



Master 1 Informatique CERI

Rapport
Remote Automatic Watering System

Responsable
M. BENSLIMANE

Réalisé par:
NDIAYE Elhadj Pathé
KANTE Alpha Omar
THIEN Bao Tran

Année Universitaire
2018 - 2019

Remerciement

Nous tenons à remercier sincèrement Monsieur Benslimane Abderrahim, qui en tant que tuteur s'est toujours montré à l'écoute tout au long de la réalisation de ce travail ainsi pour sa générosité, ses conseils fructueux qu'il nous a prodigués le long de notre projet et la grande patience dont il a su faire preuve, malgré ses charges professionnelles.

Nous remercions également Monsieur Philippe Gozlan, Monsieur Marc Silanus et Monsieur Sahand Khodaparast, qui nous ont accompagné tout au long de notre projet et nous a guidé dans sa réalisation.

PLAN

PLAN	3
Liste des figures	5
1. Introduction	7
2. Présentation du projet	8
2.1 Problématique	8
2.2. Objectifs	8
2.3. Solution proposés	8
3. Mise en place de l'architecture de déploiement	9
3.1. Architecture Logique	9
3.2. Architecture Physique	11
3.3. Conception et réalisation du montage	12
4. Implémentation de l'architecture proposée	13
4.1. Programme arduino	13
4.2 Application Cloud TTN	17
4.2.1 Création du compte TTN	18
4.2.2 Création de l'application	18
4.2.3 Création de l'équipement	20
4.2.4 Formatage des données	21
4.3. Application Node-Red	22
4.3.1 Ajouter des palettes	23
4.3.2 Le protocole MQTT et configuration du node	24
4.3.3 Influxdb et configuration du node	31
4.5 Application Grafana	33
4.5.1 Ajouter des données sources	34
4.5.2 Conception des Graphes	35
4.6. Test de fonctionnement	37
4.7. Difficultés rencontrées	38
5. CONCLUSION	40
ANNEXES	41
Installation de node-red	41
Installation and configuration	41

Liste des figures

Figure 1:Architecture logique du réseau LORA	10
Figure 2: Architecture physique proposée	11
Figure 3 : Schéma du montage	12
Figure 4: Montage réalisé	13
Figure 5 : La fonction SETUP.....	14
Figure 6 : Code de récupération des données des capteurs	15
Figure 7: Code d'envoi de données vers TTN	15
Figure 8: code de traitements des données de Downlink	16
Figure 9 : Programme de pilotage de l'électrovanne	17
Figure 10 : Création compte TTN.....	18
Figure 11: Page d'accueil de l'application sur TTN.....	19
Figure 12: Création de l'application	19
Figure 13 : Rubrique device	20
Figure 14 : Création du matériel	20
Figure 15 : Data Application	21
Figure 16 : Fonction de décodage.....	21
Figure 17 : Résultat du décodage	22
Figure 18 : Flow uplink de TTN vers la base de données.....	23
Figure 19: Flow downlink de l'interface utilisateur et la carte Arduino MKR	23
Figure 20 : Installation des nodes.....	24
Figure 21 : Ajout d'un node	24
Figure 22 : Node mqtt connexion	26
Figure 23 : TTN Application ID	26
Figure 24 : TTN default key devices messages.....	26
Figure 25 : Configuration de la sécurité.....	27
Figure 26 : Node MQTT configuration serveur	27
Figure 27 : Node MQTT configuration en uplink.....	28
Figure 28 : Node MQTT configuration en downlink	29
Figure 29 : Configuration Node button	30
Figure 30 : Résultat configuration bouton.....	30
Figure 31: Fonction formatage des données à enregistrer	33
Figure 32:Configuration data source	34
Figure 33 : Configuration data source	34
Figure 34 : Configuration InfluxDB	35
Figure 35 : Configuration de la graphe de température	36
Figure 36 : Configuration de la graphe d'humidité	36
Figure 37: Configuration température actuelle.....	37
Figure 38: Configuration humidité actuelle	37
Figure 39 : Visualisation de la température et de l'humidité en temps réel sur Grafana.....	38
Figure 40: Démarrez Grafana	42

1. Introduction

L'internet des objets (IoT) est une technologie qui permet d'interconnecter des objets électronique à travers d'internet. Cette technologie permet l'extension d'internet aux objets physiques. Ces objets sont en général des systèmes électroniques embarqués équipés de capteur pour faire des mesures telles que la température, l'humidité, position, le débit, la distance et autres.

Ces objets connectés permettent de récolter une masse de données qui favorise une meilleure compréhension de notre environnement mais peuvent être configuré pour contrôler notre environnement et faciliter notre vie quotidienne.

L'internet des objets est utilisé dans divers domaines tel que la santé, la domotique, la robotique alors pourquoi ne pas l'utiliser dans le domaine de l'agriculture ?

L'agriculture étant le secteur utilisant la majeure partie des ressources en eau de la planète, alors un contrôle de cette utilisation de l'eau pourrait permettre une réduction considérable de la consommation en eau.

L'application de l'Internet des Objets, permettra ainsi de recueillir les données nécessaires à la compréhension de l'environnement des plantes et de les approvisionner en eau en fonctions de leurs besoins.

Dans le cadre de ce projet de Master 1 nous allons mettre en place un **Système automatique d'irrigation à distance en utilisant arduino et LoraWan (Remote Automatic Watering System)**.

Ce projet vise l'application de l'internet des objets au domaine de l'agriculture pour faire une irrigation contrôlé et distante d'une espace de culture.

Durant le premier semestre, nous avons établi un cahier de charge et proposé une architecture. Dans le deuxième semestre, nous sommes passés à l'implémentation du projet.

2. Présentation du projet

2.1 Problématique

Nous devons mettre en place un réseau LoRaWan qui permet à nos différents objets de communiquer avec le cloud TTN, de collecter des informations sur les capteurs, d'effectuer des actions automatiques pour arroser un espace de culture.

Cependant, pour une étude de l'art l'architecture proposée nous allons vous présenter les différentes technologies utilisées pour la collecte et la communication entre les équipements de bout en bout.

2.2. Objectifs

Nous allons mentionner 3 parties via l'architecture de LoRa:

- Réseau de collecte des données
- Réseau de transport des données
- Applications Node-red, InfluxDB et Grafana

2.3. Solution proposés

Nous allons concevoir et déployer un réseau de collecte de données à savoir la température et l'humidité du sol grâce au protocole LORA et au réseau LORAWAN. Ces données seront ensuite envoyées vers le réseau TTN afin de pouvoir les stocker et de pouvoir piloter les objets connectés. Nous allons enfin démarrer la valve électrique pour arroser la plante en fonction des données recueillies.

3. Mise en place de l'architecture de déploiement

3.1. Architecture Logique

Un réseau LORA est composé de plusieurs éléments qui sont comme suit:

- **Devices ou objets connectés:**

Ce sont des matériaux électroniques généralement des systèmes embarqués qui sont connectés à des capteurs permettant de récolter des données et de les envoyer sur le réseau LoRa mais peuvent aussi être contrôlés pour accomplir des tâches diverses pour améliorer la vie quotidienne comme ouvrir une porte, détecter la présence d'un objet, déclencher un moteur ect.

Ces objets peuvent être des cartes arduino, des cartes raspberry pi et autres.

- **Gateway ou passerelle:**

Ce sont des matériaux électroniques qui permettent d'assurer la communication entre les objets connectés et le réseau TTN des objets connectés. Ils servent de passerelles et sont connectés au réseau LORAWAN et aussi à Internet.

- **Network ou le réseau TTN:**

C'est un réseau des objets connectés qui nous permet de gérer les applications et les objets connectés sur nos propres serveurs.

Cela permet alors de recueillir les données et de les stocker sur nos serveurs personnels et aussi de piloter les objets connectés à distance grâce au réseau TTN.

- **Application:**

Sur le réseau TTN, nous pouvons créer des applications pour gérer nos objets connectés et échanger des données entre les objets connectés et le réseau TTN.

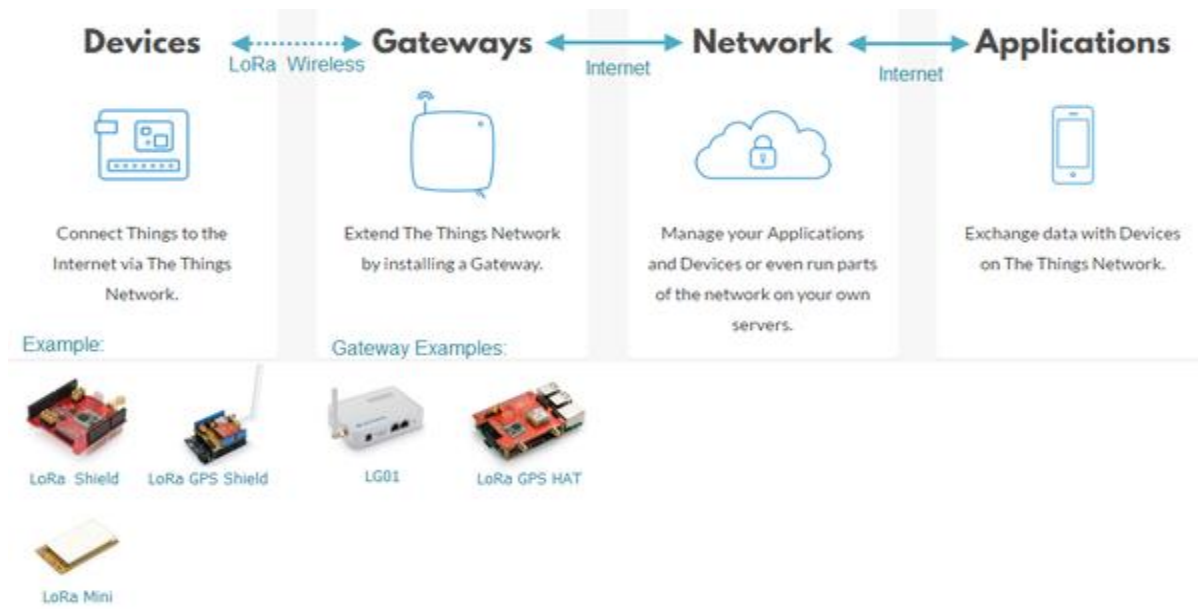


Figure 1: Architecture logique du réseau LORA

3.2. Architecture Physique

Pour implémenter notre solution l'architecture finale proposée se présente comme suit:

