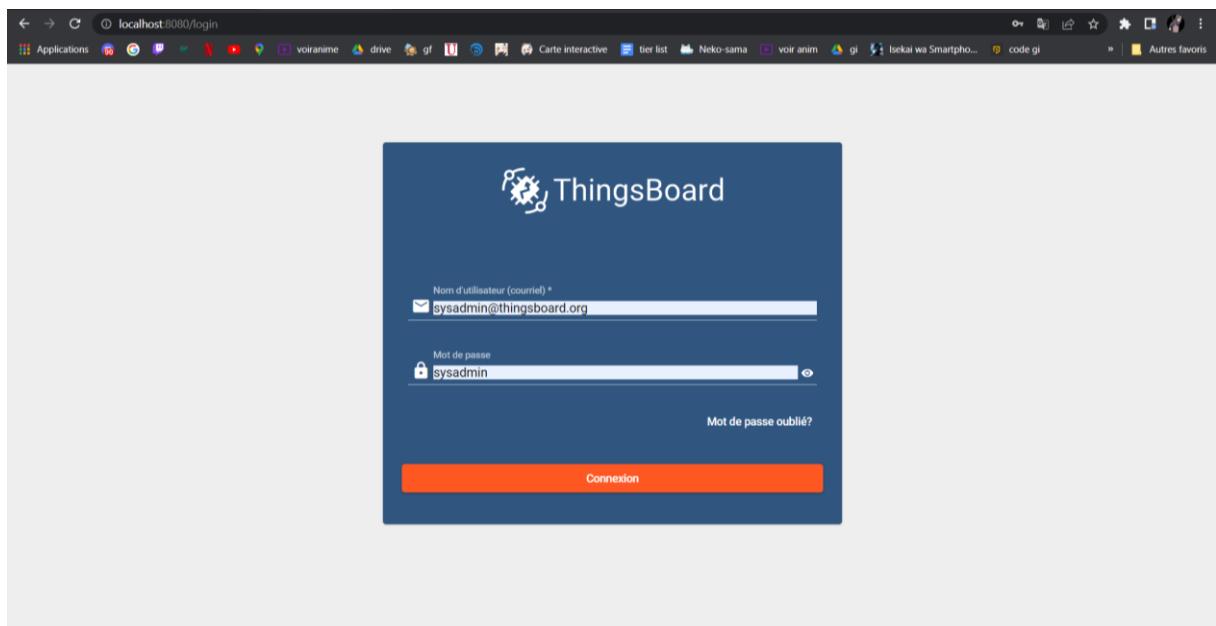


Figure 30: Connexion à Thingsboard

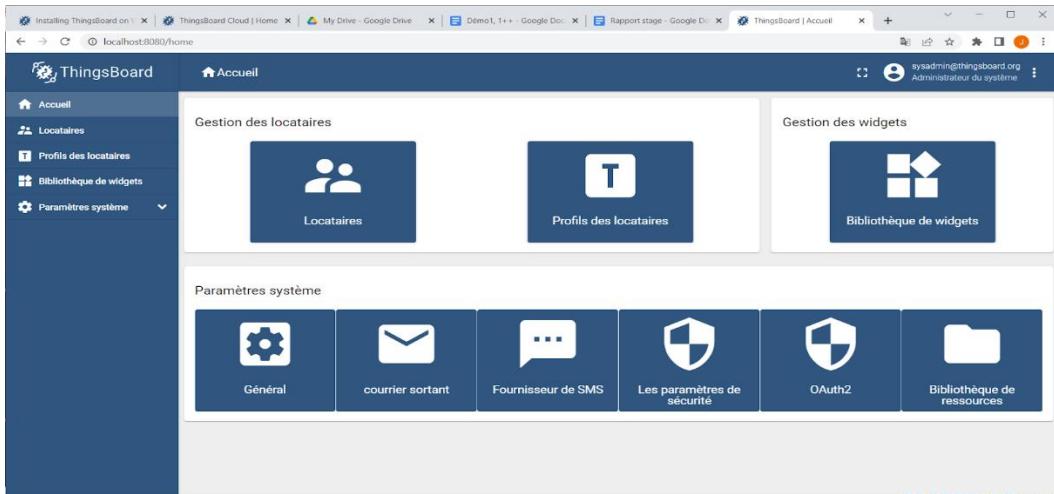


Figure 31: Accès à la plateforme.

-À présent, nous devons créer un tenant administrateur en appuyant sur case « Tenants »

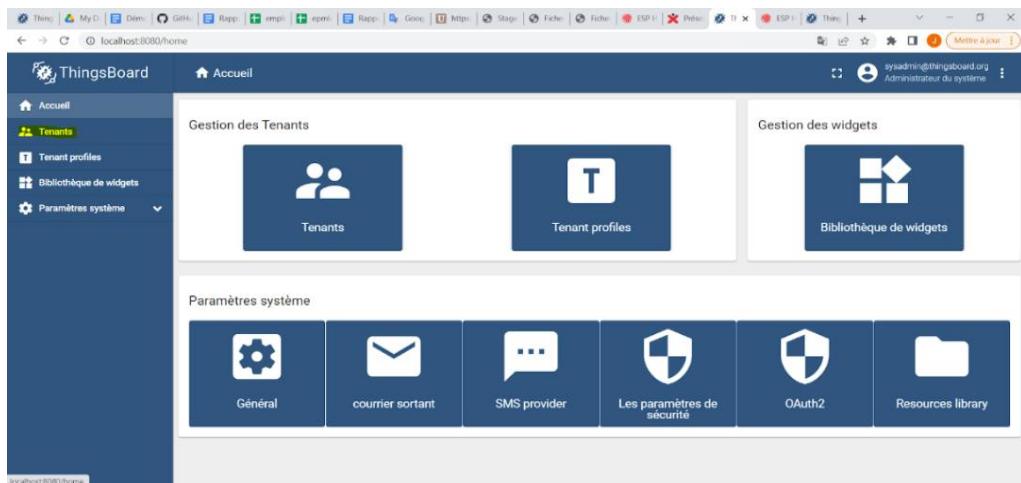


Figure 32: Ajout d'un Tenant

Ajoutez un tenant :

Tenants						
	Heure de création	Titre	Tenant profile	Email	Pays	Ville
<input type="checkbox"/>	2022-05-10 11:17:21	Tenant Test	Default			
<input type="checkbox"/>	2022-05-05 17:01:19	tenant	Default			

Figure 33: Tenant créé

Création du tenant administrateur appartenant à ml l'administrateur du système :

The screenshot shows the ThingsBoard web interface. The left sidebar has links for Accueil, Tenants, Tenant profiles, Bibliothèque de widgets, and Paramètres système. The main content area is titled 'Tenants > administrateurs du Tenant' and shows 'Tenant Test: administrateurs du Tenant'. It includes a search bar with filters for 'Heure de création' (dropdown), 'Prénom' (text input), 'Nom de famille' (text input), and 'Email' (text input). Below the search bar is a table with one column labeled 'Email'. A message 'Aucun utilisateur trouvé' is displayed. At the bottom right of the table are buttons for 'Items per page' (set to 10), '0 of 0', and navigation arrows.

Figure 34: Création d'administrateurs

En ajoutant le mail qui était donné par défaut : tenant@thingsboard.org / tenant

The screenshot shows the 'Ajouter un utilisateur' dialog box over the ThingsBoard interface. The dialog has fields for 'Email*' (containing 'tenant@thingsboard.org'), 'Prénom', 'Nom de famille', and 'Description'. Under 'Méthode d'activation', it says 'Afficher le lien d'activation'. At the bottom are 'Annuler' and 'Ajouter' buttons. The background shows the same 'Tenants > administrateurs du Tenant' page as Figure 34, with the search bar and table visible.

Figure 35: champs d'utilisateur à renseigner.

Rentrez dans un tenant administrateur :

The screenshot shows a browser window with the URL `localhost:8080/tenants/384c8110-cc84-11ec-aaac-f344dcb5708/users`. The title bar says "administreurs du Tenant". The left sidebar has "Accueil", "Tenants", "Tenant profiles", "Bibliothèque de widgets", and "Paramètres système". The main content area shows a table with one row for "tenant: administrateurs du Tenant". The table columns are "Heure de création" (Creation time), "Prénom" (First name), "Nom de famille" (Last name), and "Email". The creation time is "2022-05-05 17:04:35", the first name is "tenant", the last name is "", and the email is "tenant@thingsboard.org". There is a yellow "Mettre à jour" (Update) button at the bottom right of the table.

Figure 36: Accès à la plateforme admin.

NB : Nous pouvons utiliser le mail par défaut qu'une seule fois.

À la fin nous aurons la page d'accueil suivante avec tous les blocs nécessaires pour l'utilisation de **Thingsboard**, comme le montre la figure suivante :

The screenshot shows a browser window with the URL `localhost:8080/home`. The title bar says "Accueil". The left sidebar has "Accueil", "Chaines de règles", "Clients", "Actifs", "Dispositifs", "Device profiles", "OTA updates", "Vues d'entité", "Instances de Bord", "Gestion des bordures", "Bibliothèque de widgets", "Tableaux de bord", "Journaux d'audit", "Api Usage", and "Paramètres système". The main content area is divided into several sections: "Gestion des règles" (Rules), "Gestion des clients" (Clients), "Gestion d'actifs" (Assets), "Gestion des dispositifs" (Devices), "Device profiles", and "OTA updates". Each section contains a large blue button with an icon and text.

Figure 37: Blocs de ThingsBoard

e) Choisissez le service de file d'attente ThingsBoard

ThingsBoard est capable d'utiliser divers systèmes/courtiers de messagerie pour stocker les messages et la communication entre les services ThingsBoard.

L'implémentation de la file d'attente en mémoire : est intégrée et par défaut. Il est utile pour les environnements de développement (PoC) et ne convient pas aux déploiements de production ou à tout type de déploiements de clusters.

Kafka est recommandé pour les déploiements en production. Cette file d'attente est désormais utilisée dans la plupart des environnements de production ThingsBoard. Il est utile pour les déploiements sur site et dans le cloud privé. C'est également utile si vous souhaitez rester indépendant de votre fournisseur de cloud. Cependant, certains fournisseurs ont également des services gérés pour Kafka. Voir AWS [MSK](#) par exemple.

RabbitMQ est recommandé si vous n'avez pas beaucoup de charges et que vous avez déjà une expérience avec ce système de messagerie.

AWS SQS est un service de file d'attente de messages entièrement géré d'AWS. Utile si vous envisagez de déployer ThingsBoard sur AWS.

Google Pub/Sub est un service de file d'attente de messages entièrement géré de Google. Utile si vous prévoyez de déployer ThingsBoard sur Google Cloud.

Azure Service Bus est un service de mise en file d'attente de messages entièrement géré d'Azure. Utile si vous envisagez de déployer ThingsBoard sur Azure.

Confluent Cloud est une plateforme de streaming entièrement gérée basée sur Kafka. Utile pour les déploiements indépendants du cloud.

Pour notre projet, nous avons utilisé kafka :

f) Implémentation du service de file d'attente Kafka

L'implémentation en mémoire par défaut est suffisante pour le fonctionnement de la plateforme.

En cas d'utilisation plus avancée de l'environnement nous avons choisi Kafka pour les raisons suivantes :

Kafka est recommandé pour les déploiements en production. Cette file d'attente est désormais utilisée dans la plupart des environnements de production Thingsboard. Il est utile pour les déploiements sur site et dans le cloud privé. C'est également utile si vous souhaitez rester indépendant de votre fournisseur de cloud. Cependant, certains fournisseurs ont également des services gérés pour Kafka. Voir AWS [MSK](#) par exemple.

