

2023-  
2024

# IoT platform for agriculture with Azure

PROJET IOT – 11.01.2024  
POMPILI VALÉRIE, ABDI VURAL

## Introduction

Pour ce deuxième projet, le but est de construire une plateforme IoT pour collecter et visualiser en temps réel les données issues de nos capteurs. Nous avons à disposition un dispositif Plug Sense Smart Agriculture Pro de Libelium qui permet de monitorer différents paramètres environnementaux, comme la température, la pression ou encore l'humidité. Ce dernier support plusieurs protocoles réseaux, dont LoRa. Nous avons ensuite une gateway pour récupérer nos paquets transmis par LoRa. Les différents objectifs sont les suivants :

1. Connecter et récolter les valeurs de nos trois capteurs.
2. Programmer le Plug Sense Smart Agriculture Pro afin d'envoyer les mesures à la Gateway LoRa.
3. Programmer la Gateway LoRa pour transférer les valeurs à TTN (The Things Network).
4. Utiliser Azure IoT Cloud pour visualiser les données collectées.

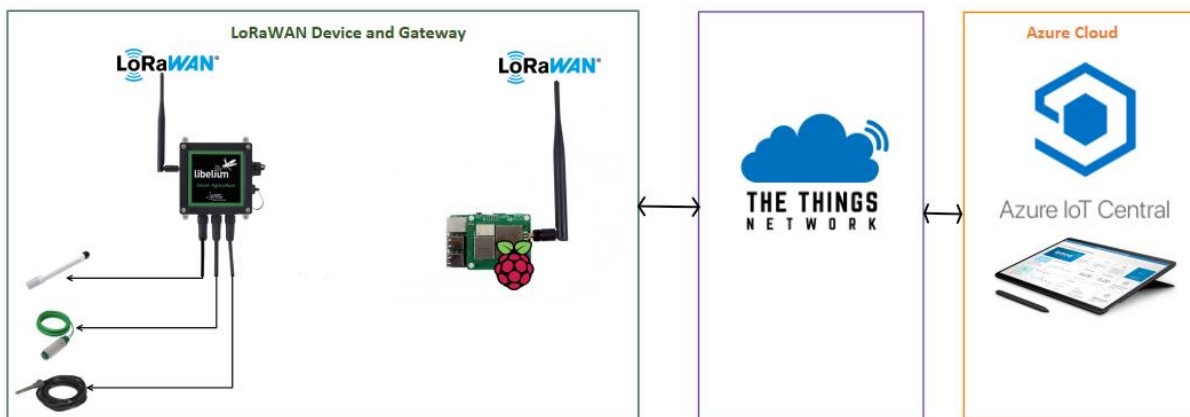


Figure 1: Architecture

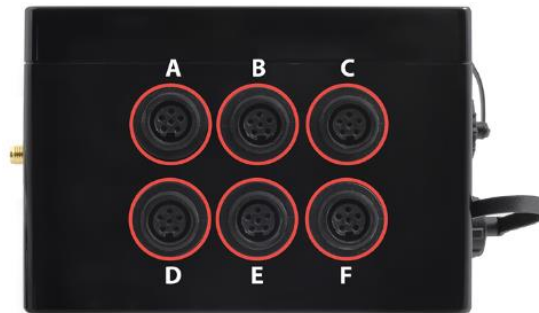
## Table des matières

Introduction.....	1
Plug Sense Smart Agriculture Pro.....	3
Sensors .....	3
Temperature, humidity and pressure sensor (BME280) .....	3
Soil moisture sensor Watermark by Irrrometer.....	3
Acquisition des données.....	4
Transmission LoRa depuis le Waspote .....	5
Librairie WaspLoRaWAN.....	5
Création de la frame .....	6
TinyFrame de la librairie WaspFrame .....	6
Solution Format CayenneLPP .....	7
Programmation Gateway.....	9
Remarque .....	10
Application TTN .....	11
Device et Application.....	11
Passerelle.....	11
Decoder CayenneLPP.....	12
Intégration Cloud.....	13
Azure IoT Hub ou Azure IoT Center .....	13
Azure IoT Hub .....	13
Azure IoT Central .....	13
Visualisation sur Azure IoT Central.....	14
Dashboard .....	15
Conclusion .....	15
Référence.....	16
Waspote .....	16
Configuration Gateway .....	16
Configuration TTN .....	16
Azure IoT.....	16

