Année 2019

REVUE DE PROJET CROSSDOCK

Zone de Stockage &
Monitoring/Administration

BTS SN

crossdock logistique

Joris LE GALLO, Clément PELOUX,
Alban JUAN, Alan JULLIARD
Lycée Alphonse Benoit
Année 2019

INTRODUCTION

Présentation de l'entreprise

CROSSDOCK

Renseignements juridiques

Dénomination	CROSSDOCK
Adresse	CROSSDOCK, 122 ALL DE LA LAVANDE 84300 CAVAILLON
Téléphone	04 90 05 06 44
SIREN	795 042 332
SIRET (siege)	79504233200028
N° de TVA Intracommunautaire	
Activité (Code NAF ou APE)	Transports routiers de fret interurbains (4941A)
Forme juridique	SASU Société par actions simplifiée à associé unique
Date immatriculation RCS	12-09-2013
Date de dernière mise à jour	01-01-2018
Tranche d'effectif	6 à 9 salariés
Capital social	20 000,00 €
Chiffre d'affaires 2017	596 900,00 €

CROSSDOCK, société par actions simplifiée à associé unique est en activité depuis 5 ans.
Située à CAVAILLON (84300), elle est spécialisée dans le secteur d'activité des transports routiers de fret interurbains. Son effectif est compris entre 6 et 9 salariés.

Sur l'année 2017 elle réalise un chiffre d'affaires de 596 900,00 €.

Le total du bilan a augmenté de 20,97 % entre 2016 et 2017.

Societe.com recense 1 établissement actif et le dernier événement notable de cette entreprise date du 20-11-2017.

Mohamed BIJOU, est président de l'entreprise CROSSDOCK.

Situation Géographique de CROSSDOCK :

Comme on peut le voir sur le plan le site est assez éloigné, et donc nous n'avons pas de photos récentes de CROSSDOCK.

L'entreprise se situe dans la périphérie de CAVAILLON comme nous pouvons le voir ci-dessus.

Présentation du projet

Constitution de l'équipe de projet :	Étudiant 1		Étudiant 2		Étudiant 3		Étudiant 4	
	EC	**IR**	EC	**IR**	**EC**	IR	**EC**	IR
Projet développé :	Au lycée ou en centre de formation		En entreprise		**Mixte**			
Type de client ou donneur d'ordre (commanditaire) :	Entreprise ou organisme commanditaire : **Oui** Non Nom : CrossDock Adresse : 122, allée de la Lavande – 84300 CAVAILLON Contact : M. BIJOU Mohamed Origine du projet : ➢ Idée : Lycée **Entreprise** ➢ Cahier des charges : Lycée **Entreprise** ➢ Suivi du projet : **Lycée** **Entreprise**							
Si le projet est développé en partenariat avec une entreprise :	Nom de l'entreprise : CrossDock Adresse de l'entreprise : 122, allée de la Lavande – 84300 CAVAILLON Site WEB : N/A Tél. : 0490050644 Courriel : mohamed.bijou@crossdk.com							

L'entreprise Crossdock, spécialisée dans le secteur d'activité des transports routiers de fret interurbains (code APE: 4941A), souhaite superviser à terme la température et l'hygrométrie de 2 entrepôts situés sur leur site de Cavaillon (1 seul entrepôt opérationnel à ce jour).

•Les relevés doivent permettre une traçabilité sur les bonnes conditions d'entreposage des produits présents dans l'entreprise

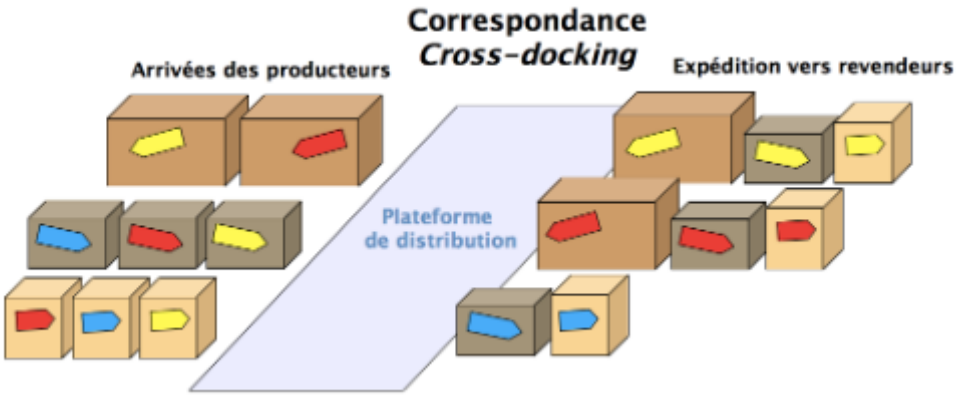
•Les mesures doivent être consultables en direct par CrossDock mais aussi par leurs clients

•Le projet porte sur la surveillance de la température et de l'humidité relative (abrégées T/HR par la suite) au niveau des zones d'expédition des commandes clients

•Ces mesures doivent compléter celles effectuées dans les zones de stockage de chaque entrepôt (cf. projet «Supervision Hygro-Temp–Stockage»

Présentation du principe « Cross-Docking »

CrossDock est une entreprise de logistique dont l'activité consiste à préparer puis à expédier des commandes constituées de produits cosmétiques et/ou de parapharmacie.
La société tire son nom d'un mode d'organisation de flux logistiques, appelé «Cross-docking», qui vise à faire passer des marchandises des quais d'arrivée aux quais de départ en limitant au maximum leur stockage dans les entrepôts de la plateforme de distribution.



Chez CrossDock, les produits arrivent des fournisseurs puis sont entreposés provisoirement en attente de leur reconditionnement au cours de la préparation des commandes. Les colis préparés sont ensuite aiguillés vers des postes regroupant les commandes propres à un revendeur (client final de Crossdock) puis sont enfin expédiés.

Analyse du système existant

Une supervision de température et d'hygrométrie élémentaire est déjà en place chez CrossDock. Elle s'articule autour l'utilisation de thermomètres USB (Réf.: PCE-HT 71Nde chez PCE Instruments) qui font également office d'enregistreur de données(ou «data logger»).

La capacité mémoire de ces dispositifs limite l'enregistrement des variations de températures et taux d'hygrométrie à une période d'1 semaine environ. Ceci implique qu'une personne doit chaque semaine :

•Récupérer les 4 «data-logger» répartis aux 4 coins de l'unique entrepôt actuellement utilisé

•Transférer les mesures effectuées dans le système d'information de l'entreprise en veillant à respecter l'affectation de chaque «data-logger» à la zone de l'entrepôt à laquelle il est associé

•Effacer les données de chaque «data-logger»

•Remettre en place les «data-logger» dans l'entrepôt.

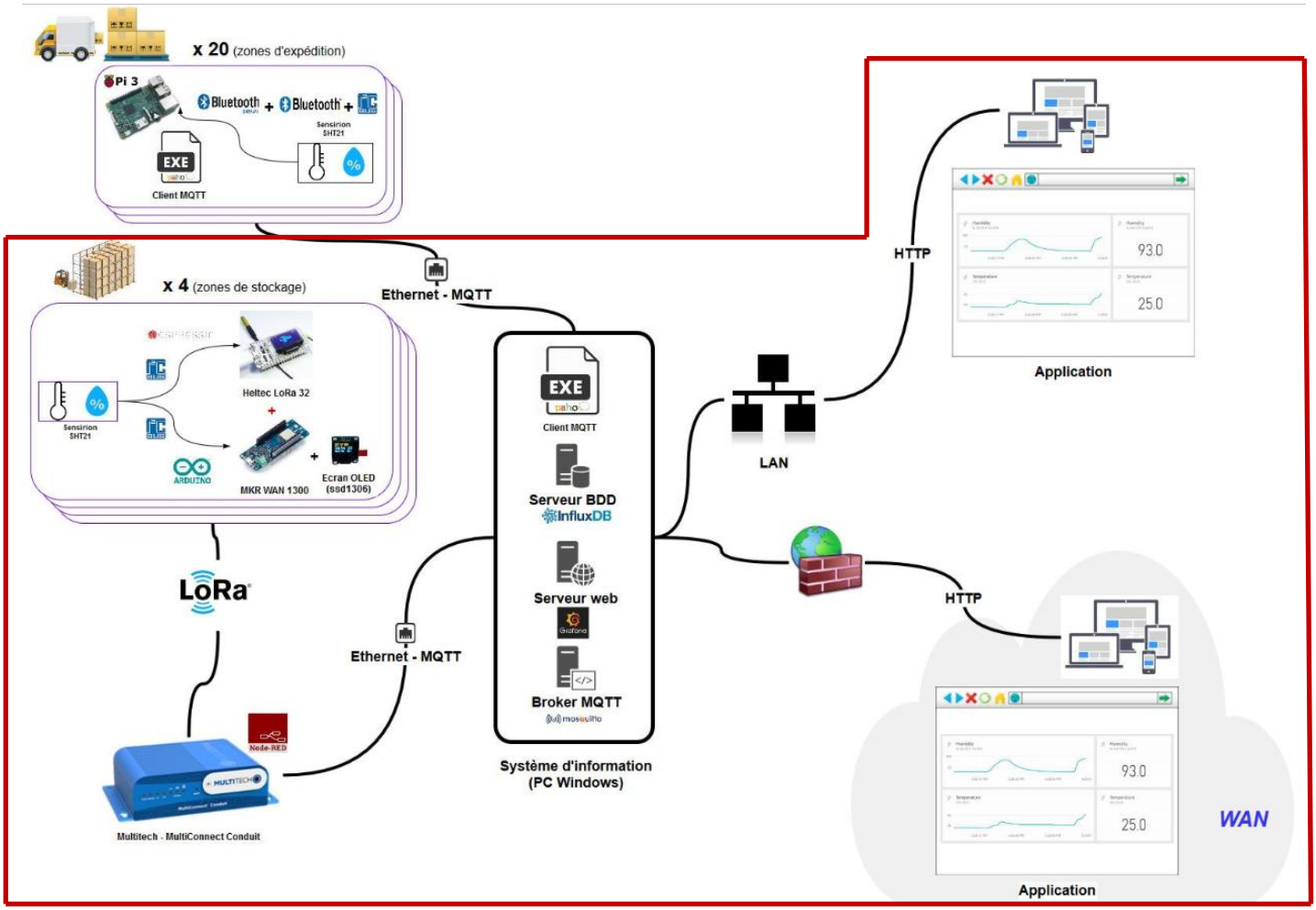
Outre le côté rébarbatif de cette tâche, les données ne sont consultables que la semaine suivant celle de la mesure.
La supervision actuelle ne permet pas non plus de surveiller la température et l'hygrométrie des articles une fois sortis de leur zone de stockage pour constituer les commandes clients et attendre leurs expéditions.

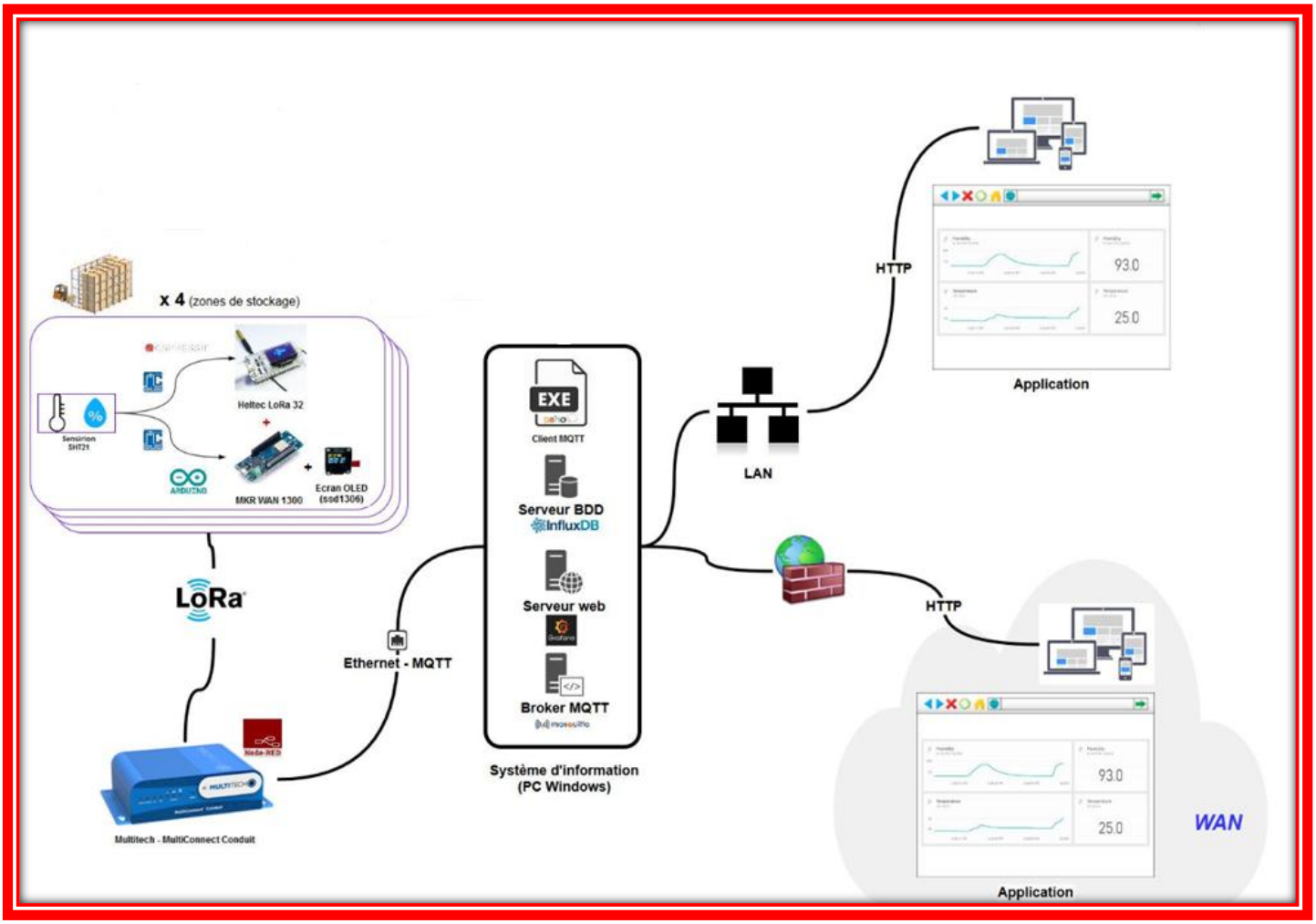
Cahier des charges de l'entreprise

L'entreprise n'a pas fourni de cahier des charges écrit.

Celui-ci a été élaboré au cours de 2 entretiens réalisés dans les locaux de l'entreprise et complété à la suite d'échanges téléphoniques et de mails.

Solution globale proposée





Notre travaille consiste à récupérer les mesures de température et d'humidité, de les centraliser sur un serveur afin de les stocker et de les retransmettre sur des applications, une pour l'entreprise Crossdock et une application pour ses clients.

Spécifications

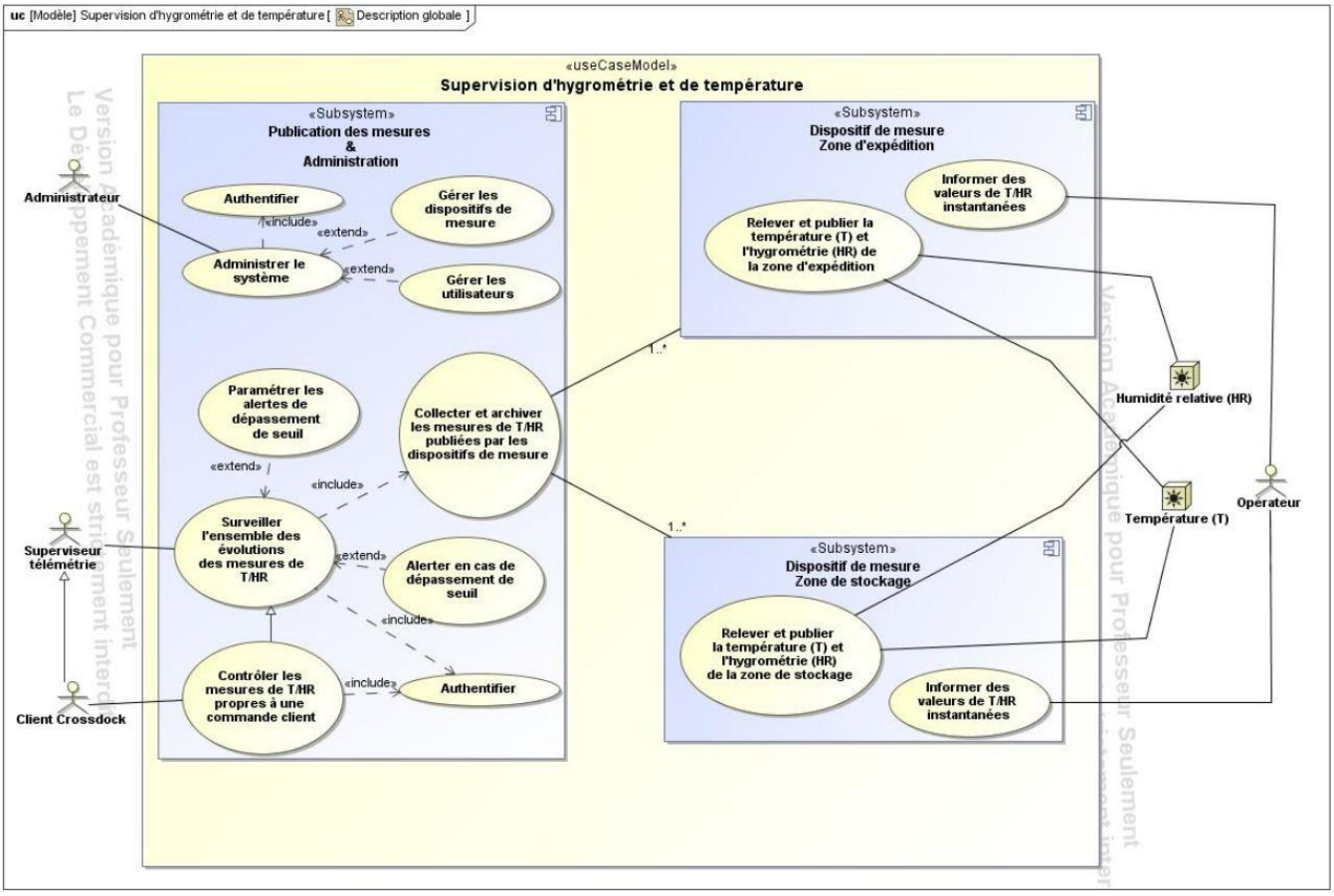
Diagrammes UML / SYSML

Diagrammes des cas d'utilisation

Ci-dessous figure le diagramme des cas d'utilisation du système global.
Celui-ci est décomposable en 3 sous-systèmes:

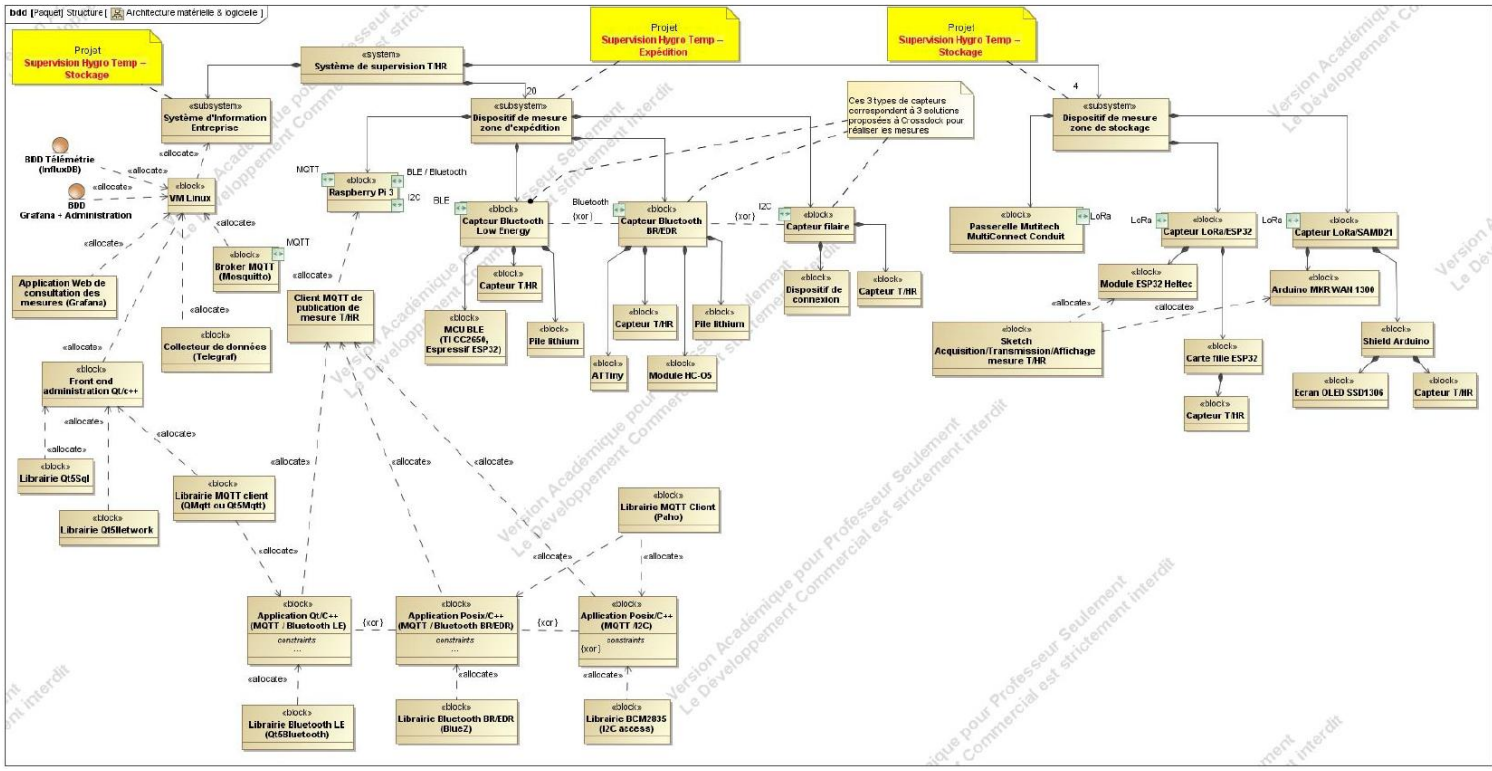
1. Dispositif de mesure – Zone d'expédition
2. Dispositif de mesure – Zone de stockage
3. Sous-système de Publication des mesures & Administration

Le projet décrit dans ce document porte sur le sous-système (1).



Architectures Matérielle & Logicielle

L'architecture matérielle et logicielle du système global est présentée ci-dessous. Ce diagramme rappelle sous forme de notes à quel projet se rapporte chacun des sous-systèmes.

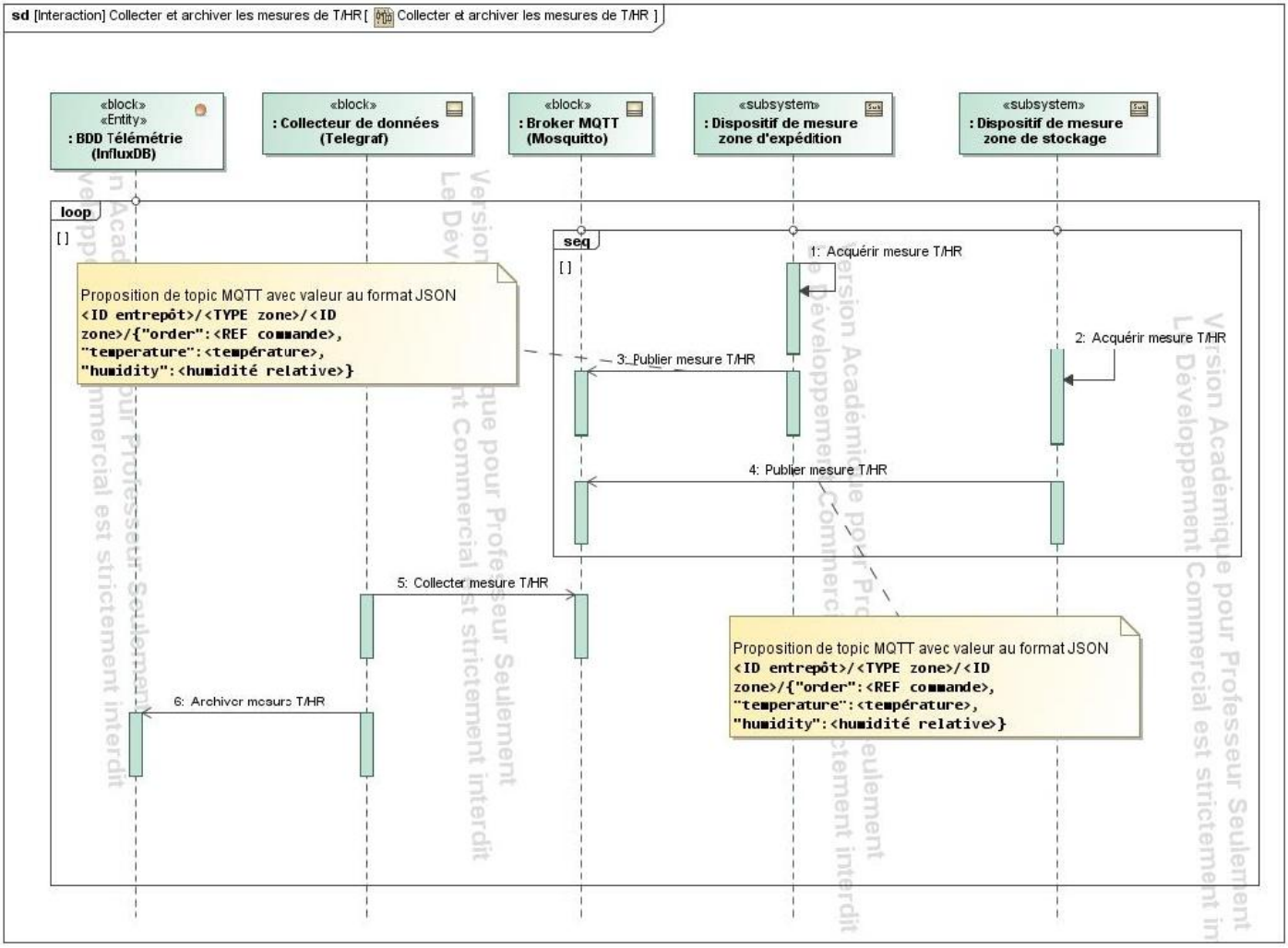


Scénarios des cas d'utilisation

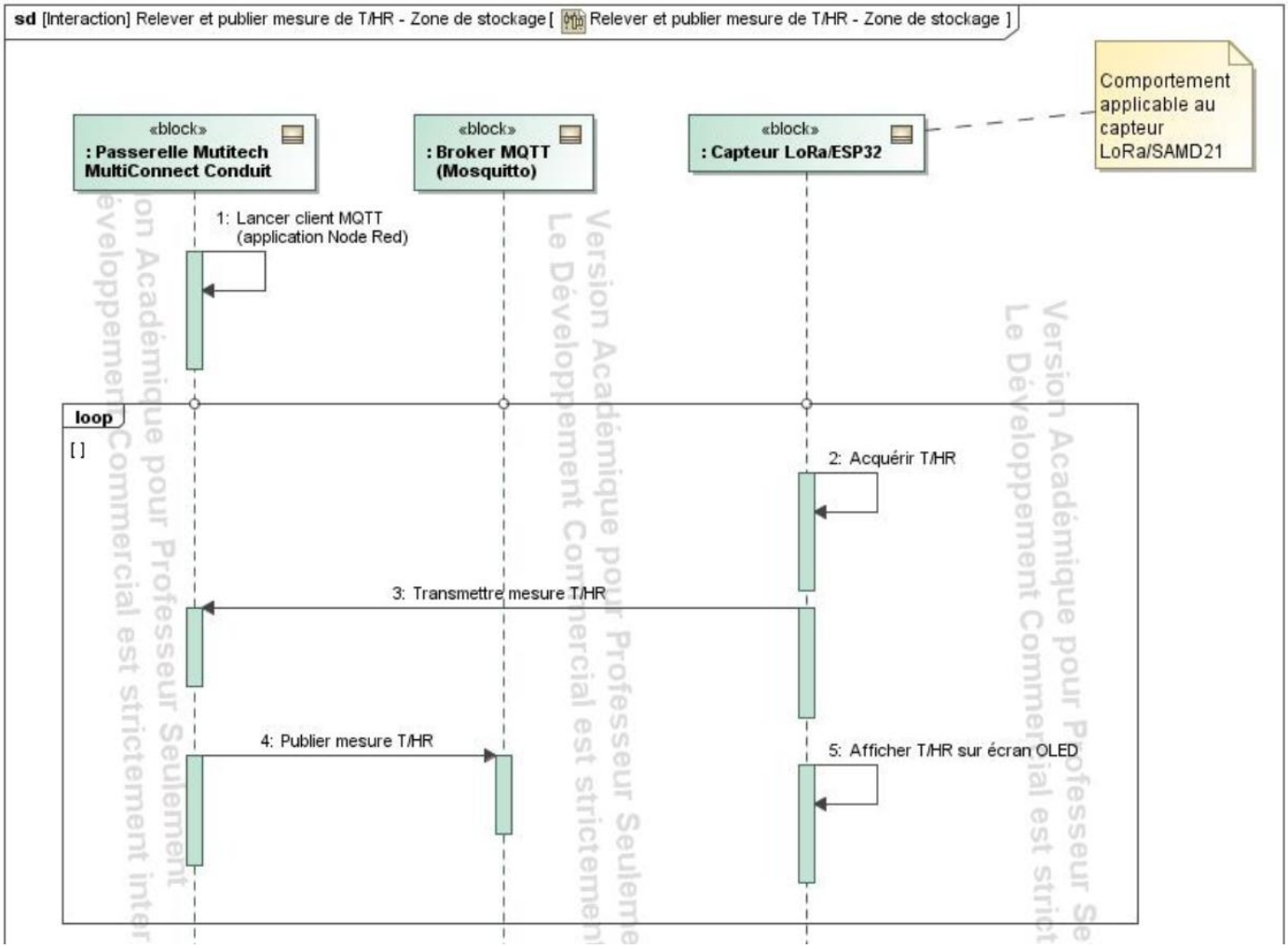
(Sous-forme de diagrammes de séquence SysML)

Ce diagramme de séquence illustre l'ensemble du processus de collecte et d'archivage des mesures de T/HR.

La publication sur le broker MQTT des mesures de T/HR issues du dispositif de la zone d'expédition est assurée par l'équipe projet "Supervision Hygro Temp - Expédition".



Ce diagramme représente le relever et la publication des mesures de température et d'hygrométrie dans la zone de Stockage.



Contraintes de réalisation

Contraintes financières (budget alloué):
Budget estimé de 2000€
L'entreprise CrossDock participe au financement du projet.

Contraintes de développement (matériel et/ou logiciel imposés, technologies utilisées):
La spécification, conception et codage seront modélisés.

Contraintes qualité (conformité, délais, ...):
Maintenable, maniable (ergonomie)

Contraintes de fiabilité, sécurité:
Les accès logiciels seront sécurisés.
La protection des données sera assurée (RGPD)

Matériels, Logiciels et Documentation

Matériels :
- PCs Windows/Linux
- Cartes ESP32-Heltec (ECs + IRs)
- Cartes Arduino MKR WAN 1300
- Passerelle LoRa (Multitech Multiconnect Conduit)
- Nano-ordinateur Raspberry Pi
- Composants et matériel de câblage
- Platine d'essai type Labdec (ECs+IRs)
- Appareils de mesure (oscilloscope, multimètre, analyseur logique)
- Breakouts pour prototypage (capteur T/HR SHT31)

Logiciels :
- Logiciel de modélisation SysML/UML : MagicDraw v7.02
- Logiciels de conception électronique : KiCad 5
- Logiciel de conception électronique Fritzing uniquement pour illustrer le prototypage rapide
- IDE Arduino + ajout gestionnaire cartes et bibliothèques ESP32/SAMD21
- Environnement de développement Qt5
- Serveur BDD pour télémétrie (InfluxDB) + Collecteur de données (Telegraf)
- Application web de visualisation de données (Grafana)
- Serveur BDD pour administration et dashboards Grafana (MySQL ou Postgre)
- Broker MQTT (Mosquitto)
- Librairies client MQTT (Paho et QMqtt/Qt5Mqtt)
- Librairie d'accès aux E/S de la raspberry Pi (bcm2835)

Documentation :
- Site de la section BTS SN mettant à disposition les différentes documentations.

Sommaire

Partie LE GALLO Joris

Introduction

Dans le cadre de ma deuxième année de BTS Systèmes Numériques Option Informatique et Réseaux au lycée Alphonse Benoit, j'ai été affecté au projet CrossDock.

Ce projet consiste à effectuer l'acquisition de la température et de l'hygrométrie dans deux entrepôts afin de permettre une traçabilité sur les bonnes conditions d'entreposage des produits CrossDock.

Ainsi, le système pourra fournir la possibilité d'être supervisée à distance afin de surveiller le seuil des températures et d'hygrométrie dans les entrepôts.

C'est de ce dernier point que je vais traiter durant ce projet.

Mon but est de récupérer les mesures sur le serveur MQTT et de les insérer dans une base de données en l'occurrence InfluxdB.

Apres, je l'ai retranscris à l'aide de l'application web Graf ana dans des Dashboard disponible par CrossDock et ses clients ce qui permettra la consultation et le suivi des mesures effectuées.

Et pour finir je devrai assurer la supervision du système global en créant des utilisateurs et faire une gestion des droits des utilisateurs du système à l'aide d'une application QT.

Schéma Synoptique

Voici un schéma simplifié du système CrossDock pour mieux comprendre le trajet de la chaîne d'information dans le projet.

Pour ma part je vais m'occuper de la partie "Traiter" et "Communiquer".

Les mesures de température et d'hygrométrie seront transmises par deux moyens différents vers deux Client MQTT. Par la suite ces deux clients communiqueront les informations à un Broker MQTT.

C'est à partir d'ici que j'interviens. L'agent Telegraf va collecter les mesures du Broker MQTT et va les insérer dans la base de données d'InfluxdB afin de les stocker en temps réel, c'est à dire que les mesures seront horodatées.

Apres cette étape l'application Web Grafana va afficher les informations sur le Web et plus particulièrement dans ce qu'on appelle des dashboards. Dans notre système il y aura deux dashboards. Un Dashboard "Administrateur" pour la société CrossDock et des autres Dashboards "Client" pour ses clients. Le Dashboard "Administrateur" permettra de consulter bien plus de mesures que les Dashboards "Client". Les Dashboards "Client" seront dédiés aux différents clients de la société Crossdock.

Le programme MQTT fx me permettra tout au long de ce projet de tester le fonctionnement du serveur MQTT.

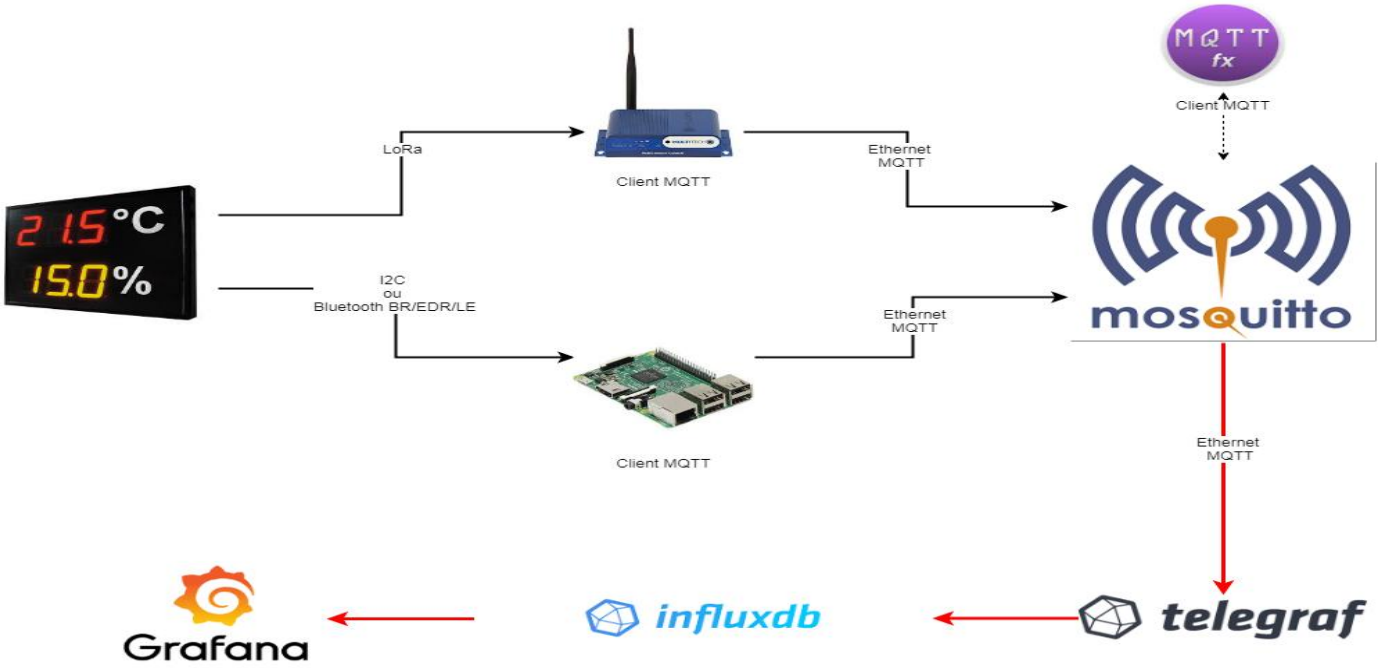
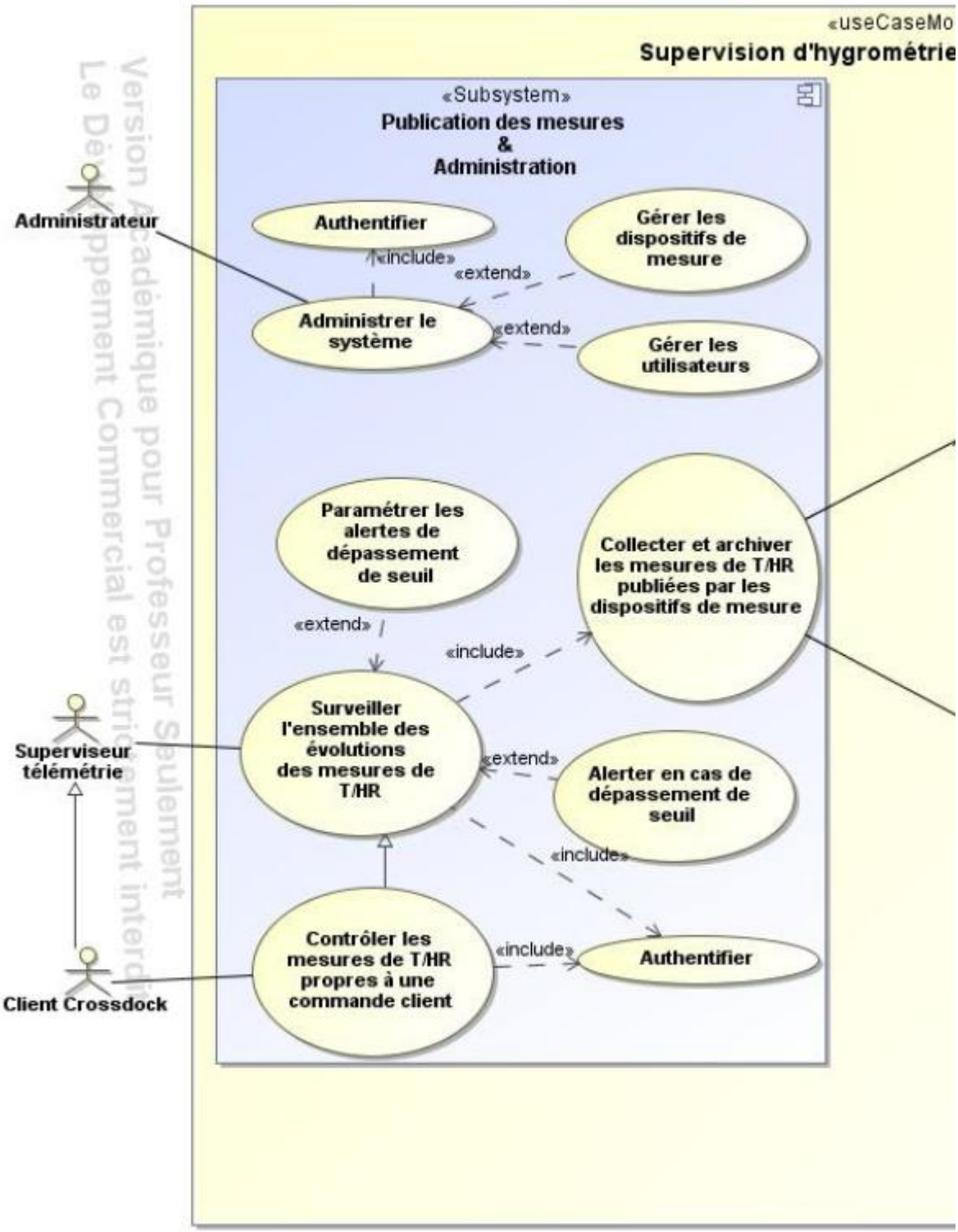


Diagramme de Cas d'Utilisation

Ce diagramme représente les cas d'Utilisations que je vais mettre en place, de la surveillance de l 'ensemble des mesures à l'administration du système.



