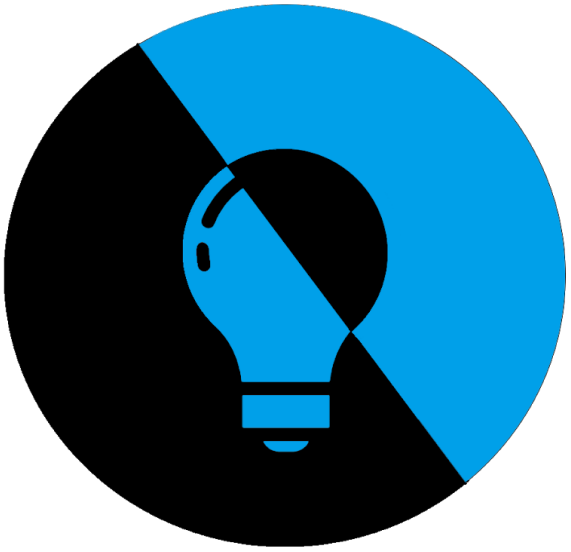


API GTC Lampadaire LoRaWan

Dossier de projet

Revue 3

Professeur référent :

Silanus Marc

Le commanditaire :

API, 402 avenue des Lacs 84270 VEDENE

Présentation du projet

L'entreprise API a été sollicitée par plusieurs communes pour développer une gestion technique centralisée (GTC) du réseau d'éclairage public (EP). Elle propose que la section de BTS SN développe un avant-projet sur ce sujet.

Situation du projet dans son contexte

L'éclairage public participe à la fois à la sécurité publique, en jouant un rôle important dans la perception nocturne des espaces publics (identification des différents usagers, perception de leur comportement, détection des obstacles éventuels de la voirie), à la convivialité et à l'embellissement des espaces publics en mettant en valeur le patrimoine et en créant des ambiances nocturnes agréables. Il est aujourd'hui au cœur des préoccupations dans la construction de la ville de demain et, ce, pour plusieurs raisons.

La première raison est économique. En France, 9 millions de lampes fonctionnent entre 3 500 et 4 300 heures par an pour une puissance installée d'environ 1 260 MW. L'éclairage public des villes représente près de la moitié de la consommation d'électricité des collectivités territoriales, soit 18 % de leur consommation toutes énergies confondues. Le poids de l'éclairage public dans la facture des collectivités est donc très important.

Il est donc nécessaire pour les collectivités territoriales d'investir dans des technologies intelligentes capables de faire baisser leur consommation d'électricité. Cela est d'autant plus vrai que, lors du Grenelle de l'environnement, une analyse de l'état des lieux des installations d'éclairage a fait apparaître d'importants besoins de rénovation. Plus de la moitié du parc est composée de matériels obsolètes (40 % des luminaires en service ont plus de 25 ans) et énergivores : boules diffusantes, lampes à vapeur de mercure (environ 1/3 du parc), etc...

Poussées par les contraintes réglementaires et budgétaires, mais également afin de concilier les enjeux sociaux, environnementaux et d'attractivité, les villes cherchent des solutions innovantes pour mieux gérer leur éclairage public et développer un mobilier urbain adapté aux attentes des citoyens.

Les « prérequis » de la gestion des services locaux liés au réseau d'éclairage

Un certain nombre de « prérequis » [cf. Commissariat général au développement durable, Études et documents, n° 73, novembre 2012], valables pour l'évolution de l'éclairage lui-même, mais aussi, pour les « extensions » sur le réseau qui tendent à se démultiplier ont été identifiés :

• la compatibilité avec la préservation des ressources de la planète et plus particulièrement les économies d'énergie et les émissions de CO2 ;

• les exigences de sécurité, continuité et qualité du service public ;

• une préoccupation majeure en matière de maîtrise de la dépense publique, avec un enjeu « central » concernant « l'investissement » ;
• un questionnement sur la possibilité d'appropriation (propriété, opérationnalité, maîtrise technologique) par la collectivité, notamment quand il s'agit d'un service délégué.

Ces « prérequis » correspondent aux termes d'une équation complexe dont la résolution ne relève pas exclusivement de la technique, même si la montée en puissance des technologies de l'information et de la communication qui s'opère depuis le milieu des années 1990 a déjà fait la preuve qu'elle peut apporter des solutions.

Agir sur l'interface équipement/réseau : la télégestion au point lumineux

La télégestion au point lumineux consiste à utiliser le réseau d'éclairage pour déployer un système de communication de type Intranet pilotant l'ensemble de l'infrastructure à distance, au moyen d'un boîtier électronique positionné sur le candélabre ou sur l'armoire de commande et connecté à un terminal informatique.

L'aspect « sécurité, continuité et qualité du service public » repose sur la conformité aux normes régissant l'éclairage et les réseaux de communication. Opérationnellement, le système de télégestion contribue à améliorer les qualités fonctionnelles des installations (durabilité des sources, etc.) et surtout à optimiser les opérations d'exploitation et de maintenance (réactivité, continuité, prévention, sécurisation des intervenants). Cette approche de la maintenance est également à l'origine d'une solution informatique de la maintenance (GMAO, gestion de maintenance assistée par ordinateur).

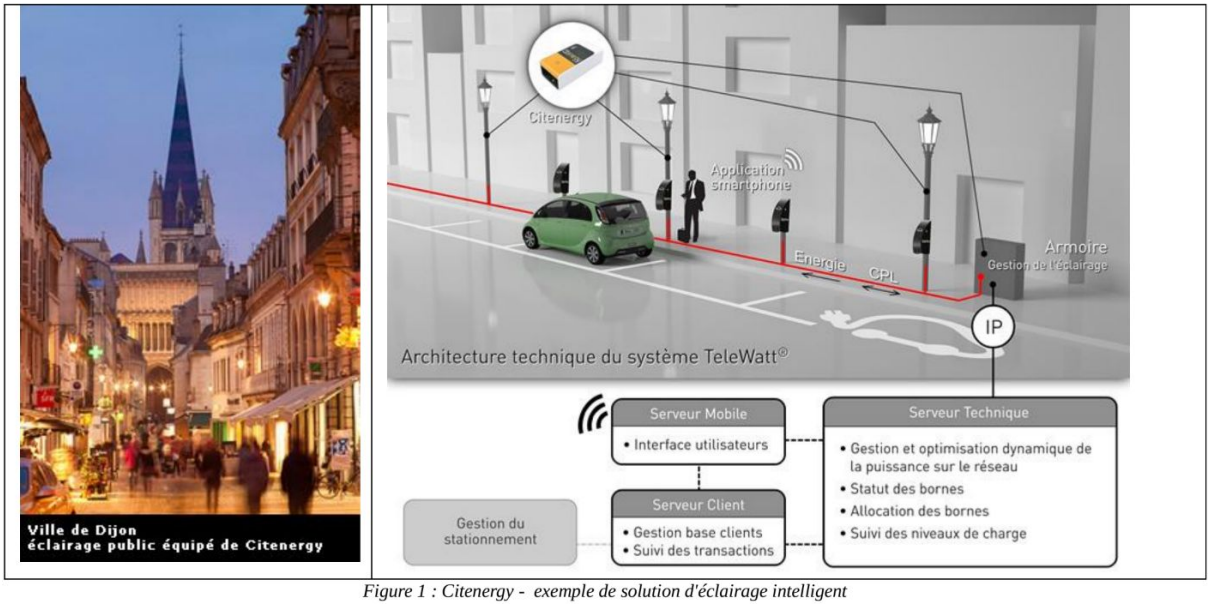
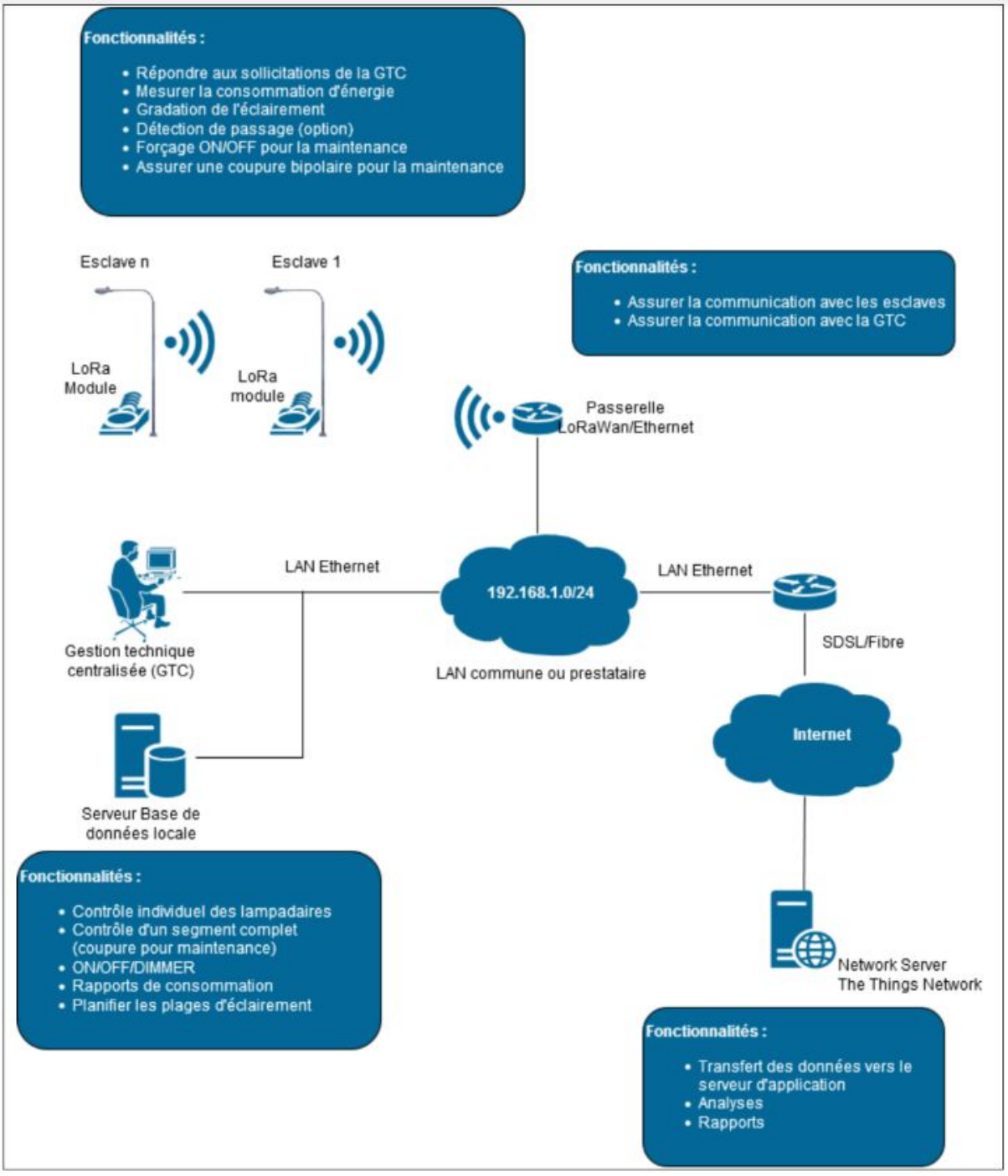
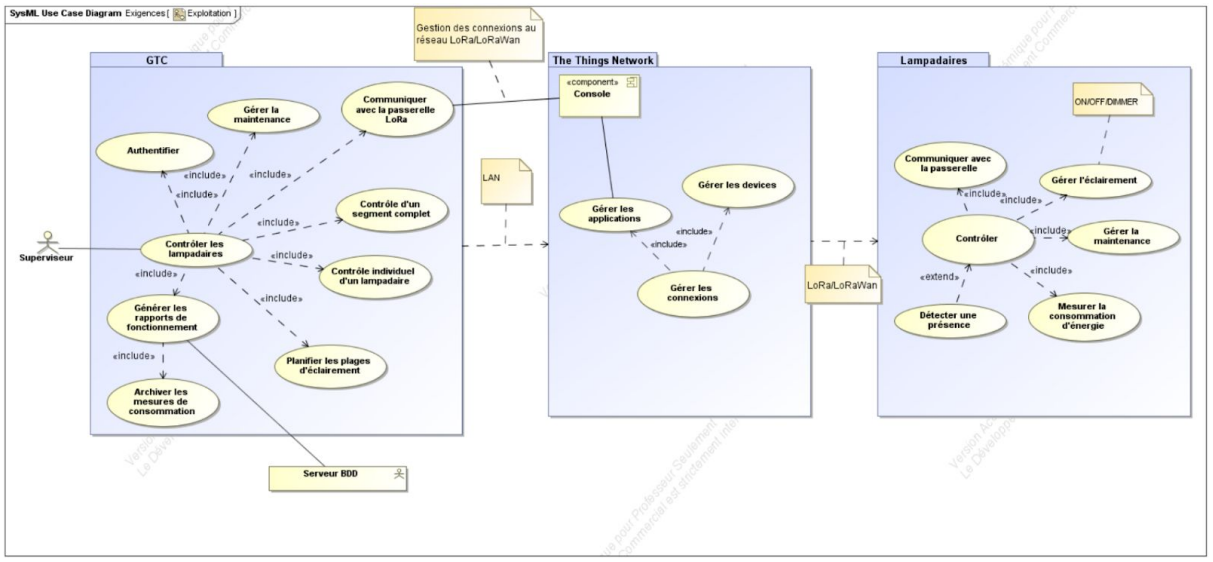
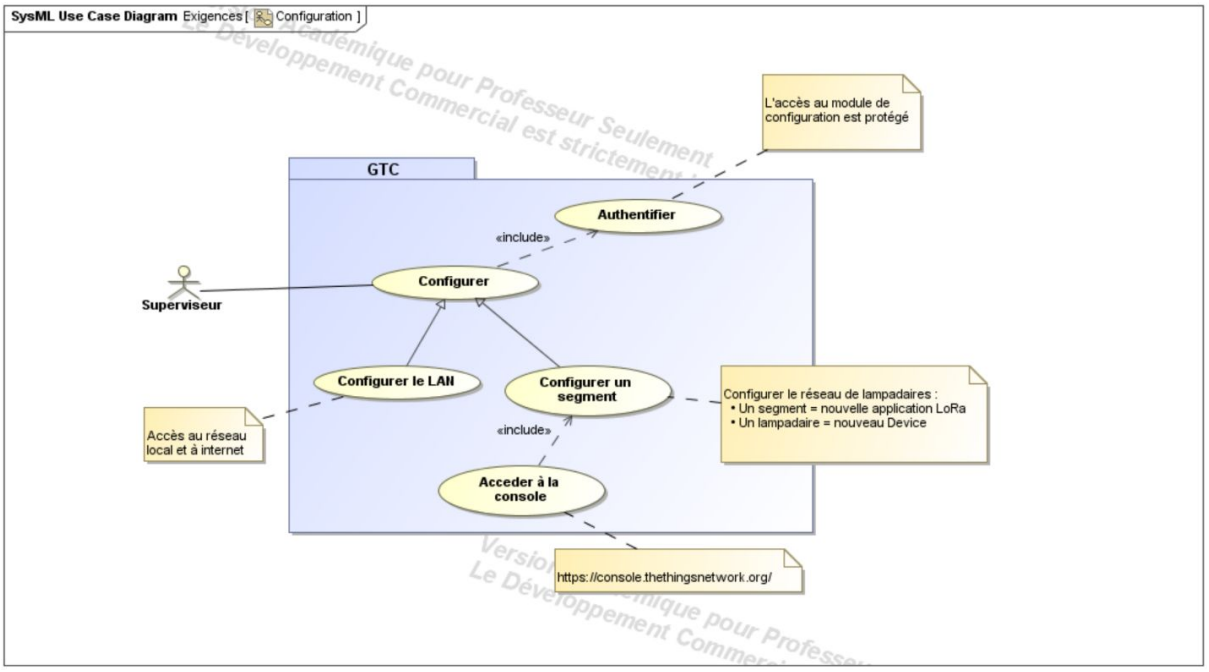
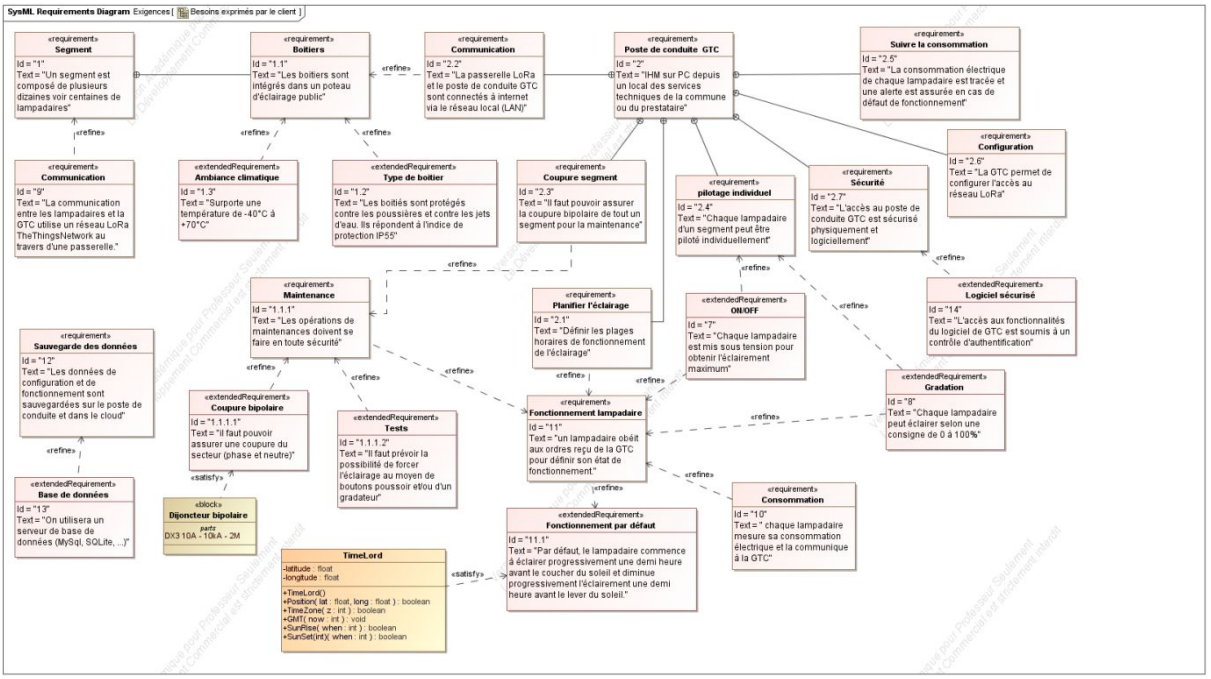


Figure 1 : Citenergy - exemple de solution d'éclairage intelligent


Fonctionnalités :
Répondre aux sollicitations de la GTC
Mesurer la consommation d'énergie
Gradation de l'éclairement
Détection de passage (option)
Forçage ON/OFF pour la maintenance
Assurer une coupure bipolaire pour la maintenance
Esclave n
Esclave 1
LoRa Module
LoRa module
Fonctionnalités :
Assurer la communication avec les esclaves
Assurer la communication avec la GTC
Passerelle LoRaWan/Ethernet
LAN Ethernet
192.168.1.0/24
LAN Ethernet
SDSL/Fibre
Gestion technique centralisée (GTC)
LAN commune ou prestataire
Internet
Serveur Base de données locale
Fonctionnalités :
Contrôle individuel des lampadaires
Contrôle d'un segment complet (coupure pour maintenance)
ON/OFF/DIMMER
Rapports de consommation
Planifier les plages d'éclairement
Network Server The Things Network
Fonctionnalités :
Transfert des données vers le serveur d'application
Analyses
Rapports


SysML Use Case Diagram Exigences [Exploitation]
Gestion des connexions au réseau LoRa/LoRaWan
GTC
The Things Network
Lampadaires
Gérer la maintenance
Communiquer avec la passerelle LoRa
Authentifier
«component» Console
LAN
Gérer les devices
Gérer les applications
Gérer les connexions
ON/OFF/DIMMER
Communiquer avec la passerelle
Gérer l'éclairement
Contrôler
Gérer la maintenance
Détecter une présence
Mesurer la consommation d'énergie
Superviseur
Contrôler les lampadaires
Contrôle d'un segment complet
Contrôle individuel d'un lampadaire
Générer les rapports de fonctionnement
Planifier les plages d'éclairement
Archiver les mesures de consommation
LoRa/LoRaWan
Serveur BDD
«include»
«extend»


SysML Use Case Diagram Exigences [Configuration]
L'accès au module de configuration est protégé
GTC
Authentifier
«include»
Superviseur
Configurer
Configurer le LAN
Configurer un segment
Accès au réseau local et à internet
Configurer le réseau de lampadaires :
• Un segment = nouvelle application LoRa
• Un lampadaire = nouveau Device
«include»
Acceder à la console
https://console.thethingsnetwork.org/


