



# **MISE EN PLACE D'UNE BADGEUSE CONNCTÉE**

Rapport de Stage de fin d'études  
(DIPLÔME UNIVERSITAIRE DE TECHNOLOGIE)

présenté par :

**Zihao GAO**

**Nom de la Société : IUT Nice Côte d'Azur Département G.E.I.I.**

Maître et tuteur de Stage : Alliou Diallo

Date : 19/06/2020

## Remerciements

Je tiens à remercier M. Diallo qui m'a confié un projet en télétravail durant cette période difficile au niveau des recherches de stages de 2020.

Je remercie également l'équipe pédagogique de la module ER4 de l'IUT parce que leurs travaux m'ont permis de programmer et tester sans carte microcontrôleur sur un simulateur WEB pour mettre en pratique les conceptions concernant ce stage.

J'adresse mes remerciements à Mme. Dubois et tout le personnel de l'IUT de Nice pour ses efforts à l'établissement de la convention et à la fin, je remercie M. Benvenuti, un de mes collègues de 2<sup>e</sup> année, et tous ce qui a partagé les informations pratiques pour la configuration de l'environnement de travail et d'autres questions concernant ce projet sur Internet. Je n'arriverais pas à progresser sans vos aides.

## Table des matières

Remerciements.....	2
Table des matières.....	3
Introduction .....	5
Présentation de l'entreprise -- « IUT de Nice Côte d'Azur ».....	6
Présentation du stage -- « MISE EN PLACE D'UNE BADGEUSE CONNCTEE » .....	7
Internet des objets.....	7
Cahier des charges .....	8
Matériel — Carte Microcontrôleur .....	9
Protocole de communication -- LoRaWAN .....	10
Mise en place de l'environnement de travail .....	12
THE THINGS NETWORK.....	12
Simulateur Mbed.....	14
Node-RED .....	14
InfluxDB .....	16
1 Envoi et réception des données sous format Cayenne.....	17
1.1 Cayenne Low Power Payload (LPP).....	17
1.2 Recodage des données hexadécimales sous format Cayenne .....	18
2 Simulation de l'envoi des données Température et Humidité .....	20
3 Réception des données TTN sur Node-RED et Importation dans InfluxDB .....	21
4 Première tentative à sortir une liste des absents .....	25
5 Deuxième tentative à sortir une liste des absents.....	26
6 Importation d'une liste d'étudiant dans InfluxDB .....	29
7 Finalisation de l'automatisation sur Node-RED.....	30
7.1 Résolution pour démarrer et arrêter la badgeuse : .....	30
7.2 Automatisation pour sortir un fichier excel de la liste des absents : .....	32
7.3 Contrôle exhaustif en boucle fermée .....	33

8 Badgeuse à machine à état.....	38
Conclusion.....	39
Sources et documentations .....	41
Table des illustrations .....	42

## Introduction

Se déroule du 18/05/2020 jusqu'au 19/06/2020, le stage « mise en place d'une badgeuse connectée » a été réalisé par moi au domicile sur mon ordinateur portable tout en télétravail avec M. Diallo en tant que le maître de stage.

Ce stage est un projet de la réalisation d'une badgeuse connectée dans le but de compter les étudiants absents d'une séance d'amphithéâtre automatiquement et sortir une liste des absents tout au long d'un semestre avec la date des séances et les noms d'absents. Cette badgeuse pourrait faciliter le travail du secrétariat et épargner le temps du maître de l'amphithéâtre pour la signature de la feuille de présence.

Ce stage est établi à la base du projet « “Kiffy” véhicule urbain Tricycle en libre-service ‘Smart City’ » du module ER4 dans le cadre de l'enseignement du département GEII de l'IUT de Nice Côte d'Azur. Il est au bénéfice de l'IUT de Nice Côte d'Azur.

## Présentation de l'entreprise -- « IUT de Nice Côte d'Azur »

L'IUT à Nice a été ouvert le 5 octobre 1970. Elle est une des composantes les plus importantes de l'Université de Nice Sophia Antipolis, par son budget et ses effectifs. Ses 10 départements d'enseignement répartis sur 5 sites dans un rayon de 70 kilomètres (Cannes-la-Bocca, Cannes, Sophia Antipolis, Nice, Menton) délivrent aujourd'hui plus de 30 diplômes nationaux de niveau Bac + 2 : DUT, et Bac + 3 : Licences Professionnelles.



*figure 1 Logo de l'IUT de Nice Côte d'Azur*



*figure 2 Vue de dessus du département GEII*

Le département G.E.I.I est une partie intégrante de l'IUT Nice Côte d'Azur. Il offre une formation de haut niveau dans un large domaine : électronique, télécommunications, production et transformation d'énergie, automatismes, robotique, informatique, systèmes numériques, réseaux industriels, systèmes temps réel, etc.

## Présentation du stage -- « MISE EN PLACE D'UNE BADGEUSE CONNECTEE »

La mise en place d'une badgeuse connectée concerne un développement d'une application sous IoT( Internet des objets ).

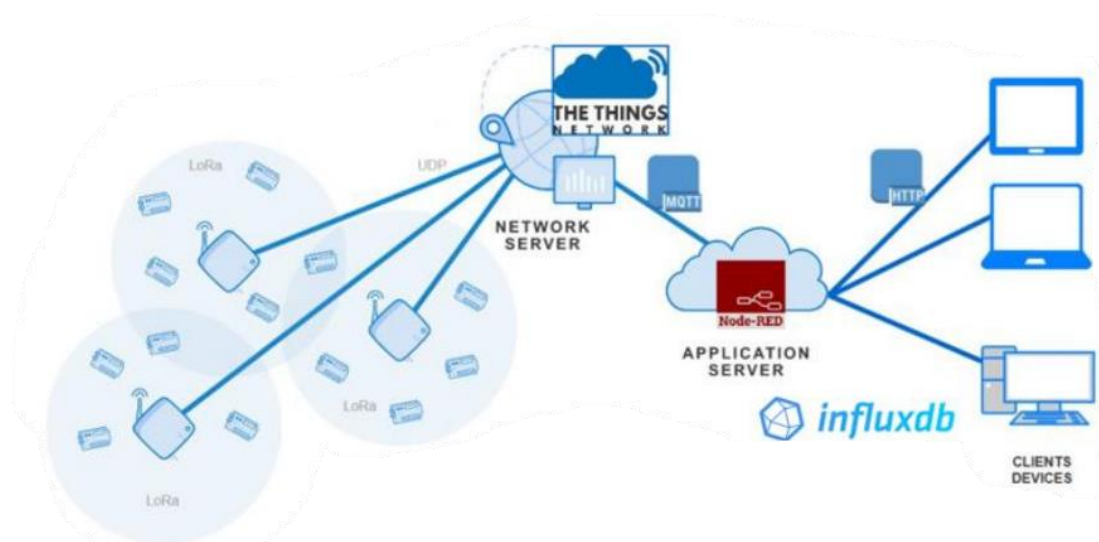


figure 1 Schéma du réseau des objets connectés

### Internet des objets

L'Internet des objets ou IdO (en anglais (the) Internet of Things ou IoT) est l'interconnexion entre l'Internet et des objets, des lieux et des environnements physiques. L'appellation désigne un nombre croissant d'objets connectés à l'Internet permettant ainsi une communication entre nos biens dits physiques et leurs existences numériques. Ces formes de connexions permettent de rassembler de nouvelles masses de données sur le réseau et donc, de nouvelles connaissances et formes de savoirs.

## Cahier des charges

La badgeuse sera connectée au réseau IoT via un Gateway. Elle lit et envoie des données de RFID tag de la carte possédée par chaque étudiant sur le réseau IoT.

Lorsque la première lecture, l'étudiant sera demandé de saisir son nom via un interface homme-machine de la badgeuse pour créer la première liste avec les noms et les tags RFID associés.

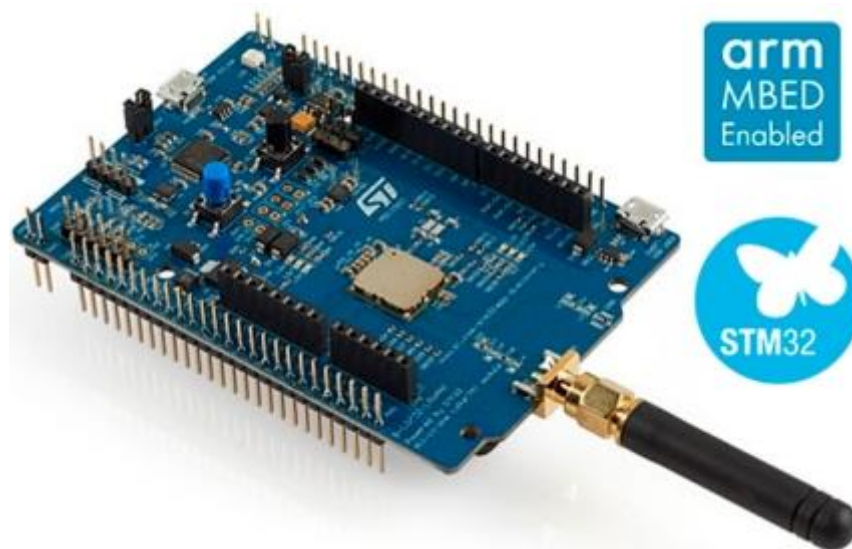
À partir de cette liste, une application sur un serveur aussi connecté au réseau IoT réalisera l'automatisation de génération des listes d'absents avec la date et les noms d'étudiants.

À la fin, les listes d'absents seront récupérées par un appareil de client connecté, cela peut être un PC par exemple.

Fonctionnements détaillés :

- 1- Création de la liste des étudiants lors du 1er badge
- 2- Si un étudiant badge lors du 2nd cours, il sera rajouté dans cette liste, mais rajouté dans la liste des absents du 1er cours
- 3- Option 1 : Led verte qui donne l'autorisation de badger, puis led rouge qui clignote jusqu'à envoi et accusé de réception avec un downlink, et led verte rallumer pour un autre badge... Et fermeture des badges 15mn après le 1er badge avec led rouge qui s'allume pour dire fait d'appel et interdiction de badger
- 3- Option 2 : On badge sans que le système n'envoie et led verte allumée après chaque lecture puis Envoie des données 15mn après le 1er badge et fermeture des badges avec led rouge allumée, mais attention à la quantité de données envoyée
- 4- Création de la liste des absents (On pourra rajouter entête avec date, heure) Si c'est un étudiant qui était dans le cas 2, on mettra à côté la date de la séance précédente





*figure 2 B-L072Z-LRWAN1 Discovery kit*

Le module « B-L072Z-LRWAN1 LoRa®/Sigfox™ Discovery kit » ([www.st.com/en/evaluation-tools/b-l072z-lrwan1.html](http://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html)) sera utilisé comme le module programmable de la badgeuse.