Objectifs de ma partie personnelle

Numéro	Objectifs
1	Installer un Broker MQTT
2	Stocker les mesures dans une base de données
3	Afficher les mesures sur une application WEB
4	Administrer le système à l'aide d'applications QT

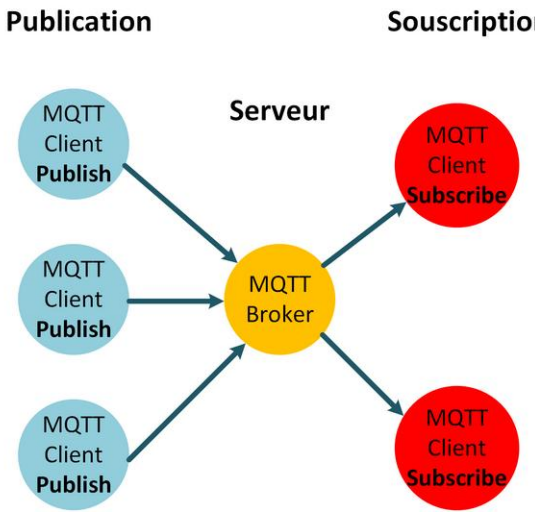
Situation avant la première revue de projet.

Phase d'Analyse et de Compréhension.

Au début du projet, je me suis lancé dans une phase de compréhension et d'analyse du projet.
Mon but est de remonter les mesures jusqu'à une application web en les stockant dans une base de données.

Nous allions utiliser le protocole MQTT pour remonter les mesures vers un Broker MQTT puis vers une base de données.

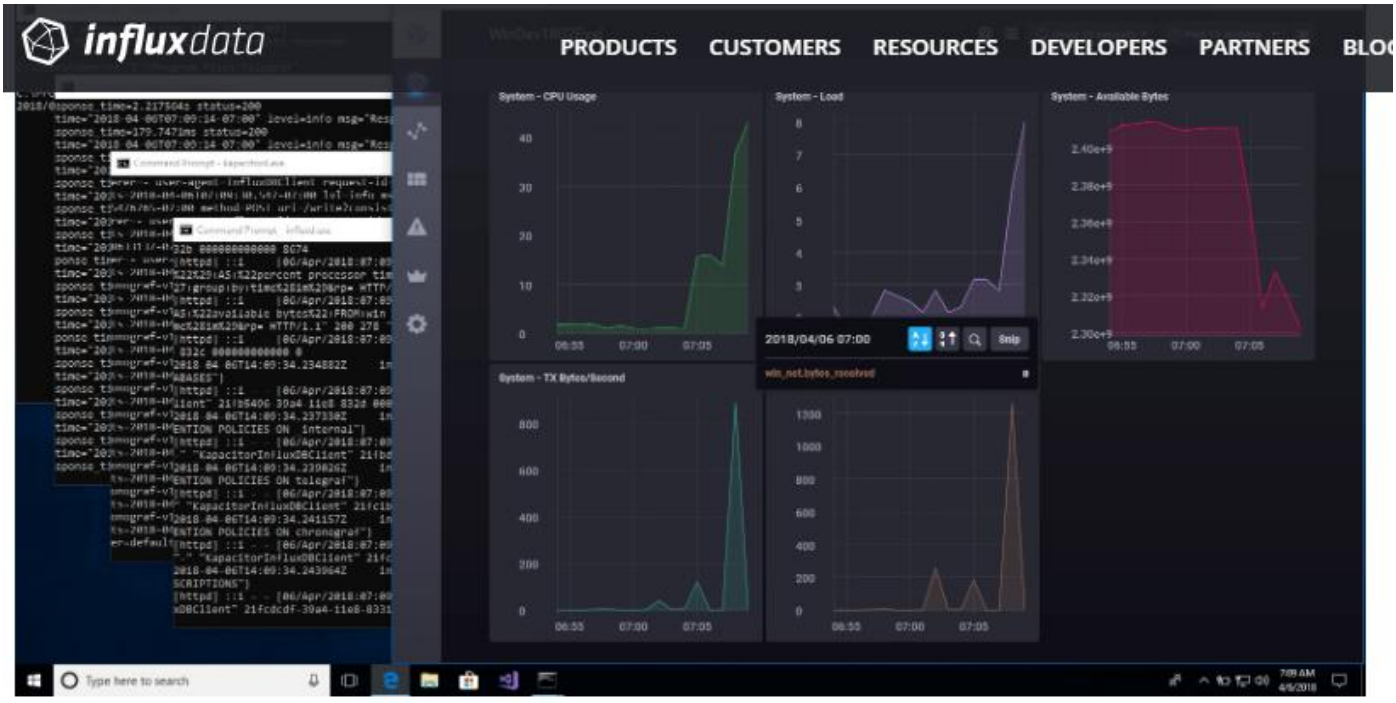
C'est pourquoi je me suis intéressé à ce protocole pour bien comprendre son fonctionnement ce qui me permettra par la suite d'installer le Broker MQTT et de bien configurer celui ci pour assurer la communication.



Ensuite, il était temps pour moi de commencer à installer le serveur qui contiendra, la base de données InfluxdB, l'application Web Grafana et le Broker MQTT qui seront situés sur le même poste.

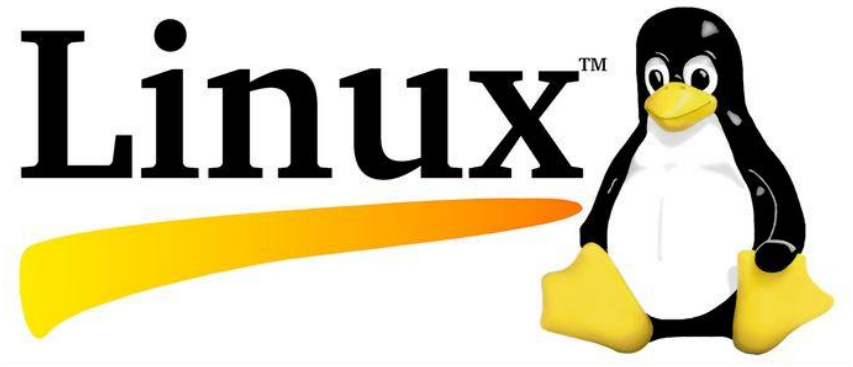
Je me suis demandé quel système d'exploitation serait le plus adapté pour ce projet.

C'est alors que j'ai appris que les développeurs de la base de données (InfluxdB) que j'allais utilisé ont indiqué que la version Windows de la base de données était une version expérimentale et qu'ils ne la recommandaient pas son utilisation en production.



> Every now and then we get requests from a developer using Windows, or a Windows-using participant at one of our workshops or events. Windows support is considered **"experimental"** for all components except Telegraf, but the full TICK Stack will still build and run on the platform, and we provide binaries for Windows on our downloads page. So while Windows is not an officially supported operating system and is **not recommended in production**, we wanted to provide some additional resources for those adventurous developers who still want to

Pour ne prendre aucun risque inutile je me suis naturellement tourné vers un système d'exploitation Linux



J'avais maintenant le choix entre l'utilisation d'une raspberry ou d'une machine virtuelle. Bien que mon professeur m'ait conseillé d'utiliser une raspberry pour sa petite taille et pour avoir un bien matérielle, j'ai tout de même décidé d'utiliser une machine virtuelle car on peut lui allouer plus de puissance et avoir un plus grand espace de stockage.

J'ai ainsi décidé d'utiliser VMware Player pour concevoir la machine virtuelle car il s'agissait du logiciel avec lequel j'ai appris à virtualiser.

J'ai ensuite choisi la distribution d'Ubuntu LTS 18.04 car elle me semblait compétitive par rapport aux autres et qu'il s'agit d'une version LTS (Support à Long Terme) ce qui est adapté au monde de l'entreprise car c'est une version qui est régulièrement mit à jour par le développeur afin de combler certaines failles de sécurité par exemple.

J'ai ensuite choisi la solution TIG (Telegraf, InfluxdB, Grafana) qui permettra de récupérer les informations du Broker MQTT, de les insérer dans une base de données, times series et de les retransmettre sur une application web.

J'ai ensuite commencer l'installation de la machine virtuelle avec Vmware Player Workstation 14, petite particularité j'ai installé cette machine ainsi que tous les logiciels qui vont suivre avec des mots de passes respectant la nouvelle loi européenne en matière de protection de données, la loi RGPD.

J'ai continué avec l'installation de la base de données InfluxdB qui me permettra de stocker les données d'une durée d'un an.

Commande :

```
apt-get install InfluxdB
```

J'ai poursuivi avec l'installation de l'agent Telegraf qui est un logiciel développé par les mêmes développeurs qu'InfluxdB et qui permet de récupérer les mesures du Broker MQTT et de les insérer dans la base de données.

Commande :

```
apt-get install Telegraf
```

J'ai ensuite installé l'application Web Grafana qui me permettra d'afficher les mesures stocker dans la base de données.

Commande :

```
echo 'deb https://packagecloud.io/grafana/stable/ubuntu/ > /etc/apt/sources.list.d/grafana.list

curl https://packagecloud.io/gpg.key | sudo apt-key add -

sudo apt update; sudo apt install grafana
```

J'ai ensuite créer une base de données ainsi qu'un utilisateur pour cette base. L'avantage d'InfluxdB, c'est que son langage est très similaire à celui de MySQL donc je ne fus pas déstastabilisé.

J'ai ainsi configuré Grafana pour qu'il puisse communiquer avec la base de données

Et j'ai ensuite conçu un dashboard de test pour vérifier la communication entre la base de données et l'application Web.

Il me restait à configurer la liaison entre le Broker MQTT, le collecteur de données Telegraf et la base de données InfluxdB.

Après la première revue de projet.

Installation du Broker MQTT

Très rapidement après la première revue de projet, j'ai installé le broker mqtt mosquitto, rien de bien compliqué il suffisait simplement d'utiliser la commande :

```
apt-get install mosquitto
```

Après ça je me suis informé sur les différentes commandes pour démarrer le broker mqtt.

Pour démarrer le Broker MQTT :

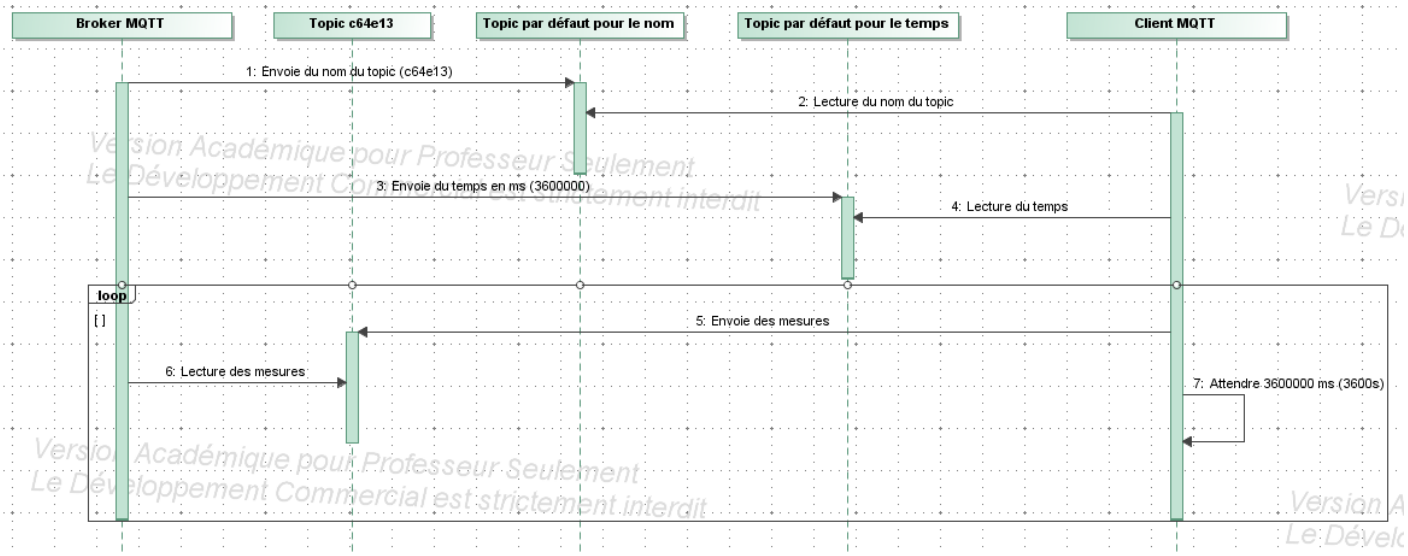
```
mosquitto
```

Le Broker MQTT fonctione d'une façon particulière, il utilise ce qu'on appelle des topics. Il s'agit d'une sorte de chemin ou les clients MQTT vont publier les mesures. Le Broker MQTT va ensuite pouvoir s'abonner a ce topic pour lire les données. Cependant le Broker MQTT peut lui aussi publier sur ces topics pour que les clients puissent recevoir des informations.

Fonctionnement des topics dans le système

Dans le système nous avons une utilisation un peu spécial des topics, pour avoir plus flexibilité en cas de panne ou de changement d'appareils coté Client MQTT, on utilise un topic par défaut. Le Broker MQTT va publier sur le topic par défaut le nom d'autres topics spécifiques a chaque client MQTT

Voici un diagramme de séquence pour expliquer ce procédé :



Le Broker MQTT va publier sur le topic par défaut le nom du topic (ce3801) qui correspond au 6 derniers caractères du l'adresse mac du client MQTT. Le Broker va ensuite publier sur un autre topic par défaut le temps avec lequel les clients MQTT doivent publier les mesures.

Le client MQTT va ensuite lire ces 2 valeurs, dans notre exemple il s'agit du topic /crossdock/c64e13 et de 3600 secondes. Le client va alors publier ses mesures toutes les heures sur le topic /crossdock/c64e13.

Voici la commande pour que le Broker MQTT puisse publier sur un topic :

```
mosquitto_pub -t /crossdock/sensors -m /crossdock/ce3801
```

Avec cette commande le Broker MQTT publie sur le topic par défaut "/crossdock/sensors" un topic "/crossdock/ce3801"

Le serveur MQTT publiera le topic d'une manière spéciale pour que les clients MQTT puissent comprendre a qui le serveur MQTT communique. Cette manière est la suivante les topics seront écris de la manière suivante : /crossdock/deuxiemepartiedel'adressemac

exemple pour mon ordinateur :

/crossdock/7470da

Configuration de Telegraf

Après avoir fait la liaison entre Grafana et InfluxdB il fallait maintenant compléter le puzzle en assurant la liaison entre le Broker MQTT et InfluxdB voici un petit rappel pour illustrer mon propos

C'est pourquoi j'ai tout d'abord commencer par configurer l'agent Telegraf qui permet de collecter les données du Broker MQTT et de les insérer dans la base de données.

Il fallait que je rentre dans le dossier de configuration de Telegraf et que j'y ajoute deux plugins.

Un plugin en input (entré) qui se nomme Mqtt_consumer qui va se connecter au Broker MQTT et plus particulièrement à ce qu'on appelle des topics dans le Broker MQTT. Ces topics sont des chemins ou sont postés les mesures.
Le plugin va faire en sorte que Telegraf s'abonne à certains topics pour récupérer par la suite automatiquement les informations.

Voici un exemple du plugin situait dans le fichier de configuration situait lui même sur le serveur.

```
[[inputs.mqtt_consumer]]

  ## MQTT broker URLs to be used. The format should be scheme://host:port,
  ## schema can be tcp, ssl, or ws.
  servers = ["tcp://192.168.4.65:1883"]

  ## QoS policy for messages
  ##   0 = at most once
  ##   1 = at least once
  ##   2 = exactly once
  ##
  ## When using a QoS of 1 or 2, you should enable persistent session to allow
  ## resuming unacknowledged
  qos = 0

  connection_timeout = "30s"


  ## Topics to subscribe to
  topics = [
    "/crossdock/sensors",
    "/crossdock/louis",
    "/crossdock/clement",
```

Adresse du Broker MQTT

Possibilité de faire un système d'accusé de réception, ce n'est pas le cas ici

Exemple de topics

```
]

# if true, messages that can't be delivered while the subscriber is offline
# will be delivered when it comes back (such as on service restart).
# NOTE: if true, client_id MUST be set
persistent_session = false
client_id = ""

## username and password to connect MQTT server
username = "legallo"
password = "8Byzw2sl!"

## Optional TLS Config

## Data format to consume.
## Each data format has its own unique set of configuration options, read
## more about them here:
## https://github.com/influxdata/telegraf/blob/master/docs/DATA_FORMATS_INPUT.md
data_format = "json"
```

Identifiant et mot de passe pour se connecter au Broker MQTT

Format avec lequel les mesures doivent être envoyées

Il fallait à présent installer un deuxième plugin en output (sorti) qui se nomme influxdb, il est un peu plus simple à comprendre et à configurer que le précédent il suffit simplement de rentrer le nom de la base de données ainsi que l'adresse et les identifiants de celle-ci.

Voici un exemple du plugin influxdb :

```
[[outputs.influxdb]]
  database = "crossdock"
  precision = "s"
  urls = [ "http://192.168.4.65:8086" ]
  username = "telegraf"
  password = "8Byzw2sl!"
```

Nom de la base de

Adresse de la base de données
Identifiant et mot de passe de la base de

Analyse d'une trame MQTT

Afin de vérifier le bon fonctionnement du Broker MQTT, j'ai décidé d'analyser les trames MQTT qui arrivaient sur le serveur avec le logiciel Wireshark. Il s'agit d'un analyseur de packet qui dans notre cas va intercepter les messages MQTT pour vérifier leur exactitude.

Ce logiciel est utilisé dans le dépannage et l'analyse des réseaux informatiques, le développement des protocoles et

l'éducation.

Exemple de ce qu'on peut recevoir avec le logiciel :

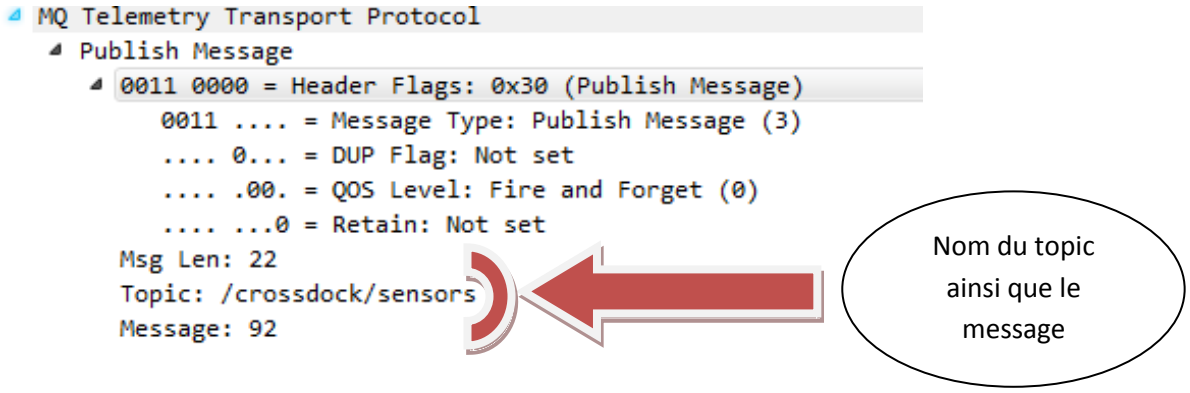
70	163.999113	192.168.4.19	192.168.4.65	MQTT	79 Subscribe Request
71	163.999455	192.168.4.65	192.168.4.19	MQTT	60 Subscribe Ack
72	164.198736	192.168.4.19	192.168.4.65	TCP	54 49523 → 1883 [ACK] Seq=56 Ack=12 Win=256 Len=0
73	168.900523	192.168.4.19	192.168.4.65	MQTT	78 Publish Message
74	168.907606	192.168.4.65	192.168.4.19	MQTT	78 Publish Message

Il s'agit de plusieurs trames, quand on publie ou quand on s'abonne on arrive à intercepter les messages avec le logiciel.

J'ai envoyé à l'aide d'MQTT.fx le nombre 92 sur le topic /crossdock/sensors

On reçoit alors comme ci-dessus un "Publish Message" avec le protocole MQTT

Quand on click sur "Publish Message" on obtient une multitude d'information voir ci dessous :

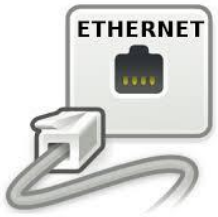


On obtient différente information comme le nom du Topic, le message en lui même ainsi que différentes autres informations comme le QOS ou la taille du message.

Le message est ici bien envoyé.

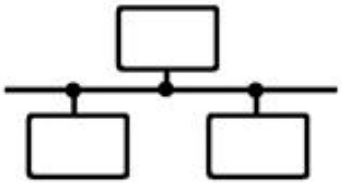
Ethernet

Bien que l'on utilise le protocole MQTT dans le système, nous utilisons aussi une connexion Ethernet comme c'est le cas dans un réseau informatique classique.

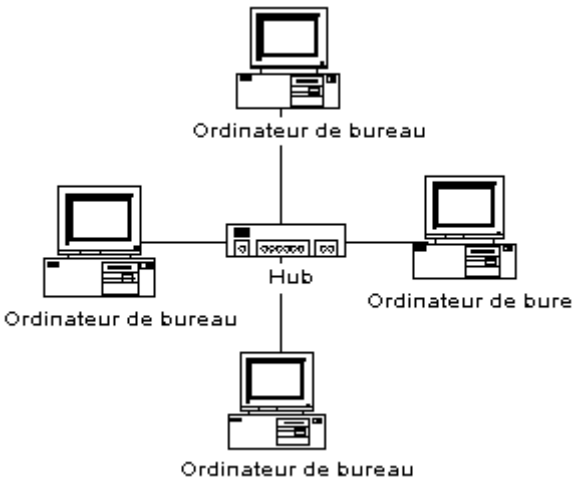


Ethernet est un protocole de réseau local à commutation de paquets, ce protocole a quelques particularités comme le fait de ne pas garantir la bonne réception des paquets.

Ce protocole a été conçu à l'origine pour une topologie en bus. Tout le monde peut envoyer, reçevoir les trames et voir les collisions dans le réseau.



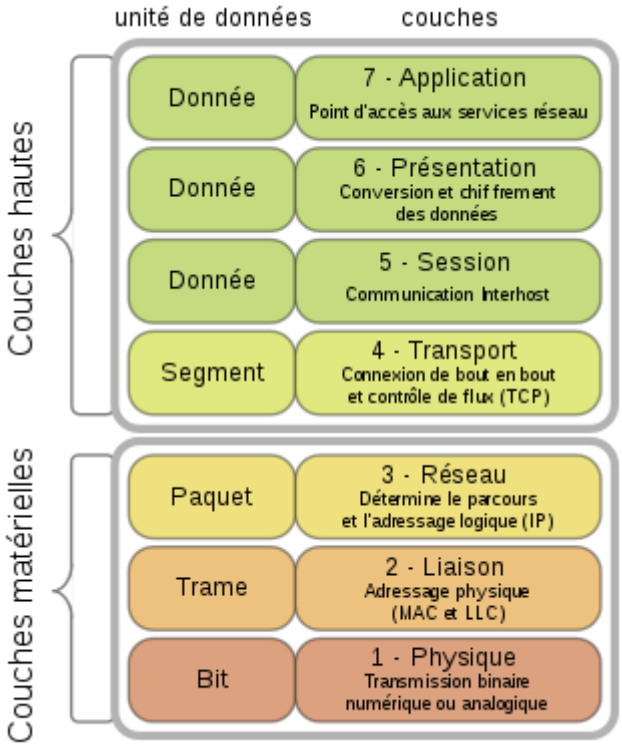
Il fut ensuite adapté pour une topologie physique en étoile, chaque périphériques étant raccordées à des hubs ce qui ne change pas la nature d'Ethernet qui fonctionne toujours de la même façon.



Le plus avantage de l'Ethernet est la possibilité d'exécuter à grande vitesse à faible coût.

Il permet de communiquer à 1 Gigabits par seconde et jusqu'a 100 Gigabits par seconde dans certains cas notamment en utilisant la fibre optique. Les premières versions d'Ethernet avaient certaines contraintes comme le fait de pouvoir communiquer à 2.5km maximum. L'introduction des hubs a permis de communiquer à 40km et en full duplex. C'est à dire que les périphériques peuvent envoyer et recevoir des paquets en même temps.

Ethernet se situe dans les couches 1 (physique) et 2 (liaison de données) du modèle OSI qui est une norme de communication des réseaux qui décrit les fonctionnalités nécessaires à la communication et l'organisation de ses fonctions.



Source : Wikimedia

Ethernet utilise des connecteurs RJ45 qui sont souvent ce qu'on appelle des pairs torsadées.

Une paire torsadée est une ligne symétrique formée de deux fils conducteurs enroulés en hélice l'un autour de l'autre. Cette configuration a pour but principal de limiter la sensibilité aux interférences et la diaphonie dans les câbles multipaires. 

Conception de deux Dashboard pour Crossdock

En attendant d'avoir des dashboards dynamique qui se créaient en fonction de qui aura accès aux mesures j'ai décidé de concevoir deux types de dashboard différent pour Crossdock uniquement.

Le premier panel, je l'ai imaginé sous forme de tableau avec d'un côté les graphiques et l'évolution de la température de l'autre côté une gauge avec vérification du seuil pour chaque mesures.

Voici le résultat :

L'inconvénient de cette proposition c'est la taille des dashboards, il faut descendre la page web pour avoir accès aux autres zones ce qui ne donne pas un aperçu global des mesures.

C'est pourquoi j'ai décidé de faire un deuxième panel de dashboards avec uniquement des Gauges cependant encore une fois il est difficile d'avoir un aperçu global de toutes les mesures.

Autre inconvénient de ces deux panels, toutes les mesures sont répertoriés on ne peut donc pas montrer ses dashboards aux clients de Crossdock car ils ne sont pas concernés.

Ces dashboards feront office de dashboards "Administrateur".

C'est pourquoi il est essentiel de créer une application QT qui va gérer la création de dashboards ainsi que des droits de qui aura accès aux dashboards.

Les dashboards créer par l'application Qt seront les dashboards "Client".

Objectifs de l'application qt

La dernière tâche que j'ai été chargé de faire, était une application Qt creator ayant pour but d'administrer le système dans son ensemble.

La première fonctionnalité était de créer des dashboards automatiquement dans Grafana et d'autoriser ces dashboards seulement aux clients de crossdock concernés par les mesures en question.

Ainsi j'ai décidé de rajouter une deuxième fonctionnalité nécessaire pour le bon fonctionnement de l'application. La création d'utilisateurs qui correspondraient à des clients Crossdock depuis une autre application Qt.

La troisième de fonctionnalité était de gérer l'envoie du nom des topics et du timer pour chaque Client MQTT dans le système.

Conception de l'application QT Creator

J'ai commencé à concevoir une application QT qui permettrait de d'administrer le système dans sa globalité.

Elle permettrait aux clients de crossdock d'avoir accès uniquement a leurs mesures et pas à celles des autres. Pour se faire il faut que l'application puisse avoir différentes fonctionnalités.

Dans un premier temps Crossdock rentre les informations sur les emplacements des colis et de quel client il s'agit. L'application va ensuite envoyer des requetes API à grafana pour créer une dashboard avec les mesures liés à la zone en question.

Autre fonctionnalité il faudrait que l'application puisse créer de nouveaux utilisateurs pour les nouveaux clients de crossdock.

Egalement sil y a un changement de raspberry. Il faut pouvoir indiquer si une nouvelle raspberry a été installée pour ensuite gérer les topics.

Il faut que l'application puisse aussi envoyer les bons topics au Client MQTT

Pour le moment j'ai essayé différentes méthodes afin de commencer, j'ai essayé d'envoyer des requetes URL pour pouvoir accéder à Grafana et par la suite créer des utilisateuurs ou dashboards malheureusement j'ai utilisé une méthode Qt obsolète sous QT5.

Voici l'exemple du programme :

```
QNetworkAccessManager *manager = new QNetworkAccessManager(this);

QUrl adresse = QUrl("http://192.168.4.65:3000");
QDesktopServices::openUrl(adresse);

QUrl url("http://192.168.4.65:3000");
QNetworkRequest request(url);

request.setHeader(QNetworkRequest::ContentTypeHeader, "/login HTTP/1.1");

QUrl params;
params.addQueryItem("login", "legallo");
params.addQueryItem("password", "7Nuex3qm");
manager->post(request, params.encodedQuerry());
```

On peut voir que j'ouvre une page Web pour accéder à Grafana et en essayant de rentrer les identifiants avec la méthode "addQueryItem" et qui m'aurait permis par la suite de créer des utilisateurs. Malheureusement ça ne fonctionne pas car celle-ci est obsolète sous qt5.

Authentifications HTTP (Basic vs. Bearer)

C'est ainsi que je me suis rendu compte que l'on pouvait se connecter différemment à Grafana depuis une application Qt en utilisant l'Authentifications Basic ou Bearer.

Dans la documentation de Grafana en fonction de ce que l'on veut faire (créer des utilisateurs ou consulter des dashboards) on doit utiliser soit l'authentification Basic soit l'authentification Bearer.

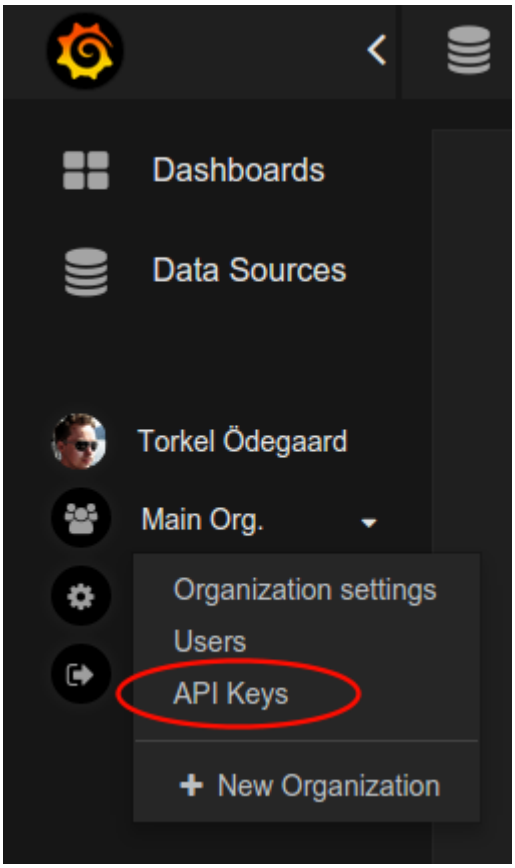
Dans le contexte d'une transaction HTTP, l'authentification d'accès de base est un procédé permettant à un agent d'utilisateur HTTP (par exemple, un navigateur Web) de fournir un nom d'utilisateur et un mot de passe lors d'une requête.

Dans l'authentification HTTP de base, une demande contient un champ d'en-tête de la forme Autorisation: Basic <credentials>, où credentials correspond au codage en base64 de l'id et du mot de passe joints par un simple signe deux-points (:).

L'authentification de support (également appelée authentification par jeton) est un schéma d'authentification HTTP qui implique des jetons de sécurité appelés jetons de support. Le nom «Authentification du porteur» peut être compris comme «donne accès au porteur de ce jeton». Le jeton du porteur est une chaîne cryptique, généralement générée par le serveur en réponse à une demande de connexion. Le client doit envoyer ce jeton dans l'en-tête Authorization lorsqu'il envoie des demandes aux ressources protégées:

Autorisation: porteur <jeton>

La méthode Bearer utilise une API Keys que l'on peut créer directement sur le site ou via un code.



Application Qt pour créer des utilisateurs

J'ai conçu une application pour créer des utilisateurs pour les clients de Crossdock, cela permet de gérer les droits de chaque utilisateurs. Un utilisateur aura accès aux mesures qui le concerne et pas à toutes les mesures.

```cpp
#include "mainihm.h"
#include "ui_mainihm.h"

#include <QtNetwork/QNetworkRequest>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonObject>

MainIHM::MainIHM(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::MainIHM)
{
    ui->setupUi(this);


    nam.connectToHost("192.168.4.65", 3000);


    QNetworkRequest nr;
    nr.setUrl(QUrl("http://192.168.4.65:3000/api/admin/users"));
    nr.setRawHeader(QByteArray("Authorization"), QByteArray("Basic YWRtaW46OEJ5encyc2wh"));
    nr.setRawHeader(QByteArray("Accept"), QByteArray("application/json"));
    nr.setRawHeader(QByteArray("Content-Type"), QByteArray("application/json"));

    QString data = "{\
                    \"name\":\"louis\",\
                    \"email\":\"louisgauthier@hotmail.com\",\
                    \"login\":\"louis\",\
                    \"password\":\"louis\"\
                   }";

    QByteArray ba;

    ba += data;

    nrep = nam.post(nr, ba);

    connect(nrep, &QNetworkReply::finished, this, &MainIHM::onNetworkReplyFinished);


}

void MainIHM::onNetworkReplyFinished(){

    QByteArray response = nrep->readAll();
    ui->tedtReply->append(response);

}
```

Différentes librairies

Connection à Grafana

Connection à Grafana

Création de l'utilisateur Louis

Utilisation de la méthode POST

Requêtes HTTP (GET vs. POST) en général et dans le contexte de l'API Grafana

Pour pouvoir s'authentifier, créer des utilisateurs ou des dashboards dans Grafana depuis une application Qt, on doit utiliser des requêtes HTTP. Soit la requête GET, soit la requête POST.

En Get, les données que l'on veut insérer sont rajoutés à l'URL derrière un attribut action que l'on représente par un "?".

Exemple : https://fonts.googleapis.com/css?family=Roboto+Mono:400,700

Ce qui se situe après le "?" sont des paramètres insérés avec la requête HTTP GET.

En Post, les données sont incluses dans un formulaire et ne sont pas affichés dans l'URL

Exemple Structure d'un formulaire **:**

```
POST /api/admin/users HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "name":"User",
  "email":"user@graf.com",
  "login":"user",
  "password":"userpassword"
}
```

On utilise pas la requête HTTP GET et la requête HTTP POST pour les mêmes raisons.

En effet, en GET les paramètres sont visibles pour tout le monde dans l'URL et restent dans l'historique du navigateur. Dans notre cas il n'est pas intéressant pour la création des utilisateurs et de leurs mots de passes.

En POST, Les données ne sont pas enregistrées dans le navigateur, ne sont pas visibles. Cette méthode est plus adapté pour le transfert de mot de passe, c'est pour ça que j'ai utilisé celle-ci dans l'application Qt pour créer des utilisateurs.

Pour ne pas faire d'erreurs, la documentation Grafana nous indique quelle requête utilisée pour chaque action que l'on souhaite faire depuis une application de type Qt par exemple.

Si je souhaite créer un dashboard, on peut voir que la documentation nous propose directement la requête POST.

```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