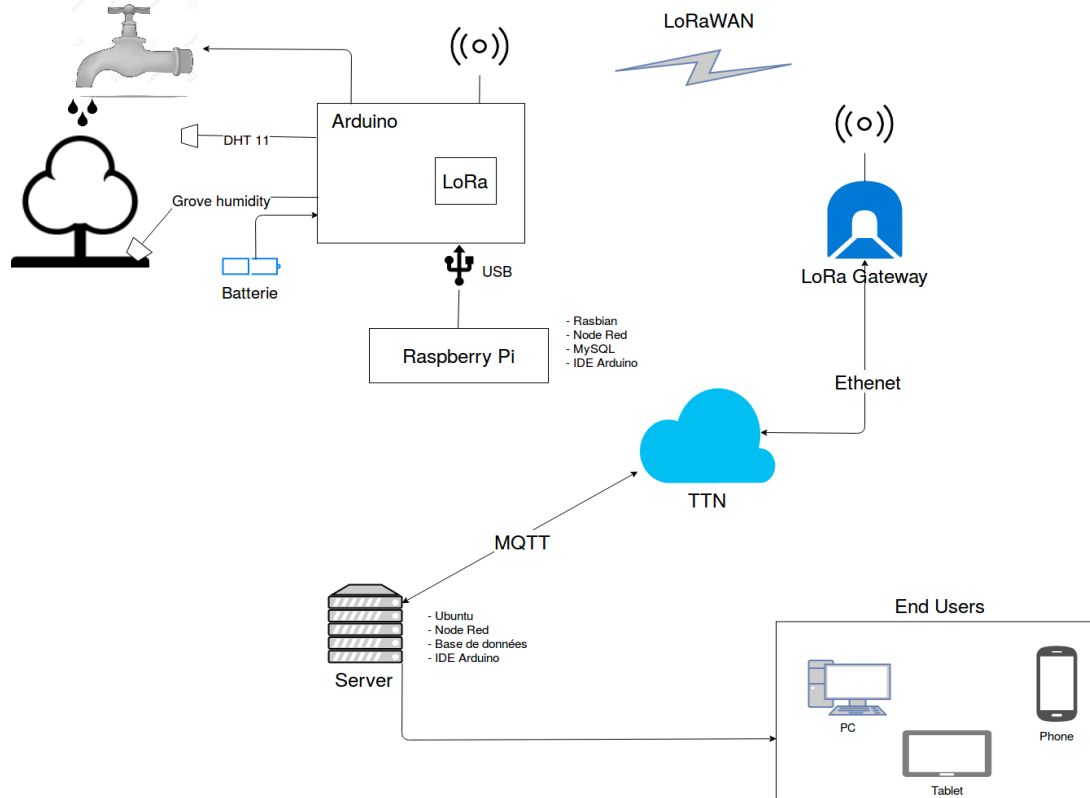


Figure 2: Architecture physique proposée

Cette architecture est composée des éléments suivants:

- Une carte arduino MKRWAN 1300 connectée à des capteurs d'humidité et de température et à une valve électrique qui peut être pilotée pour arroser une culture,
- Une Passerelle Lora TTN qui permet de transférer les données recueillies par la carte arduino vers le réseau TTN,

- Un serveur de base de données influxDB pour stocker les données des capteurs,
- Un **serveur Node-Red** pour traiter les données reçues et les formater, pour implémenter les procédures de stockage des données et de pilotage des objets connectés,
- Des Utilisateurs finaux au travers de PC, Tablette ou Téléphones qui peuvent visualiser les données et piloter la valve électrique à distance.

3.3. Conception et réalisation du montage

Un montage de l'architecture de collecte des données se présente comme suit:

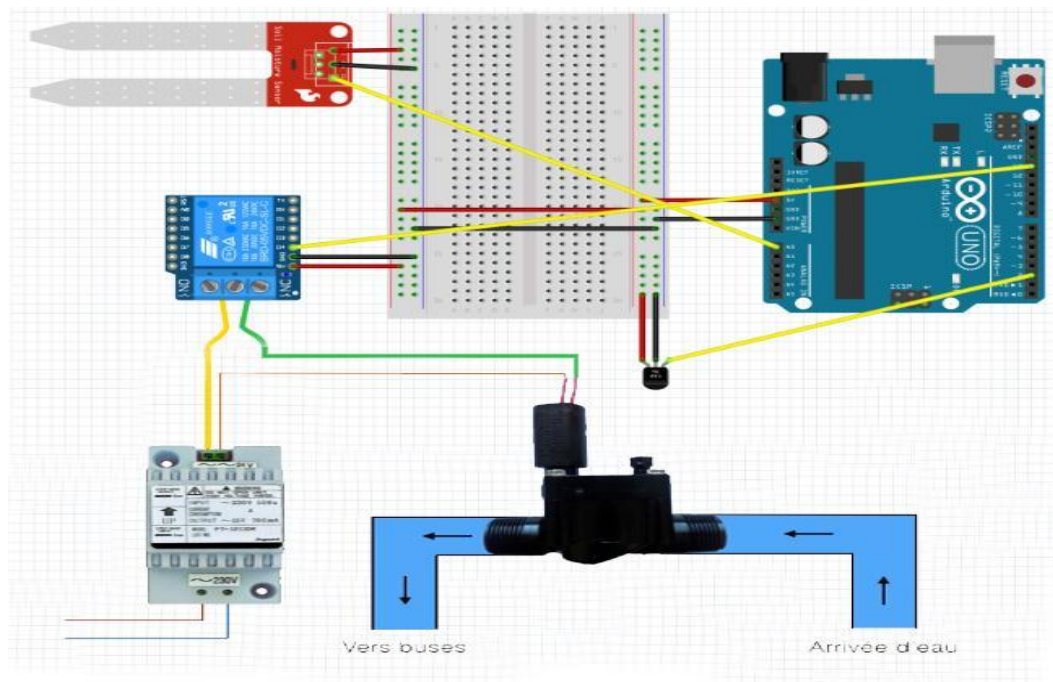


Figure 3 : Schéma du montage

Il est composé d'une carte arduino, De capteur de température et d'humidité, d'une valve électrique pour déclencher l'arrosage et d'un relai.

Le montage réalisé est comme suit:

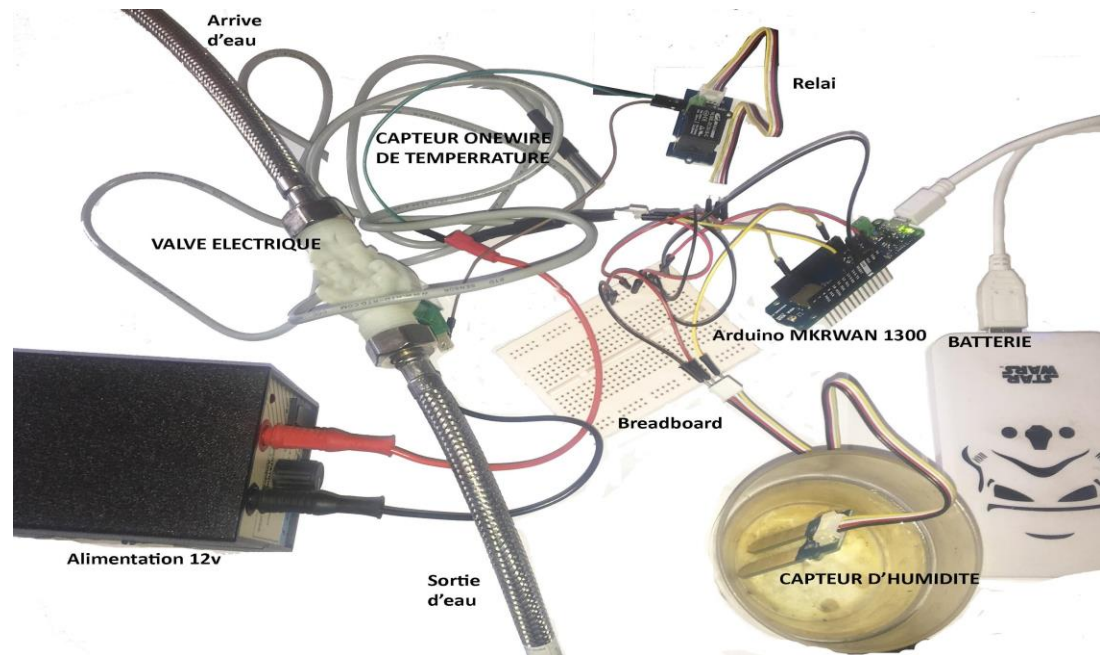


Figure 4: Montage réalisé

4. Implémentation de l'architecture proposée

4.1. Programme arduino

Pour programmer la carte arduino MKR WAN 1300, nous avons écrit un programme arduino en C++ qui est composé d'instruction à savoir la déclaration des variable et des librairies, une fonction SETUP et une fonction LOOP.

La Fonction SETUP permet de faire la configuration initiale de la carte arduino et des différents PINs en Input ou output, la récupération de l'identifiant de l'objet et la connexion au serveur TTN.

```

void setup() {
  Serial.begin(115200);
  sensors.begin();
  pinMode(moteurPin, OUTPUT);
  while (!Serial);
  // selection de la frequence correspondante à la zone (eg. US915, AS923, ...)
  if (!modem.begin(EU868)) {
    Serial.println("Echec de demarage du module");
    while (1) {}
  };
  // _____ Recuperation de l'identifiant de l'objet
  Serial.print("Votre version est : ");
  Serial.println(modem.version());
  Serial.print("L'identifiant EIU de votre materiel est : ");
  Serial.println(modem.deviceEUI());
  // _____ Connexion au serveur TTN
  int connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    Serial.println("Une erreur s'est produite. Etes vous dans un batiment? Deplacez vous vers la fenetre");
    while (1) {}
  }
  // Fixe l'interval de temps à 120 secs.
  modem.minPollInterval(120);
}

```

Figure 5 : La fonction SETUP

La fonction LOOP est exécutée de façon itérative et est constituée d'un ensemble d'instruction permettant de faire des tâches diverses:

- **Collecte des données de température et d'humidité:**

Ce code permet de récupérer les données d'humidité et de température recueillies grâce aux capteurs onewire de température et le capteur d'humidité. Elles sont ensuite concaténées dans la variable *result* qui sera ensuite envoyé vers le serveur TTN.

```

void loop() {
  //_____Recuperation des données de temperature
  Serial.println();
  Serial.println("Entrer le message à envoyer au reseau");
  Serial.println("(taper entrer apres avoir renseigner la donnée)");
  Serial.println("La temperature est: ");
  sensors.requestTemperatures();

  Serial.print(sensors.getTempCByIndex(0));
  //////////////// // HUMIDITE
  int temperature;
  temperature=sensors.getTempCByIndex(0);

  result = result + temperature;
  //_____Recuperation des données de l'humidité
  sensorValue = analogRead(sensorPin);
  Serial.print("Moisture = " );

  sensorValue =map(sensorValue, 0,680,0,100);
  Serial.println(sensorValue);
  Serial.print("\nHumidity _____\n");
  Serial.print(sensorValue);
  result = result + sensorValue;

  Serial.println(result);
}