Le module est alimenté par un microcontrôleur STM32L072CZ et un émetteur-récepteur SX1276. L'émetteur-récepteur est équipé du modem LoRa® longue portée, offrant une communication à spectre étendu ultra longue portée et une immunité aux interférences élevée, minimisant la consommation de courant.

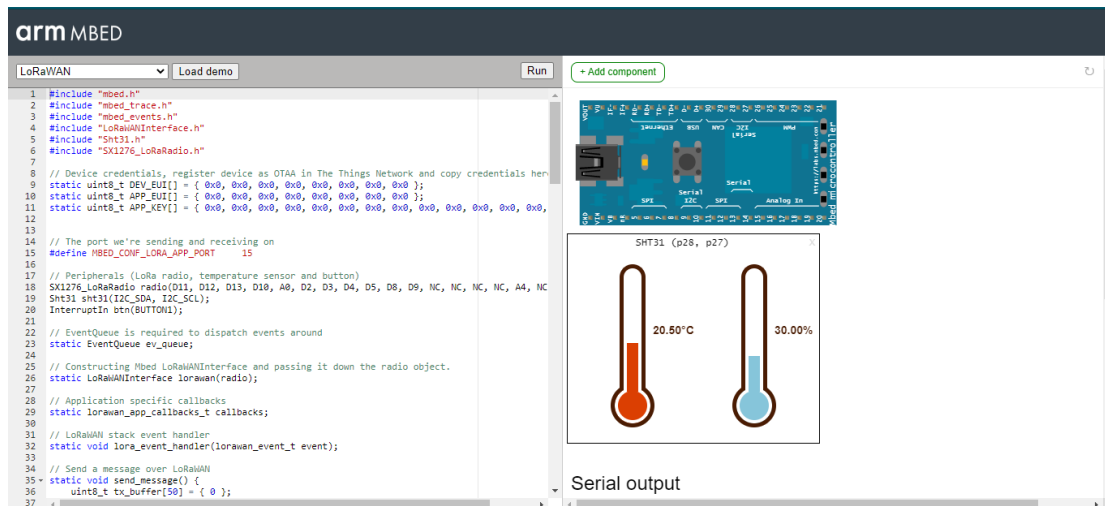


figure 3 Interface du simulateur en ligne

Dans la réalité, je n'ai pas accès au matériel de ce type, un simulateur en ligne ( <https://simulator.mbed.com/#lorawan> ) a été utilisé pour continuer le projet.

## Protocole de communication -- LoRaWAN

LoRaWAN est une spécification de protocole basée sur la technologie LoRa développée par LoRa Alliance. LoRaWAN cible les besoins de base de l'utilisation de LoRa pour l'IoT en fournissant l'adressage, le routage et la sécurité.

La topologie d'un réseau LoRaWAN comprend plusieurs éléments.

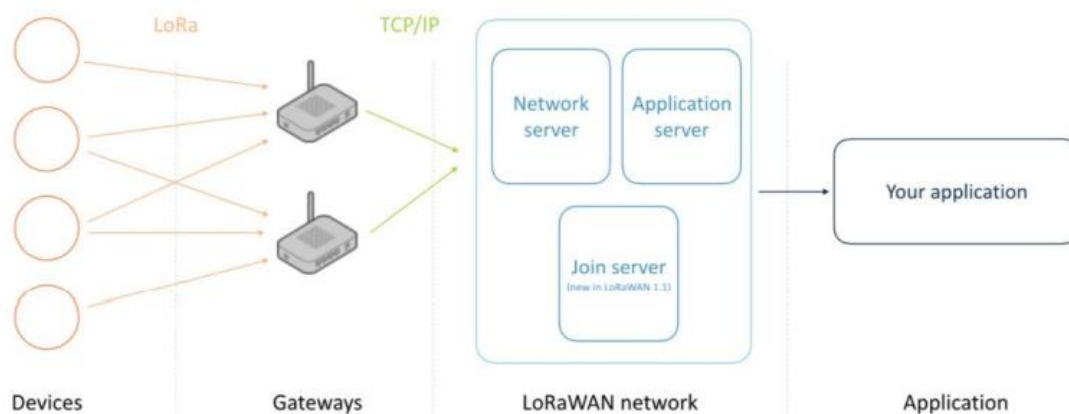


figure 4 Topologie d'un réseau LoRaWAN

**Le Device ou Nœud final :** Les nœuds finaux sont des éléments tels que des capteurs, généralement situés à distance et qui communiquent avec les Gateways en utilisant la modulation LoRa.

**La Gateway ou Concentrateur / passerelle :** les passerelles sont des points d'accès pour les nœuds finaux qui agrègent des données et les communiquent à un serveur de réseau central via des connexions IP standard.

**Serveur de réseau :** Le serveur de réseau LoRa agit pour éliminer les paquets en double, gère la sécurité et les débits de données. Exemple : The Things network est un serveur réseau Open source donc gratuit, contrairement à Bouygues ou Orange qui nécessitent un abonnement

**Serveurs d'applications :** les serveurs d'applications gèrent la sécurité de la charge utile et effectuent des analyses pour utiliser les données du capteur. Exemple : Cayenne fonctionne comme un serveur d'application interne car elle permet de décoder un payload (trame), alors que Node-red est un serveur d'application externe permettant de récupérer les données arrivant au serveur réseau et de les stocker dans une base de données.

**Serveurs d'activations :** Un nœud final doit être activé avant de pouvoir communiquer avec le serveur de réseau. Il existe 2 méthodes pour activer un composant dans un serveur LoRaWAN :

la méthode **OTAA** (Over-The-Air-Activation) et la **ABP** (Activation-By-Personalisation).

**OTAA** est la méthode d'activation préférée car elle fournit le moyen le plus sûr de connecter des périphériques finaux à un serveur réseau. Ici les clés de chiffrement sont obtenues par un échange avec le réseau, contrairement à l'**ABP** pour laquelle les clés de chiffrement sont stockées dans les équipements.

Avant l'activation, l'objet connecté doit connaître et stocker (dans un fichier .json) ses **DevEUI** (identifiant du composant), **AppEUI** (identifiant de l'application ; une application peut contenir plusieurs devices) et **AppKey** (clé de sécurité de l'application) fournis par le serveur réseau lors de l'enregistrement du composant. Ces identifiants seront utilisés par le serveur pour identifier l'objet lors de la

communication.



figure 5 Les 2 types d'activation OTAA et ABP

## Mise en place de l'environnement de travail

### THE THINGS NETWORK

The Things network ( <https://thethingsnetwork.org/> ) est un serveur réseau Open source donc gratuit, il agit pour éliminer les paquets en double, gère la sécurité et les débits de données.

À fin de l'utiliser, il est nécessaire de créer un compte sur le site TTN.

Une fois le compte est créé, dans l'onglet « console », j'ajoute une nouvelle application qui s'appelle « stage01 ». Les clé d'application et les clés d'appareil seront générées automatiquement.

## DEVICE OVERVIEW

**Application ID** stage01

**Device ID** badges

**Activation Method** OTAA

**Device EUI** <> ⇄ 00 F0 0E 60 C4 0F AB 24 [📄]

**Application EUI** <> ⇄ 70 B3 D5 7E D0 02 F3 16 [📄]

**App Key** <> ⇄ 🔒 BF 37 64 89 2D 14 C9 F4 40 4B 1D A2 1B E7 3B 7A [📄]

**Device Address** <> ⇄ 26 01 22 BC [📄]

**Network Session Key** <> ⇄ 🔒 51 2E EF 16 90 F1 A9 63 0B 44 37 A3 A7 FC 15 DE [📄]

**App Session Key** <> ⇄ 🔒 4C 8D DA 83 9E 2B 5D 41 78 7E 57 D9 EA 1A 5C EE [📄]

figure 6 Paramètres de l'application et l'appareil sur the things network

THE THINGS NETWORK CONSOLE COMMUNITY EDITION Applications Gateways Support kinue00

Applications > Add Application

### ADD APPLICATION

**Application ID**  
The unique identifier of your application on the network  
stage01 [✓]

**Description**  
A human readable description of your new app  
Eg. My sensor network application [✓]

**Application EUI**  
An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page.  
EUI issued by The Things Network

**Handler registration**  
Select the handler you want to register this application to  
ttn-handler-eu [✓]

figure 7 l'ajout d'une nouvelle application TTN

Une fois l'application est créée, dans l'onglet « Collaborators » de l'application j'ajoute le compte du maître de stage pour qu'il en ait accès.

## Simulateur Mbed

Dans la demo « LoRaWAN » utilisé pour ce stage je modifie les premières lignes concernant la configuration des clés OTAA en utilisant les clés générées par le serveur de réseau TTN qui a été présenté dans la partie précédente.

```
8 // Device credentials, register device as OTAA in The Things Network
  and copy credentials here
9 static uint8_t DEV_EUI[] = { 0x00, 0xF0, 0x0E, 0x60, 0xC4, 0x0F,
  0xAB, 0x24 };
10 static uint8_t APP_EUI[] = { 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x02, 0xF3,
  0x16 };
11 static uint8_t APP_KEY[] = { 0xBF, 0x37, 0x64, 0x89, 0x2D, 0x14, 0xC9,
  0xF4, 0x40, 0x4B, 0x1D, 0xA2, 0x1B, 0xE7, 0x3B, 0x7A };
```

*figure 8 Configuration des clés de l'appareil*

Maintenant cet appareil est reconnu par le serveur de réseau. Je peux constater les données envoyés par cet appareil sur le serveur de réseau.

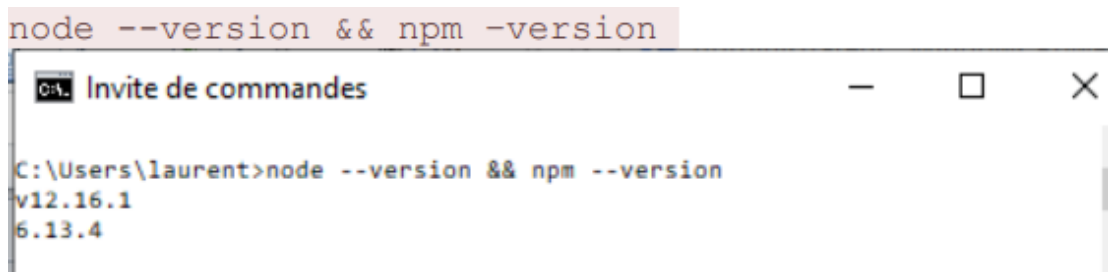
## Node-RED

Node-RED est un outil de développement basé sur les flux pour la programmation visuelle développé à l'origine par IBM pour connecter des périphériques matériels, des API et des services en ligne dans le cadre de l'Internet des objets.