```
{
  "dashboard": {
    "id": null,
    "uid": null,
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0
  },
  "folderId": 0,
  "overwrite": false
}
```

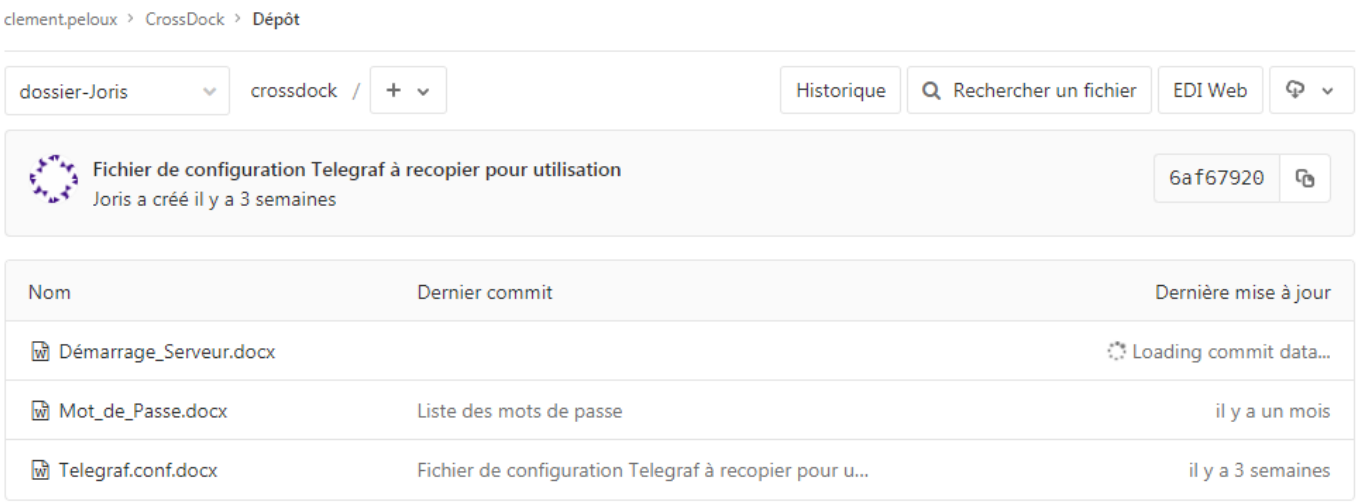

On peut aussi remarquer que le type d'authentifications est le mode Bearer.

Gitlab

Afin de faciliter la compréhension en vue d'une installation ou d'une réinstallation du système par l'entreprise Crossdock. Nous avons mis en place un Gitlab commun aux deux équipes du projet Crossdock.

Gitlab nous permet de rassembler tous les codes, applications et fichiers de configuration de l'ensemble du projet sur Internet.

Voici à quoi ressemble l'un des dossiers du projet :



Ce projet reste un prototype est pourra être installer ou améliorer par un technicien travaillant chez Crossdock c'est pourquoi chaque membres déposent régulièrement leurs travaux.

Liste des mots de passe

L'entreprise Crossdock peut être amener à confier le projet à un technicien afin de le compléter, de le modifier ou de l'utiliser.

C'est pourquoi afin de lui faciliter l'utilisation de la machine virtuelle, du serveur Ubuntu ainsi que de la base de données et de Grafana. J'ai conçu un document répertoriant tous les noms d'utilisateurs et des mots de passes sur le serveur.
De plus, ce document comporte une annexe avec quelques explications sur la localisation des mots de passes.

Ce document permettra de faciliter la tâche de quelqu'un qui souhaiterait modifier le serveur.

Liste des mots de passes

Nom de la machine virtuelle = crossdockserver
Mot de passe machine virtuelle = aucun

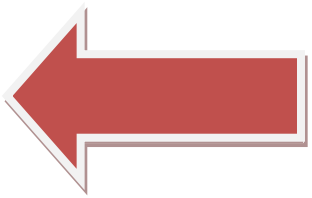
Nom du serveur Ubuntu = crossdockserver
Mot de passe = 8Byzw2sl!

Nom de l'utilisateur par défaut = legallo
Mot de passe = aucun

Nom de l'administrateur = root
Mot de passe = 8Byzw2sl!

Nom de la Base de Données = crossdock
Mot de passe = aucun

Nom de l'administrateur Grafana = admin
Mot de passe = 8Byzw2sl!



Liste des mots de passes

Annexes

Nom de la machine virtuelle = Lors du lancement de la machine virtuelle VMware Workstation 14

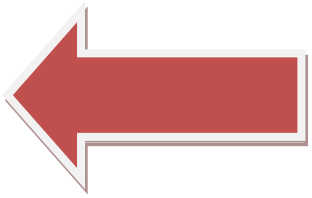
Nom du serveur Ubuntu = Lors du lancement du serveur, le premier mot de passe à rentrer

Nom de l'utilisateur par défaut = Dans un terminal après le lancement du serveur (peu de droits)

Nom de l'administrateur = Dans un terminal avec la commande sudo su (tous les droits)

Nom de la Base de Données = Dans InfluxdB, le nom de la base de données (use crossdock)

Nom de l'administrateur Grafana = En tapant l'adresse du serveur /3000 pour accéder à l'interface de Grafana



Annexes

Guide de démarrage de la machine virtuelle et du Broker MQTT

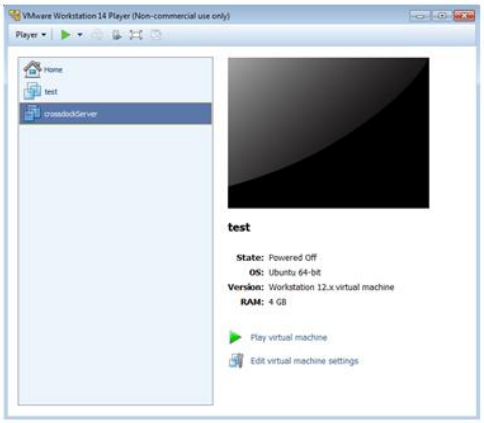
Tout comme les mots de passes, l'entreprise Crossdock sera surement amené à redémarrer le serveur c'est pourquoi j'ai fais un tutoriel très détaillé pour redémarrer ou démarrer la machine virtuelle, le serveur Ubuntu et le reste des applications très rapidement.

Voici un petit extrait de ce tutoriel :

I/ Démarrage de la machine virtuelle

1) Ouvrir VMware Workstation 14

2) Selectionner le projet "crossdockserver"



3) Appuyer sur "Play virtual Machine"

4) Selectionner l'utilisateur "crossdockserver"

Conclusion

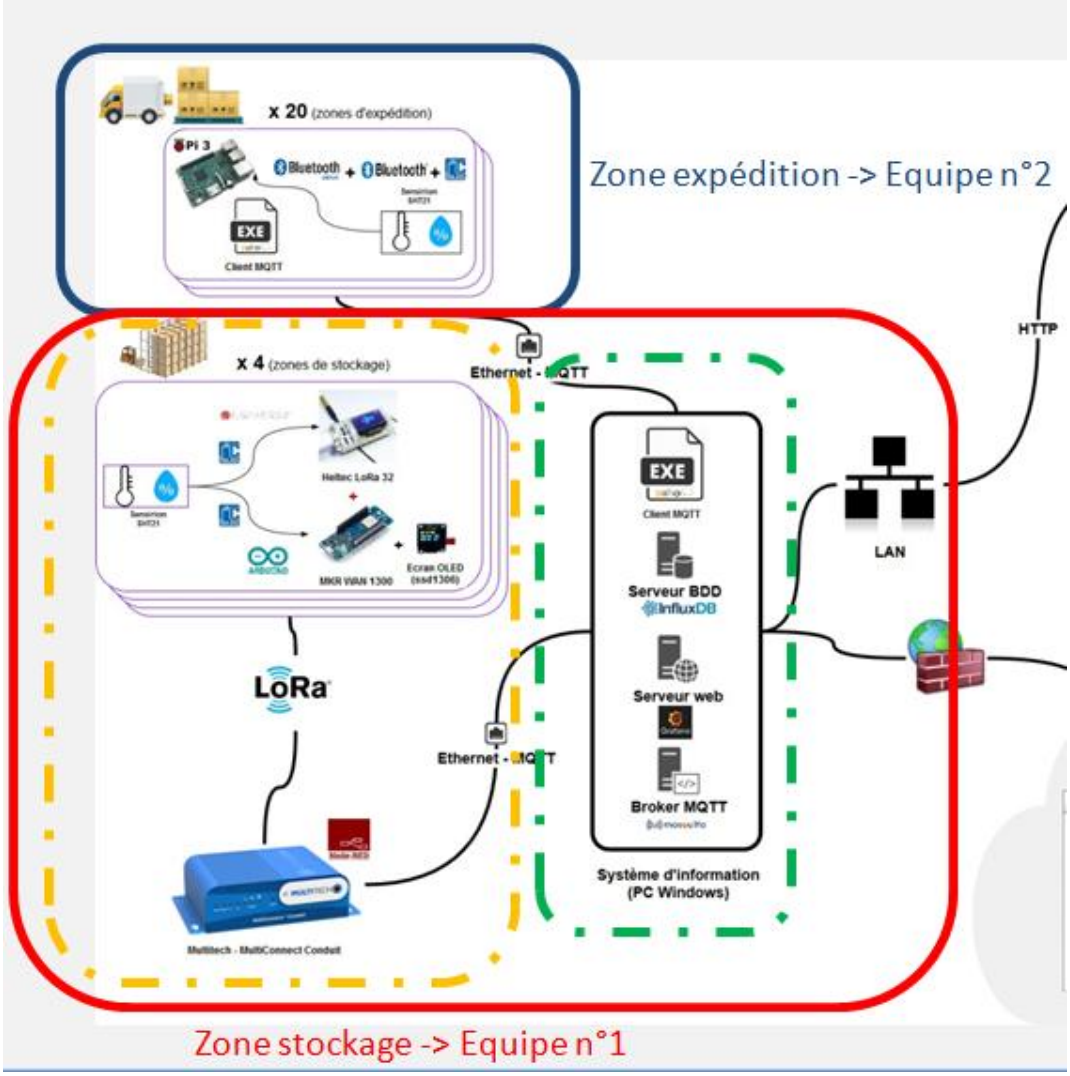
Pour conclure, le système est opérationnel, les mesures sont correctement stocker et retransmises dans des dashboards. Crossdock pourra vérifier la traçabilité des mesures. Au niveau de l'administration du système, on peut s'authentifier et créer des utilisateurs, il reste à créer une application pour créer des dashboards automatiquement en utilisant la même méthode que pour créer des utilisateurs.

Partie Personnelle ~ Etudiant n°2 IR2 ~

Peloux Clément

Ma partie au sein de ce projet va consister à coder un « sketch » pour les deux modules : carte heltech ESP 32 et l'arduino MKR WAN 1300 qui seront tous deux reliés à un capteur de température et d'humidité.

Mon objectif sera d'acquérir ces données, pour ensuite les afficher sur un écran OLED directement relié sur les deux cartes, et, enfin les retransmettre en LORA à la passerelle multitech multiconnect, qui elle sera reliée par câble ethernet au système d'information. Par la suite je vais devoir installer / configurer **la passerelle Multitech Multiconnect** qui assure la réception des données des cartes ESP 32 et MKR WAN pour ensuite les transmettre au serveur base de données via une passerelle LORA.



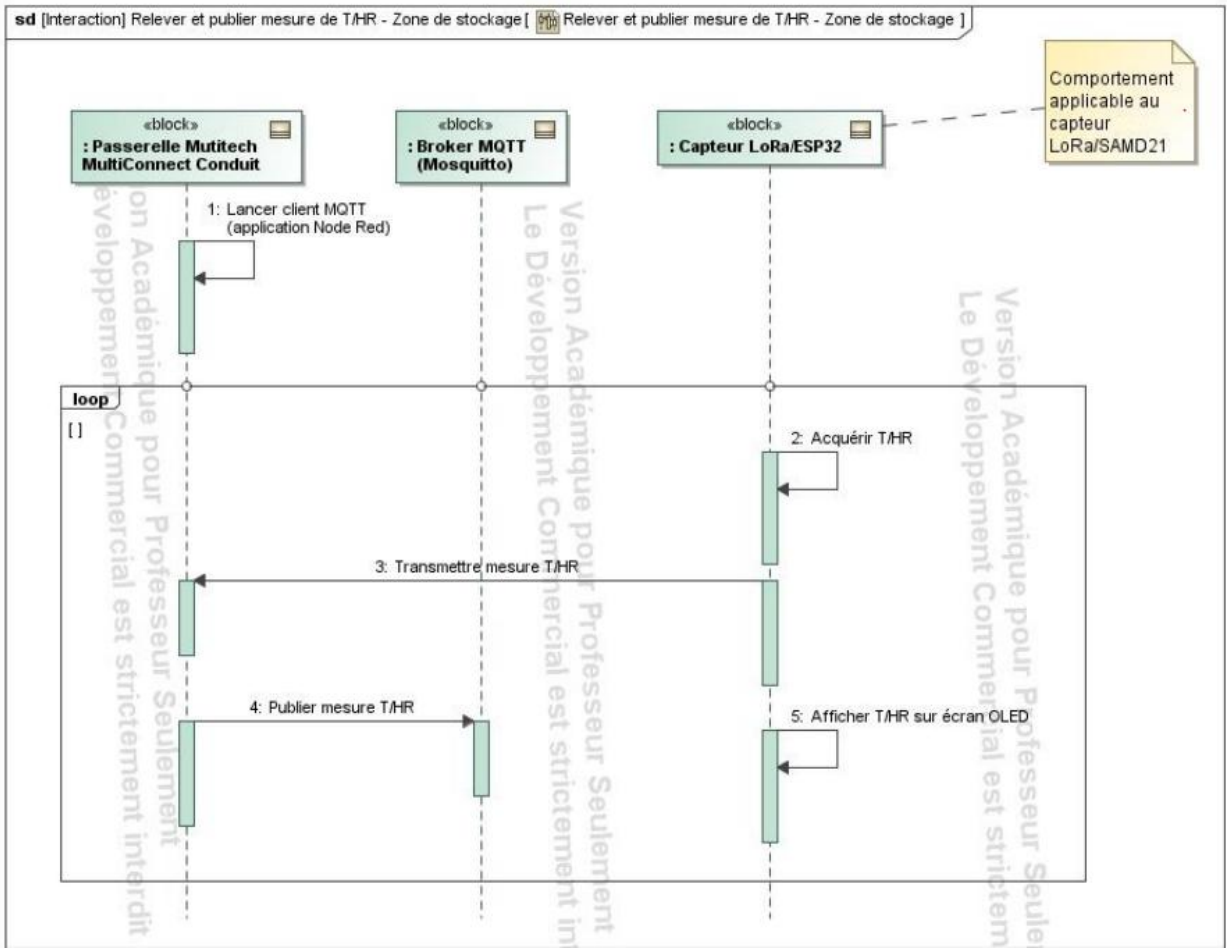
rouge : Zone de stockage

orange : Ma partie

vert : la partie de mon coéquipier

Suite à cela, j'ai réalisé un diagramme de séquences : « relevé , publication des mesures – Zone de stockage »

2.1.3.5 Relever et publier mesures de T/HR – Zone de stockage

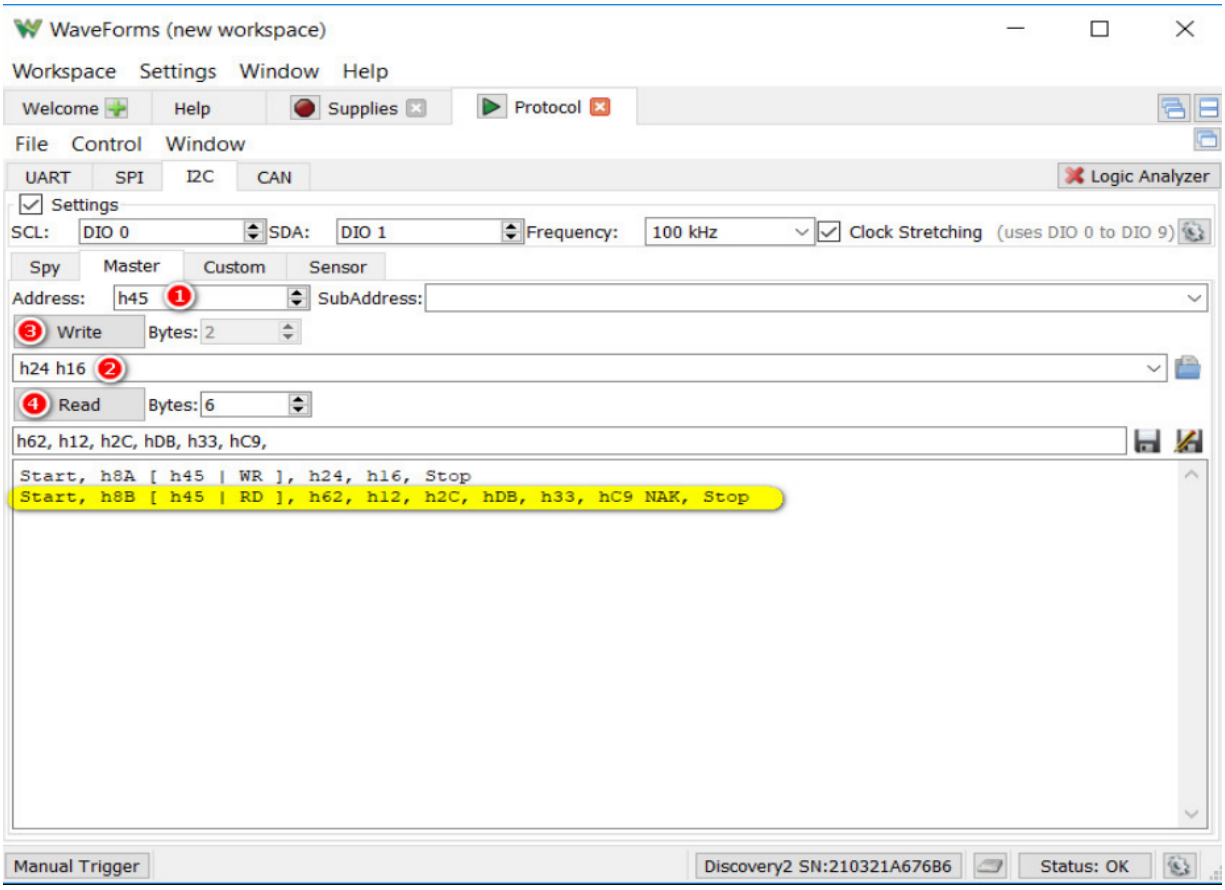


Ce diagramme me permet de mieux comprendre comment se fait l'acquisition des mesures et comment elles sont transmises à la passerelle multitech multiconnect. Ce diagramme de sequence correspond à l'acquisition des mesures de temperature et d'hydrométrie. On peut voir que le client MQTT doit être lancer en premier, par la suite on fait l'acquisition de la température et de l'humidité. Ces deux étapes bouclent automatiquement en cas d'échec. Le capteur ESP32 communique ses données à la passerelle multitech puis la passerelle multitech publie ces mesures sur le broker MQTT. En parrallèle, la carte esp32 affiche aussi en boucle la température et l'humidité sur l'écran OLED integer.

Ce qui a été mis en œuvre – Revue de Projet 1

Le travail effectué pour la revue de projet n°1 a donc été de comprendre le capteur SHT31 , de savoir comment et sous quelle forme les mesures étaient envoyées. Donc pour le coup, j'ai étudié ce capteur à l'aide de waveforms (digilent) pour analyser comment les trames étaient envoyées.

Analog Discovery	Breakout SHT31
V+	Vin et AD
Masse	Gnd
DIO 0	SCL
DIO 1	SDA



```
h62, h12, h2C, hDB, h33, hC9,

Start, h8A [ h45 | WR ], h24, h16, Stop
Start, h8B [ h45 | RD ], h62, h12, h2C, hDB, h33, hC9 NAK, Stop
```

Donc pour voir comment fonctionnait ce capteur, j'envoyais en écriture **« h24,h16 »** ce qui correspondait en vérité à :

Condition		Hex. code	
Repeatability	Clock stretching	MSB	LSB
High	enabled	0x2C	06
Medium			0D
Low			10
High	disabled	0x24	00
Medium			0B
Low			16
e.g. 0x2C06: high repeatability measurement with clock stretching enabled			

On peut voir ici qu'on envoie comme signal h24 h16 : on désactive le clockstretching et on se met en « low repeatability » = LSB = 16

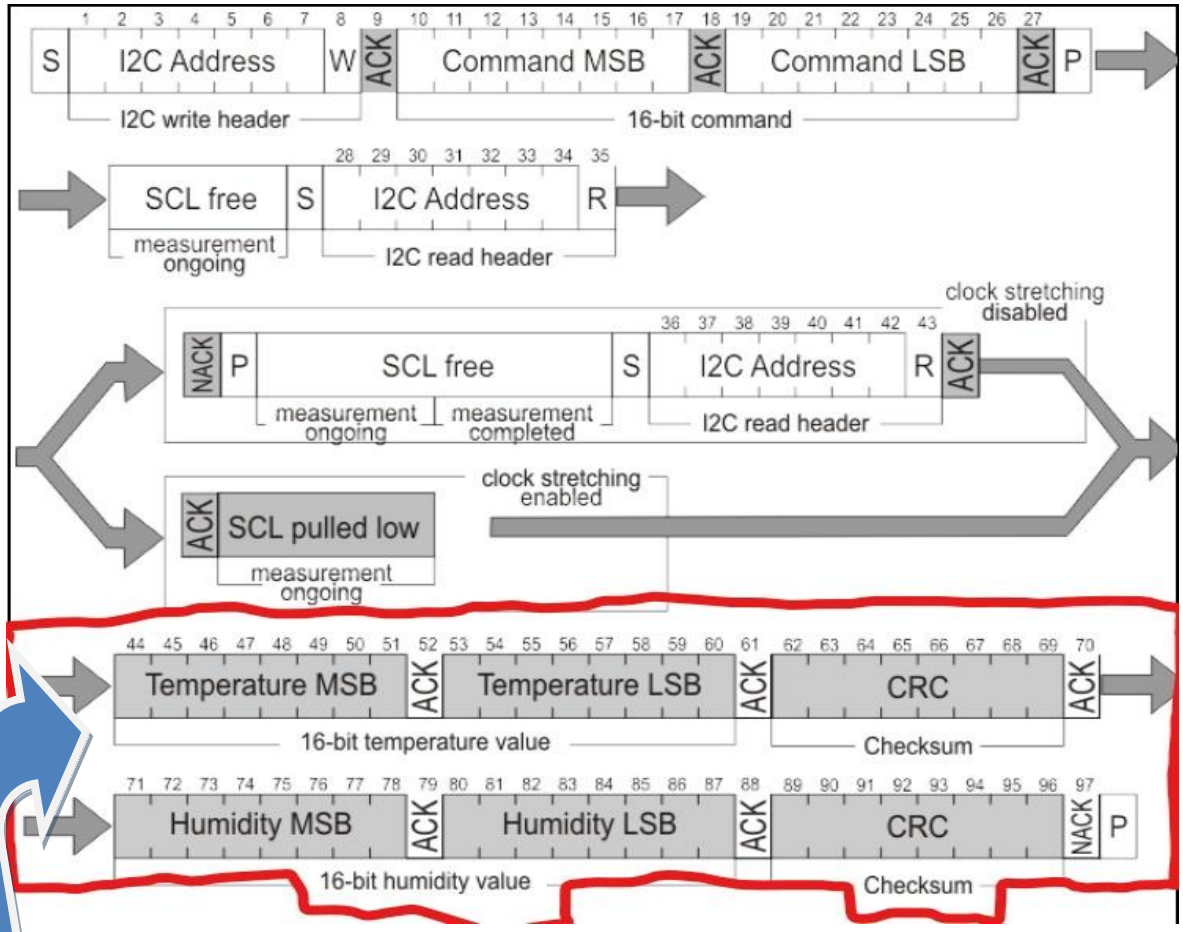
Et donc en réponse on a :

```
h62, h12, h2C, hDB, h33, hC9,

Start, h8A [ h45 | WR ], h24, h16, Stop
Start, h8B [ h45 | RD ], h62, h12, h2C, hDB, h33, hC9 NAK, Stop
```

H62 ,h12,h2C,hDB,h33,hC9,NAK

Ce qui correspond à :

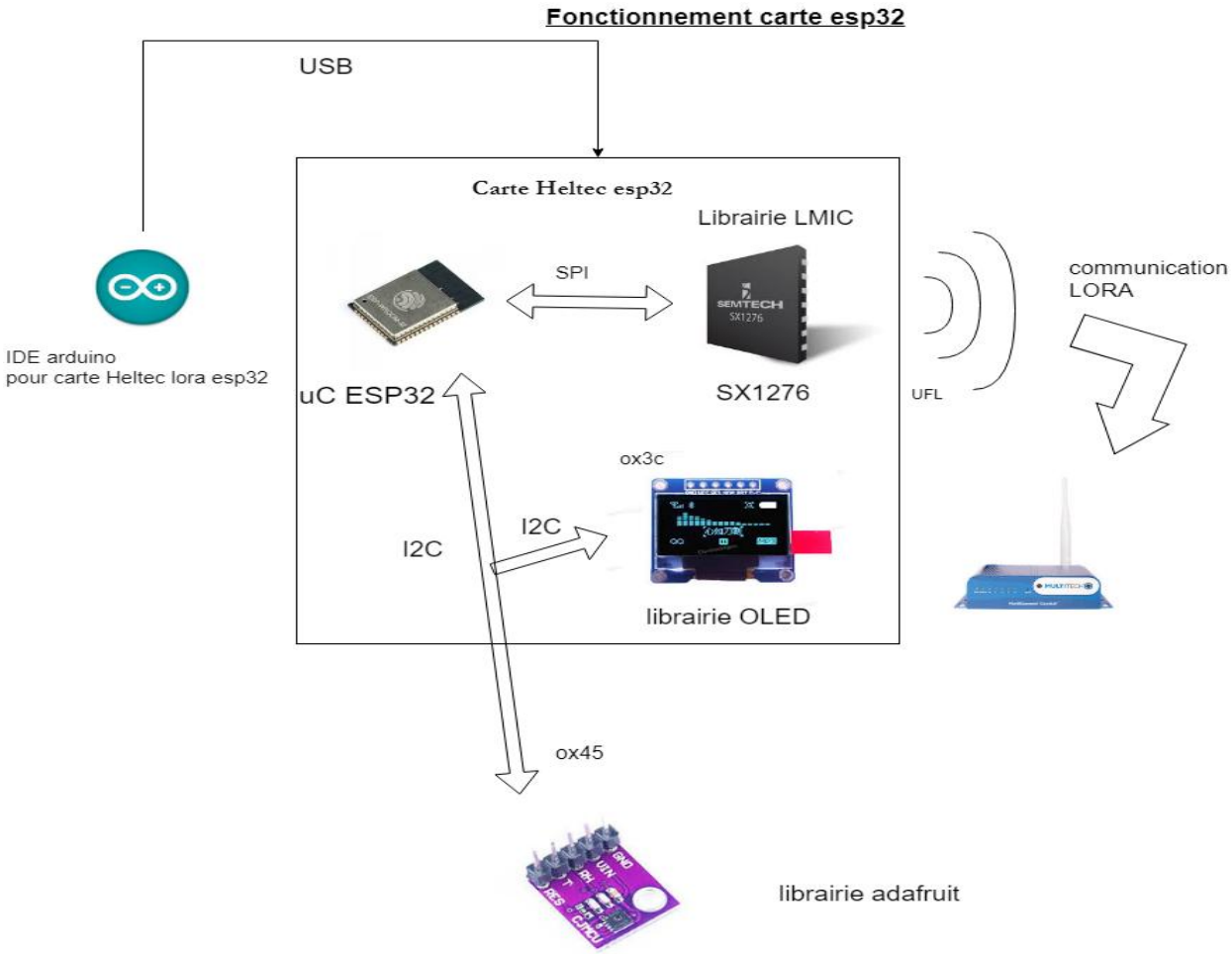


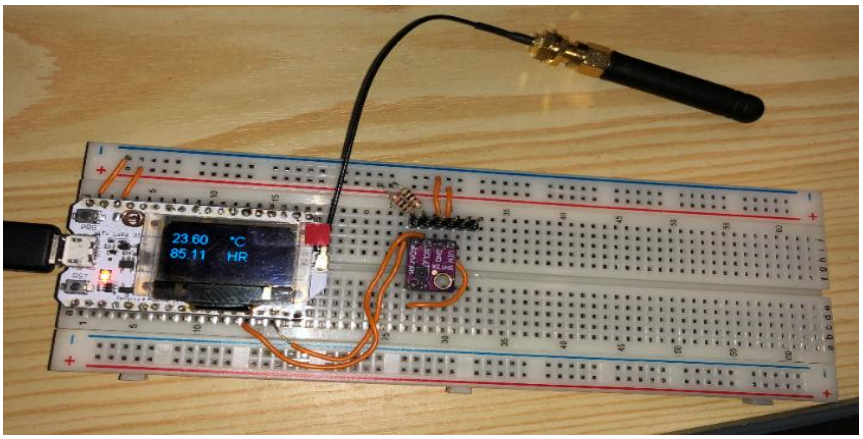
H62 ,h12,h2C : 2 premier octet pour la température puis un octet pour le CRC de la température vient

Ensuite **hDB,h33,hC9,NAK**

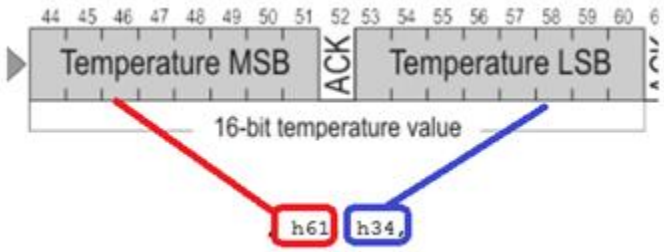
Qui correspond à 2 octets pour les mesures de l'humidité puis 1 octet du CRC de l'humidité et enfin 1 octet qui correspond à l'acknoledge

Mise en œuvre du module ESP32 Heltech avec le capteur sht31





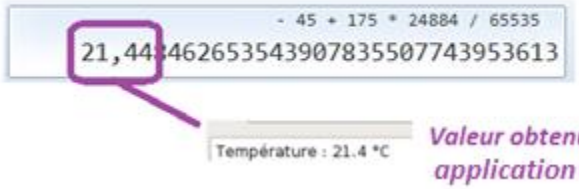
Par la suite j'ai mis en oeuvre ce capteur de température et d'hydrométrie avec mon premier module : la carte ESP 32, reliée à l'afficheur oled ssd1306. J'ai donc reussi à acquérir des températures et des degrés d'humidité, et à les afficher sur l'écran oled. Cependant, l'expérience montre que l'interfaçage de l'ESP32 avec le breakout SHT31-DIS nécessite l'ajout de résistances de pull-up de 1kΩ supplémentaires pour obtenir des signaux I2C valides.



Hexadecimal *0x61 0x34* -----> *Décimal* *24884* (S_T)

$$T[°C] = -45 + 175 \cdot \frac{S_T}{2^{16} - 1}$$

(Formule fournie avec la DataSheet permettant de calculer la température.)

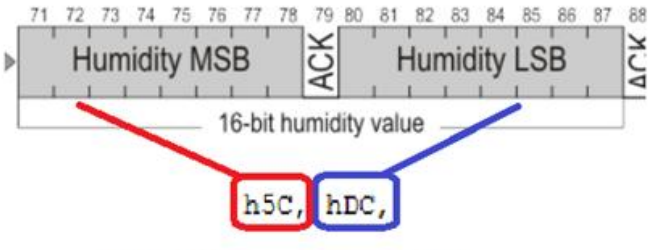


Valeur obtenue via mon application C++

On peut ainsi vérifier par ces calculs, si la température et l'hydrométrie affichées sur l'écran oled sont coherentes avec les calculs.

Calcul de la temperature

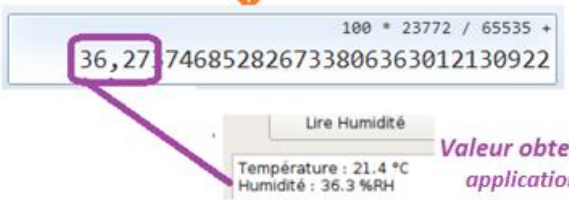
Calcul de l'humidité



Hexadecimal *0x5C 0xDC* -----> *Décimal* *23772* (S_{RH})

$$RH = 100 \cdot \frac{S_{RH}}{2^{16} - 1}$$

(Formule fournie avec la DataSheet permettant de calculer l'humidité .)



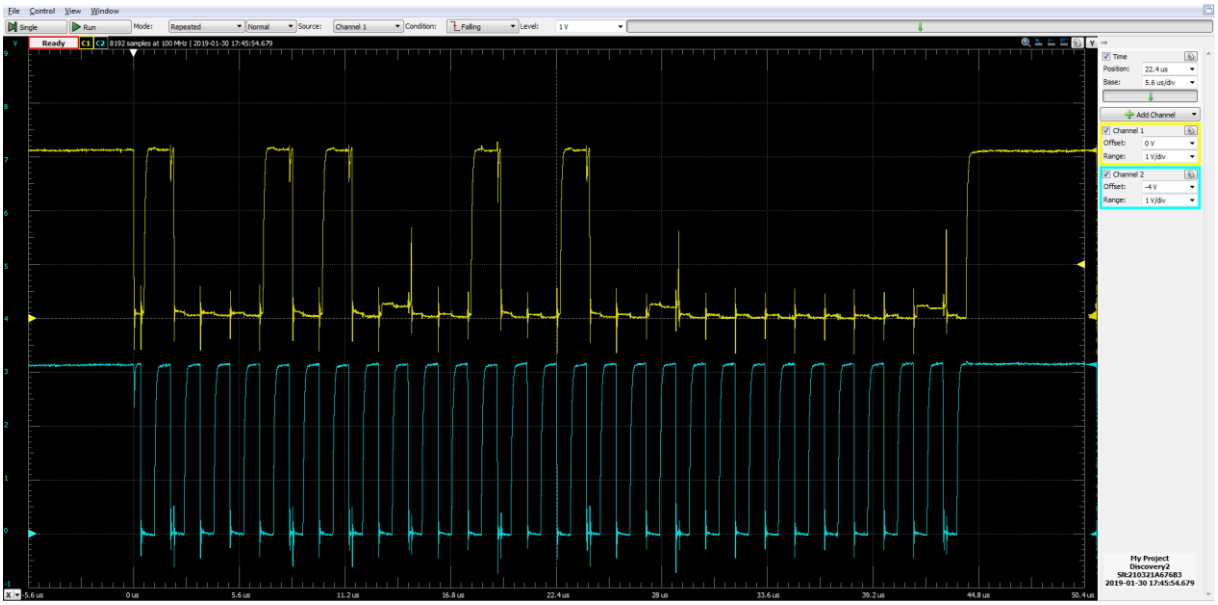
Valeur obtenue via mon application C++

Cependant, l'ajout de deux résistances pull-up de 1000 ohm est obligatoire car les signaux n'étaient pas adaptés :

Sans résistance :

on peut voir ici que les signaux ne sont pas carrés comme attendus

Avec les résistances pull-up :



Trame capteur sht31 avec pull-up

Enfin, il ne me restait plus qu'à installer les interfaces de développement sur arduino pour la carte Arduino MKW wan 1300, recueillir des données avec cette carte, puis les envoyer à ma passerelle LORA – en même

temps, il fallait aussi que j'installe les paramètres et les données avec les deux modules (esp 32 , MKR WAN 1300).

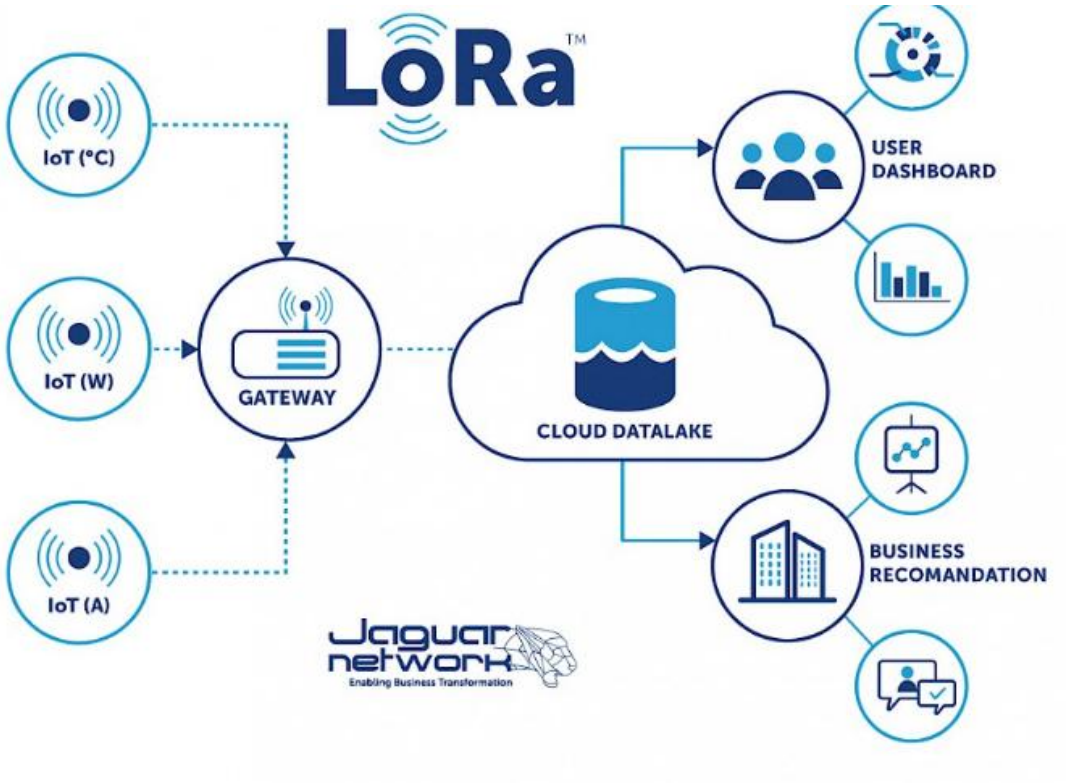
Par la suite ces données seraient envoyées par ma passerelle au serveur base de données.

Ce qui a été mis en œuvre – Revue de Projet n°2

A ce stade, j'ai créé un sketch pour que la MKR wan reçoive les mesures du sht31 et les envoie en lora à la passerelle multitech multiconnect.

Donc suite à cela, il a fallu installer et paramétrer cette passerelle LORA .

Mais d'abord il faut expliquer qu'est ce que le LORA :



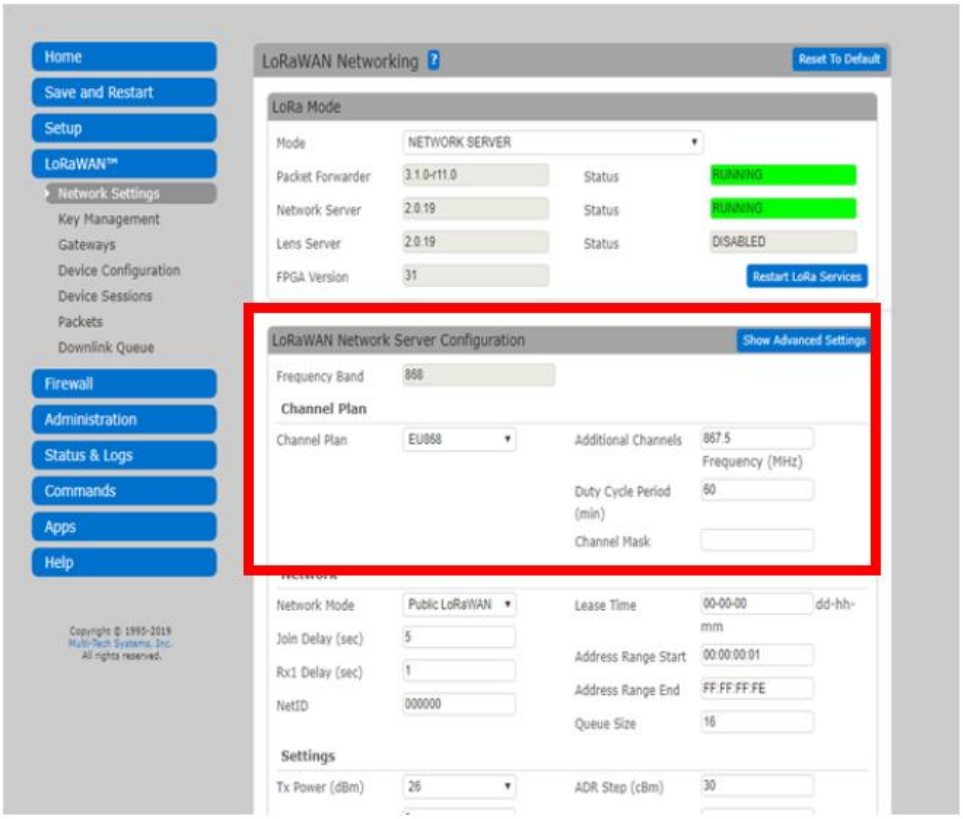
LoRaWAN est un protocole de télécommunication permettant la communication à bas débit, par radio, d'objets à faible consommation électrique communiquant selon la technologie LoRa et connectés à l'Internet via des passerelles, participant ainsi à l'Internet des objets.

Pour transmettre correctement mes mesures à la base de données, il a donc fallu installer et configurer la **passerelle multitech multiconnect .**

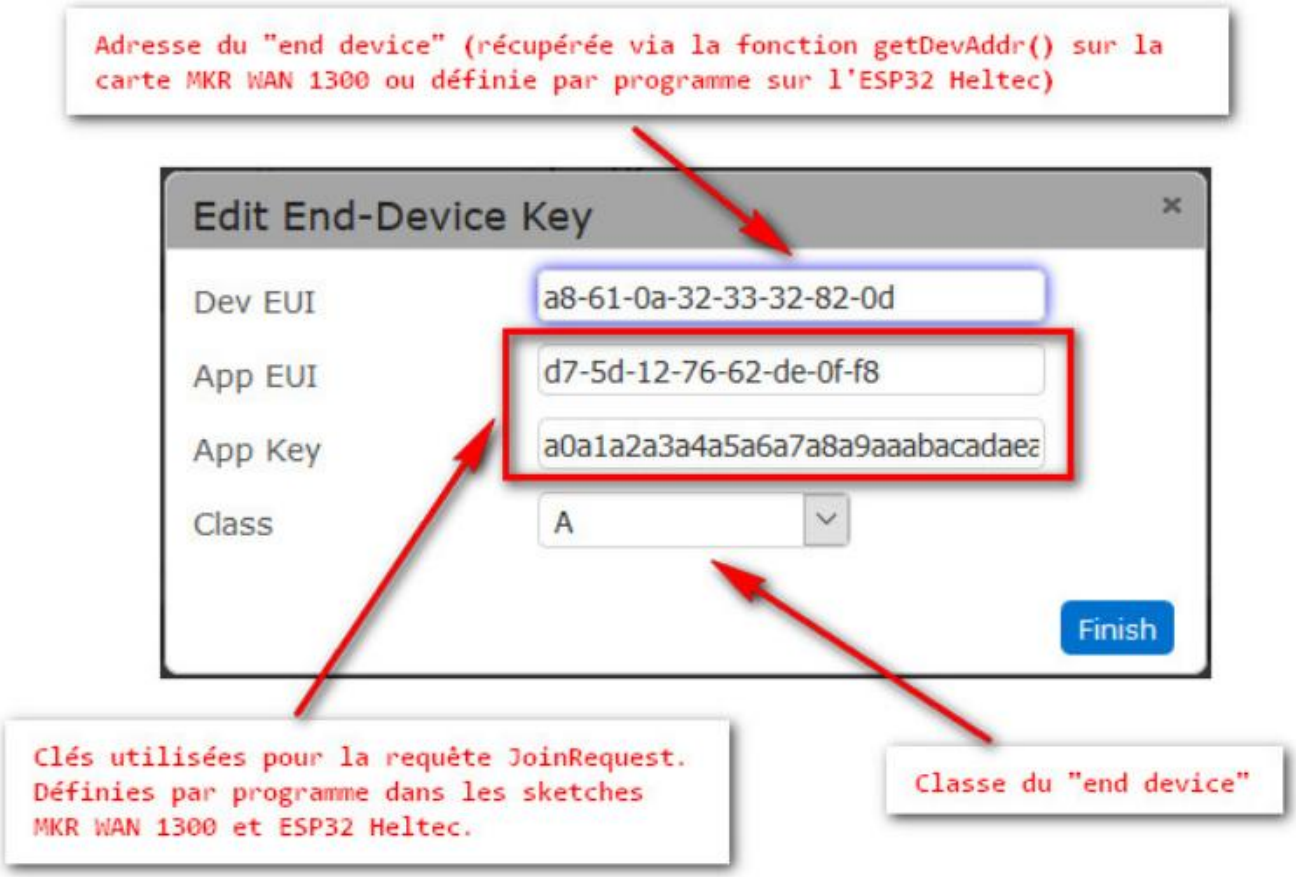
La nouvelle **passerelle programmable MultiConnect Conduit de MultiTech** est conçue pour les **applications M2M/IdO (Internet des Objets).**

Avec son environnement de développement ouvert, sa gamme de technologies cellulaires disponibles, son store d'applications (Device HQ) et sa plateforme de gestion (device management), le MultiConnect Conduit est la gateway parfaite pour le développement et le déploiement de vos applications M2M/IoT.

Il s'agit de la **première passerelle embarquant l'environnement de IBM Node-RED !**



J'ai configuré la passerelle comme ci-dessus de sorte qu'elle puisse bien communiquer en LORA avec les deux modules, notamment sur la bande de fréquences. Mettre la bande de frequencies correspondant à l'europe : 868



Une fois la passerelle bien connectée au reseau, il a fallu rentrer les DEVS EUI et APP EUI de chaque module dans la passerelle, plus précisément dans la partie **key management** puis dans la partie **devices configurations.**

Une fois que les deux cartes ont été enregistrées dans la passerelle, elles peuvent communiquer entr' elles. Par la suite, on va utiliser nodered pour pouvoir envoyer les données au broker MQTT relié au serveur base de données.

Mais d'abord il faut expliquer qu'est-ce que node-red

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

Node-RED fournit un éditeur de flux basé sur un navigateur Web , qui peut être utilisé pour créer des fonctions JavaScript . Des éléments d'applications peuvent être sauvegardés ou partagés pour être réutilisés.

Le temps d'exécution est construit sur Node.js . Les flux créés dans Node-RED sont stockés à l'aide de JSON .

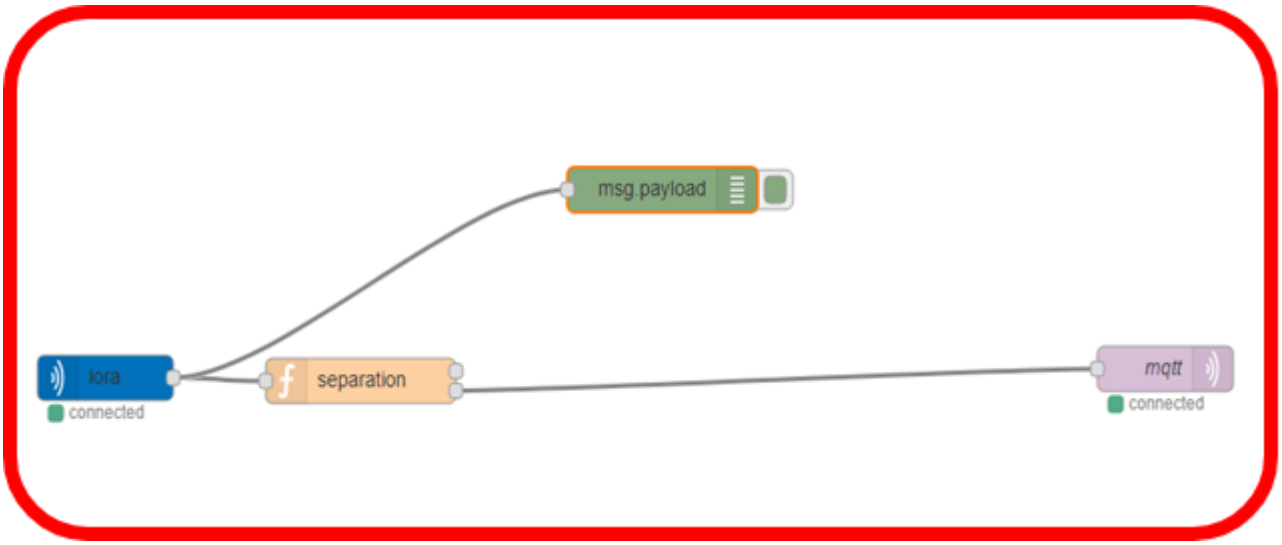
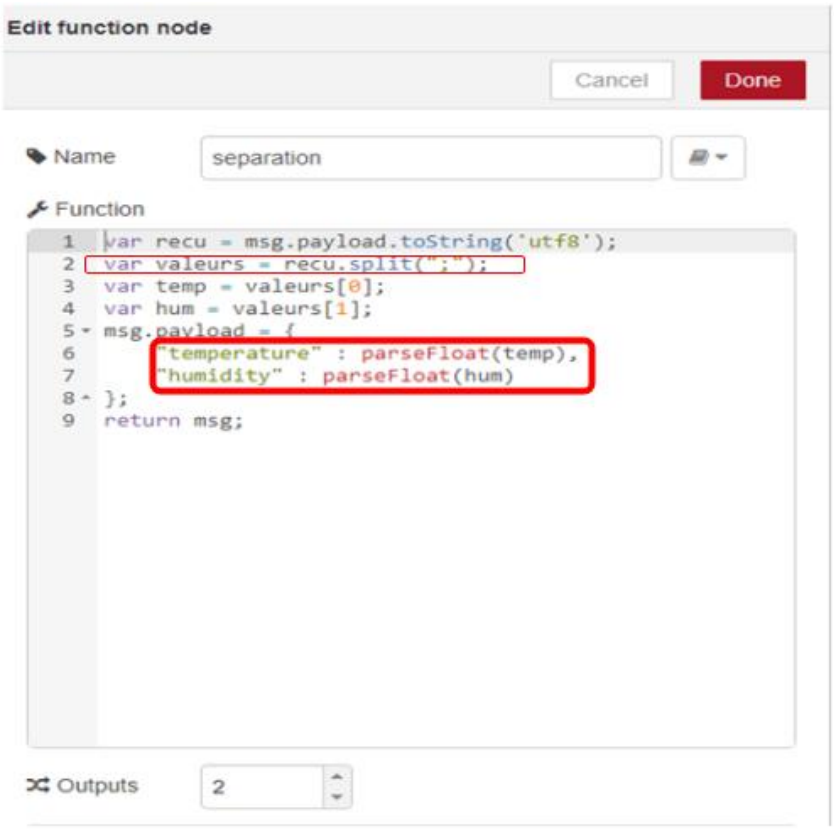


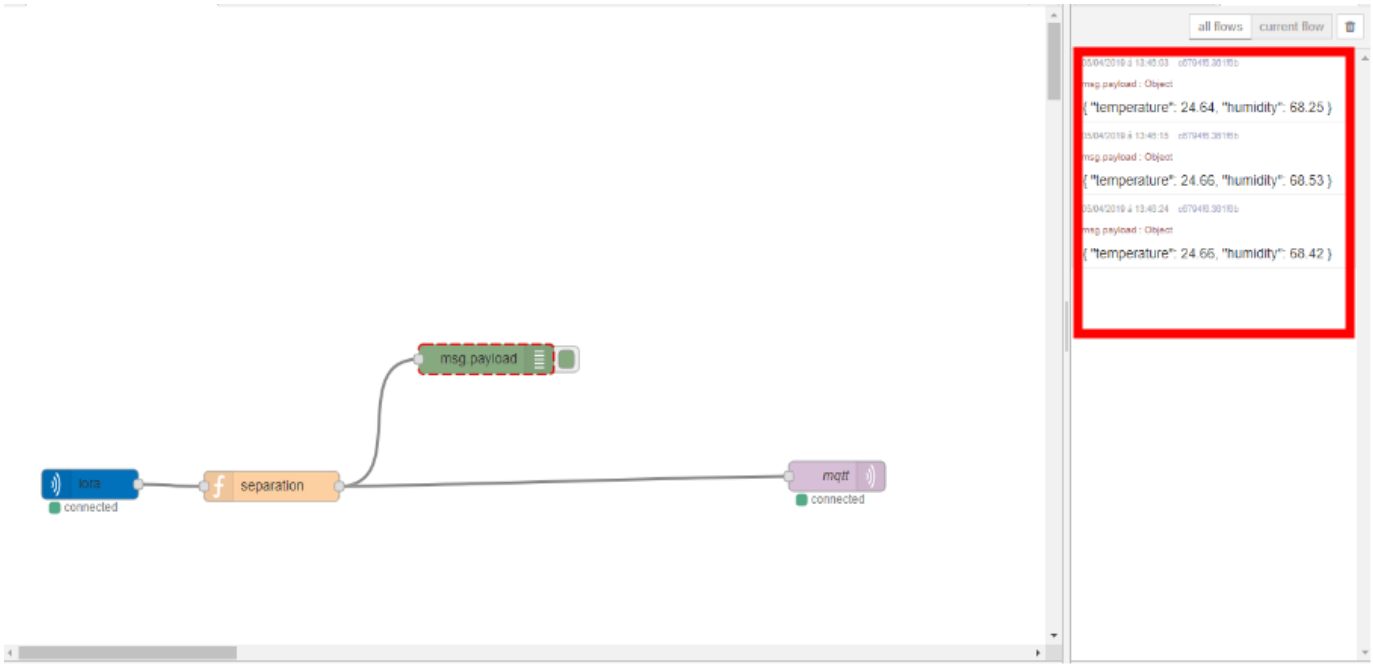
Schéma node-red : modules LORA vers broker mqtt

On peut donc voir qu'ici dans node-red de la passerelle, j'ai créé un schéma pour que les données arrivent aux normes au broker mqtt (connecté avec le serveur). Les données arrivent en LORA (modules rentrés dans la passerelle - voir partie ci-dessus) et sont envoyées par le réseau au broker mqtt, cependant pour que les données arrivent correctement à ce broker il faut rajouter une fonction "Séparation" :

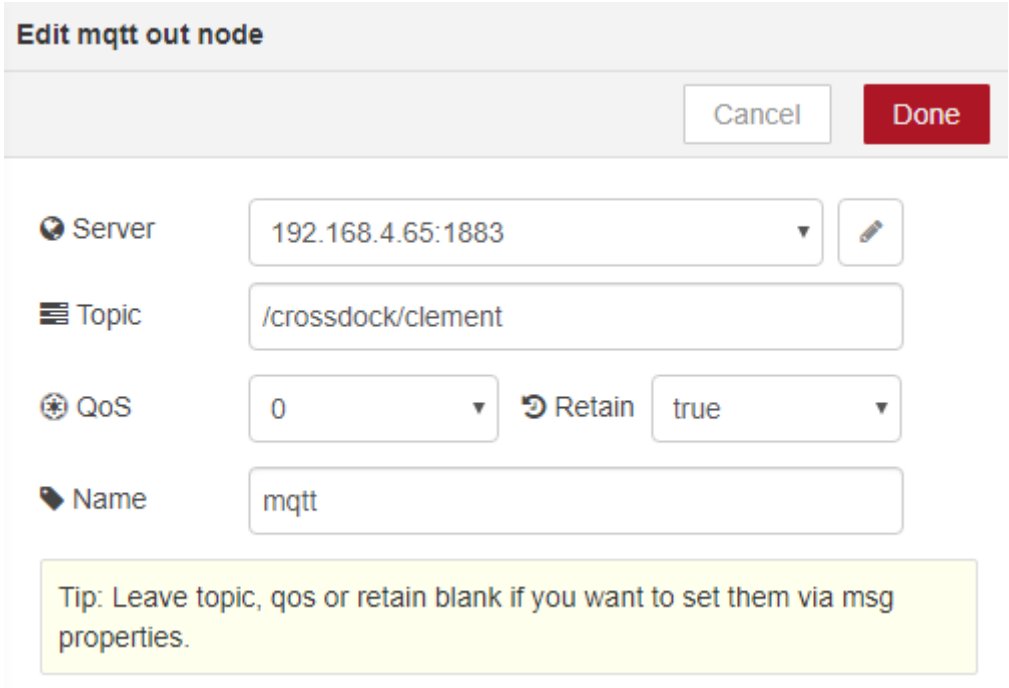


Cette fonction Séparation sert à la fois : à convertir mes valeurs que j'envoie sous forme de nombre à virgule à des nombres à virgules avec deux "0" après la virgule. Elle sert aussi à séparer les valeurs de la temperature et de l'humidité par un ";" comme on peut le voir dans la variable valeurs → recu.split(";")

Exemple :

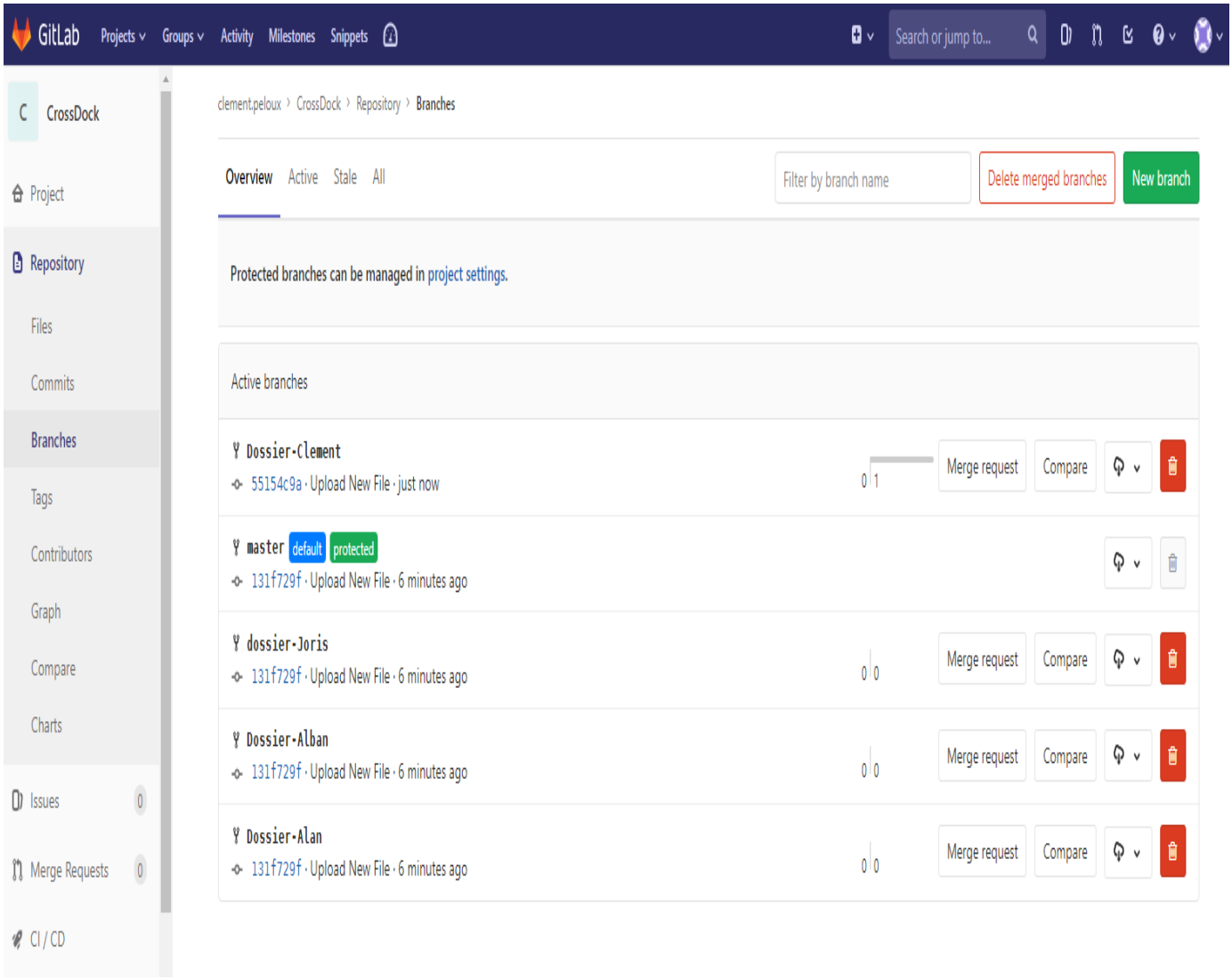


On peut voir qu'en sortie de cette fonction Séparation et donc aussi au broker mqtt que l'on reçoit bien les mesures. Cependant, j'ai réussi à envoyer ces mesures qu'avec la MKR wan



Voici une image du topic ou j'envoie les données donc a une adresse : **192.168.4.65** , au topic : **/crossdock/clement** du nom de mqtt.

Par la suite, nous avons créé un dossier gitlab pour l'ensemble du projet crossdock où tous les codes seront importés.

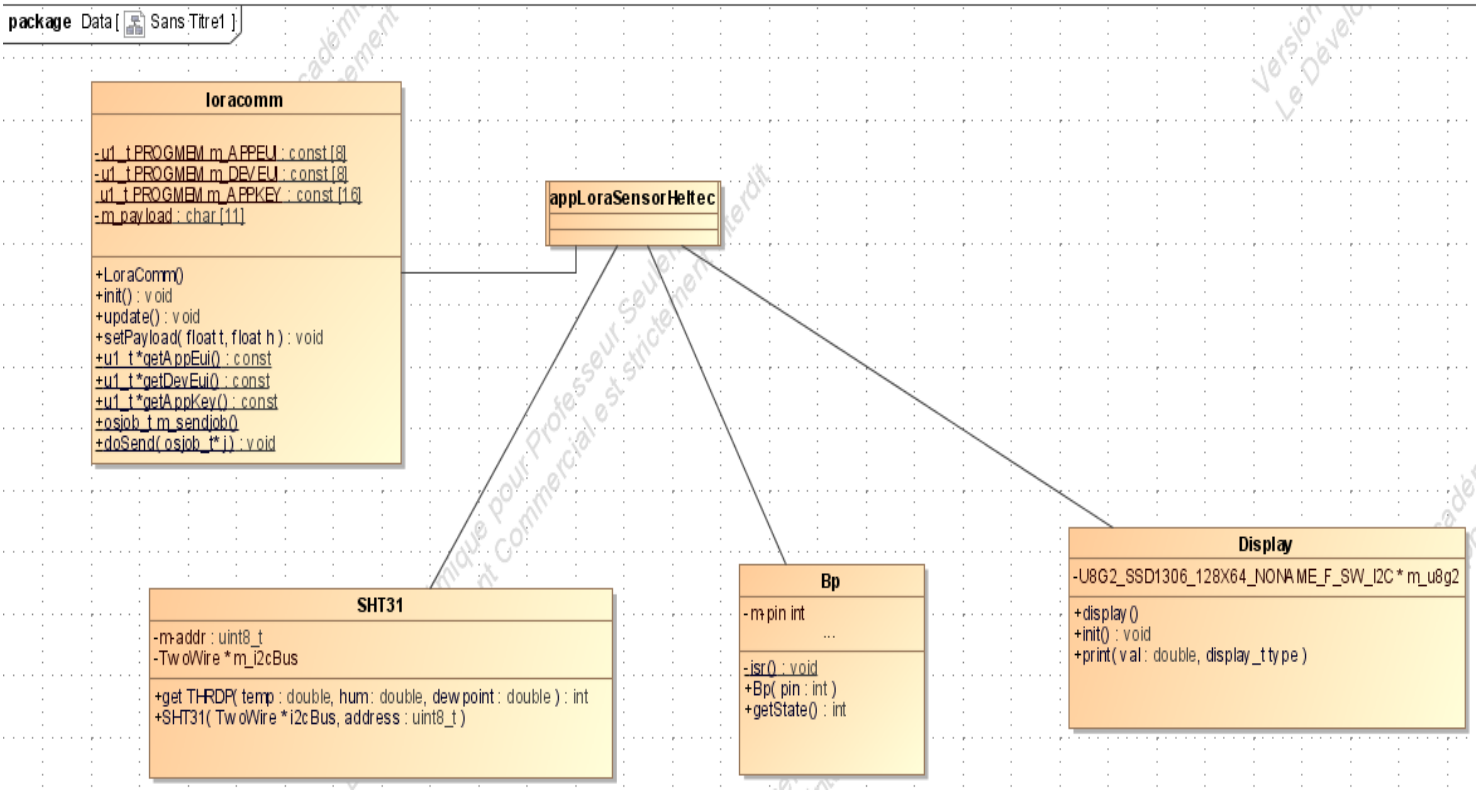


Ce qui a été mis en œuvre – Revue de Projet n°3

Pour la revue de projet n°3 il est question de recevoir les mesures aussi avec la carte esp32 et d'ajouter sur la plaquette contenant les cartes esp32 et mkr wan l'ajout de deux boutons poussoir permettant de modifier l'écran oled pour afficher soit la température, soit l'humidité, soit le point de rosée.

Pour ce faire il a fallu modifier les deux codes des deux cartes. De plus j'ai aussi modifié ces deux « sketchs » en langage de programmation « objet ».

Comme nous l'avons dit précédemment avec l'ajout de deux boutons poussoir sur la plaquette, j'ai du modifier mes sketchs en conséquence, et j'ai créé un diagramme de classe représentatif :



Ce diagramme de classe correspond la carte esp32 permettant la mofication de l'écran oled ss1306 grâce aux boutons poussoir.

Et voici la classe Bp

sketch-Afficheur-Oled | BP.cpp | BP.h