```

Figure 6 : Code de récupération des données des capteurs

- Envoi des données des capteurs vers le Serveur TTN en UPLINK

Les données sont envoyées au serveur TTN grâce à l'objet *modem* de la librairie du *MKRWAN*.

```

//_____Envoie des données vers le serveur TTN en UPLINK
int err;
modem.beginPacket();
modem.print(result);
result="";

err = modem.endPacket(true);
if (err > 0) {
  Serial.println("Message envoyé avec succès!");
} else {
  Serial.println("Erreur lors de l'envoi du message :(");
  Serial.println("Vous pouvez envoyer un nombre limité de message par minute, en fonction de la force du signal");
  Serial.println("Cela peut varier d'un message toutes les dix secondes à un message toutes les minutes");
}
}

```

Figure 7: Code d'envoi de données vers TTN

- Traitement des données reçus depuis TTN:

Ce code permet de lire les données reçues sur la carte arduino à partir du réseau TTN. Cette donnée est envoyée depuis le serveur Node-Red par l'utilisateur.

```
// _____Traitement des données reçues du serveur TTN en DOWNLINK
if (!modem.available()) {
  Serial.println("Aucun message reçu en downlink pour le moment.");
  return;
}
char rcv[64];
int i = 0;
while (modem.available()) {
  rcv[i++] = (char)modem.read();
}
Serial.print("Received: ");

Serial.print("\nLe caractère 1 reçu est : "); // Variable servant à récupérer les données reçues
Serial.print(rcv[0]);
char rcv1;
Serial.print("\nLe caractère 2 reçu est : ");
rcv1=rcv[0] >> 4;
Serial.print(rcv1);
char received;
received=rcv[0];
Serial.print(received);
```

Figure 8: code de traitements des données de Downlink

- Pilotage de la valve électrique

On déclenche la valve pendant une minute lorsque la donnée reçue en Downlink est 'A' ou que la température > 20 ° C ou que l'humidité < 20 %.


```

// _____ Pilotage de la valve electrique
if(rcv[0] == 'A' or temperature > 20 or sensorValue < 20)
{
    // Si "A" est reçu
    Serial.print("Et que la Moteur est allumée");
    if (moteurState==HIGH)
    {
        // Et que la Moteur est allumée
        Serial.print("Et que la Moteur est allumée");
        moteurState = LOW;
        digitalWrite(moteurPin, moteurState);
        delay(60000);
        moteurState = HIGH;
        digitalWrite(moteurPin, moteurState);// On l'éteint
    }
    else
    {
        //Sinon
        moteurState = HIGH;
        digitalWrite(moteurPin, moteurState);
        delay(60000);
        moteurState = LOW;
        digitalWrite(moteurPin, moteurState);// On l'allume
    }
    digitalWrite(moteurPin, moteurState); // Enfin on change l'état de la Moteur
}
// }
Serial.print(rcv);

```

Figure 9 : Programme de pilotage de l'électrovanne

4.2 Application Cloud TTN

The Things Network (TTN) est un ensemble d'outils ouverts et un réseau mondial pour construire une application IoT à faible coût en utilisant la technologie LoRa, avec une sécurité maximale et une capacité d'évolution.

C'est une application cloud gratuite et Open Source. En revanche, le fait d'utiliser TTN impose à ceux qui enregistrent des Gateways LoRa de les mettre à disposition de tous les autres utilisateurs de TTN. L'objectif de la mise en place de cette application est de réaliser un réseau global ouvert.

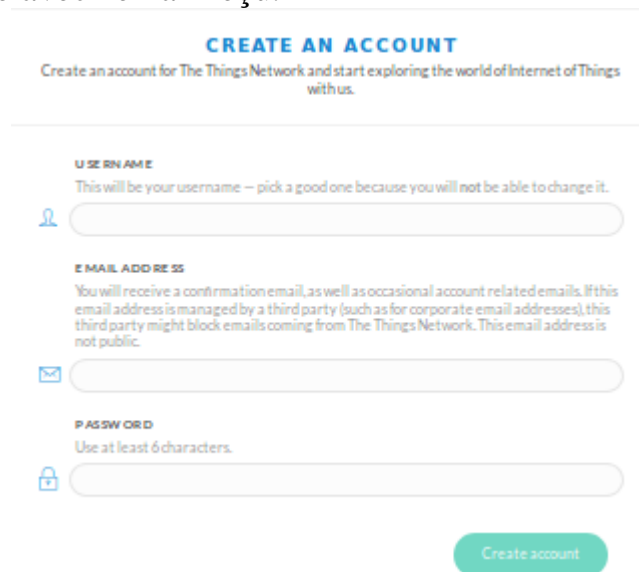
Pour pouvoir utiliser l'application TTN il faut d'abord créer un compte sur le site web <https://account.thethingsnetwork.org/register>. Le compte va permettre d'identifier l'utilisateur sur l'application cloud mais aussi d'identifier les données de l'utilisateur dans le réseau LoRa qui est ouvert et gratuit.

Ensuite enregistrer le matériel Iot qui va échanger des données avec l'application cloud via le réseau LoRa. L'enregistrement est précédé de la création de l'équipement sur l'application qui sera créée sur le cloud.

Cependant il est important de savoir que les données reçues de la carte *arduino MKR 1300* est sous format hexadécimal donc il est nécessaire de mettre en place une fonction de décodage pour transformer les données en ascii.

4.2.1 Création du compte TTN

La création du compte est une étape très simple, il faut se rendre sur le site web officiel puis cliquer sur **Sign In** (lien <https://account.thethingsnetwork.org/register>), renseigner les informations demandées (login, email et mot de passe) et enfin valider la création du compte avec l'email reçu.



The screenshot shows the 'CREATE AN ACCOUNT' page for The Things Network. The page has a light blue header with the title 'CREATE AN ACCOUNT' and a subtitle 'Create an account for The Things Network and start exploring the world of Internet of Things with us.' Below the header, there are three input fields: 'USERNAME' with a user icon, 'EMAIL ADDRESS' with an email icon, and 'PASSWORD' with a lock icon. Each field has a small icon to its left and a descriptive text below it. The 'USERNAME' field has the text 'This will be your username -- pick a good one because you will not be able to change it.' The 'EMAIL ADDRESS' field has the text 'You will receive a confirmation email, as well as occasional account related emails. If this email address is managed by a third party (such as for corporate email addresses), this third party might block emails coming from The Things Network. This email address is not public.' The 'PASSWORD' field has the text 'Use at least 6 characters.' At the bottom right, there is a green 'Create account' button.

Figure 10 : Création compte TTN

Le compte créé va permettre de créer des applications qui peuvent être des projets IoT différentes et chaque application contient une ou plusieurs équipements qui sont des matériels IoT connectés au réseau LoRaWAN.

4.2.2 Création de l'application

La création de l'application nécessite de se connecter à votre compte TTN puis se rendre sur le console qui va se présenter comme suit :

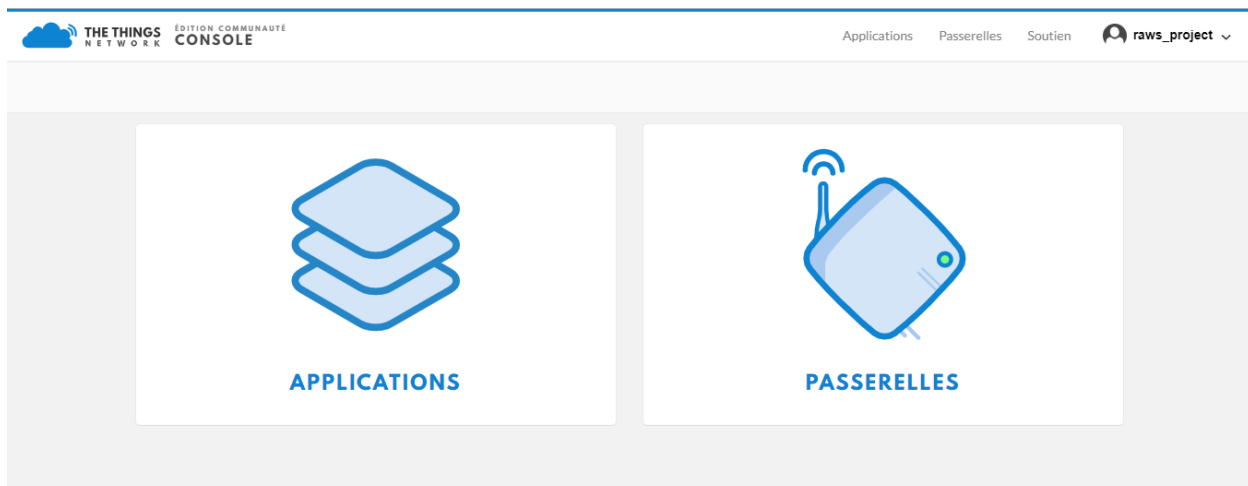


Figure 11: Page d'accueil de l'application sur TTN

Pour ajouter une application cliquer sur **APPLICATIONS**, puis renseigner les informations demandées à savoir l'identifiant de l'application qui ne doit pas contenir d'espace et la description de l'application (les autres informations ne sont pas nécessaires à renseigner) et enfin enregistrer.

Figure 12: Création de l'application

L'application ainsi créée contiendra le ou les différent(s) équipement(s) utilisé(s). Dans notre cas elle contiendra comme matériel la carte arduino MKR qui va collecter

et envoyer les données à notre serveur via le réseau LoRa.

4.2.3 Création de l'équipement

L'équipement qui sera créé est le matériel connecté à l'autre bout qui permet d'envoyer des données vers l'application TTN ou de recevoir des informations de TTN. Dans ce projet l'équipement dont il s'agit est la carte arduino MKR 1300.

L'équipement est créé en allant dans l'application créer sur la rubrique device cliquez sur *registered device*.

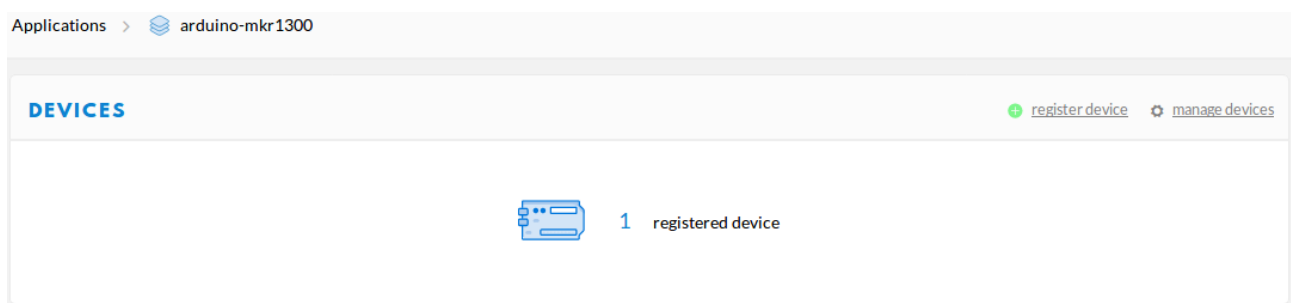


Figure 13 : Rubrique device

Il reste plus qu'à renseigner le nom du device et enregistrer.