SysML Requirements Diagram Exigences [Besoins exprimés par le client]
«requirement» Segment
Id = "1"
Text = "Un segment est composé de plusieurs dizaines voir centaines de lampadaires"
«requirement» Boitiers
Id = "1.1"
Text = "Les boitiers sont intégrés dans un poteau d'éclairage public"
«requirement» Communication
Id = "2.2"
Text = "La passerelle LoRa et le poste de conduite GTC sont connectés à internet via le réseau local (LAN)"
«requirement» Poste de conduite GTC
Id = "2"
Text = "IHM sur PC depuis un local des services techniques de la commune ou du prestataire"
«requirement» Suivre la consommation
Id = "2.6"
Text = "La consommation électrique de chaque lampadaire est tracée et une alerte est assurée en cas de défaut de fonctionnement"
«requirement» Configuration
Id = "2.6"
Text = "La GTC permet de configurer l'accès au réseau LoRa"
«requirement» Communication
Id = "9"
Text = "La communication entre les lampadaires et la GTC utilise un réseau LoRa TheThingsNetwork au travers d'une passerelle."
«extendedRequirement» Ambiance climatique
Id = "1.3"
Text = "Surporte une température de -40°C à +70°C"
«extendedRequirement» Type de boitier
Id = "1.2"
Text = "Les boîtiés sont protégés contre les poussières et contre les jets d'eau. Ils répondent à l'indice de protection IP55"
«requirement» Coupure segment
Id = "2.3"
Text = "Il faut pouvoir assurer la coupure bipolaire de tout un segment pour la maintenance"
«requirement» pilotage individuel
Id = "2.4"
Text = "Chaque lampadaire d'un segment peut être piloté individuellement"
«requirement» Sécurité
Id = "2.7"
Text = "L'accès au poste de conduite GTC est sécurisé physiquement et logicielleement"
«requirement» Maintenance
Id = "1.1.1"
Text = "Les opérations de maintenances doivent se faire en toute sécurité"
«requirement» Planifier l'éclairage
Id = "2.1"
Text = "Définir les plages horaires de fonctionnement de l'éclairage"
«extendedRequirement» ON/OFF
Id = "7"
Text = "Chaque lampadaire est mis sous tension pour obtenir l'éclairement maximum"
«extendedRequirement» Logiciel sécurisé
Id = "14"
Text = "L'accès aux fonctionnalités du logiciel de GTC est soumis à un contrôle d'authentification"
«requirement» Sauvegarde des données
Id = "12"
Text = "Les données de configuration et de fonctionnement sont sauvegardées sur le poste de conduite et dans le cloud"
«extendedRequirement» Coupure bipolaire
Id = "1.1.1.1"
Text = "Il faut pouvoir assurer une coupure du secteur (phase et neutre)"
«extendedRequirement» Tests
Id = "1.1.1.2"
Text = "Il faut prévoir la possibilité de forcer l'éclairage au moyen de boutons poussoir et/ou d'un gradateur"
«requirement» Fonctionnement lampadaire
Id = "11"
Text = "un lampadaire obéit aux ordres reçu de la GTC pour définir son état de fonctionnement."
«extendedRequirement» Gradation
Id = "8"
Text = "Chaque lampadaire peut éclairer selon une consigne de 0 à 100%"
«extendedRequirement» Base de données
Id = "13"
Text = "On utilisera un serveur de base de données (MySql, SQLite, ...)"
«block» Disjoncteur bipolaire
parts
DX3 10A - 10kA - 2M
«extendedRequirement» Fonctionnement par défaut
Id = "11.1"
Text = "Par défaut, le lampadaire commence à éclairer progressivement une demi heure avant le coucher du soleil et diminue progressivement l'éclairement une demi heure avant le lever du soleil."
«requirement» Consommation
Id = "10"
Text = " chaque lampadaire mesure sa consommation électrique et la communique à la GTC"
TimeLord
-latitude : float
-longitude : float
+TimeLord()
+Position(lat : float, long : float) : boolean
+TimeZone(z : int) : boolean
+GMT(now : int) : void
+SunRise(when : int) : boolean
+SunSet(int)(when : int) : boolean
«refine»
«satisfy»

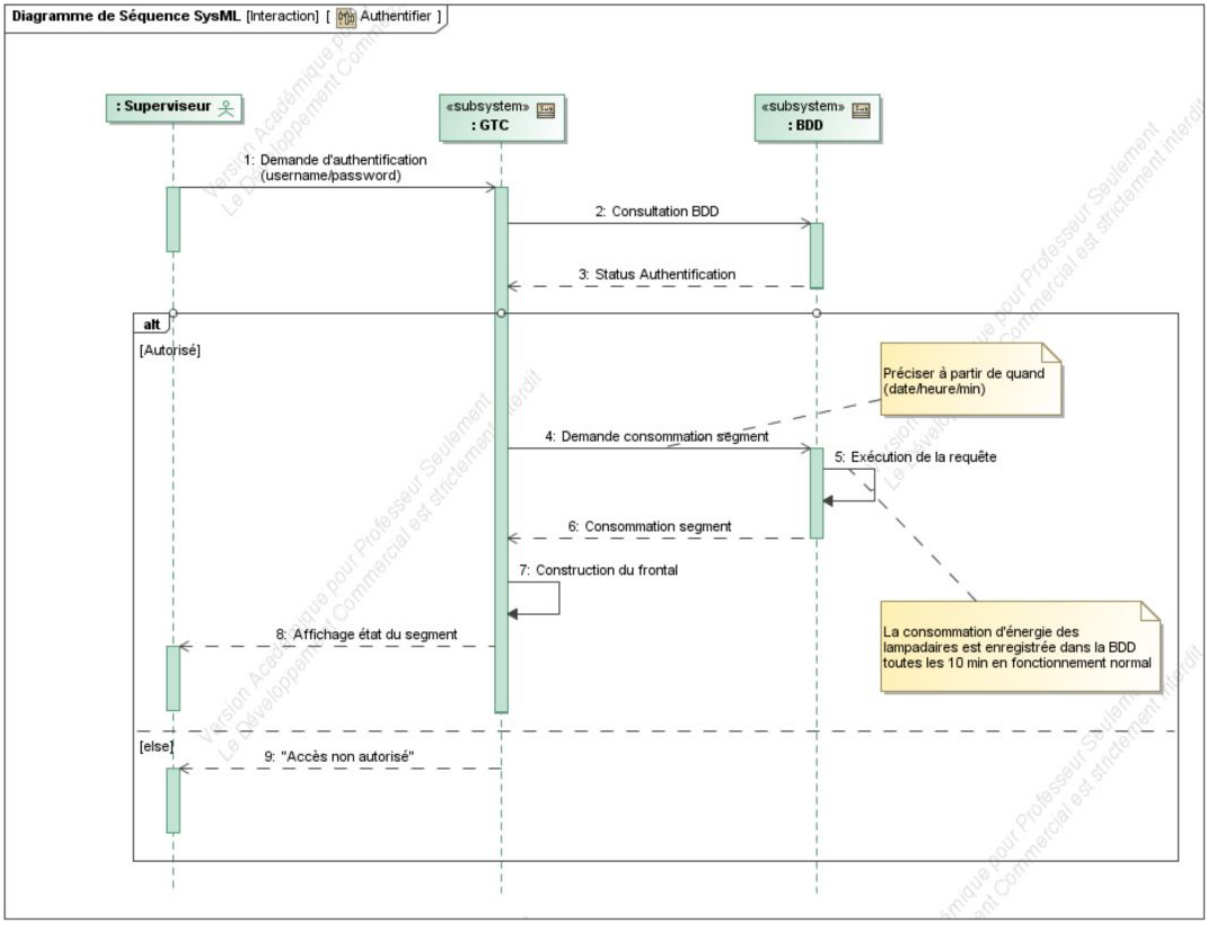


Diagramme de Séquence SysML [Interaction] [Authentifier]
: Superviseur
«subsystem» : GTC
«subsystem» : BDD
1: Demande d'authentification (username/password)
2: Consultation BDD
3: Status Authentification
alt
[Autorisé]
Préciser à partir de quand (date/heure/min)
4: Demande consommation segment
5: Exécution de la requête
6: Consommation segment
7: Construction du frontal
8: Affichage état du segment
La consommation d'énergie des lampadaires est enregistrée dans la BDD toutes les 10 min en fonctionnement normal
[else]
9: "Accès non autorisé"

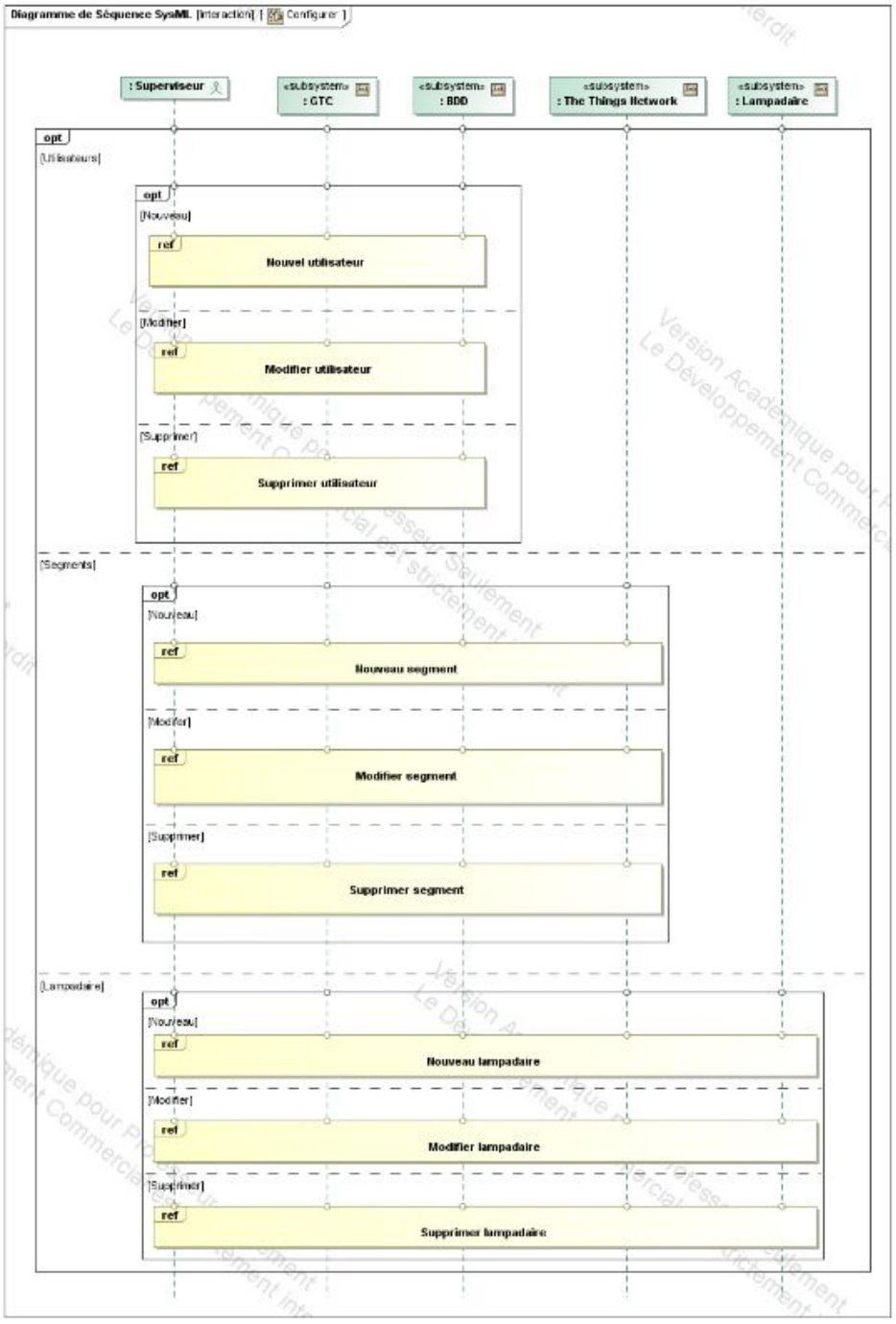
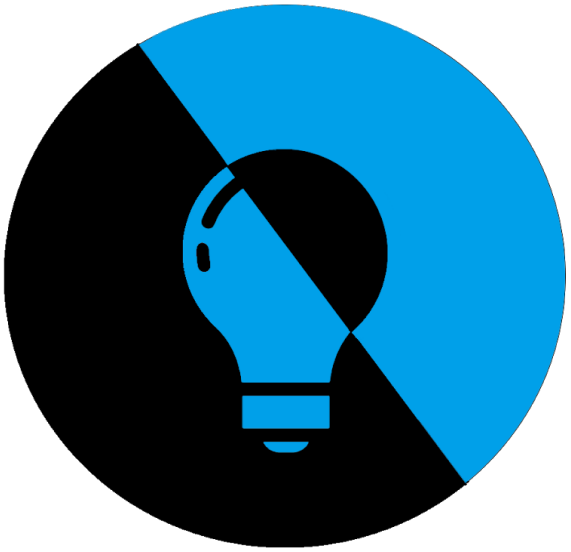


Diagramme de Séquence SysML [Interaction] [Configurer]
: Superviseur
«subsystem» : GTC
«subsystem» : BDD
«subsystem» : The Things Network
«subsystem» : Lampadaire
opt
[Utilisateurs]
opt
[Nouveau]
ref
Nouvel utilisateur
[Modifier]
ref
Modifier utilisateur
[Supprimer]
ref
Supprimer utilisateur
[Segments]
opt
[Nouveau]
ref
Nouveau segment
[Modifier]
ref
Modifier segment
[Supprimer]
ref
Supprimer segment
[Lampadaire]
opt
[Nouveau]
ref
Nouveau lampadaire
[Modifier]
ref
Modifier lampadaire
[Supprimer]
ref
Supprimer lampadaire



Dossier de projet

Partie IR1 : Serre Eliot

Professeur référent :

Silanus Marc

Le commanditaire :

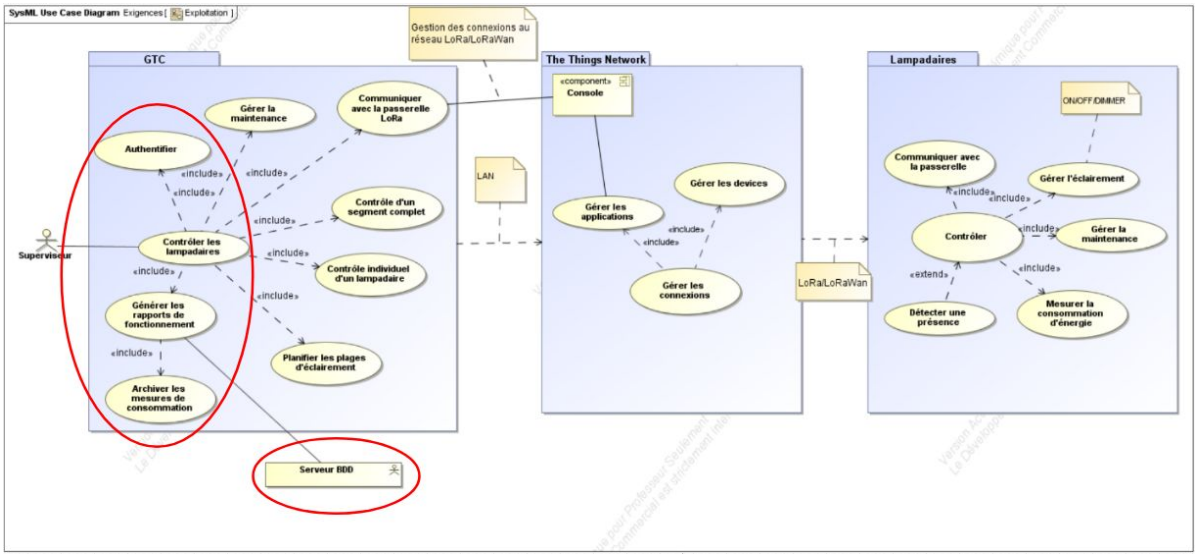
API, 402 avenue des Lacs 84270 VEDENE

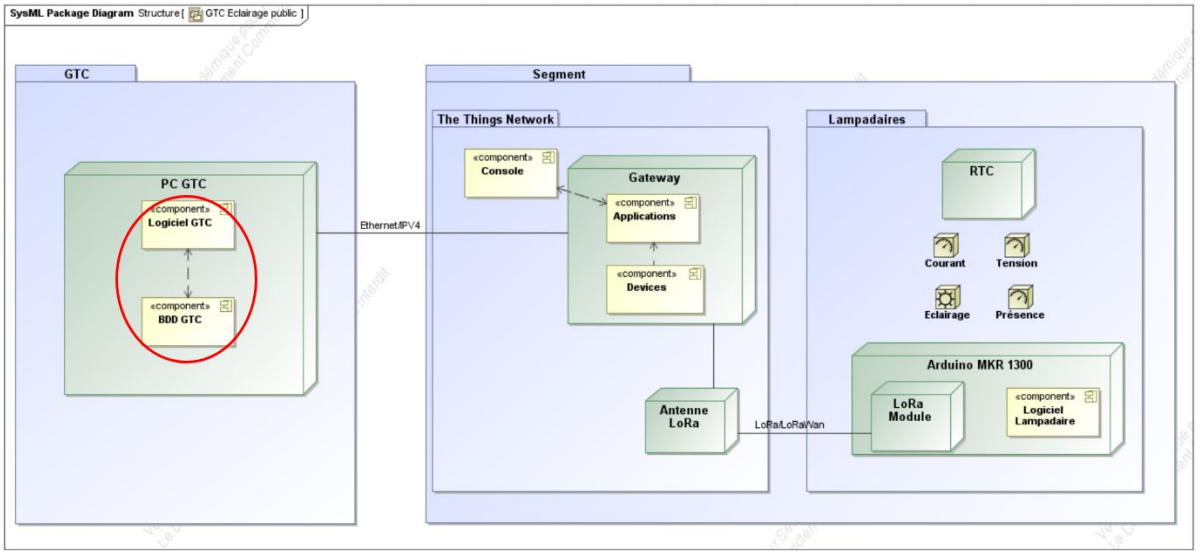
API GTC Lampadaire LoRaWan

Étudiant 1 **IR**	*Liste des tâches assurées par l'étudiant* **La Gestion Technique Centralisée (GTC)** • Production du logiciel de gestion qui répond aux cas d'utilisations de la GTC : "Configurer", "Générer les rapports de fonctionnement", "Archiver les mesures de consommation" et les cas d'utilisations inclus.	**Installation** : EDI QT Framework Qt/C++ Module QtMQTT Serveurs de bases de données **Mise en œuvre :** **Configuration** : Serveurs de bases de données **Réalisation** : BDD et tables Logiciel partiel GTC en Qt/C++ Bibliothèque C++ d'accès à la BDD sur GTC Bibliothèque C++ de contrôle d'un lampadaire **Documentation** : Documentation logicielle Guide d'utilisation rapide

Activités IR1	**186 h**	**Jeu 10/01/19**	**Ven 31/05/1**	
installations des logiciels et des	8 h	Jeu 10/01/19	Ven 11/01/19	2
réalisation	**178 h**	**Ven 11/01/1**	**Ven 31/05/1**	
BDD et tables	60 h	Ven 11/01/1	Mer 06/03/1	4
Bibliothèque C++ d'accès à la BDD sur	4 h	Ven 24/05/19	Mer 29/05/19	9
logiciel partiel GTC en Qt/C++	**118 h**	**Mer 06/03/19**	**Ven 31/05/19**	
grafana	110 h	Mer 06/03/1	Jeu 23/05/19	6
codage logiciel de base	4 h	Mer 29/05/19	Ven 31/05/19	7

J'ai été chargé de la mise en place des bases de données, la récupération des données et leur affichage, ainsi que l'accès aux données depuis l'application. Je m'occupe également d'une partie de la configuration impliquant la base de donnée, comme l'authentification.





Pour cela, je dois établir une communication sécurisée entre la base de donnée et The Thinks Network (TTN), plateforme en ligne permettant de communiquer via le protocole LoRa avec les lampadaires.

Qu’est ce que le LoRa ?

La technologie LoRa repose sur le réseau LoRaWAN (Long Range Wide-area network | réseau étendu à longue portée) et participe à l’Internet des Objets (IoT, Internet of Things). Des modules rattachés à des appareils vont envoyer des données à des passerelles (Gateway) qui seront ensuite dirigées vers des serveurs, dans notre cas ceux de TTN. Une application permet d’accéder à ces données des appareils qui y sont liés. Nous utilisons TTN car il a l’avantage d’être gratuit et open source alors que d’autres opérateurs comme Orange proposent un service payant, limitant ainsi les coûts aux modules (moins de 10€ par module, le prix peut être réduit avec la quantité commandée) et à la passerelle (jusqu’à 400€ pour les passerelles les plus performantes, cependant TTN fournit des plans pour monter soit même une passerelle).
Une passerelle LoRa a une portée d’environ 10km, et va capter les émissions des modules sans avoir besoin de créer un lien entre eux, permettant ainsi d’étendre le réseau LoRa grâce à des passerelles mises en place par des particuliers comme des collectivités.