```
#include <Arduino.h>

class BP {
public :
  BP(int pin);
  int getState();
private:
  int m_pin;
  static void isr();
};
```

sketch-Afficheur-Oled | BP.cpp | BP.h

```
#include <hardwareserial.h>

#include "BP.h"

BP::BP(int pin): m_pin(pin) {
  pinMode(pin, INPUT);
  attachInterrupt(m_pin, isr, LOW);
  interrupts();
}

int BP::getState() {
  return digitalRead(m_pin);
}

void BP::isr() {
  Serial.println("BP interrupt");
}
```

Syntaxe

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)(recommandé)
```

Les interruptions sont utiles pour que les choses se passent automatiquement dans les programmes de microcontrôleur et peuvent aider à résoudre les problèmes de minutage. Les ISR sont des types spéciaux de fonctions qui ont des limitations uniques que la plupart des autres fonctions n'ont pas. Un ISR ne peut avoir aucun paramètre et il ne devrait rien renvoyer.

Par la suite j'ai donc fusionné mon code d'afficheur oled avec le code d'envoie des mesure en OTAA pour que la personne puisse modifier l'écran afin de voir les données utiles mais aussi pour envoyer ces mesures au broker mqtt de mon camarade.

sketch-Afficheur-Oled | BP.cpp | BP.h | SHT31.cpp | SHT31.h | display.cpp | display.h

```
//inclure la bibiliothèque de l'afficheur Oled

#include "SSD1306.h"
#include <Arduino.h>
#include "SHT31.h"
#include "BP.h"
#include "display.h"

SHT31 * thrSensor = new SHT31(&Wire1, 0x45);
//BP *nextBP = new BP(17);
//BP *previousBP = new BP(13);

double t, hr, dp;
int grandeur = 0;
Display oled;

// Initialiser l'oled en utilisant la bibliothèque de gestion de l'I2C ( Adresse I2C, pin SDA, pin SCL)
//SSD1306 oled(0x3c, 4, 15);

long last_micros = 0;

void affiche() {

    oled.flipScreenVertically();

    if (grandeur == 0) {
      oled.print(t, Display::T);
    } else if (grandeur == 1) {
      oled.print(hr, Display::HR);
    } else if (grandeur == 2) {
      oled.print(dp, Display::DP);
    }
}
```

```
void IRAM_ATTR isr17() {
  noInterrupts();
  if( (long)(micros() - last_micros) > 20*1000) {
    grandeur = (grandeur == 0) ? 2 : (grandeur - 1);
    affiche();
    last_micros =micros();
  }
  interrupts();
}

void setup() {

  Serial.begin(9600);
  pinMode(17, INPUT);
  attachInterrupt(13, isr13, FALLING);
  attachInterrupt(17, isr17, FALLING);

  while (!Serial)
    delay(10);     // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("SHT31 test");
}

void loop() {
  thrSensor->getTHRDP(t, hr, dp);

  delay(500);
}//fin loop
```

Sketch Afficheur oled

multitech-otaa-heltec

```
//
#ifdef COMPILE_REGRESSION_TEST
# define FILLMEIN 0
#else
# warning "You must replace the values marked FILLMEIN with real values from the TTN control panel!"
# define FILLMEIN (#dont edit this, edit the lines that use FILLMEIN)
#endif

// Variable pour déclencher une mesure à intervalle de temps régulier
unsigned long previous;

// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,
// 0x70.

//static const u1_t PROGMEM APPEUI[8]={ 0xe8, 0xe7, 0xe6, 0xe5, 0xe4, 0xe3, 0xe2, 0xe1 };
static const u1_t PROGMEM APPEUI[8]={ 0x70, 0x70, 0x41, 0x6B, 0x63, 0x6F, 0x44, 0x78 };
// APPEUI Peloux : ppAkcoDx (xDockApp à l'envers)
// static const u1_t PROGMEM APPEUI[8]={ 0x70,0x70, 0x41, 0x63, 0x3b, 0x6f, 0x44, 0x78 };
//
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
//                                    0x000001           "xDock"
//                                 0x01, 0x00, 0x00,  'k',  'c',  'o',  'D',  'x'
static const u1_t PROGMEM DEVEUI[8]={ 0x02, 0x00, 0x00, 0x6B, 0x63, 0x6F, 0x44, 0x78};
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
static const u1_t PROGMEM APPKEY[16] = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

Sketch multitech-otaa-heltech

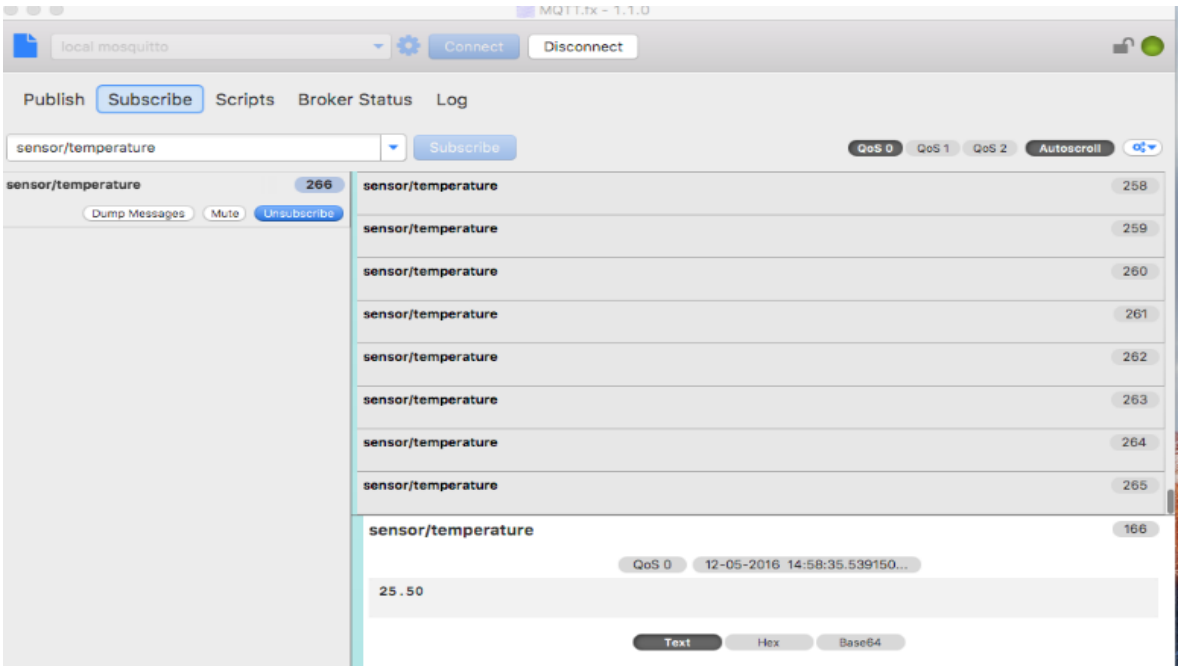
J'ai dû faire de même avec la carte MKR wan , j'ai fusionné les codes d'affichage et d'envoi de données.

J'ai aussi utilisé le client mqttfx pour simuler l'arrivée des mesures au broker mqtt de mon camarade, cela me permet de vérifier si les mesures reçues par mon camarade sont réelles et vérifiables.

Mais d'abord il faut expliquer qu'est-ce que le client mqtt.fx

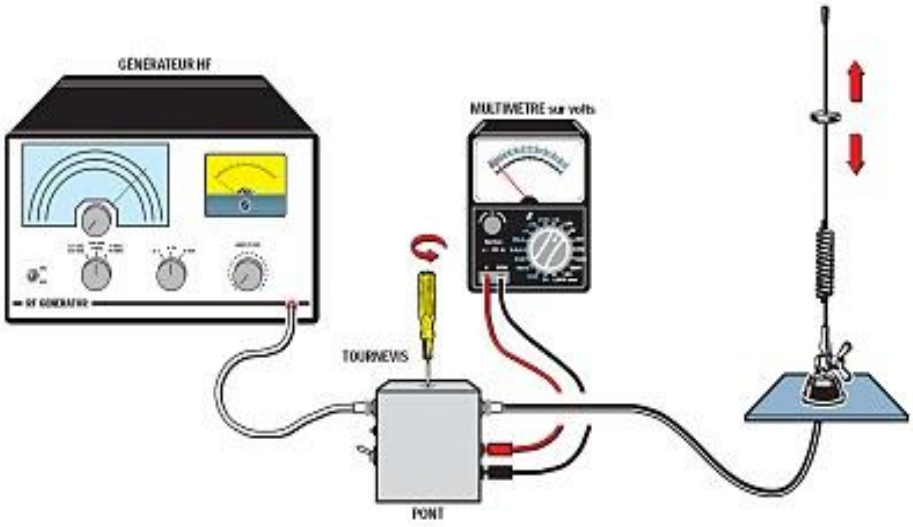
A la place de Node-RED on peut également utiliser MQTT.fx, un logiciel open source multiplateforme. MQTT.fx est un outil assez complet qui permet :

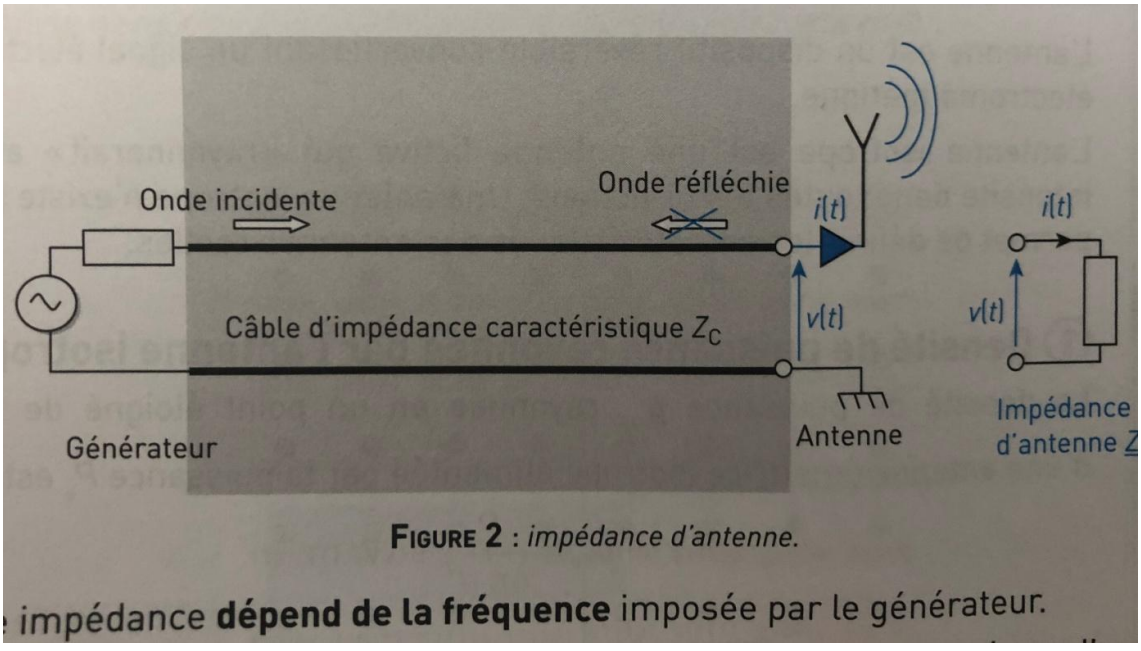
- De souscrire ou de publier des messages en ajustant le QoS (quality of service : la qualité du message)
- De connaître le statut de votre Broker (nombre de clients connectés, messages reçus, envoyés, stockés, trafic réseau…)
- D'avoir accès au log du Broker
- De créer des scripts en Javascript



Partie physique

Lors du branchement des cartes esp32 et mkrwan1300, il est obligatoire de brancher l'antenne avant de relier les cartes à un port USB car cela pourrait entrainer le retour du signal au generateur - ici les cartes - ce qui reduirait l'efficacité de propagation des ondes. Pour que la ligne soit parfaitement adaptée, il faut que impedance d'antenne soit égale à impedance du câble :





Si en sortie il n'y a pas d'antenne, alors contrairement au schéma , il y aura une onde reflechie qui reviendra à son point de depart - ici un générateur - et dans notre cas, une des deux cartes, ce qui pourrait abimer, voir déteriorer certains composants des cartes.

Etudiant EC1

Alban JUAN

Introduction / Mise en situation

Pour valider mes compétences en BTS SN, j'ai participé cette année à un Projet Technique. Nous avons travaillé à quatre sur un certain nombre de prototypes pour une entreprise : Crossdock, basée à Cavaillon (Vaucluse). Une autre équipe de projet a également fait un projet pour Crossdock, car l'entreprise est grande.

Cette entreprise fait du « cross-docking » (ré-empaquetage) de produits pharmaceutiques et cosmétiques, et a donc des contraintes de températures. Le responsable, M. Bijou possède déjà plusieurs capteurs dans son entrepôt pour surveiller les conditions de stockage. Les capteurs réalisent quatre mesures par jour, qui doivent être récupérées manuellement (le capteur ne possédant pas d'émetteur de données).

Récupérer les données chaque jour n'est pas pratique, notre tâche a donc été basée sur une demande précise : récupérer des valeurs mesurées par des capteurs de température/hygrométrie, et permettre au responsable d'avoir un œil sur ces données directement depuis un poste informatique.

I) Etude du cahier des charges

1) Analyse des attentes du client

Nous prenons la posture d'un technicien lors de ce projet, en étant souvent aidé par un ou plusieurs professeurs référents. Cela nous impose d'être efficace dès le début du projet en lisant correctement le cahier des charges. Dans notre cas, il a été rédigé par nos professeurs M Hortolland *(EC)* et M Defrance *(IR)*, en accord avec M Bijou, le responsable de *Crossdock*.

Les points importants établis sont :

• Présenter des comptes rendus d'essais de différents capteurs, notamment pour tester leur fiabilité (choix du meilleur capteur à utiliser)

• Réussir à obtenir des résultats au moins aussi bons que les actuels, grâce à des cartes de tests sur lesquelles sont soudées des capteurs de température/hygrométrie

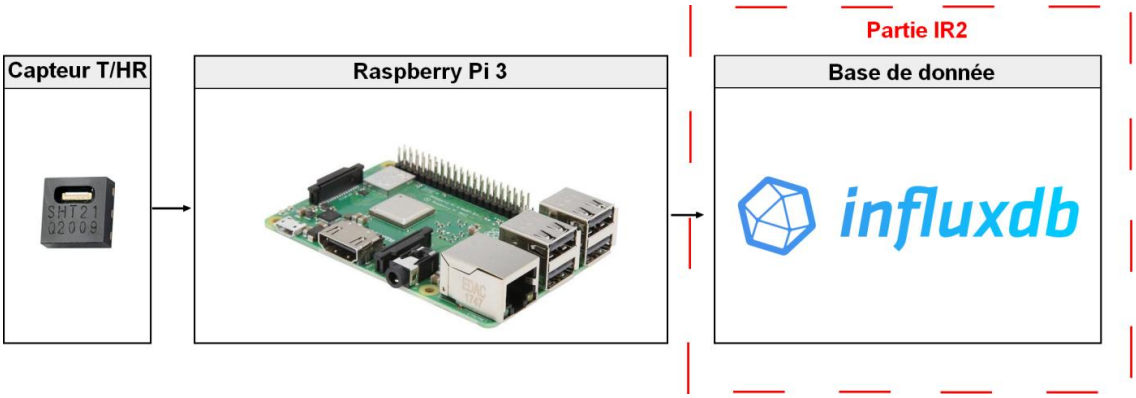
• Pouvoir récupérer les données depuis un poste, grâce à la gestion automatique d'une base de donnée (travail étudiant *IR*)

• Proposer plusieurs prototypes, telles qu'une carte filaire puis une carte sans-fil.

Nous avons conclu que nous pourrions ultérieurement ajuster des détails selon l'avancement du projet, étant donné que le cahier des charges n'est pas totalement fixé. Respecter ces règles de base nous a permis de convenir au mieux aux besoins de l'entreprise " cliente ".

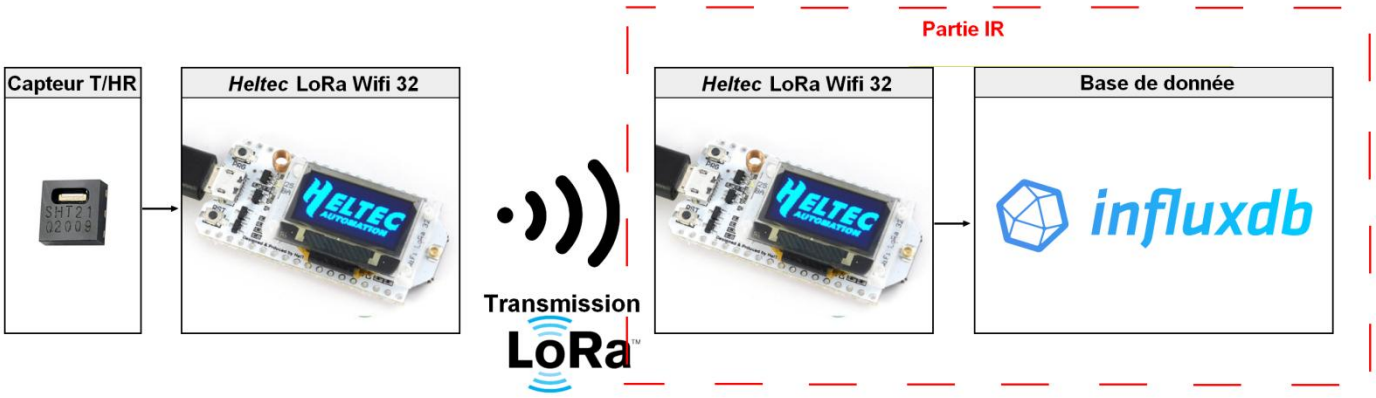
2) Ma partie

Globalement, c'est moi qui ai choisi le capteur que chaque étudiant concerné a dû utiliser. À la suite de ces tests, j'ai dû créer un circuit imprimé grâce à KiCad. Celui-ci est une « mise au propre » du montage utilisé pour capturer des valeurs. Le montage est celui-ci :



Les données captées sont gérées par un *Raspberry Pi 3*, puis envoyées sur une base de donnée gérée par *InfluxDB*. Tous les programmes finaux sont créés par des étudiants IR, les EC s'occupant en majeure partie de la conception de cartes électroniques.

Une solution sans-fil devra être proposée. Selon le cahier des charges, le principe de base qu'on utilisera pour celle-ci est le suivant :

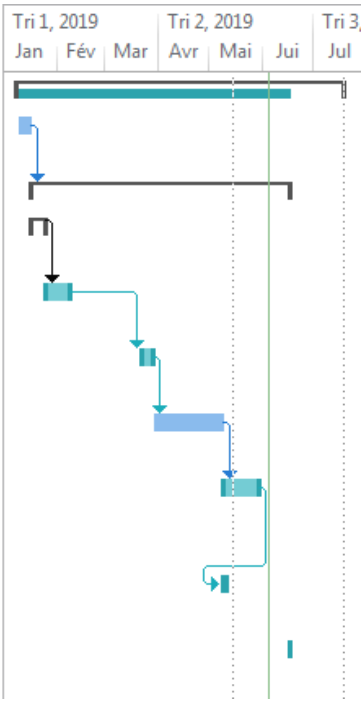


Le module émetteur possède un écran, ce qui permettra d'afficher les températures, humidités et points de rosée de manière instantanée. Il transmet ces relevés à l'autre module Heltec via **LoRa**, qui est un sous-réseau WiFi en basses fréquences.

3) Estimations de temps

Le diagramme de Gantt suivant est celui créé au départ. En effet, ce ne sont que des estimations, que j'ai avons ajustées lors de l'avancement du projet. Le Gantt permet donc d'avoir un aperçu intéressant sur les différentes tâches de mon projet, ainsi que le temps estimé pour chacune.

	Mode Tâche	Nom de la tâche	Durée	Début	Fin
1		**Projet BTS SN**	**39,57 jours**	**Mer 09/01/1**	**Jeu 18/07/19**
2		Analyse cahier des charges	2 jours	Mer 09/01/19	Mer 16/01/19
3		**Capteurs**	**28,86 jours**	**Mer 16/01/1**	**Lun 17/06/19**
4		**Comparaison des capteurs**	**3 jours**	**Mer 16/01/19**	**Ven 25/01/19**
7		Tests des capteurs	24 hr	Ven 25/01/19	Ven 08/02/19
8		Comparaison des résultats	12 hr	Ven 22/03/19	Ven 29/03/19
9		Fabrication solution filaire	36 hr	Ven 29/03/19	Jeu 09/05/19
10		Fabrication solution sans-fil	36 hr	Jeu 09/05/19	Mer 29/05/19
11		Installation Crossdock	10 hr	Jeu 09/05/19	Ven 10/05/19
12		Soutenance projet	1 hr	Lun 17/06/19	Lun 17/06/19



II) Capteurs de température / hygrométrie

1) Choix d'une gamme de capteurs

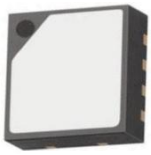
Le cahier des charges n'est pas totalement défini et je dois choisir un capteur parmi les nombreux disponibles sur le marché.

Comme M Bijou utilise des **Raspberry Pi 3** dans son entreprise, il faut que le capteur y soit adapté. Ce « nano-ordinateur » possède un certain nombre d'entrées / sorties. Parmi elles se trouvent les broches ***SDA*** et ***SCL***, permettant une **communication I²C**, un standard en électronique. Le capteur devra donc permettre une communication I²C, et **être au moins aussi bon** que le **capteur** utilisé à l'**origine** par Crossdock. L'entreprise aidera à financer le projet, mais nous devons choisir du matériel dont le prix est raisonnable.

Trois capteurs offrent des caractéristiques similaires :

Capteur	Gamme température	Précision température	Résolution température	Gamme humidité	Précision humidité	Résolution H.R.	Type de communication	Distributeurs	Gamme de Prix
PCE HT71	-40 à +105 °C	±1°C	0,1°C	0 à 100 %	±3 %	0,1 %	xxxxxx	PCE Instruments	60€ TTC

SHT21



SHT31

Si7021

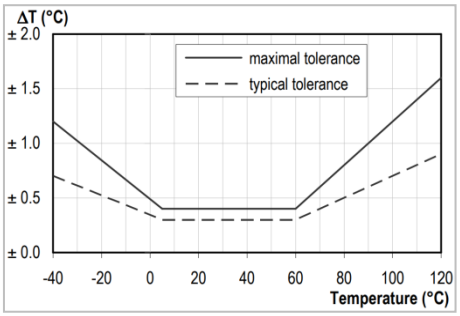
Capteur	Gamme temp. (°C)	Précision température	Résolution température	Gamme humidité	Précision humidité	Résolution H.R.	Type communication	Distributeurs	Gamme de prix
SHT21	-40➢+125	±0,3°C	0,01°C	0%➢100%	±2%	0,05%	I²C	RS	5,81€
SHT31	-40➢+125	±0,2°C	0,01°C	0%➢100%	±2%	0,05%	I²C	RS	5,93€
Si7021	-40➢+125	±0,4°C	0,01°C	0%➢100%	±3%	0,05%	I²C	Farnell	3,21€

Globalement, le SHT31 offrirait une meilleure précision.

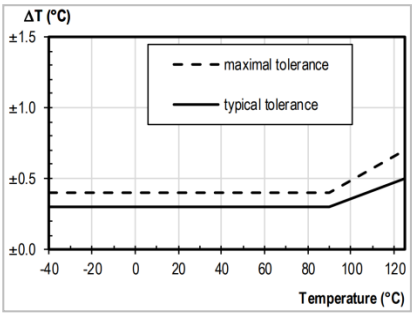
2) Précision des capteurs

Les *datasheets* (documentations) des capteurs nous informent sur leurs courbes de précision.

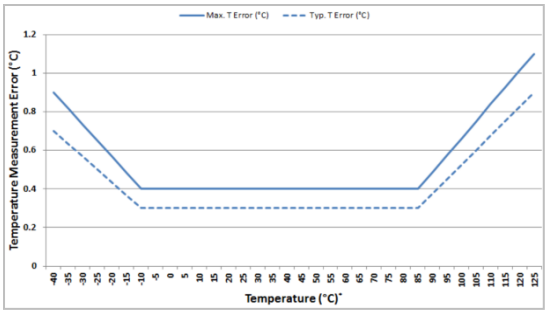
Courbes d'erreur pour la mesure de température :



SHT21

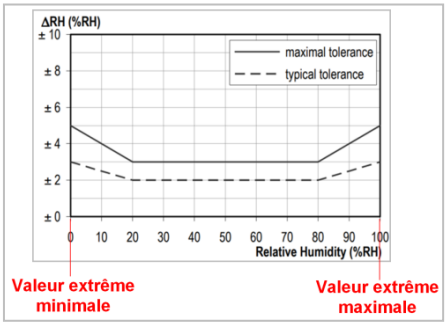


SHT31

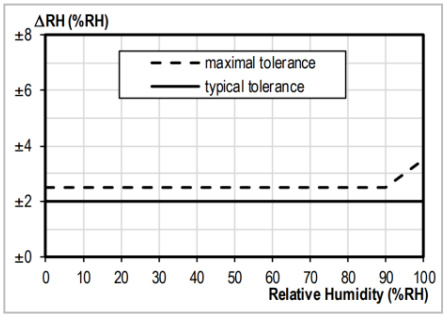


Si7021

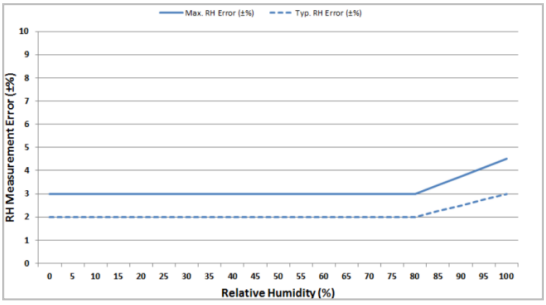
Courbes d'erreur pour la mesure d'humidité relative :



SHT21



SHT31



Si7021

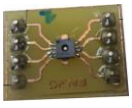
On voit une erreur sur la température légèrement moins importante sur le capteur **Si7021**, et une erreur sur l'humidité plus faible sur le SHT31 **(5%** d'erreur environ lors des **« extrêmes »**). Ces valeurs correspondent effectivement au capteur d'origine de Crossdock, et sont même **meilleures**.

3) Tests de bon fonctionnement

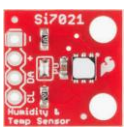
Mon équipe de projet *Crossdock 1* et l'équipe *Crossdock* 2, utiliserons le même type de capteurs. Je dois donc compléter la lecture des documentations, par des tests réels.

M Hortolland, mon professeur référent, m'a fourni un capteur Si7021 « breakout » (capteur pré-monté), du fournisseur **Sparkfun.** Il m'a également confié deux capteurs qu'il a routé, un SHT21 et un SHT31.

SHT21

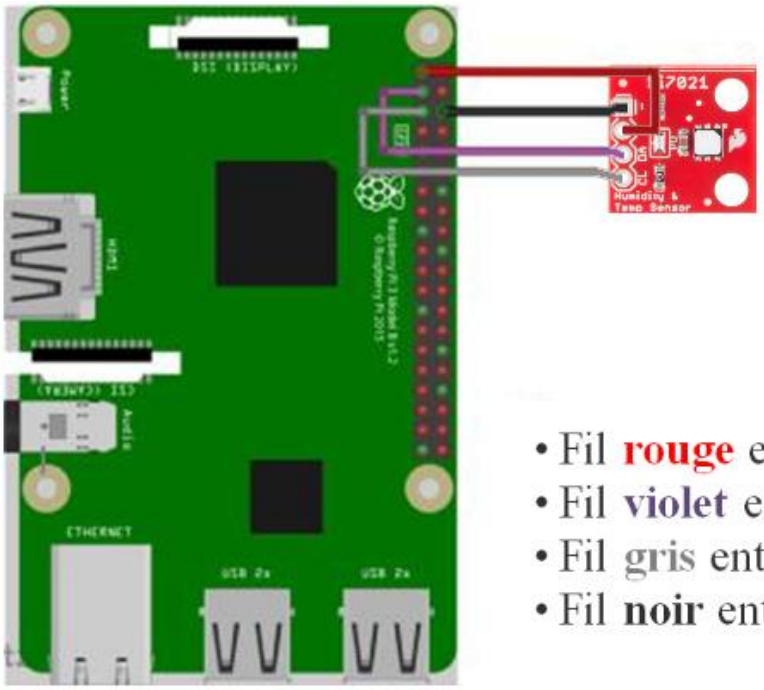


SHT31



Si7021

Comme la solution filaire fonctionne grâce à un Raspberry Pi 3, j'utiliserai ce type de structure :

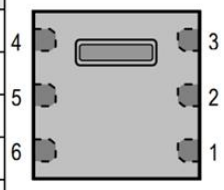


RPI3		Capteur
• Fil **rouge** entre 3.3V	et	+
• Fil **violet** entre I2C_SDA	et	DA
• Fil **gris** entre I2C_SCL	et	CL
• Fil **noir** entre GND	et	-

Ici, le capteur est un breakout Si7021, sur lequel les noms des broches (+, DA, CL et -) sont inscrits. Pour les deux autres capteurs (SHT21, SHT31) il m'a fallu regarder dans les documentations, la correspondance des broches.

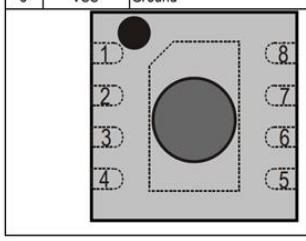
SHT21

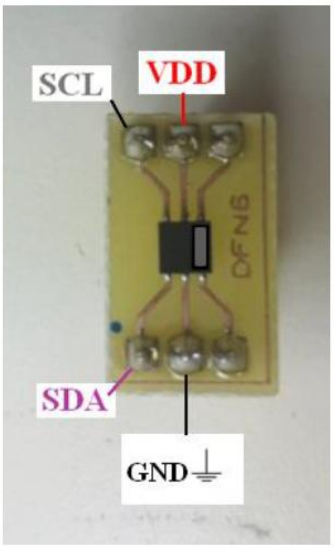
Pin	Name	Comment
1	SDA	Serial Data, bidirectional
2	VSS	Ground
5	VDD	Supply Voltage
6	SCL	Serial Clock, bidirectional
3,4	NC	Not Connected

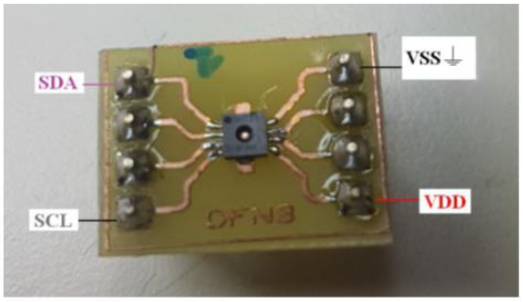


SHT31

Pin	Name	Comments
1	SDA	Serial data; input / output
2	ADDR	Address pin; input; connect to either logic high or low, do not leave floating
3	ALERT	Indicates alarm condition; output; must be left floating if unused
4	SCL	Serial clock; input / output
5	VDD	Supply voltage; input
6	nRESET	Reset pin active low; input; if not used it is recommend to be left floating
7	R	No electrical function; to be connected to VSS
8	VSS	Ground







Bien que l'on n'utilise que quatre broches pour les breakouts routés par M Hortolland, on voit que le SHT31 possède en réalité huit broches. Ce capteur est intéressant car il possède une pin « ADDR » qui permet de changer l'adresse du capteur sur le Bus I2C (adresse qui permet d'identifier le capteur), en la connectant au VDD *(3,3V)*.

Tests

Mon professeur m'a donné un programme de base, que l'on peut exécuter sur le Raspberry Pi.

Ce programme est en C++ et est à recompiler à chaque modification. Il fonctionne sur un principe simple : récupérer une valeur sur le bus I2C, la convertir en température grâce à des variables (disponibles dans la documentation des capteurs), et l'afficher sur un invite de commande :

Programme et annotations :