En choisissant la dernière version :

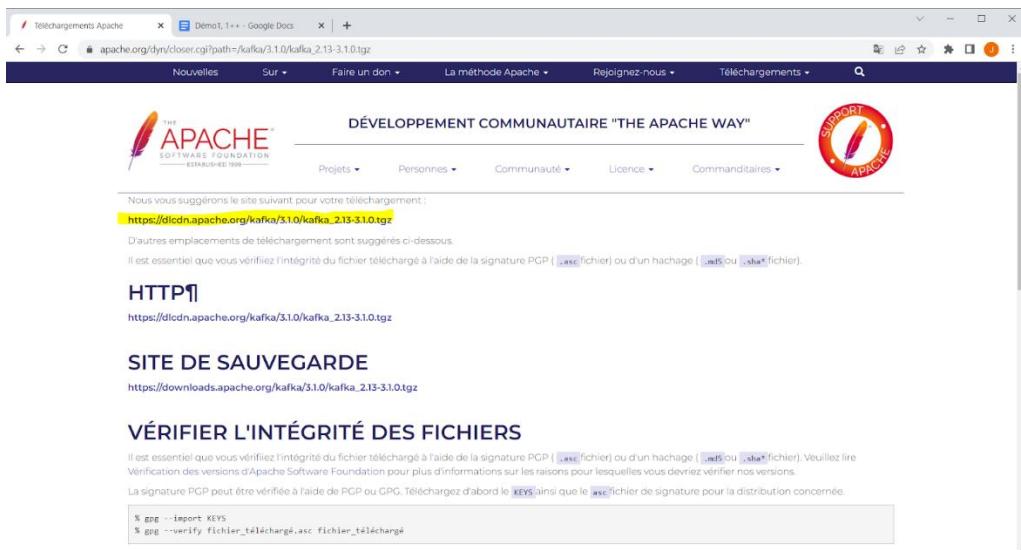


Figure 38: Lien pour installer kafka

Lien : https://apache.org/dyn/closer.cgi?path=/kafka/3.1.0/kafka_2.13-3.1.0.tgz

Renommez le dossier kafka et le déplacez sur le C:\

Lien : <https://github.com/wurstmeister/kafka-docker>

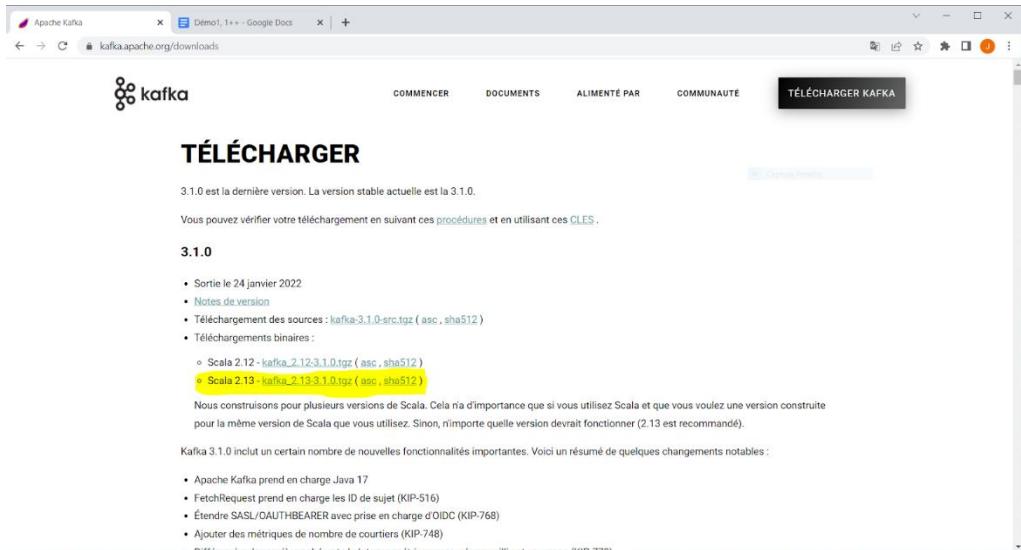


Figure 39: Kafka .tgz

<https://kafka.apache.org/downloads>

```

Administrator : Windows PowerShell
Afficher uniquement les noms de distribution.

--verbose, -v
    Affiche les informations détaillées sur toutes les distributions.

--online, -o
    Affiche une liste des distributions disponibles pour l'installation avec « wsl --install ».

--set-default, -s <Distribution>
    Définit la distribution comme valeur par défaut.

--set-version <Distribution> <Version>
    Modifie la version de la distribution spécifiée

--terminate, -t <Distribution>
    Termine la distribution spécifiée.

--unregister <Distribution>
    Annule l'inscription de la distribution et supprime le système de fichiers racine.

PS C:\Windows\system32> dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

Outil Gestion et maintenance des images de déploiement
Version : 10.0.19041.844

Version de l'image : 10.0.19044.1586

Activation de la ou des fonctionnalités
[=====100.0%=====]
L'opération a réussi.

PS C:\Windows\system32> dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

Outil Gestion et maintenance des images de déploiement
Version : 10.0.19041.844

Version de l'image : 10.0.19044.1586

Activation de la ou des fonctionnalités
[=====100.0%=====]
L'opération a réussi.

PS C:\Windows\system32> wsl --set-default-version 2
Pour plus d'informations sur les différences de clés avec WSL 2, visitez https://aka.ms/wsl2
L'opération a réussi.

PS C:\Windows\system32> wsl --set-default-version 2
Pour plus d'informations sur les différences de clés avec WSL 2, visitez https://aka.ms/wsl2
L'opération a réussi.

PS C:\Windows\system32>

```

Figure 40: Installation sous PowerShell

-Installation kafka:

Les commandes demandées :

Port de kafka: 2181

Partir dans le cmd en mode admin, il faut exécuter de nouveau ces commandes :

cd C:\kafka

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

```
C:\kafka>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

Figure 41: zookeeper.properties

2. Installation sous Linux

a) Installez Java 11 (OpenJDK)

Le service ThingsBoard s'exécute sur Java 11. Suivez ces instructions pour installer OpenJDK.

```
sudo apt update  
sudo apt install openjdk-11-jdk
```

N'oubliez pas de configurer votre système d'exploitation pour utiliser OpenJDK 11 par défaut. Vous pouvez configurer la version par défaut à l'aide de la commande suivante :

```
sudo update-alternatives --config Java
```

Vous pouvez vérifier l'installation à l'aide de la commande suivante :

```
java -version
```

La sortie de commande attendue est :

```
1 openjdk version "11.0.xx"  
2 OpenJDK Runtime Environment (...)  
3 OpenJDK 64-Bit Server VM (build ...)
```

b) Installation du service ThingsBoard

Télécharger le package d'installation :

```
wget https://github.com/thingsboard/thingsboard/releases/download/v3.3.4.1/thingsboard-3.3.4.1.deb
```

Installer ThingsBoard en tant que service :

```
sudo dpkg -i thingsboard-3.3.4.1.deb
```

c) Configurer la base de données ThingsBoard

- Installation de PostgreSQL :

- Installer **wget** s'il n'est pas déjà installé :

```
sudo apt install -y wget
```

- Importer la clé de signature du référentiel :

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

- Ajouter le contenu du référentiel à votre système :

```
RELEASE=$(lsb_release -cs)
```

```
echo "deb http://apt.postgresql.org/pub/repos/apt/ ${RELEASE}-pgdg main" | sudo tee  
/etc/apt/sources.list.d/pgdg.list
```

- Installer et lancer le service postgresql :

```
sudo apt update
```

```
sudo apt -y install postgresql-12
```

```
sudo service postgresql start
```

Une fois PostgreSQL installé, vous souhaiterez peut-être créer un nouvel utilisateur ou définir le mot de passe de l'utilisateur principal. Les instructions ci-dessous vous aideront à définir le mot de passe pour l'utilisateur principal de postgresql :

```
sudo su - postgres
```

```
psql
```

```
\password
```

```
\q
```

Ensuite, appuyez sur "Ctrl + D" pour revenir à la console utilisateur principale et vous connecter à la base de données pour créer la base de données Thingsboard :

```
psql -U postgres -d postgres -h 127.0.0.1 -W
```

```
CREATE DATABASE thingsboard.
```

```
\q
```

d) Configuration de ThingsBoard

Modifier le fichier de configuration ThingsBoard

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

Ajoutez les lignes suivantes au fichier de configuration. N'oubliez pas **de remplacer "PUT_YOUR_POSTGRESQL_PASSWORD_HERE"** par votre **vrai mot de passe utilisateur postgres**

```
# DB Configuration
export DATABASE_TS_TYPE=sql
export SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect
export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
export SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=PUT_YOUR_POSTGRESQL_PASSWORD_HERE
# Specify partitioning size for timestamp key-value storage. Allowed values: DAYS, MONTHS, YEARS, INDEFINITE.
export SQL_POSTGRES_TS_KV_PARTITIONING=MONTHS
```

e) Installation de Kafka

[Apache Kafka](#) est une plate-forme logicielle open source de traitement de flux.

Installer ZooKeeper

Kafka utilise [ZooKeeper](#), vous devez donc d'abord installer le serveur ZooKeeper :

```
sudo apt-get install zookeeper
```

Installer Kafka:

```
wget https://archive.apache.org/dist/kafka/2.6.0/kafka_2.13-2.6.0.tgz
```

```
tar xzf kafka_2.13-2.6.0.tgz
```

```
sudo mv kafka_2.13-2.6.0 /usr/local/kafka
```

f) Configurer le fichier ZooKeeper Systemd Unit

Créez un fichier d'unité systemd pour Zookeeper :

```
sudo nano /etc/systemd/system/zookeeper.service
```

Ajoutez ci-dessous le contenu :

```
[Unit]
```

```
Description=Apache Zookeeper server
```

```
Documentation=http://zookeeper.apache.org
```

```
Requires=network.target remote-fs.target
```

```
After=network.target remote-fs.target
```

```
[Service]
```

```
Type=simple
```

```
ExecStart=/usr/local/kafka/bin/zookeeper-server-start.sh  
/usr/local/kafka/config/zookeeper.properties
```

```
ExecStop=/usr/local/kafka/bin/zookeeper-server-stop.sh
```

```
Restart=on-abnormal
```

[Install]

```
WantedBy=multi-user.target
```

g) Configurer le fichier Kafka Systemd Unit

Créez un fichier d'unité systemd pour Kafka :

```
sudo nano /etc/systemd/system/kafka.service
```

Ajoutez le contenu ci-dessous. Assurez-vous **de remplacer "PUT_YOUR_JAVA_PATH"** par votre **vrai chemin JAVA_HOME** selon le Java installé sur votre système, par défaut comme "/usr/lib/jvm/java-11-openjdk-xxx":

[Unit]

```
Description=Apache Kafka Server
```

```
Documentation=http://kafka.apache.org/documentation.html
```

```
Requires=zookeeper.service
```

[Service]

```
Type=simple
```

```
Environment="JAVA_HOME=PUT_YOUR_JAVA_PATH"
```

```
ExecStart=/usr/local/kafka/bin/kafka-server-start.sh /usr/local/kafka/config/server.properties
```

```
ExecStop=/usr/local/kafka/bin/kafka-server-stop.sh
```

[Install]

WantedBy=multi-user.target

Démarrez ZooKeeper et Kafka

```
sudo systemctl start zookeeper
```

```
sudo systemctl start kafka
```

h) Configuration de ThingsBoard

Modifier le fichier de configuration ThingsBoard

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

Ajoutez la ligne suivante au fichier de configuration. N'oubliez pas de remplacer "localhost:9092" par vos vrais serveurs d'amorçage Kafka :

```
1     export TB_QUEUE_TYPE=kafka  
2     export TB_KAFKA_SERVERS=localhost:9092
```

g) [Facultatif] Mise à jour de la mémoire pour les machines lentes (1 Go de RAM)

Modifier le fichier de configuration ThingsBoard

```
sudo nano /etc/thingsboard/conf/thingsboard.conf
```

Ajoutez les lignes suivantes au fichier de configuration.