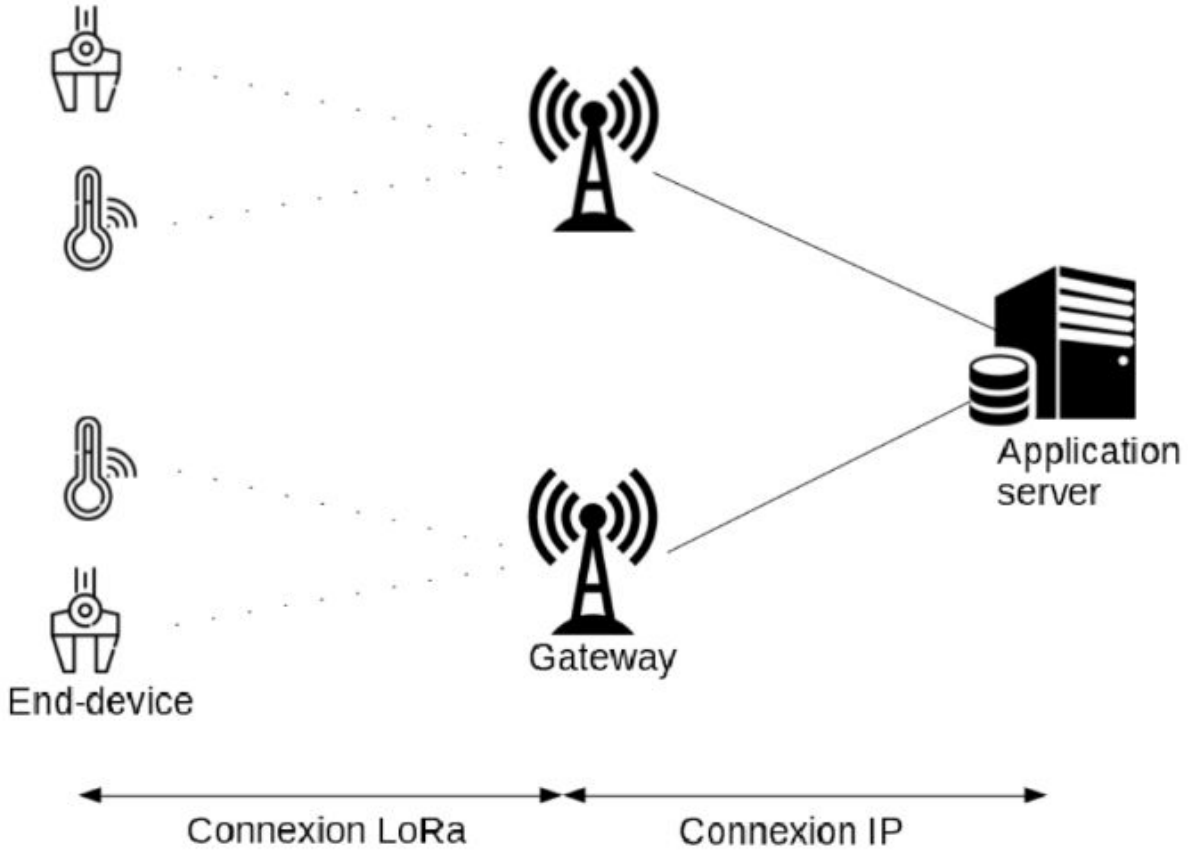


Schéma de fonctionnement du LoRa.

Le LoRa, communiquant à une fréquence autour des 868 MHz, possède 3 classes de fonctionnement:
-La classe A: Permet d'envoyer un message au module uniquement après que celui-ci ait communiqué. C'est la classe la plus contraignante mais aussi la plus économe en énergie.
-La classe B: Permet d'envoyer un message au module à intervalle réguliers.
-La classe C: Permet d'envoyer un message au module sans la moindre contrainte. C'est la classe la plus énergivore.
A cause de la carte MKR1300 utilisée lors du projet, nous sommes restreint à l'utilisation de la classe A, bien que la classe C soit plus avantageuse. En effet, nous devrons attendre qu'un lampadaire envoie un message avant de pouvoir lui donner un ordre.
Ces données vont ensuite être stockées dans une base de données (BDD).

Comment stocker les données ?

Nous avions plusieurs options dans le cahier des charges pour le stockage des données: mariadb/mysql, sqlite et influxDB. Sqlite est une version adaptée à l'équipement embarqué de mysql, nous pouvons donc nous permettre d'utiliser une option plus performante. Bien que nous avons utilisé mysql durant l'année, notre choix s'est porté sur influxDB car c'est une base de données temporelle, gérant le temps nativement. Le langage utilisé par InfluxDB, le langage InfluxQL, est proche du SQL de mysql, simplifiant son apprentissage.

InfluxDB a l'avantage de fonctionner sous Linux et aussi sous Windows. Seul une interface en ligne de commande est disponible pour ce logiciel open source et donc gratuit.
La base de données nous sert à stocker les lampadaires, les segments les regroupant et leurs données. Une table est également prévue pour les utilisateurs et leurs mots de passe d'accès à l'application de contrôle des lampadaires (GTC).

```
> INSERT adresse,Segment=testslampadaire1,Lampadaire=lampadaire1 nb=1 1000000000
000
> select * from adresse

name: adresse
time                Lampadaire      Segment          nb
----                ----------      -------          --
1000000000000000000 arduinomkr13002 testslampadaire1 1
1000000000000000000 cartechristo    testslampadaire1 1
1000000000000000000 lampadaire1     testslampadaire1 1
>
```

Table contenant les lampadaires et les segments.

Les tables contenant les lampadaires et segments ainsi que celle des utilisateurs pourront être modifiées depuis la GTC, et la table des données est remplie à partir des retours de TTN. Ces retours sont obtenu à l'aide de Telegraf.

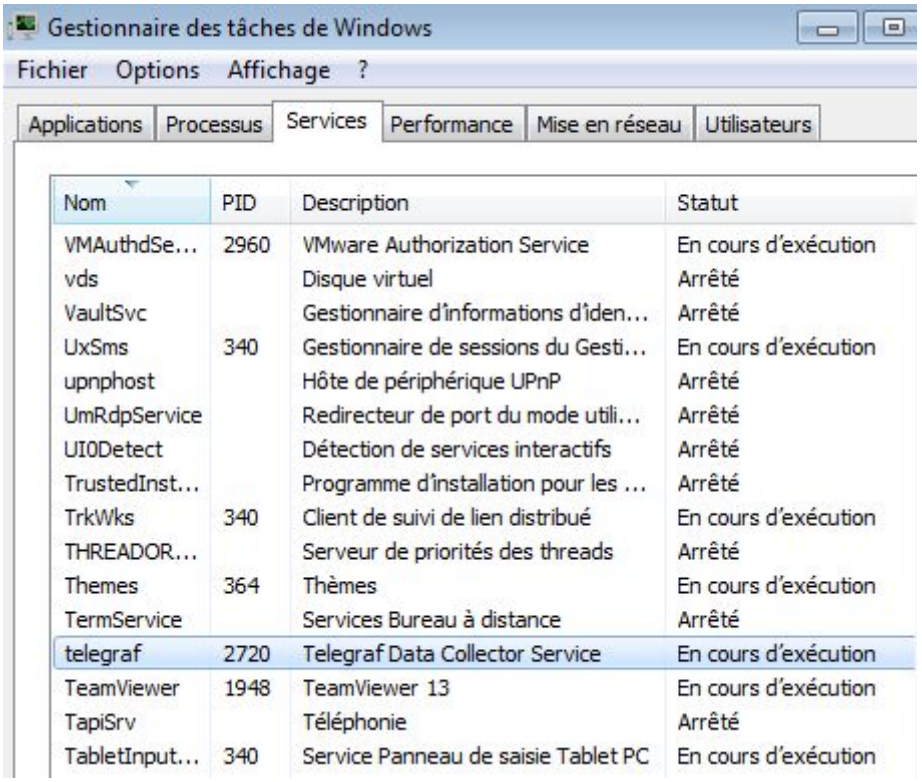
Comment les recevoir ?

Telegraf est un agent de collecte open source produit par InfluxData, la société à l'origine d'InfluxDB, et a été conçu pour fonctionner avec ce dernier. Son rôle est de récupérer les informations de TTN pour remplir la BDD. Pour cela, nous utilisons des plugins d'entrée et de sortie à placer dans le fichier .conf de Telegraf. Après avoir commenté les lignes des plugins servant d'exemple (récupération de l'état de la machine), j'ai pu placer le plugin de sortie vers InfluxDB et le plugin d'entrée en MQTT paramétré pour avoir TTN comme source.
Le paramétrage choisi permet à Telegraf de générer automatiquement la table contenant les données fournies par TTN, et pour cela un compte pour InfluxDB fut mis en place pour Telegraf, lui donnant ainsi tous les droits d'édition sur la base de données créée.

```
###############################################################################
#                                  INPUTS                                     #
###############################################################################

# Windows Performance Counters plugin.
# These are the recommended method of monitoring system metrics on windows,
# as the regular system plugins (inputs.cpu, inputs.mem, etc.) rely on WMI,
# which utilize more system resources.
#
# See more configuration examples at:
#   https://github.com/influxdata/telegraf/tree/master/plugins/inputs/win_perf_counters

[[inputs.mqtt_consumer]]
    servers = ["tcp://eu.thethings.network:1883"]
    qos = 0
    connection_timeout = "30s"
    topics = [ "+/devices/+/up" ]
    client_id = ""
    username = "testslampadaire1"
    password = "ttn-account-v2.Ey9GH4WrQ[redacted]"
    data_format = "json"


# [[inputs.win_perf_counters]]
```

Extrait du fichier .conf, ici le plugin d'entrée MQTT.

Sous Windows Telegraf s'installe sous la forme d'un service windows. Cela nécessite l'utilisation du Windows Powershell (une évolution de la console de commande). Avec quelques lignes il est ainsi possible d'activer Telegraf, de l'éteindre ou de le relancer. Il est requis de relancer le service pour appliquer les changements possibles. L'avantage d'installer Telegraf en service est qu'il est actif tant que la machine est en route.

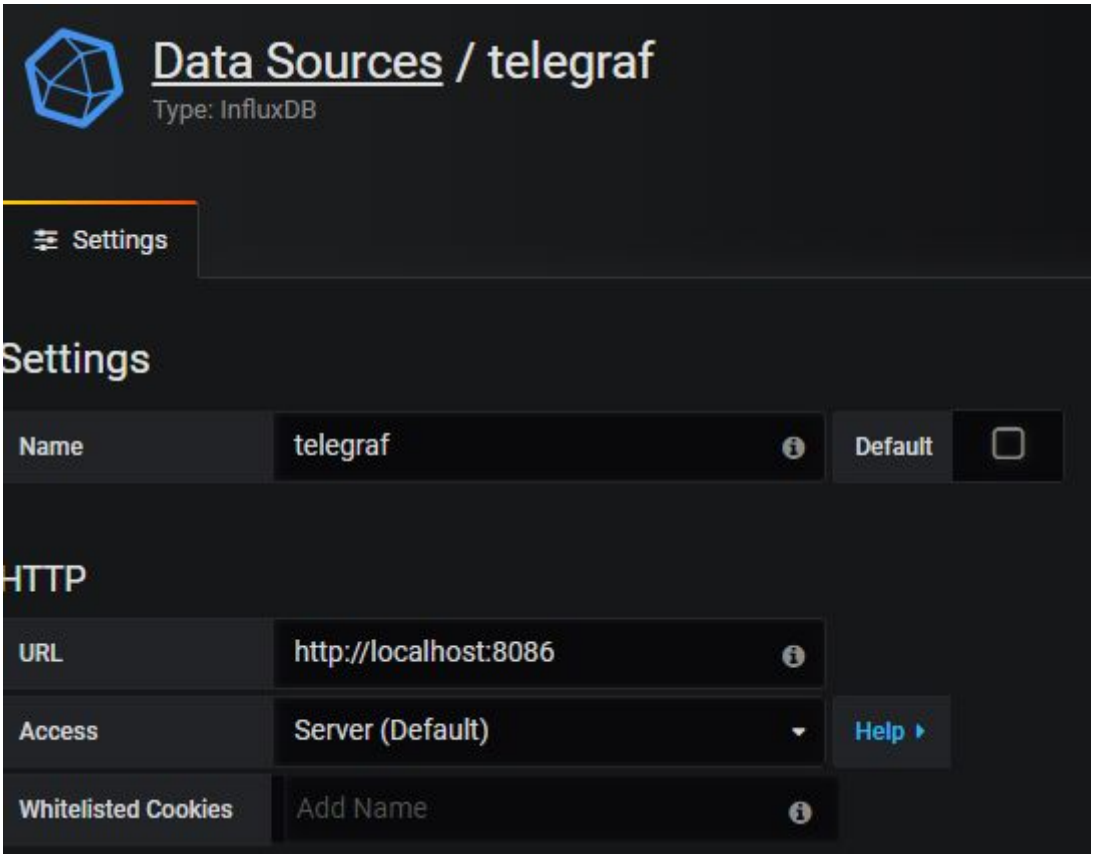


Le service telegraf en cours d'exécution sous windows

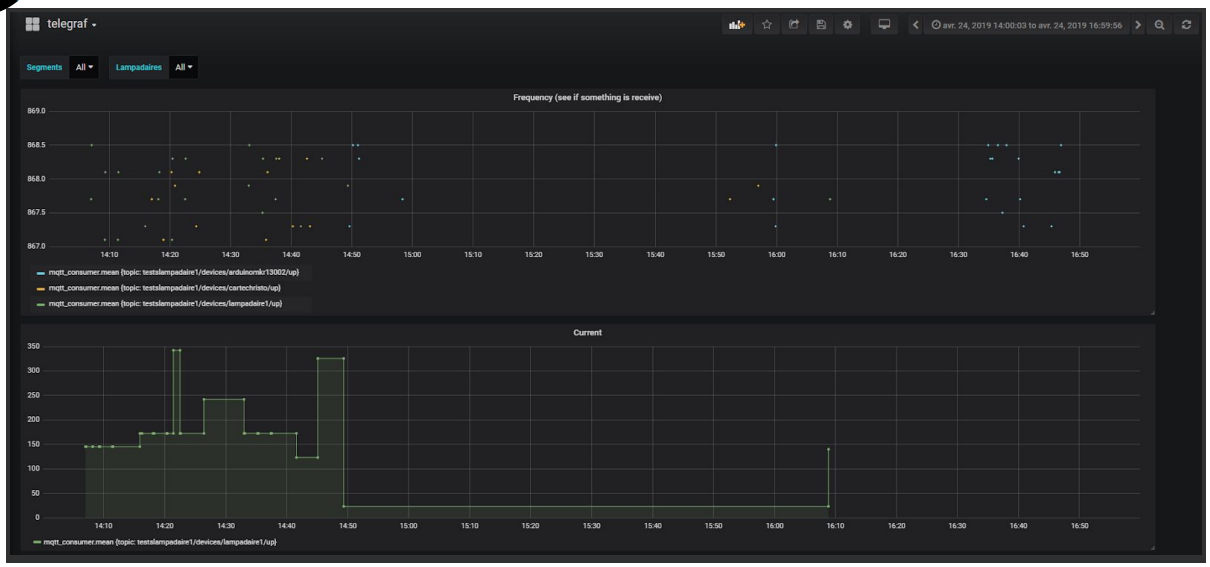
Une fois le lien entre TTN et la base de données effectué et cette dernière paramétrée, nous voulons afficher les données reçues.

Comment les afficher ?

Pour l'affichage des données nous nous servons du logiciel open source grafana. Celui-ci permet, après paramétrage, d'afficher facilement des données de manière intelligible pour l'utilisateur. Grafana intègre également un lien natif vers InfluxDB en tant que source de données.



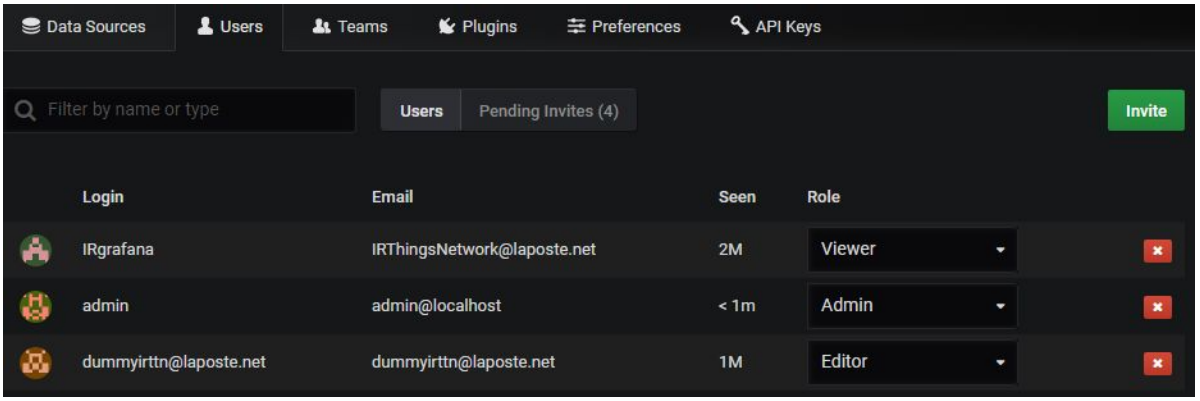
Fenêtre de configuration de la source où récupérer les données (ici la table "telegraf" remplie par telegraf).



Affichage du dashboard actuel de grafana.

Grafana fonctionne avec des dashboards, une interface contenant un ou plusieurs panneaux servant à afficher des données. Un dashboard peut correspondre à un lampadaire, mais le choix fut fait d'utiliser des variables pour plus de souplesse dans l'affichage et plus d'ergonomie pour l'utilisateur.

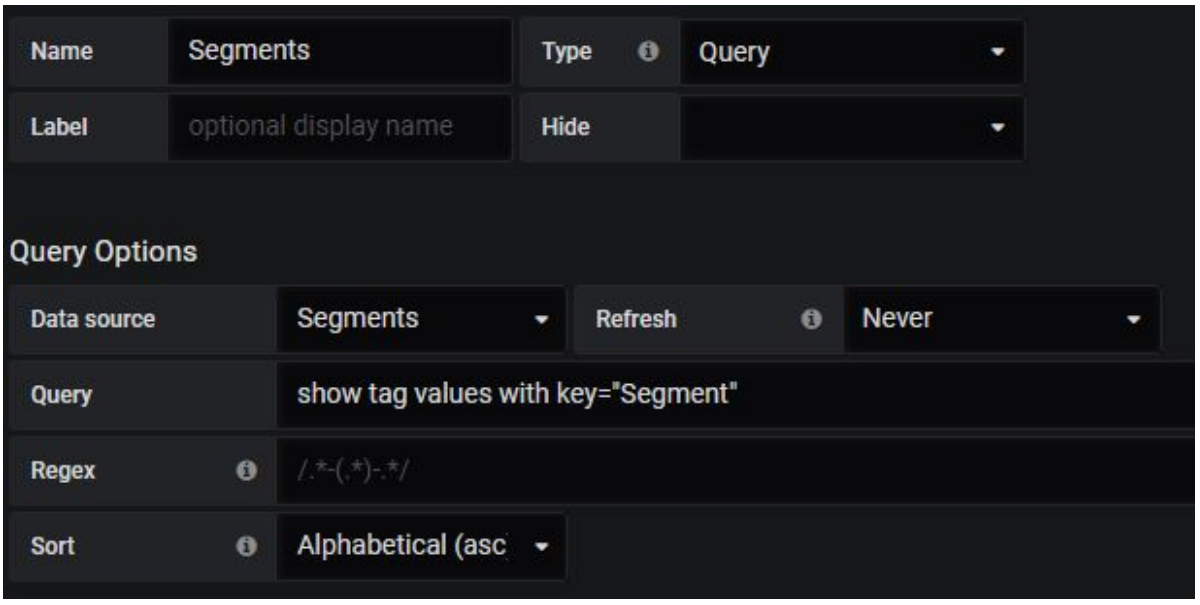
L'accès à grafana se fait via un compte ayant l'autorisation de consulter le dashboard, un compte pouvant avoir l'un des 3 droits d'accès suivants:
-Admin: Possibilité de tout faire, il vaut mieux éviter d'utiliser ce rôle si ce n'est pas nécessaire.
-Editor: Moins de droits qu'un admin, il reste possible de modifier les dashboards à sa guise.
-Viewer: Peut uniquement voir les dashboards et interagir avec les variables. Il est préférable que les utilisateurs courants aient ce rôle pour éviter toute erreur de manipulation.