```
// Mesure de temperature avec SHT21
// Fichier SHT21.cpp
#include <iostream>
#include <bcm2835.h>
#include <math.h>
using namespace std;

#define clk_div BCM2835_I2C_CLOCK_DIVIDER_2500

// g++ -o SHT21 bcm2835.cpp SHT21.cpp

void init_I2c_bcm2835()
{
    if (!bcm2835_init())
        {
        printf("bcm2835_init failed. Are you running as root??\n");
        }
    if (!bcm2835_i2c_begin())
        {
        printf("bcm2835_i2c_begin failed. Are you running as root??\n");
        }
    bcm2835_i2c_setClockDivider(clk_div);          // Division de l'horloge Rpi pour obtenir une vitesse de 100KHz
}

int main(void)
{
    float mesureT, mesureRH;
    float alpha, Tr;
    int ST, SRH;
    unsigned int slave_address=0x40; // Adresse du SHT21
    char i2cOut[3];     // Tableau des données I2C à sortir
    char i2cIn[2];      // Tableau des données I2C entrées

    init_I2c_bcm2835();
    bcm2835_i2c_setSlaveAddress(slave_address);   // Affectation de l'adresse de l'esclave
    cout<<"Mesures sur le SHT20/SHT21 : "<<endl;


    i2cOut[0] = 0xFE;                  // Soft reset
    bcm2835_i2c_write(i2cOut, 1);  //

    delay(1000);                       // 1 seconde avant première lecture

    while(1)
        {
        // Mesure température
        i2cOut[0] = 0xF3;                  // Octet  lecture température (no hold master)
        bcm2835_i2c_write(i2cOut, 1);  // Commande lecture température
        delay(100);                            // 100 msec de délais en dessous cela dysfonctionne
        bcm2835_i2c_read(i2cIn,2);     // Lecture des 2 octets du resultat
        //cout<<"i2cIn[0] = "<< (int)i2cIn[0]<< "     i2cIn[1] = "<< (int)i2cIn[1]<<endl;
        ST=256*(int)i2cIn[0] + (int)(i2cIn[1]&0b11111100);     // Signal Temperature
        //cout<<"ST = "<< ST<<endl;
        mesureT= (-46.85 +175.72 * ST/65536);
        cout<<"Temperature du SHT20/21 : "<<mesureT<<" deg."<<endl;

        // Mesure humidité relative
        i2cOut[0] = 0xF5;                  // Octet  lecture humidité relative (no hold master)
        bcm2835_i2c_write(i2cOut, 1);  // Commande lecture humidité relative
        delay(100);                            // 100 msec de délais en dessous cela dysfonctionne
        bcm2835_i2c_read(i2cIn,2);     // Lecture des 2 octets du resultat
        //cout<<"i2cIn[0] = "<< (int)i2cIn[0]<< "     i2cIn[1] = "<< (int)i2cIn[1]<<endl;
        SRH=256*(int)i2cIn[0] + (int)(i2cIn[1]&0b11110000);   // Signal humidité relative
        //cout<<"SRH= "<< SRH<<endl;
        mesureRH= (-6.00 +125.00 * SRH/65536);
        cout<<"Humidité relative du SHT20/21 : "<<mesureRH<<" %"<<endl;

        // Calcul du point de rosée
        alpha=(17.27*mesureT)/(237.7+mesureT)+log(mesureRH/100);
        Tr = (237.7*alpha)/(17.27-alpha);
        cout<<"Point de rosée : "<<Tr<<" deg."<<endl;

        cout<<endl;
        delay(1000);                           // 1 seconde entre chaque conversion
        }

    bcm2835_close();
    return 0;
}
```

Inclut les fonctions de texte console (nécessaire à la compilation)

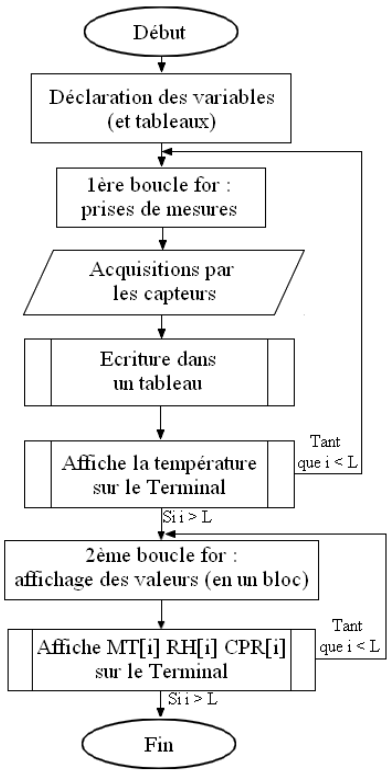
Initialisation de l'I2C sur le RPI

Déclaration des variables

Mesure température
> placement dans la variable mesureT
Annonce valeur dans la console

Mesure humidité relative
> placement dans la variable mesureRH
Annonce valeur dans la console

Détermination du point de rosée selon un calcul (relation entre l'humidité et la température)

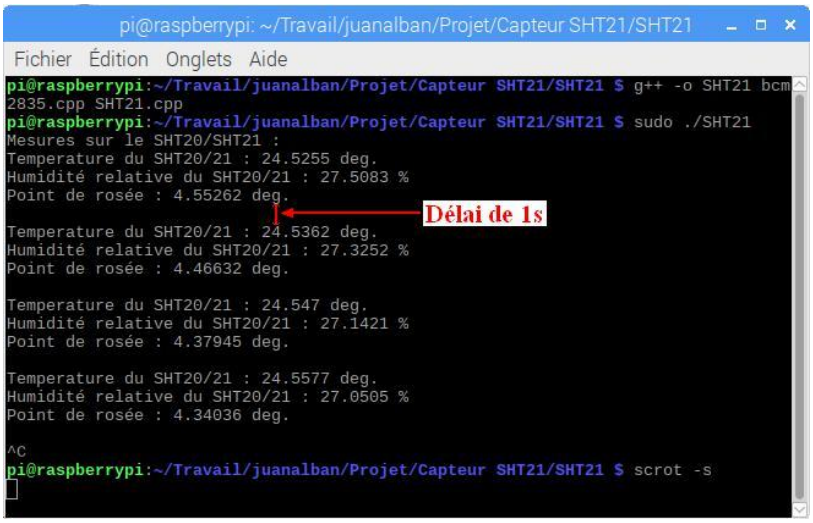


J'utilise donc deux boucles for. Pour résumer, la première boucle for permet d'enregistrer un certain nombre d'acquisitions de températures, d'humidités et de points de rosée

Dans le programme, on peut utiliser des tableaux de valeurs, ici MT[i] RH[i] et CPR[i], ils contiennent les valeurs captées (Température / Humidité Relative / Point de Rosée)

En haut du code, en commentaires (//) se trouve la ligne de commande à entrer dans le terminal du RPI3 pour lancer la compilation du programme : **g++ -o SHT21 bcm2835.cpp SHT21pp.c**

Une fois ce programme lancé avec **./SHT21** voilà le résultat dans un terminal :



Le « Délai de 1s » est paramétrable en modifiant cette ligne : `delay(1000);`

Le fonctionnement est exactement le même pour nos deux autres capteurs, SHT31 et Si7021. La seule différence dans leur programme est le nom dans la console « Temperature du **SHT20/21** ».

En faisant chauffer – avec le doigt – le capteur, je me suis aperçu qu'il réagissait correctement, que ce soit pour la température ou pour l'humidité.

Améliorations :

Le problème dans ce programme, c'est que les valeurs mesurées sont stockées sur plusieurs variables, qui sont directement affichées, puis remplacées à chaque boucle. On ne peut donc pas retrouver ces valeurs par un autre biais. Il est donc intéressant d'améliorer ce programme pour permettre un stockage durable des données. Le programme final du projet devra être créé par un étudiant IR, grâce auquel les valeurs captées seront envoyées sur une base de données, et consultables par un technicien Crossdock.

Couleurs :

En changeant la valeur (ex. : 35) dans \E[35;1mExemple\E[m on peut changer la couleur du mot

« Exemple » apparaissant dans le Terminal. Je vais utiliser cette ligne de code à plusieurs endroits dans le 2ème programme.

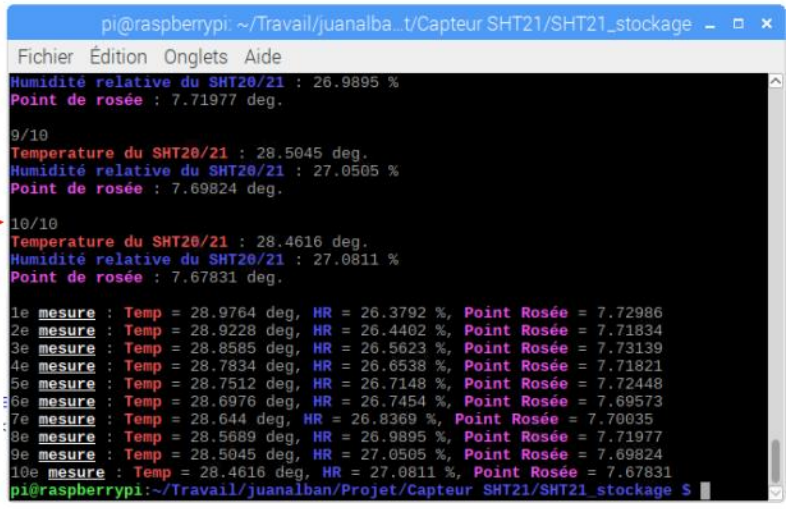
Exemple : la ligne *cout<< "\E[32;1mBONJOUR"<<endl;* affichera **BONJOUR** en vert dans la console.

Valeur	Couleur
31	Rouge
32	Vert
33	Jaune
34	Bleu
35	Rose
36	Cyan
37	Blanc

Programme modifié et utilisation des couleurs

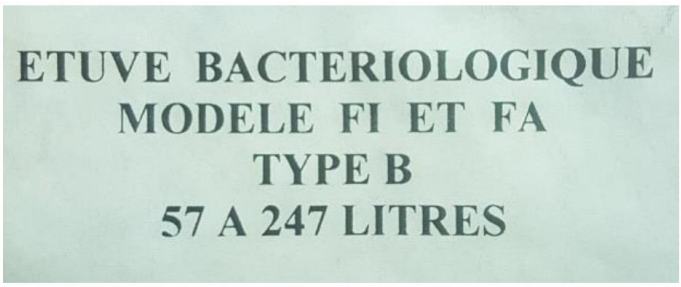
Le programme, une fois modifié, me permet de récupérer des valeurs sur le Bus I2C, en ayant un aperçu du nombre de mesures :

Compteur



4) Tests en étuve

Les capteurs étant fonctionnels, je dois déterminer le quel est le plus précis. Pour cela, le laboratoire de physique m'a prêté une étuve. Cet appareil est une sorte d'enceinte climatisée, qui permet de garder une atmosphère constante, pour avoir une référence pour les tests.



Fabricant : FIRLABO
Plage de températures : inconnue
Chauffe : Résistance 300W

Mes tests n'ont pas l'ambition d'être extrêmement précis, ils me permettront juste de départager au mieux les trois capteurs. Ces derniers ont déjà été testés par des entreprises spécialisées, et je ne peux pas réaliser ce genre de tests avec mes compétences, et le matériel à ma disposition.

Comptes rendus de tests en étuve (réglée sur 24°C) :

SHT21

```
170e mesure : Temp = 23.3994 deg, HR = 34.3748 %, Point Rosée = 6.79354
171e mesure : Temp = 23.3672 deg, HR = 34.4053 %, Point Rosée = 6.77825
172e mesure : Temp = 23.4745 deg, HR = 34.2832 %, Point Rosée = 6.82052
173e mesure : Temp = 23.7104 deg, HR = 33.8865 %, Point Rosée = 6.85763
174e mesure : Temp = 23.882 deg, HR = 33.6423 %, Point Rosée = 6.90238
175e mesure : Temp = 23.9249 deg, HR = 33.5203 %, Point Rosée = 6.88691
176e mesure : Temp = 23.9249 deg, HR = 33.5508 %, Point Rosée = 6.90017
177e mesure : Temp = 23.8927 deg, HR = 33.6423 %, Point Rosée = 6.91176
178e mesure : Temp = 23.8606 deg, HR = 33.7034 %, Point Rosée = 6.91004
179e mesure : Temp = 23.8177 deg, HR = 33.7644 %, Point Rosée = 6.89889
180e mesure : Temp = 23.7748 deg, HR = 33.8254 %, Point Rosée = 6.88768
181e mesure : Temp = 23.7211 deg, HR = 33.8865 %, Point Rosée = 6.86702
182e mesure : Temp = 23.6675 deg, HR = 33.9475 %, Point Rosée = 6.84629
183e mesure : Temp = 23.6246 deg, HR = 34.0085 %, Point Rosée = 6.8349
184e mesure : Temp = 23.5817 deg, HR = 34.1001 %, Point Rosée = 6.83649
185e mesure : Temp = 23.5388 deg, HR = 34.1611 %, Point Rosée = 6.82496
186e mesure : Temp = 23.4959 deg, HR = 34.2222 %, Point Rosée = 6.81337
187e mesure : Temp = 23.453 deg, HR = 34.2832 %, Point Rosée = 6.80172
188e mesure : Temp = 23.3994 deg, HR = 34.3442 %, Point Rosée = 6.78061
189e mesure : Temp = 23.3779 deg, HR = 34.4053 %, Point Rosée = 6.78765
190e mesure : Temp = 23.4959 deg, HR = 34.2832 %, Point Rosée = 6.83932
```

SHT31

```
170e mesure : Temp = 22.9011 deg, HR = 37.9294 %, Point Rosée = 7.79279
171e mesure : Temp = 23.056 deg, HR = 37.7173 %, Point Rosée = 7.84783
172e mesure : Temp = 23.3551 deg, HR = 37.1481 %, Point Rosée = 7.88941
173e mesure : Temp = 23.4939 deg, HR = 36.9192 %, Point Rosée = 7.92144
174e mesure : Temp = 23.51 deg, HR = 36.9329 %, Point Rosée = 7.94107
175e mesure : Temp = 23.4939 deg, HR = 37.0077 %, Point Rosée = 7.95663
176e mesure : Temp = 23.4379 deg, HR = 37.1038 %, Point Rosée = 7.94515
177e mesure : Temp = 23.3818 deg, HR = 37.1893 %, Point Rosée = 7.92934
178e mesure : Temp = 23.3551 deg, HR = 37.351 %, Point Rosée = 7.96949
179e mesure : Temp = 23.299 deg, HR = 37.5006 %, Point Rosée = 7.97856
180e mesure : Temp = 23.2269 deg, HR = 37.5677 %, Point Rosée = 7.94097
181e mesure : Temp = 23.1708 deg, HR = 37.7539 %, Point Rosée = 7.96391
182e mesure : Temp = 23.1147 deg, HR = 37.85 %, Point Rosée = 7.95156
183e mesure : Temp = 23.072 deg, HR = 37.9477 %, Point Rosée = 7.95153
184e mesure : Temp = 22.9999 deg, HR = 38.056 %, Point Rosée = 7.92945
185e mesure : Temp = 22.9438 deg, HR = 38.1262 %, Point Rosée = 7.90674
186e mesure : Temp = 22.9305 deg, HR = 38.2025 %, Point Rosée = 7.92426
187e mesure : Temp = 23.0293 deg, HR = 37.9828 %, Point Rosée = 7.9272
188e mesure : Temp = 23.3684 deg, HR = 37.3281 %, Point Rosée = 7.9723
189e mesure : Temp = 23.5233 deg, HR = 37.113 %, Point Rosée = 8.02439
190e mesure : Temp = 23.5927 deg, HR = 37.0306 %, Point Rosée = 8.05312
```

Si7021

