Projets

- LoRaWAN - The Things Network
- ESP32 - LED RGB pilotée via Wifi
- ESP32 - BMP180
- Schéma LoRa + Grafana-Influx + NodeRED

LoRaWAN - The Things Network

Objectif(s)

Créer un objet qui enverra via LoRa l'humidité et la température ambiante.

The Things Network (TTN)



The Things Network est une association qui a permis de créer un réseau communautaire LoRaWAN. Tout ceux qui le veulent peuvent apporter leur pierre à l'édifice en installant chez eux ou dans leur entreprise une passerelle LoRa. L'EPSI en possède une installée dans le myDIL.

Créer un compte et une application sur https://console.thethingsnetwork.org

Mise en place

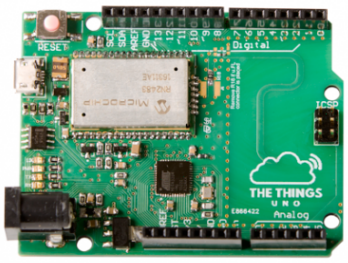
Voici en résumé ce que nous devons faire :

1. Installation de la carte sous Arduino.
2. Récupérer le devEUI de la carte.
3. Créer l'objet dans TTN à partir de ce devEUI.
4. Utiliser un code typique pour tester la connexion en utilisant les clés fournies lors de l'étape précédente.
5. Connecter le capteur (sans LoRa).
6. Fusionner les deux codes précédents pour envoyer la température et l'humidité.

7. Récupérer les données et les afficher dans un dashboard NodeRed.

Installation de la carte Things Uno sous Arduino





Au niveau matériel, rien à installer, sélectionner "Arduino Leonardo" dans le gestionnaire de carte.

La conception de la carte est assez simple : un modem LoRa est associé avec un microcontroleur (µC) Atmega32u4 qui équipe habituellement les Arduino Léonardo. Le µC et le modem communiquent en liaison série en s'envoyant des commandes formatées appelées commande AT. La librairie installée permet de faire abstraction de cette communication, mais on pourrait très bien communiquer avec ce modem directement depuis le moniteur série du PC par exemple.

Si jamais la carte vous pose des problèmes (port série non reconnu etc ..) => https://www.thethingsnetwork.org/docs/devices/node/troubleshooting.html

Dans le gestionnaire de librairies, installer **TheThingsNetwork**.

Récupération du devEUI

Récupérer le **devEUI** de votre carte => https://www.thethingsnetwork.org/docs/devices/uno/quick-start.html

Créer le device sur TTN

Sur la console TTN, dans votre application créer un device. Saisissez le **devEUI** précédemment récupéré et laissez les autres champs en génération automatique.

Tester la liaison

Sur Arduino, allez chercher *Fichier>Exemples>TheThingsNetwork>SendOTAA*. Modifier les différents champs importants pour correspondre à votre configuration (AppKey, FrequencyPlan ...)

et vérifier dans le moniteur série que tout se passe bien (JOIN et envoi).

Vérifier sur la console TTN que les données arrivent bien.

> Si vous rencontrer des problèmes de type "invalid_param" en console, il vous faut downgrader votre version de la librairie TheThingsNetwork en 2.5.16

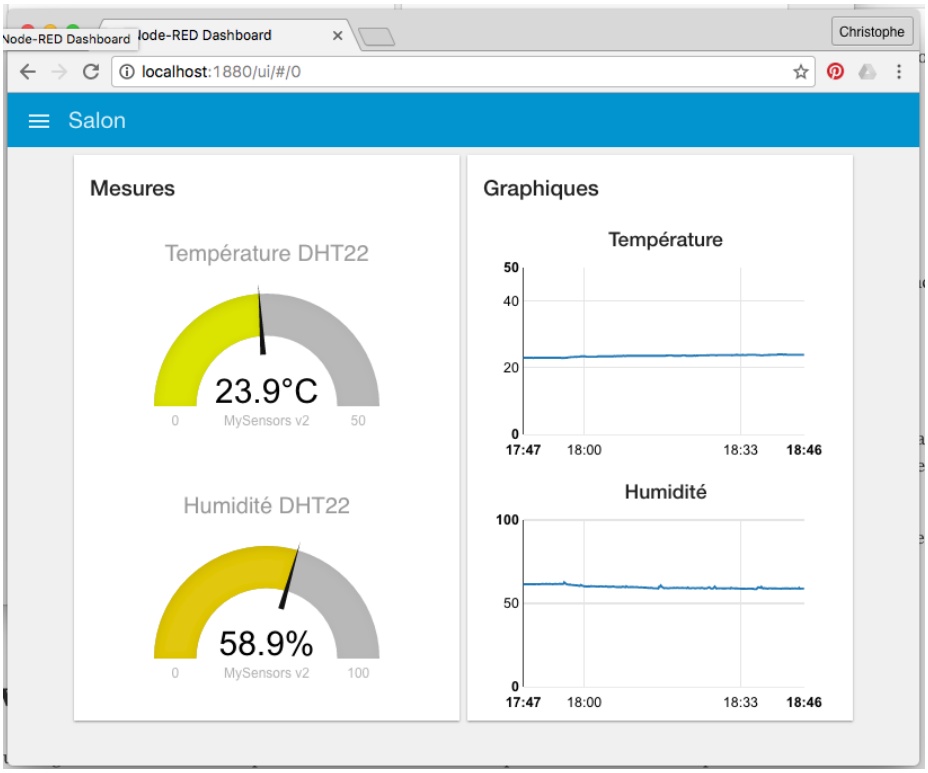
Connecteur de capteur d'humidité/température (DHT11)