Procédure d'installation sous Windows :

- Installer node.js ( <https://nodejs.org/dist/v12.16.1/node-v12.16.1-x64.msi> ) Une fois tout installé, Vérifier ensuite dans une fenêtre de commande CMD que node.js est bien installé en tapant la commande

```
node --version && npm -version
```



```
C:\Users\laurent>node --version && npm --version
v12.16.1
6.13.4
```

figure 9 Vérification de la version node.js et npm en CMD

- Installer NODE-RED : Fenêtre CMD : taper la commande  
npm install -g --unsafe-perm node-red
- Lancement du serveur local nodered : Depuis une invite de  
commande lancer « node-red »
- Lancement de nodered : depuis un navigateur web lancer  
l'URL : <http://127.0.0.1:1880/>

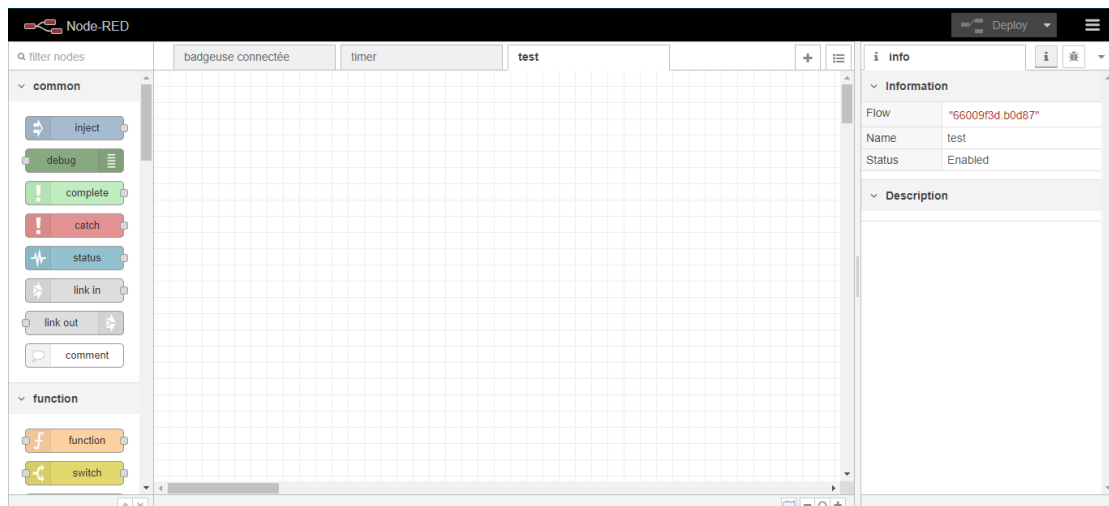


figure 10 Interface Node-RED

## Tutoriel pour installer les nœuds TTN sur Node-RED

( <https://www.youtube.com/watch?v=RpzmUqRq6x0&feature=youtu.be> )

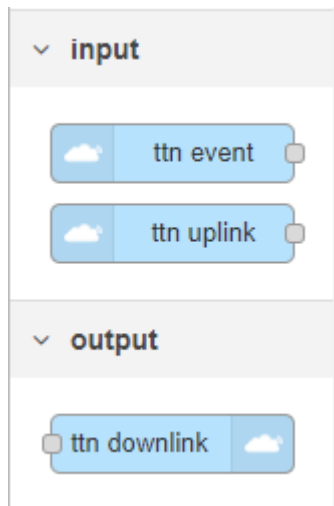


figure 11 les noeuds TTN

Comme résultat, on obtient des nœuds de type TTN sur le gauche de l'interface de Node-RED.

Le « ttn uplink » permet de recevoir un message de l'appareil connecté et le « ttn downlink » permet d'envoyer un message à l'appareil.

Le « ttn event » permet de détecter un événement de l'appareil et déclencher un suite d'actions.

## InfluxDB

InfluxDB est un système de gestion de base de données orientée séries chronologiques hautes performances, écrit avec le langage de programmation Go et distribué sous licence MIT.

Il sera utilisé pour stocker les données de tags RFID et les noms d'étudiant envoyés par la badgeuse avec une date précise.

- Télécharger :  
( [https://dl.influxdata.com/influxdb/releases/influxdb-1.7.10\\_windows\\_amd64.zip](https://dl.influxdata.com/influxdb/releases/influxdb-1.7.10_windows_amd64.zip) ) Dézipper l'archive directement à la racine de c:\
- Lancement du serveur de data base : dans le dossier crée rechercher et lancer influxd.exe
- Lancement de la console de commande influx db : dans le dossier crée rechercher et lancer influx.exe
- Quelques commandes à connaître sous influx db : Depuis la console :
  - show data base -> permet de visualiser les data bases enregistrées
  - Use nom\_de\_votre\_database -> permet de définir la data base par défaut



- Show séries : permet d'afficher les tables de données de la data base par défaut
- Select \* from nom\_de\_la\_serie -> permet d'afficher le contenu de la data base
- Drop le\_nom\_de\_la\_serie → permet d'effacer une série
- Insert nom\_de\_la\_serie, champ1=val1 champ2=val2 → permet d'insérer manuellement un donnée dans la série

## 1 Envoi et réception des données sous format Cayenne

### 1.1 Cayenne Low Power Payload (LPP)

Cayenne (LPP) de Cayenne offre un moyen pratique et simple d'envoyer des données sur des réseaux LPWAN tels que LoRaWAN.

De plus, Cayenne LPP permet à l'appareil d'envoyer différentes données de capteur dans différentes trames. Pour ce faire, chaque donnée de capteur doit être précédée de deux octets :

- Canal de données : identifie de manière unique chaque capteur de l'appareil sur plusieurs images.
- Type de données : identifie le type de données dans la trame, par exemple. "Temperature".

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

*figure 12 Payload Structure*

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 13 Tableau des types Cayannes

## 1.2 Recodage des données hexadécimales sous format Cayanne

```

44  uint8_t tx_buffer[7]={0x01,0x067,
    (uint16_t(sht31.readTemperature()*10))>>8,
    uint16_t(sht31.readTemperature()*10),0x02,0x68,
    uint16_t(sht31.readHumidity()*2)};
45
46  uint16_t packet_len=sizeof(tx_buffer);
47
48  printf("Sending %u bytes: \"%u\" \n", packet_len,
    tx_buffer);
49
50  int16_t retcode = lorawan.send(MBED_CONF_LORA_APP_PORT,
    tx_buffer, packet_len, MSG_UNCONFIRMED_FLAG);

```

figure 14 Envoi d'une trame de température et humidité en cayanne

Un buffer ( array en C ) est créé pour regrouper les données à envoyer en Octet ( Byte, groupe de 8 bits ).

Prenons par exemple une trame de donnée du type Température. Selon le format Cayenne, ce trame ( payload ) serait composée de 4 octets au total dont le premier pour la chaîne utilisé, le deuxième pour son type et les 2 octets qui restent pour la valeur de température comme la présentation ci-dessous :

{ 0x01, 0x67, XXXX XXXX XXXX XXXX } ( X pour la valeur de température en binaire )

Pour envoyer correctement la valeur de température qui prend 2 octets, un variable de format uint16\_t qui peut présenter un entier non-signé d'un word ( 16 bits ) est créé pour le stocker.

La température est de précision 0.1 et une multiplication de 10 est nécessaire pour que la trame soit un entier en octet ( l'encodage est fait ici et le décodage sera fait extérieur ).

Une méthode de masquage et décalage ( cf. « 3. Use words », <https://www.thethingsnetwork.org/docs/devices/bytes.html>)

```
46 uint8_t tx_buffer[4] = { 0 };
47 uint16_t temp = sht31.readTemperature()*10;
48 tx_buffer[0] = 0x01;
49 tx_buffer[1] = 0x67;
50 tx_buffer[2] = (temp & 0xFF00) >> 8;
51 tx_buffer[3] = (temp & 0x00FF);
```

figure 15 Création d'un buffer pour un data température

Sous la même idée, il existe une autre version admise par le syntaxe C pour atteindre la même fonction.

```
50 uint8_t
tx_buffer[4]={0x01,0x67,(uint16_t(sht31.readTemperature()*10))>>8,uint16_t(sht31.readTemperature()*10)}; // another variation
```

figure 16 une autre variation pour la méthode masquage et décalage

## 2 Simulation de l'envoi des données Température et Humidité

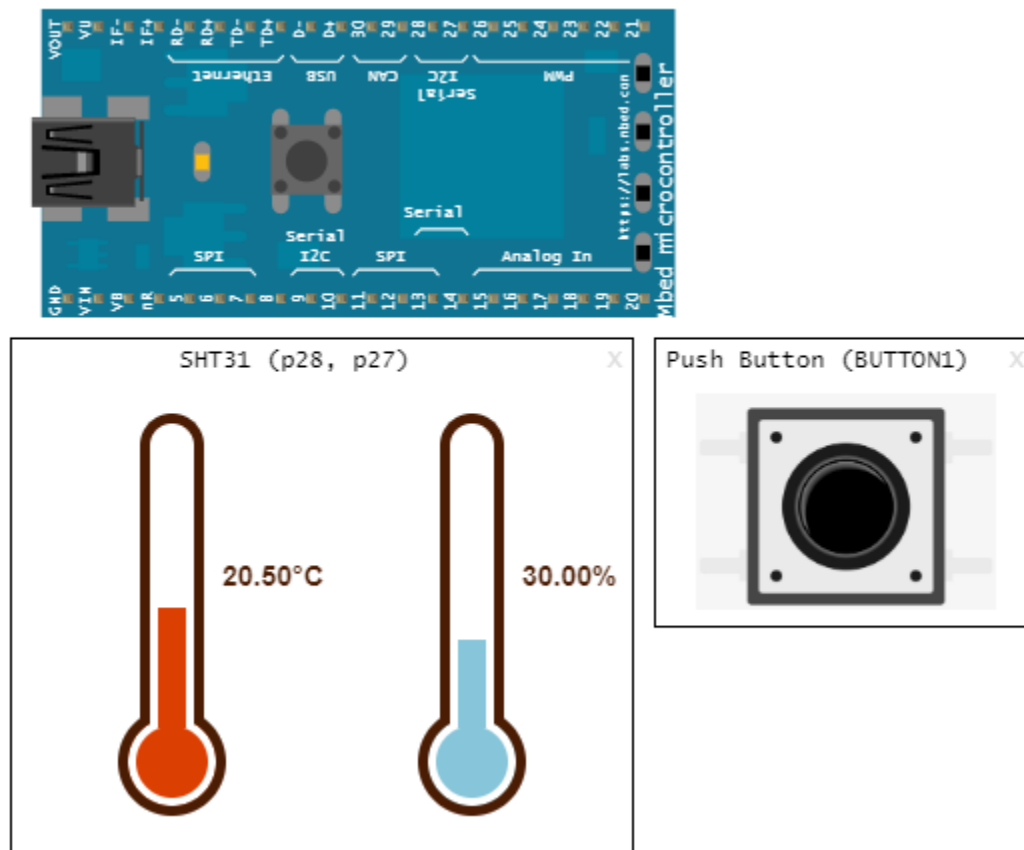


figure 17 mbed LPC1768 et sht31 simulés

Avec la démo et la modification que l'on vient de faire, j'essaie d'envoyer la température et l'humidité via un bouton qui autorise l'envoi.