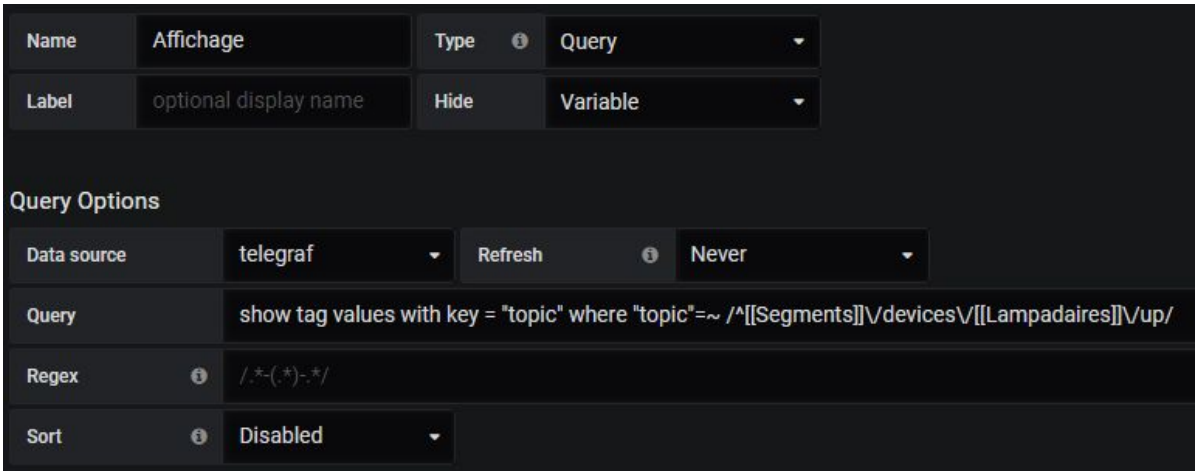
Interface des rôles de grafana, vu par l'utilisateur admin.

Les courbes présentes dans le dashboard sont celle de l'intensité du courant (en bas) et des fréquences (en haut). Cette dernière n'a pas d'utilité pour le client, mais permet lors de la phase de test de voir si un message a bien été envoyé par les lampadaires, et ce même

si les données reçues sont inexploitables. Par exemple, seule une courbe verte correspondant aux valeurs retournées par une carte de test est visible à cause d'un mauvais traitement du payload en amont.
L'utilisation de variables permet l'automatisation des dashboards: plutôt que de créer un dashboard par lampadaire et par segment pour voir leurs données séparément, les variables permettent d'afficher les données du nombres de lampadaires et des segments souhaités sans avoir besoin de changer de dashboard. Leur usage a nécessité des recherches plus poussées sur des points plus spécialisés du langage influxQL, ainsi que sur la possible utilisation des expressions régulières (regex). En effet les regex peuvent permettre de sélectionner avec plus de précision des champs dans une table (par exemple tout ceux commençant par le nom du segment).

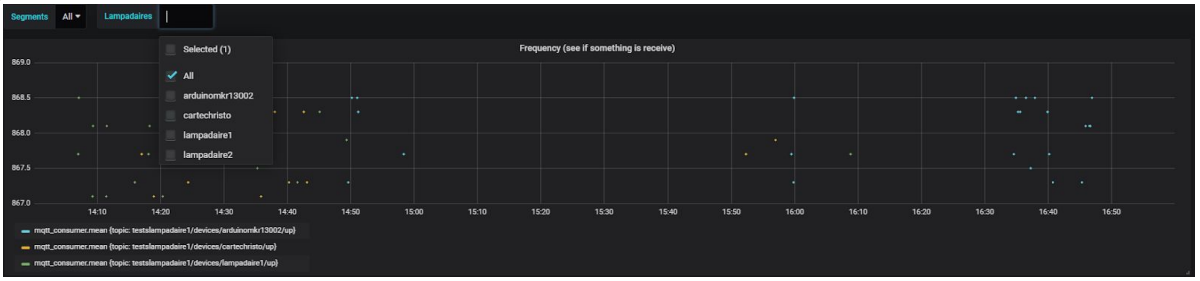


Variable assez basique utilisée pour le choix du segment.

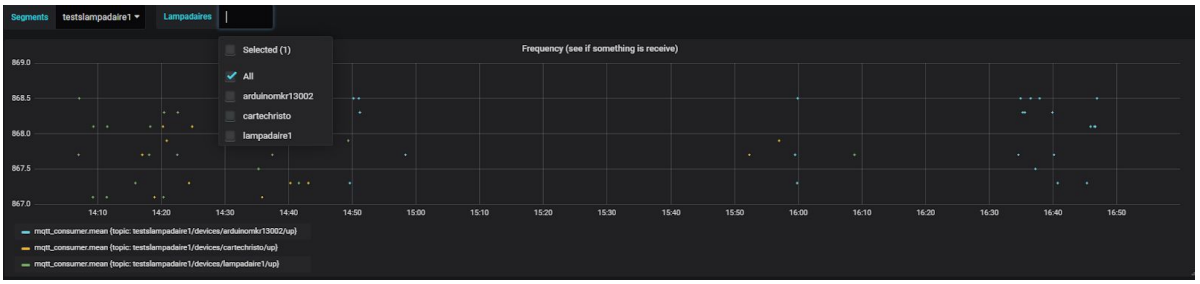


Variable plus complexe servant à l'affichage.

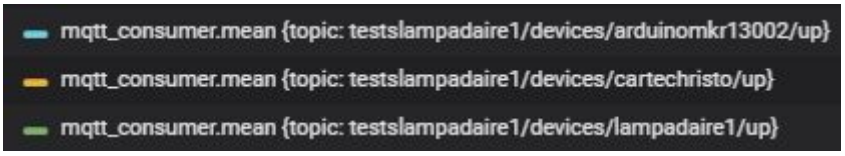
L'utilisateur peut choisir quel(s) segment(s) grâce à une première variable, puis quel(s) lampadaire(s) en particulier. Par défaut, tous les segments et lampadaires sont visible.



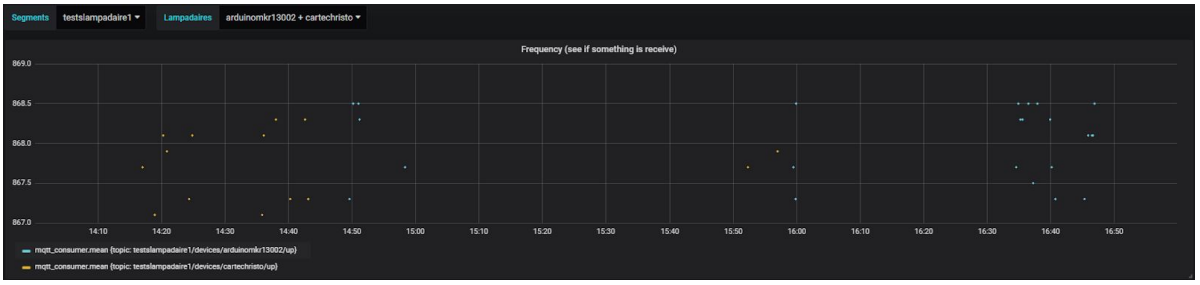
Tous les segments sont sélectionnés, l'utilisateur voit tout les lampadaires.



Seul le segment "testlampadaire1" est sélectionné donc seuls ses lampadaires sont visible.



La légende indique à quel segment appartient chaque lampadaire.

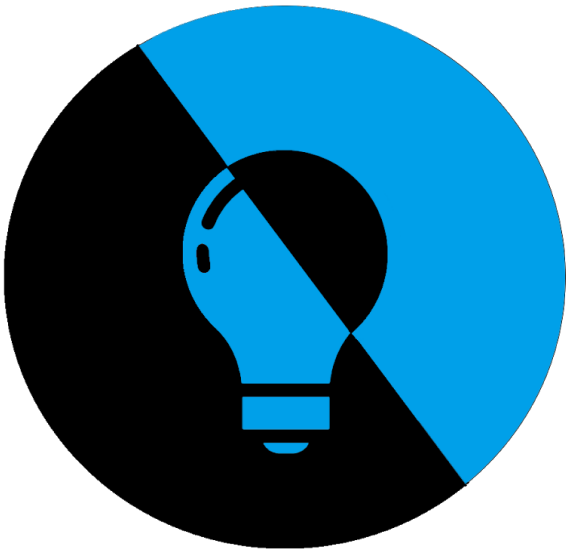


2 lampadaires sont sélectionnés dans le segment "testlampadaire1", seuls leurs données apparaissent, y compris dans la légende.

L'accès à la visualisation des données est assuré et protégé par grafana, mais il faut pouvoir influer sur certaines tables, notamment si l'on doit rajouter un lampadaire voir un segment. Cette fonctionnalité est assurée par la GTC.

Comment protéger les données ?

Tout comme l'accès à grafana, il faut sécuriser l'accès à la GTC. Je travaille actuellement sur la mise en place de l'authentification vers la GTC. Des tests ont été fait avec un identifiant entré en dur dans le code, la structure de la table qui devra accueillir les identifiants est également en cours de conception.



Dossier de projet

Partie IR2 : Cerda Aymeric

Professeur référent :

Silanus Marc

Le commanditaire :

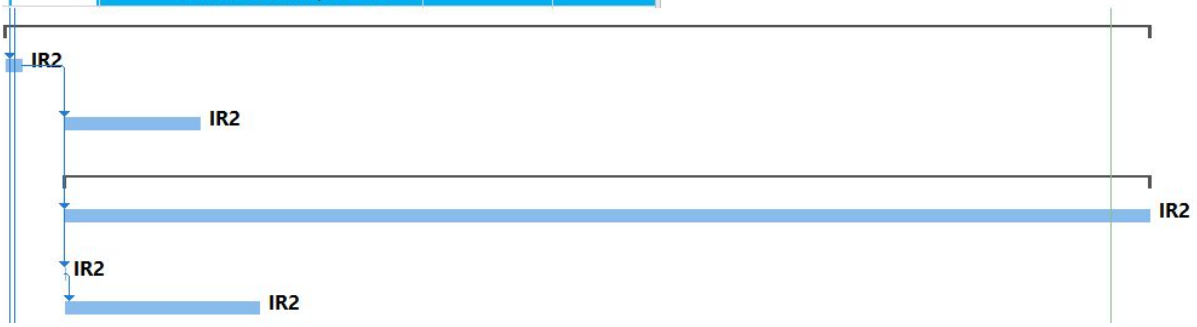
API, 402 avenue des Lacs 84270 VEDENE

API GTC Lampadaire LoRaWan

Je m'occupe de la communication entre les lampadaires et The Things Network,et de TTN à l'application GTC.

Ci dessous, le diagramme de Gantt qui sert à résumer la manière dont j'ai fait le projet, il m'a servi de référence pendant le projet, j'ai également effectué d'autres tâches pour répondre au besoins et optimiser mon temps.

	activités IR2	**180 h**	**Mer 09/01,**
	installations des logiciels et des bibliothèques	10 h	Mer 09/01/
	conf console The Things Network	30 h	Mer 16/01/
	réalisation	**170 h**	**Mer 16/01,**
	logiciel partiel GTC en Qt/C++	170 h	Mer 16/01/
	liaison MQTT	2 h	Mer 16/01/
	bibliothèque C++ de contrôle de lampadaire	40 h	Mer 16/01/



Pour faire communiquer effectivement de multiples appareils et à bas coûts, que ce soit énergétique ou financier, nous utilisons le protocole de communication LoRaWan. Il nous permet de remplir les conditions d'utilisation se trouvant dans le cahier des charges:

- une très faible consommation d'énergie
- une faible perte de données
- des échanges de données avec le protocole LoRa

Le protocole LoRa nécessite sur les cartes Arduino un composant qui permet d'utiliser la communication en LoRaWan, et permet, lorsqu'on le lui demande, de chiffrer les données pour éviter que n'importe qui puisse les voir et les exploiter.

Ce protocole est notre choix pour ce projet, car The Things Network nous permet d'utiliser son réseau de communication radio de manière open source. Cela nous permet de faire plus de choses qu'avec un logiciel propriétaire, et est complètement gratuit. Et nous autorise aussi à communiquer des données de manière sécurisée en utilisant l'étalement de spectre.

L'étalement de spectre permet de garantir que les données arriveront en entier au récepteur, il qui permet de ne pas nous préoccuper de bruit et de l'effet Doppler (communication réussie entre un appareil immobile et un autre sur une moto).

THE THINGS
NETWORK

Pour faire le lien entre les lampadaires, InfluxDB et l'application, nous devons utiliser un opérateur qui permet d'utiliser un réseau LoRa avec un minimum de frais possibles (orange facture son service à 3€/mois et par appareils). C'est pour cela que nous allons utiliser le réseau The Things Network, qui permet de communiquer à bas débit et à très faible coût, ce qui correspond à notre cahier des charges.

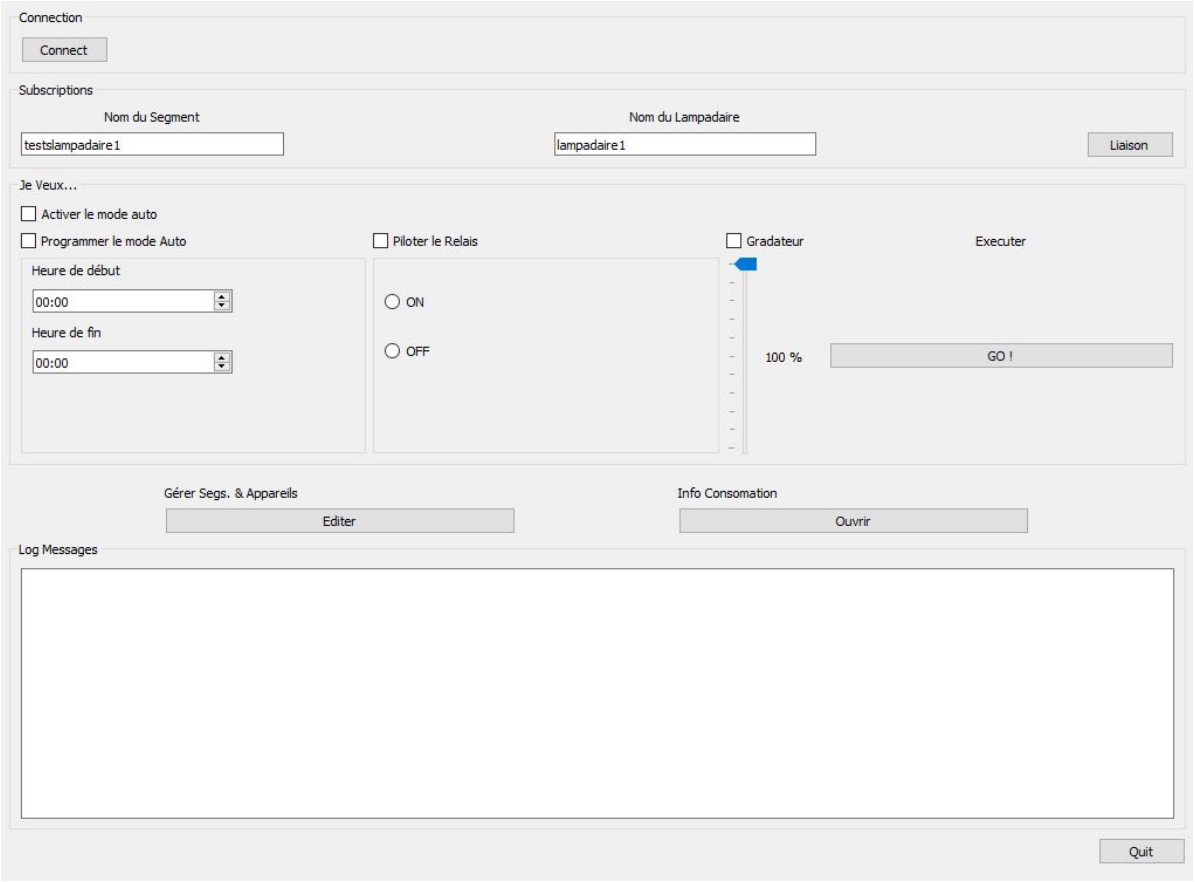
TTN (The Things Network) est un réseau LoRaWAN open source qui
peut être utilisé sans contrainte commerciale ou privée. il propose une solution ouverte pour créer (et utiliser) localement un réseau sans fil et bas débit.

TTN nous permettra de faire le lien entre les lampadaires, la BDD et l'application, en nous relayant via un relais les informations qui transiteront à travers l'application créée dans TTN, ce qui nous permettra de communiquer via le protocole MQTT.

MQTT signifie Message Queuing Telemetry Transport. C'est un protocole de messagerie de publication / abonnement extrêmement simple et léger, conçu pour les périphériques limités et les réseaux à faible bande passante, à latence élevée ou peu fiables. Les principes de conception consistent à réduire au minimum les besoins en bande passante réseau et en ressources des périphériques, tout en veillant à assurer la fiabilité et une certaine assurance de la livraison.

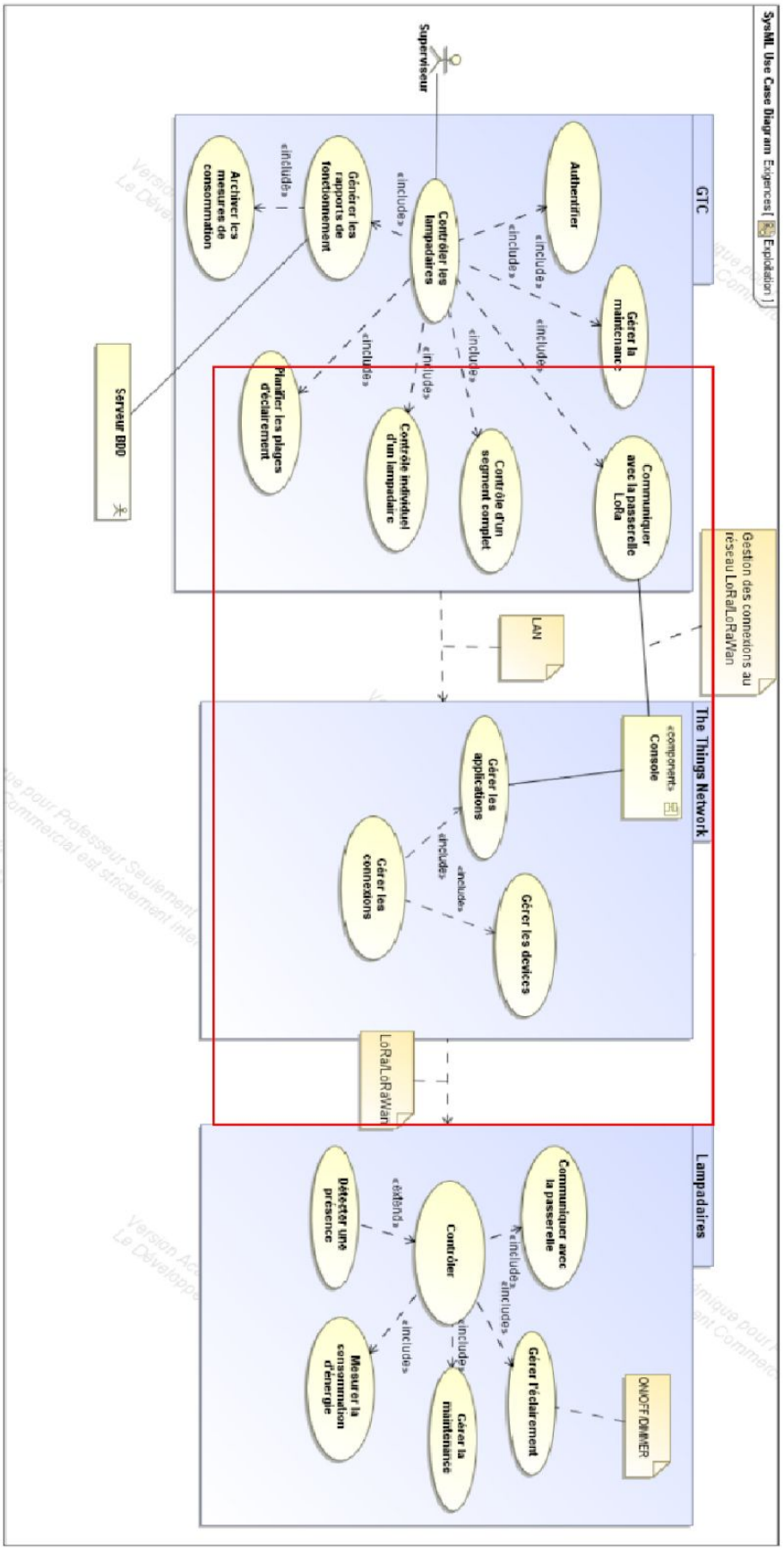
Ces principes ont également pour effet de rendre ce protocole idéal pour le monde émergent des appareils connectés «machine-à-machine» ou «Internet des objets», ainsi que pour les applications mobiles où la bande passante et la puissance de la batterie sont primordiales.

Dans le cadre du projet, on a dû créer une IHM pour pouvoir permettre aux utilisateurs de communiquer simplement avec la BDD et TTN pour envoyer des ordres, et recevoir des données.



Avec cet IHM, nous pouvons nous connecter à The Things Network, les paramètres étant codés en dur dans l'application, puisque l'application n'aura pas besoins d'en changer (elle restera en europe).
Pour la partie subscription, nous devons juste rentrer le nom de l'application et le nom de l'appareil, l'application créera une trame plus complète qu'elle enverra à TTN, et nous pourrons communiquer avec la base de données en InfluxQL, ainsi qu'avec les cartes de chaque lampadaire ou avec chaque segment.
Ici, nous ne pouvons que renvoyer des données, la visualisation des données se fera directement sur grafana.