## Plug Sense Smart Agriculture Pro

### Sensors

Le système Plug Sense Smart Agriculture Pro est utilisé principalement pour l'agriculture de précision, les systèmes d'irrigation ou encore pour les serres. On peut connecter jusqu'à 15 capteurs. Voici les différents ports sur lesquels brancher les capteurs :



*Figure 2: Sensor probes*

Temperature, humidity and pressure sensor (BME280)



*Figure 3: Sensor BME280*

Ce capteur permet de mesurer 3 valeurs différentes :

1. Temperature entre -40 et 85°C
2. Humidité entre 0 et 100%
3. Pression entre 30 et 110kPa

On le branche sur le port F.

Soil moisture sensor Watermark by Irrrometer



*Figure 4: Soil moisture sensor*

Le capteur donne des mesures entre 50 et 10'000Hz et se branche sur le port B.

Soil temperature sensor Pt-1000



Figure 5: PT1000 sensor

Le capteur donne des valeurs entre -50 et 300°C et se branche sur le port D.

### Acquisition des données

Avant de créer notre paquet pour la transmission LoRa, il faut tout d'abord récupérer les valeurs de nos capteurs avec le code suivant :

```
#include <WaspSensorAgr_v30.h>

// Variable to store the read value
float temp, humd, pres;
float tempPT1000;
float watermark1;
//Instance object
pt1000Class pt1000Sensor;
watermarkClass wmSensor1(SOCKET_1);
void setup()
{
    // Turn on the USB and print a start message
    USB.ON();
    USB.println(F("Start program"));

    // Turn on the sensor board
    Agriculture.ON();
}

void loop()
{
    //////////////////////////////////////
    // 1. Read BME280: temp, hum, pressure
    //////////////////////////////////////
    watermark1 = wmSensor1.readWatermark();
    temp = Agriculture.getTemperature();
    humd = Agriculture.getHumidity();
    pres = Agriculture.getPressure();
    tempPT1000 = pt1000Sensor.readPT1000();
}
```

Figure 6: Acquisition des données

## Transmission LoRa depuis le Wasp mote

Pour connecter notre système et établir la connexion avec TTN, il faut établir une liaison sécurisée avec le réseau LoRa. Généralement, on privilégie l'OTAA car il est plus sécurisé qu'ABP. Lors de la première connexion, on utilisera OTAA afin d'obtenir des clés de session dynamiques, établir des identifiants uniques et synchroniser nos paramètres avec TTN. Une fois la connexion établie, on passera en mode ABP afin de se connecter plus rapidement et être moins gourmand en énergie.

### Librairie WaspLoRaWAN

Lors de l'enregistrement de notre device sur TTN, on peut donc choisir ou générer les paramètres suivants :

- JoinEUI/AppEUI
- DevEUI
- AppKey
- End device ID

On se connecte une première fois pour obtenir les clés pour la connexion sécurisée et ensuite on peut faire une boucle pour envoyer nos données :