Ajouter ensuite un DHT11, récupérer ses données de température et d'humidité, et les envoyer via LoRa.

Connecter NodeRED

Lancer ensuite un serveur NodeRed afin de récupérer les données sur le MQTT TTN :
https://www.thethingsnetwork.org/docs/applications/mqtt/api.html

Et les afficher sur un dashboard NodeRed (node-red-dashboard).

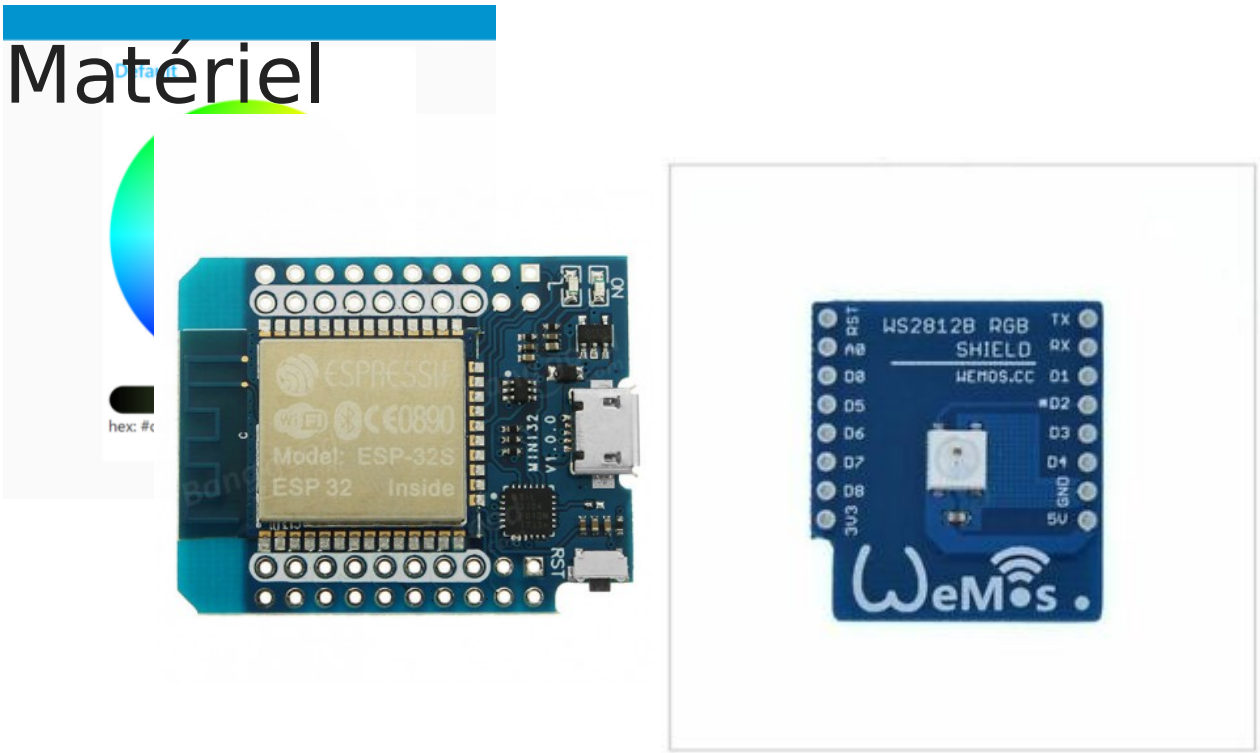


ESP32 - LED RGB pilotée via Wifi

Objectif

Contrôler une **LED RGB**, elle même piloté par un **ESP32**, au travers d'un dashboard NodeRed et de son outil "Color picker". Le protocole utilisé sera le **MQTT** et le transit des information se fera au format **json**.