Si après, le diagramme des cas d'utilisation, et entouré, là ou je vais principalement agir.



API GTC Lampadaire LoRaWan

Du cahier des charges à l'application:

Pour le projet, j'ai dû créer de toute pièces une application et une IHM fonctionnelle et confortable pour le client. Pour cela, je me suis appuyer sur le cahier des charges

(extrait du cahier des charges)

- Poste de conduite GTC : IHM sur PC depuis un local des services techniques de la commune ou du prestataire.
- Plusieurs centaines de segments d'EP. Variable en fonction des communes.

l'IHM présentée plus haut permet de répondre à ces deux besoins, techniquement, on peut soutenir ainsi une infinité de lampadaires, puisque il n'y a pas de limite dans le nombre de paramètres différents que l'on peut injecter dans la GTC.

(extrait du cahier des charges)

- Un segment comporte un candélabre dit « maître » et plusieurs candélabres « esclaves ». Le nombre d'esclave sur un segment n'est pas encore fixé.
    - Le « maître » de chaque segment est en liaison permanente Ethernet TCP/IP avec la GTC.
      Cette liaison sera de 2 types :
        - réseau fibre communal
        - OU passerelle 3G / Ethernet
- Le « maître » de chaque segment est en liaison avec tous les esclaves :

  A étudier :
    - Liaison radio
    - OU courant porteur en ligne.

Pour cette demande,elle a été simplement ignorée, car il n'existe pas de maître ou d'esclave pour les cartes,chaque carte peut émettre et recevoir à TTN sans problèmes, cela permet une meilleure flexibilité au niveau de l'architecture, puisqu'il suffit juste qu'une carte soit enregistrée à TTN pour qu'elle fonctionne.

(extrait du cahier des charges)

- Les fonctions des maîtres sont :
    1. recevoir les plages horaires de fonctionnement depuis la GTC
    2. Pilotage de tous les esclaves qui lui sont rattachés.
    3. Activation / inhibition indépendantes des esclaves.
    4. Contrôle du bon fonctionnement des esclaves :
       remontée à la GTC, des défauts de fonctionnement (mesure du courant consommé).
    5. Coupure bipolaire de tout le réseau pour la maintenance.

Les fonctions des Esclaves sont :

1. recevoir les commandes de « Marche / Arrêt »
2. Auto-contrôle de fonctionnement : mesure du courant consommé pour remonter les informations au maître.
3. Coupure bipolaire pour intervention ou isolement du lampadaire.

Comme dit précédemment, il n'y a pas de maîtres esclaves, mais néanmoins, chaque carte va envoyer sa consommation ainsi que l'heure de son horloge à elle (pour vérifier que le mode automatique fonctionne correctement (cette fonctionnalitée est en cours de développement)). On peut aussi passer des commandes d'allumage manuel.

(extrait du cahier des charges)

Variation de puissance

Etudier l'intégration d'une variation de puissance de l'éclairage (dimmer).

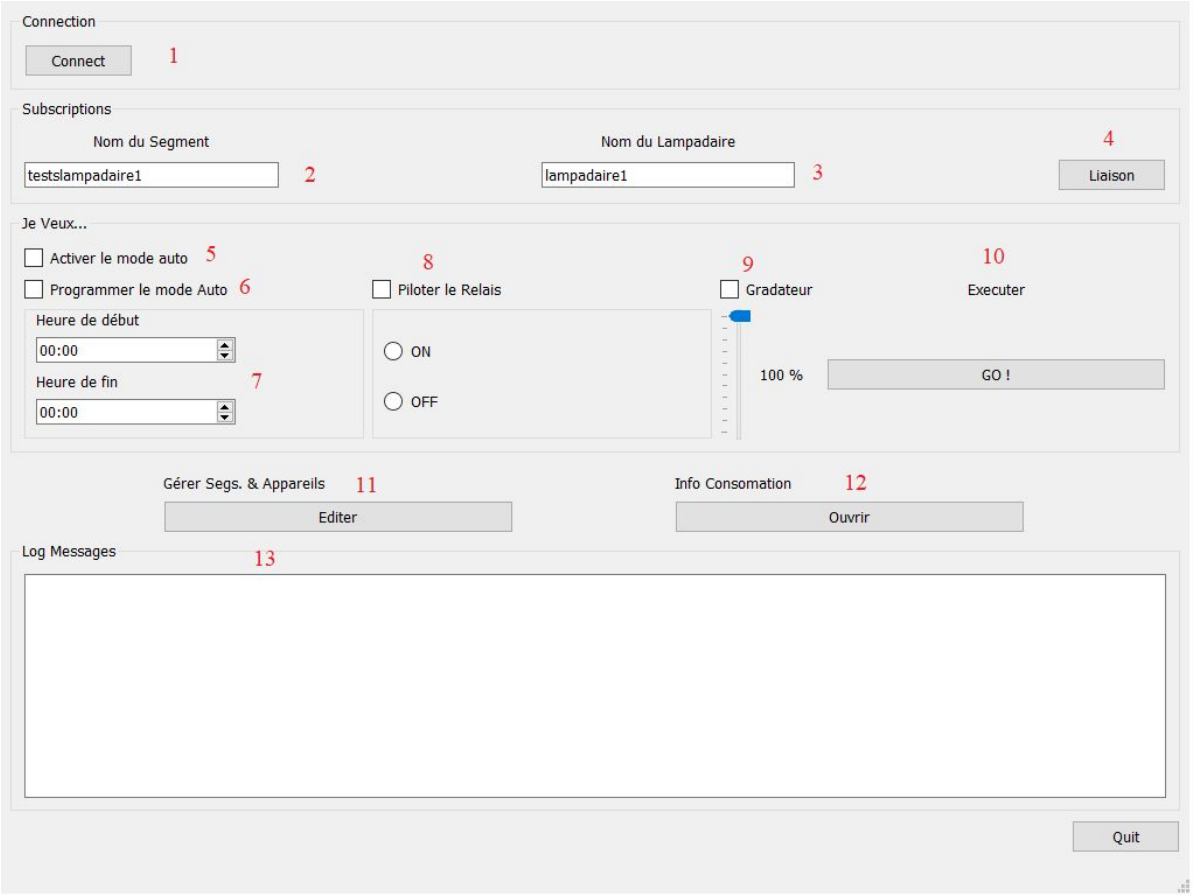
Détection :

Prévoir sur toutes les cartes (maitres et esclaves) la possibilité de raccorder une entrée TOR (détection de passage).

Maintenance :

En local sur chaque carte équipant un EP, prévoir 2 boutons de commandes de forçage pour les opérations de maintenance.

Mes coéquipiers en EC ont intégré un Dimmer que l'on peut utiliser si l'on passe les bon paramètres dans la trame et des boutons poussoirs pour forcer manuellement l'allumage et l'extinction du relais. Pour la détection, après avoir contacté le client, il s'est avéré que la détection ne serait pas nécessaire.



1: Ce bouton permet de se connecter au réseau TTN.

2: Cette zone de texte correspond au nom du segment.

3: Cette zone de texte correspond au nom du lampadaire, la possibilité de choisir tout les lampadaire de ce segment est en cours de développement.

A noter que tous les noms que ce soit de segment ou d'appareil doivent être sans espaces, caractères spéciaux, ni majuscules.

4: Ce bouton permet de souscrire à un topic, qui est la destination du message que l'on va envoyer. Pour changer de topic, il suffit de changer l'un des paramètres de la souscription, et de ré appuyer sur subscribe.

5: Cette case sert à envoyer une commande qui permet d'activer le mode automatique des lampadaires.

6: Cette case permet de donner des paramètres au mode automatique, elle ne l'active pas, elle sert juste pour les paramètres.

7: Lorsque la case "programmer le mode Auto" est cochée, il devient possible d'éditer les paramètres d'allumage (en haut) et d'extinction (en bas) de la lampe en mode Automatique.

8: Ce paramètre permet de piloter le relais.

9: Une fois coché, le gradateur permet de contrôler la puissance d'allumage de la lampe en mode manuel.

10: Permet d'envoyer la trame qui va permettre de faire exécuter les instructions données dans l'application par la lampe.

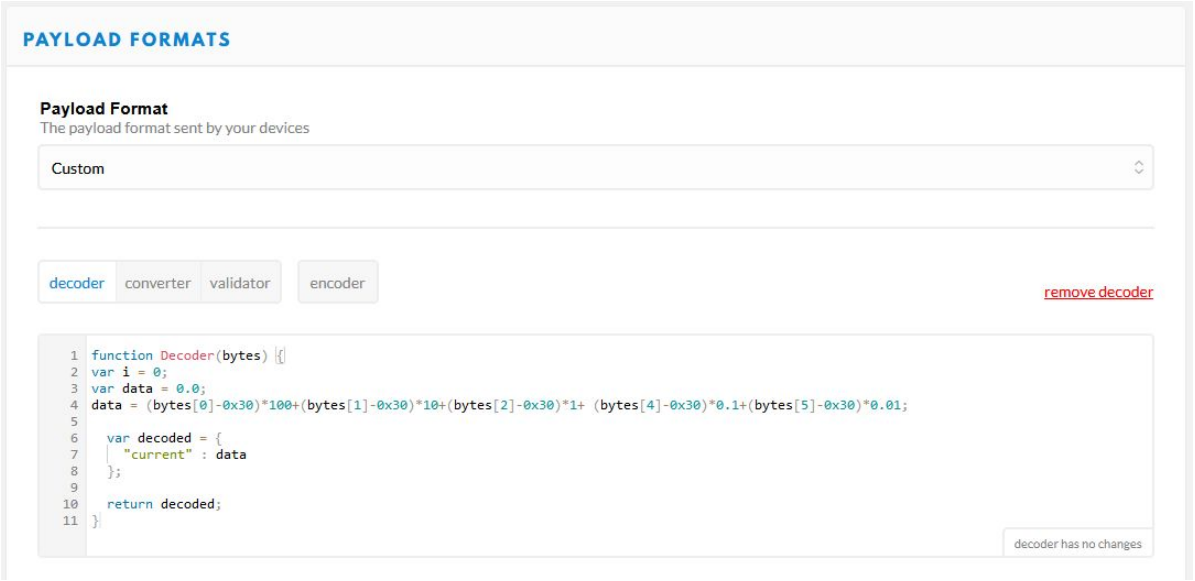
11: Permet d'ouvrir une seconde fenêtre qui permettra d'éditer les segments et appareils.

12: Permet d'ouvrir le navigateur web par défaut pour afficher graphiquement la consommation des appareils.

13: Cette zone de texte est réservée pour afficher les messages de tout ce qui se passe au niveau de l'application (connexion en cours, réussie…).

Après la conception de l'IHM, il fallait pouvoir établir une connexion en l'application et les appareils, et c'est là que The Things Network entre en jeu. Cette combinaison de serveur web + broker MQTT permet de mettre en oeuvre un système de communication a gratuite et à faible débit.

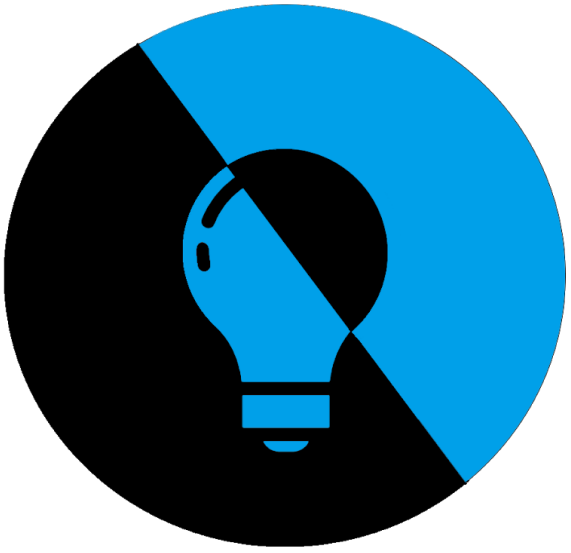
Le but premier de TTN est de servir de broker, c'est à dire, c'est un endroit ou les données doivent passer pour pouvoir transiter vers le bon endroit. En plus, on peut choisir de filtrer certaines données pour n'envoyer que l'essentiel, ou pour chiffrer les messages.



Pour notre projet, nous utilisons seulement le décodeur, puisque pour les autres fonctionnalités, soit on ne les utilise pas, soit cela sera utilisé en amont ou en aval.

Ici, dans le décodeur, nous allons extraire la valeur de la consommation qui se trouve dans un tableau de données, et mettre le tout dans une variable qui sera envoyé ensuite vers InfluxDB. A noter que les données de l'application ne sont pas traitées, c'est seulement les données qui proviennent de la carte. Chaque partie de la valeur (3 chiffres entiers, 2 fractionnaires) est extraite, et ajouté à une variable nommée "data". Puis, nous créons une nouvelle variable nommée "decoded" qui va correspondre à "data" mais avec comme propriété "current" qui va correspondre ici à "la donnée principale".

Ce que nous recevons dans TTN est un tableau de données. puisque nous ne pouvons pas traiter une donnée de la sorte, il nous faut la transformer. Pour cela, nous prenons toutes les parties de ce tableau, et les mettons dans une seule valeur, qui sera envoyé ensuite.



Dossier de projet

Partie EC1 : Paquier Christopher

Professeur référent :

Silanus Marc

Le commanditaire :

API, 402 avenue des Lacs 84270 VEDENE

API GTC Lampadaire LoRaWan

Introduction :

Sur le cahier des charges, plusieurs tâches m'ont été données :

• Tester et valider les éléments proposés sur le synoptique du prototype et sur le site du projet.

• Concevoir/Réaliser/Tester un shield Arduino intégrant les éléments retenus, y compris ceux de l'étudiant 3 EC pour fournir une solution complète.

• Développer une application (la plus avancée possible) permettant de mettre en évidence les différentes fonctionnalités du boitier Lampadaire.

Dans la suite de la revue de projet je vais expliquer toutes les tâches que j'ai pu réaliser durant mes heures de travails.

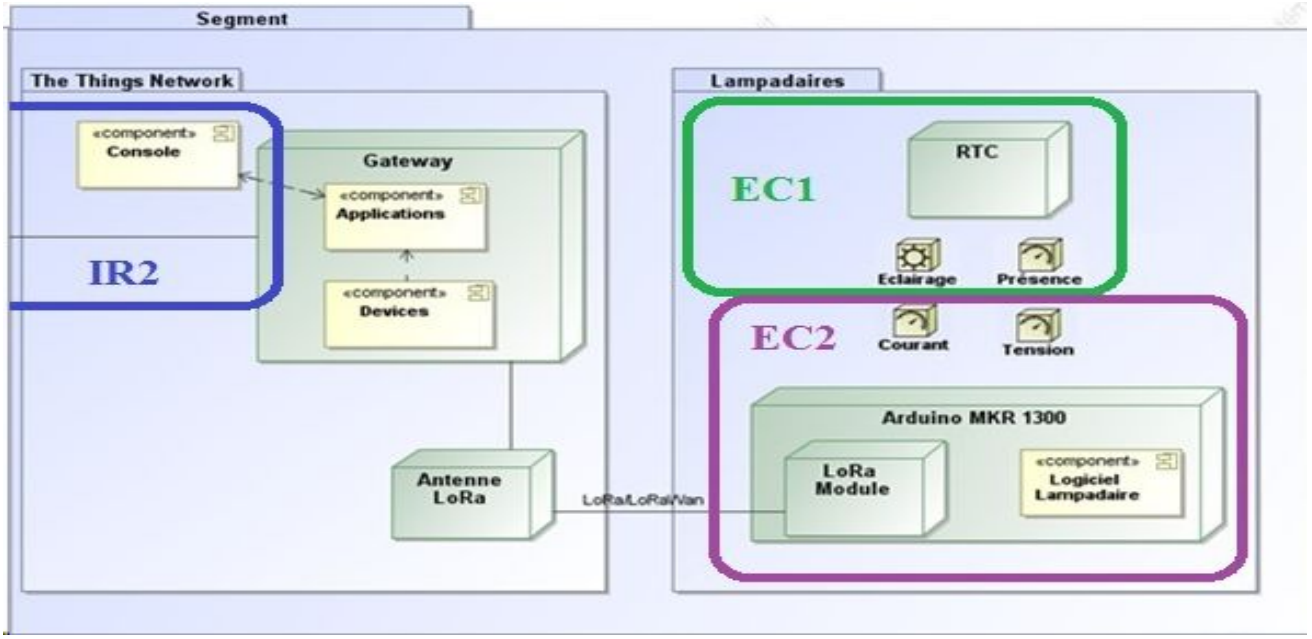
Activités EC1	**196,5 hr**	**Jeu 10/01/19**	**Mer 12/06/1**		
Installation librairies MKR 1300	30 min	Jeu 10/01/19	Jeu 10/01/19	2	EC1
instalation des librairies de l'horloge temps réel	30 min	Ven 01/03/19	Ven 01/03/19	28	EC1
Faire mon diagramme de gantt	2 hr	Mer 06/03/19	Mer 06/03/19	29	EC1
vérifier si les alim de la carte peut se faire depuis le GPIO	8 hr	Jeu 07/03/19	Ven 08/03/19	30	EC1
Valider l'horloge temps réel , apporter des modification si nécessaire	1 hr	Ven 08/03/19	Ven 08/03/19	31	EC1
Faire mon schéma de cablages sur fritzing	8 hr	Mer 13/03/19	Jeu 14/03/19	32	EC1
Réalisation	**109 hr**	**Mer 20/03/1**	**Mer 12/06/1**		
Concevoir un circuit imprimé qui réponde au cahier des charges	50 hr	Mer 20/03/19	Ven 26/04/19	33	EC1
document SysML a complétés et a adapter	8 hr	Jeu 02/05/19	Ven 03/05/19	35	EC1
Développer une application	50 hr	Jeu 09/05/19	Mer 12/06/19	36	EC1

Estimation de mon 1er gantt EC1

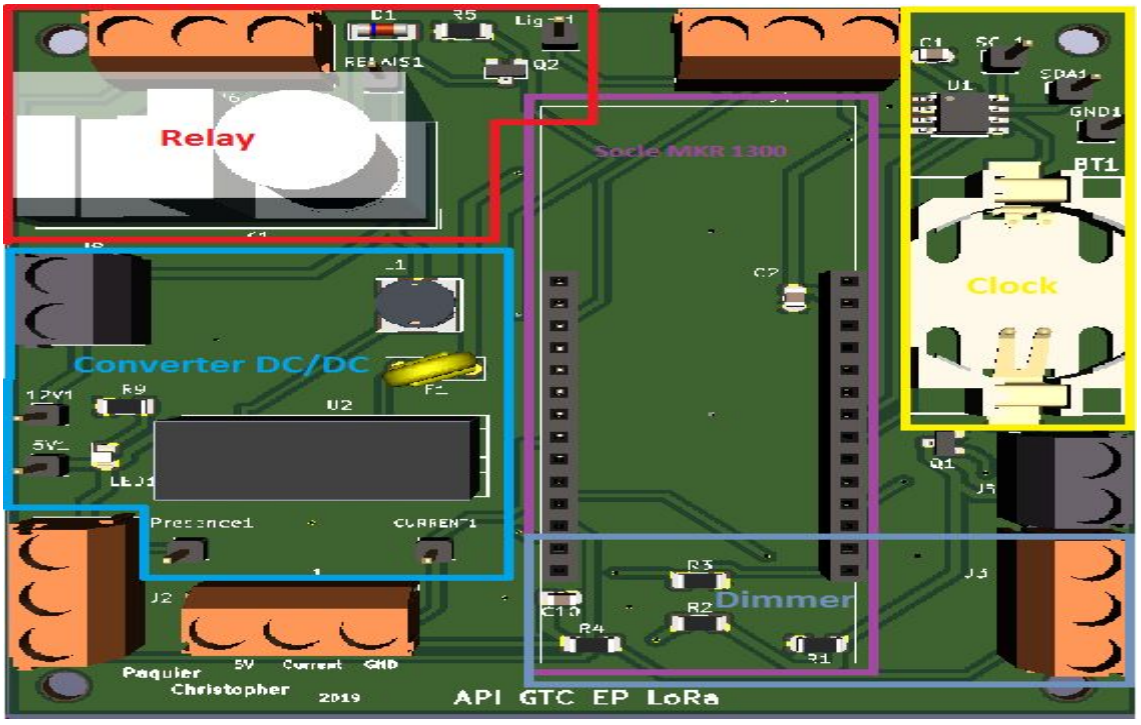
Activités EC1	**126 hr**	**Jeu 10/01/19**	**Mer 24/04/1**		
Installation librairies MKR 1300	30 min	Jeu 10/01/19	Jeu 10/01/19	2	EC1
instalation des librairies de l'horloge temps réel	30 min	Ven 01/03/19	Ven 01/03/19	24	EC1
Faire mon diagramme de gantt	2 hr	Mer 06/03/1	Mer 06/03/1	25	EC1
vérifier si les alim de la carte peut se faire depuis le GPIO	8 hr	Jeu 07/03/19	Ven 08/03/19	26	EC1
Valider l'horloge temps réel , apporter des modification si nécessaire	4 hr	Ven 08/03/19	Mer 13/03/19	27	EC1
Faire mon shémas de cablages sur fritzing	8 hr	Mer 13/03/19	Ven 15/03/19	28	EC1
Réalisation	**103 hr**	**Ven 15/03/1**	**Mer 05/06/1**		
Concevoir un circuit imprimé qui réponde au cahier des charges	65 hr	Ven 15/03/19	Jeu 09/05/19	29	EC1
document SysML a complétés et a adapter	8 hr	Jeu 09/05/19	Mer 15/05/19	31	EC1
Développer une application	30 hr	Mer 22/05/1	Ven 07/06/1	32	EC1