The screenshot shows the 'REGISTER DEVICE' form. At the top right, there's a link 'bulk import devices'. The form has four main sections: 1. 'Device ID' with a description 'This is the unique identifier for the device in this app. The device ID will be immutable.' and a text input field containing 'aduino-sensors'. 2. 'Device EUI' with a description 'The device EUI is the unique identifier for this device on the network. You can change the EUI later.' and a text input field containing 'this field will be generated'. 3. 'App Key' with a description 'The App Key will be used to secure the communication between you device and the network.' and a text input field containing 'this field will be generated'. 4. 'App EUI' with a text input field containing '70 B3 D5 7E D0 01 A7 2B'. At the bottom right, there are two buttons: 'Cancel' and 'Register'.

Figure 14 : Création du matériel

Maintenant que l'application et l'équipement sont créés, l'équipement physique c'est à dire la carte *Arduino MKR 1300* pourra envoyer les données collectées à l'équipement TTN.

4.2.4 Formatage des données

Il faut savoir que par défaut les informations reçues en **uplink** par TTN sont sous le format **hexadécimal** d'où l'importance de formater ces informations pour ressortir les données température et humidité envoyées par la carte arduino.

Les données reçues sans le formatage seront présentées sous la forme suivant:

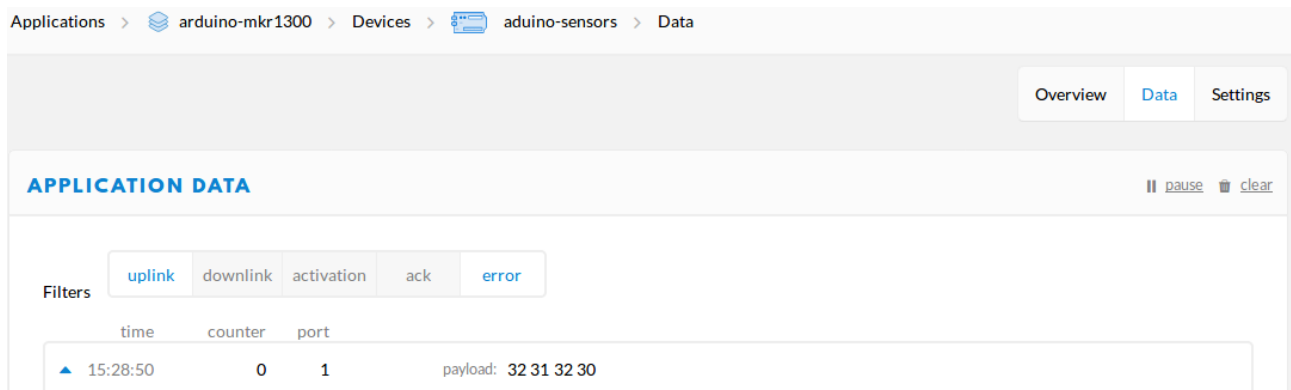


Figure 15 : Data Application

Au niveau de l'application créée sur TTN, on trouve une rubrique **Payload Formats** qui contient une autre rubrique **decoder** qui permettra d'écrire un programme en JavaScript pour **decoder** les données reçues en **uplink** c'est à dire de la carte arduino vers TTN.



Figure 16 : Fonction de décodage

Et après l'ajout de la fonction décodé e les informations vont se présenter sous la forme suivante:

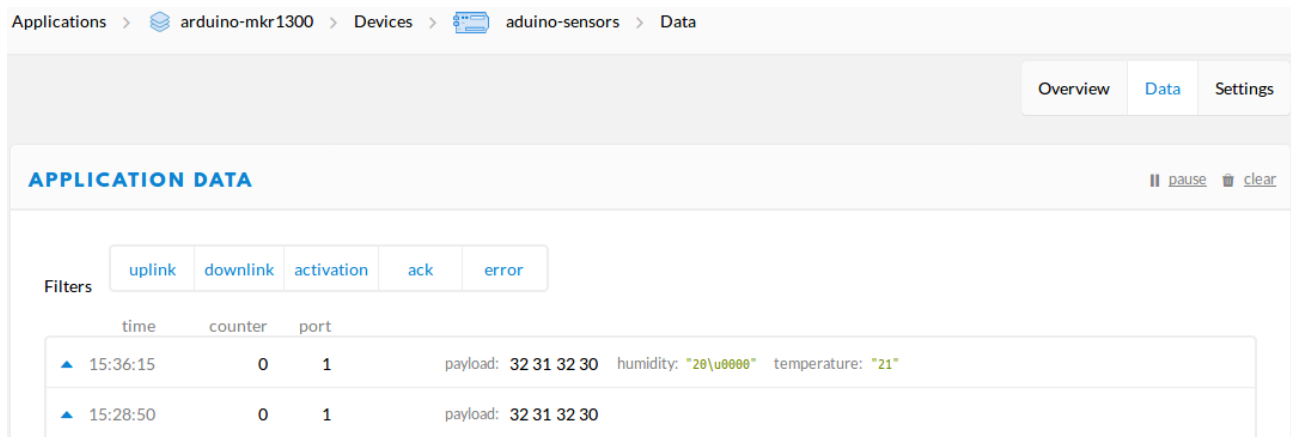


Figure 17 : Résultat du décodage

4.3. Application Node-Red

Node-red est une application codée en **JavaScript** qui permet de visualiser et de traiter des données. Cependant il existe d'autres moyens de traitement et de visualisation en IoT beaucoup plus sophistiqué qui sont utilisés par les industrielles qui traite du **BIG DATA** comme *Cayenne*, *IBM Watson IoT Platform*, *AT&T M2X* ... Puisque nous avons un projet qui ne traite pas méga donnée *node-red* répond à nos exigences. Elle est très simple à utiliser et intègre des modules à installer pour un traitement spécifique.

Pour le projet on va ajouter le module **MQTT** pour les informations en *uplink* de TTN vers la base de données influxDB et en *downlink* de l'interface utilisateur vers la carte *arduino MKR*. Tous les nodes seront dans un même un flow. Un flow est la plage qui intègre tous les nodes à utiliser pour la réalisation. Le *flow* se présente comme suit:

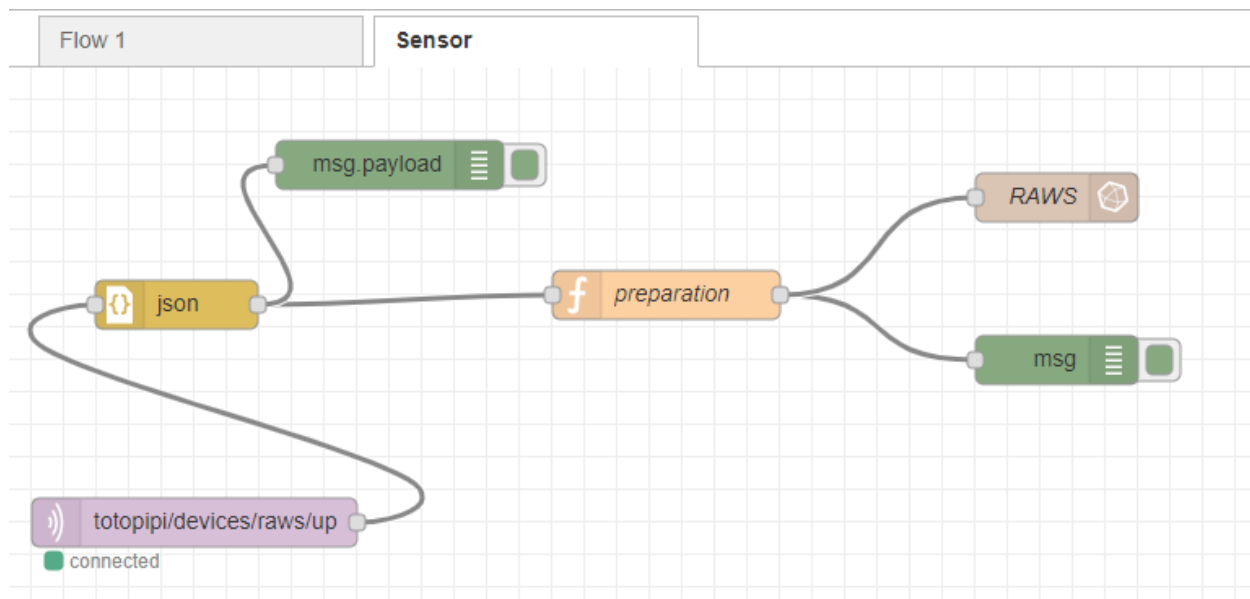


Figure 18 : Flow uplink de TTN vers la base de données



Figure 19 : Flow downlink de l'interface utilisateur et la carte Arduino MKR

4.3.1 Ajouter des palettes

L'ajout des palettes se fait soit en ligne de commande soit sur l'interface de l'application. La deuxième méthode sera utilisée car elle est la plus simple.

Avant tout il faut démarrer l'application en ligne de commande avec la commande suivante **node-red start** puis accéder à l'application au niveau du navigateur et renseigner le lien <http://localhost:1880>. Enfin installer les palettes en allant sur le **menu de l'application** puis cliquer sur **manager palettes** et installer les modules suivantes: **node-red-node-serialport**, **node-red-dashboard** et **node-red-node-influxDB**. Le node serialport permet de faire les tests en local, le node dashboard permet de visualiser les données reçues sous forme de graphes, gauges, etc et le node influxDB permet se connecter à la base de données influxDB et également de créer les tables de la base et d'enregistrer les données.

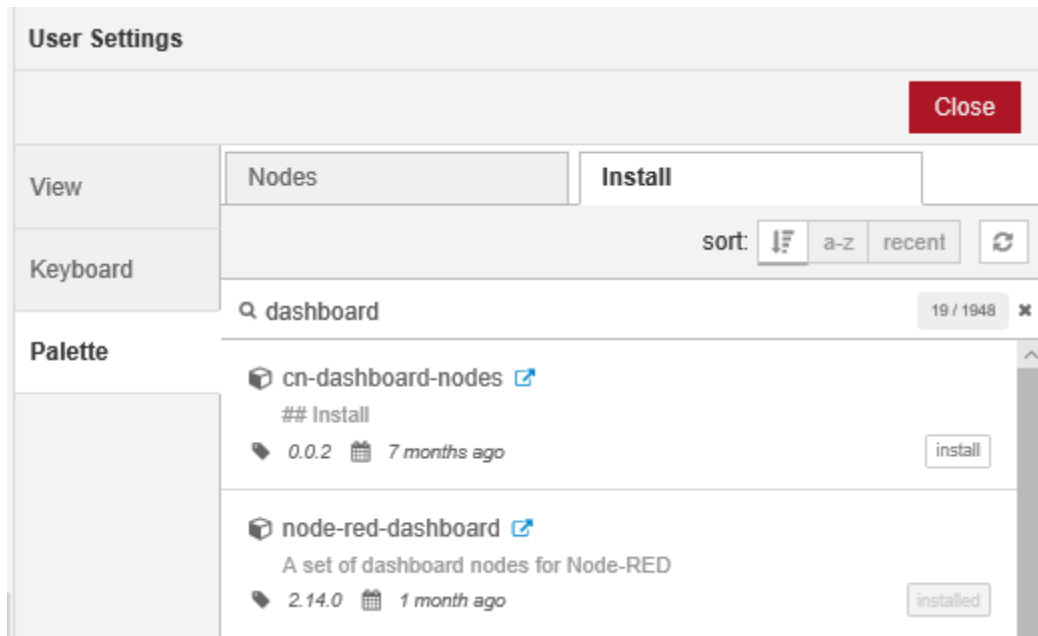


Figure 20 : Installation des nodes

Après l'ajout des modules, on peut observer la présence des nouveaux nodes à gauche de l'interface de node-red.