Matériel



- Un **ESP32**
  - Carte de développement avec Wifi et bluetooth BLE.
- Un **shield LED WS1812**
  - Se pilote grâce à un fil
  - **La LED est reliée au pin 21 de l'ESP**

Logiciel

Ajout de l'ESP 32 à Arduino si ce n'est pas déjà fait : https://doc.creatronic.fr/books/i2---conception-dun-syst%C3%A8me-embarqu%C3%A9-temps-r%C3%A9el/page/installation-arduino

> Utiliser la board **Wemos LOLIN32**.

Pour contrôler la LED il existe beaucoup de librairies. Une des plus simple pour débuter est **Adafruit Neopixel**. Pour l'installer il faudra ajouter l'URL suivante dans le menu *préférences* :

```
https://adafruit.github.io/arduino-board-index/package_adafruit_index.json
```

> Il y a un **bug** dans le mariage Adafruit Neopixel <-> ESP32. Les couleurs peuvent ne pas correspondre à vos attentes, notamment quand ça touche à la couleur rouge ... Pour le résoudre, il vous suffit de **doubler l'appel à la fonction show()**.

Pour le MQTT, utilisez **PubSubClient** et **ArduinoJson** pour la gestion des json.

Livrable attendu

Pour obtenir la moyenne vous devrait produire un démonstrateur fonctionnel. Pour obtenir des points supplémentaires, ajoutez des fonctionnalités et fiabilisez votre code.

ESP32 - BMP180

Objectif(s)

Créer un objet connecté en wifi qui enverra via **MQTT** la température et la pression atmosphérique vers un serveur qui stockera ces informations en base de données **InfluxDB** avant de les restituer au travers de la solution **Grafana**.

> Répartissez vous bien les tâches : le travail est divisible et il y a pas mal de choses à faire ...

Arduino

Ajout de l'ESP 32 à Arduino si ce n'est pas déjà fait : https://doc.creatronic.fr/books/i2---conception-dun-syst%C3%A8me-embarqu%C3%A9-temps-r%C3%A9el/page/installation-arduino

> Utiliser la board **Wemos LOLIN32**.

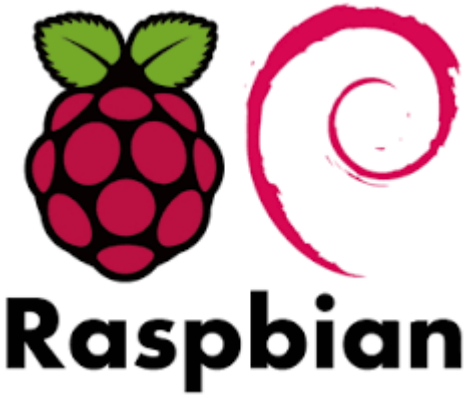
Librairie conseillé(e) : Adafruit_BMP085

Pour l'installer il faudra ajouter l'URL suivante dans le menu préférences :

```
https://adafruit.github.io/arduino-board-index/package_adafruit_index.json
```

Rpi

Préparation du Raspberry Pi (installation Raspian)



Si Raspbian est déjà installé vous pouvez sauté cette étape.

Télécharger la dernière version de Raspbian : https://www.raspberrypi.org/downloads/raspbian/

Flasher l'image sur la carte SD avec Balena Etcher : https://www.balena.io/etcher/

> Avant de débrancher la carte SD, sur la partition boot ajouter un fichier **ssh** (pas un .ssh, un fichier ssh tout court **sans extension**) afin que le système autorise le ssh bloqué par défaut.

Si vous souhaitez travailler en Wifi, il faut le configurer avant de mettre la carte SD dans le Raspberry : https://www.raspberrypi-spy.co.uk/2017/04/manually-setting-up-pi-wifi-using-wpa_supplicant-conf/

Mise à jour du système :

```
sudo apt update && sudo apt dist-upgrade && sudo apt upgrade
```

Installation de NodeRed



```
cd ~
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
sudo systemctl enable nodered.service
sudo node-red-start
```