```
export JAVA_OPTS="$JAVA_OPTS -Xms256M -Xmx256M"
```

h) Exécutez le script d'installation.

Une fois, le service ThingsBoard installé et la configuration de la base de données misent à jour, vous pouvez exécuter le script suivant :

```
sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo
```

i) Démarrer le service ThingsBoard

Exécutez la commande suivante pour démarrer ThingsBoard :

```
sudo service thingsboard start
```

Une fois démarré, vous pourrez ouvrir Web UI en utilisant le lien suivant :

<http://localhost:8080/>

Les informations d'identification par défaut suivantes sont disponibles si vous avez spécifié *--loadDemo* lors de l'exécution du script d'installation :

Administrateur système : sysadmin@thingsboard.org / sysadmin

Administrateur locataire : tenant@thingsboard.org / tenant

Utilisateur client : customer@thingsboard.org / client

Vous pouvez toujours modifier les mots de passe de chaque compte dans la page de profil du compte.

IV. Les bases de Thingsboard

1. Fonctionnalités de Thingsboard Community

Attributs : capacité de la plate-forme à attribuer des attributs de valeur clé personnalisés à vos entités (par exemple, configuration, traitement des données, paramètres de visualisation).

Explication :

Attribuer des attributs personnalisés aux entités et pouvoir les gérer. Ces attributs seront stockés dans la base de données et peuvent être utilisés pour la visualisation des données.

Les attributs permettent d'accéder aux données et les gérer.

Il existe 3 types d'attributs :

***Attribut côté serveur** : prise en charge par les entités, configurable via l'interface utilisateur d'administration ou **API REST**.

***Attributs partagés** : les dispositifs qui communiquent via un protocole peuvent demander la valeur des attributs.

***Attributs côté client** : disponibles uniquement pour les appareils (clients), l'appareil peut envoyer la valeur des attributs de l'appareil à la plateforme.

4. Utilisation de base des fonctionnalités les de base de ThingsBoard :

-**Connexion d'un appareil à ThingsBoard**

-Faire un push des données vers la plateforme

-Construire des tableaux de bord en temps réel.

-**Provisionner l'appareil :**

Nous allons le faire manuellement à l'aide de l'interface utilisateur.

Remarque : l'approvisionnement de l'appareil peut se faire d'autres manières telles que :

-**Approvisionnement en masse** : pour approvisionner plusieurs appareils à partir d'un fichier CSV à l'aide de l'interface utilisateur ;

-Provisionnement de l'appareil : pour permettre au micrologiciel de l'appareil de provisionner automatiquement l'appareil, vous n'avez donc pas besoin de configurer chaque appareil manuellement ;

-API REST : pour provisionner les appareils et autres entités par programmation.

5. Exemple d'utilisation de la plateforme :

a) Ajouter un nouveau dispositif :

Nous pouvons soit l'ajouter et le configurer sur l'interface, soit l'importer de l'extérieur sous forme d'un fichier csv.

- Ajout d'un nouveau dispositif :**

Dans la case « device profile » on peut afficher les dispositifs par catégories. On procède à l'ajout en cliquant sur le bouton « plus ».

The screenshot shows the ThingsBoard web interface with the sidebar navigation open. The 'Dispositifs' (Devices) option is selected. The main content area is titled 'Dispositifs' and shows a table with one row of data. The columns are: 'Heure de création' (Creation Date), 'Nom' (Name), 'Device profile' (Device profile), 'Label', 'Client', 'Public', and 'Est une passerel' (Is a gateway). The data in the table is: '2022-05-12 15:37:53', 'base', 'default', 'Public', checked, unchecked, and icons for edit, delete, and other actions. At the bottom of the table, there are pagination controls for 'Items per page: 10' and '1 - 1 of 1'.

Figure 42: Dispositifs

Configuration nécessaire : Nous pouvons sélectionner le profil par défaut, dans le cas où nous voudrions créer un nouveau profil hormis celui fourni par défaut, nous devons sélectionner la chaîne de règles « rule chaine », et la « queue name », pour gérer la priorité des blocs dans la chaîne de règles.

Dans notre cas nous avons sélectionné le profil par défaut.

Figure 43: Ajout de dispositif

b) Etablissement de la connexion du dispositif :

Afin de pouvoir connecter le dispositif (device) nous aurons besoin de ses informations d'identification, nous allons utiliser le jeton d'accès (acces Token) qu'il faudra copier.

Figure 44: détail du dispositif

Access token : GDMqk73z0YziWxwW8RVw

c) Création du tableau de bord :

Nous créons tout d'abord un nouveau tableau de bord qui sera attribué à notre dispositif :

A screenshot of a web-based dashboard creation tool. The top navigation bar shows 'Tableaux de bord > Tableau de mon dispositif'. On the right, there's a user profile for 'warda acheroufene Administrateur du Tenant' with a gear icon and three dots. Below the bar, there are icons for search, refresh, and download. The main area is titled 'Tableau de mon dispositif' and has a placeholder 'Titre *'. The background is light gray.

Figure 45: Crédit d'un tableau de bord

d) L'ajout d'un alias associé à l'entité :

Cet alias est considéré comme une référence d'une entité, cette dernière peut être unique ou sous forme d'un groupe d'entités, cet alias peut donc être statique ou dynamique selon l'entité à laquelle il est attribué.

Les alias permettent de définir les données à partir desquelles les entités seront utilisées.

Modifier l'alias

Nom de l'alias * Résoudre en plusieurs entités

Type de filtre *

Type * Dispositif *

Figure 46: Création de l'alias associé à notre dispositif créé.

Nous avons créé un alias en l'attribuant à une entité de type « dispositif » puisque nous avons un seul appareil.

e) Ajout du widget dans le tableau de bord :

Il sera sélectionné dans la bibliothèque de widgets. Afin d'afficher la valeur de la donnée envoyée, nous devons configurer la source de données qui correspond à notre widget.

Le widget sera donc affecté à l'alias précédemment créé, il faut aussi définir le type de la donnée à envoyer (nous pouvons l'ajouter si elle n'existe pas sur dans la liste des suggestions).

Ajouter un widget: Simple card

Données Paramètres Avancé Actions

Sources de données
Maximum 1 datasource est autorisé.

Type	Paramètres
= 1. Entité	Alias de l'entité * <input type="text" value="Température"/> <input type="button" value="= ⚙️ ✎ X"/> Filter <input type="text"/>

Maximum 1 timeseries / attribut est autorisé.

Data settings

Symbol spécial à afficher à côté de la valeur Nombre de chiffres après virgule flottante

Figure 47: Configuration de la source des données.

f) Attribution de l'appareil au client

- *Création du client*

Clients							
<input type="checkbox"/>	Heure de création	Titre	Email	Pays	Ville		
<input type="checkbox"/>	2022-05-05 01:07:06	Mon client		France	Poitiers		

Figure 48: Client

En attribuant l'appareil à notre client, permettra aux utilisateurs qui seront définis par ce dernier à lire et écrire la télémétrie et d'envoyer des commandes aux appareils.

- *Affection du tableau de bord*

En affectant le tableau au client donnera accès à ses utilisateurs pour visualiser l'interface d'affichage des valeurs de données.

<input type="checkbox"/>	Heure de création	Nom	Device profile	Label	Client	Public	Est une passerelle		
<input type="checkbox"/>	2022-05-04 12:22:29	Mon dispositif	default	Température	<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:25	Chargeur			<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:25	Chargeur			<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:25	Air Qualité T1			<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:25	Air Qualité C1			<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:25	Sensor			<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:24	Sensor T1	Temperature Sensor		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2022-05-02 09:30:15	Test Device C1	default		Customer C	<input type="checkbox"/>	<input type="checkbox"/>		
<input type="checkbox"/>	2022-05-02 09:30:15	Test Device B1	default		Customer B	<input type="checkbox"/>	<input type="checkbox"/>		

Affecter des dispositifs au client

Veuillez sélectionner le client pour attribuer le ou les dispositifs

Client *

Figure 49: Affectation d'un client

a) Crédation d'un utilisateur client

Nous allons créer, l'utilisateur appartenant à notre client, cet utilisateur aura accès uniquement à la lecture des données et le tableau de bord.

Pour ajouter un utilisateur, une adresse mail de celui-ci est requise, afin de lui créer son espace sur la plateforme.

totoalice@gmail.com

Détails de l'utilisateur

Email *

totoalice@gmail.com

Prénom

Alice

Nom de famille

TOTO

Description

Tableau de bord par défaut Toujours en plein écran

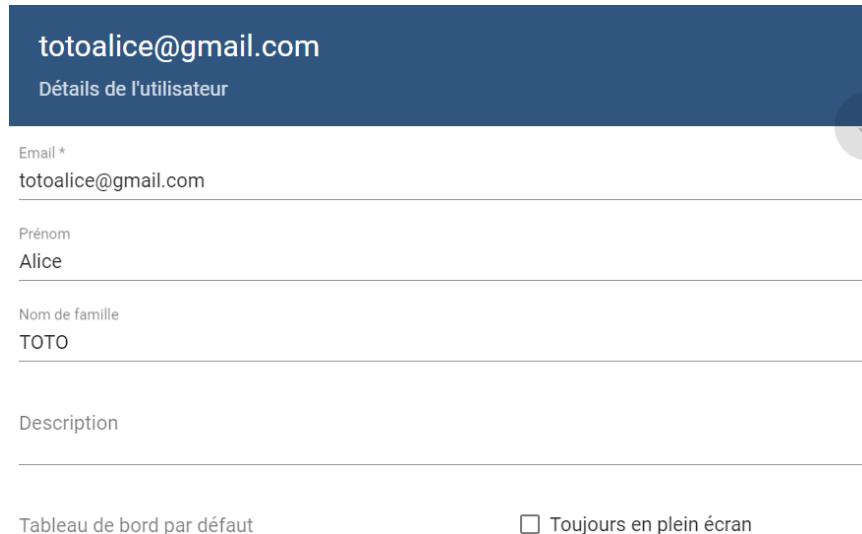


Figure 50: Détail de l'utilisateur

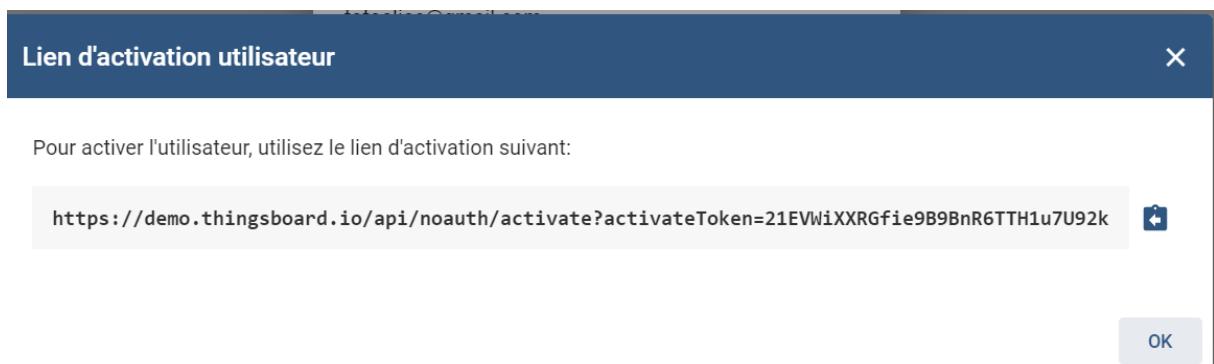


Figure 51: Lien de génération

Le lien d'activation :

<https://demo.thingsboard.io/api/noauth/activate?activateToken=ygqts5rI9jOAQswH05nhIcJJbZlIc>

Un lien d'activation a été transmis et sera utilisé pour le mot de passe de notre utilisateur plus tard.

- *Activation de l'utilisateur client :*

Nous allons utiliser le lien d'activation fourni lors de la création de l'utilisateur.

En se connectant à la plateforme en via le lien, nous pourrons créer le mot de passe et nous connecter en tant qu'utilisateur.



Figure 52: Mot de passe de l'utilisateur

Nous pouvons ainsi accéder à la plateforme :

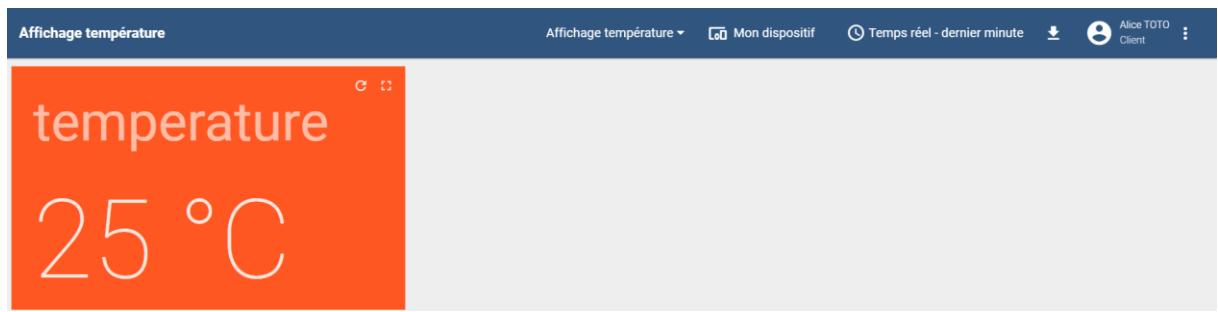


Figure 53: Plateforme à disposition du client

- L'accès peut s'effectuer en utilisant le lien public fourni dans le cas où nous rendons notre dispositif public, les données seront ainsi visibles par tous le monde.

V. Exemple d'utilisation avec Esp32 :

Nous avons utilisé la carte esp32 munie de capteur Température, pression et humidité afin d'envoyer la requête à Thingsboard.

1. Environnement de développement et carte ESP32

Nous utiliserons ESP32-WROVER-KIT et sa communication par Wifi ainsi que sa carte X-NUCLEO-IKS01A3 possède un détecteur de mouvement et un capteur environnemental, elle contient plusieurs capteurs comme température, pression, humidité, accéléromètre et d'autres encore. La communication avec nos capteurs se fera par port I2C.

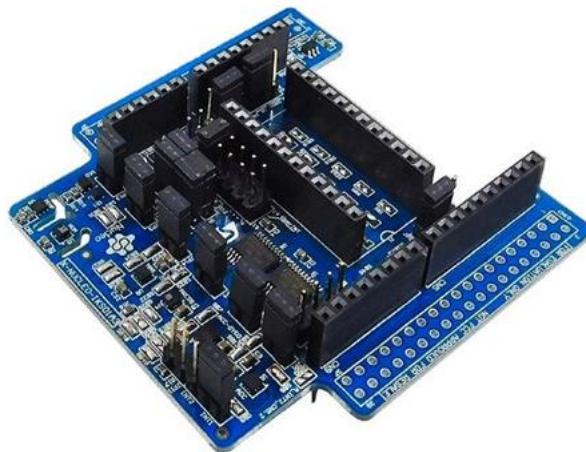


Figure 54: Carte X-NUCLEO-IKS01A3

Pour le développement de notre carte ESP nous utiliserons Espressif IDE le logiciel de développement, disponible sur Linux et Windows. Nous utiliserons aussi Thingsboard version installée, ce sera notre plateforme IoT. Nous aurons aussi besoin d'une base de données, donc on devra installer PostgreSQL.

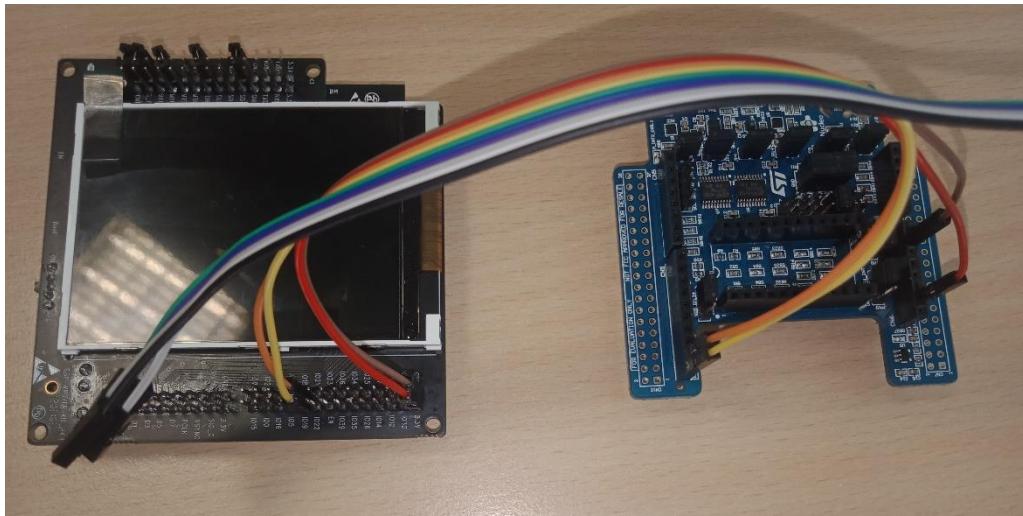


Figure 55: Branchement de la carte eESp32 WROVERKIT (à gauche) avec XNUCLEO(à droite)

Nous avons envoyé les données de notre carte Esp32 via un protocole HTTP en indiquant l'adresse locale et le port ainsi que le jeton d'accès.

-La partie logicielle :