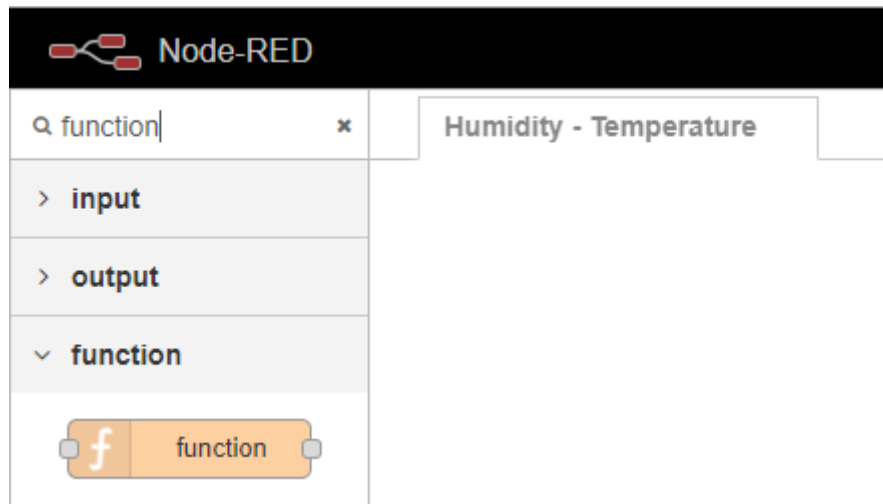


Figure 21 : Ajout d'un node

4.3.2 Le protocole MQTT et configuration du node

C'est un protocole qui est utilisé dans les réseaux IoT et est utilisé avec différents technologies comme le raspberry pi.

Dans ce projet le protocole MQTT (Mosquitto) intervient entre le serveur TTN et l'application node-red. Il permet de transmettre des informations en uplink comme en downlink entre l'application TTN et la base de donnée influxDB ou de l'interface utilisateur Grafana et l'application TTN.

Node-red intègre par défaut les nodes MQTT en uplink et en downlink qui vont permettre de remplir les fonctions d'envoi et de réception d'informations.

En uplink les données seront sous forme de chaîne de caractère, de ce fait, il faut une fonction JSON pour mettre les données dans un objet json et en downlink un bouton sera mis en place pour l'envoi de l'information 'A' qui va permet à la carte d'actionner l'ouverture de l'électrovanne.

- **Configuration de MQTT en uplink:**

Après l'ajout du node MQTT au flow, il faut renseigner les informations nécessaires pour le faire connecter à TTN. Les informations de connexion de TTN seront retrouvées dans l'API référence du site de officiel de TTN <https://www.thethingsnetwork.org/docs/applications/mqtt/api.html>. Il faut d'abord configurer le serveur avec les informations suivantes:

- **Hôte:** eu.thethings.network où eu est la <Region> qui est la dernière partie du gestionnaire que vous avez enregistré votre application.
- **Port:** 1883 ou 8883 pour TLS

Il faut renseigner les informations de connexion à savoir le serveur, le client et le port trouvés dans l'API.

Edit mqtt out node > **Edit mqtt-broker node**

Delete Cancel Update

Name

Connection Security Messages

Server Port

☐ Enable secure (SSL/TLS) connection

Client ID

Keep alive time (s) ☒ Use clean session

☒ Use legacy MQTT 3.1 support

Figure 22 : Node mqtt connexion

- **Username:** l'identifiant de l'application TTN est le username.

Applications > arduino-mkr1300

APPLICATION OVERVIEW

Application ID arduino-mkr1300 [documentation](#)

Figure 23 : TTN Application ID

- **Password:** La clef d'accès de l'application TTN doit être renseignée.

Applications > arduino-mkr1300

ACCESS KEYS [manage keys](#)

default key devices messages

Figure 24 : TTN default key devices messages

À ce niveau il faut renseigner le username et le mot de passe pour avoir accès aux données reçues par TTN.

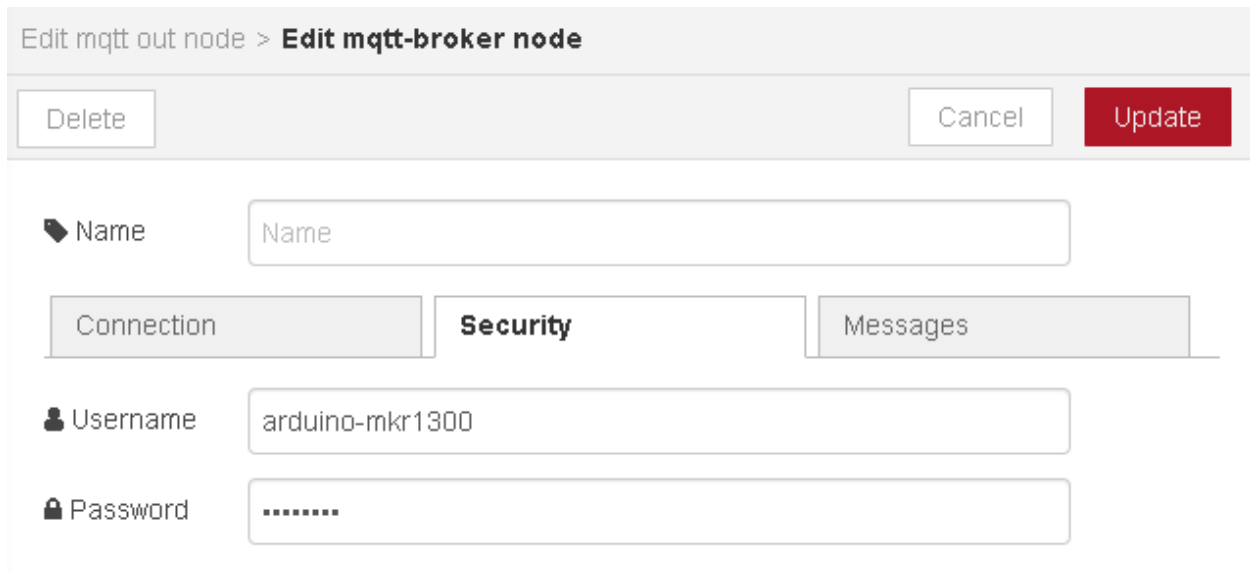


Figure 25 : Configuration de la sécurité

Ces informations permettent d'identifier les messages ou paquets qui sont destinés à l'utilisateur.

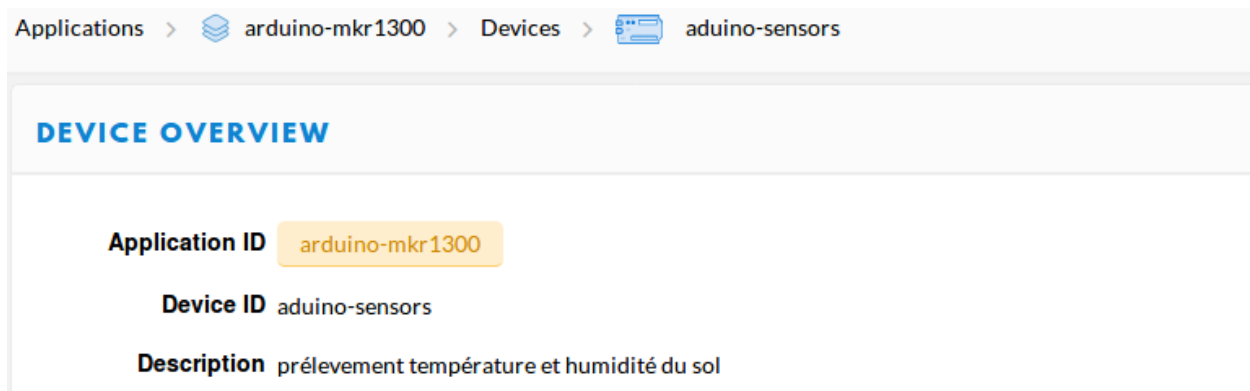


Figure 26 : Node MQTT configuration serveur

Ensuite il faut configurer le **topic** avec les informations en les renseignant comme indiqué ci-dessous.

Edit mqtt out node

Delete

Cancel

Done

▼ node properties

🌐 Server

arduino-mkr1300@eu.thethings.network ▼

✎

📄 Topic

arduino-mkr1300/devices/arduino-sensors/up

🌐 QoS

▼

🔄 Retain

▼

🏷 Name

Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Figure 27 : Node MQTT configuration en uplink

Cette configuration permet de spécifier les informations en uplink.

- **Configuration de MQTT en downlink:**

La seule différence entre la configuration de l'uplink et du downlink est la configuration du topic. La configuration est en uplink pour le mettre en downlink il faut juste remplacer *up* par *down*.

28

Edit mqtt out node

Delete
Cancel
Done

▼
node properties

Server
arduino-mkr1300@eu.thethings.network

Topic
arduino-mkr1300/devices/aduino-sensors/down

QoS

Retain

Name
Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

Figure 28 : Node MQTT configuration en downlink

- Configuration du bouton:

Le bouton permet de déclencher la descente de l'information 'A' qui sera envoyé à la carte MKR pour actionner l'électrovanne pour arroser la plante. Il faut noter que la node MQTT prendre une valeur spécifique en entrée qui est un objet json qui contient deux variables:

- la variable port: pour spécifier le *port* d'accès
- et la variable *payload_raw* qui contient l'information à envoyer en base 64. 'A' en base64 vaut "QQ=="

La configuration du node se fait comme suit:

29

Edit button node

Delete Cancel Done

▼ **node properties**

Icon optional icon

Label button

Tooltip optional tooltip

Colour optional text/icon color

Background optional background color

☒ When clicked, send:

Payload {"port": 1,"payload_raw": "QQ=="}

Figure 29 : Configuration Node button

La configuration des nodes est terminée maintenant, on peut déployer le flow. En actionnant le bouton sur l'interface <http://localhost:1880/ui>, on voit sur le device de TTN la donnée reçue en hexadécimal:

Applications > arduino-mkr1300 > Devices > arduino-sensors > Data

APPLICATION DATA					pause	🗑 clear
Filters						
uplink downlink activation ack error						
time	counter	port				
15:48:37	1	scheduled	payload: 41			

Figure 30 : Résultat configuration bouton

4.3.3 Influxdb et configuration du node

InfluxDB a été conçue pour conserver, analyser et injecter en temps réel des données de séries temporelles (métriques et événements) avec de fortes exigences en termes de disponibilité et de performances.