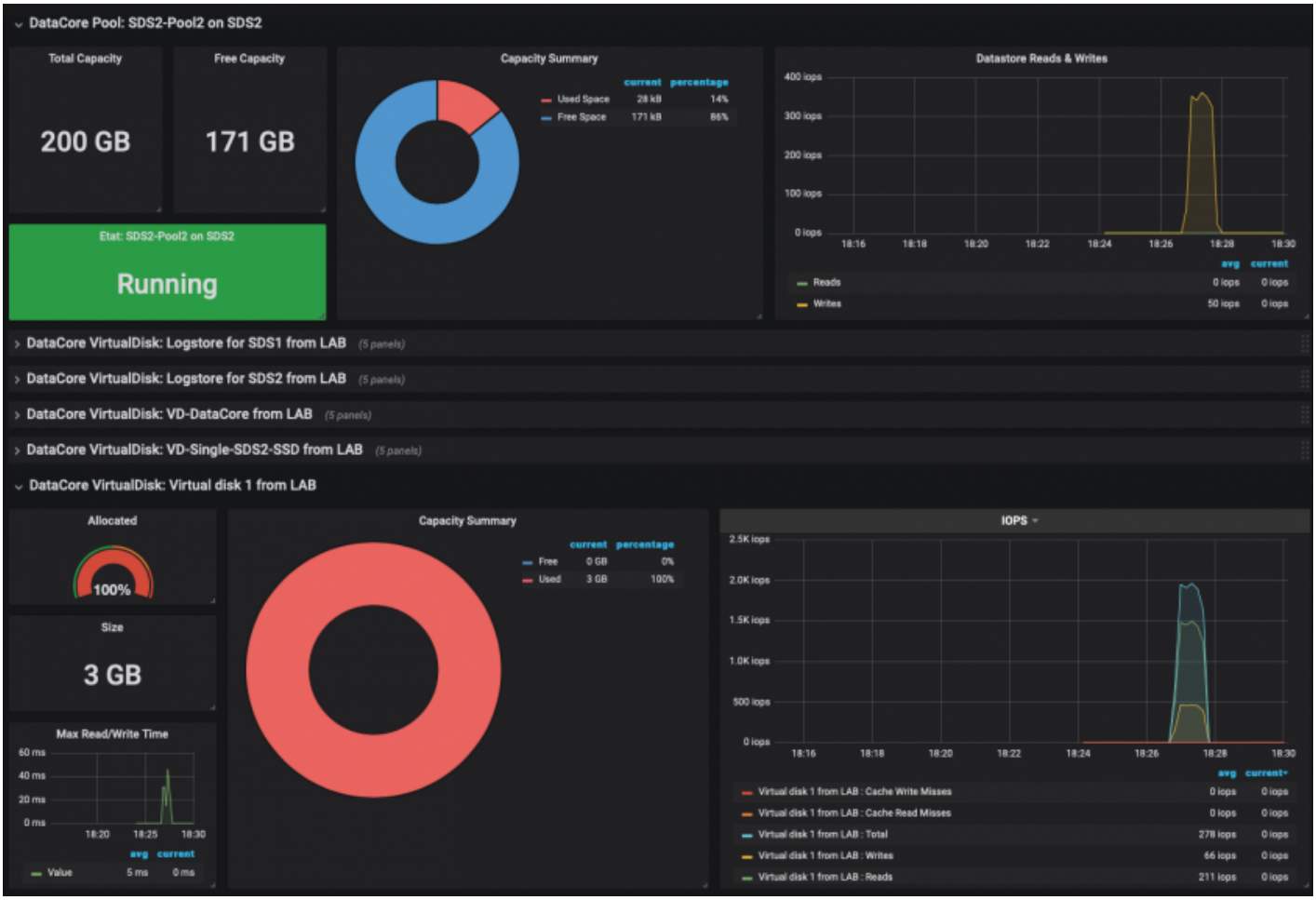
Installation de InfluxDB

```
curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
echo "deb https://repos.influxdata.com/debian buster stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
sudo apt-get update && sudo apt-get install influxdb
sudo service influxdb start
```

Si tout va bien, taper "influx" pour accéder à la console influxdb.

Créer une base de donnée : `create database lenomquevousvoulez;`

Installation de Grafana



```
sudo apt update && sudo apt-get install -y software-properties-common apt-transport-https
sudo nano /etc/apt/sources.list.d/grafana.list
```

Ajouter le dépôt suivant puis enregistrer.

```
deb https://packages.grafana.com/oss/deb stable main
```

Puis :

```
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
sudo apt-get update && sudo apt-get install grafana
sudo service grafana-server start
```

Doc officielle en cas de problème : https://grafana.com/docs/installation/debian/

Accessible sous http://ip:3000

Login/mdp par défaut : admin/admin

Ajout des nodes nécessaires dans node-red

Ajouter les nodes suivants :

node-red-contrib-influxdb => va nous permettre de dialoguer avec les bases influxDB

Travail à réaliser

Sur **nodeRED**, créer un flow permettant de récupérer et stocker les données de l'objet connecté et de l'insérer dans un *measurement* influxDB.

Sur Grafana : créer un nouveau dashboard qui permettra d'afficher un graph et une gauge de la charge CPU (penser à activer l'auto refresh).

Schéma LoRa + Grafana-Influx + NodeRED