```

static void http_rest_with_url(void)
{
    esp_err_t err;
    esp_http_client_config_t config = {
        .url = "http://192.168.221.148:8888/api/v1/bPEaeandnr9pAUoHtoPY/telemetry",
    };
    esp_http_client_handle_t client = esp_http_client_init(&config);
    // first request
    err = esp_http_client_perform(client);

    char post_data[500];// = "{\"temperature\": 1, \"humidity\": \"value1\"}";
    sprintf((char *)post_data, "{\"temperature\": %f }", temperature_degs);
    esp_http_client_set_url(client, "http://192.168.221.148:8888/api/v1/bPEaeandnr9pAUoHtoPY/telemetry");
    esp_http_client_set_method(client, HTTP_METHOD_POST);
    esp_http_client_set_header(client, "Content-Type", "application/json");
    esp_http_client_set_post_field(client, post_data, strlen(post_data));
    err = esp_http_client_perform(client);

    esp_http_client_cleanup(client);
}

```

Annotations sur le code :

- Adresse local du PC : 192.168.221.148
- Numéro de Port utilisé par Thingsboard : 8888
- Access token : bPEaeandnr9pAUoHtoPY
- Protocole HTTP : http://
- La clé : 192.168.221.148:8888/api/v1/bPEaeandnr9pAUoHtoPY/telemetry
- Donnée mesurée : temperature

Figure 56: Programme du protocole HTTP

-Résultat sur le terminal :