Press **BUTTON1** to send the current value of the temperature sensor!

[DBG ][LSTK]: Initializing MAC layer

[DBG ][LSTK]: Initiating OTAA

[DBG ][LSTK]: Sending Join Request ...

[DBG ][LMAC]: Frame prepared to send at port 0

[DBG ][LMAC]: TX: Channel=2, TX DR=5, RX1 DR=5

[DBG ][LRAD]: transmit channel=868500000 power=13  
bandwidth=7 datarate=7

**Connection - In Progress ...**

[DBG ][LSTK]: Transmission completed

[DBG ][LMAC]: RX1 slot open, Freq = 868100000

```

[DBG ][LRAD]: ][LMAC]: RX1 slot open, Freq = 868100000
[DBG ][LMAC]: RX2 slot open, Freq = 869525000
[DBG ][LSTK]: OTAA Connection OK!
Connection - Successful
Sending 7 bytes: "32260"
[INFO][LMAC]: RTS = 7 bytes, PEND = 0, Port: 15
[DBG ][LMAC]: Frame prepared to send at port 15
[DBG ][LMAC]: TX: Channel=6, TX DR=5, RX1 DR=5
[DBG ][LRAD]: transmit channel=867700000 power=13
bandwidth=7 datarate=7
7 bytes scheduled for transmission
[DBG ][LSTK]: Transmission completed
[DBG ][LMAC]: RX1 slot open, Freq = 868500000
[DBG ][LMAC]: RX2 slot open, Freq = 869525000
Message Sent to Network Server

```

*figure 18 Serial output pour l'envoi de température et humidité*

Sur le terminal simulé, on constate que l'envoi a bien réussi et sur le console TTN on peut récupérer le payload originel et son interprétation sous format Cayenne, ce qui donne bien les données détectés par le capteur simulé.

time	counter	port				
▲ 18:06:04	0	15	retry	dev id: <a href="#">badges</a>	payload: 01 67 00 CD 02 68 3C	relative_humidity_2: 30 temperature_1: 20.5

*figure 19 Application data reçu ( température et humidité )*

### 3 Réception des données TTN sur Node-RED et Importation dans InfluxDB

Afin de réaliser l'automatisation sur Node-RED, on doit d'abord recevoir des données de l'appareil sur Node-RED. Pour ce faire, j'ai mis en place le nœud `ttn_uplink` qui permet de recevoir des données depuis le serveur TTN.

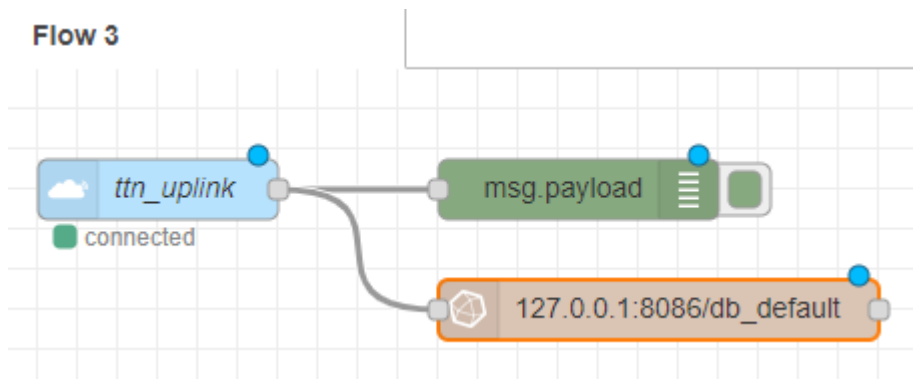


figure 20 Synoptique avec ttn\_uplink

Comme la figure ci-dessus, j'ai ajouté deux sorties suivant l'entrée ttn\_uplink, l'une pour déboguer qui affichera sur la fenêtre de débogage les données exactes qui sort de ttn\_uplink et l'autre--« influxdb out » permet de importer et stocker les donnée dans une bases de données influxdb.

Les configurations concernant ces 2 nœuds ci-dessous.

The screenshot shows the "Edit ttn uplink node" configuration window. At the top, there are three buttons: "Delete", "Cancel", and "Done". Below these is a "Properties" section with a gear icon. The properties are as follows:

Property	Value
Name	ttn_uplink
App	stage01
Device ID	badges
Field	

figure 21 Configuration ttn uplink node

Edit ttn uplink node > **Edit ttn app node**

**Properties**

stage01

.....

discovery.thethingsnetwork.org:1900

figure 22 Configuration ttn app node

La configuration de ttn\_uplink se fait référence aux informations générées du serveur TTN vues dans la mise en place de l'environnement de travail « THE THINGS NETWORK ».

**Edit influxdb out node**

**Properties**

127.0.0.1:8086/db\_default

temp\_humid

☐ Advanced Query Options

Name

figure 23 Configuration influxdb out

*figure 24 Configuration influxdb node*

Avant de configurer le nœud influxdb sur Node-RED, j'ai créé une base de données appelé « db\_default » en utilisant le shell de InfluxDB.

Il faut faire attention que le serveur InfluxDB soit en marche lorsque l'envoi de l'appareil.

```
Connected to http://localhost:8086 version 1.7.10
InfluxDB shell version: 1.7.10
> show databases
name: databases
name
----
_internal
db_default
> use db_default
Using database db_default
> show series
key
---
```



```

mesures
temp&humid
temp_humid
> select*from temp_humid
name: temp_humid
time                relative_humidity_2  temperature_1
-----
1589971714104871000  30                20.5
>

```

figure 25 InfluxDB shell

On peut retrouver les données concernant l'humidité et la température dans la base de données après un envoi.

```

20/05/2020 à 12:48:34  node: 2757f82f.f4d848
msg.payload : Object
  ▶ { relative_humidity_2: 30,
    temperature_1: 20.5 }

```

figure 26 Debug Node-red

La sortie débogage donne la même chose en vérifiant que l'on a bien importé les données de l'appareil dans la base de données décidée.

## 4 Première tentative à sortir une liste des absents

Pour extraire les noms des absents automatiquement j'ai réalisé un synoptique comme ci-dessous.

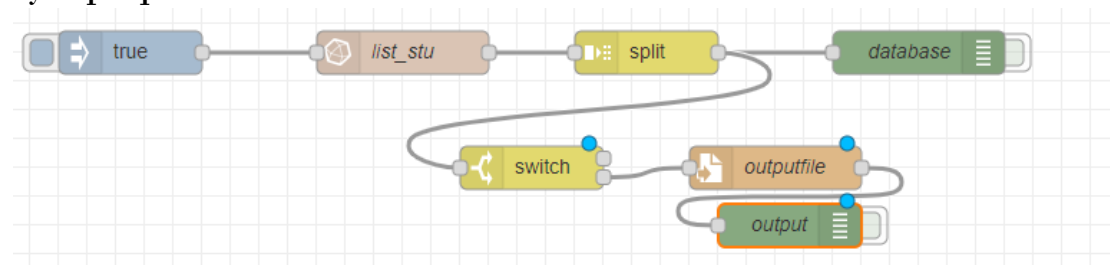


figure 27 1er synoptique

À l'entrée, j'ai ajouté un nœud d'événement vrai pour faire exporter manuellement les données dans la base de données « list\_stu ». j'ai ajouté quelques noms et ses RFID dans « list\_stu » avant avec le shell.

Avec l'application d'un node « split » à la sortie du query infludb j'ai pu traiter séparément les informations d'étudiant ( RFID et name ).

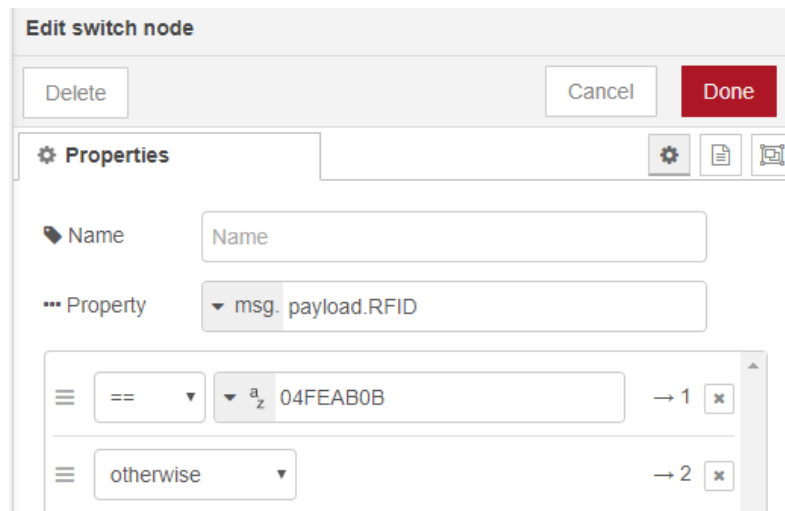


figure 28 Configuration du switch

En suite comparaison des RFID ( pour l'instant, une variable entrée manuellement ) et récupération de la liste des étudiants qui n'est pas de la même RFID.

Cependant il reste quelques problèmes :

- Comparaison d'un seul RFID à la fois.
- Conversion du type de donnée : ttn\_uplink envoie un array de 4 bytes, RFID enregistré dans influxdb de char.

Afin de les résoudre, j'ai lancé la deuxième tentative.

## 5 Deuxième tentative à sortir une liste des absents

Objectifs atteints :

- Réception d'un seul RFID de l'appareil ttn
- Comparaison de ce RFID avec la liste du serveur Influxdb
- Extraction des noms des absents ( pas du même RFID ) et enregistrement dans un fichier txt

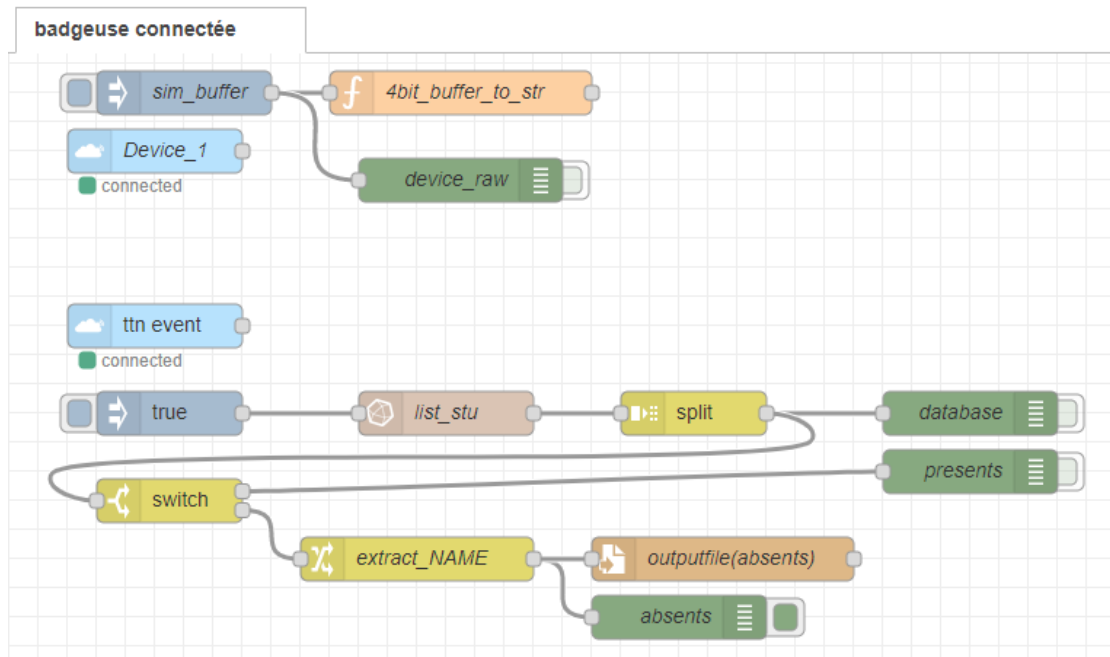


figure 29 2er synoptique

Problème réglé : conversion des données du ttn uplink ( buffer de 4bits à string ).