```
170e mesure : Temp = 22.3269 deg, HR = 37.8538 %, Point Rosée = 7.25397
171e mesure : Temp = 22.3054 deg, HR = 37.9148 %, Point Rosée = 7.25848
172e mesure : Temp = 22.2733 deg, HR = 37.9453 %, Point Rosée = 7.24168
173e mesure : Temp = 22.2518 deg, HR = 37.9758 %, Point Rosée = 7.23439
174e mesure : Temp = 22.2196 deg, HR = 38.0063 %, Point Rosée = 7.21756
175e mesure : Temp = 22.284 deg, HR = 37.8538 %, Point Rosée = 7.2159
176e mesure : Temp = 22.4985 deg, HR = 37.4875 %, Point Rosée = 7.26406
177e mesure : Temp = 22.6379 deg, HR = 37.1824 %, Point Rosée = 7.26808
178e mesure : Temp = 22.6915 deg, HR = 37.0603 %, Point Rosée = 7.26748
179e mesure : Temp = 22.6915 deg, HR = 37.0298 %, Point Rosée = 7.25544
180e mesure : Temp = 22.7023 deg, HR = 37.0298 %, Point Rosée = 7.26493
181e mesure : Temp = 22.6486 deg, HR = 37.0603 %, Point Rosée = 7.22951
182e mesure : Temp = 22.6486 deg, HR = 37.0908 %, Point Rosée = 7.24154
183e mesure : Temp = 22.6165 deg, HR = 37.1519 %, Point Rosée = 7.23709
184e mesure : Temp = 22.5843 deg, HR = 37.1824 %, Point Rosée = 7.2206
185e mesure : Temp = 22.5521 deg, HR = 37.2129 %, Point Rosée = 7.2041
186e mesure : Temp = 22.5199 deg, HR = 37.2739 %, Point Rosée = 7.19955
187e mesure : Temp = 22.4878 deg, HR = 37.335 %, Point Rosée = 7.19495
188e mesure : Temp = 22.477 deg, HR = 37.3655 %, Point Rosée = 7.19739
189e mesure : Temp = 22.4341 deg, HR = 37.4265 %, Point Rosée = 7.18322
190e mesure : Temp = 22.402 deg, HR = 37.457 %, Point Rosée = 7.16661
```

Pour ces trois relevés, le temps entre chaque mesure est de **5 minutes**.

Sous **Excel**, on peut entrer les valeurs mesurées, et obtenir des résultats de calculs :

SHT21

	A	B	C	D	E	F
1	T	HR	Moyenne T	Moyenne HR	Variation T (%)	Variation HR (%)
2	23,3994	34,3748	23,68521	33,647865	2,386678763	4,231637306
3	23,3672	34,4053				
4	23,4745	34,2832				
5	23,7104	33,8865				
6	23,882	33,6423				
7	23,9249	33,5203				
8	23,9249	33,5508				
9	23,8927	33,6423				
10	23,8606	33,7034				
11	23,8177	33,7644				
12	23,7748	33,8254				
13	23,8177	33,8865				
14	23,7748	33,9475				
15	23,7211	33,0085				
16	23,6675	33,1001				
17	23,6246	33,1611				
18	23,5817	33,2222				
19	23,5388	33,2832				
20	23,4959	33,3442				
21	23,453	33,4053				

Le calcul des moyennes est simple car il est réalisable grâce à une commande :
Calcul Moyenne T : =MOYENNE(A2:A21)
Calcul Moyenne HR : =MOYENNE(B2:B21)

Le calcul des variations se fait grâce à plusieurs commandes, MAX(…) et MIN(…), qui récupèrent les valeurs extrêmes d'une colonne de valeurs.
Calcul Variation T : =(MAX(A2:A21)-MIN(A2:A21))/MIN(A2:A21)*100
Calcul Variation HR : =(MAX(B2:B21)-MIN(B2:B21))/MIN(B2:B21)*100

J'entre également les relevés pour les deux autres capteurs, et j'obtiens un comparatif :

Capteur	Moyenne T	Moyenne HR	Variation T (%)	Variation HR (%)
SHT21	23,64445 °C	33,997865 %	2,386678763	2,640191168
SHT31	23,29245 °C	37,477005 %	3,019942274	3,475969143
Si7021	22,49258 °C	37,42194 %	2,172406344	2,637065283

On peut voir que les relevés du SHT21 se rapprochent le plus de la réalité en terme de température (étuve réglée à 24°C). Le capteur qui subit le plus de variations de températures et d'humidité est le SHT31. Néanmoins, les trois capteurs obtiennent des résultats corrects. Un entretien avec M Bijou, le responsable de Crossdock, nous a permis de nous assurer que ces valeurs lui convenaient.

5) Choix

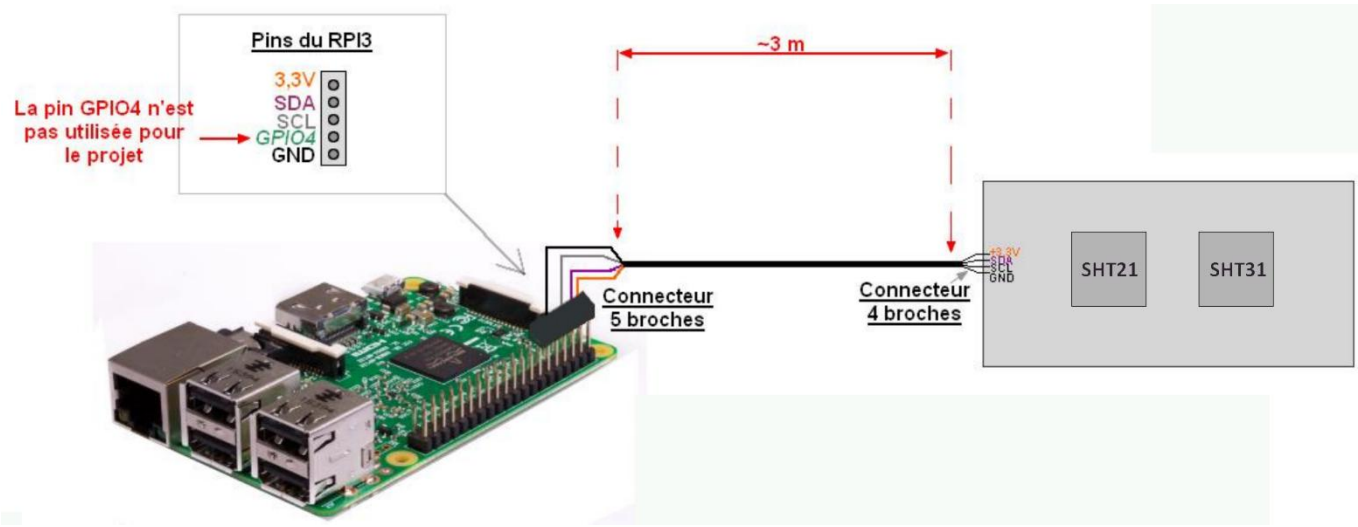
Le SHT31 pouvant changer d'adresse I^2C, j'ai décidé de le garder pour le projet. Il sera associé à un capteur SHT21 durant le projet, car celui-ci obtient des résultats proches de la réalité. En fait, chacun des capteurs semble respecter les valeurs indiquées dans leur documentation.

Egalement, ce choix ne perturbera pas le travail des étudiants IR, qui avaient déjà commencé à créer des programmes pour le SHT31, en attendant que je fasse les tests.

III) Circuits imprimés

1) Interconnexion des appareils

Pour savoir comment schématiser le montage, il faut d'abord réfléchir à comment les appareils seront connectés entre eux. L'installation sera du même type que celle prévue au départ, à savoir l'utilisation : - d'une carte Raspberry Pi 3
- d'un capteur SHT21 et d'un SHT31



Il me faudra utiliser cette structure :

Comme la Pin GPIO4 du RPI3 sépare le GND des trois autres broches, j'aurai besoin d'un connecteur 5 broches, bien qu'il ne me faille que 4 broches pour alimenter les capteurs. En sortie du câble (sur la droite) il n'y a donc que quatre fils, +3,3V, SDA / SCL et GND. Il a été défini qu'il faudrait un câble d'environ 3m de long, pour connecter les capteurs à la carte Raspberry Pi 3.

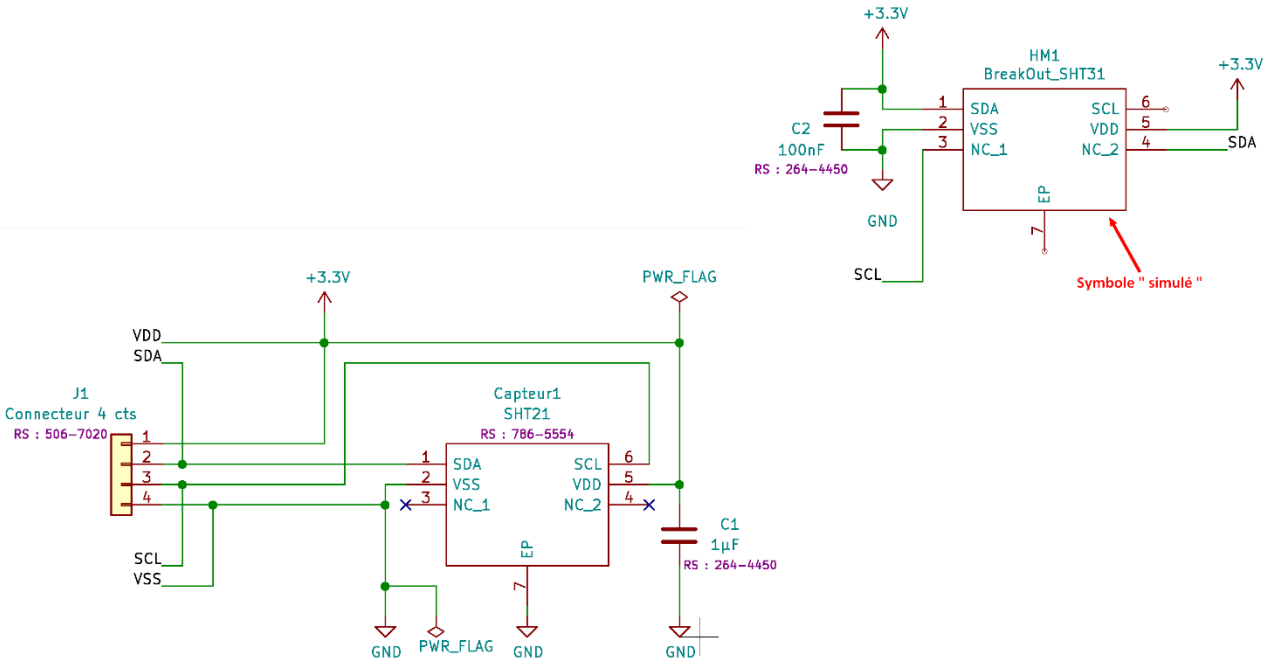
2) Conception du schéma

a) Schéma KiCad

Logiciel utilisé pour le projet : KiCad 5

Ce logiciel intègre des " sous-logiciels " dont Eeschema, qui permet de tracer un schéma, et Pcbnew qui permet d'éditer ou créer des Circuits Imprimés (CI).

Lors de la réalisation d'un circuit imprimé, la première étape est de tracer un schéma.
Les composants sur KiCad (appelés « Symboles ») m'ont été donnés par M Hortolland.
Lorsque j'ai commencé à créer le schéma, j'avais une contrainte de temps. En effet je devais aller chez Crossdock pour installer des prototypes de capteurs. À cause de ce manque de temps, M Hortolland a modifié le symbole du SHT21 pour créer celui du SHT31 (voir Symbole " simulé " sur le schéma) :



Des composants importants, que je n'avais pas utilisé lors de mes tests de capteurs, ont dû être ajoutés : les condensateurs de découplage, notés C1 et C2. Ils agissent ici comme une réserve d'énergie « de secours ». Le SHT21 sera proche du connecteur, il faut que son condensateur de découplage soit plus important que celui du SHT31 : **1µF** pour C1 contre **100nF** pour C2.

Les PWR_FLAG 

permettent au logiciel d'indiquer que telle ou telle broche est bien une broche d'alimentation. C'est pour cela qu'on en retrouve régulièrement, connectés aux broches d'alimentation telles que le GND ou le +3,3V

Justement, je peux maintenant passer au « test des règles électriques » géré par KiCad. Grâce à ce test, on peut avoir un visuel sur d'éventuelles erreurs électriques : fil mal connecté, oubli d'un PWR_FLAG, etc … Il suffit de cliquer sur ce bouton  et de lancer le test. Dans mon cas, j'ai 3 "erreurs" :

Problème de conflit entre pins. Sévérité: warning
@(182,88 mm, 80,01 mm): Pin 1 (Power output) du composant #FLG0102 connectée à
@(179,07 mm, 101,60 mm): pin 5 (Bidirectionnel) du composant Capteur1 (net 3).

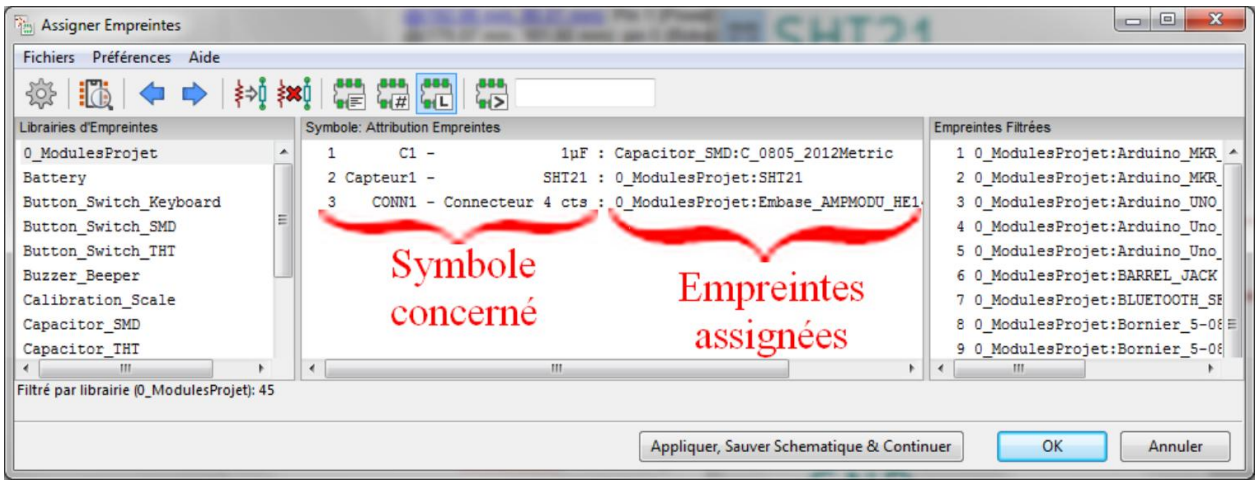
Problème de conflit entre pins. Sévérité: warning
@(148,59 mm, 101,60 mm): Pin 2 (Bidirectionnel) du composant Capteur1 connectée à
@(152,40 mm, 118,11 mm): pin 1 (Power output) du composant #FLG0101 (net 5).

Problème de conflit entre pins. Sévérité: warning
@(152,40 mm, 118,11 mm): Pin 1 (Power output) du composant #FLG0101 connectée à
@(163,83 mm, 116,84 mm): pin 7 (Bidirectionnel) du composant Capteur1 (net 5).

Ce ne sont pas réellement des erreurs, mais plutôt des mises en garde (warnings). Ici, ces warnings m'indiquent que le capteur possède des pins bidirectionnelles, ce qui peut être source d'erreur, mais dans mon cas ce n'est pas dérangeant, il faudrait juste modifier ce paramètre dans le symbole depuis KiCad.

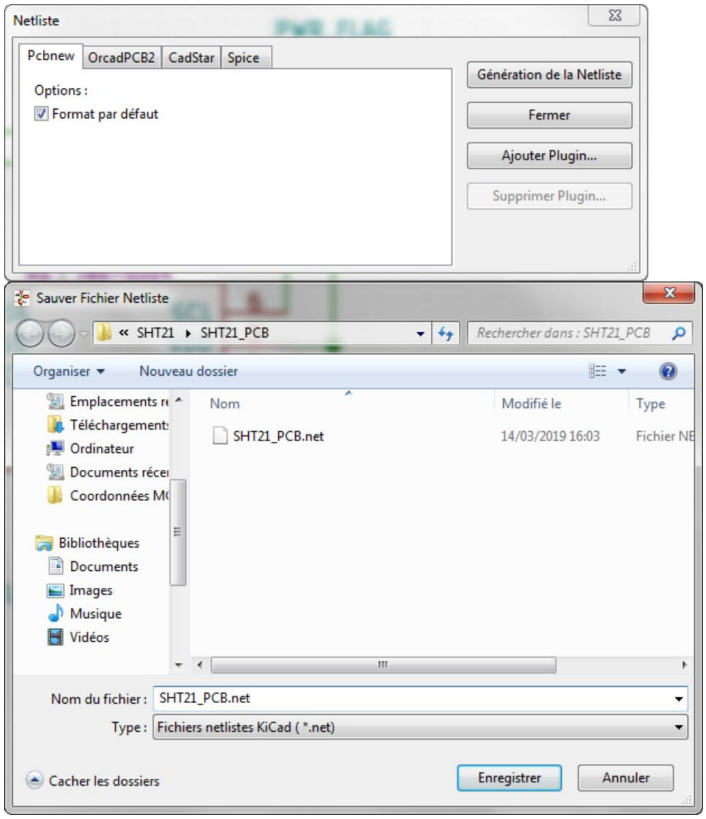
b) Association des empreintes

Les empreintes sont la correspondance entre des symboles du schéma, et un autre objet, qui lui possède de vraies dimensions, et un affichage 3D. Grâce à cela, on va pouvoir transformer notre schéma Eeschema, en « composants 3D » utiles au routage. Outils => Assigner Empreintes =>



En général, les empreintes peuvent être trouvées sur le site du fournisseur d'un composant. Dans d'autres cas, elles peuvent être incluses directement dans KiCad, comme pour les condensateurs par exemple.

Au départ, le champ "Empreintes assignées" n'est pas rempli, il faut cliquer sur le "Symbole concerné" et sélectionner une empreinte dans la catégorie "Empreintes Filtrées" (à droite), qui est un sousensemble des librairies de la partie gauche "Librairie d'Empreintes". Une fois ceci fait, il faut faire la transition entre Eeschema et Pcbnew (pour construire le CI). Pour cela on utilise la « netliste », qui va posséder toutes les informations utiles, comme les noms de broches, les empreintes associées … Cette netliste est sauvegardée depuis Eeschema, puis importée dans Pcbnew, et doit être ré-importée à chaque fois qu'on apporte une modification au schéma.



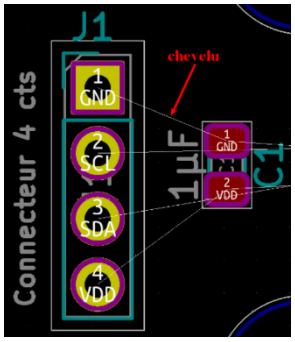
Enregistrement de la netliste

Une fois enregistrée (ici au nom de SHT21_PCB.net), je peux passer au routage de la carte, donc la conception du circuit imprimé.

3) Conception du Circuit Imprimé (CI)

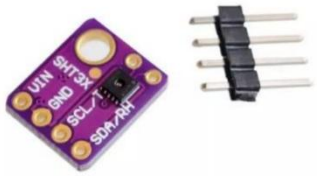
a) Utilisation de Pcbnew

Après avoir paramétré les largeurs de pistes, largeurs des vias (trous de perçage) et autres, j'importe dans Pcbnew la netliste, créée sur Eeschema. Cela a pour conséquence de m'importer tous les composants du schéma, ainsi que de m'indiquer quelles broches connecter ensemble, grâce au chevelu.

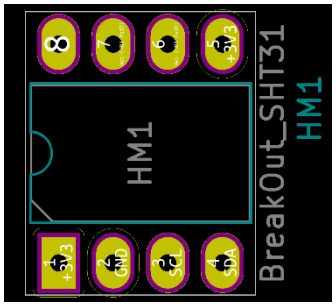


3)b) Routage du Circuit Imprimé

L'empreinte étant l'objet définissant les dimensions réelles des composants du circuit imprimé, il a fallu utiliser une empreinte spécifique pour le SHT31, sachant que celui-ci n'était pas, au départ, un composant CMS. En effet, j'ai utilisé dans un premier temps un capteur « tout fait » :

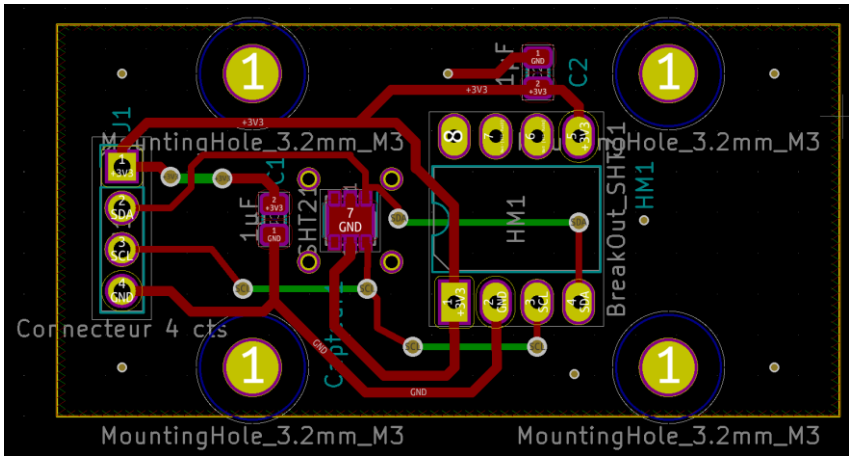


Sur le routage, j'ai utilisé une empreinte d'un composant quelconque, qui possède 8 pin et dont l'espacement entre chaque pin est le même que pour le SHT31 ci-dessus :



On voit que les pin SDA, SCL, GND et +3V3 sont bien associées aux pins du composant 8 broches, cela grâce à l'association des empreintes faite plus tôt

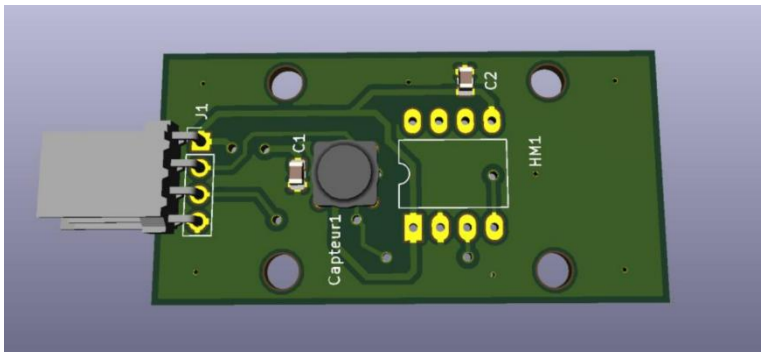
Le routage, une fois terminé ressemble à cela :



Les pistes de couleur verte sont les pistes routées sur la face « dessous » du PCB. Ici, chacune d'elles est droite, ce qui permet de fabriquer le circuit imprimé au lycée, en utilisant des fils pour ne pas avoir à router la face arrière. Les fils soudés remplacent les pistes routées.
Les pistes de couleur rouge, elles, sont routées directement grâce à du perchlorure.

L'intérêt de router une première carte au lycée est de pouvoir apporter un prototype dans l'entreprise, sans avoir à attendre qu'un sous-traitant nous envoie une carte, nous avons donc un gain de temps et d'argent pour ce premier prototype.

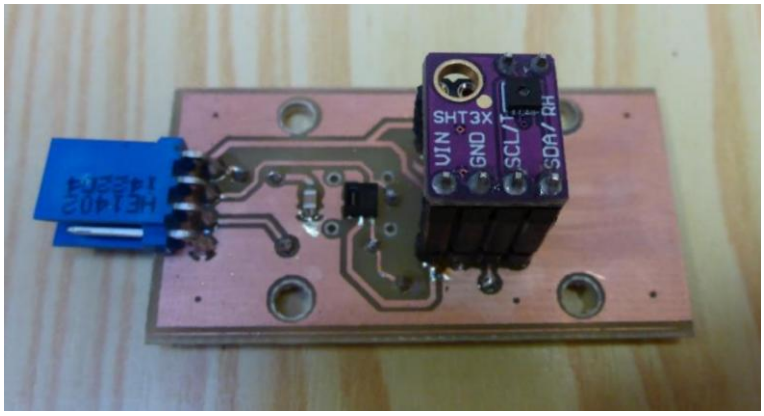
La carte devrait ressembler à cela :



Affichage 3D sur Pcbnew

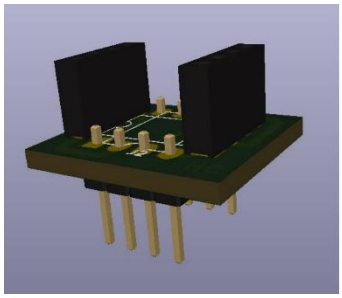
On voit que l'empreinte HM1 sur la droite possède bien 8 pins, et c'est dans ces trous de perçage que sera soudée l'embase pour le breakout SHT31.

La carte, routée par M Hortolland :



3)c) Modifications

Pendant que je continuais mon projet, M. Defrance, professeur IR, a réalisé des tests sur la carte routée. Il trouvait des valeurs qui n'avaient pas de sens (des températures beaucoup trop élevées). Nous avons donc décidé de modifier l'aspect de la carte, et M Hortolland a créé un adaptateur :



Cet adaptateur permet de connecter à la carte un autre capteur SHT31, qui a également été routé par M. Hortolland

IV) Communication avec l'entreprise

1) Revue au lycée

Le 20 mars 2019, l'ensemble des membres du projet (dont l'équipe Crossdock 2) s'est réunie pour présenter un diaporama à M Bijou, responsable de l'entreprise.

Cette revue était intéressante car cela nous a permis de lui montrer notre avancement. Egalement, nous avons pu éclaircir certaines zones d'ombre, notamment la distance du câble à utiliser (c'est de là que nous avons choisi 3m de câble par capteur). D'une certaine manière, le fait d'avoir le responsable devant nos yeux, nous a permis de prendre conscience de notre retard sur certains points, et nous a motivé à rendre un travail plus soigné, notamment au niveau des tests.

2) a) Intervention dans l'entreprise Crossdock

Le vendredi 5 avril 2019, je suis allé, avec deux professeurs M Hortolland et M Defrance, dans l'entreprise Crossdock. Reçu par M Bijou, nous avons convenu avec lui de l'installation de plusieurs capteurs dans son entrepôt :

Photo 1

Photo 2

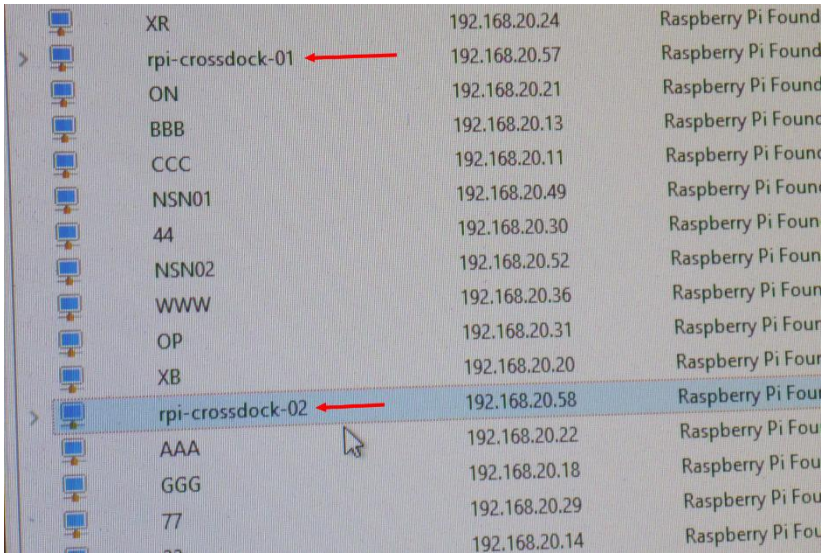
Nous avons utilisé des colliers Rilsan pour fixer les capteurs à des montants de l'entrepôt (**Photo 2**). On les place dans la zone de stockage, de manière à ne pas gêner le travail des employés, soit à environ 20cm au-dessus des cartons. Nous avons apporté deux cartes (**Photo 1**), soit 4 capteurs en tout, et nous avons demandé à M Bijou si on pouvait lui emprunter un de ses capteurs d'origine, pour pouvoir comparer "nos" mesures aux "siennes".

On peut voir sur la **Photo 1** que le capteur utilisé est celui avec l'adaptateur, comme expliqué plus tôt.

2) b) Connexion au réseau de Crossdock

Nous avons connecté nos capteurs à deux **Raspberry Pi 3** plus haut :

À présent, M Defrance peut rechercher son Raspberry Pi 3 sur le réseau grâce au logiciel IpScanner :



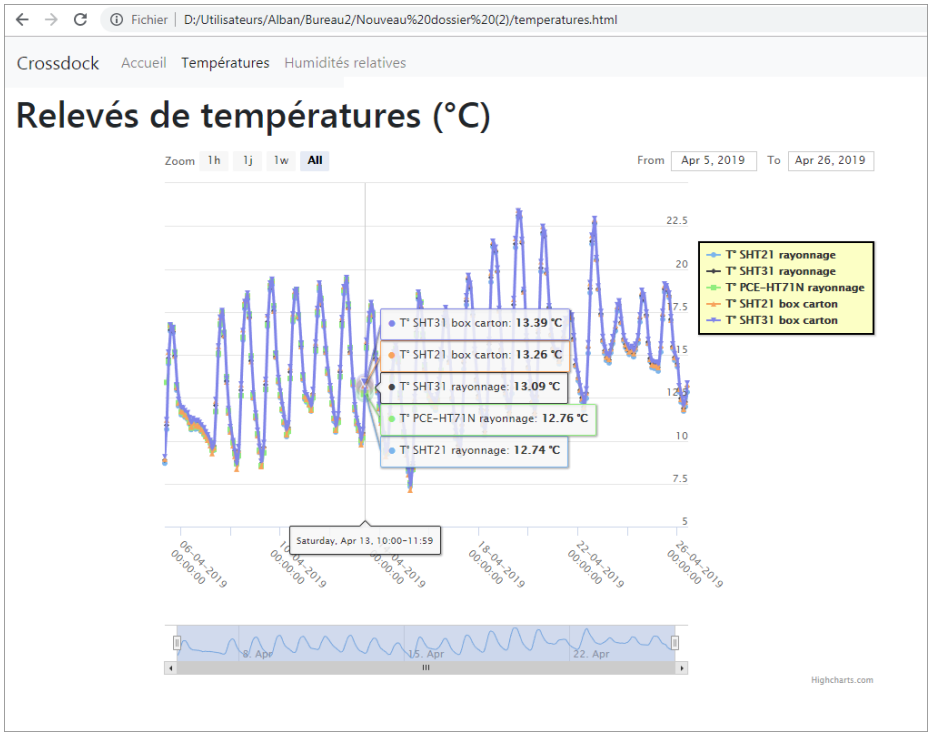
Les deux **RPI3** sont bien **détectés**. D'ailleurs on voit que M Bijou utilise déjà des Raspberry Pi dans son entreprise, donc en ajouter de nouvelles ne posera aucun problème. M Defrance a réunis les deux programmes que j'avais créé, et le sien enregistrera directement dans un fichier les **relevés des températures et humidité** des 4 capteurs (2 par RPI).

2) c) Conclusion de l'intervention

Nous avons convenu avec M Bijou que nous reviendrions après les vacances pour récupérer les relevés des capteurs, afin de pouvoir comparer l'**efficacité** de chaque capteur dans des **conditions réelles**. Cette intervention m'a permis de confirmer la **taille du câble** (3m) ainsi que de choisir le **positionnement** final des capteurs pour lequel nous avions un doute.

2) d) Résultats des tests

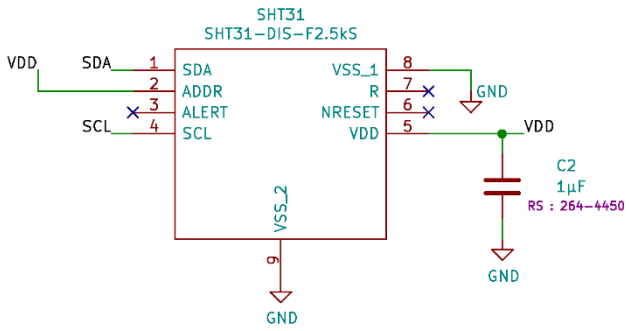
Après environ deux semaines, nous avons récupéré les capteurs. Le professeur d'Informatique et Réseaux M. Defrance a créé un « micro-site » web qui permet d'avoir un visuel sur les résultats, notamment grâce à des courbes. L'avantage d'utiliser un site web plutôt que des courbes Excel est la rapidité d'ouverture : le fichier Excel met beaucoup de temps à s'ouvrir, alors que le site, lui, est très fluide. Ce site sera disponible dans mon dossier de projet, en voilà un visuel :



On peut accéder à des onglets pour afficher les températures ou les humidités relatives.
Egalement, on a accès à des marqueurs pour savoir à un instant T, quelles valeurs ont été captées par chaque capteur.
Globalement, chacun des capteurs a suivi la même courbe.

3) Modifications

Ayant plus de temps après l'intervention, mon professeur M. Hortolland a pu créer un vrai symbole pour le SHT31, en remplacement du symbole « simulé » :



Ce symbole-ci est beaucoup plus lisible, surtout que deux de ses Pins sont connectées au VDD, ce qui est difficile à comprendre lorsqu'on utilise un mauvais symbole.

V) Solution sans-fil

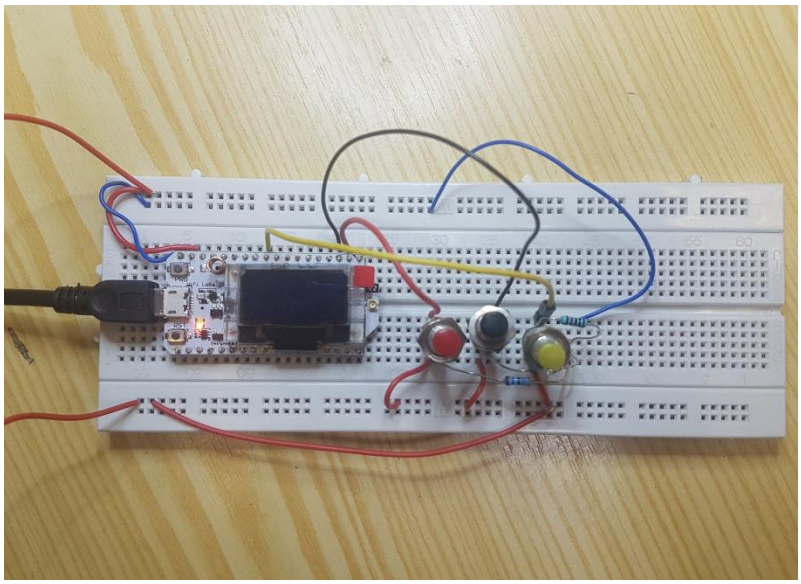
1) Heltec ESP32 WiFi

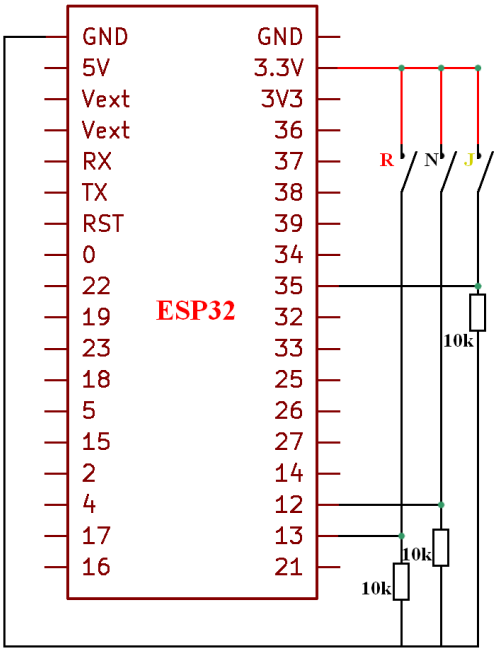
Nous avons décidé d'utiliser un ESP32 ***WiFi LoRa 32*** de chez **Heltec**. Celui-ci possède des entrées / sorties digitales et des entrées analogiques. Il peut également communiquer en I2C, et possède son propre écran OLED (commandé en I_2C).

a) Test des broches

Pour m'assurer du fonctionnement des broches, j'ai créé un programme sur l'IDE d'Arduino. Le logiciel peut directement envoyer le programme sur l'ESP32, comme il le ferait avec une Arduino.

J'utilise 3 boutons poussoir de couleur différente et 3 résistances de pull-down.



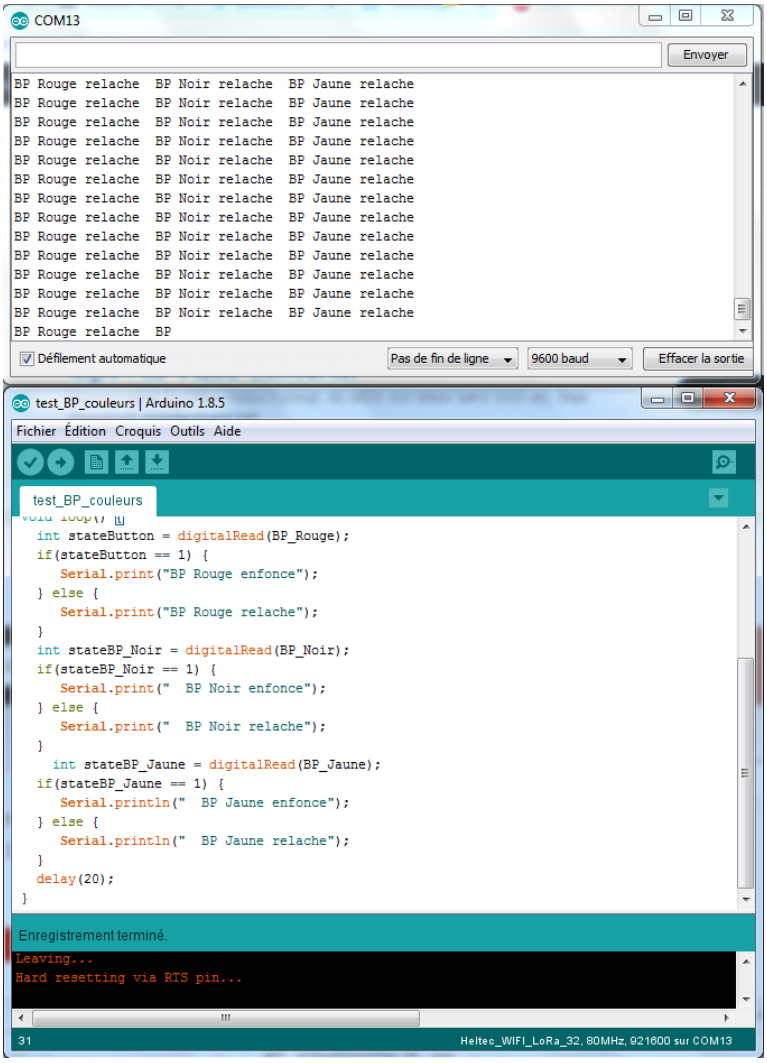


b) Communication avec l'ESP32

J'ai utilisé une librairie pour faire fonctionner l'écran de l'ESP32 :

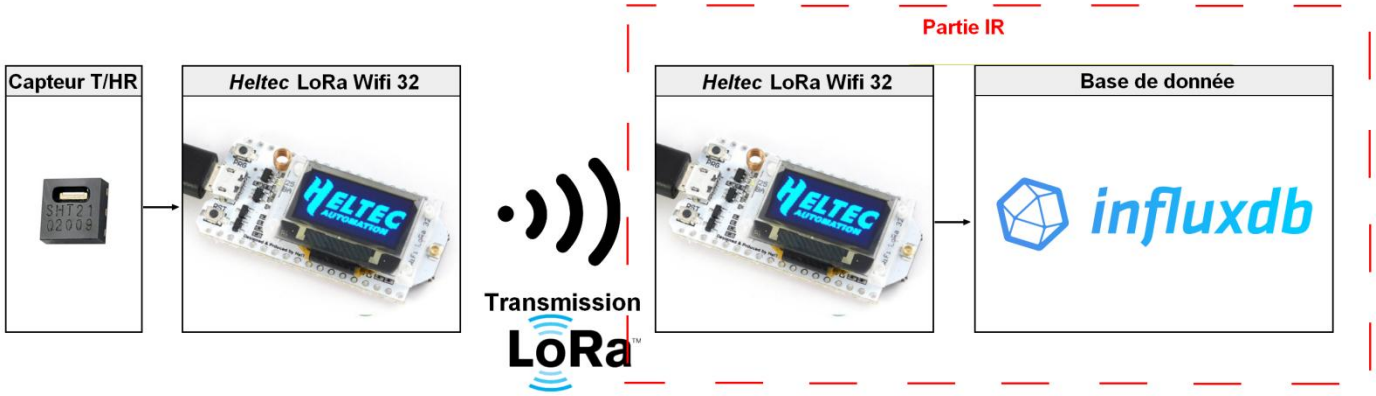
Je n'ai pas utilisé l'écran dans mon programme car son fonctionnement a été testé par un autre étudiant. En effet mon programme ne me servira qu'à confirmer quelles broches choisir pour connecter mes boutons poussoirs.

Une fois le programme télé-versé :



Les capteurs ayant été testés avec un prototype « fait-main », je peux passer à l'élaboration de la solution sans-fil. Comme l'étudiant **IR1** doit configurer un module *Heltec WiFi 32*, c'est le même que j'utiliserai dans ma solution.

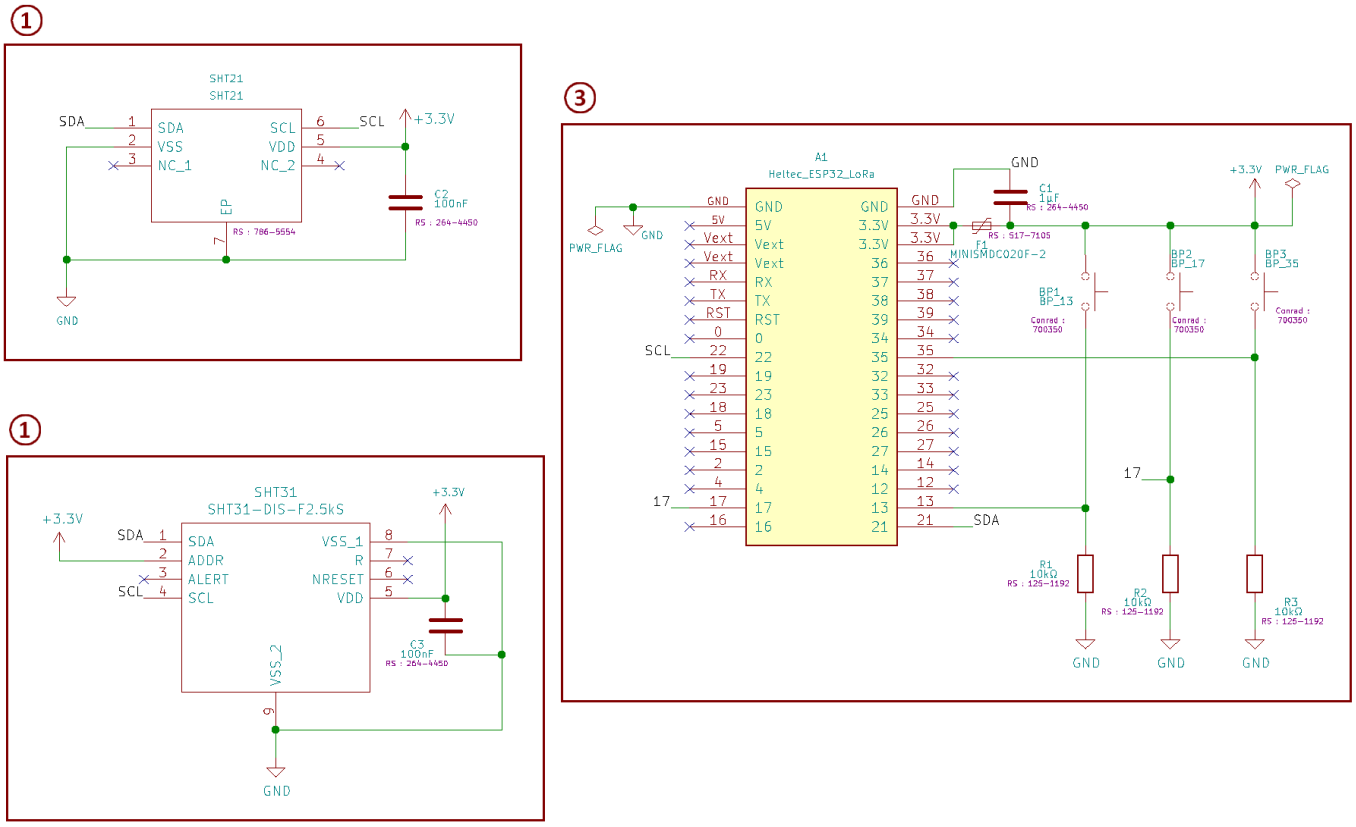
Pour rappel, voilà le visuel de la solution :



La carte finale aura des caractéristiques similaires à celle de la solution filaire :

- une communication I^2C
- deux capteurs, SHT21 et SHT31

Comme le module Heltec possède un écran, je n'ai pas à en ajouter un sur la carte. La particularité de cette solution est qu'elle affiche sur l'écran, les températures, humidités et points de rosée en temps réel. Pour que le technicien Crossdock puisse choisir quelle valeur afficher, je dois munir la carte de boutons poussoirs. Nous avons décidé avec les membres du projet que la carte posséderait trois boutons.



En ① se trouve le capteur SHT21, avec son condensateur de découplage C2. On utilise un condensateur de 100nF, car il sera éloigné de l'alimentation sur le routage.

En ② se trouve le capteur SHT31, et son condensateur de découplage C3, également à 100nF. Pour éviter de laisser le fil de la broche ADDR à la masse, plus sensible aux parasites, j'ai connecté cette broche au +3.3V.

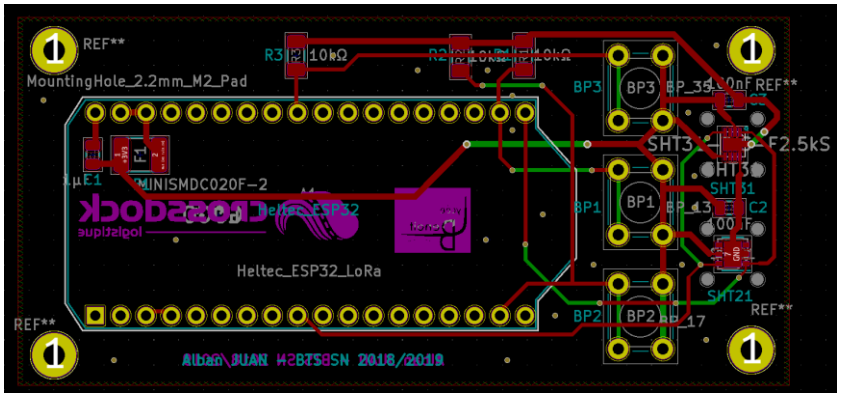
En ③ se trouve le symbole de la carte Heltec (qui correspond à une embase sur le routage). Les pins que j'utilise pour les boutons poussoir sont les broches 17, 13 et 35. Ces pins peuvent être utilisées comme entrées analogiques.
On retrouve les pins SDA et SCL, qui sont les broches utilisées pour la communication I^2C.
L'étudiant IR Clément PELLOUX m'a confirmé quelles broches utiliser pour le capteur. Les broches SDA et SCL étant déjà utilisées par l'écran OLED, on peut en utiliser d'autres :

- **SDA = pin 21**
- **SCL = pin 22**

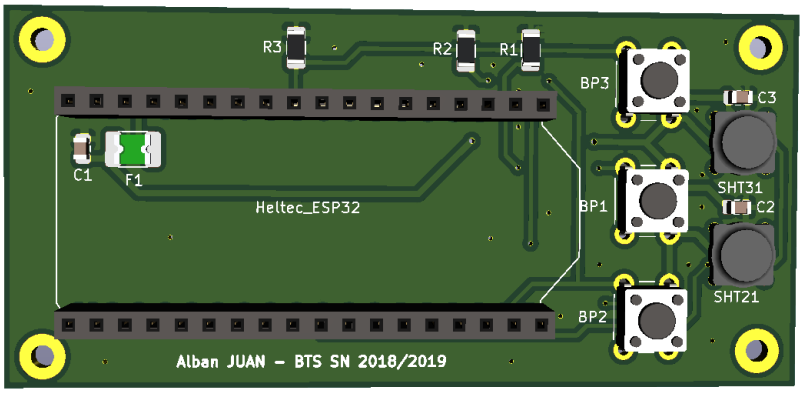
Le condensateur de découplage C1 fait 1µF, il est donc plus gros que les deux autres condensateurs du circuit, puisque celui-ci est au plus proche de l'alimentation. Les résistances R1, R2 et R3 permettent de ramener le potentiel à la masse (tirage bas / pull down). Cela permet d'être sûr que la tension sur le fil est de 0V lorsque l'on n'appuie pas sur le bouton poussoir.

On retrouve bien des PWR_FLAG pour indiquer à KiCad quelles sont les broches d'alimentation.
Une fois les empreintes associées aux symboles, je suis passé au routage de la carte

2) Routage



Sur la droite, on peut voir les deux capteurs avec des trous de perçage qui permettront d'enficher une protection. Cette protection est visible sur l'affichage 3D de la carte :



Le composant F1 est un fusible réarmable : le Littelfuse MINISMDC020F-2.
Ce fusible se trouve, comme le condensateur C1, au plus proche de l'alimentation 3.3V de la carte Heltec WiFi 32.

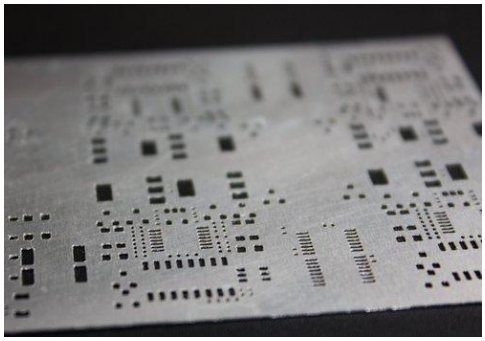
Après le test des règles électriques (DRC), la carte est terminée.

VI) Conception des cartes

Les deux cartes de mon projet étant routées, je peux extraire des fichiers de fabrication depuis KiCad. Ces fichiers appelés communément fichiers Gerber, permettent à un sous-traitant de concevoir nos PCB. Il faut également fournir au professionnel un Fichier de Perçage, qui contient d'autres information, telles que le type de mesure (pouce / centimètre) ou le type de fichier voulu (Gerber en l'occurrence).

1) Les sous-traitants

Par rapport aux entreprises françaises, des sites tels que JLCPCB ou SeeedStudio proposent des prix relativement faibles. Nous avons décidé de faire confiance à ces deux sites pour la conception des cartes. On peut également commander un « stencil » pour chaque circuit imprimé, qui est une sorte de pochoir, qui va permettre de déposer une fine couche de pate à braser sur les pastilles correspondantes :



Nom du site	**Avantage(s)**	**Inconvénient(s)**
SeeedStudio	Prix plus bas pour les PCB	Stencil beaucoup trop grands
JLCPCB	Stencil plus petits : plus écologique et moins cher	Panélisation plus compliquée Plus cher pour les PCB

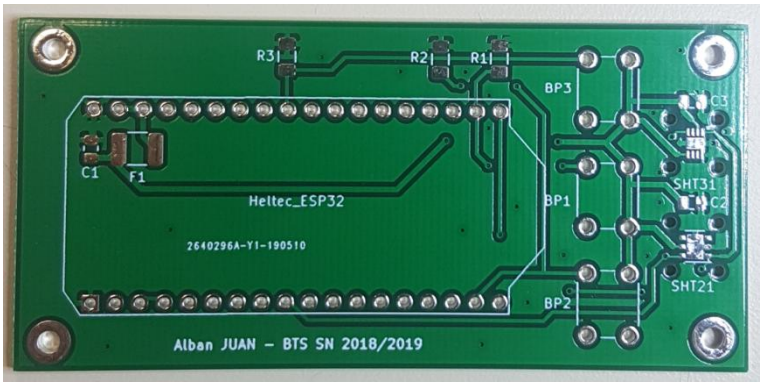
La panélisation consiste à demander au sous-traitant de créer une carte en plusieurs exemplaires, pour utiliser l'espace vide d'un PCB. Cela est particulièrement utile dans le cas d'une carte de petite taille, puisque les PCB commandés ont des dimensions minimales obligatoires, et la partie non utilisée du PCB est gaspillée.

2) Soudure des composants

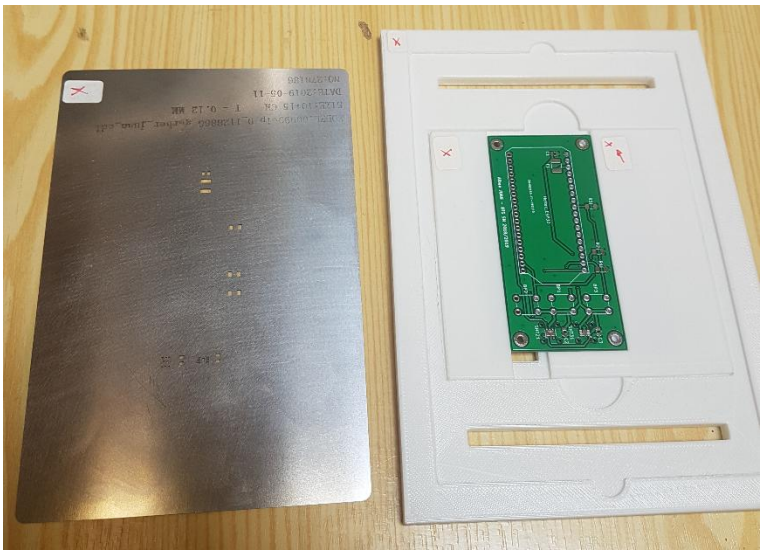
Lorsqu'on a reçu les PCB, on peut les préparer à la soudure. Pour cela, on utilise les stencils commandés, en vérifiant que les pastilles sont correctement placées sous les trous.

Lors d'une panélisation, il est conseillé de ne souder que une ou deux cartes à la fois, pour être sûr de ne pas décaler le stencil et ainsi créer de mauvaises soudures.

PCB de la version sans fil :



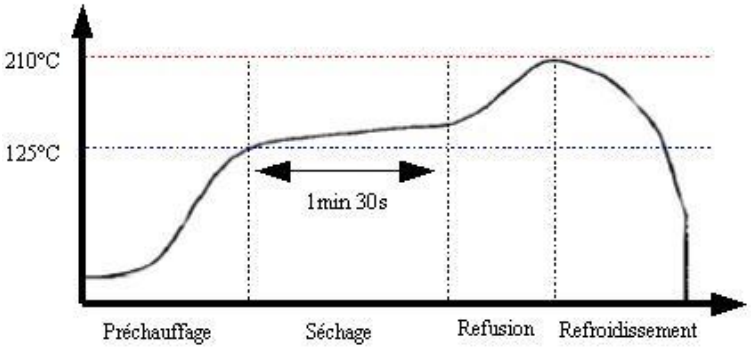
Je peux préparer la carte à la soudure grâce au stencil.



On voit sur la gauche le stencil, avec des trous aux endroits des pastilles présentes sur le PCB.
Il faut poser le stencil, et le fixer lorsqu'on est sûr qu'il est bien positionné.