```
I (2591) esp_netif_handlers: sta ip: 192.168.221.49, mask: 255.255.255.0, gw: 192.168.221.248
E (2601) HTTP_CLIENT: Connection failed, sock < 0
W (3481) wifi:<ba->addr:idx:0 (ifx:0, be:8d:6a:d8:f0:72), tid:0, ssn:17, winSize:64
Temperature [degC]:26.00
Temperature [degC]:26.00
Temperature [degC]:26.00
```

2. Télémétrie :

Nous pouvons vérifier la réception des valeurs envoyées en accédant au champ de notre dispositif dans dernière télémétrie :

The screenshot shows a device detail view titled "carte1". The top navigation bar includes tabs for "Details", "Attributs", "Dernière télémétrie" (which is selected and highlighted in blue), "Alarmes", "Événements", and "Relations". A red circle highlights the edit icon in the top right corner. The main content area is titled "Dernière télémétrie" and displays a table with one row of data:

Dernière mise à jour	Clé ↑	Valeur
2022-05-25 16:10:27	temperature	25.75

Figure 57: réception de la valeur de température envoyée.

La température est bien téléversée en temps réel sur le dispositif en télémétrie.

3. Tableau de bord

a) Tableau de bord pour un capteur

Nous pouvons utiliser le dispositif pour pouvoir représenter la température en temps réelle, nous avons aussi accès à la moyenne et aussi à toutes les données enregistrées par ce capteur grâce à la base de données.

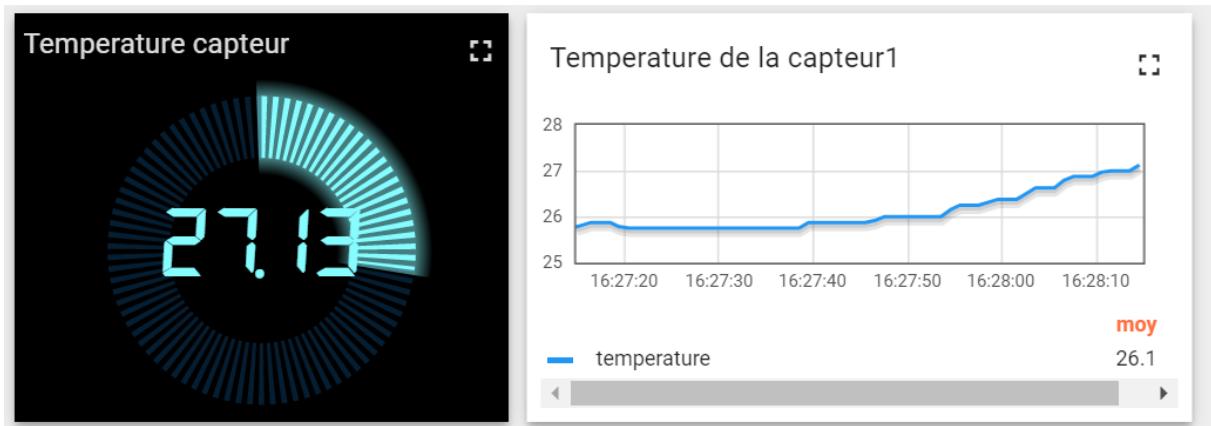


Figure 58: Tableau de bord de la température

b) Tableau de bord avec plusieurs capteurs

À présent, nous pouvons aussi utiliser d'autres capteurs sur la même carte.

Capteur utilisé : Humidité, pression et température

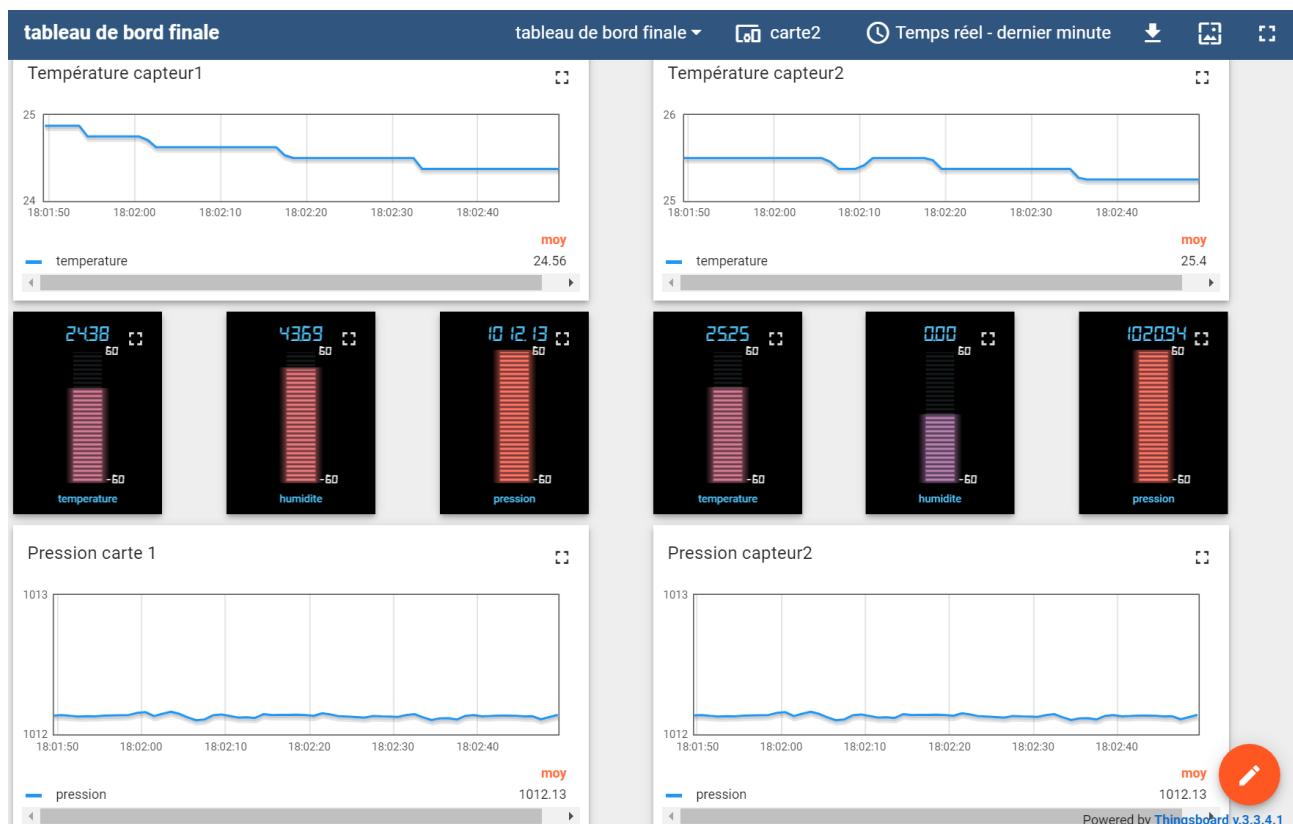


Figure 59: Tableau de bord avec deux cartes ESP et capteurs

g) Utilisation de plusieurs cartes :

Pour ajouter plusieurs cartes, il suffit de refaire la même configuration sur la deuxième carte, lui téléverser le programme et la mettre sous tension. La valeur sera mise à jour sur le tableau de bord.

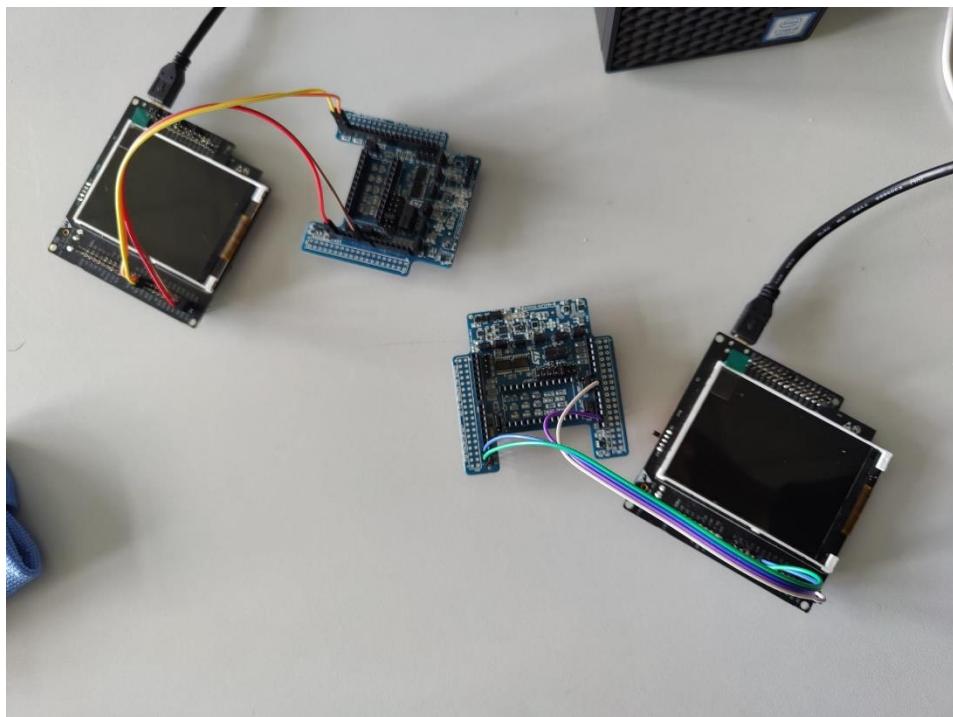


Figure 61: Branchement des deux cartes

Configuration de la clé de données

Clé *	temperature	X
Label *	temperature	Couleur * #2196f3
Symbole spécial à afficher à côté de la vale...		Nombre de chiffres après virgule flottante
<input type="checkbox"/> Utiliser la fonction de post-traitement des données		

Annuler Enregistrer

Figure 60: Configuration de la donnée à transmettre

La configuration de la source de la donnée à envoyer, il faut veiller à saisir la même valeur dans la clé et dans le code pour réussir à faire la transmission, si non ça ne marchera pas.

Pour rendre publique la donnée, il suffit de rendre public le tableau de bord :

The screenshot shows a user interface for managing dashboards. At the top, there's a search bar and a toolbar with icons for creating, deleting, and filtering. Below that, a list of boards is shown, with one board named "Tableau des 2 clients" selected. This board has a blue header with a question mark icon and a red circular button containing a white pencil icon. The main content area shows details about the board, including its creation date (2022-05-26 01:22:46), title, and access rights (attributed to clients, public). Below this, there are buttons for opening the board, exporting it, making it private, managing affected clients, and deleting it. At the bottom, a public link is provided: <http://localhost:8888/dashboard/95ce7430-dc81-11ec-90b3-75dee524e362?publicId=a4aabe60-e27f-11e>.

Figure 62: Le lien publique pour accéder aux données.

Code utilisé

En se basant sur l'exemple ESP HTTP Client de la documentation **espressif ESP-IDF** et sur les exemples du GitHub [STMicroelectronics](#), le code utilisé est le suivant :

La récupération des valeurs des capteurs sensors.h et sensors.c :

Sensors.h : initialisation de I2c(ports utilisés, fréquences...), déclaration des fonctions et variables.

```
#ifndef MAIN_SENSORS_H_
#define MAIN_SENSORS_H_

#include <stdio.h>
#include <string.h>
#include "esp_log.h"
#include "stts751_reg.h"
#include "driver/i2c.h"
#include "sdkconfig.h"
#include "lps22hh_reg.h"
#include "hts221_reg.h"

#define _I2C_NUMBER(num) I2C_NUM_##num
```

```

#define I2C_NUMBER(num) _I2C_NUMBER(num)

#define DELAY_TIME_BETWEEN_ITEMS_MS 1000 /*!< delay time between different test
items */

#define I2C_MASTER_SCL_IO 19           /*!< gpio number for I2C master clock
*/
#define I2C_MASTER_SDA_IO 18           /*!< gpio number for I2C master data */
#define I2C_MASTER_NUM I2C_NUM_0 /*!< I2C port number for master dev */
#define I2C_MASTER_FREQ_HZ 400000      /*!< I2C master clock frequency */
#define I2C_MASTER_TX_BUF_DISABLE 0    /*!< I2C master
doesn't need buffer */
#define I2C_MASTER_RX_BUF_DISABLE 0    /*!< I2C master
doesn't need buffer */

#define STTS751_ADDR_7BITS 0x4a
#define LPS22HH_ADDR_7BITS 0x5D
#define HTS221_ADDR_7BITS 0x5F

#define SENSOR_BUS I2C_MASTER_NUM
#define WRITE_BIT I2C_MASTER_WRITE      /*!< I2C master write */
#define READ_BIT I2C_MASTER_READ       /*!< I2C master read */
#define ACK_CHECK_EN 0x1               /*!< I2C master will check ack
from slave*/
#define ACK_CHECK_DIS 0x0              /*!< I2C master will not check ack
from slave */
#define ACK_VAL 0x0                  /*!< I2C ack value */
#define NACK_VAL 0x1                 /*!< I2C nack value */

typedef union{
    int16_t i16bit;
    uint8_t u8bit[2];
} axis1bit16_t;

stmdev_ctx_t dev_ctx;
stmdev_ctx_t dev_ctxp,dev_hum;
axis1bit16_t data_raw_temperature;
float temperature_degC;

int32_t i2c_master_read_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_rd,
uint16_t size);
int32_t i2c_master_write_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t
*data_wr, uint16_t size);
esp_err_t i2c_master_init(void);
void get_temp_task(void *args);
void init_stts751();

// privates variables de capteur LPS22HH
uint32_t data_raw_pressure;
float pressure_hPa;
uint8_t whoamI, rst;
int32_t i2c_LPS22HH_read_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_rd,
uint16_t size);
int32_t i2c_LPS22H_write_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_wr,
uint16_t size);
void lps22hh_init();
void get_press(void *args);

```

```

// privates variables de capteur hts221
int16_t data_raw_humidity;
float humidity_perc;
typedef struct {
    float x0;
    float y0;
    float x1;
    float y1;
} lin_t;
lin_t lin_hum;

uint8_t whoamI1, rst1;
int32_t i2c_hts221_read_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_rd,
uint16_t size);
int32_t i2c_hts221_write_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t
*data_wr, uint16_t size);

void hts221_init();
void get_hum(void *args);
#endif /* MAIN_SENSORS_H_ */

```

Sensors.c :

```

#include "sensors.h"

int32_t i2c_master_read_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_rd,
uint16_t size)
{
    if (size == 0) {
        return ESP_OK;
    }
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (STTS751_ADDR_7BITS << 1) | I2C_MASTER_WRITE,
ACK_CHECK_EN);
    i2c_master_write_byte(cmd, regaddr, ACK_CHECK_EN);

    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (STTS751_ADDR_7BITS << 1) | I2C_MASTER_READ,
ACK_CHECK_EN);
    if (size > 1) {
        i2c_master_read(cmd, data_rd, size - 1, ACK_VAL);
    }
    i2c_master_read_byte(cmd, data_rd + size - 1, NACK_VAL);
    i2c_master_stop(cmd);
    esp_err_t ret = i2c_master_cmd_begin(i2c_num, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
    return ret;
}

/**
 * @brief Test code to write esp-i2c-slave
 *        Master device write data to slave memory like device,
 *        the data will be stored in slave buffer.
 *        We can read them out from slave buffer.

```

```

*
*



---


* | start | slave_addr + wr_bit + ack | write 1 byte (register address) + ack |
write n bytes + ack | stop |
* -----|-----|-----|-----|-----|-----|-----|-----|
-----|-----|-----|-----|-----|-----|-----|-----|
*
*/
int32_t i2c_master_write_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t
*data_wr, uint16_t size)
{
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (STTS751_ADDR_7BITS << 1) | WRITE_BIT,
ACK_CHECK_EN);
    i2c_master_write_byte(cmd, regaddr, ACK_CHECK_EN);
    i2c_master_write(cmd, data_wr, size, ACK_CHECK_EN);
    i2c_master_stop(cmd);
    esp_err_t ret = i2c_master_cmd_begin(i2c_num, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
    return ret;
}

/**
 * @brief i2c master initialization
 */
esp_err_t i2c_master_init(void)
{
    int i2c_master_port = I2C_MASTER_NUM;
    i2c_config_t conf={0};
    conf.mode = I2C_MODE_MASTER;
    conf.sda_io_num = I2C_MASTER_SDA_IO;
    conf.sda_pullup_en = GPIO_PULLUP_DISABLE; // Pullup resistors are already
present on X-NUCLEO-IKS01A3
    conf.scl_io_num = I2C_MASTER_SCL_IO;
    conf.scl_pullup_en = GPIO_PULLUP_DISABLE;
    conf.master.clk_speed = I2C_MASTER_FREQ_HZ;
    i2c_param_config(i2c_master_port, &conf);
    return i2c_driver_install(i2c_master_port, conf.mode,
I2C_MASTER_RX_BUF_DISABLE, I2C_MASTER_TX_BUF_DISABLE, 0);
}
void get_temp_task(void *args)
{
    /* Read output only if not busy */
    uint8_t flag;

    while (1) {
        stts751_flag_busy_get(&dev_ctx, &flag);
        if (flag)
        {
            /* Read temperature data */
            memset(data_raw_temperature.u8bit, 0, sizeof(int16_t));
            stts751_temperature_raw_get(&dev_ctx, &data_raw_temperature.i16bit);
            temperature_degC =
stts751_from_lsb_to_celsius(data_raw_temperature.i16bit);

            printf("Temperature [degC]:%3.2f\r\n", temperature_degC);
        }
    }
}

```

```

        vTaskDelay(pdMS_TO_TICKS( 1000 ));
    }
}

void init_stts751()
{
    /* This acts as the entry point of ST's STTS751 driver */
    dev_ctx.write_reg = i2c_master_write_slave;
    dev_ctx.read_reg = i2c_master_read_slave;
    dev_ctx.i2c_number = SENSOR_BUS;