Solution :

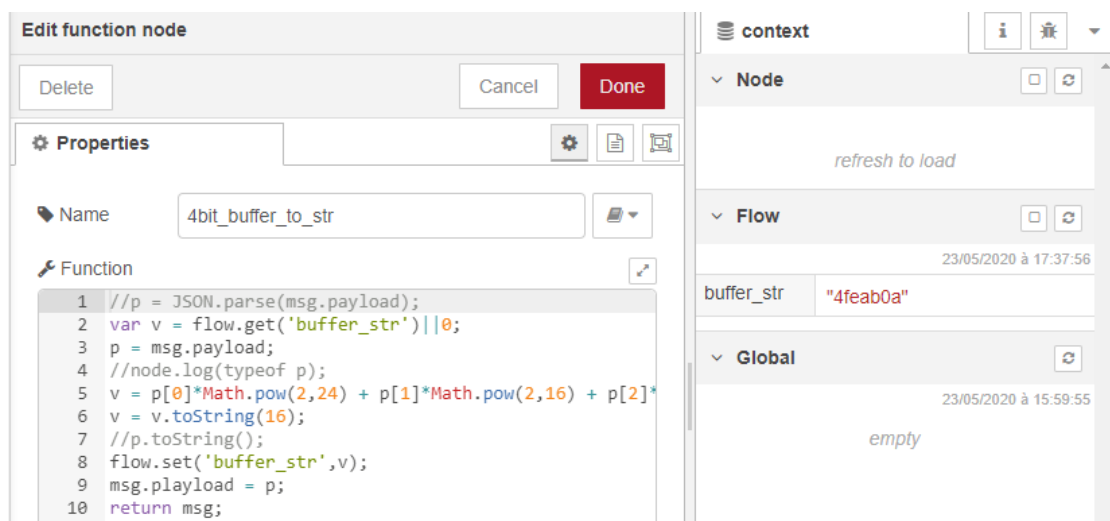


figure 30 4bit\_buffer\_to\_str

Un node « fonction » rajouté pour convertir du buffer à string et conserver ce donnée dans un flow contextdata.

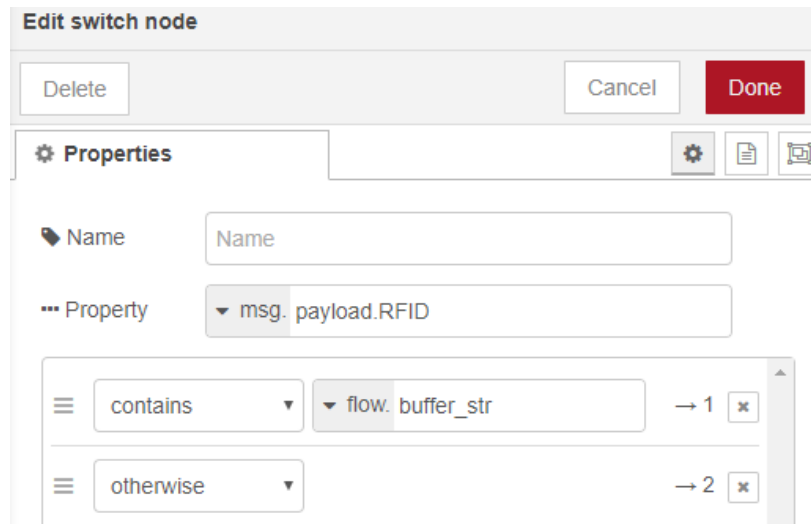


figure 31 Modification « switch »

Modification du node « switch » qui compare les données reçues du serveur influxdb avec la donnée de l'appareil qui a été converti et conservé dans flow contextdata ( buffer\_str ).

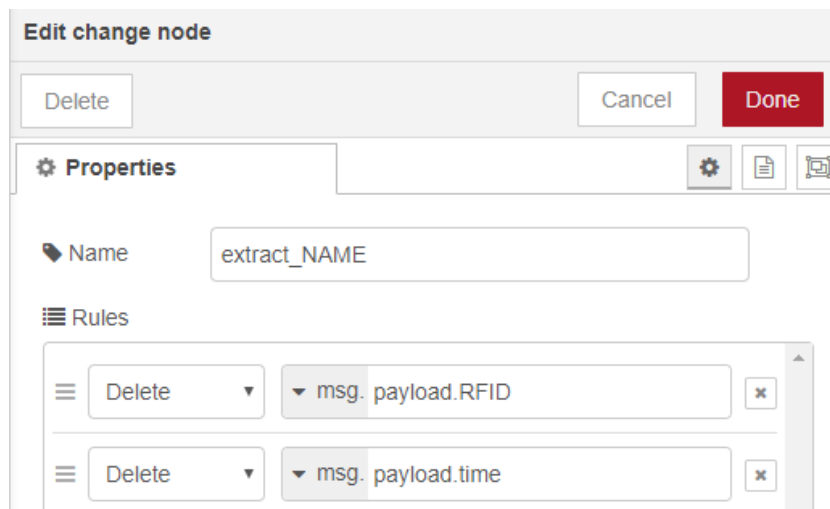


figure 32 Configuration « change »

Mise en place d'un node « change » pour l'extraction des noms d'étudiant.

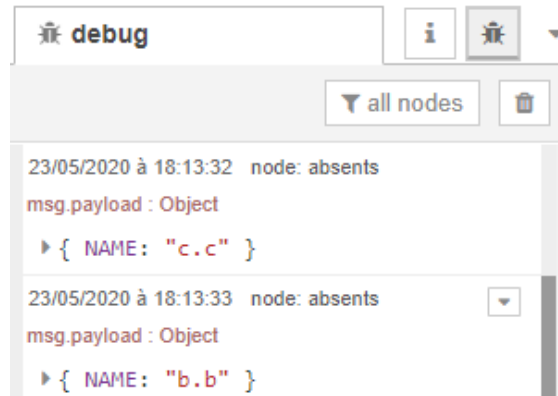


figure 33 Sortie débogage

Et à la fin mise en place d'un node pour exporter des noms des absents dans un fichier txt.

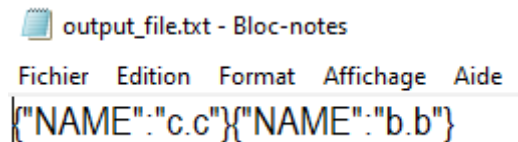


figure 34 Fichier txt

Prochains Objectifs :

- Simulation de plusieurs RFID dans l'envoi de l'appareil ttn
- Amélioration de la mise en forme du fichier output

## 6 Importation d'une liste d'étudiant dans InfluxDB

À cause de la manque du matériel il n'est pas possible d'ajouter un IHM sur la badgeuse donc j'ai importé une liste d'étudiant avec les noms et RFIDs ( générés aléatoirement ) dans InfluxDB en utilisant un tableau des notes DM de la promo GEII 2e année à fin de faciliter les simulations suivantes.

	A	B	C	D
1	timestamp	LASTNAME	FIRSTNAME	RFID
2	2020-06-01 00:00:00	AMINE	SMAIN	5B3FD39E
3	2020-06-01 00:00:01	ANANOS	MORGANE	8004A192
4	2020-06-01 00:00:02	BA	HASSIROU	F7480711
5	2020-06-01 00:00:03	BARCAROLI	LOIC	9BEBE35E
6	2020-06-01 00:00:04	BENVENUTI	TRISTAN	F23411E6
7	2020-06-01 00:00:05	BENZAOUIA	OUSSAMA	C024F4C4
8	2020-06-01 00:00:06	BIN ADNAN	MUADZ LUQMAN	31CF2325
9	2020-06-01 00:00:07	BIN MOHAMAD	MUHAMMAD ADAM	74FBC3B6
10	2020-06-01 00:00:08	BIN MOHD NAZARUDDIN	MUHAMMAD AQIL	F93FDF90
11	2020-06-01 00:00:09	BINTI MOHD RAMLI	NUR AQILAH	765A1EB7

figure 35 Liste d'étudiant en Excel

```

C:\influxdb-1.7.10_windows_amd64\influx.exe
> show series
key
---
student, FIRSTNAME=ALEXANDRE, LASTNAME=NAHON, RFID=E7578836
student, FIRSTNAME=ALEXIA, LASTNAME=HULLIN, RFID=3C1FBCEC
student, FIRSTNAME=ALLEN, LASTNAME=MUJWIL, RFID=6C412842
student, FIRSTNAME=ANAEL, LASTNAME=RONCHARD, RFID=DCB242B4
student, FIRSTNAME=ANTONIN, LASTNAME=LECRON-DEVAUCHELLE, RFID=6AF0A528
student, FIRSTNAME=ARNAUD, LASTNAME=TIBERGHIE, RFID=8D6008EA
student, FIRSTNAME=CEDRIC, LASTNAME=FERNANDES\ VAZ, RFID=7DDE9869
student, FIRSTNAME=ENZO, LASTNAME=GAUTIER, RFID=5FF1DA6B
student, FIRSTNAME=FLORIAN, LASTNAME=THUILLIER, RFID=E31B2102
student, FIRSTNAME=GABRIEL, LASTNAME=FERNANDES, RFID=A54ABF4F

```

figure 36 Liste bien stockée dans InfluxDB

Source : <https://github.com/fabio-miranda/csv-to-influxdb>

## 7 Finalisation de l'automatisation sur Node-RED

### 7.1 Résolution pour démarrer et arrêter la badgeuse :

Nœud ajouté : vacation-timer

(<https://flows.nodered.org/node/node-red-contrib-vacation-timer>)

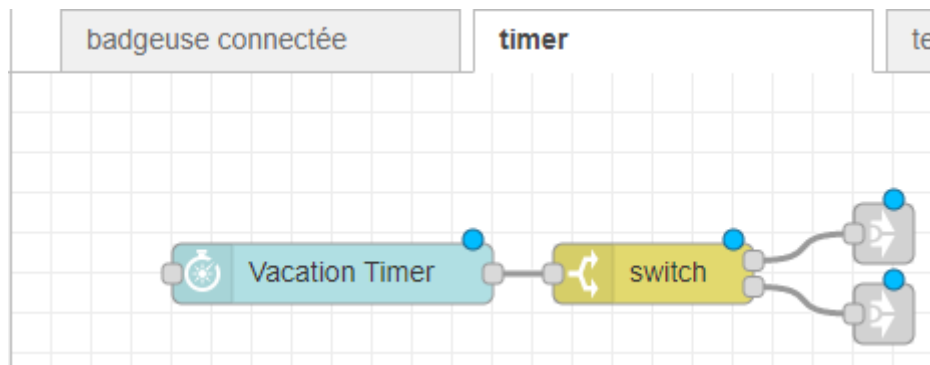


figure 37 Vacation timer

Delete
Cancel
Done

⚙️ Properties

📅 Days of the Week:

☐ Sun
☒ Mon
☒ Tue
☒ Wed
☒ Thu
☒ Fri
☐ Sat

☒ Issue **Off** command if it spans Midnight onto a non-scheduled day

🕒 Choose the Device On Time and Delay:

Time On: 7:00 AM
Max Delay: 0

⚙️ Choose whether you want a Length of Time to turn off the device, or a Specific Time with a Delay:

Use Length of Time
Use Specific Time

🕒 Choose the Device Off Time and Delay:

Time Off: 7:15 AM
Max Delay: 0

figure 38 Configuration Vacation timer

C'est un nœud configurable qui peut envoyer un message à l'heure programmée.