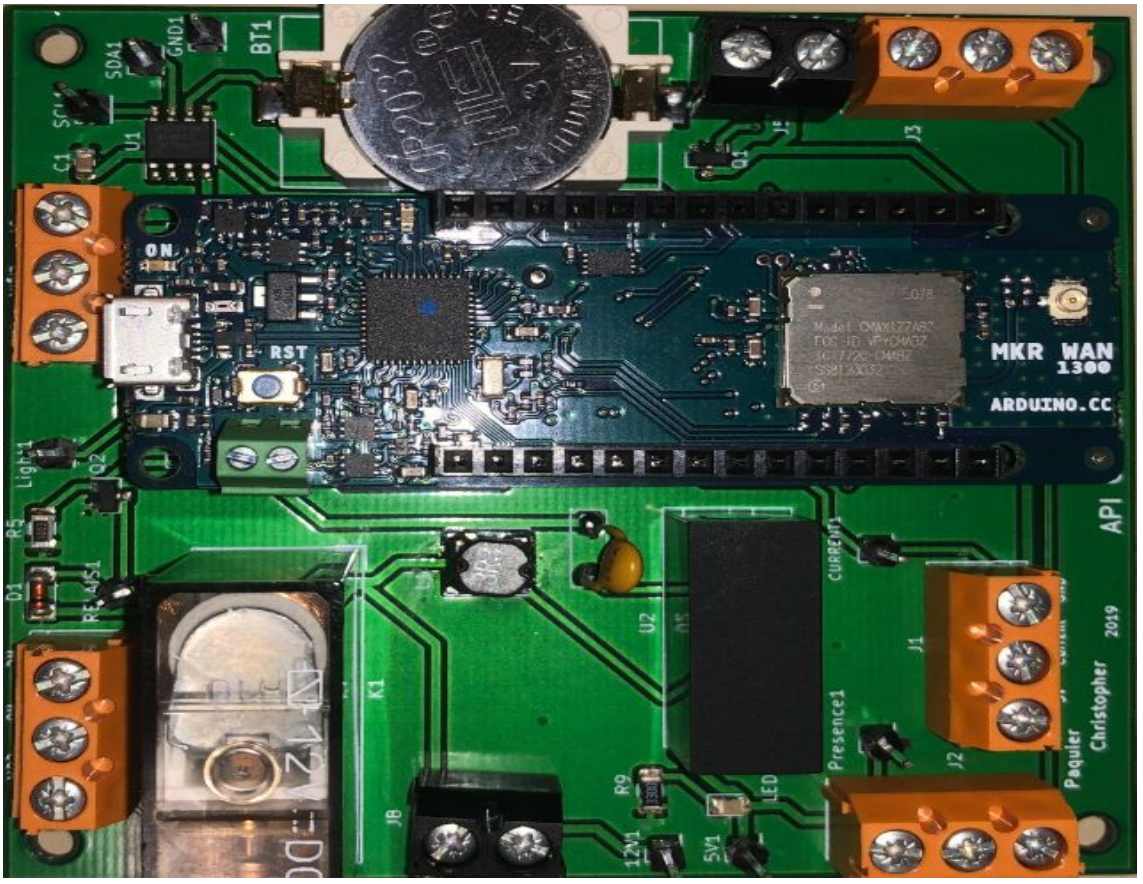
Deuxième Gantt final EC1

Présentation de mon équipe au sein de mon projet

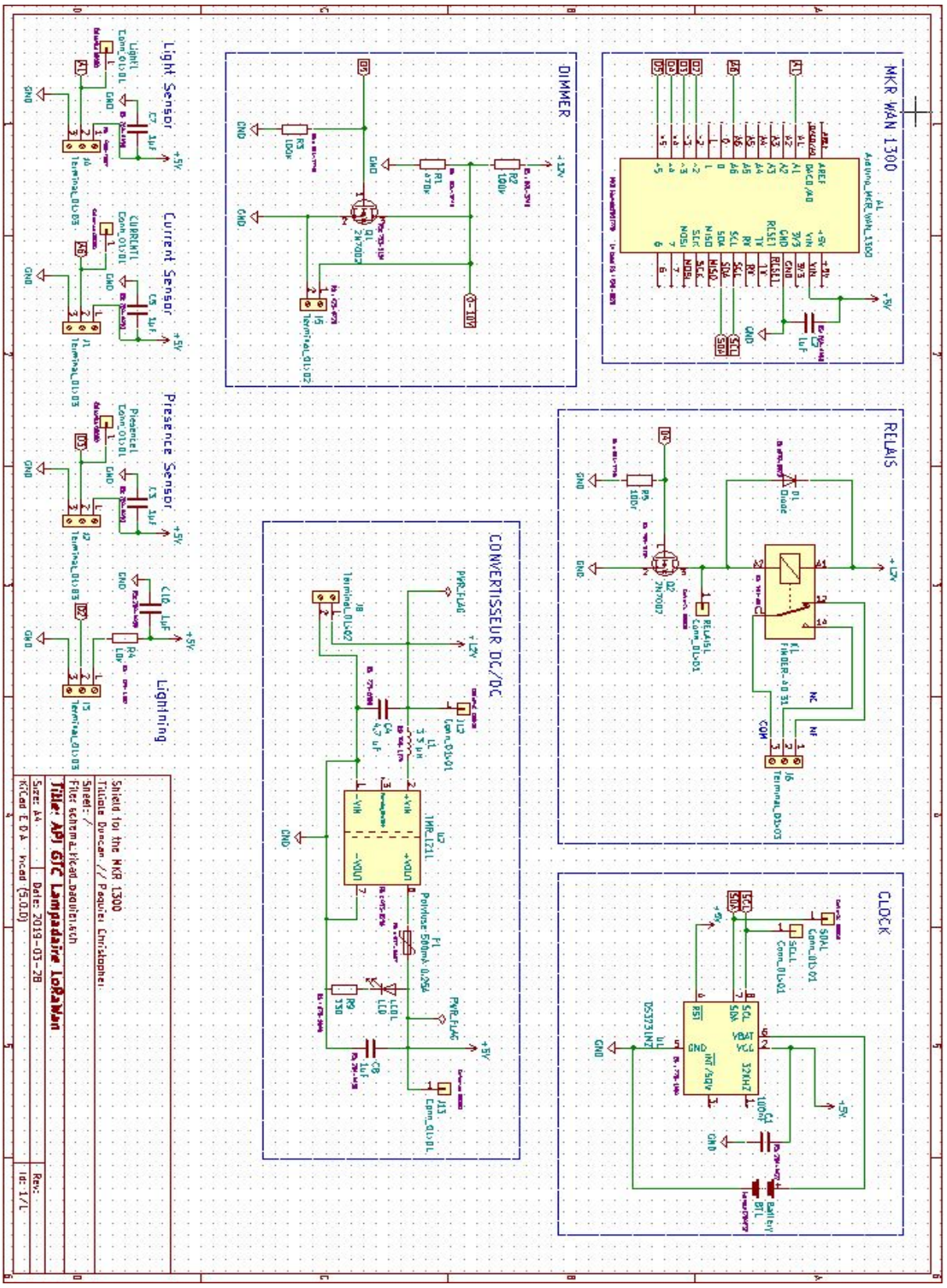


Carte Final :


Relay
Socle MKR 1300
Converter DC/DC
Clock
Dimmer
Presence1
CURRENT1
Paquier
Christopher
5V
Current
GND
2019
API GTC EP LoRa


MKR WAN 1300
ARDUINO.CC
RST
BT1
API

Saisie du schéma structurel sur kicad avec Duncan.

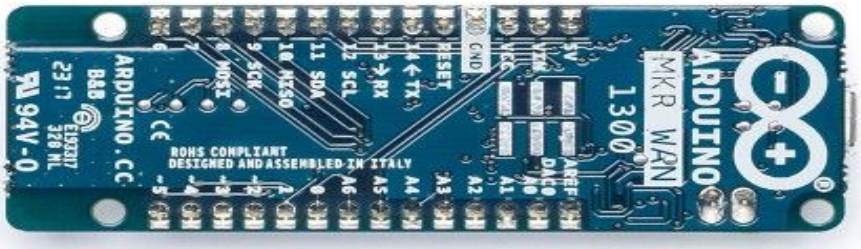


MKR 1300

Je dois utiliser le bus I2C, ce qui facilite la connexion. Je dois d'abord identifier quelles broches de l'arduino ou de cartes compatibles sont utilisées pour le bus I2C. Celles-ci seront appelées SDA (ou données) et SCL (ou horloge).

---

arduino MKR 1300 Bus I2C



Pour cette carte on va donc utiliser le bus I2C pour l'horloge.

Les broches de la carte MKR 1300 qui fournissent I2C sont les broches 12 (SCL) et 11 (SDA).

L’horloge

DS3231M ±5ppm, I²C Real-Time Clock

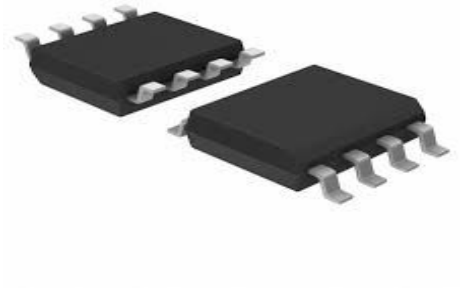
Pin Description (continued)

PIN		NAME	FUNCTION
8 SO	16 SO		
7	15	SDA	Serial-Data Input/Output. This pin is the data input/output for the I²C serial interface. This open-drain pin requires an external pullup resistor. The pullup voltage can be up to 5.5V, regardless of the voltage on VCC.
8	16	SCL	Serial-Clock Input. This pin is the clock input for the I²C serial interface and is used to synchronize data movement on the serial interface. The pullup voltage can be up to 5.5V, regardless of the voltage on VCC.

Traduction :

SDA :Données série Entrée / sortie. Cette broche est l'entrée / sortie de données pour l'interface série I2C. Ce drain ouvert ,La broche nécessite une résistance de rappel externe. La tension de rappel peut aller jusqu’à 5,5V, quelle que soit la tension sur VCC.

SCL : Entrée d'horloge série. Cette broche est l'entrée d'horloge pour l'interface série I2C et est utilisée pour synchroniser les mouvements de données sur l'interface série. La tension de rappel peut aller jusqu’à 5,5V, quelle que soit la tension sur VCC.

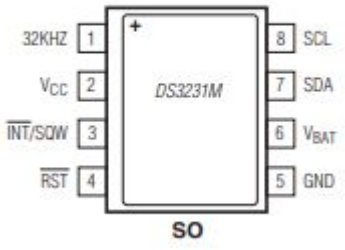


L'horloge temps réel ou RTC (Real Time Clock) DS3231M permet de fournir la date et l'heure à la carte Arduino via la liaison I2C. Elle sert à compléter le dimmer qui lui graduera la luminosité en fonction de c’est critere.

Pin Description

PIN		NAME	FUNCTION
8 SO	16 SO		
1	1	32KHZ	32.768kHz Output (50% Duty Cycle). This open-drain pin requires an external pullup resistor. When enabled with the EN32KHZ bit in the Status register (0Fh), this output operates on either power supply. This pin can be left open circuit if not used.
2	2	V_{CC}	DC Power Pin for Primary Power Supply. This pin should be decoupled using a 0.1µF to 1.0µF capacitor. Connect to ground if not used.
3	3	$\overline{INT}$/SQW	Active-Low Interrupt or 1Hz Square-Wave Output. This open-drain pin requires an external pullup resistor connected to a supply at 5.5V or less. It can be left open if not used. This multifunction pin is determined by the state of the INTCN bit in the Control register (0Eh). When INTCN is set to logic 0, this pin outputs a 1Hz square wave. When INTCN is set to logic 1, a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled.
4	4	$\overline{RST}$	Active-Low Reset. This pin is an open-drain input/output. It indicates the status of V_{CC} relative to the V_{PF} specification. As V_{CC} falls below V_{PF}, the $\overline{RST}$ pin is driven low. When V_{CC} exceeds V_{PF}, for t_{RST}, the $\overline{RST}$ pin is pulled high by the internal pullup resistor. The active-low, open-drain output is combined with a debounced pushbutton input function. This pin can be activated by a pushbutton reset request. It has an internal 50kΩ (R_{PU}) nominal value pullup resistor to V_{CC}. No external pullup resistors should be connected. If the oscillator is disabled, t_{REC} is bypassed and $\overline{RST}$ immediately goes high.
—	5–12	N.C.	No Connection. These pins must be connected to ground.
5	13	GND	Ground
6	14	V_{BAT}	Backup Power-Supply Input. When using the device with the V_{BAT} input as the primary power source, this pin should be decoupled using a 0.1µF to 1.0µF low-leakage capacitor. When using the device with the V_{BAT} input as the backup power source, the capacitor is not required. If V_{BAT} is not used, connect to ground. The device is UL recognized to ensure against reverse charging when used with a primary lithium battery. Go to www.maximintegrated.com/qa/info/ul for more information.

TOP VIEW



PIN SCL (8) = câble blanc du prototype de 2018
PIN SDA (7) = câble noir du prototype.
PIN Mase (5) = câble couleur rouge.
PIN VCC (2) = câble jaune

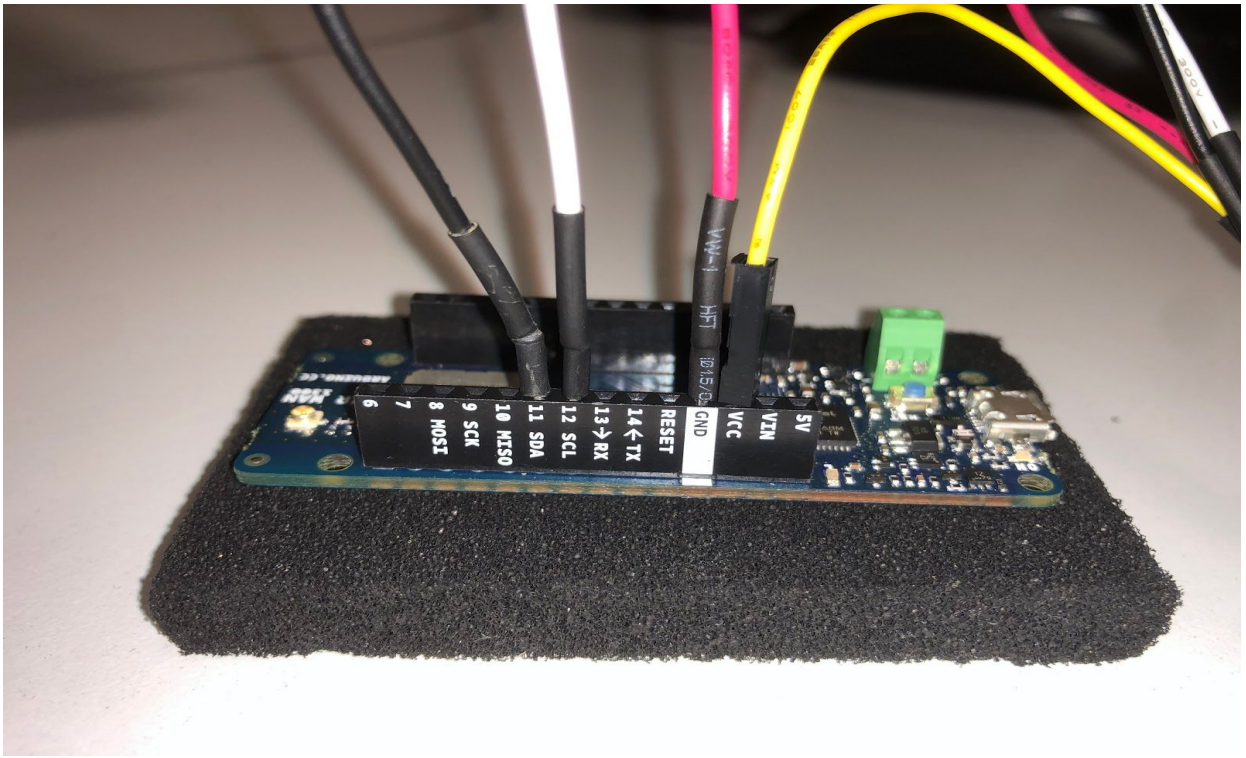
J'ai ensuite vérifier l'alimentation de l'horloge pour ne pas abîmer ou détruire le composant.
J'ai trouvé cela dans la documentation de l'horloge DS3231.

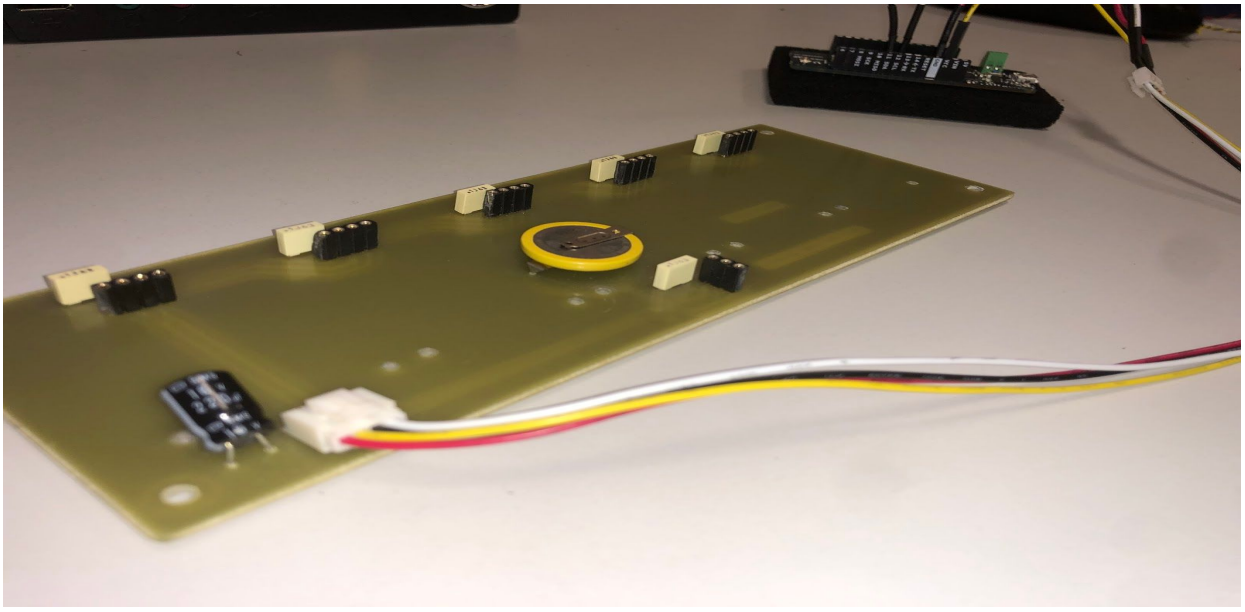
On peut donc l'alimenter entre 2.3V à 5.5V.