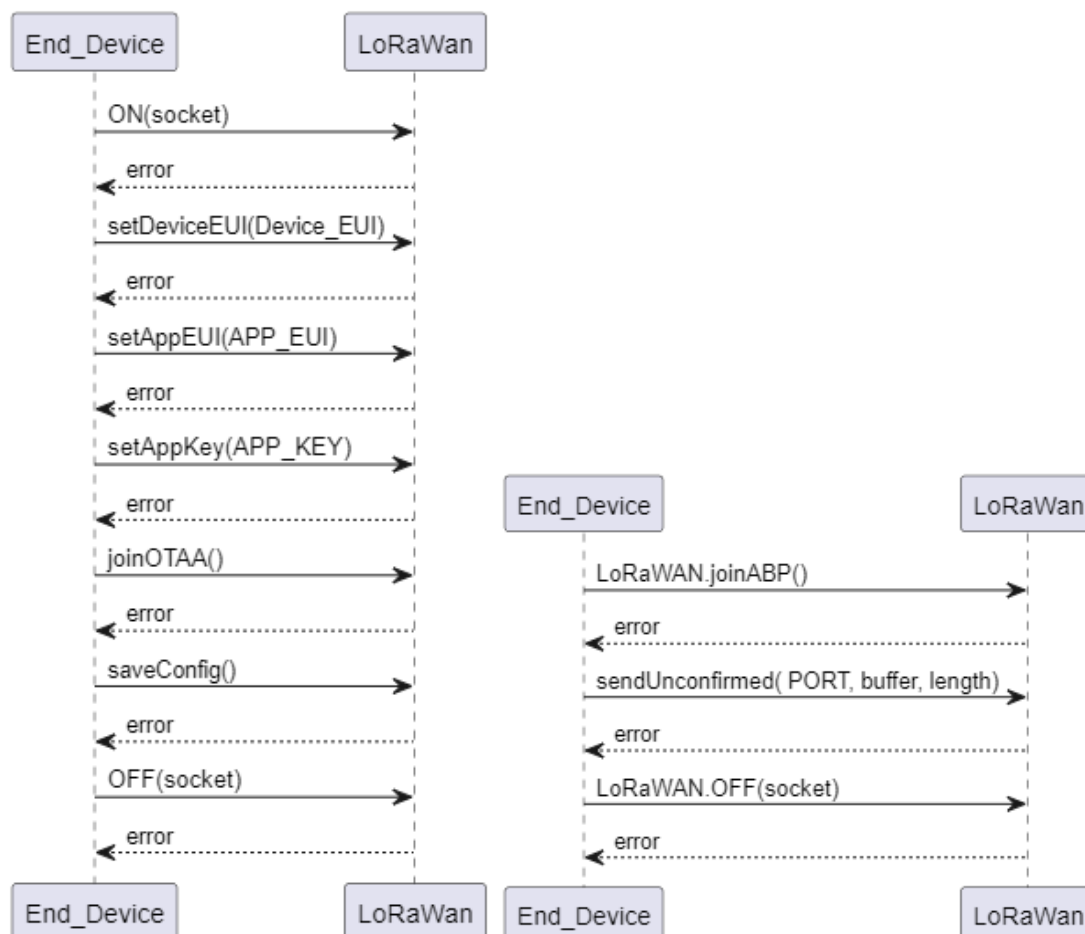


Figure 3 : init LoRa UML

Figure 4 : Loop LoRa UML

## Création de la frame

Libelium nous met à disposition plusieurs librairies pour l'envoi de frame en LoRa. Il suffit d'aller voir les exemples sur le site : <https://development.libelium.com/plug-and-sense/code-examples/communication/lorawan>

Nous avons testé différentes variantes pour l'envoi de nos paquets :

TinyFrame de la librairie WaspFrame

Le but de ce format est d'être le plus compact possible. Le header, composé de 2 bytes, contient le numéro de la séquence ainsi que la longueur de la frame. Pour le payload, on aura pour chaque capteur 1 byte d'identifiant (donné dans la [documentation](#)) ainsi que 4 bytes de valeur.

HEADER		PAYLOAD			
Sequence	Lenght	Sensor_1	Sensor_2	...	Sensor_n

Figure 7: Structure TinyFrame

On a donc les tags suivant pour chaque capteur :

- Watermark : TAG = SENSOR\_AGR\_SOIL1, ID = 150 (0x96), TYPE = float
- PT1000 : TAG = SENSOR\_AGR\_SOILTC, ID = 153 (0x99), TYPE = float
- Température : TAG = SENSOR\_AGR\_TC, ID = 74 (0x4A), TYPE = float
- Humidity : TAG = SENSOR\_AGR\_HUM, ID = 76 (0x4C), TYPE = float
- Pressure : TAG = SENSOR\_AGR\_PRES, ID = 77 (0x4D), TYPE = float

Malheureusement, les données envoyées contenues dans la frame ne font pas sens. On retrouve bien la structure de la frame ainsi que les identifiants de chaque capteurs mais le contenu ne correspond pas.

```

Temperature: 21.0000000000 Celsius
Humidity: 51.0000000000 %
Pressure: 30573.0000000000 Pa

PT1000: 21.000 Â°C
Watermark 1 - Frequency: 0.0000000000 Hz
1. Creating an BINARY frame
=====
Current BINARY Frame:
Length: 47
Frame Type: 6
frame (HEX): 3C3D3E062A726B1CE819623C746E6F64655F303123004A0000A8414C00004C424D00DAEE469600000000990000A841
frame (STR): <=>rkBb<tnode_01#J"ALLBMÚ1F-TM"A
=====
2. Switch ON OK
3. Join network OK
4.1. LoRaWAN maximum payload: 115
4.2. Tiny frame generated:001B4A0000A8414C00004C424D00DAEE469600000000990000A841
5. LoRaWAN confirmed sending...
```