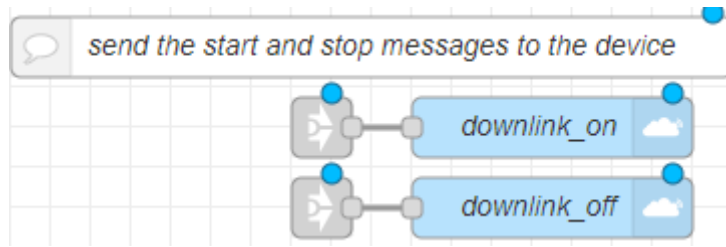


figure 39 Marche/arrêt programmé

Je l'utilise pour envoyer les messages de démarrage et arrêt sur la badgeuse connectée via downlink.

## 7.2 Automatisation pour sortir un fichier excel de la liste des absents :

Nœuds ajoutés :

Excel (<https://flows.nodered.org/node/node-red-contrib-excel>)

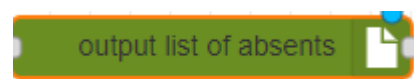


figure 40 Nœud excel

figure 41 Configuration «excel»

C'est un nœud qui permet d'enregistrer les données dans un fichier excel local.

Join-wait (<https://flows.nodered.org/node/node-red-contrib-join-wait>)

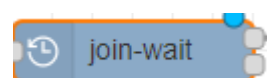


figure 42 Nœud join-wait



C'est un noeud qui permet de conserver des données reçues temporairement et d'attendre un événement pour les envoyer à la sortie.

Avec cette fonction pratique j'ai pu faire une boucle filtrant un groupe de données afin de trouver ce qui reste après l'itération.

### 7.3 Contrôle exhaustif en boucle fermée

Vue générale :

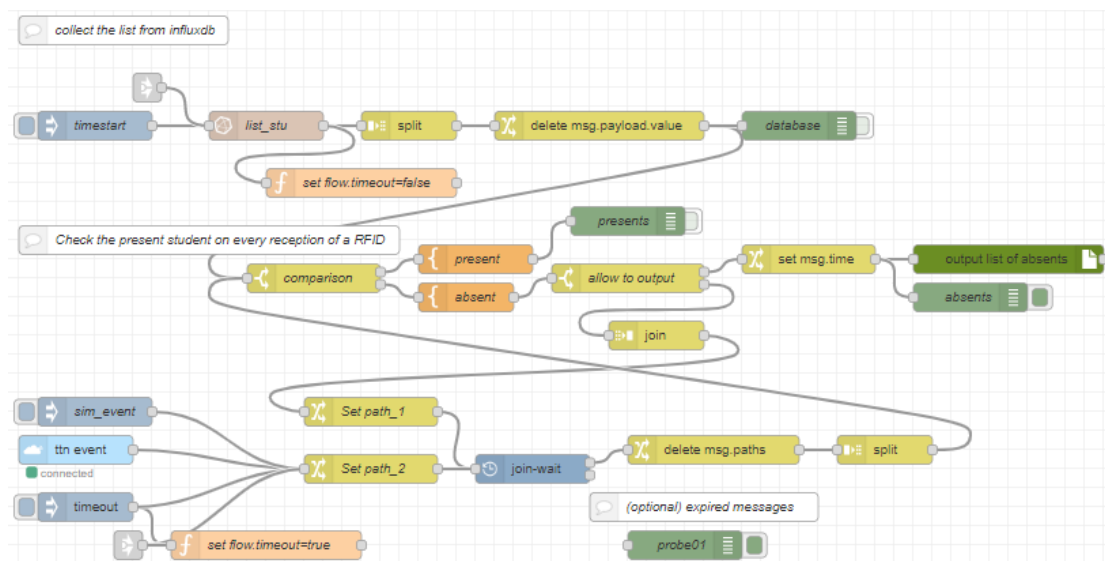


figure 43 3er synoptique

Je récupère la liste des étudiants à partir de la base de données influxdb sur l'événement « on » du vacance-timer qui est présenté dans la partie précédente.

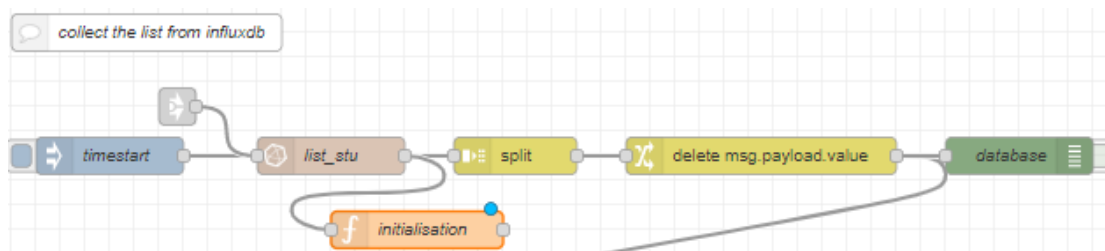


figure 44 Importation de InfluxDB

Comparaison des RFIDs de la liste avec ce qui est reçu de la badgeuse et le champs PRESENCE est ajouté pour chaque étudiant.

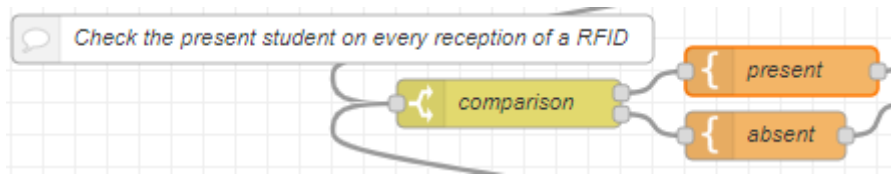


figure 45 « comparison »

figure 46 Configuration « present »

Dans un parcours de la boucle la comparaison est effectuée pour qu'un étudiant donc l'absence n'est pas définitive et il faut refaire la comparaison jusqu'à la fin du temps réparti ( défini par vacance-timer ).

Un nœud join-wait est mis en place pour que les données à itérer ne s'envoient que l'envoi du RFID badgeuse est fini c'est-à-dire quand le RFID à comparer a été mis à jour.

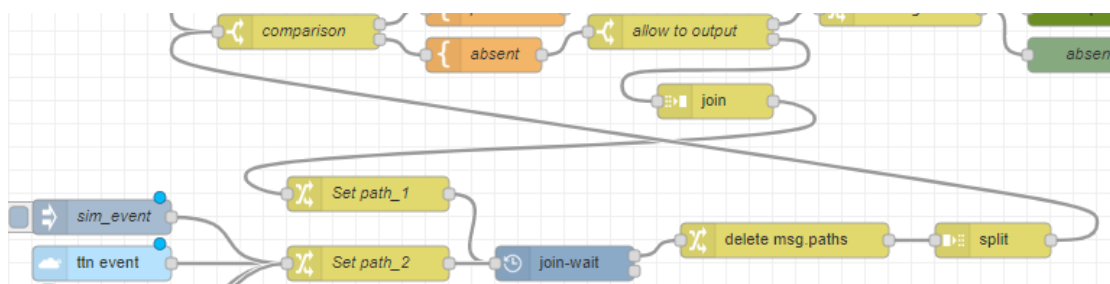


figure 47 Contrôle exhaustif

Autorisation de sortir des données au fichier excel et mise à jour de la date de l'absence

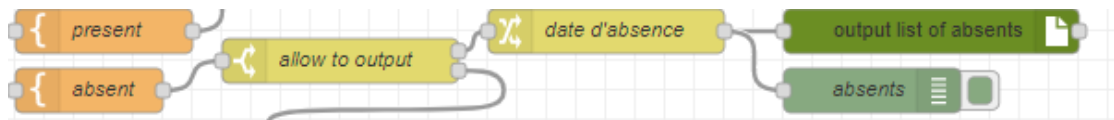


figure 48 Autorisation de sortie

L'autorisation utilise un variable en contextdata « timeout » qui est initialisé à false au début de la boucle et configuré à « true » sur l'événement fin du timer.