    /* Enable interrupt on high(=49.5 degC)/low(=-4.5 degC) temperature. */
    float temperature_high_limit = 49.5f;
    stts751_high_temperature_threshold_set(&dev_ctx,
stts751_from_celsius_to_lsb(temperature_high_limit));

    float temperature_low_limit = -4.5f;
    stts751_low_temperature_threshold_set(&dev_ctx,
stts751_from_celsius_to_lsb(temperature_low_limit));

    stts751_pin_event_route_set(&dev_ctx, PROPERTY_ENABLE);

    /* Set Output Data Rate */
    stts751_temp_data_rate_set(&dev_ctx, STTS751_TEMP_ODR_250mHz);

    /* Set Resolution */
    stts751_resolution_set(&dev_ctx, STTS751_11bit);
}

```

```

/// FOn
int32_t i2c_LPS22HH_read_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_rd,
uint16_t size)
{
    if (size == 0) {
        return ESP_OK;
    }
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (LPS22HH_ADDR_7BITS << 1) | I2C_MASTER_WRITE,
ACK_CHECK_EN);
    i2c_master_write_byte(cmd, regaddr, ACK_CHECK_EN);

    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (LPS22HH_ADDR_7BITS << 1) | I2C_MASTER_READ,
ACK_CHECK_EN);
    if (size > 1) {
        i2c_master_read(cmd, data_rd, size - 1, ACK_VAL);
    }
    i2c_master_read_byte(cmd, data_rd + size - 1, NACK_VAL);
    i2c_master_stop(cmd);
    esp_err_t ret = i2c_master_cmd_begin(i2c_num, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
    return ret;
}

```

```

}

/** 
* @brief Test code to write esp-i2c-slave
*        Master device write data to slave memory like device,
*        the data will be stored in slave buffer.
*        We can read them out from slave buffer.
*
* | start | slave_addr + wr_bit + ack | write 1 byte (register address) + ack |
* write n bytes + ack | stop |
* -----|-----|-----|-----|-----|-----|
*-----|-----|-----|-----|
*
*/
int32_t i2c_LPS22HH_write_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t
*data_wr, uint16_t size)
{
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (LPS22HH_ADDR_7BITS << 1) | WRITE_BIT,
ACK_CHECK_EN);
    i2c_master_write_byte(cmd, regaddr, ACK_CHECK_EN);
    i2c_master_write(cmd, data_wr, size, ACK_CHECK_EN);
    i2c_master_stop(cmd);
    esp_err_t ret = i2c_master_cmd_begin(i2c_num, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
    return ret;
}

//// fonctions de LPS22HH///
void lps22hh_init()
{
    /* Initialize mems driver interface */
    dev_ctxp.write_reg = i2c_LPS22HH_write_slave ;
    dev_ctxp.read_reg = i2c_LPS22HH_read_slave;
    dev_ctxp.i2c_number= SENSOR_BUS;
    /* Initialize platform specific hardware */

    /* Check device ID */
    whoamI = 0;
    lps22hh_device_id_get(&dev_ctxp, &whoamI);

    /* Restore default configuration */
    lps22hh_reset_set(&dev_ctxp, PROPERTY_ENABLE);

    do {
        lps22hh_reset_get(&dev_ctxp, &rst);
    } while (rst);

    /* Enable Block Data Update */
}

```

```

        lps22hh_block_data_update_set(&dev_ctxp, PROPERTY_ENABLE);
        /* Set Output Data Rate */
        lps22hh_data_rate_set(&dev_ctxp, LPS22HH_10_Hz_LOW_NOISE);

    }

void get_press(void *args)
{
    lps22hh_reg_t reg;
    while (1) {
        /* Read output only if new value is available */
        lps22hh_read_reg(&dev_ctxp, LPS22HH_STATUS, (uint8_t *)&reg, 1);

        if (reg.status.p_da) {
            memset(&data_raw_pressure, 0x00, sizeof(uint32_t));
            lps22hh_pressure_raw_get(&dev_ctxp, &data_raw_pressure);
            pressure_hPa = lps22hh_from_lsb_to_hpa( data_raw_pressure );
            printf( "pressure [hPa]:%6.2f\r\n", pressure_hPa );
            vTaskDelay(pdMS_TO_TICKS( 1000 ));
        }
    }
}

///////////////////////////////////////////////////////////////////
int32_t i2c_hts221_read_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_rd,
uint16_t size)
{
    if (size == 0) {
        return ESP_OK;
    }
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (HTS221_ADDR_7BITS << 1) | I2C_MASTER_WRITE,
ACK_CHECK_EN);
    i2c_master_write_byte(cmd, regaddr, ACK_CHECK_EN);

    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (HTS221_ADDR_7BITS << 1) | I2C_MASTER_READ,
ACK_CHECK_EN);
    if (size > 1) {
        i2c_master_read(cmd, data_rd, size - 1, ACK_VAL);
    }
    i2c_master_read_byte(cmd, data_rd + size - 1, NACK_VAL);
    i2c_master_stop(cmd);
    esp_err_t ret = i2c_master_cmd_begin(i2c_num, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
    return ret;
}

< /**
 * @brief Test code to write esp-i2c-slave
 *        Master device write data to slave memory like device,
 *        the data will be stored in slave buffer.
 *        We can read them out from slave buffer.
 *
 */

```

```

*
-----|-----|-----|-----|-----|-----|
* | start | slave_addr + wr_bit + ack | write 1 byte (register address) + ack | -----
write n bytes + ack | stop |
-----|-----|-----|-----|-----|-----|
*
*/
int32_t i2c_hts221_write_slave(uint8_t i2c_num, uint8_t regaddr, uint8_t *data_wr,
uint16_t size)
{
    i2c_cmd_handle_t cmd = i2c_cmd_link_create();
    i2c_master_start(cmd);
    i2c_master_write_byte(cmd, (HTS221_ADDR_7BITS << 1) | WRITE_BIT, ACK_CHECK_EN);
    i2c_master_write_byte(cmd, regaddr, ACK_CHECK_EN);
    i2c_master_write(cmd, data_wr, size, ACK_CHECK_EN);
    i2c_master_stop(cmd);
    esp_err_t ret = i2c_master_cmd_begin(i2c_num, cmd, 1000 / portTICK_RATE_MS);
    i2c_cmd_link_delete(cmd);
    return ret;
}

float linear_interpolation(lin_t *lin, int16_t x)
{
    return ((lin->y1 - lin->y0) * x + ((lin->x1 * lin->y0) -
                                              (lin->x0 * lin->y1)))
           / (lin->x1 - lin->x0);
}

void hts221_init()
{
    dev_hum.write_reg = i2c_hts221_write_slave;
    dev_hum.read_reg = i2c_hts221_read_slave;
    dev_hum.i2c_number = SENSOR_BUS;
    /* Check device ID */
    whoamI1 = 0;
    hts221_device_id_get(&dev_hum, &whoamI1);

    /* Read humidity calibration coefficient */

    hts221_hum_adc_point_0_get(&dev_hum, &lin_hum.x0);
    hts221_hum_rh_point_0_get(&dev_hum, &lin_hum.y0);
    hts221_hum_adc_point_1_get(&dev_hum, &lin_hum.x1);
    hts221_hum_rh_point_1_get(&dev_hum, &lin_hum.y1);

    /* Enable Block Data Update */
    hts221_block_data_update_set(&dev_hum, PROPERTY_ENABLE);
    /* Set Output Data Rate */
    hts221_data_rate_set(&dev_hum, HTS221_ODR_1Hz);
    /* Device power on */
    hts221_power_on_set(&dev_hum, PROPERTY_ENABLE);
}

void get_hum(void *args)
{
    while (1) {
        /* Read output only if new value is available */
}

```

```

hts221_reg_t reg;
hts221_status_get(&dev_hum, &reg.status_reg);

if (reg.status_reg.h_da) {
    /* Read humidity data */
    memset(&data_raw_humidity, 0x00, sizeof(int16_t));
    hts221_humidity_raw_get(&dev_hum, &data_raw_humidity);
    humidity_perc = linear_interpolation(&lin_hum, data_raw_humidity);

    if (humidity_perc < 0) {
        humidity_perc = 0;
    }

    if (humidity_perc > 100) {
        humidity_perc = 100;
    }

    printf( "Humidity [%]:%3.2f\r\n", humidity_perc);
    vTaskDelay(pdMS_TO_TICKS( 1000 ));
}

}

```

Dans main, initialisation et connexion wifi, connexion à la serveur et l'envoi les valeurs : température, humidité et pression.

```

#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "nvs_flash.h"

#include "lwip/err.h"
#include "lwip/sockets.h"
#include "lwip/sys.h"
#include "lwip/netdb.h"
#include "lwip/dns.h"

#include "lwip/opt.h"
#include "lwip/arch.h"
#include "lwip/api.h"

#include "esp_http_client.h"

#include "sensors.h"

#define EXAMPLE_WIFI_SSID "aaa"/"A"
#define EXAMPLE_WIFI_PASS "alaa1111"/"joelle123"
const int CONNECTED_BIT = BIT0;

//float temperature_degC;

```

```

/* FreeRTOS event group to signal when we are connected & ready to make a request */
*/
static EventGroupHandle_t wifi_event_group;
static const char *TAG = "example";

static esp_err_t event_handler(void *ctx, system_event_t *event)
{
    switch(event->event_id) {
    case SYSTEM_EVENT_STA_START:
        esp_wifi_connect();
        break;
    case SYSTEM_EVENT_STA_GOT_IP:
        xEventGroupSetBits(wifi_event_group, CONNECTED_BIT);
        break;
    case SYSTEM_EVENT_STA_DISCONNECTED:
        /* This is a workaround as ESP32 WiFi libs don't currently
         * auto-reassociate. */
        esp_wifi_connect();
        xEventGroupClearBits(wifi_event_group, CONNECTED_BIT);
        break;
    default:
        break;
    }
    return ESP_OK;
}

static void initialise_wifi(void)
{
    tcpip_adapter_init();
    wifi_event_group = xEventGroupCreate();
    ESP_ERROR_CHECK( esp_event_loop_init(event_handler, NULL) );
    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK( esp_wifi_init(&cfg) );
    ESP_ERROR_CHECK( esp_wifi_set_storage(WIFI_STORAGE_RAM) );
    wifi_config_t wifi_config = {
        .sta = {
            .ssid = EXAMPLE_WIFI_SSID,
            .password = EXAMPLE_WIFI_PASS,
        },
    };
    ESP_LOGI(TAG, "Setting WiFi configuration SSID %s...", wifi_config.sta.ssid);
    ESP_ERROR_CHECK( esp_wifi_set_mode(WIFI_MODE_STA) );
    ESP_ERROR_CHECK( esp_wifi_set_config(WIFI_IF_STA, &wifi_config) );
    ESP_ERROR_CHECK( esp_wifi_start() );
}

esp_err_t _http_event_handle(esp_http_client_event_t *evt)
{
    switch(evt->event_id) {
    case HTTP_EVENT_ERROR:
        ESP_LOGI(TAG, "HTTP_EVENT_ERROR");
        break;
    case HTTP_EVENT_ON_CONNECTED:
        ESP_LOGI(TAG, "HTTP_EVENT_ON_CONNECTED");
        break;
    case HTTP_EVENT_HEADER_SENT:
        ESP_LOGI(TAG, "HTTP_EVENT_HEADER_SENT");
        break;
    