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		2.3	3.3	5.5	V
	V_{BAT}		2.3	3.0	5.5	
Logic 1	V_{IH}		0.7 x V_{CC}		V_{CC} + 0.3	V
Logic 0	V_{IL}		-0.3		0.3 x V_{CC}	V

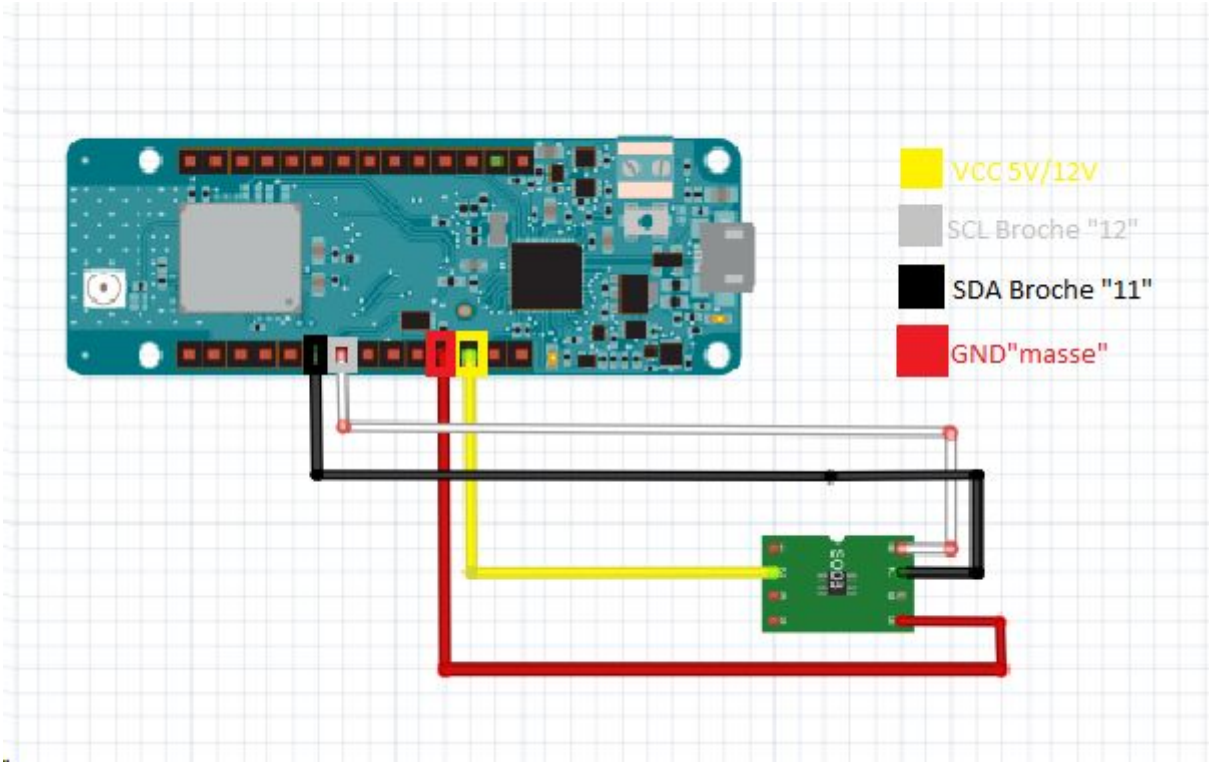
Grâce à toute ces informations sur l'horloge temps réel, on va pouvoir tester la carte d'un ancien projet fourni. Pour ensuite pouvoir l'utiliser dans mon projet.

Voici mon câblage :





Voici mon cablage sur fritzing:



Après ce travail effectué il faut aussi le programme et la librairie arduino pour pouvoir fournir des valeurs, tout ce que l'horloge peut nous apporter durant notre projet.

J'ai donc ajouté la librairie de l'horloge "DS3231" :
J'ai téléversé mon programme , il y a eu une erreur :

Je suis donc allé dans la librairie changer un paramètre qui me mène à un conflit car il etait defini 2 fois pour deux choses différentes .

- une fois dans la librairie DS3231.cpp j'ai dû changer le paramètre "PM" je l'ai remplacer par "pm" .
- puis aller sur le fichier DS3231.h de la librairie et remplacer le paramètre "PM" par "pm".
- puis j'ai enregistré.

Voici comment j'ai procédé :

Avant :

DS3231.cpp

```
    Wire.endTransmission();

    Wire.requestFrom(CLOCK_ADDRESS, 1);
    return bcdToDec(Wire.read());
}

byte DS3231::getMinute() {
    Wire.beginTransmission(CLOCK_ADDRESS);
    Wire.write(0x01);
    Wire.endTransmission();

    Wire.requestFrom(CLOCK_ADDRESS, 1);
    return bcdToDec(Wire.read());
}

byte DS3231::getHour(bool& h12, bool& PM) {
    byte temp_buffer;
    byte hour;
    Wire.beginTransmission(CLOCK_ADDRESS);
    Wire.write(0x02);
    Wire.endTransmission();

    Wire.requestFrom(CLOCK_ADDRESS, 1);
    temp_buffer = Wire.read();
    h12 = temp_buffer & 0b01000000;
    if (h12) {
        PM = temp_buffer & 0b00100000;
        hour = bcdToDec(temp_buffer & 0b00011111);
    } else {
        hour = bcdToDec(temp_buffer & 0b00111111);
    }
    return hour;
}
```

DS3231.h | DS3231.cpp

```
69      byte getSecond();
70      byte getMinute();
71      byte getHour(bool& h12, bool& PM);
72          // In addition to returning the hour register, this function
73          // returns the values of the 12/24-hour flag and the AM/PM flag.
74      byte getDoW();
75      byte getDate();
76      byte getMonth(bool& Century);
77          // Also sets the flag indicating century roll-over.
78      byte getYear();
79          // Last 2 digits only
```

Une fois le changement effectué :

DS3231.h | DS3231.cpp

```
161         Wire.endTransmission();
162
163         Wire.requestFrom(CLOCK_ADDRESS, 1);
164         return bcdToDec(Wire.read());
165     }
166
167     byte DS3231::getMinute() {
168         Wire.beginTransmission(CLOCK_ADDRESS);
169         Wire.write(0x01);
170         Wire.endTransmission();
171
172         Wire.requestFrom(CLOCK_ADDRESS, 1);
173         return bcdToDec(Wire.read());
174     }
175
176     byte DS3231::getHour(bool& h12, bool& pm) {
177         byte temp_buffer;
178         byte hour;
179         Wire.beginTransmission(CLOCK_ADDRESS);
180         Wire.write(0x02);
181         Wire.endTransmission();
182
183         Wire.requestFrom(CLOCK_ADDRESS, 1);
184         temp_buffer = Wire.read();
185         h12 = temp_buffer & 0b01000000;
186         if (h12) {
187             pm = temp_buffer & 0b00100000;
188             hour = bcdToDec(temp_buffer & 0b00011111);
189         } else {
190             hour = bcdToDec(temp_buffer & 0b00111111);
191         }
192         return hour;
```

DS3231.h | DS3231.cpp

```
69          byte getSecond();
70          byte getMinute();
71          byte getHour(bool& h12, bool& pm);
72              // In addition to returning the hour register, this function
73              // returns the values of the 12/24-hour flag and the AM/PM flag.
74          byte getDoW();
75          byte getDate();
```

J’ai ensuite téléversé puis compilé , ça a fonctionné .
J’ai aussi renseigné quelques données pour l’horloge :

- l'année
- la date
- l’heure

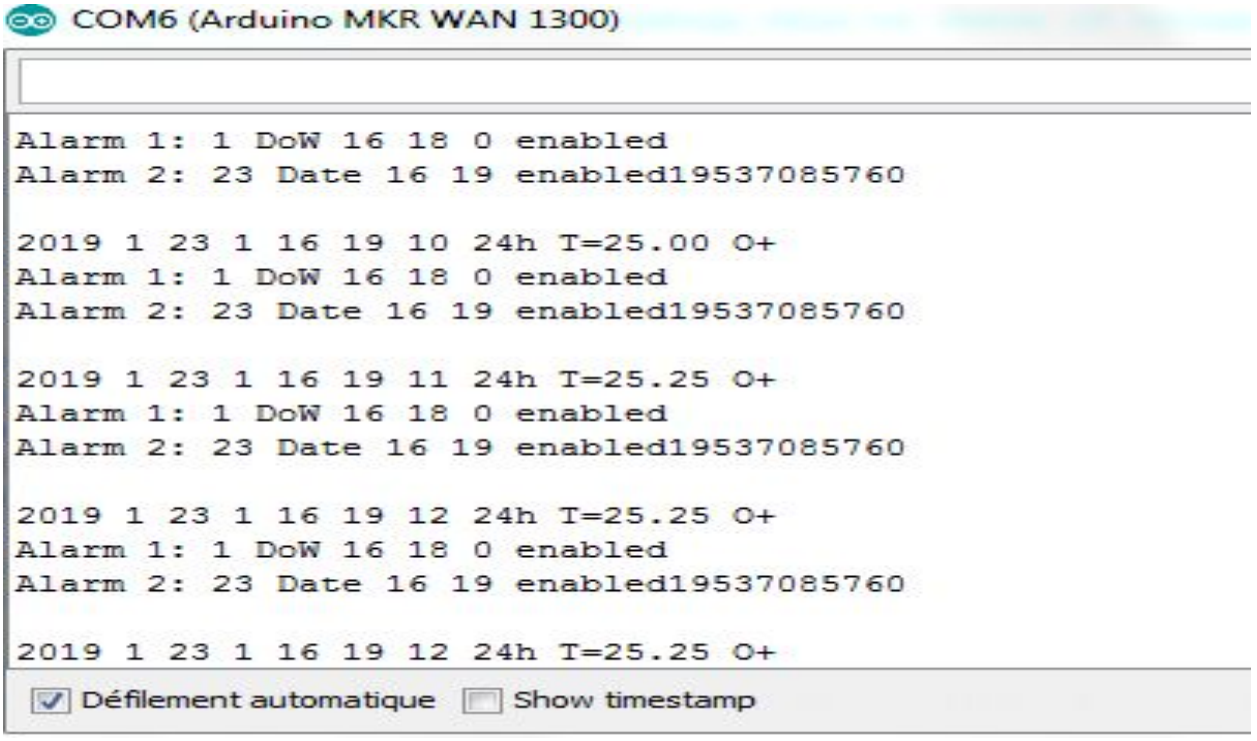


Diagramme de séquence : ”Consommation d’un segment“

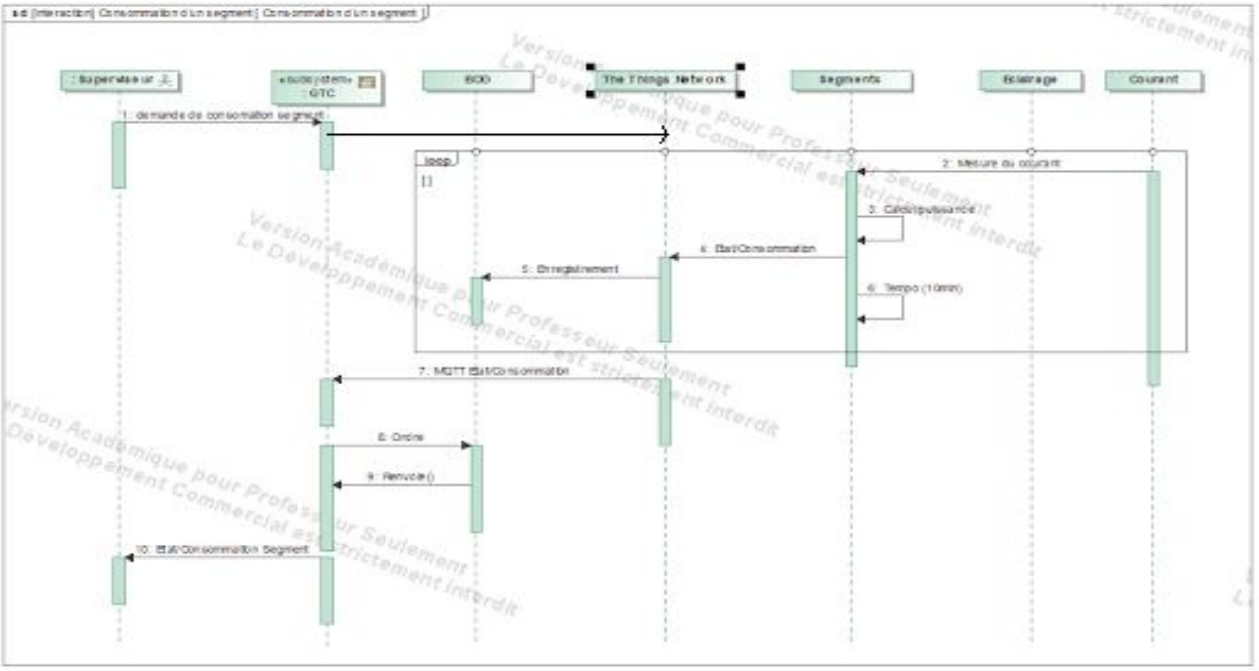
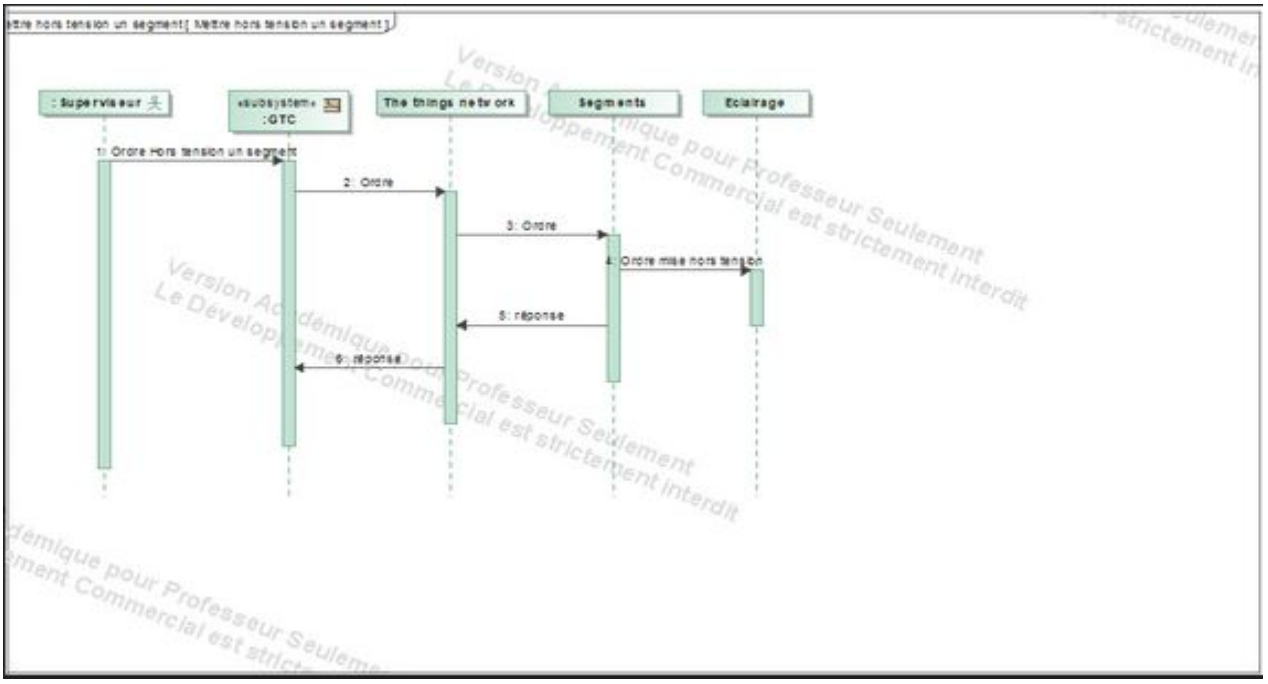
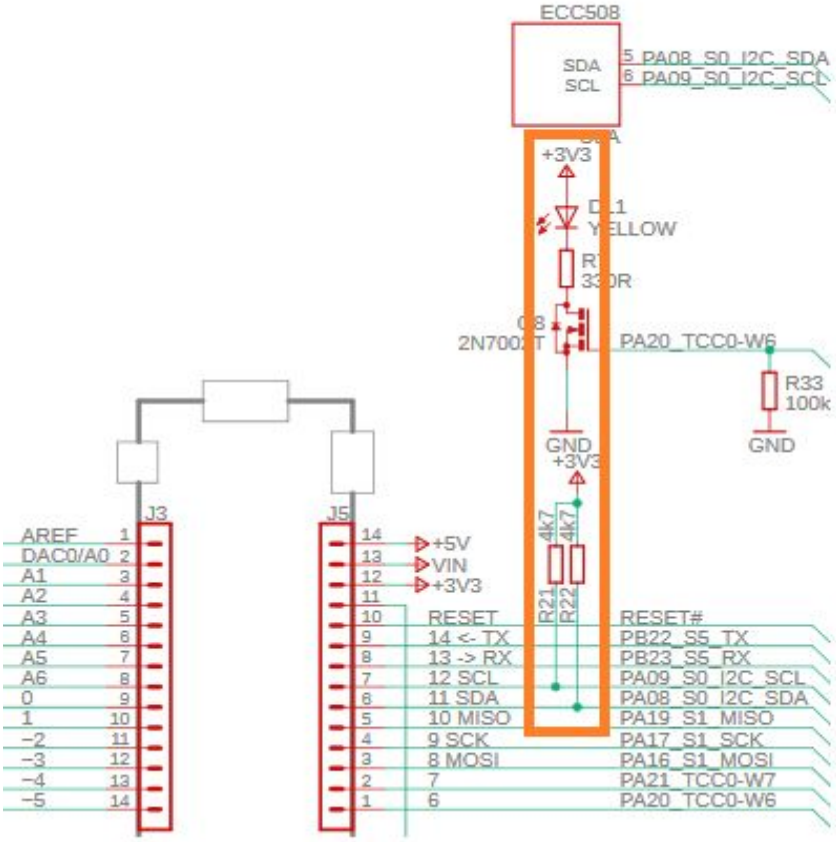


Diagramme de séquence : “Mettre hors tension un segment”



Schémas structurel arduino MKR 1300 pour les entrée/sortie.



On peut voir que sur le bus I2C qu'il n'a pas besoin de résistance de pull up, car il y en a déjà intégré dans la carte Arduino MKR

Avertissement: Contrairement à la plupart des cartes Arduino et Genuino, le MKR WAN 1300 fonctionne à 3,3V. La tension maximale tolérée par les broches d'E / S est de 3,3V. L'application de tensions supérieures à 3,3 V à n'importe quelle broche d'E / S peut endommager la carte. Bien que la sortie sur des appareils numériques 5V soit possible, la communication bidirectionnelle avec des appareils 5V nécessite un décalage de niveau approprié.

Communication et transmission LoRa

The Things Network est un réseau LoRaWAN communautaire et open source. L'Arduino MKR 1300 WAN est équipé d'une puce LoRa. La technologie LoRa est un standard pour l'Internet des Objets.

Étape 5 - Modifier le croquis

Afin d'envoyer les données vers "The Things Network" (ou un autre réseau LoRa) il faut inclure dans le croquis :

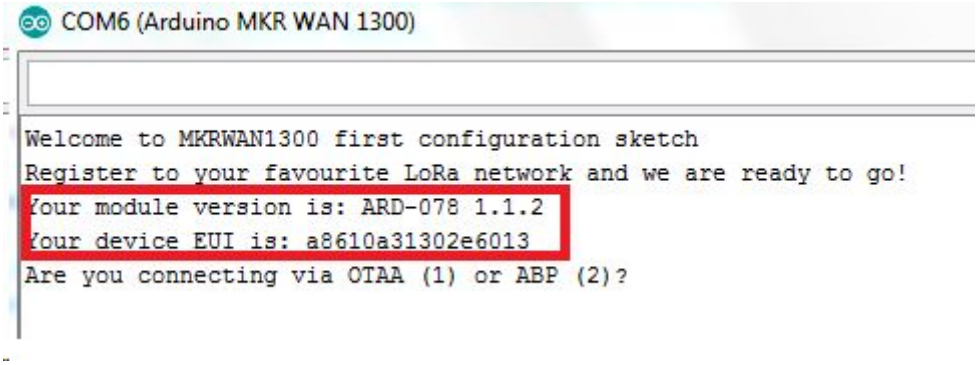
- L'APP EUI
- L'APP KEY

Concernant "The Things Network" vous trouverez ces informations dans la zone "Applications" de votre console d'administration.