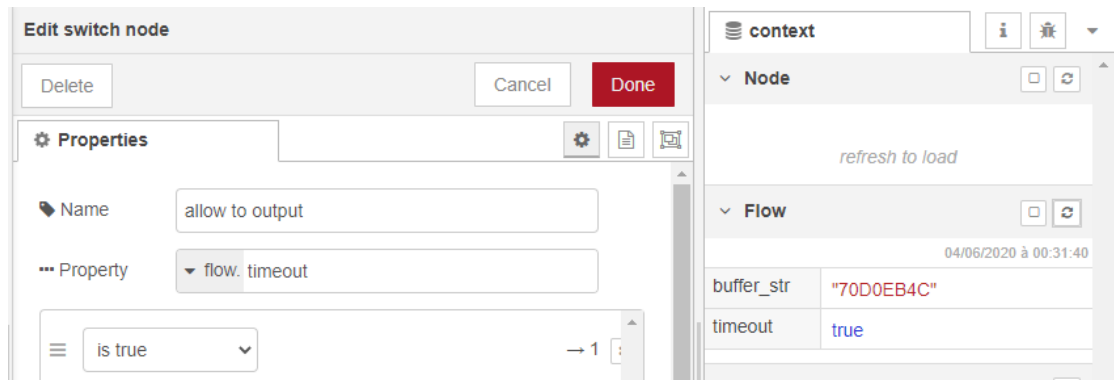


figure 49 Configuration switch-2

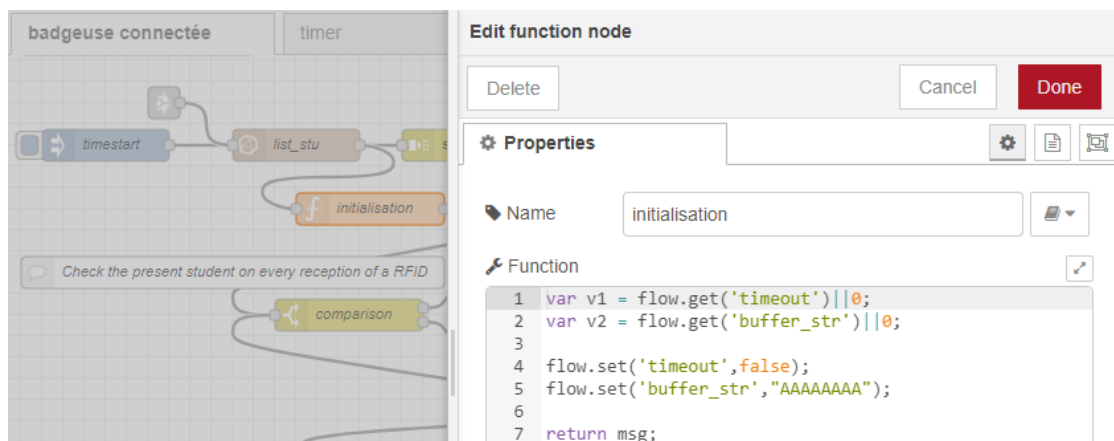


figure 50 «Initialisation»

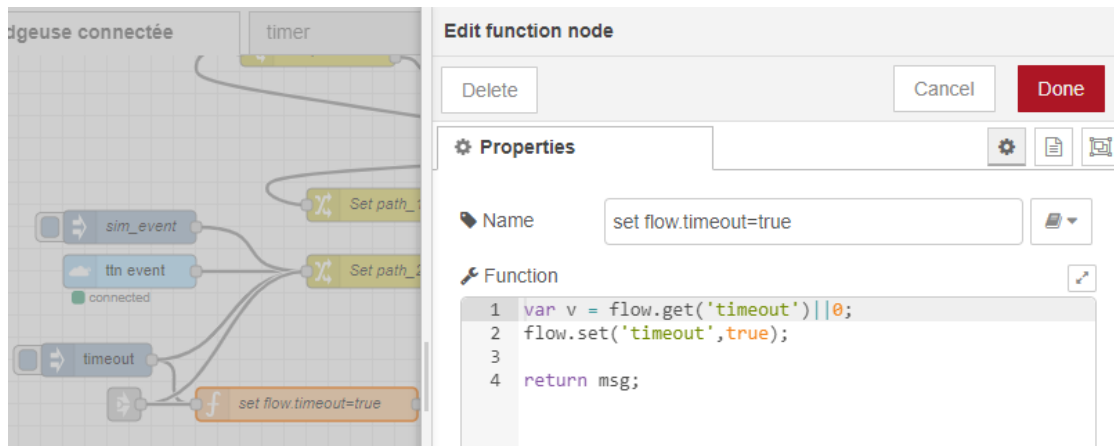


figure 51 Mise à jour de la variable timeout

Avant d'exporter les absents dans la liste excel, j'ai ajouté un nœud pour mettre à jour la date.

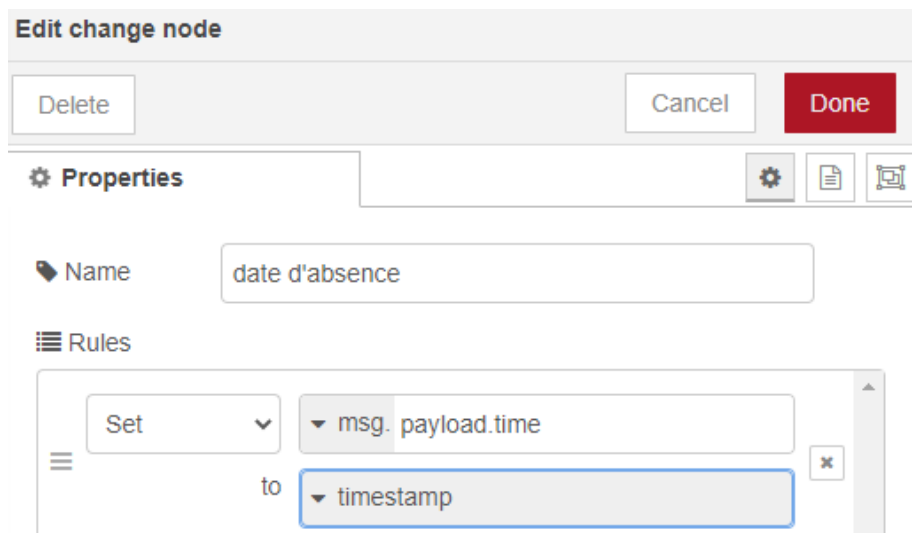


figure 52 Mise à jour de date

Comme résultat final, je récupère dans la répertoire local le fichier excel des absents :

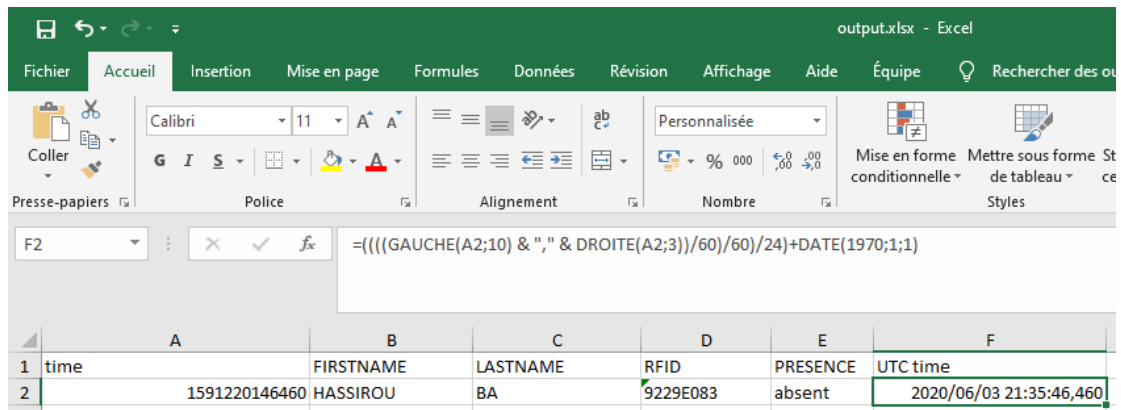


figure 53 Fichier de sortie Excel

Toutes les colonnes sont générées automatiquement par le nœud excel sauf le format du temps n'est pas adapté par excel donc j'ai ajouté une colonne « UTC time » pour afficher la date correctement avec la formule

$$UTC\ time = (((GAUCHE(A2;10) \& "," \& DROITE(A2;3))/60)/60)/24) + DATE(1970;1;1)$$

En mettant le format de cellule comme ci-dessous.

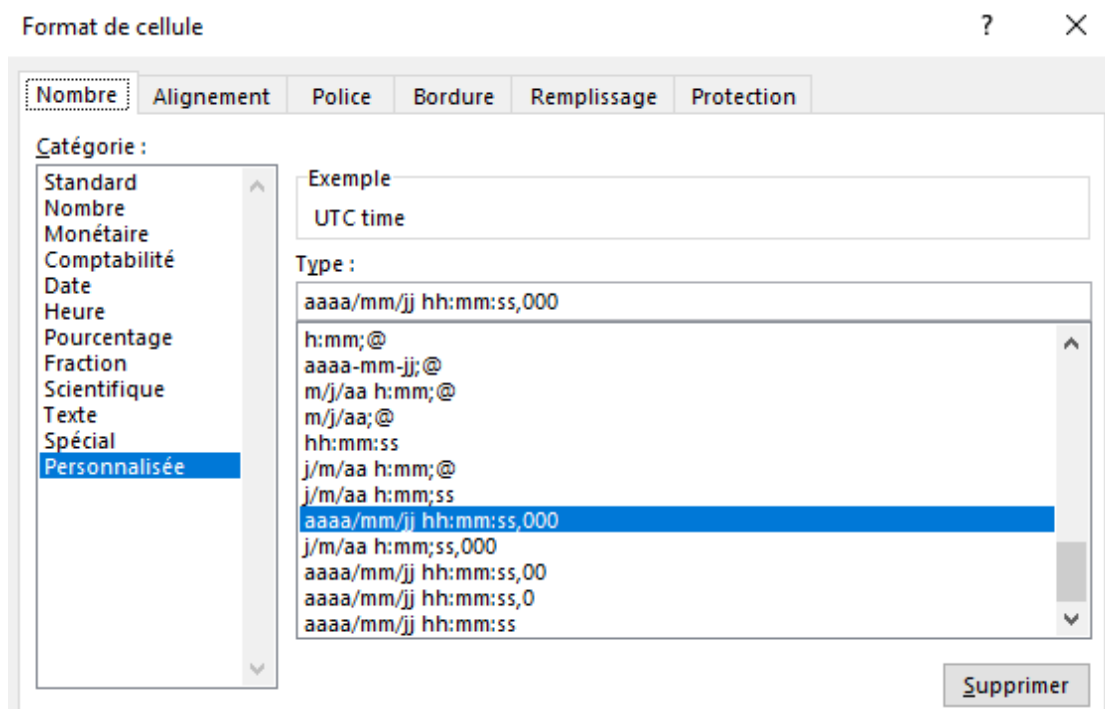


figure 54 Format de cellule «date»

## 8 Badgeuse à machine à état

Afin d'assurer le fonctionnement normal de la badgeuse selon le cahier des charges, j'ai conçu une machine à état.