Lorsqu'on a appliqué la pâte à braser, il faut retirer le stencil et placer les composants CMS minutieusement, notamment le SHT31, qui possède des pins très fines. Après une vérification au microscope (loupe binoculaire) de la pâte à braser sous les composants, on peut placer le PCB au four à refusion. Au lycée, on a utilisé un **F31114 CIF** :

Le four à refusion fonctionne sur un préchauffage lent, un séchage à une température fixe, puis une refusion, un pic de température. C'est à ce moment-là que la pâte à braser fusionne avec le composant CMS, avant que le four ne refroidisse :



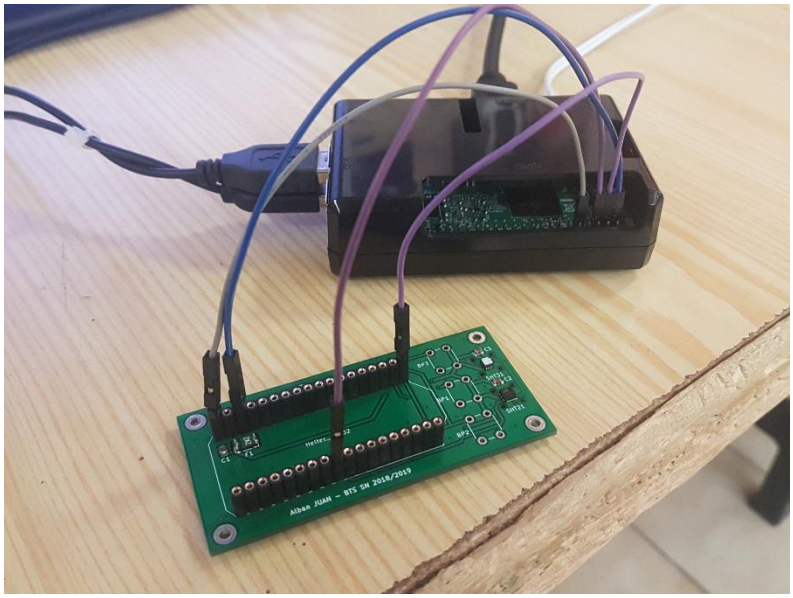
Au bout d'environ 5min, les composants CMS sont soudés. Il faut ensuite souder les composants traversant tels que les boutons poussoirs, connecteurs ou embases au fer à souder.

3) Problèmes rencontrés

Une fois les composants soudés, je me suis rendu compte que l'embase du Heltec ne correspondait pas aux dimensions du module. En effet, les broches étaient bien trop éloignées. M. Hortolland m'a aidé à adapter le module Heltec, en pliant ses broches. Cette version est un prototype, et il faudra commander d'autres PCB une fois les problèmes de routage corrigés.

4) Tests du bus I²C et montage

Les composants de la carte sans-fil sont soudés, je dois maintenant tester le fonctionnement du bus I2C.
Pour cela, j'utilise un Raspberry Pi 3 et des câbles :

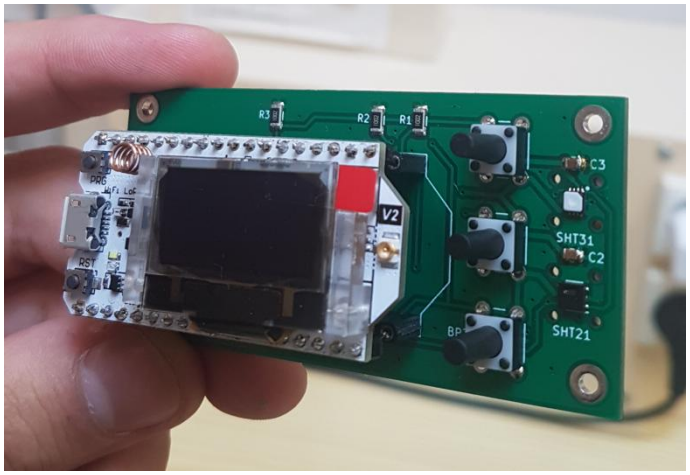


La carte est alimentée grâce au RPI3, et grâce à une commande, on peut savoir quelles adresses se trouvent sur le bus.

Cette commande est **sudo i2cdetect -y 1** et donne ce résultat :

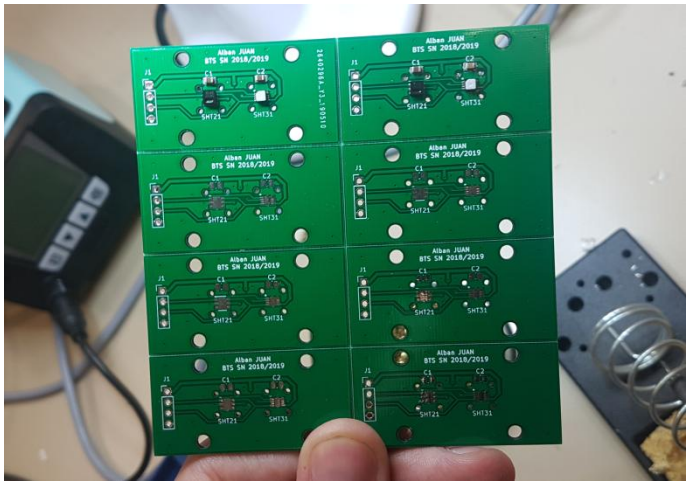
```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- 45 -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi ~ $
```

Les adresses 40 (SHT21) et 45 (SHT31) sont bien détectées sur le bus.
Je peux maintenant souder les composants restants, à savoir les boutons poussoirs et l'embase.
Le prototype de la solution sans-fil est à présent terminé :



5) Carte panélisée

La carte de la solution filaire est panélisée. Chacun des PCB de cette solution contient en fait huit cartes identiques :



Pour l'instant, je n'ai pas terminé le câblage de cette carte. Il sera terminé dès que possible.

VII) Phénomènes physiques

1) Les capteurs

Selon le capteur utilisé, la technologie interne est différente. Par exemple, pour un capteur de type PT100, la température agit sur la résistance du métal dont est composé le capteur, et grâce à des courbes connues, on sait combien vaut la température. Dans notre cas, le SHT21 et le SHT31 sont composés de transistors qui comparent la tension à deux endroits dans le capteur, cela nous donne une tension qui correspond à une température. On appelle cela un capteur à semi-conducteurs.

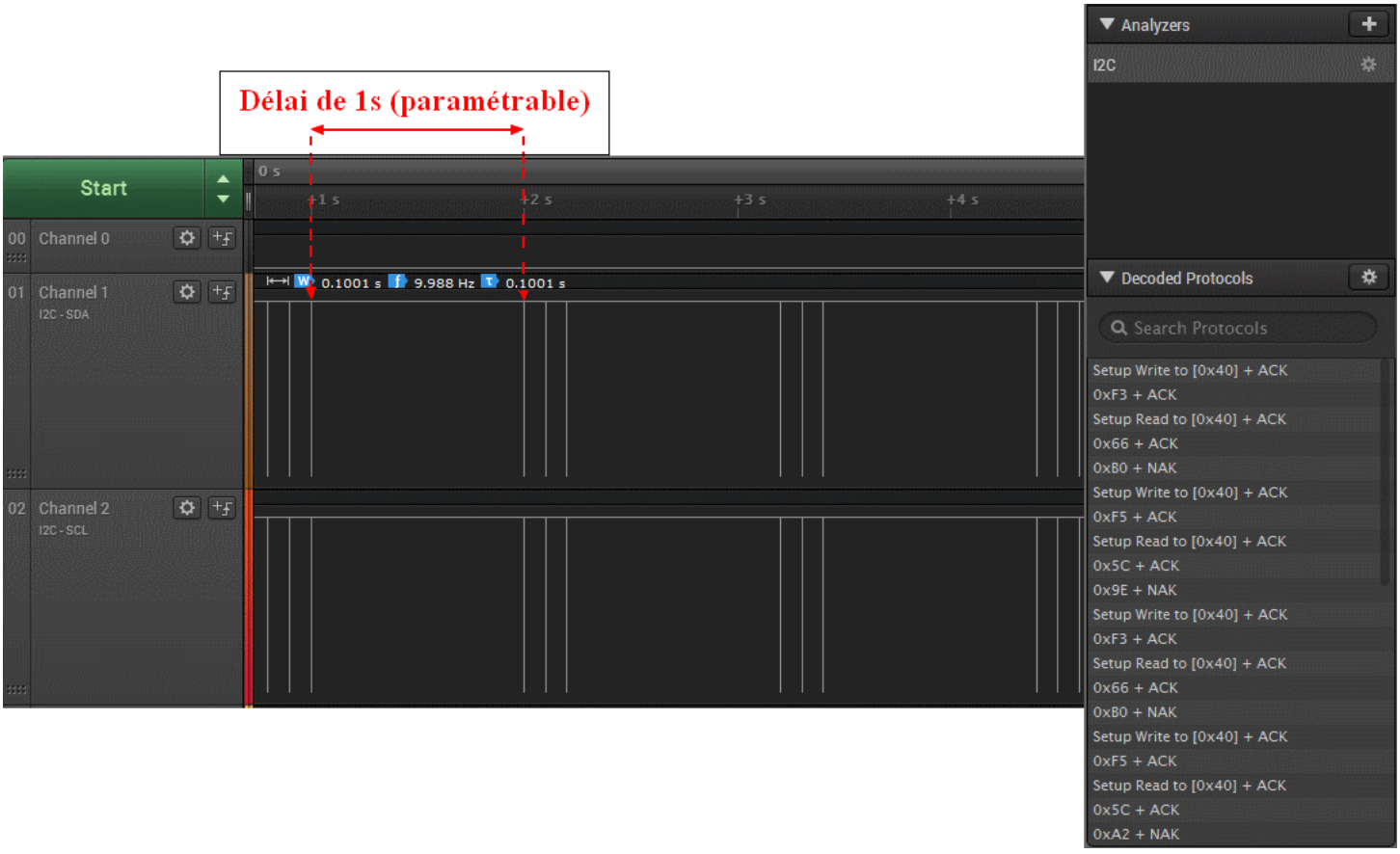
2) Les condensateurs de découplage

Pour créer une réserve d'énergie, évitant d'éventuelles micro-coupures, on utilise des condensateurs de découplage. En général, on utilise soit des condensateurs de 1µF, soit de 100nF. Il vaut mieux utiliser un plus gros condensateur (1µF) lorsque le composant à découpler est proche de l'alimentation du circuit, et utiliser des 100nF pour les composants plus éloignés. Les condensateurs que j'utilise sont diélectriques « à film isolant ».

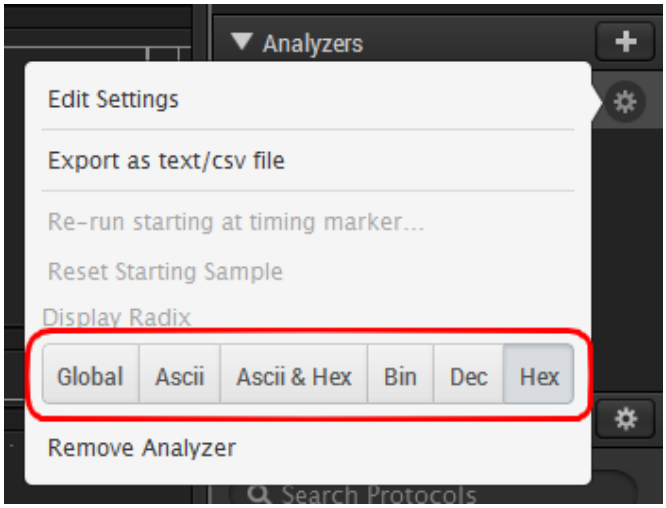
3) Trame I^2C

Pour observer physiquement la communication I2C, on utilise un analyseur logique. Nous possédons un analyseur de chez **Salae**, le **Logic 4**.

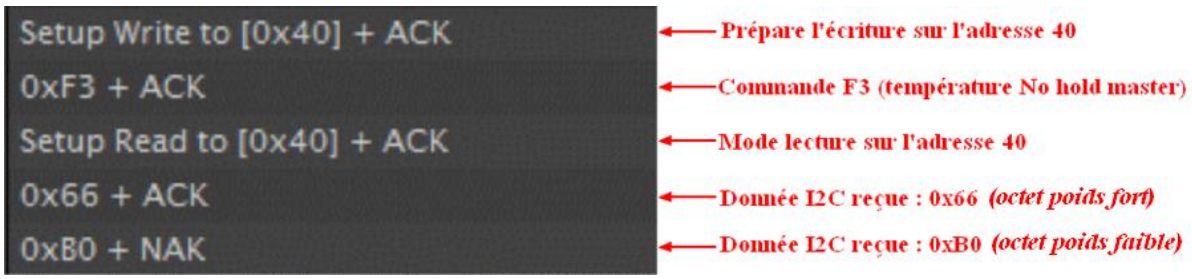
On peut voir que le ***Salae* Logic 4** nous permet bien de récupérer une trame lors de la commande d'un capteur de température en I_2C, grâce au logiciel du constructeur :



On voit sur cette capture que nous avons bien le délai de 1s entre chaque trame, conformément au programme préalablement modifié. Dans la partie ***Decoded Protocols***, on voit le résultat de chaque trame en Hexadécimal. On peut également régler la base (base 10, 16, 2 …) directement dans les paramètres de l'analyseur I_2C :



Trame décodée par le **Salae** (température) :



On voit dans la datasheet du capteur que 0xF3 est bien une demande de température :

Trigger T measurement	no hold master	1111'0011	<=> **0xF3**
Trigger RH measurement	no hold master	1111'0101	

Le critère « no hold master » signifie que le bus pourra être utilisé en même temps, cette commande ne bloque pas l'utilisation du bus.

Calcul de la température :

Formule : $T = \frac{D}{2^{16}} * 175{,}72 - 46{,}85$

T ⇔ Température (°C)

D ⇔ Données reçues (en décimal)

Calcul :

0x66B0 <=> $26288_{(10)}$

26288 / (2^16) * 175,72 - 46,85 = **23,63°C**

Trame décodée par le **Salae** (humidité relative) :

Setup Write to [0x40] + ACK	← Prépare l'écriture sur l'adresse 40
0xF5 + ACK	← Commande F5 (Humidité No hold master)
Setup Read to [0x40] + ACK	← Mode lecture sur l'adresse 40
0x5C + ACK	← Donnée I2C reçue : 0x5C *(octet poids fort)*
0xA2 + NAK	← Donnée I2C reçue : 0xA2 *(octet poids faible)*

Calcul de l'humidité relative :

Formule : $HR = \frac{D}{2^{16}} * 125 - 6$

HR ⇔ Humidité Relative (%)
D ⇔ Données reçues (en décimal)

Calcul :
0x5CA2 <=> $26288_{(10)}$
26288 / (2^16) * 125 - 6= **39,23%**

Les valeurs 175,72 ; 46,85 ; 125 et 6 sont des valeurs normalisées pour le type de capteur que j'utilise.
Le point de rosée combine la température et l'humidité, et selon leur valeur, de la rosée apparaît (à un point appelé point de rosée).

Calcul du point de rosée :

$$T_r = \frac{b\alpha(T, RH)}{a - \alpha(T, RH)}$$

avec : $\alpha(T, RH) = \frac{aT}{b + T} + \ln RH$

où a = 17,27 et b = 237,7
T <=> température en °C
RH <=> humidité relative entre 0,01 et 1

⇔ $T_{rosée}$ = 8,9°C

Conclusion

Au final, ce projet technique m'a été très bénéfique. En effet, il a permis de concrétiser une grande partie des enseignements apportés par mes professeurs durant mes deux années de BTS. Egalement, cela nous a permis, à moi et mon équipe, de devenir plus autonomes, de savoir mieux gérer notre temps et nous organiser.

Ce projet m'a donné une base solide pour ma poursuite d'études en électronique, que ce soit au niveau de mes compétences ou au niveau de mon livret scolaire.

Je remercie toute l'équipe pédagogique, à savoir M. Hortolland et M. Defrance, mais également M. Escuret, professeur de physique pour son aide durant le projet, et M. Silanus qui nous est venu en aide pour la partie électronique. Je remercie M. Bijou, responsable de Crossdock pour son accueil son implication dans le projet.

Partie EC 2 COMPTE RENDU ALAN JULLIARD

Remise en situation du projet, introduction

L'entreprise Crossdock dirigée par M. Bijou est spécialisée dans le secteur d'activité des transports routiers de fret interurbains dans l'acheminement de médicaments souhaite superviser à terme la température et l'hygrométrie de leur entrepôt situé sur Cavaillon.

La surveillance de la température et de l'humidité s'effectue au niveau des zones de stockage et les capteurs sont localisés aux 4 coins de chaque entrepôt pour une mesure uniforme.

L'entreprise consistant à préparer des commandes de produits cosmétiques ou parapharmaceutiques, une surveillance des conditions d'entreposage est importante car les produits sont soumis à des normes de qualité. De ce fait, maitriser et connaître la température et l'humidité des produits est important.

Un système est déjà mis en place dans l'entrepôt qui permet de récolter les valeurs par USB. Le problème avec cette solution c'est que pour observer les valeurs perçues ont doit se déplacer au Thermomètre USB, ensuite l'implanter sur un pc via une prise USB, pour ensuite lire les données et reposer après à son emplacement. Monsieur Bijou le gérant n'ayant pas son bureau à l'intérieur de l'entrepôt pour ces contrôles, cela le fait se déplacer plusieurs fois pour les relevés d'hygrométrie. (Image du capteur actuel).

J'ai pour rôle en tant que EC2 de concevoir le schéma d'un shield qui s'adapte à la carte Arduino MKR WAN 1300 avec un capteur de T/HR qui est retenu pour cette évolution EC1 après des tests, comportant aussi un écran OLED SSD1306.

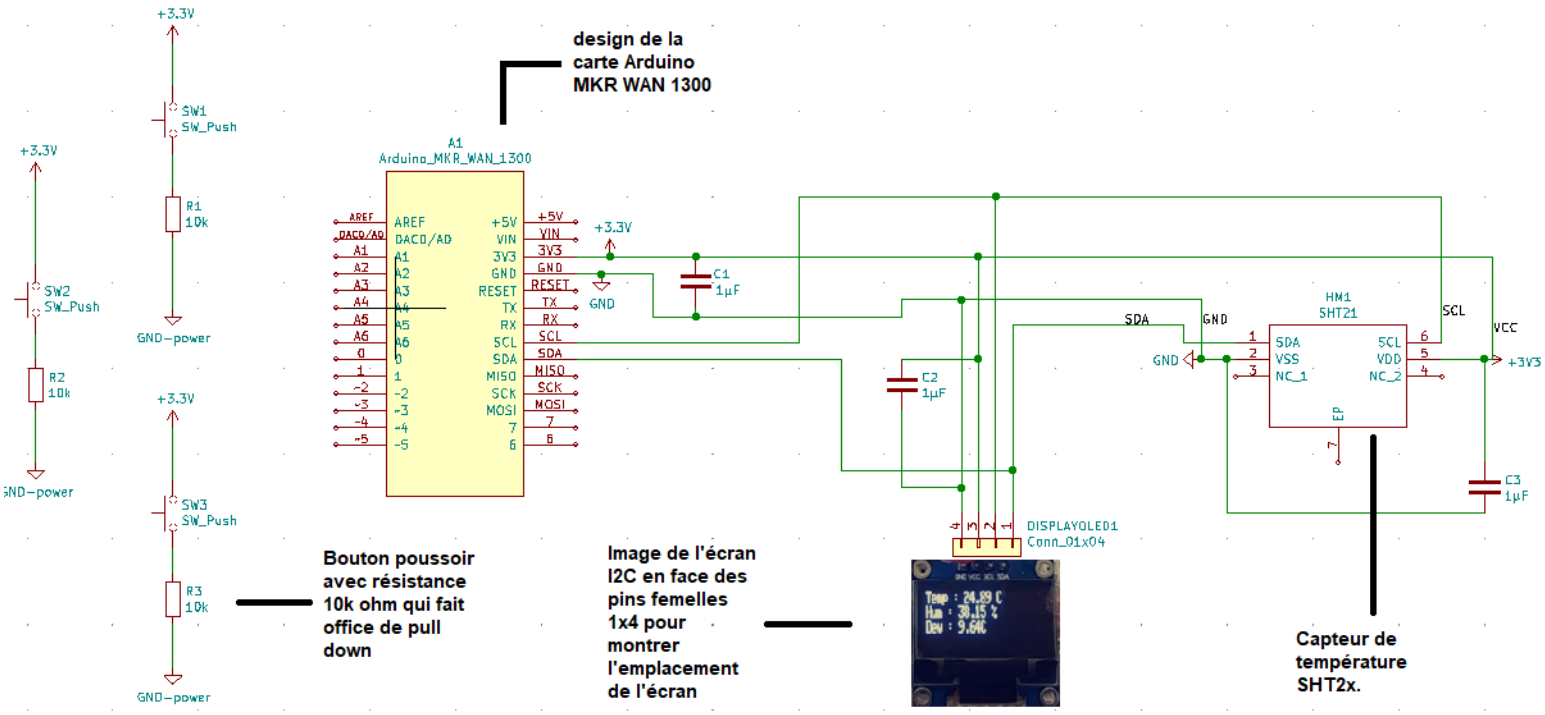
Mais aussi, de développer un programme permettant l'acquisition de la température, l'humidité et point de rosée via la carte MKR WAN 1300.

Puis faire la saisie de schéma et de routage de la carte et produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée.

Suite à la revue 2 après le développement du programme permettant l'acquisition de la température, de l'humidité et le point de rosée avec les valeurs s'affichant sur l'écran OLED, on choisit les valeurs que l'on souhaite afficher sur l'écran selon le bouton poussoir pressé.

L'aboutissement du schéma et routage du shield étant la priorité pour la dernière revue et donc obtenir un prototype viable à proposer pour l'entreprise.

Développement et conception de la carte sous Kicad



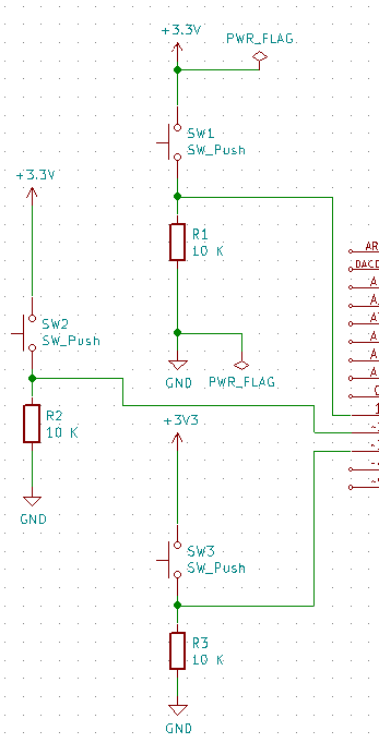
Lors de la revue 2 j'avais placé les composants sans les router de sorte à voir à quoi dans ressemblerait la carte dans le futur, ce qui au final, s'est révélé non ergonomique pour les composants. Certains étant mal placés, donc j'ai dû recommencer mon routage.

En premier lieux j'ai changé le schéma qui était peu lisible et incomplet au niveau des boutons poussoirs pour une utilisation plus fluide.

Mon professeur m'a également aidé pour le schéma afin qu'il soit plus agréable à lire.

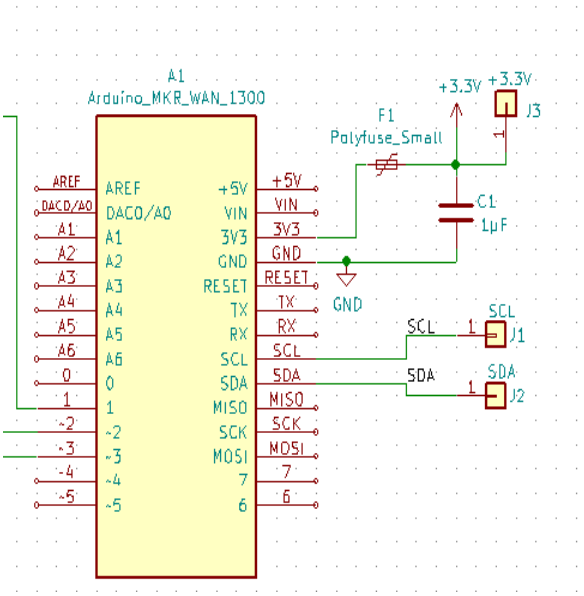
Les boutons n'étaient pas reliés aux broches 1, 2, et 3 sur l'ancien schéma, et des power flag ont été ajoutés.

Graphique d'une partie du schéma :



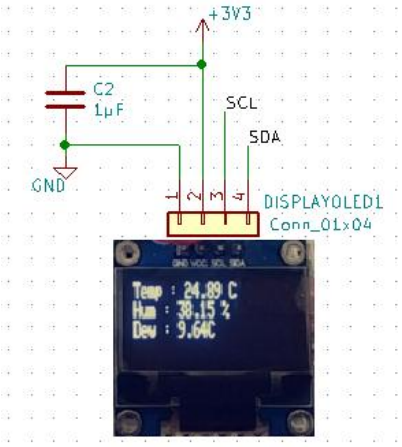
Un fusible a été rajouté pour protéger la carte MKR WAN 1300, des points de tests pour le 3,3V, SDA, SCL. Après usinage de la carte, un point de test pour le GND a été oublié. Cela a été corrigé sur la carte, un trou a été fait avec une mèche de 1mm pour rajouter le point de test du GND sur le plan de masse.

Graphique :



Rien n'a changé pour ce qui est de l'écran I2C, avec présence des labels pour identifier le SDA et SCL et un power flag 3.3V pour l'alimentation, et un GND pour la masse.

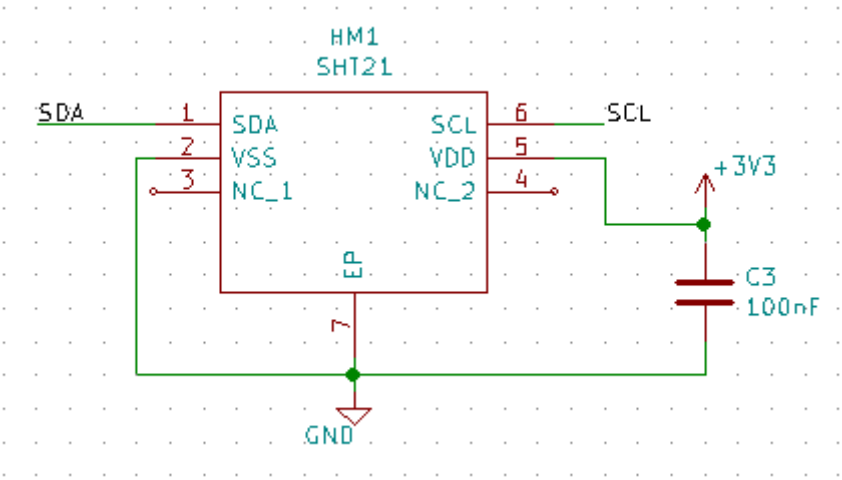
Graphique d'illustration de l'écran :



Par rapport au SHT21, capteur qui selon les tests de EC1 Alban Juan, est le plus stable au niveau des calculs de température et d'humidité, le choix se fait sur le capteur SHT21.

Sur l'ancien schéma, le capteur n'avait pas la broche 7 reliée au GND. La broche 7 n'est pas une broche comme les broches 1 à 6, celle-ci se trouve sous le capteur.

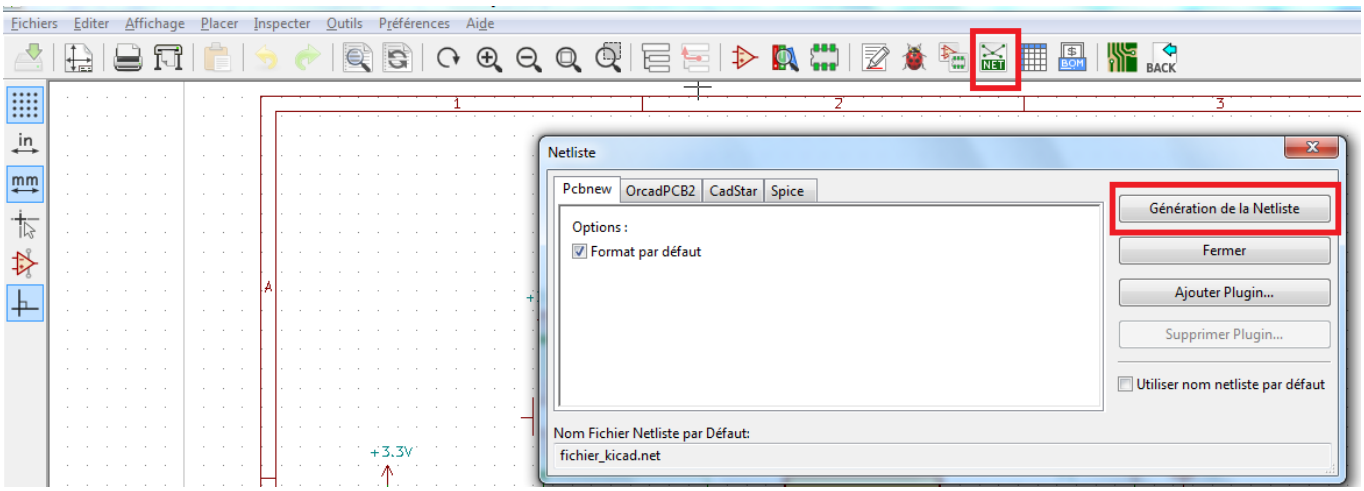
Des labels et flags ont été ajoutés pour le 3.3V et le GND et SDA SCL.



Le fait d'avoir mis chaque entrée avec des labels et power flag offre une meilleure lecture du schéma et limite le nombre de fils dans ce schéma.

Par la suite, certaines de mes empreintes étaient mauvaises. L'empreinte pour la carte arduino n'était pas bonne et après réflexion de mon professeur d'électronique et communication, l'empreinte était mâle donc la carte n'était pas un shield.

Une fois le schéma et les empreintes terminées, je génère la Netliste et j'enregistre le fichier en .net



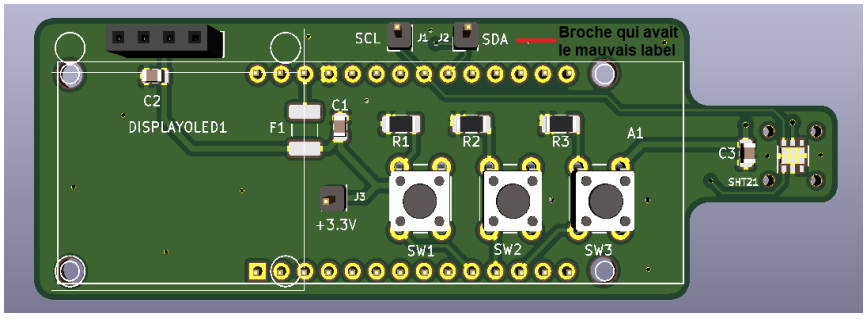
La Netliste sert à générer les empreintes et modèles 3D qui serviront pour le routage, on peut avoir un perçu 3D des composants sur la carte et de ce fait ajuster si le composant si il est trop près d'un autre modèle.

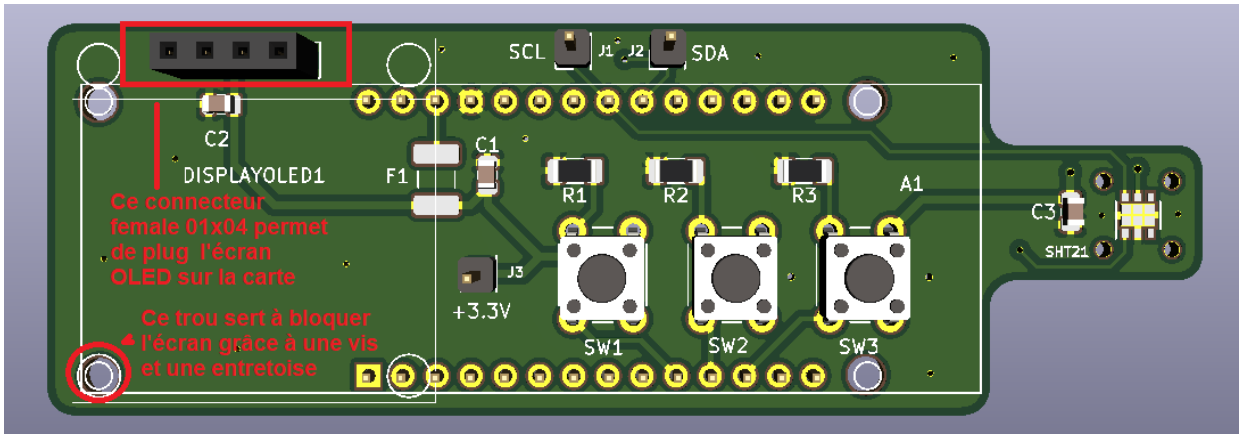
Routage

Ayant eu des difficultés pour le routage de ma carte, mon professeur de Electronique et communication m'a plusieurs fois aidé.

Comme je l'avais dit précédemment une des broches de points de test portait le mauvais label et de ce fait, deux broches avaient les mêmes noms. Ceci a été corrigé avant usinage.

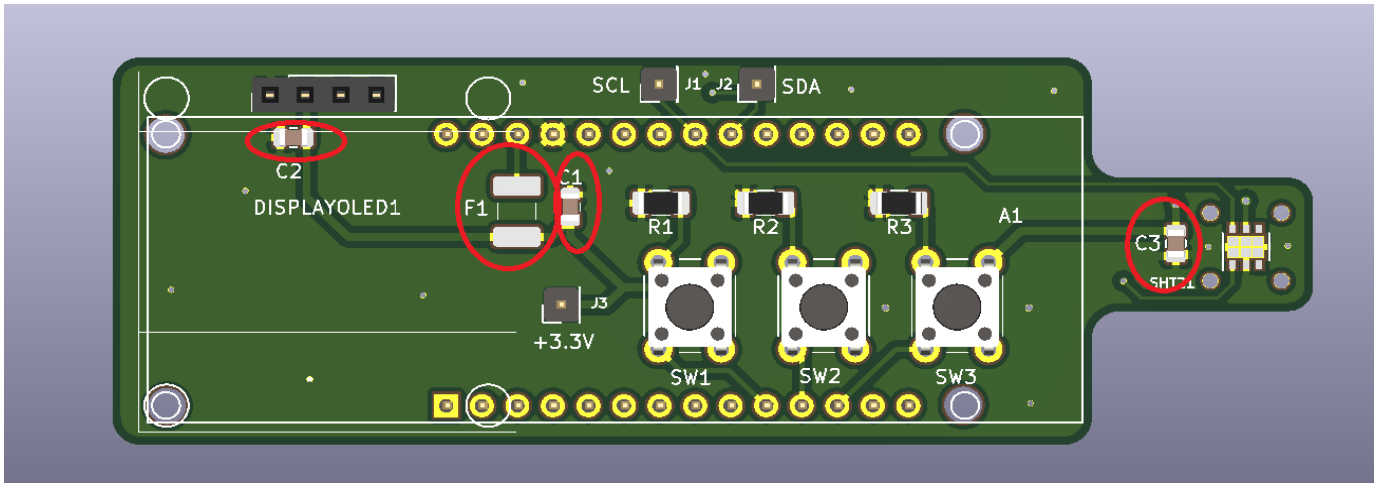
La carte possède plusieurs trous. 3 serviront pour tenir la carte qui sera attachée à une structure, le 4ème sert pour soutenir la carte avec une vis et une entretoise. L'écran OLED se plug sur le connecteur 01x4 mais vu que l'entrée est haute que la carte plus la taille des pins de l'écran, il y a un écart de quelques centimètres entre l'écran et la carte. Donc, cela provoque un jeu conséquent pour empêcher que l'écran ne soit abîmé la vis et l'entretoise soutiennent l'écran.



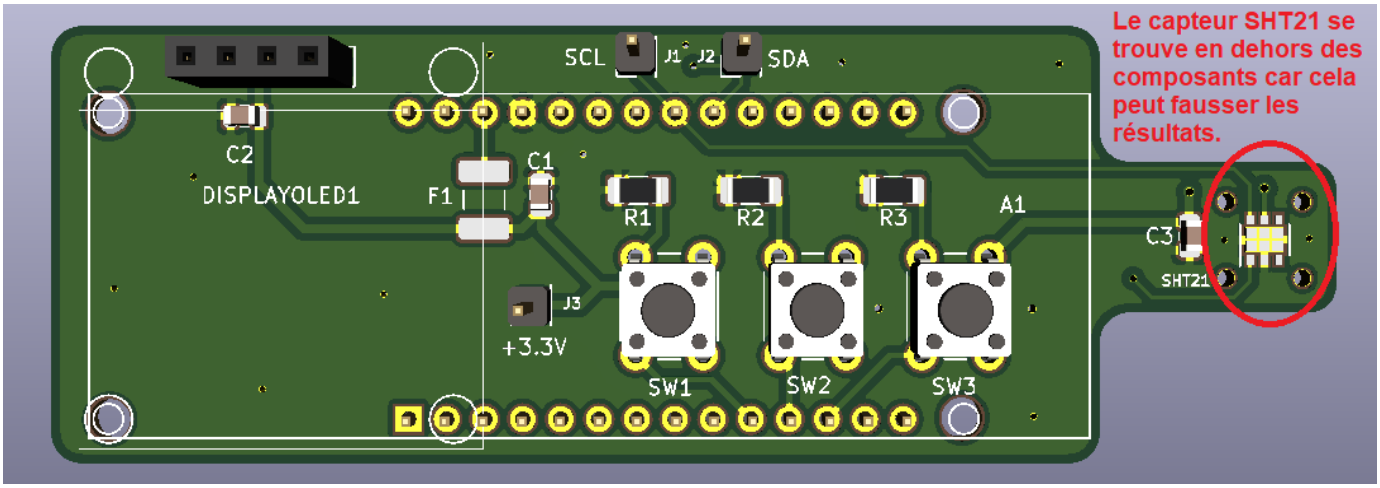


Un fusible a été ajouté pour protéger la carte Arduino MKRWAN1300 en cas de tension extérieure non prévue pour celle-ci.

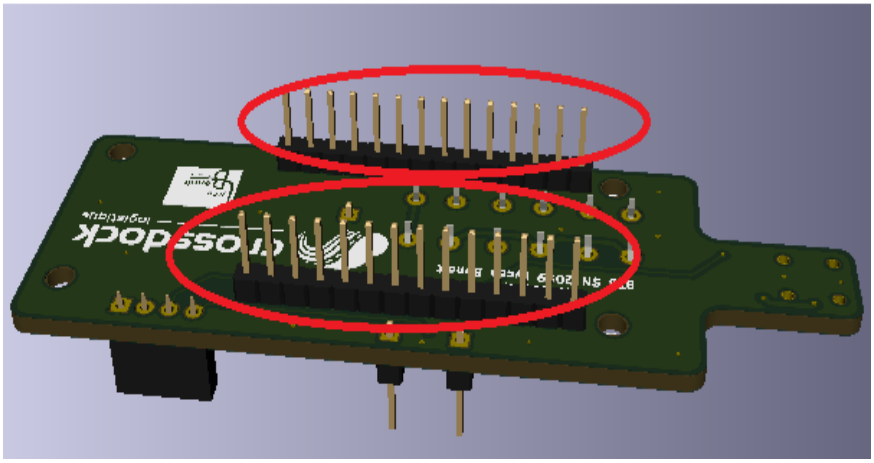
Des condensateurs de découplage au plus proche des circuits sont présents. Leur utilité est d'améliorer le fonctionnement des circuits et composants. Cela permet de palier à une demande de courant très brève du composant et à éliminer tout risque d'auto-oscillation.



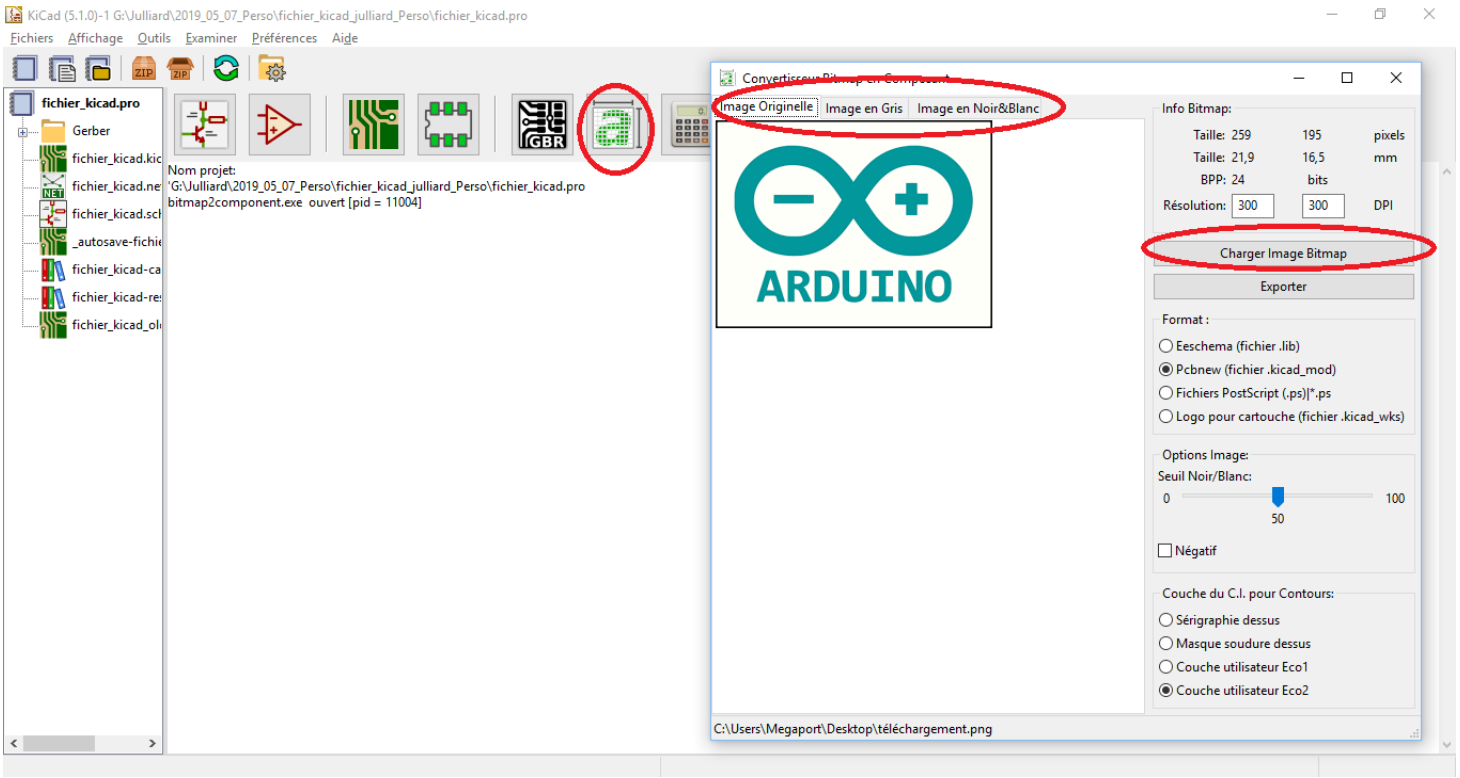
Le SHT2x donc le capteur de température est éloigné du reste des composants il seul avec un condensateur sur cette cellule, ce n'est pas anodin car si le capteur est A proximité des autres circuits ou composants, les valeurs risquent d'être faussées. En effet, si le capteur est trop près des boutons poussoirs lorsque l'on va appuyer sur n'importe lequel, la chaleur corporelle des doigts va fausser encore plus les valeurs du capteur d'où l'emplacement décalé sur cette carte.



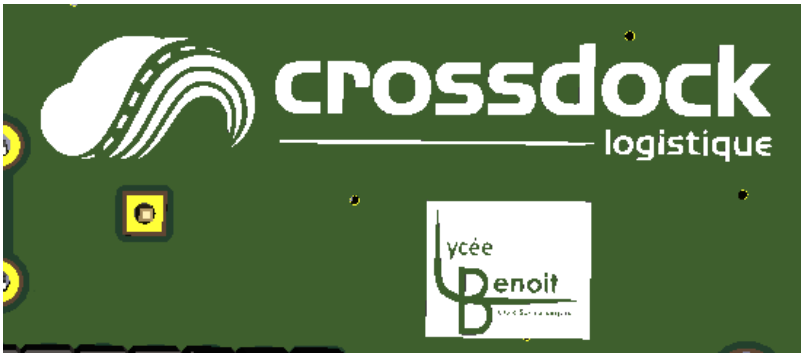
La carte est un shield et elle s'implante sur les pins female de la carte Arduino MKRWAN1300. Pour ne pas se tromper de sens lorsque je connecte les cartes entre elles, j'utilise les pins des points de test SDA et SCL comme repères car sur la carte, on peut voir que les pins sont routés juste à côté des pins et sur la carte Arduino MKRWAN1300 sur les connecteurs femelle on voit SDA et SCL écrit.



Une fois tous les composants routés dans le cadre du projet, j'ajoute le Logo de l'entreprise Crossdock et le logo de mon lycée ainsi que mon nom pour l'usinage.



Résultat sur la carte :



Fabrication de la carte

Le fait de faire sous-traiter la fabrication des PCB en Asie plutôt qu'en France ou en Europe, est une obligation purement financière avec un rapport qualité / prix convenable. Sinon sans cette possibilité, nous n'aurions tout simplement pas les moyens de faire sous-traiter professionnellement les circuits imprimés des différents projets de cette année.

Le PCB a été commandé chez JLCPCB. C'est 1.79€ pour 5 circuits. Le stencil a été commandé chez SeeedStudio, c'est 8.90$ le stencil de 10x15cm.

En premier lieu j'ai établi une liste des composants pour chiffrer le prix d'une carte finalisée :

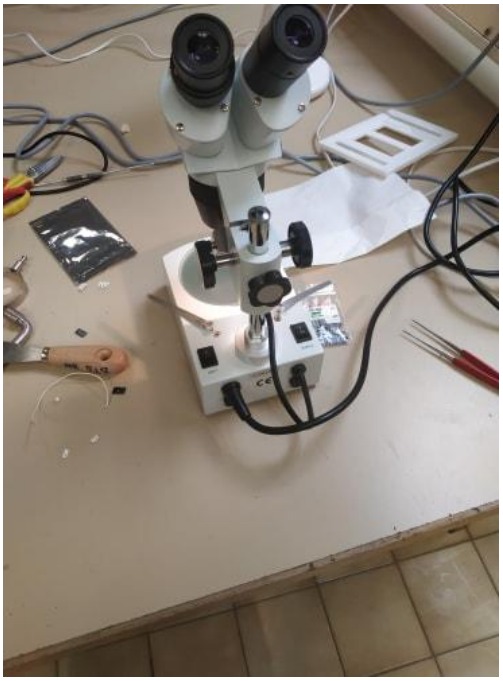
1	HM1	SHT21	0_ModulesProjet:SHT21-SF2	0_ModulesProjet:SHT21-SF2		6,84 €		
2	DISPLAYOLED:	https://www.az-delivery.de/products/0-96zolldisplay		Conn_01x04		4,99 €		
3	C1	RS : 264-4450		1µF		0,09 €	Vendu par 25	
4	C2	RS : 264-4450		1µF		x		
5	C3	RS : 264-4422		100nF		0,164		
6	A1	https://www.arduino.cc/en/Main/ArduinoBoardLeonardo		Arduino_MKR_WAN_1300		35.00		
7	SW1	Conrad : 700350		SW_Push		0,30 €	1 par pièce donc 0,30 par SW	
8	R1	RS : 125-1192		10 K		0,014 €	vendu par 100	
9	SW2	Conrad : 700350		SW_Push		0,30 € TTC		
10	R2	RS : 125-1192		10 K		x		
11	SW3	Conrad : 700350		SW_Push		0,30 € TTC		
12	R3	RS : 125-1192		10 K		x		
13	J3	Gotronic : 08000		+3.3V		0,60 €	1 rangée de 36	
14	J1	Gotronic : 08000		SCL		x		
15	J2	Gotronic : 08000		SDA		x		
16	F1	RS : 867-5233		Polyfuse_Small		0,38 €		
17	J4	Gotronic : 08000		GND		x		
18	VIS	Farnell : 247 2692				10,68 €	paquet de 100	
19	ECROU	Farnell : 701 6920				3,47 €	paquet de 50	
20	ENTRETOISE	Farnell : 249 4574				0,29 €	par 10	
PRIX TTC					TOTAL TTC :	63,41 €		
					stencil +PCB =	8,90 + 1,79 €	soit 74,1€	

Le coût total est estimé à 74,10 € en comprenant le frais pour le stencil et le PCB. La plus grosse part sur l'entièreté de la carte est la MKR WAN 1300 coûtant à elle seule 35€ sur le site Arduino.

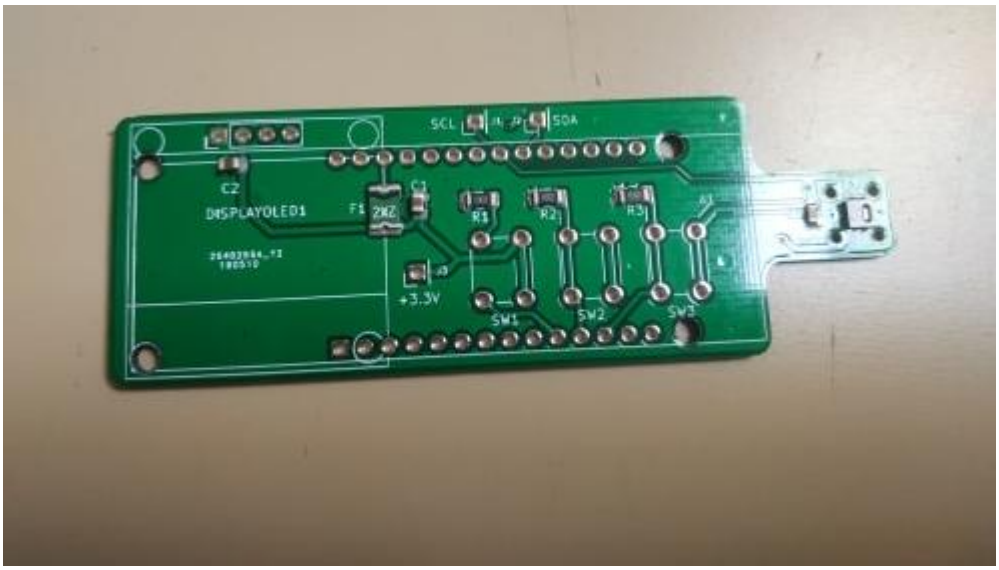
Etant donné qu'il faut souder les composants CMS en utilisant le stencil, il faudra compter le prix de la pâte à braser, ce qui vaut relativement cher (25 g pour 40 euros).

Je n'ai pas pris en compte le prix du Connecteur femelle 01x04 qui permet de plug l'écran OLED car le lycée en possédait déjà.

Une fois tous les composants recueillis, je commence par placer mon stencil au-dessus de mon PCB pour faire la seule face avec les CMS. Il y a 8 CMS à souder avec la pâte à braser, le reste sont des composants traversants donc soudés avec de l'étain et un fer à souder.



Sur l'image on peut voir tous les composants posés sur la pâte à brasser prête à être mise au four à fusion. Avant de l'enfourner je vérifie au microscope si toutes les soudures et positions sont conformes.



Problème constaté avec mon professeur, lorsque j'ai vérifié la carte, le SHT2x risquait de ne pas avoir suffisamment de pâte à brasser sur une de ses broches.

Le risque a été pris de l'enfourner mais la soudure n'a pas prise.

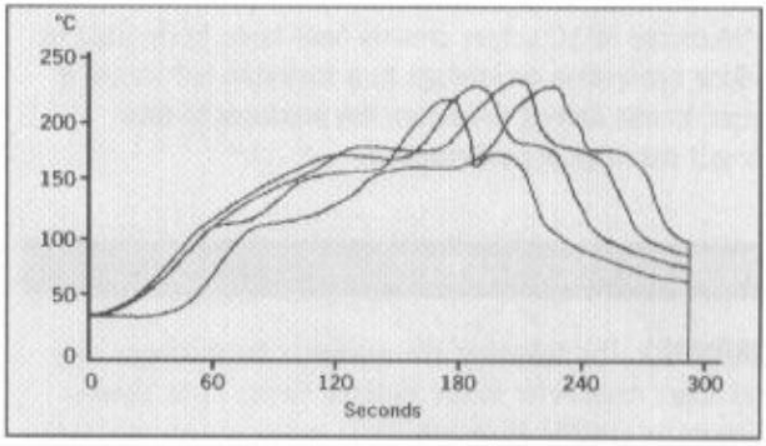
Pour résoudre le problème mon professeur d'Electronique et communication m'a aidé et m'a expliquer comment résoudre ce type de problème :

Il faut prendre une panne de fer à souder la plus fine possible, dans ce cas une panne de 0.3mm avec de l'étain assez fin, et souder avec à l'aide du microscope pour vérifier la conformité soudures.

Pour enfourner la carte il faut quelques réglages sur le four à fusion : température, durée, type.

Ceci est le four à fusion du lycée Alphonse Benoit, tous les projets utilisent ce four pour souder leurs cartes.

Avant de mettre directement nos cartes dans le four il faut sélectionner les bons paramètres :

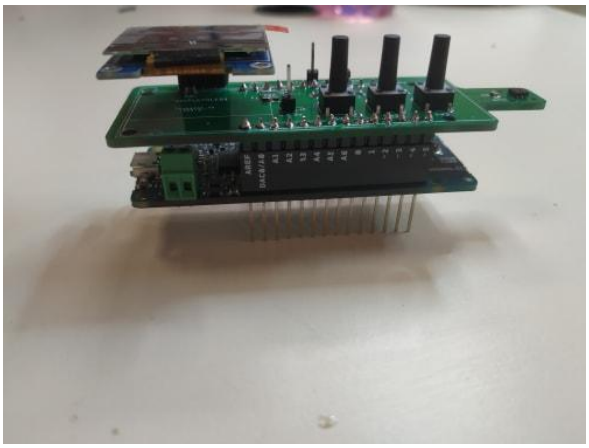


Les paramètres sont les suivants :

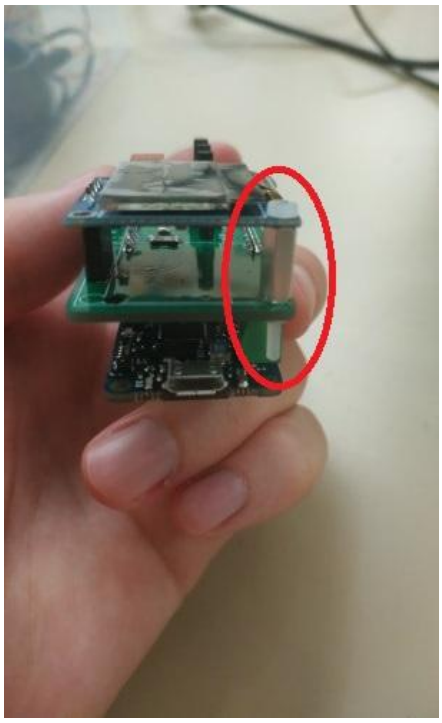
- Préchauffage 90°C
- 2 minutes 30 secondes montée à 220°C
- 1 minute montée à 250°C (re fusion)
- 1 minute 50 secondes : refroidissement (Prendre tout de même des pinces brucelles pour sortir la carte)

Une fois les composants soudés, je vérifie au microscope si des CMS ce sont décalés ou non car pas bien fixés pendant la fusion. Vu que tout était mis à part une broche du SHT2x, j'ai pu commencer à souder mes composants traversant. Après avoir fini les soudures, je les nettoie avec de l'alcool isopropanol.

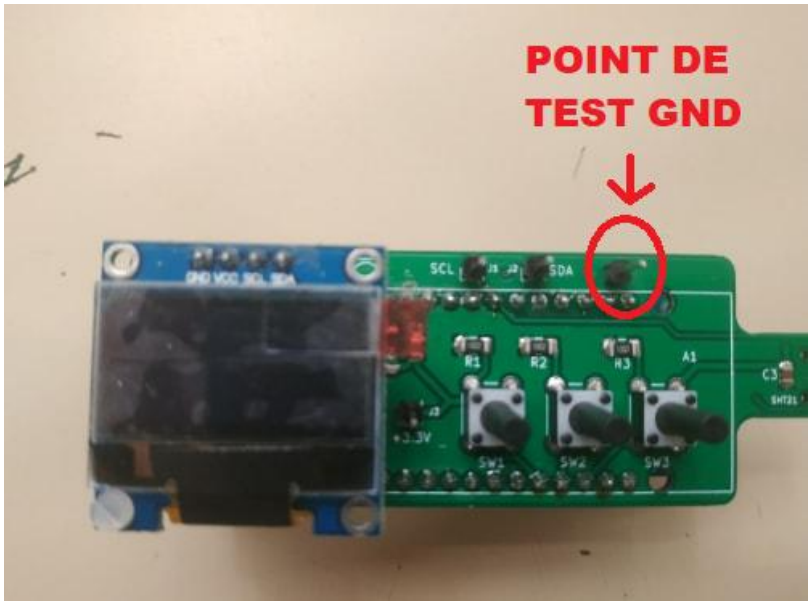
Il ne reste plus qu'à plug la carte sur la MKR WAN1300 et l'écran OLED.*



N'ayant pas l'entretoise, la vis et l'écrou, après avoir fini les soudures, j'ai donc dû faire attention avec l'écran OLED le temps de recevoir ce qu'il fallait pour le soutenir.



De plus, un point de test pour le GND a été ajouté vu qu'il n'y était pas. Il a été percé avec une mèche de 0,1mm puis soudé.



Une fois les soudures terminées la carte est donc opérationnelle et prête à utilisation.

Programme Arduino

Suite à l'entretien avec M. Bijou avant la revue 2, le programme avait changé pour inclure des boutons poussoirs permettant d'afficher ce que l'on voulait afficher étant donné qu'il y a 3 valeurs : température, humidité et point de rosée, et 3 boutons poussoirs donc 1 valeur pour 1 bouton.

De plus il fallait ajouter un petit menu permettant de voir les valeurs en plus petits caractères et dire quel bouton correspond à quelle valeur.

Suite à la revue 2 j'ai pu comprendre pendant la démonstration les problèmes liés au programme actuel que j'avais élaboré et qui nécessitait une amélioration.

En premier lieu, le programme générait du rebond lors de l'appui d'un bouton poussoir et de ce fait, lorsqu'on appuyait l'information n'était pas transmisse et rien ne se passait.

Deuxièmement, lorsque l'écran OLED démarrait le menu que j'avais fait ne s'affichait pas. Cela restait sur le message lors de l'alimentation. Le menu s'affichait uniquement lorsqu'un bouton était pressé pour la première.

Troisièmement le programme n'était pas propre, il y avait d'anciennes parties qui étaient mises en commentaire ce qui rendait le programme peu lisible. De plus, dans le void loop() il y avait beaucoup trop de if pour chaque bouton 2 if (lorsqu'il est pressé et non pressé) donc je devais optimiser cela et le faire plus propre.

Pour régler le problème de rebond je suis allé demander à mon professeur d'Informatique et Réseaux comment solutionner ces difficultés grâce au programme.

En premier lieu il m'a expliqué que je pouvais simplifier mon programme en utilisant la méthode du SWITCH CASE à la place d'utiliser de nombreux IF.

Ne connaissant ce domaine, je lui ai demandé des renseignements par rapport à cette méthode, je suis donc allé voir sur le site qu'il m'a proposé de consulter:

https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/

En lisant la description sur le site de référence arduino, j'ai directement compris que c'était parfaitement adapté à la situation et que de plus, cela réduira le nombre lignes de code dans le programme étant donné que la RAM (SRAM) n'est pas extensible donc limitée.

Description

Like if statements, switch case controls the flow of programs by allowing programmers to specify different code that should be executed in various conditions. In particular, a switch statement compares the value of a variable to the values specified in case statements. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The break keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions ("falling-through") until a break, or the end of the switch statement is reached.

Syntax