Figure 8: Moniteur série (analyse paquet TinyFrame)

On retrouve en vert le header qui représente le premier paquet (0x00) et la taille de 27Bytes (0x1B) ainsi que les ID des capteurs en rouge. Mais les valeurs des 2 capteurs de températures (ici castés en int32\_t) ne correspondent pas à 21 degrés (0x15).

Après consultation de la documentation, il semblerait que les floats sont codifiés et la valeur ne s'obtient pas par simple conversion. Malheureusement aucune trace pour savoir comment décoder le payload et récupérer les valeurs.

- Simple Data:** The sensor field is composed by a unique data. The format of this field is: the first byte codifies the sensor type. Following the first byte and according to the sensor table, there is a number of bytes which correspond to the sensor value. For example, the temperature sensor is a float number, so it is a 4-byte field. Thus, the sensor field for 27°C will be set as follows:

ID (1 Byte)	Byte1	Byte2	Byte3	Byte4
SENSOR_TCA	0x00	0x00	0xD8	0x41

Figure 5: Binary simple sensor field

**Note:** Floats are codified so they are not a simple conversion.

Figure 9: Floats codifiés

### Solution Format CayenneLPP

L'avantage du format CayenneLPP, c'est qu'il peut être décodé automatiquement par TTN. Voici la structure de payload de CayenneLPP :

1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Latitude : 0.0001 ° Signed MSB
					Longitude : 0.0001 ° Signed MSB
					Altitude : 0.01 meter Signed MSB

Figure 10: Structure et identifiant capteur pour le format CayenneLPP



```

#define LPP_TEMPERATURE          103 //2 bytes, 0.1°C signed
#define LPP_RELATIVE_HUMIDITY    104 //1 byte, 0.5% unsigned
#define LPP_BAROMETRIC_PRESSURE  115 //2 bytes 0.1 hPA Unsigned
#define LPP_LUMINOSITY           101 // 2 bytes, here to stock moisture (KHz)

#define LPP_TEMPERATURE_SIZE     4 //2 bytes, 0.1°C signed
#define LPP_LUMINOSITY_SIZE      4 // Use here to stock moisture
#define LPP_RELATIVE_HUMIDITY_SIZE 3 //1 byte, 0.5% unsigned
#define LPP_BAROMETRIC_PRESSURE_SIZE 4 //2 bytes 0.1 hPA Unsigned
#define FRAME_SIZE               27

uint8_t bufferFrame[FRAME_SIZE];
uint8_t cursor;

```

Figure 11: Déclaration des taille et ID pour CayenneLPP

```

uint8_t addTemperature(uint8_t channel, int16_t val) {
    if ((cursor + LPP_TEMPERATURE_SIZE) > FRAME_SIZE) {
        return 0;
    }
    bufferFrame[cursor++] = channel;
    bufferFrame[cursor++] = LPP_TEMPERATURE;
    bufferFrame[cursor++] = val >> 8;
    bufferFrame[cursor++] = val;

    return cursor;
}

uint8_t addRelativeHumidity(uint8_t channel, uint16_t rh) {
    if ((cursor + LPP_RELATIVE_HUMIDITY_SIZE) > FRAME_SIZE) {
        return 0;
    }
    bufferFrame[cursor++] = channel;
    bufferFrame[cursor++] = LPP_RELATIVE_HUMIDITY;
    bufferFrame[cursor++] = rh;

    return cursor;
}

```

Figure 12: Exemple pour remplir le buffer

## Programmation Gateway

Tout d'abord il a fallu réinstaller une image propre car la Gateway semblait déjà programmée et nous n'avions pas les identifiants. Nous avons donc utilisé Pi Imager pour remettre une image LITE. Grâce à Pi Imager, on peut aussi configurer les identifiants pour la connexion, ainsi que les paramètres Wi-Fi.