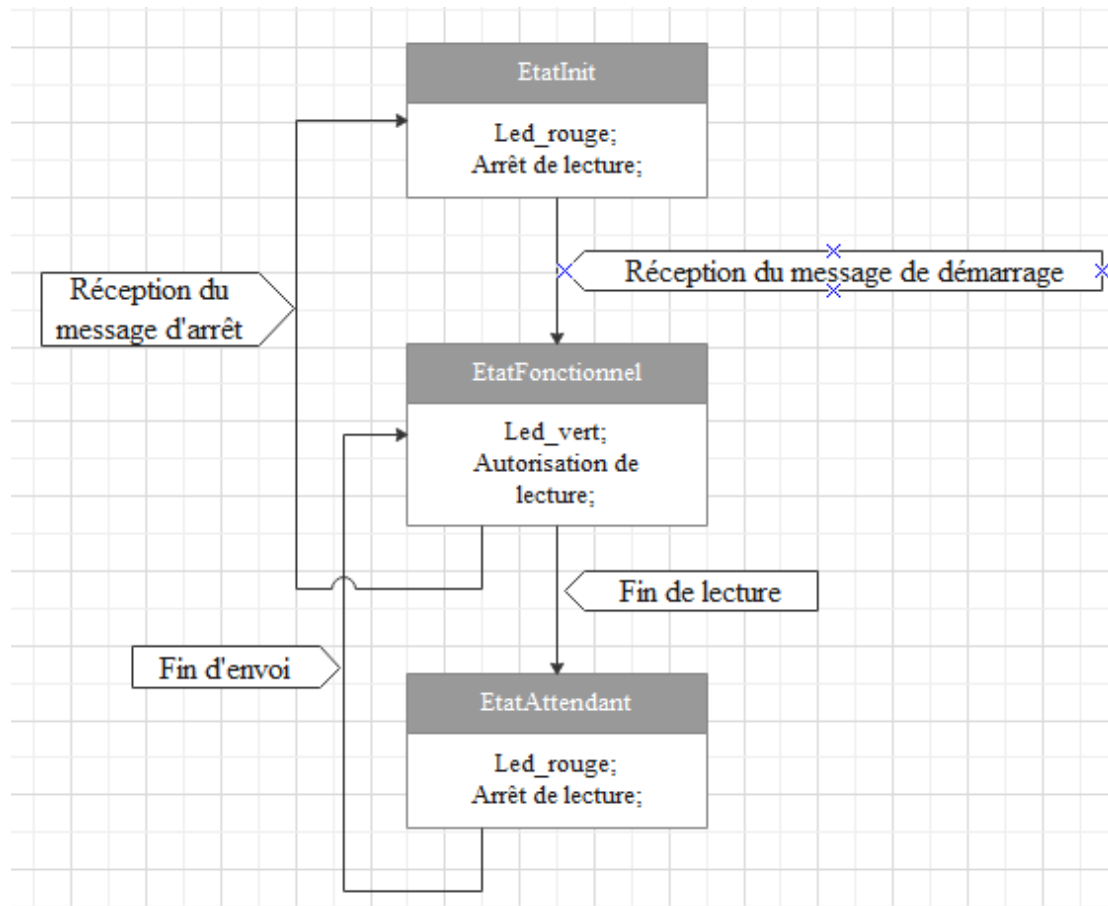


figure 55 Machine à état de la badgeuse connectée

La badgeuse se démarre quand elle reçoit le message de démarrage du serveur Node-RED et elle s'arrête lorsque le reçu du message d'arrêt.

À part l'état initial, un état fonctionnel et un état attendant sont mis en place pour assurer la lecture du tag RFID.

Cette conception est basée sur la supposition qu'une liste d'étudiant soit existante déjà dans le serveur InfluxDB, dans le cas contraire il faut considérer d'ajouter un IHM et de faire une autre machine à état.

Dû à la durée limitée du stage, je n'avais pas pu le faire.

## Conclusion

Nous sommes dans une période spéciale et difficile, il est compréhensible que la durée du stage ne soit pas assez longue que l'année dernière.

Personnellement l'internet of things était un sujet peu connu pour moi car je n'étais pas dans la groupe avec M.Diallo et M.Laurent dans ER4. Il m'a pris beaucoup de temps à installer l'environnement de travail et à apprendre l'utilisation des nouveaux outils de développement. Cependant M.Diallo est très responsable et pédagogique, il m'a donné des instructions qui m'a permis d'avancer.

Ce projet n'est pas fini techniquement, le programme de la badgeuse reste à compléter et j'aurais dû faire plusieurs tests pour le programme du Node-RED. À cause par manque du temps et du matériel, je n'ai pas pu finaliser.

En tant qu'un premier stage, j'ai appris le travail en autonomie et la communication avec le tuteur de stage et je pourrais en servir dans les expériences professionnelles à venir.

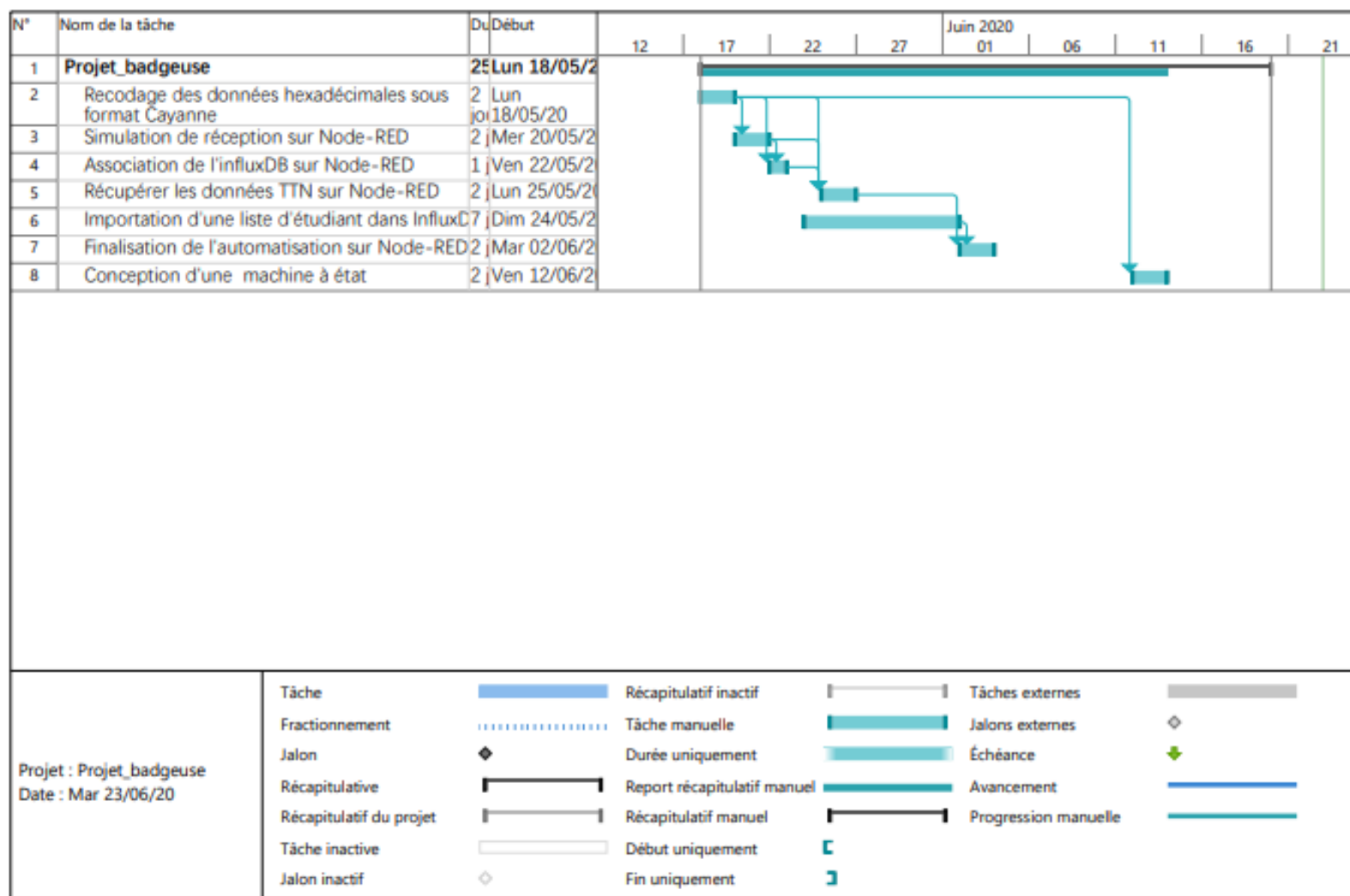


figure 56 Diagramme Gantt du projet



## Sources et documentations

- Codes complets du simulateur\_LoRaWAN et Node-RED ( [https://kinue00.github.io/Stage\\_S4/](https://kinue00.github.io/Stage_S4/) )
- Working with Bytes  
( <https://www.thethingsnetwork.org/docs/devices/bytes.html> )
- Tutorial d'installation TTN par M.Benvenuti  
( <https://www.youtube.com/watch?v=RpzmUqRq6x0&feature=youtu.be> )
- Gestion des bases de données InfluxDB  
( <https://docs.influxdata.com/influxdb/v1.8/tools/shell/> )

## Table des illustrations

figure 1 Schéma du réseau des objets connectés .....	7
figure 2 B-L072Z-LRWAN1 Discovery kit.....	9
figure 3 Interface du simulateur en ligne .....	10
figure 4 Topologie d'un réseau LoRaWAN .....	10
figure 5 Les 2 types d'activation OTAA et ABP .....	12
figure 6 Paramètres de l'application et l'appareil sur the things network .....	13
figure 7 l'ajout d'une nouvelle application TTN.....	13
figure 8 Configuration des clés de l'appareil .....	14
figure 9 Vérification de la version node.js et npm en CMD .....	15
figure 10 Interface Node-RED.....	15
figure 11 les noeuds TTN .....	16
figure 12 Payload Structure.....	17
figure 13 Tableau des types Cayannes .....	18
figure 14 Envoi d'une trame de température et humidité en cayanne	18
figure 15 Création d'un buffer pour un data température .....	19
figure 16 une autre variation pour la méthode masquage et décalage	19
figure 17 mbed LPC1768 et sht31 simulés .....	20
figure 18 Serial output pour l'envoi de température et humidité .....	21
figure 19 Application data reçu ( température et humidité ) .....	21
figure 20 Synoptique avec ttn_uplink .....	22
figure 21 Configuration ttn uplink node .....	22
figure 22 Configuration ttn app node .....	23
figure 23 Configuration influxdb out.....	23
figure 24 Configuration influxdb node.....	24
figure 25 InfluxDB shell .....	25
figure 26 Debug Node-red.....	25
figure 27 1er synoptique .....	25
figure 28 Configuration du switch .....	26
figure 29 2er synoptique .....	27
figure 30 4bit_buffer_to_str.....	27
figure 31 Modification « switch » .....	28
figure 32 Configuration « change ».....	28
figure 33 Sortie débogage.....	29
figure 34 Fichirt txt.....	29
figure 35 Liste d'étudiant en Excel.....	30

figure 36 Liste bien stockée dans InfluxDB .....	30
figure 37 Vacation timer .....	31
figure 38 Configuration Vacation timer.....	31
figure 39 Marche/arrêt programmé.....	32
figure 40 Noeud excel .....	32
figure 41 Configuration «excel».....	32
figure 42 Noeud join-wait .....	32
figure 43 3er synoptique .....	33
figure 44 Importation de InfluxDB.....	33
figure 45 « comparison » .....	34
figure 46 Configuration « present » .....	34
figure 47 Contrôle exhaustif .....	34
figure 48 Autorisation de sortie.....	35
figure 49 Configuration switch-2 .....	35
figure 50 «Initialisation» .....	35
figure 51 Mise à jour de la variable timeout .....	36
figure 52 Mise à jour de date .....	36
figure 53 Fichier de sortie Excel.....	37
figure 54 Format de cellule «date» .....	37
figure 55 Machine à état de la badgeuse connectée.....	38
figure 56 Diagramme Gantt du projet .....	40