```
switch (var) {
  case label1:
    // statements
    break;
  case label2:
    // statements
    break;
  default:
    // statements
    break;
}
```

Donc pour pouvoir utiliser le SWITCH CASE, mes boutons ont été assignés à des poids binaire donc 1, 2 et 4 vu qu'il y a 3 boutons :

```
#define BOUTON1 1
#define BOUTON2 2
#define BOUTON3 4
```

Une fois que j'ai #define mes 3 boutons, je dois déclarer l'entrée de chacun et définir leur état naturel respectif :

```
const int ButtonPin1 = 1;
const int ButtonPin2 = 2;
const int ButtonPin3 = 3;

int buttonsState = 0;
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
```

je déclare les entrées de chaque boutons

Je déclare que mes boutons sont à l'état bas naturellement

Ensuite on rentre dans le void loop on display.clearDisplay () ; pour effacer l'écran et le buffer.

J'utilise une méthode nommée bitshiftleft qui consiste de provoquer le décalage à gauche des bits de l'opérande de gauche du nombre de positions spécifié par l'opérande de droite.

Donc mon professeur m'a montré comment structurer mon code à partir du descriptif sur le site d'Arduino sur le bitsshiftleft :

```
buttonsState = digitalRead(ButtonPin1);
buttonState2 = digitalRead(ButtonPin2);
buttonsState += buttonState2<<1;
buttonState3 = digitalRead(ButtonPin3);
buttonsState += buttonState3<<2;
```

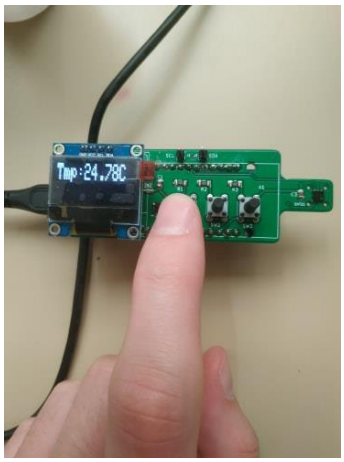
Une fois ceci fait, je commence mon switch case pour définir les actions de mes 3 boutons poussoir :

```
switch(buttonsState) {
  case BOUTON1:
      display.clearDisplay();
      display.setTextSize(2);
      display.setTextColor(WHITE);
      display.setCursor(0,0);
      display.print("Tmp:");
      display.print(SHT2x.GetTemperature());
      display.println("C");
      display.display();
      delay(2000);
    break;
  case BOUTON2:
     display.clearDisplay();
     display.setTextSize(2);
     display.setTextColor(WHITE);
     display.setCursor(0,0);
     display.print("Hum:");
     display.print(SHT2x.GetHumidity());
     display.println("%");
     display.display();
     delay(2000);
  break;
  case BOUTON3:
```

On rentre dans le switch

case BOUTON1: dans le cas où un bouton serait appuyé il va affiché la temperature en taille 2 pour bien lire la valeur avec le C pour le Celsius et va rester figé pendant 2 secondes. lorsque le case est terminé il faut ajouter break à la fin.

même principe pour BOUTON2 sauf que celui ci va afficher l'humidité et derrière le %. Pareil pour BOUTON3 avec le point de rosée.

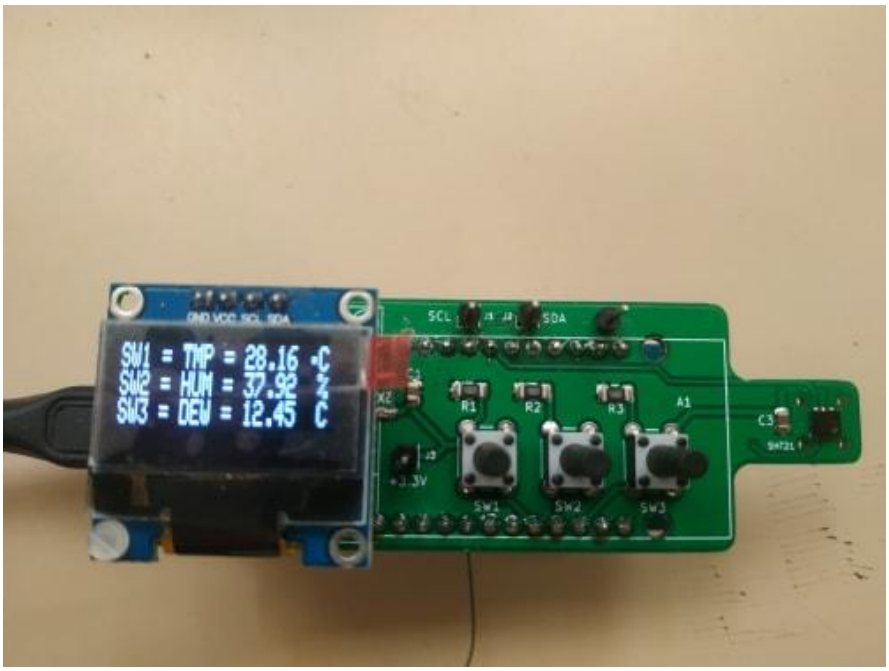


Sur cette image je montre un cas d'utilisation lorsque j'appuie sur le bouton poussoir 1. Cela affiche la température ambiante de la salle de classe et l'affiche en taille 2. Cette taille de caractère a été modifiée car après remarque, la taille 1 est peu lisible pour une personne ayant des problèmes de vue.

```
break;
  default: default défini l'action qu'il va se passer lorsque les 3 boutons ne sont pas utilisés
     display.clearDisplay();
     display.setTextSize(1);
     display.setTextColor(WHITE);
     display.setCursor(0,0);
     display.print("SW1 = ");
     display.print("TMP = ");
     display.print(SHT2x.GetTemperature());
     display.print(" \xF8\x43 \r\n");
     display.print("SW2 = ");
     display.print("HUM = ");
     display.print(SHT2x.GetHumidity());
     display.println("  %");
     display.print("SW3 = ");
     display.print("DEW = ");
     display.print(SHT2x.GetDewPoint());
     display.print("  C");
     display.display();
   break;
```

Ensuite après les 3 cases, je décide de faire un menu permettant d'afficher les 3 valeurs des autres sans appuyer sur les boutons tout en actualisant les données toutes les 1 seconde et indiquant quel bouton correspond à quelle valeur :

L'image ci-dessous montre le menu lorsqu'aucun des 3 boutons n'est utilisé :



<u>On peut constater sur la ligne de TMP qu'il y a le Celsius « ° » et non pas lorsqu'on appuie sur le bouton, Pourquoi</u> ?

Car en taille de caractère il n'y a pas suffisamment de place en largeur d'écran pour pouvoir l'écrire en taille 2 de caractère. Ayant la place en taille 1 de caractère, je le fais pour le menu.

Pour pouvoir afficher le Celsius « ° » il faut utiliser la table ascii : il faut faire un display.print(« \xF8\x43 \r\n ») ; = °C F8 = ° ; 43 = C majuscule

<u>L'image ci-dessous montre les bibliothèques que j'ai installées pour utiliser le capteur SHT2x <Sodaq_SHT2x.h>, l'écran donc les includes mentionnant Adafruit.</u>

Le #include Spi.h permet de définir que la carte Arduino est la carte maître.

```
#include <SPI.h>
#include <Wire.h>
// two libraries inserted into the 'libraries' folder
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
#define OLED_RESET 4
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

#include <Sodaq_SHT2x.h>
```

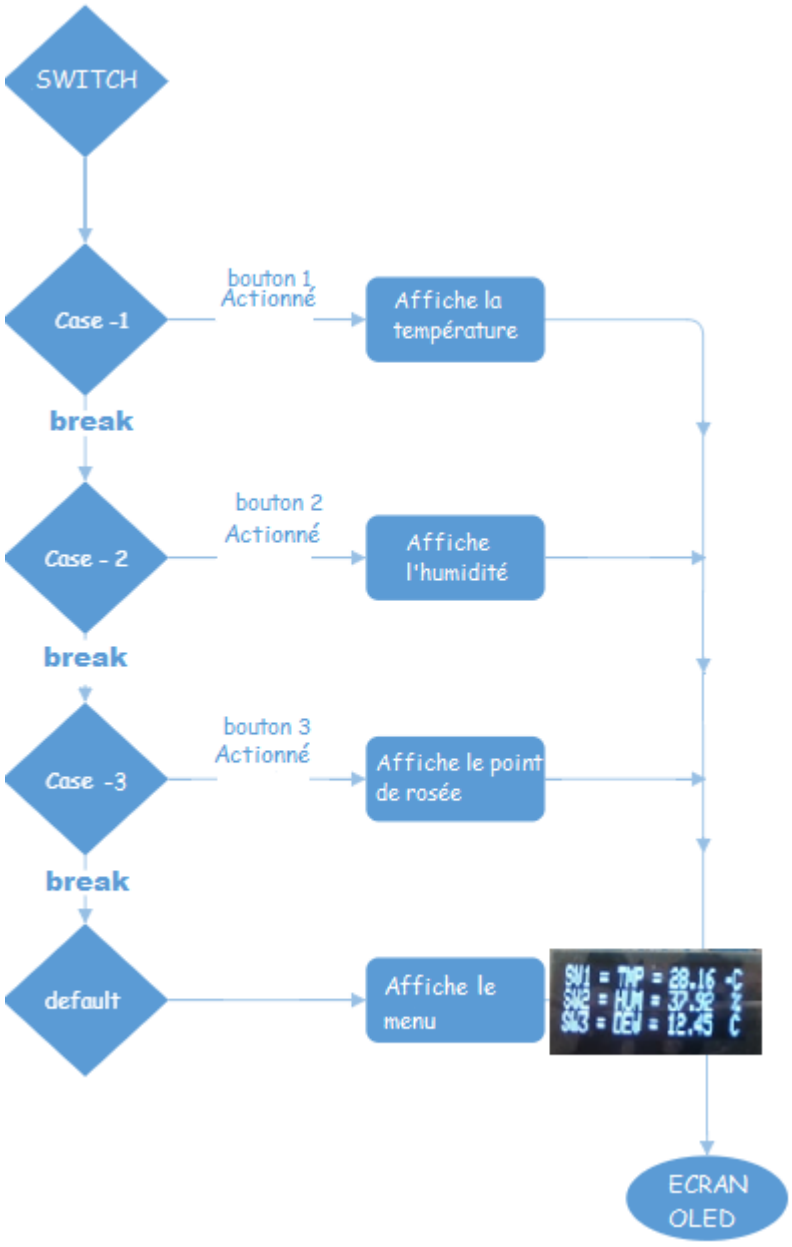
Permet la communication en I2C

Bibliothèque permettant d'utiliser l'écran OLED I2C

On définit la taille de l'écran (l'argeur et longeur)

Librairie pour le capteur SHT21

J'ai ensuite réalisé un organigramme pour montrer comment fonctionne le switch case en intégrant l'utilisation des boutons et le capteur SHT2x avec l'écran OLED.



La déclaration de switch commencera du haut vers le bas et dans le premier cas, elle vérifiera si la valeur de l'expression correspond ou non.

Dans le cas ou 2 boutons sont pressés simultanément, rien ne va se passer et l'écran restera sur le menu.

Le temps d'affichage de chaque valeur est de 2 secondes mais il y a une fonction de réglable.

Mon programme a évolué plusieurs fois durant ce projet en fonction des rectifications nécessaires après tests et suite à la demande d'évolution de Monsieur Bijou.

Initialement j'utilisais des IF pour le cas d'utilisation des boutons :

```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.print("Push BP 1 = Tmp");
display.print("Push BP 2 = Hum");
display.print("Push BP 3 = Dew");

buttonState1 = digitalRead(ButtonPin1);

if (buttonState1 == HIGH) {
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.print("Temp:");
display.print(SHT2x.GetTemperature())
display.println(" C");
display.display();
delay(2000);
}

buttonState2 = digitalRead(ButtonPin2);

   if (buttonState2 == HIGH) {

   display.setTextSize(1);
   display.setTextColor(WHITE);
   display.setCursor(0,0);
   display.print("Hum:");
   display.print(SHT2x.GetHumidity());
   display.println("%");
   display.display();
   delay(2000);
   }
 buttonState3 = digitalRead(ButtonPin3);

if (buttonState3 == HIGH) {
display.setTextSize(1);
display.setTextColor(WHITE);
```

Lorsque qu'aucun bouton poussoir ne sera utiliser l'écran affichera les lignes déclaré à display.print

Pour pouvoir utiliser 3 BP j'ai utiliser des IF du fait que les BP sont définis à l'état BAS dès que buttonState va lire le boutton au niveau HAUT il va afficher soit la température,l'humidité, le point de rosée selon le bouton.

Mais en utilisant le SWITCH CASE cela règle le problème de « rebond » et libère de l'espace vu qu'il y a clairement moins de ligne de code.

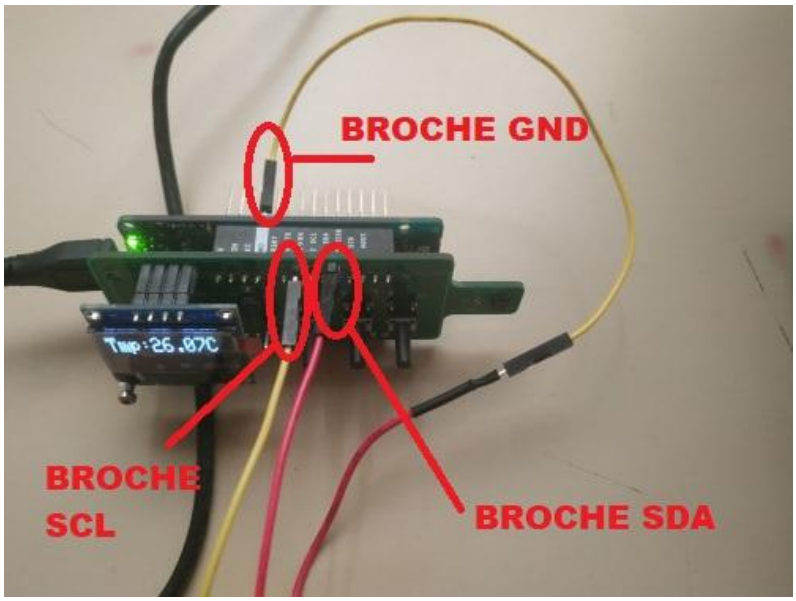
Le programme utilisant SWITCH CASE est abouti et opérationnel donc ce sera la solution finale donnée à M. Bijou et correspondant à ses souhaits de fonctionnalité.

Partie physique

Pour comprendre comment fonctionne la bibliothèque du SHT2x sur Arduino, j'ai fait une analyse de trame avec le logiciel Logicport et l'outil Salae en me branchant sur le SCL et le SDA et au GND de me carte.

En lisant la documentation du SHT2x j'ai trouvé les méthodes pour calculer la température, l'humidité et le point de rosée qui est un calcul en fonction de la température et de l'humidité calculée.

Ci-dessous image du câblage du salae sur la carte

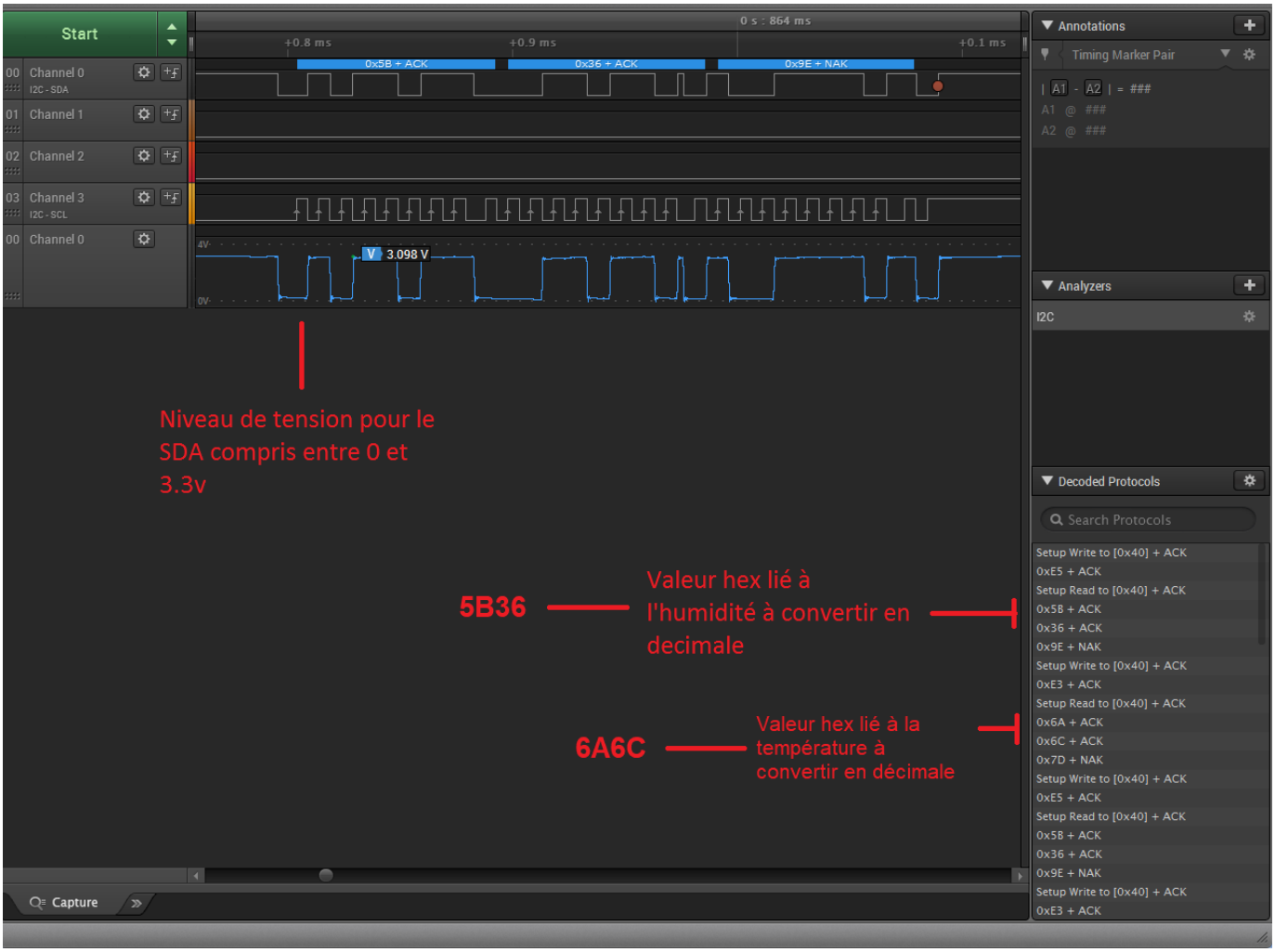


A ce moment-là le point de test du GND n'était pas encore mis sur le PCB donc j'ai utilisé le GND de la carte Arduino MKR WAN 1300.

Le tout relié au Salae qui est alimenté par l'USB du PC :

Pour comprendre comment calculer la température à partir des trames, il faut regarder la documentation du SHT2x. et celle-ci nous dit que le premier setup read contient 2 Acknowledge pour l'humidité et que le prochain setup read contient 2 Acknowledge pour la température sous forme de valeur Hexadécimale.

Sur la trame que j'ai analysée, je recueille mes données :



Le premier Acknowledge nous indique une valeur Hex de 5B36 pour l'humidité et le deuxième 6A6C pour la température.

Pour comprendre comment exploiter ces valeurs j'ai regardé comment fonctionnent les calculs dans la documentation du SHT2x.

Images des calculs dans la documentation :

6.2 Temperature Conversion

The temperature T is calculated by inserting temperature signal output S_T into the following formula (result in °C), no matter which resolution is chosen:

$$T = -46.85 + 175.72 \cdot \frac{S_T}{2^{16}}$$

6.1 Relative Humidity Conversion

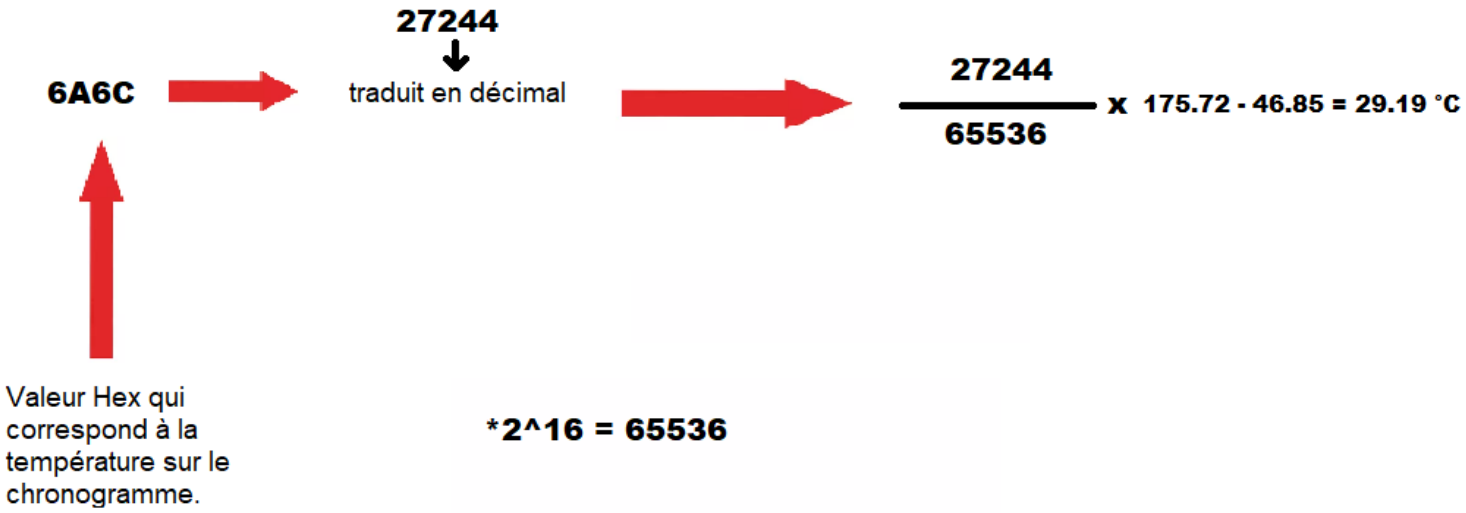
With the relative humidity signal output S_{RH} the relative humidity RH is obtained by the following formula (result in %RH), no matter which resolution is chosen:

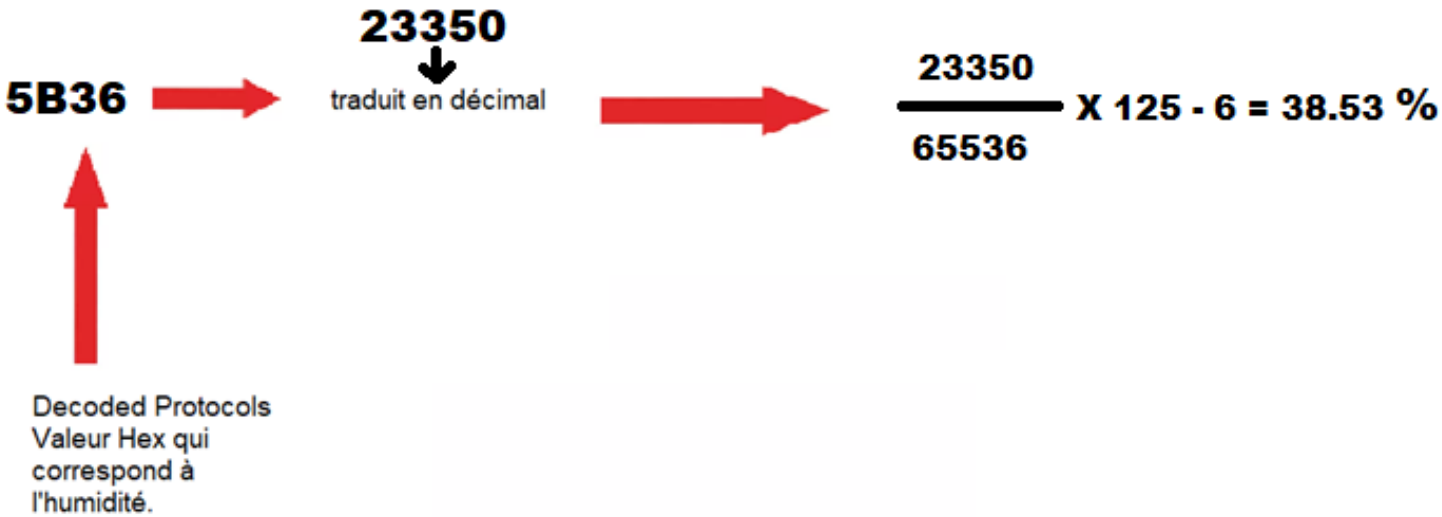
$$RH = -6 + 125 \cdot \frac{S_{RH}}{2^{16}}$$

St correspond à la valeur Hex traduite en Décimal et idem pour Srh.

J'ai donc schématisé pour expliquer le procédé :

Pour la température :





Ce sont des valeurs tout à fait cohérentes qui confirment les calculs.

Néanmoins il manque le calcul du point de rosée. Le point de rosée est un calcul qui se fait avec la température et l'humidité relative. Pour le connaître je suis allé sur le site Wikipédia qui indique 2 façons de le calculer.

Les 2 formules :

Formule de Heinrich Gustav Magnus-Tetens[5]

Domaine de validité :

- T, température mesurée : 0 °C < T < 60 °C
- RH, humidité relative : 0,01 (1 %) < RH < 1,00 (100 %)
- T_r, point de rosée : 0 °C < T_r < 50 °C

$$T_r = \frac{b\alpha(T,RH)}{a-\alpha(T,RH)}$$

avec : $\alpha(T,RH) = \frac{aT}{b+T} + \ln RH$

- a = 17,27 et b = 237,7 [°C].

Autre formule

$$T_r = \sqrt[8]{(H/100)} \cdot [112 + (0{,}9 \cdot T)] + (0{,}1 \cdot T) - 112$$

- T_r, point de rosée en °C
- T, température en °C
- H, humidité relative

J'ai donc schématisé la première formule avec des valeurs que j'ai observées sur mon écran OLED

CALCUL POINT DE ROSEE

$$Tr = \frac{b*a(T, RH)}{A-a(T, RH)} = \frac{237,7 * 0,4748}{17,27 - 0,4748} = 6,719\ °C$$

Cette valeur correspond à ce qui a été affiché sur l'écran

$$\text{avec : } a(T, RH) = \frac{A*T}{b+T} + \ln RH = \frac{17,27 * 26,61}{237,7 + 26,61} + \ln 38.92 = 0,4748$$

Légende :

a= Alpha = 0,4748
A= 17,27

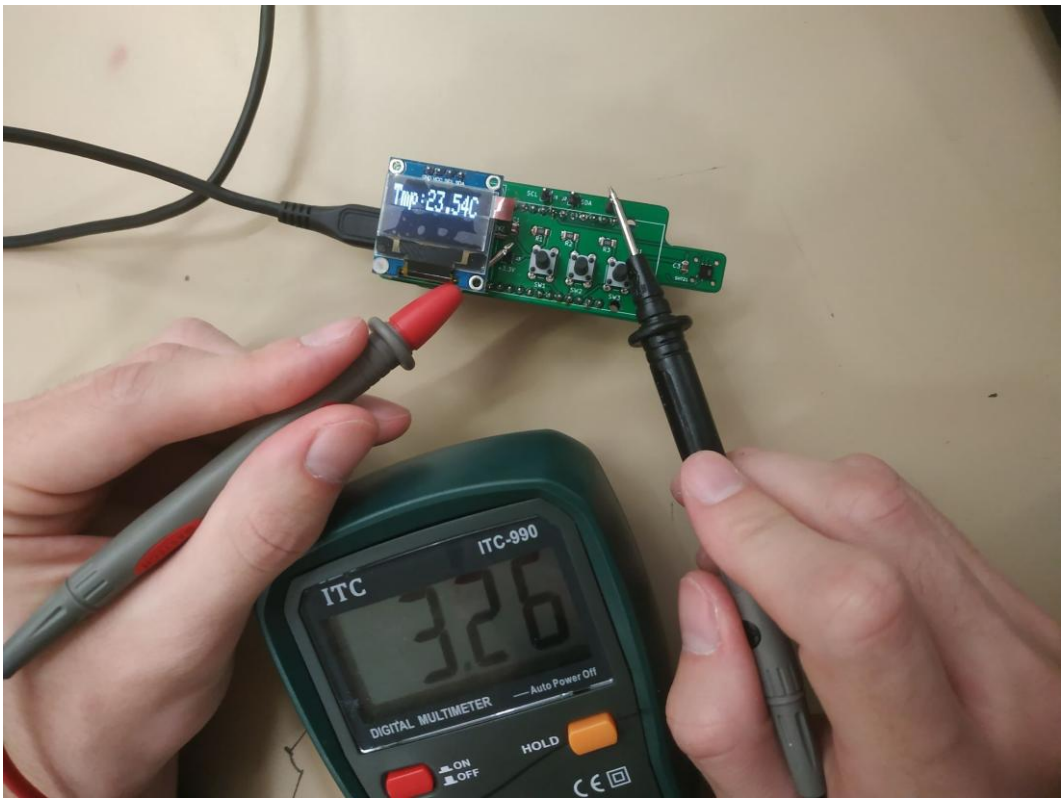
b= 237.7 [°C]

T = 26,61 — **(Température relevée sur l'écran)**
RH = 38,92 — **(Humidité relevée sur l'écran)**

Tous les calculs sont cohérents avec ce qui est relevé sur l'écran donc cela confirme les calculs.

Ayant des broches pour tester et vérifier la tension, j'ai utilisé un multimètre pour connaitre le voltage entre les broches 3.3V et GND.

Image du test avec multimètre :



La tension est cohérente mais on ne peut pas avoir 3.3V exactement. Cela n'affecte pas la conformité.

Conclusion

Dans le cadre de mon BTS Système Numérique option Electronique et Communication (EC), ce projet m'a permis de mettre en pratique mes connaissances théoriques acquises tout au long de ce cursus.

Il m'a été également bénéfique car cela m'a permis d'acquérir en autonomie.

De plus, ce projet a nécessité des connaissances dans des domaines non étudiés en cours. Cependant, l'accompagnement des professeurs tout au long du projet a permis de dépasser ces limites.

Ce projet a nécessité tout d'abord d'apprendre à travailler en équipe à partir d'un cahier des charges dont le référentiel a été respecté ainsi que le délai imparti.

Tout au long de sa concrétisation, j'ai pu également me rendre compte des difficultés techniques, théoriques et pratiques qui nécessitent des reprises. Au terme de ces 6 mois de projet, les tâches attendues ont été accomplies.

La carte shield respecte le cahier des charges et est opérationnelle tout en respectant la commande initiale. Travailler sur ce projet pratique complexe tout en bénéficiant d'un accompagnement a été une richesse dans mon cursus de formation.

Je remercie donc mes professeurs et Monsieur Bijou, gérant de Crossdock, pour la confiance et le soutien qu'ils m'ont donné.