Figure 13. Pi Imager

On installe alors SSH afin de pouvoir se connecter sur le PI en remote (IFCONFIG pour obtenir l'adresse IP du Pi) et on installe aussi Git.

```
bash
```

```
sudo systemctl enable ssh  
sudo systemctl start ssh
```

Copy code

Figure 14: Commande pour installer SSH

```
bash
```

```
sudo apt-get update  
sudo apt-get install git
```

Copy code

Figure 15: Installation des commandes Git

On peut suivre alors le tutoriel suivant pour configurer la Gateway :

<https://github.com/bigjohnson/iC880A-SPI-basics-station>

On peut voir ensuite que notre Gateway est bien connectée :

```
smartagr@raspberrypi:~$ systemctl status basicstation.service
● basicstation.service - Basic Sation TTN V3 service
   Loaded: loaded (/lib/systemd/system/basicstation.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2024-01-07 20:53:28 GMT; 36min ago
     Main PID: 382 (station)
       Tasks: 2 (limit: 1595)
          CPU: 34.319s
    CGroup: /system.slice/basicstation.service
            └─382 /opt/basicstation/bin/station -h /etc/basicstation

Jan 07 20:53:28 raspberrypi systemd[1]: Started Basic Sation TTN V3 service.
smartagr@raspberrypi:~$ systemctl status basicstation.service
● basicstation.service - Basic Sation TTN V3 service
   Loaded: loaded (/lib/systemd/system/basicstation.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2024-01-07 20:53:28 GMT; 36min ago
     Main PID: 382 (station)
       Tasks: 2 (limit: 1595)
          CPU: 34.319s
    CGroup: /system.slice/basicstation.service
            └─382 /opt/basicstation/bin/station -h /etc/basicstation

Jan 07 20:53:28 raspberrypi systemd[1]: Started Basic Sation TTN V3 service.
smartagr@raspberrypi:~$
```

Figure 16: Gateway LoRa connectée

#### Remarque

Malheureusement, sans comprendre pourquoi, la gateway ne cesse de se connecter/déconnecter. Malgré investigation nous n'avons pas trouvé la source du problème. Une seconde Gateway (personnelle), programmée selon le tutoriel suivant, a été utilisée :

[https://github.com/RAKWireless/rak\\_common\\_for\\_gateway](https://github.com/RAKWireless/rak_common_for_gateway) et

<https://www.thethingsnetwork.org/docs/gateways/rak7243c/configuring-gateway/>

## Application TTN

Nous avons fait le choix de ne pas détailler comment ajouter une application, un device, une gateway et comment faire l'intégration Cloud car TTN possède des marches à suivre très bien détaillés (Voir Configuration TTN).

## Device et Application

The screenshot shows the TTN web interface for an application named 'My\_Smart\_Agric\_Pro'. The left sidebar contains navigation links: Overview, End devices, Live data, Payload formatters, Integrations, Collaborators, API keys, and General settings. The main content area is divided into several sections:

- General information:**
  - End device ID: eui-70b3d57ed00640b1
  - Frequency plan: Europe 863-870 MHz (SF9 for RX2 - recommended)
  - LoRaWAN version: LoRaWAN Specification 1.0.2
  - Regional Parameters version: RP001 Regional Parameters 1.0.2 revision B
  - Created at: Jan 9, 2024 21:56:40
- Activation information:**
  - AppEUI: 01 02 03 04 05 06 77 88
  - DevEUI: 70 B3 D5 7E D0 06 40 B1
  - AppKey: .....
- Live data:** A list of recent messages with timestamps and details like DevAddr and payload.
- Location:** A map showing the device's location.

Figure 17: Smart Agriculture Application and device

## Passerelle

The screenshot shows the TTN web interface for a gateway named 'rak'. The left sidebar contains navigation links: Overview, End devices, Live data, Payload formatters, Integrations, Collaborators, API keys, and General settings. The main content area is divided into several sections:

- General information:**
  - Gateway ID: eui-b827ebffef88c49
  - Gateway EUI: B8 27 EB FF FE F8 8C 49
  - Gateway description: None
  - Created at: Jan 7, 2024 21:11:02
  - Last updated at: Jan 7, 2024 21:11:02
  - Gateway Server address: eu1.cloud.thethings.network
- LoRaWAN information:**
  - Frequency plan: EU\_863\_870\_TTN
  - Global configuration: Download global\_conf.json
- Live data:** A list of recent messages with timestamps and details like DevAddr, FCnt, and FPort.
- Location:** A map showing the gateway's location.