Comparer aux autres technologies, elle est la plus adaptée au réseau IoT raison pour laquelle il est utilisé pour réalisation de ce projet.

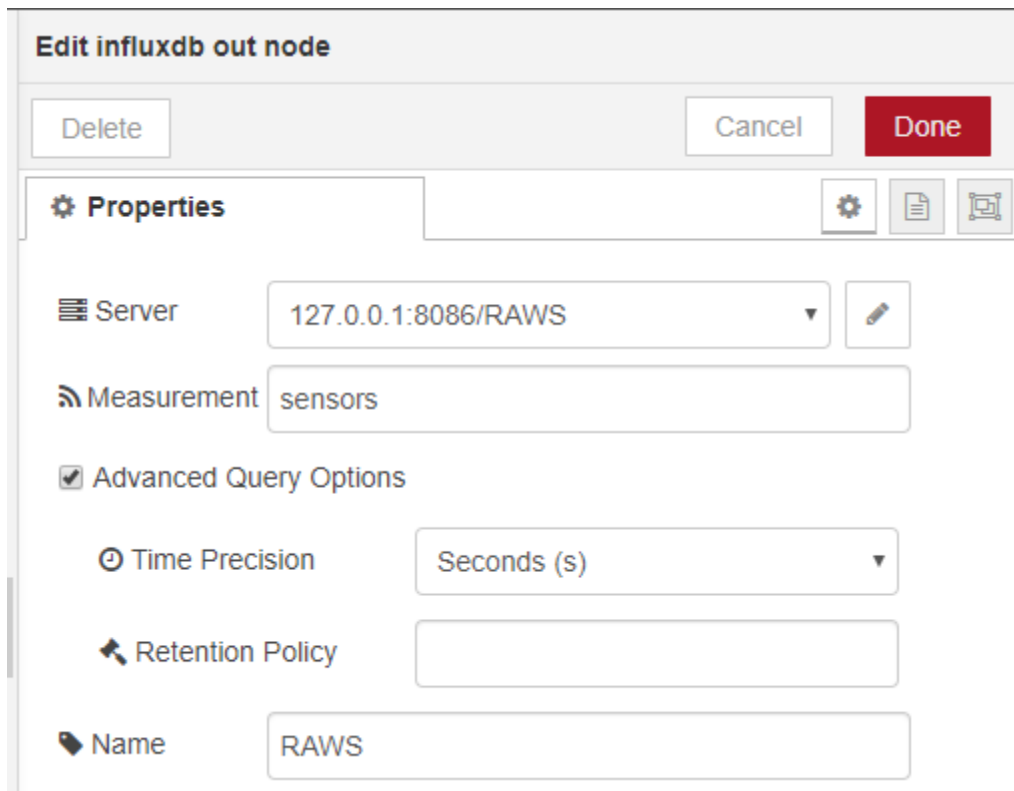
La configuration de la base est une chose très facile, il suffit juste de créer une base de données au niveau d'influxDB et le node influxDB va se charger de la création des tables et de l'insertion des données.

- **Création de la base de données:**

InfluxDB est accessible en ligne de commande de ce fait la création de la base de données se fait en console et la commande de création de la base est identique à celle de **SQL** *create database nom base*. Le nom de la base de données est **RAWS**.

- **Configuration de la node:**

Après ajout du node de la base de donnée influxDB dans le **flow** du projet, il faut juste le configurer en renseignant l'*adresse ip du serveur localhost*, le *port* utilisé **8086**, le nom la base de donnée **RAWS** et le nom de la table **sensors**.



Edit influxdb out node

Delete Cancel Done

Properties

Server 127.0.0.1:8086/RAWS

Measurement sensors

☒ Advanced Query Options

Time Precision Seconds (s)

Retention Policy

Name RAWS

Figure: Configuration node influxDB

Il faut noter que la table **sensors** sera créée automatiquement lors du déploiement du flow et les attributs de la table seront créés en fonction de la valeur d'entrée du node. C'est pourquoi il est nécessaire de formater la valeur d'entrée.

Le formatage des données est fait grâce au node *function* qui reçoit les données reçues de MQTT sous format JSON.


```

let temperature = msg.payload["payload_fields"].temperature;
let hum = msg.payload["payload_fields"].humidity;

let appId = msg.payload.app_id;
let devId = msg.payload.dev_id;
let hardware = msg.payload.hardware_serial;
let port = msg.payload.port;
let lat = msg.payload.metadata.gateways[0].latitude;
let long = msg.payload.metadata.gateways[0].longitude;
let channel = msg.payload.metadata.gateways[0].channel;
let rssi = msg.payload.metadata.gateways[0].rssi;
let snr = msg.payload.metadata.gateways[0].snr;

msg.payload = {
  temperature: parseInt(temperature),
  hum: parseInt(hum),
  appId: appId,
  devId: devId,
  hardware: hardware,
  port: port,
  lat: lat,
  long: long,
  channel: channel,
  rssi: rssi,
  snr: snr
}

```

Figure 31: Fonction formatage des données à enregistrer

Les données reçues contiennent beaucoup d'informations de ce fait on extrait juste les informations nécessaires: la température, l'humidité, l'identifiant de l'application, l'identifiant du matériel, le matériel du gateway, le port utilisé, la latitude et la longitude du gateway, le canal, la puissance du signal et le bruit. Ces informations sont insérées dans un objet *msg.payload* qui va être retournée au node influxDB et les informations seront automatiquement enregistrées.

4.5 Application Grafana

La surveillance des serveurs se fait souvent à l'aide de graphiques. Grafana propose une interface moderne, personnalisable et est une application open source. Il est utilisé dans le projet pour visualiser toutes les informations intéressantes reçues du gateway LoRaWAN.

4.5.1 Ajouter des données sources

Avant de créer votre premier tableau de bord, vous devez ajouter votre source de données.

Déplacez d'abord votre curseur sur le rouage situé sur le menu latéral qui vous montrera le menu de configuration. Si le menu latéral n'est pas visible, cliquez sur l'icône Grafana dans le coin supérieur gauche. Le premier élément du menu de configuration concerne les sources de données. Cliquez dessus pour accéder à la page des sources de données où vous pouvez ajouter et modifier des sources de données. Vous pouvez également simplement cliquer sur le rouage.

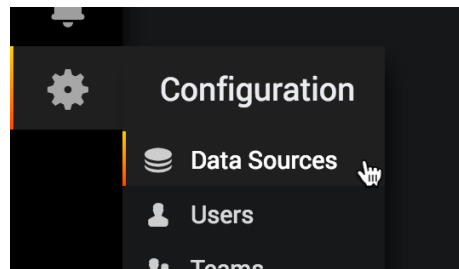


Figure 32: Configuration data source

Cliquez sur Ajouter une source de données et vous arriverez à la page des paramètres de votre nouvelle source de données.

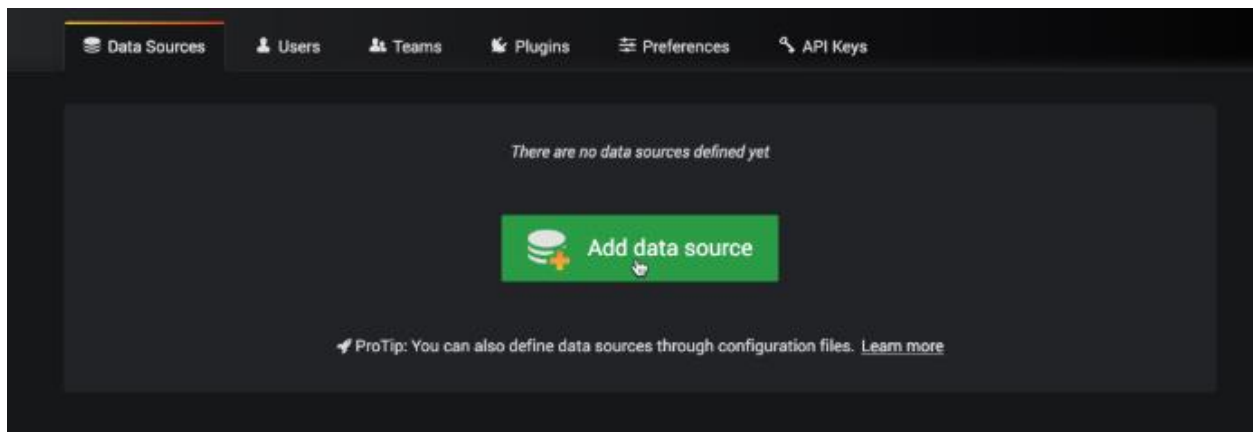


Figure 33 : Configuration data source

Commencez par attribuer un nom à la source de données, puis sélectionnez le type de source de données que vous souhaitez créer

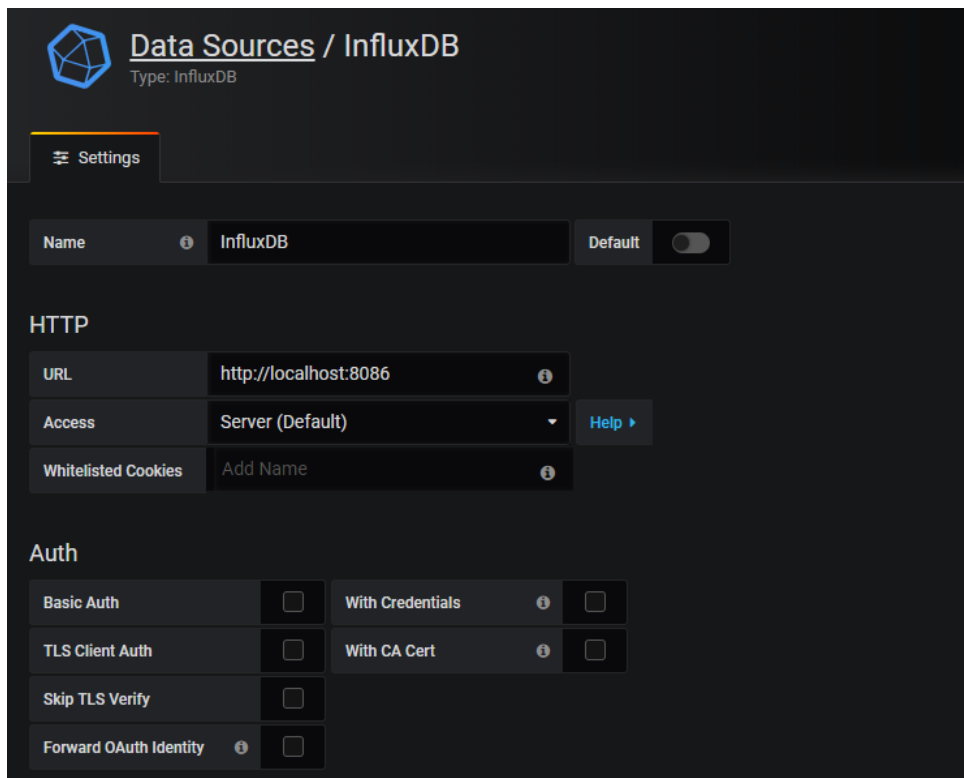


Figure 34 : Configuration InfluxDB

Après avoir configuré votre source de données, vous êtes prêt à enregistrer et à tester.