```

```

        case HTTP_EVENT_ON_HEADER:
            ESP_LOGI(TAG, "HTTP_EVENT_ON_HEADER");
            printf("%.*s", evt->data_len, (char*)evt->data);
            break;
        case HTTP_EVENT_ON_DATA:
            ESP_LOGI(TAG, "HTTP_EVENT_ON_DATA, len=%d", evt->data_len);
            if (!esp_http_client_is_chunked_response(evt->client)) {
                printf("%.*s", evt->data_len, (char*)evt->data);
            }
            break;
        case HTTP_EVENT_ON_FINISH:
            ESP_LOGI(TAG, "HTTP_EVENT_ON_FINISH");
            break;
        case HTTP_EVENT_DISCONNECTED:
            ESP_LOGI(TAG, "HTTP_EVENT_DISCONNECTED");
            break;
    }
    return ESP_OK;
}

static void http_rest_with_url()
{
    esp_err_t err;
    esp_http_client_config_t config = {
        .url =
    "http://192.168.137.1:8888/api/v1/KfuDRV45yHwOMNymd6Ep/telemetry",
    };
    esp_http_client_handle_t client = esp_http_client_init(&config);
    // first request
    err = esp_http_client_perform(client);

    char post_data[500];

    sprintf((char *)post_data, "{\"temperature\": %.3f, \"pression\": %4.2f, \"humidite\": %.2f}", temperature_degC, pressure_hPa, humidity_perc);

    esp_http_client_set_url(client, "http://192.168.137.1:8888/api/v1/KfuDRV45yHwOMNymd6Ep/telemetry");
        esp_http_client_set_method(client, HTTP_METHOD_POST);
        esp_http_client_set_header(client, "Content-Type", "application/json");
        esp_http_client_set_post_field(client, post_data, strlen(post_data));
    err = esp_http_client_perform(client);

    esp_http_client_cleanup(client);

}
void app_main(void)
{
    nvs_flash_init();
    initialise_wifi();

    ESP_ERROR_CHECK(i2c_master_init());
    init_stts751();
}

```

```

lps22hh_init();
hts221_init();

xTaskCreate(get_temp_task, "task", 10000, NULL, 1, NULL);
xTaskCreate(get_press, "task2", 10000, NULL, 1, NULL);
xTaskCreate(get_hum, "task3", 10000, NULL, 1, NULL);
xTaskCreate(http_rest_with_url, "task4", 10000, NULL, 1, NULL);
while(1)
{
    http_rest_with_url();
}
}

```

La fonction `http_rest_with_url()` pour connexion et l'envoi des valeurs en utilisant la méthode http post.

```

sprintf((char *)post_data, "{\"temperature\": %.3f,\"pression\":
%4.2f,\"humidite\": %.2f}",temperature_degC,pressure_hPa,humidity_perc);

```

`temperature,pressure,humidite` : les clés, les variables sur la plateforme thingsboard.
`temperature_degC,pressure_hPa,humidity_perc` : les variables qui contiennent les valeurs récupérées des capteurs.

VI. Sécurité

Avec wireshark toute personne connectée sur le même réseau peut récupérer le jeton et connaître les données du paquet. Pour éviter ces attaques il faut ajouté un protocole de sécurité comme SSL.

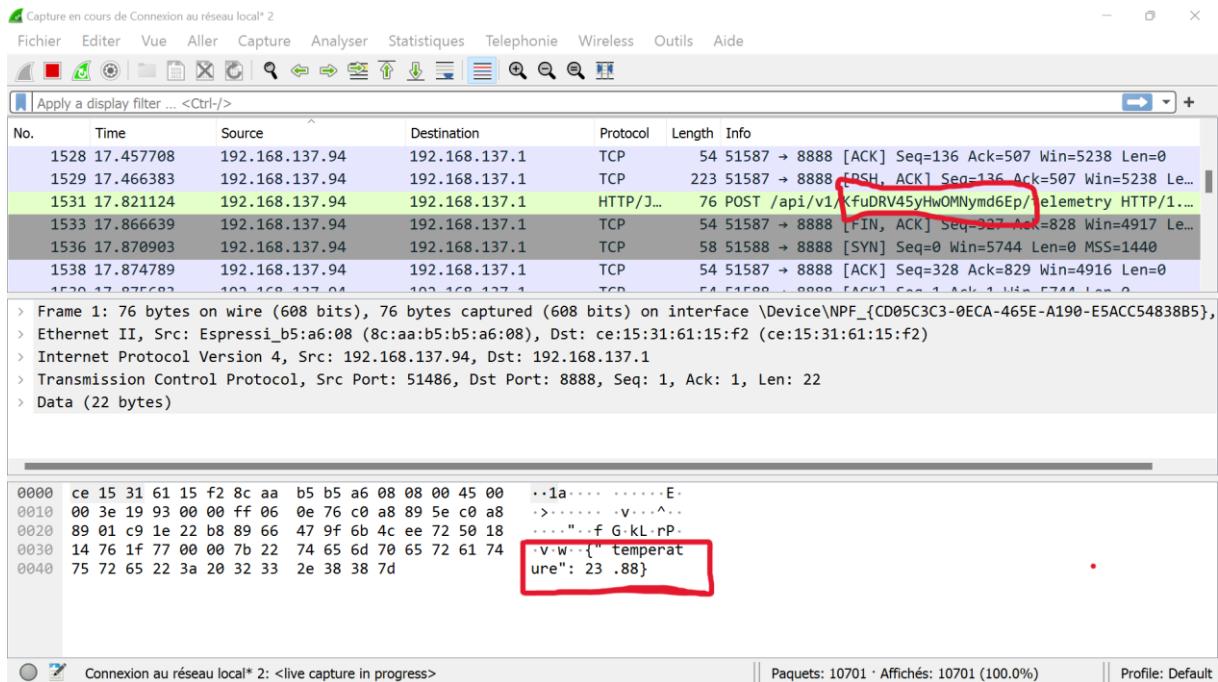


Figure 63: récupération le jeton et les données avec wireshark

ThingsBoard offre la possibilité d'exécuter un serveur HTTP qui héberge l'interface utilisateur Web et sert des appels d'API REST via SSL.

La plupart des environnements ThingsBoard utilisent l'équilibrEUR de charge comme point de terminaison pour la connexion SSL entre le client et la plate-forme. En d'autres termes, le trafic Internet est chiffré entre le navigateur de l'utilisateur et l'équilibrEUR de charge, mais est déchiffré entre l'équilibrEUR de charge et les services de la plate-forme. Pour utiliser https nous avons ajouté le protocole SSL, pour l'activer nous avons édité le fichier de configuration de ThingsBoard (thingsboard.yml).

```

export SSL_ENABLED=true
export SSL_CREDENTIALS_TYPE=PEM
export SSL_PEM_CERT=server.pem
export SSL_PEM_KEY=server_key.pem
export SSL_PEM_KEY_PASSWORD=secret

```

Il faut générer un certificat PEM (server.pem) et une clé privée (server_key.pem) auto-signés, en utilisant OpenSSL, qui est une boîte à outils de chiffrement comportant deux bibliothèques

libcrypto et libssl, fournissant respectivement une implémentation des algorithmes cryptographiques et du protocole de communication SSL/TLS, ainsi qu'une interface en ligne de commande (noté bien qu'il faut installer openSSL sur votre machine linux ou windows). Il faut que le certificat et la clé doit être dans le même répertoire, sinon il faut préciser le chemin dans le fichier de configuration thingsboard.yml.

Génération de certificats auto-signés :

Suivez les instructions ci-dessous pour générer vos propres fichiers de certificat. Utile pour les tests, mais chronophage et non recommandé pour la production.

Note Cette étape nécessite un système d'exploitation basé sur Linux avec openssl installé.

Pour générer un certificat PEM et une clé privée auto-signés par le serveur, utilisez la commande suivante :

```
1 openssl ecparam -out server_key.pem -name secp256r1 -genkey  
2 openssl req -new -key server_key.pem -x509 -nodes -days 365 -out ser
```

Puis redémarrez thingsboard sous windows en utilisant PowerShell ou cmd :

net stop thingsboard

net start thingsboard

Sous Linux :