Figure 18: Gateway LoRa Personnelle

## Decoder CayenneLPP

Applications &gt; My\_Smart\_Agric\_Pro &gt; Payload formatters &gt; Uplink

## Default uplink payload formatter

## Setup

Formatter type \*

CayenneLPP

 What is CayenneLPP? [?](#)[Save changes](#)

Figure 19: Decoder de frame

Applications &gt; My\_Smart\_Agric\_Pro &gt; Application data








Time	Entity ID	Type	Data preview	Verbose stream 	 Export as JSON	 Pause	 Clear
↑ 17:18:37	eu1-70b3d57ed00640b1	Forward uplink data message	 Payload: { barometric_pressure_4: 964.1, luminosity_5: 0, relative_humidity_3: 51.5, temperature_1: :				
↑ 17:17:53	eu1-70b3d57ed00640b1	Forward uplink data message	DevAddr: 26 0B 5D 5E  Payload: { barometric_pressure_4: 964.1, luminosity_5: 0, relative_humidity_				
↑ 17:17:09	eu1-70b3d57ed00640b1	Forward uplink data message	DevAddr: 26 0B 5D 5E  Payload: { barometric_pressure_4: 964, luminosity_5: 0, relative_humidity_3:				

Figure 20: Live Data exemple

## Intégration Cloud

Il a tout d'abord fallu créer un compte Azure afin d'utiliser les différents services de Microsoft. Grâce à notre compte étudiant, nous pouvons créer un compte avec un crédit de 100\$ :

<https://azure.microsoft.com/en-us/free/students/>

Et on peut voir le crédit restant grâce au lien suivant :

<http://www.microsoftazureponsorships.com/>

## Azure IoT Hub ou Azure IoT Center

Afin de connecter et échanger les données entre notre device et notre Cloud Azure, on utilise TTN. TTN possède des marches à suivre très simples afin de s'occuper de l'intégration Cloud. En regardant la liste des plateforme IoT, on peut voir que TTN propose 2 solutions :

1. Azure IoT Central
2. Azure IoT Hub

Mais quelles sont les différences ?

### Azure IoT Hub

Azure IoT Hub permet d'avoir un contrôle total sur la gestion des données IoT et permet une meilleure personnalisation de notre application. On peut donc l'associer avec des services supplémentaires. Malheureusement cette flexibilité implique plus d'étapes de configuration et de développement, surtout pour la partie virtualisation et création de tableau de bord. Pour visualiser et analyser nos données, il faudrait utiliser d'autre service comme Azure Stream Analytics pour le traitement des données et un service de visualisation comme Grafana.

### Azure IoT Central

Azure IoT Central permet d'avoir une plateforme prête à l'emploi et possède déjà des outils de visualisation de données et de création de tableau sans nécessité d'autre service d'Azure. Le processus est donc plus rapide et simplifié pour les utilisateurs. Azure IoT Central possède des modèles préconfigurés et est donc plus facile à utiliser. En revanche, on a moins de contrôle sur la gestion des données et donc moins de flexibilité.

## Visualisation sur Azure IoT Central

Après la création de notre application IoT Central, un URL est généré pour avoir accès à notre application et à la visualisation. Il faut se rendre dans vue d'ensemble et on trouve l'URL de notre application :

<https://proj-smart-mse-central.azureiotcentral.com>

Si l'intégration a bien fonctionné, on devrait retrouver notre device enregistré sous appareils, ainsi que les données en temps réel. Ensuite de faire les étapes suivantes :

1. Créer un groupe d'appareil
2. Créer un model d'appareil
3. Créer une requête pour récupérer les données de nos paquets grâce à l'explorateur de données
4. Créer un nouveau tableau de bord et le relier à la requête créée précédemment. On peut ensuite ajouter des nouvelles vignettes (Widget) et composer notre tableau