4.5.2 Conception des Graphes

Grafana offre plusieurs types de graphes.

- Vous ajoutez des panneaux en cliquant sur l'icône Ajouter un panneau dans le menu supérieur.
- Pour modifier le graphique, cliquez sur son titre pour ouvrir le menu du panneau, puis sur Modifier.
- Cela devrait vous mener à l'onglet Métriques. Dans cet onglet, vous devriez voir l'éditeur de votre source de données par défaut.

Lorsque vous cliquez sur l'onglet Métriques, un éditeur de requête spécifique à la source de données du panneau vous est présenté. Utilisez l'éditeur de requêtes pour créer vos requêtes et Grafana les visualiser en temps réel.

- Pour température/humidité graph, on doit choisir le query comme ça:

SELECT temperature FROM sensors

SELECT humidity FROM sensors

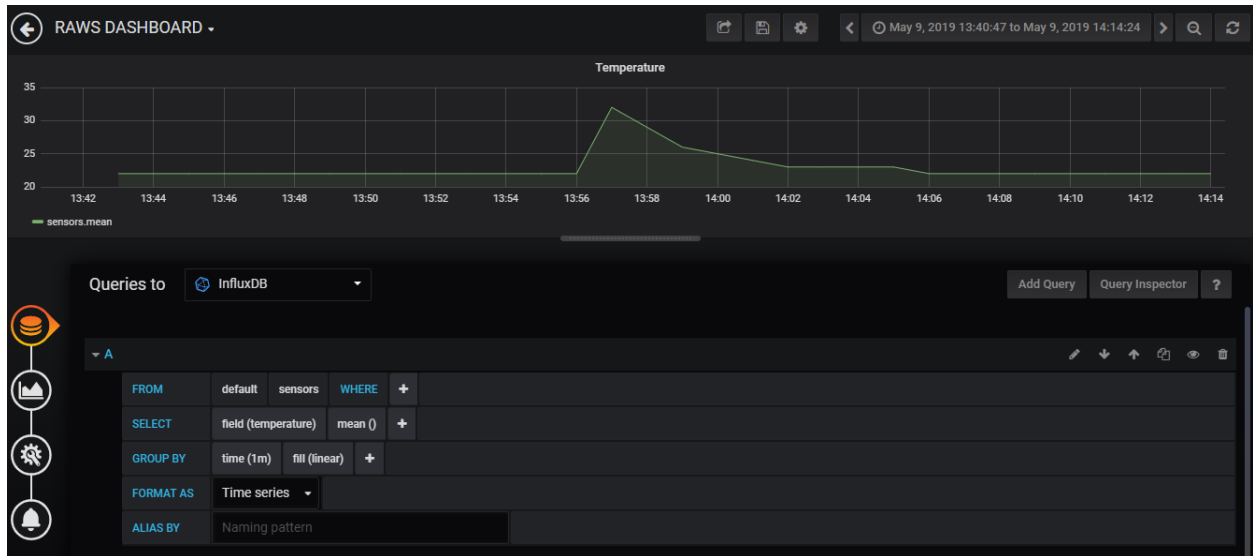


Figure 35 : Configuration de la graphe de température



Figure 36 : Configuration de la graphe d'humidité

- Pour température/humidité actuelle graph, on doit ajouter la condition pour le query comme ça:

ORDER BY DESC

LIMIT 1

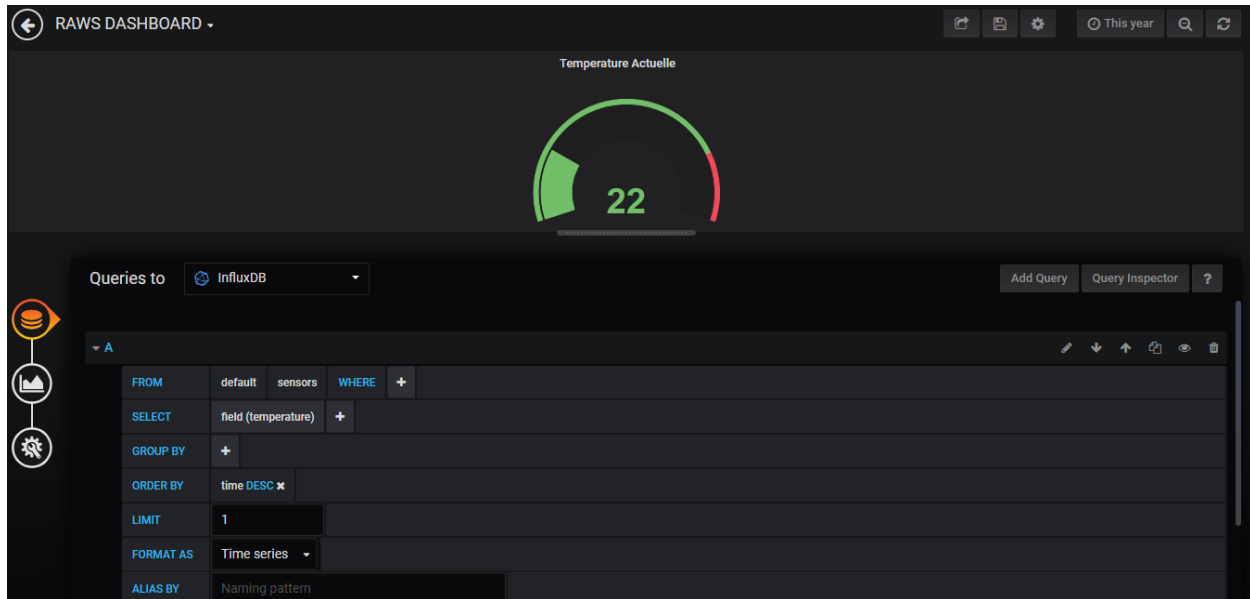


Figure 37: Configuration température actuelle

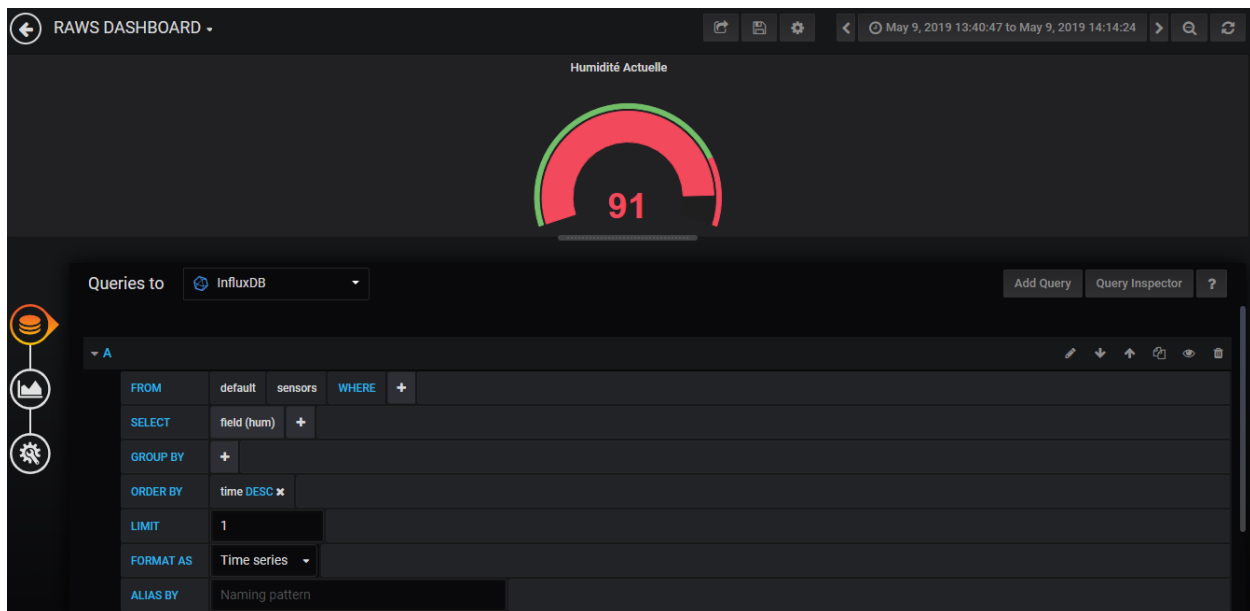


Figure 38: Configuration humidité actuelle

4.6. Test de fonctionnement

Après avoir implémenté notre solution, nous avons implémenté des tests qui nous ont permis de pouvoir allumer la valve électrique soit lorsque la température $> 20^{\circ}\text{C}$ ou que l'humidité du sol est $< 20\%$ ou en cliquant un bouton au travers de l'interface graphique de Node-Red.

Les données collectées ont été enregistrés dans une base de données influxdb. Pour les visualiser on peut utiliser l'application Grafana.