```
sudo service thingsboard stop  
sudo service thingsboard start
```

⚠️ Attention : risque probable de sécurité

Firefox a détecté une menace de sécurité potentielle et n'a pas poursuivi vers 192.168.197.122. Si vous accédez à ce site, des attaquants pourraient dérober des informations comme vos mots de passe, courriels, ou données de carte bancaire.

[En savoir plus...](#)

[Retour \(recommandé\)](#)

[Avancé...](#)

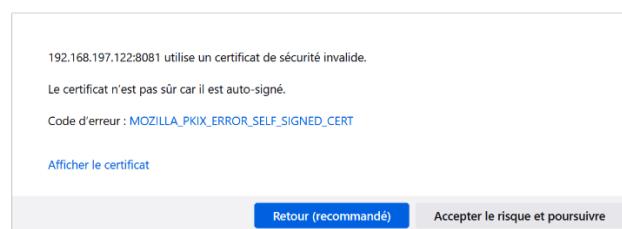


Figure 64 : risque de certificat auto-signé

Notre certificat utilisé n'est pas connu sur les navigateurs, car il est privé et pas public, alors lors de l'utilisation un message d'avertissement (risque probable de sécurité) apparaît, nous avons accepté le risque pour accéder à notre plateforme.

The screenshot shows the ThingsBoard web interface. The left sidebar has items like 'Chaines de règles', 'Clients', 'Actifs', 'Dispositifs', etc. The main area shows a 'Tableaux de bord' section with a list of dashboards. One dashboard is selected, titled 'Tableau_dispositif_demo3_carte1'. The URL in the browser bar is highlighted with a red box: 'https://172.20.10.11:8081/dashboards'. In the dashboard details panel, the URL 'https://172.20.10.11:8081/dashboard/2ca03560-d511-11ec-9f69-332bfe4e037d?pu=' is also highlighted with a red box.

Figure 65 : ThingsBoard en https

VII. Annexes

1) Installation de l'environnement Espressif IDE

a) Installation de JAVA

Voici le lien d'installation de java à faire

https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.exe

Suivre ces différentes étapes :



Figure 66: Java Download

Téléchargez-le par défaut.

b) Installation d'ESP IDF v4.4.1

Lien d'installation : https://dl.espressif.com/github_assets/espressif/esp-idf/releases/download/v4.4.1/esp-idf-v4.4.1.zip

décompressez le dossier et mettez-le sur le bureau.

c) Installation de Espressif IDE

Allez sur cette page :

<https://dl.espressif.com/dl/esp-idf/>

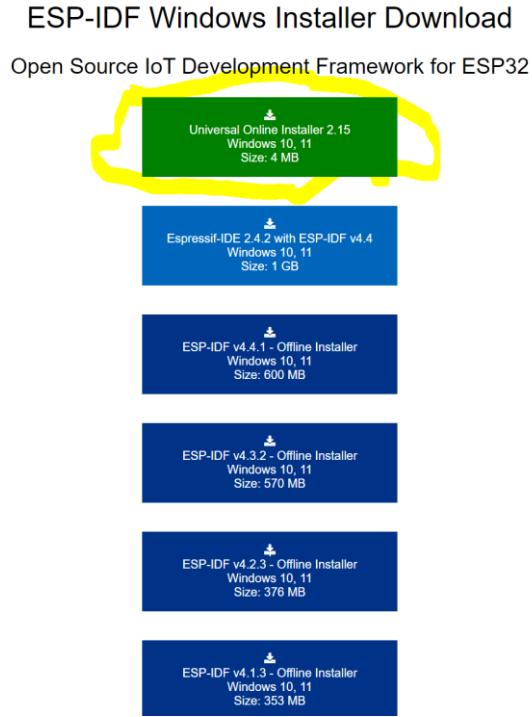


Figure 67: Installer ESP32

Exécuter puis installer Espressif.

d) Paramétrage de Espressif IDE

Help->Install new software

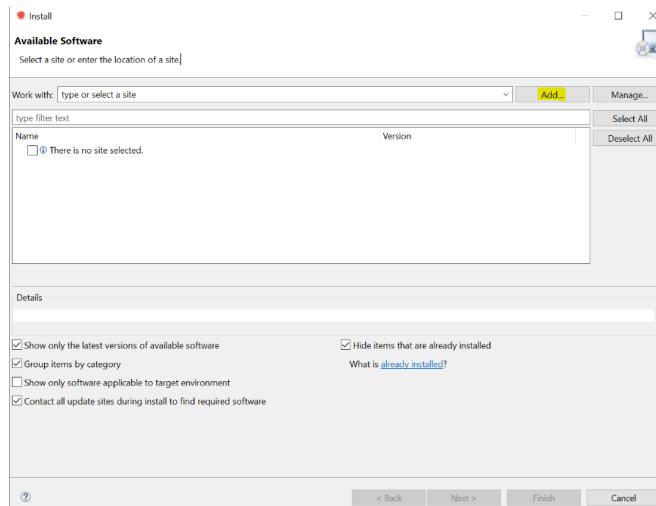


Figure 68: installer un software

Name: Espressif IDF Plugin for Eclipse

Location : <https://dl.espressif.com/dl/idf-eclipse-plugin/updates/latest/>

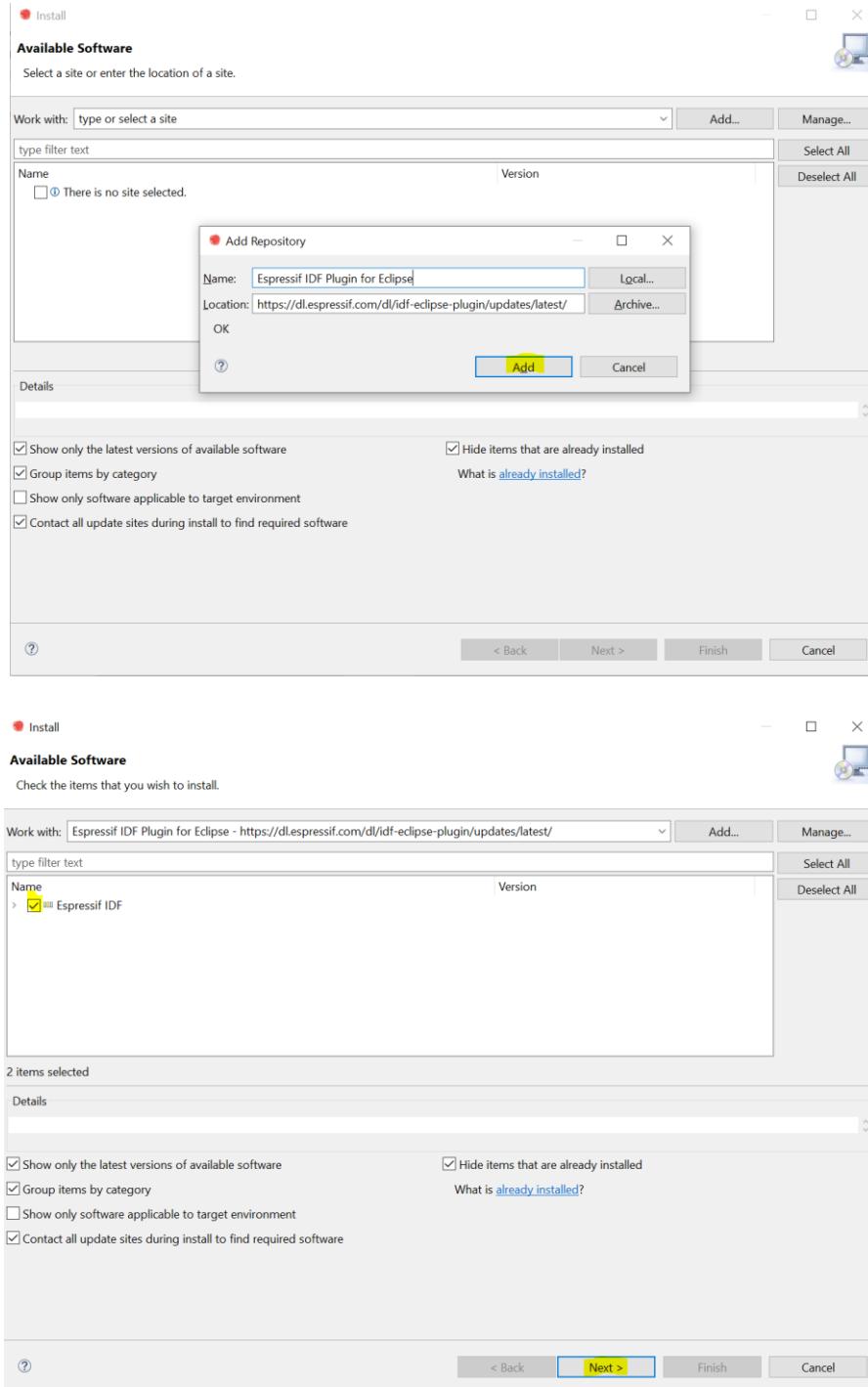


Figure 69: Création d'un nouveau software

Terminer l'installation et Espressif redémarrera.

À présent, allez dans Espressif->ESP-IDF Tools Manager->Install Tools

créez un dossier frameworks dans C:\Espressif puis coller

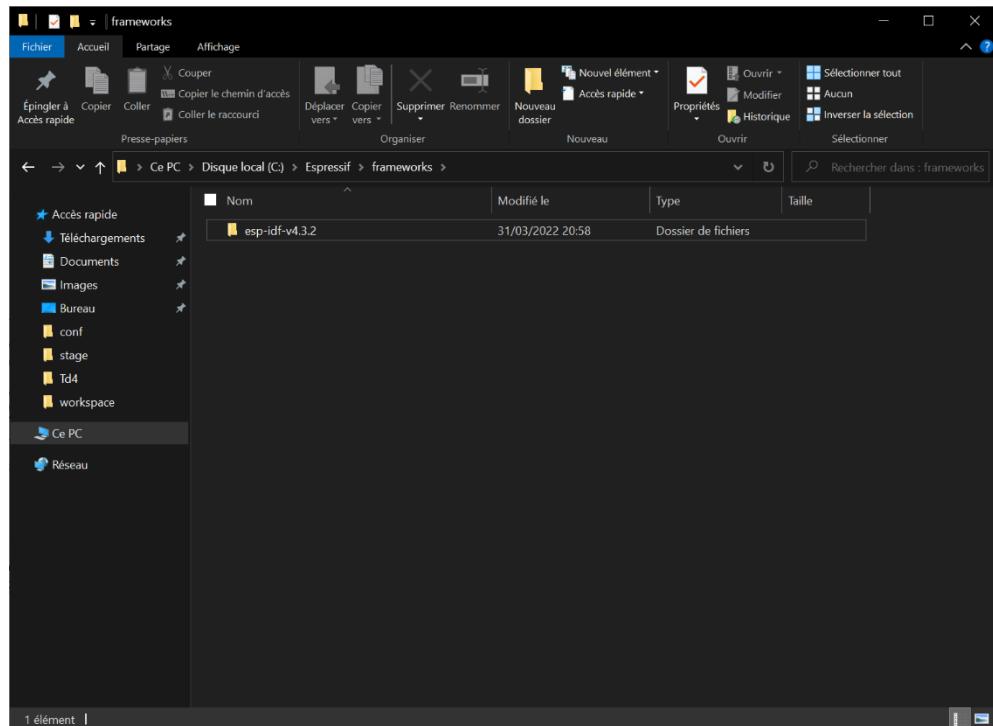


Figure 70: Dossier de esp-idfv4.3.2

Coller la version d'ESP IDF « esp-idf-v4.4.1 » dans C:\Espressif\frameworks.

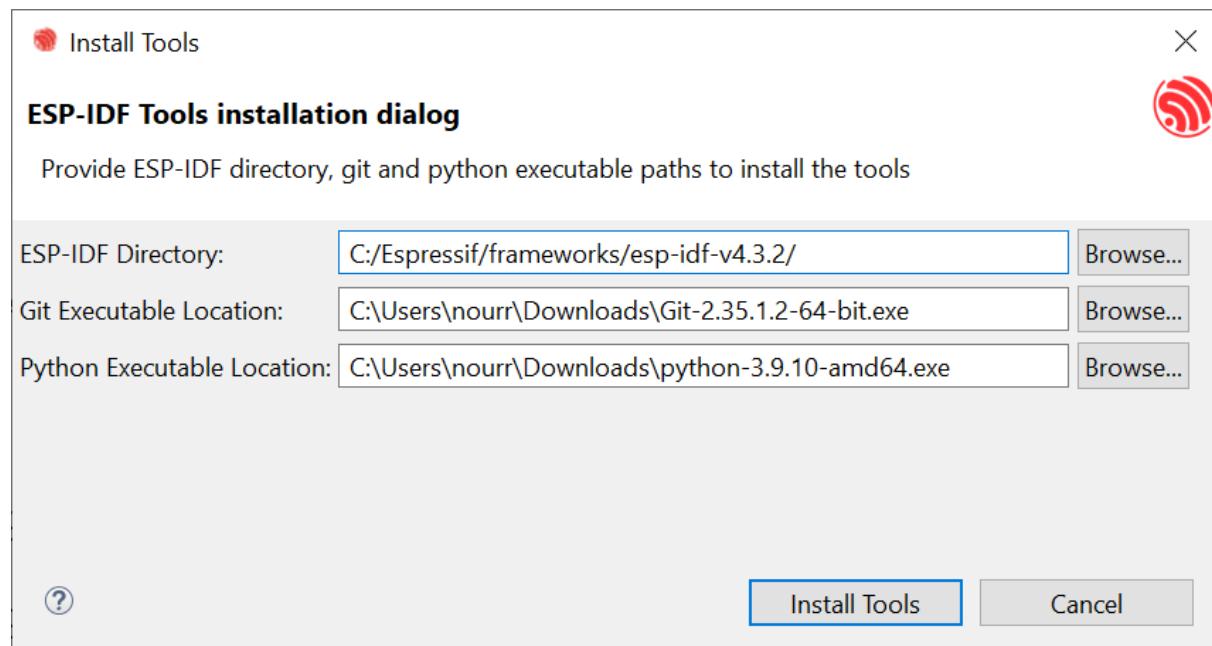


Figure 71 : ESP-IDF Tools

Choisir l'emplacement de ESP-IDF Directory : C:\Espressif\frameworks\esp-idf-v4.3.2

Télécharger exécutable de Git et Python pour l'emplacement comme indiqué sur l'image ci-dessus.

Git :<https://github.com/git-for-windows/git/releases/download/v2.36.1.windows.1/Git-2.36.1-64-bit.exe>

Python :<https://www.python.org/ftp/python/3.9.13/python-3.9.13-amd64.exe>

Appuyez sur Install Tools puis attendez la fin de l'installation.

À présent Espressif est fonctionnel.

VIII. Références

Lien d'installation de Thingsboard sous Windows :

<https://thingsboard.io/docs/user-guide/install/windows/?ubuntuThingsboardDatabase=postgresql> (Lien du guide d'installation).

Instalation de OpenSSL: [HTTP over SSL | ThingsBoard Community Edition](#)

Lien pour une documentation sur l'alias : <https://thingsboard.io/docs/user-guide/ui/aliases/>

Démo1 : https://github.com/ANOJM1/serveur_esp32-temperature.git

Démo2 : https://github.com/ANOJM1/esp32_serveur-ThingsBoard_capteurs.git