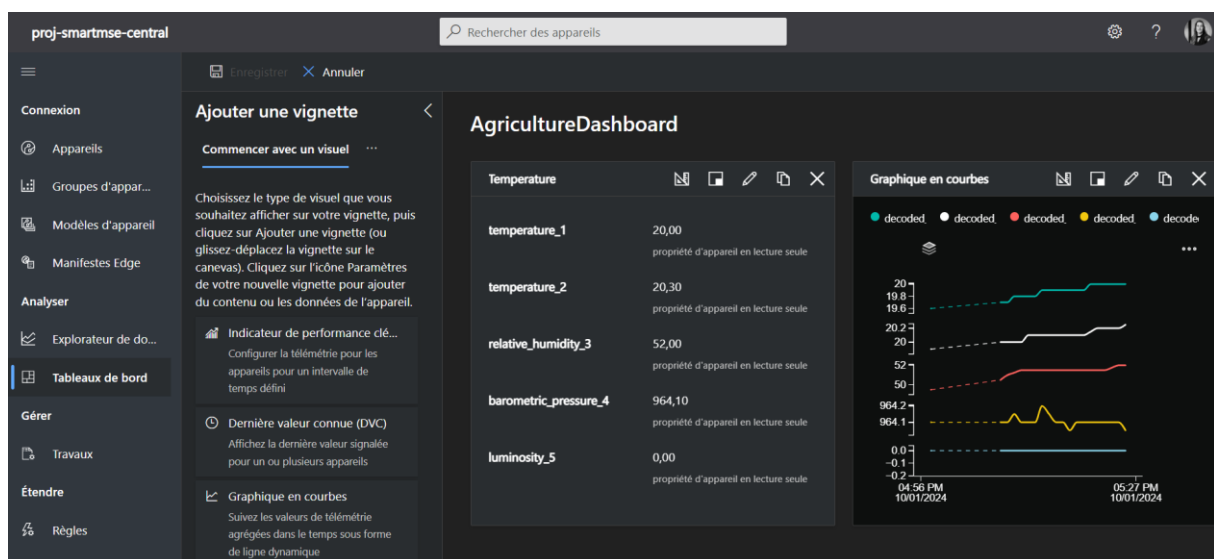


Figure 21: Création du Dashboard

## Dashboard

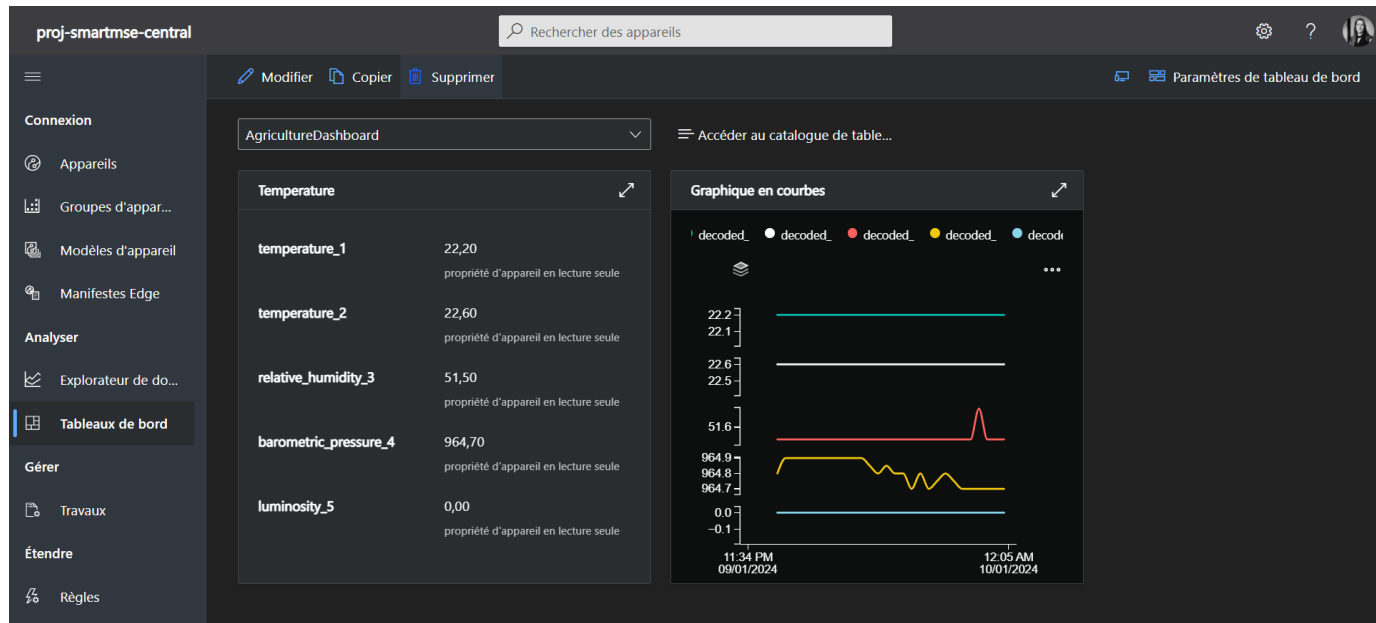


Figure 22: Dashboard final

## Conclusion

Pour résumé, nous avons pu acquérir créer notre plateforme IoT et répondre aux différents points demandés du projet. Nous avons pu acquérir les différents valeurs des capteurs de Libelium et afficher ces dernières sur un dashboard créé grâce au service Azure IoT Center. Nous avons pu envoyer programmer une Gateway LoRa pour l'envoi de nos différents paquets sous format CayenneLPP. Malheureusement nous avons perdu du temps avec les problèmes de connexion de la Gateway, ainsi que les frame codifiés de la librairie de Waspote. Cependant notre application est opérationnel et nous avons pu découvrir comment il était possible de créer rapidement une application ainsi qu'un dashboard grâce au service Azure IoT Center.

Une amélioration possible serait de créer un système d'alerte quand les températures sont trop élevées ou trop basse ou s'il y'a besoin d'allumer le système d'arrosage.



## Référence

### Waspote

Installation du software : <https://development.libelium.com/waspote-ide-v06/download-ide-windows>

Sensor Socket : <https://development.libelium.com/agriculture-sensor-guide/waspote-plug-amp-sense>

Acquisition des données : <https://development.libelium.com/ag-v30-01-temperature-sensor/>

Transmission en LoRa : <https://development.libelium.com/lorawan-11-join-otaa-send-frame/>

Création de frame : <https://development.libelium.com/ag-v30-11-frame-class-utility/>

Doc frame 1 : <https://development.libelium.com/data-frame-programming-guide/frame-structure>