Ce qui donne le résultat suivant:

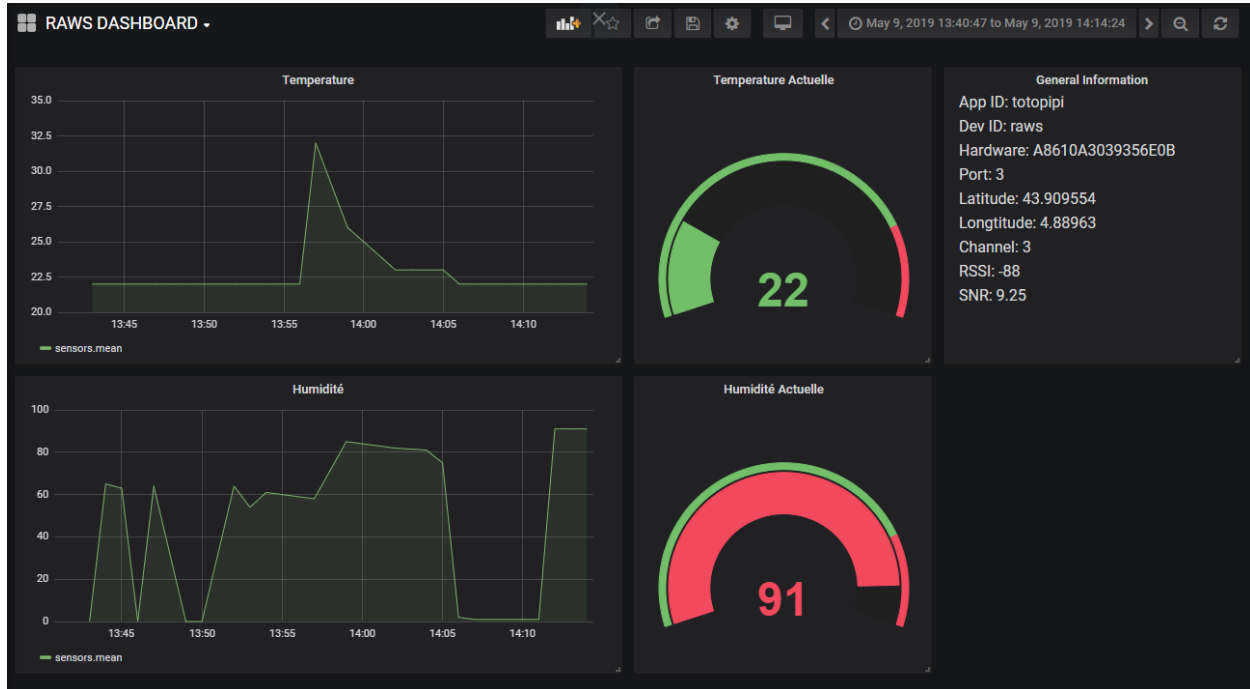


Figure 39 : Visualisation de la température et de l'humidité en temps réel sur Grafana

4.7. Difficultés rencontrées

Les problèmes que nous avons rencontrés sont :

- **Le retard dans la réception du matériel**

En effet le matériel n'étant pas disponible, il a dû être commandé par le CERI et cela a pris du temps pour le livrer et pour qu'on le reçoive. Ce qui nous a retardés dans la réalisation du projet.

- **Le Gateway:**

Pour réaliser le projet nous avons forcément besoin de nous connecter au Gateway. Celui-ci se trouvant au CERI, nous devons nous déplacer sur place pour pouvoir nous y connecter car il n'y a pas de Gateway proche de notre lieu d'habitation.

- **Le nombre limité de message en uplink**

Suite à la réglementation du serveur TTN, nous ne pouvons envoyer qu'un nombre limité de message vers le serveur. Ainsi nous avons limité le nombre de message à un (1) par minute.

Mais il peut arriver d'épuiser notre quota de message pour la journée.

5. CONCLUSION

En fonction des exigences du cahier de charges, nous avons pu réaliser un système automatique d'irrigation à distance avec le protocole LORA, la carte arduino et les données des capteurs, et le Réseaux TTN.

L'implémentation de ce projet nous a permis de prendre en compte les besoins de la plante et de s'approvisionner en eau tout en réduisant le gaspillage.

Ce qui démontre la bonne application l'internet des objets au domaine de l'agriculture.

Ainsi l'internet des objets peut représenter une solution au gaspillage de l'eau sur la planète.

Ce projet va permettre aux agriculteurs de mieux comprendre les besoins de la plante et de l'arroser au moment où cela sera nécessaire automatiquement ou à distance.

Ce qui va accroître la production et palier à la surconsommation en eau.

ANNEXES

Installation de node-red

Pour installer node-red sous une machine linux il faut taper en ligne de commande les instructions suivantes:

1. `sudo npm install -g --unsafe-perm node-red`
2. Node-red start
3. Open <http://localhost:1880>

La documentation de node-red est accessible via ce lien <https://nodered.org/docs/>.

Installation and configuration

- Pour Windows
 - Installation pour Windows

Télécharger le zip fichier dans ce source

<https://grafana.com/grafana/download?platform=windows>

Le fichier zip contient un dossier avec la version actuelle de Grafana. Extrayez ce dossier là où vous voulez que Grafana s'exécute:

- Configuration

Le port Grafana par défaut est **3000**. Ce port nécessite des autorisations supplémentaires sur Windows. Modifiez custom.ini et décommentez l'option de configuration http_port (; est le caractère de commentaire dans les fichiers ini) et remplacez-la par quelque chose comme **8080** ou similaire. Ce port ne devrait pas nécessiter de privilèges Windows supplémentaires.

Identifiant et mot de passe par défaut est **admin / admin**.

Démarrez Grafana en exécutant **grafana-server.exe**, situé dans le répertoire bin, de préférence à partir de la ligne de commande. Si vous voulez exécuter Grafana en tant que service Windows, téléchargez NSSM. Il est très facile d'ajouter Grafana en tant que service Windows à l'aide de cet outil.

```
[32mINFO[0m[05-18|17:53:17] Initializing HTTPServer [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing SqlStore [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Connecting to DB [32mlogger[0m=sqlstore [32mdbtype[0m=sqlite3
[32mINFO[0m[05-18|17:53:17] Starting DB migration [32mlogger[0m=migrator
[32mINFO[0m[05-18|17:53:17] Initializing InternalMetricsService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing SearchService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing PluginManager [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Starting plugin search [32mlogger[0m=plugins
[32mINFO[0m[05-18|17:53:17] Registering plugin [32mlogger[0m=plugins [32mname[0m=SimpleJson
[32mINFO[0m[05-18|17:53:17] Registering plugin [32mlogger[0m=plugins [32mname[0m="Annotation
Panel"
[32mINFO[0m[05-18|17:53:17] Registering plugin [32mlogger[0m=plugins [32mname[0m=AJAX
[32mINFO[0m[05-18|17:53:17] Initializing RenderingService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing AlertingService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing DatasourceCacheService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing HooksService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing LoginService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing QuotaService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing RemoteCache [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing ServerLockService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing TracingService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing UsageStatsService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing UserAuthTokenService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing CleanupService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing NotificationService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing ProvisioningService [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] Initializing Stream Manager [32mlogger[0m=server
[32mINFO[0m[05-18|17:53:17] HTTP Server Listen [32mlogger[0m=http.server [32maddress[0m=0.0.
0.0:3000 [32mprotocol[0m=http [32msubUrl[0m= [32msocket[0m=
```

Figure 40: Démarrez Grafana

Pour exécuter Grafana, ouvrez votre navigateur et accédez au port que vous avez configuré ci-dessus, par exemple <http://localhost:3000/>.

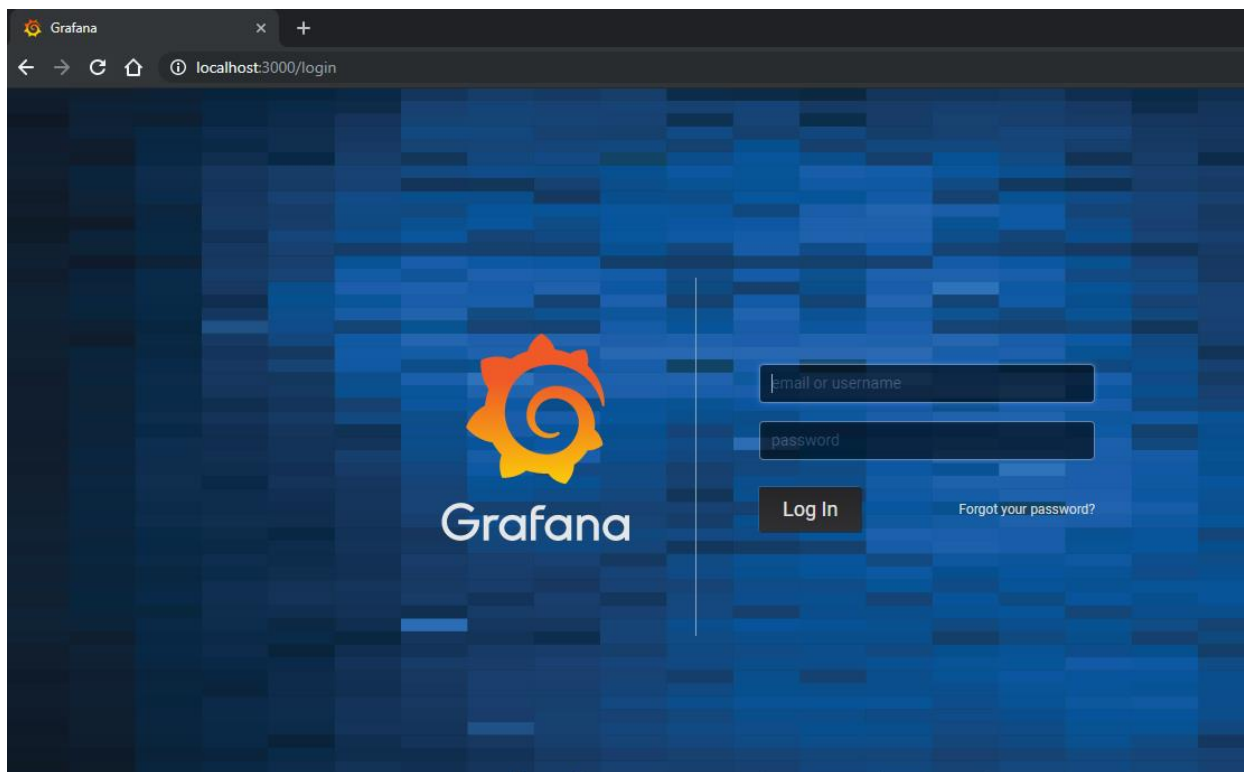


Figure: Page d'accueil Grafana

- Pour Linux/Ubuntu
 - Installation pour Linux/Ubuntu

\$ sudo yum install https://dl.grafana.com/oss/release/grafana-5.4.2-1.x86_64.rpm

Vous pouvez démarrer Grafana en lançant:

\$ sudo service grafana-server start

Cela démarrera le processus grafana-server en tant qu'utilisateur grafana, créé lors de l'installation du paquet. Le port HTTP par défaut est **3000** et l'utilisateur et le groupe par défaut sont admin.

Identifiant et mot de passe par défaut **admin / admin**

- Configuration

Le fichier de configuration est situé à **/etc/grafana/grafana.ini**

WEBOGRAPHIE

www.silanus.fr

www.thethingsnetwork.org

www.grafana.com

www.wikipedia.org

www.seeedstudio.com

<https://lora-alliance.org/about-lorawan>

<https://www.digitalmatter.com/Solutions/Whitepapers/LoRaWAN-Introduction>

<https://nodered.org/docs/getting-started/installation>

<https://www.cooking->

[hacks.com/forum/viewtopic.php?f=37&t=8856&sid=d19572860b6c1cdb2e248a6926b16837&start=10](https://www.cooking-hacks.com/forum/viewtopic.php?f=37&t=8856&sid=d19572860b6c1cdb2e248a6926b16837&start=10)

<https://www.cooking-hacks.com/forum/viewtopic.php?f=37&t=8803>