Doc frame 2 : [https://www.libelium.com/wp-content/uploads/2013/02/data\\_frame\\_guide.pdf](https://www.libelium.com/wp-content/uploads/2013/02/data_frame_guide.pdf)

### Configuration Gateway

Configuration utilisée : <https://github.com/bigjohnson/IC880A-SPI-basics-station>

Deuxième solution (non utilisée): [https://github.com/RAKWireless/rak\\_common\\_for\\_gateway](https://github.com/RAKWireless/rak_common_for_gateway) et <https://www.thethingsnetwork.org/docs/gateways/rak7243c/configuring-gateway/>

### Configuration TTN

Connexion à TTN : <https://www.thethingsnetwork.org>

Cloud Integration : <https://www.thethingsindustries.com/docs/integrations/cloud-integrations/>

### Azure IoT

Création du compte Azure : <https://azure.microsoft.com/en-us/free/students/>

Figure 1: Architecture .....	1
Figure 2: Sensor probes .....	3
Figure 3: Sensor BME280.....	3
Figure 4: Soil moisture sensor .....	3
Figure 5: PT1000 sensor .....	4
Figure 6: Acquisition des données.....	4
Figure 7: Structure TinyFrame .....	6
Figure 8: Moniteur série (analyse paquet TinyFrame) .....	6
Figure 9: Floats codifiés.....	7
Figure 10: Structure et identifiant capteur pour le format CayenneLPP .....	7
Figure 11: Déclaration des taille et ID pour CayenneLPP .....	8
Figure 12: Exemple pour remplir le buffer .....	8
Figure 13. Pi Imager .....	9
Figure 14: Commande pour installer SSH.....	9
Figure 15: Installation des commandes Git .....	9
Figure 16: Gateway LoRa connectée .....	10
Figure 17: Smart Agriculture Application and device .....	11
Figure 18: Gateway LoRa Personnelle .....	11
Figure 19: Decoder de frame .....	12
Figure 20: Live Data exemple.....	12
Figure 21: Création du Dashboard.....	14
Figure 22: Dashboard final .....	15