Pour renseigner ces informations dans le croquis vous devez créer un fichier **"arduino_secrets.h"** dans le dossier "Si7021-LoRa". Ce fichier contiendra votre "APP EUI" et votre "APP KEY" sous la forme :

```
#define SECRET_APP_EUI "Votre APP EUI"
#define SECRET_APP_KEY "Votre APP KEY"
```

Je vais donc passé à la communication des données avec mes partenaires IR : pour cela je vais lancé un programme arduino **“LoRaSendAndReceive”**

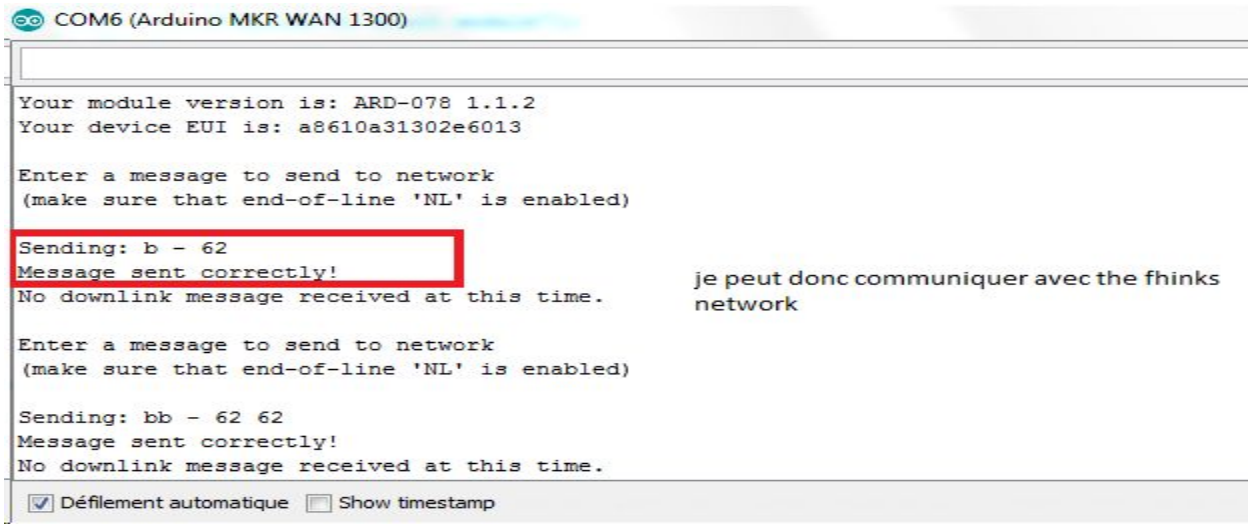


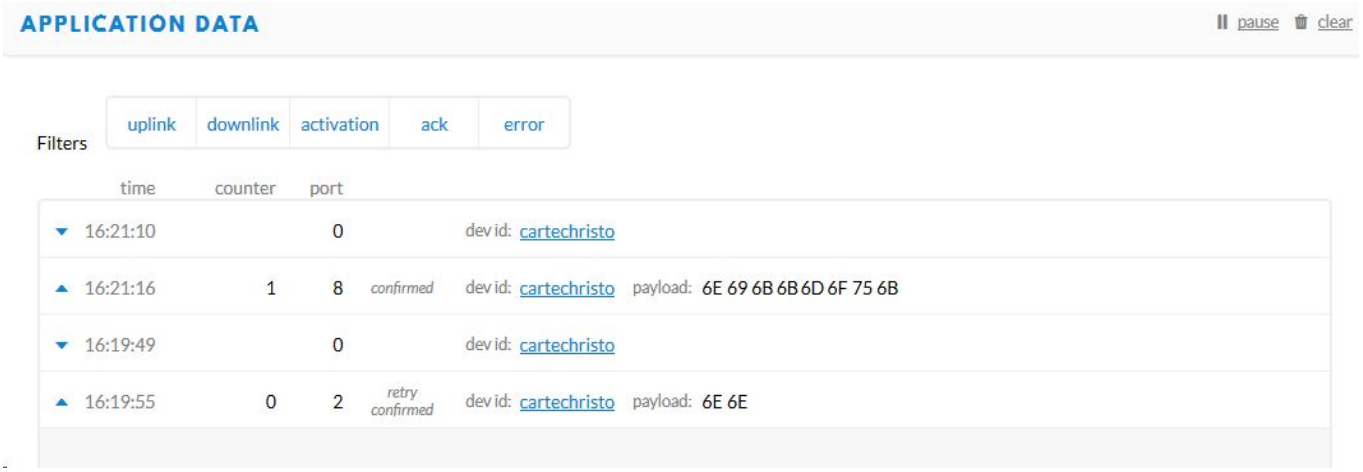
➔ Je vais communiquer l’information à Aymerick Serda (IR) : cette étape est obligatoire pour pouvoir communiquer avec lui. Il faut qu'il m’enregistre dans the things network.

Your device EUI is : a8610a31302e6013

- Sélectionné 1 OTAA
- Application EUI commun : 70B3D57ED00169C0
- App Key : CAD972BD9DAC2261DAE413227B585487

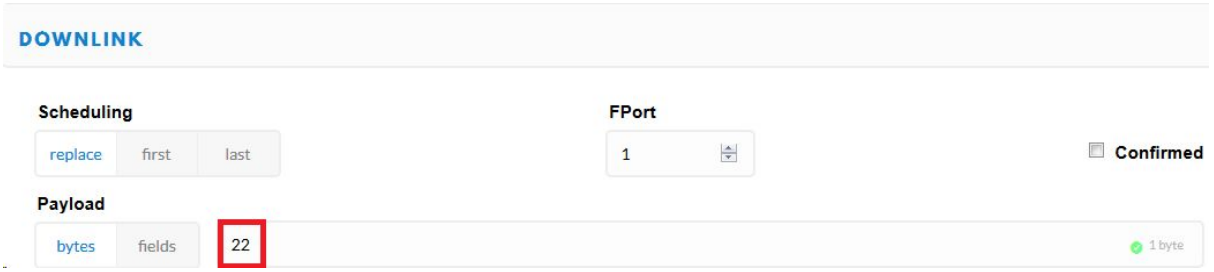
Expérimentations :



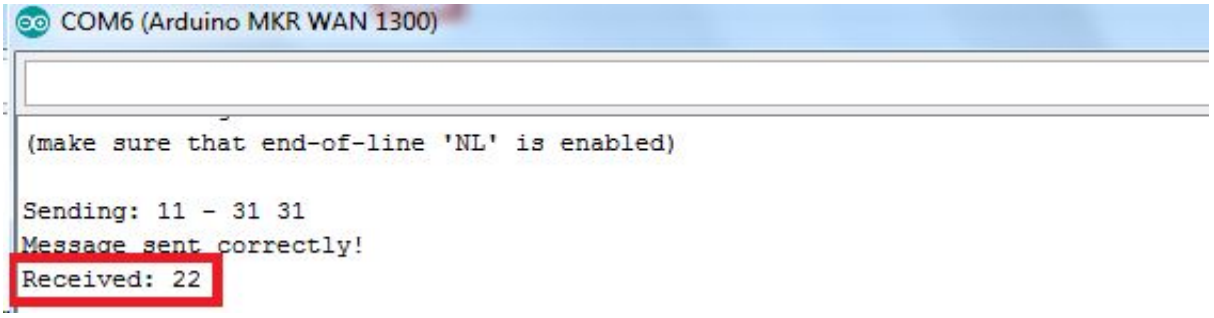


Il va falloir communiquer de the things network vers la carte MKR 1300, pour ensuite mettre le protocol de communication en marche.

Voici un envoi de données de things network vers L'arduino MKR :



Quand j'envoie des données de la MKR, je reçois et je peux envoyer en retour, "classe A".



J'ai envoyé "11" ce qui représente "31 31" , cela est normal vu qu'on communique en code ascii .
On peut voir qu'il me renvoie la valeur "22" que j'ai rentré auparavant sur the things network.

Voici notre protocol de communication : L'ordre du dimmer et variable : Ascii

TTN -> MKR		
Commande Ascii	**ORDRE demandé**	**Explications :**
30 / 31 30 30	DIM	gradateur, de 0 à 100%
4F 4E	ON	allumer lampadaire
4F 46 46	OFF	éteindre lampadaire
41 55 54 4F	AUTO	passe en mode auto
67 [2] [2] [2] [2]	CONF	
MKR -> TTN		
Code sur [XX] octects	**Envoie automatique**	**Explications :**
XX XX XX XX	Courant	Envoie du courant à chaque trame
XX	Etat	Envoie de l'état à chaque trame

Concernant la plage d'utilisation de la tension VIN sur la MKR WAN 1300, je vais me renseigné pour savoir réellement la plage d'alimentation du Pin "VIN".

Voici ce que j'ai trouvé :

Arduino MKR GSM 1400

L'arduino MKR GSM 1400 a probablement été créé pour augmenter le nombre de possibilités avec lesquelles vous pouvez réaliser vos projets IoT.

Dans les cas où d'autres systèmes de communication tels que LoRa, WiFi, Sigfox ne sont pas disponibles, il est nécessaire de disposer d'un appareil capable de communiquer via le réseau GSM: MKR GSM 1400

Arduino MKR WAN 1300

Comme vous avez pu le constater, il présente de nombreuses caractéristiques communes avec la sœur cadette MKR WAN 1300, mais présente quelques différences:

- Carte d'alimentation (USB / VIN) 5V / 5-12V
- Batterie prise en charge 3.7V LiPo
- Tension de fonctionnement du circuit 3.3V

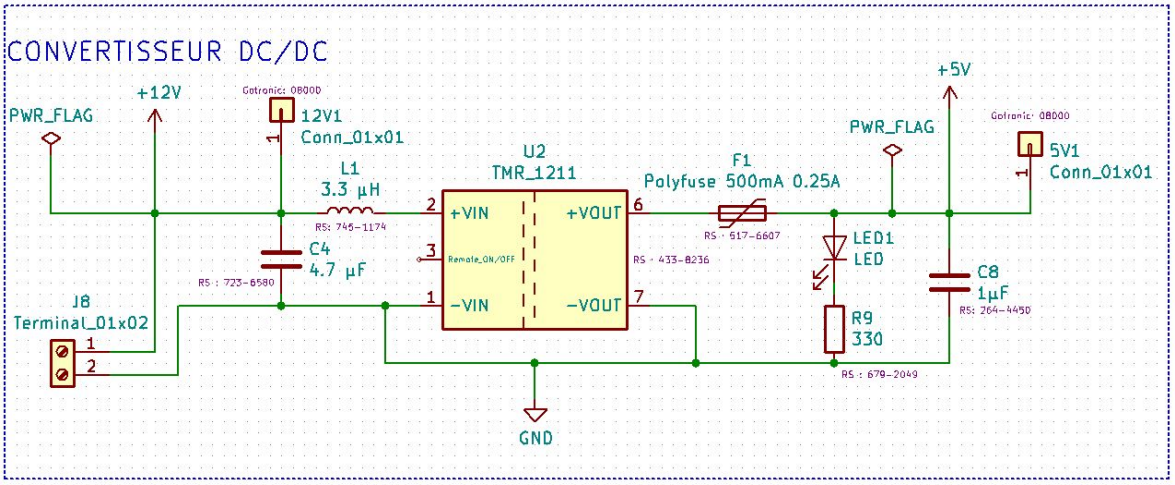
les tensions de fonctionnement restent de 3,3 V, mais cette version de la famille MKR peut l'alimenter de 5 à 12 V à l'aide de la broche Vin et à seulement 5 V de l'USB.

Source: store-cdn.arduino.cc

On va donc pouvoir de la carte shield qui elle est alimentée en 12V alimenter la carte MKR 1300 directement par la broche Vin . Pour la suite de notre projet c'est une bonne nouvelle, cela va nous permettre d'économiser un convertisseur 12/5V.

Après avoir réfléchie avec Mr Hortolan nous en avons déduit que finalement ce n'était pas possible, il faudra donc un convertisseur.

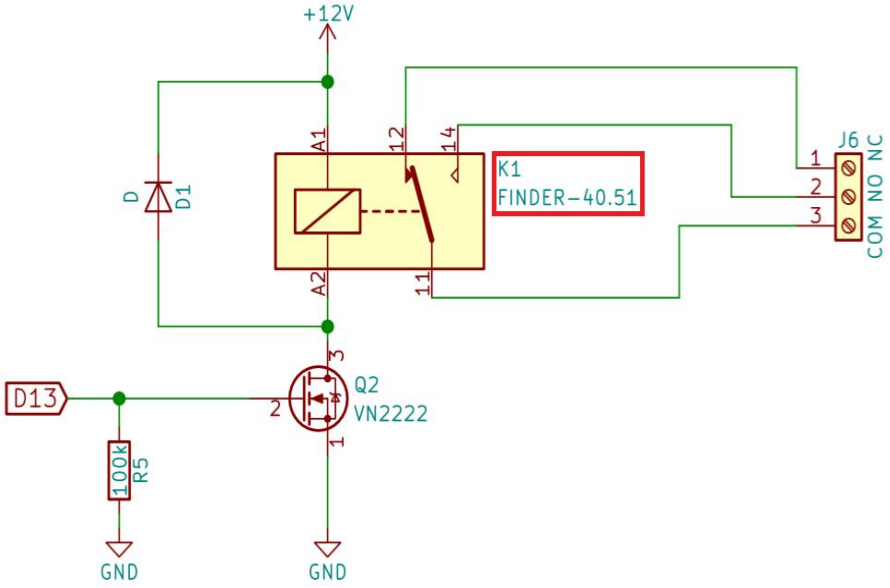
Schéma structurel convertisseur DC/DC 12V / 5V a régulateur de découpage.



Ce convertisseur va nous permettre de convertir le 12V qui va alimenter la carte fille , en 5V qui alimente la MKR 1300.

Je vais maintenant faire le relais, grâce à lui on pourra câbler une prise annex , il joue le rôle d'interrupteur. On pourra câblé une guirlande lumineuse par exemple et la contrôler à distance.

schémas structurel du relais K1 1RT :

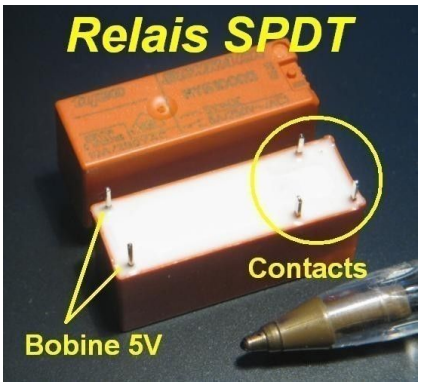


Sur ce schéma j'ai utilisé un transistor VN2222 si le transistor est saturé, le 0V sera envoyé au relais FINDER 40.31. Inversement, si le transistor est bloqué, le relais recevra du 12V.

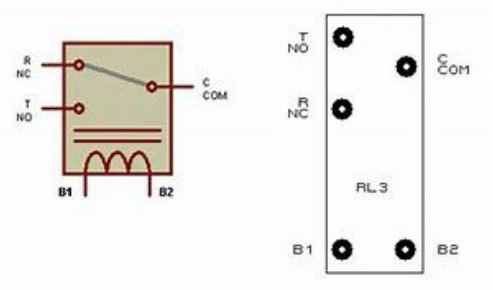
Principe du relai K1 1RT :

Relais SPDT : Single Pole Double Throw

Ce relais possède 5 broches au total
- 3 broches pour les contacts
- 2 broches pour la bobine



Un simple relais se compose d'une bobine et d'un ou plusieurs contacts. Si une tension est appliquée à la bobine, cela provoque la création d'un champ magnétique. Par le champ magnétique de la bobine, une plaquette en fer est attirée ce qui a pour effet de réaliser le contact. Si on supprime la tension, le champ magnétique disparaît et la plaquette en fer revient dans sa position de repos par l'action d'un ressort, ce qui désactive le contact.



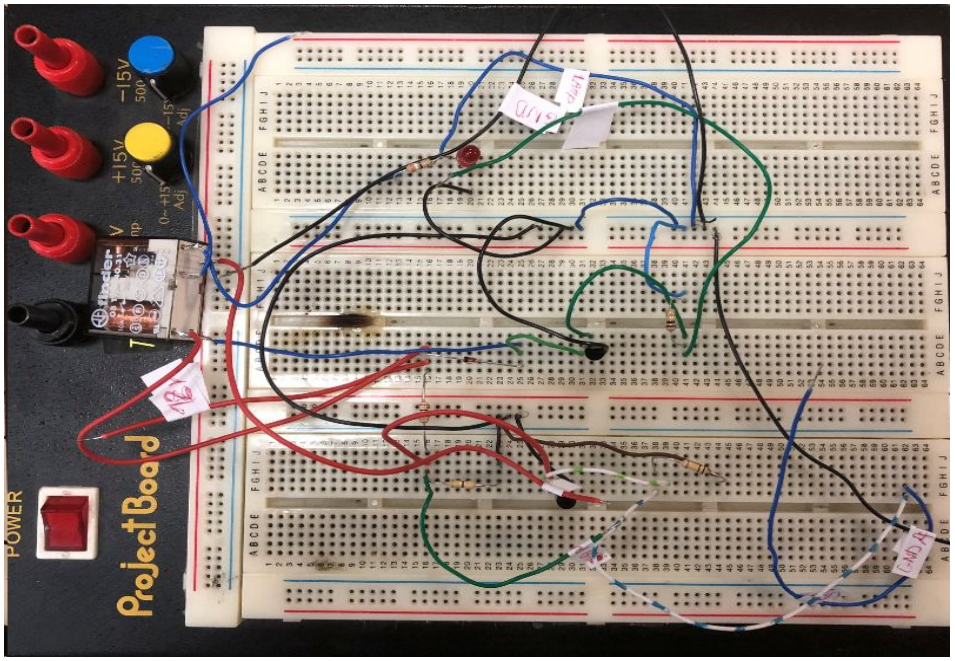
Le relais SPDT possède un seul contact mais avec une borne commune, un contact normalement ouvert (quand il n'y a pas de tension sur la bobine) et un contact normalement fermé (quand il n'y a pas de tension sur la bobine). Quand on applique une tension sur la bobine, on entend "clic" : la borne commune va se connecter sur le contact normalement ouvert (NO = normally open) et le contact normalement fermé (NC = normally closed) s'ouvre. Dès qu'on coupe la tension aux bornes de la bobine, on entend "clic" et le relais revient à son état de repos. On peut ainsi basculer d'un circuit à l'autre (allumer soit l'ampoule rouge soit l'ampoule verte par exemple).

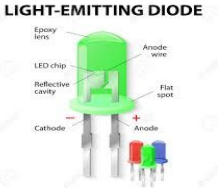
Programme arduino :

prog_Relaie §

```
#define ON HIGH
#define OFF LOW

#define RELAY_PIN 4

void setup() {
  Serial.begin(9600);
  pinMode(RELAY_PIN, OUTPUT);  // RELAY_PIN est une sortie

}

void loop() {

  if (Serial.available() > 0) // Boucle if
  {
    //End Of Frame is \n
    String message = Serial.readStringUntil('\n'); // initialisation de la variable message
                                                   //  qui sert à lire dans le moniteur série
    if (message == "ON") //
    {
      digitalWrite (RELAY_PIN, ON); // RELAY_PIN = HIGH
      Serial.println("Message envoyé : ON");
    }


    if (message == "OFF")
    {
      digitalWrite (RELAY_PIN, OFF); // REALY_PIN = LOW
      Serial.println("Message envoyé : OFF");
    }

  }
}
```

Schéma en break out :





- Relais K1
  Finder 40-31
- transistor MOs Canal “N” VN2222
- Led 5mm + résistance 390K
- Diode D1
- Résistances 100K

Dans notre montage en break out j'ai besoin d'un Transistor MOS canal "N".

J'ai trouvé la documentation du transistor pour en savoir plus sur les caractéristiques et les broches du composant.

Référence : Q2 VN2222

VN2222 Series
N-Channel Enhancement-Mode MOS Transistors

calogic CORPORATION

VN2222 Series

FEATURES

- Low $r_{DS(on)}$ <7.5Ω

APPLICATIONS

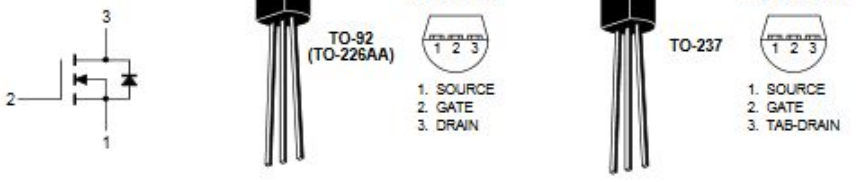
- Switching
- Amplification

ORDERING INFORMATION

Part	Package	Temperature Range
VN2222LL	Plastic TO-92	-55°C to +150°C
VN2222LM	Plastic TO-237	-55°C to +150°C

For sorted chips in carriers see 2N7000

PIN CONNECTIONS



CD5

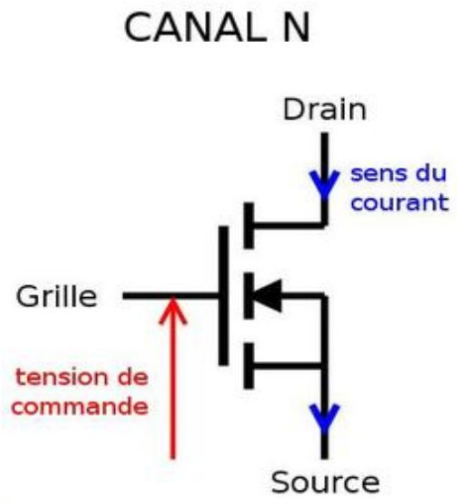
ABSOLUTE MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

SYMBOL	PARAMETER/TEST CONDITIONS		LIMITS VN2222LL	LIMITS VN2222LM	UNITS
V_{DS}	Drain-Source Voltage		60	60	V
V_{GS}	Gate-Source Voltage		±30	±30	
I_D	Continuous Drain Current	T_A = 25°C	0.23	0.26	A
		T_A = 100°C	0.14	0.16	
I_{DM}	Pulsed Drain Current[1]		1	1	
P_D	Power Dissipation	T_A = 25°C	0.8	1	W
		T_A = 100°C	0.32	0.4	
T_J, T_{stg}	Operating Junction & Storage Temperature Range		-55 to 150		°C
T_L	Lead Temperature (1/16" from case for 10 sec.)		300		

THERMAL RESISTANCE RATINGS

SYMBOL	THERMAL RESISTANCE	LIMITS VN2222LL	LIMITS VN2222LM	UNITS
R_{thJA}	Junction-to-Ambient	156	125	K/W

[1]Pulse width limited by maximum junction temperature.



Avantages :

-Temps de commutation rapides.
-Fonctionnement en haute fréquence.
-Facile à commander.
-Grande disponibilité sur le marché.

Cellule MOS Canal N : Principe 1/ Canal & Courant de conduction La polarisation positive de la grille crée un "effet de champ" qui provoque une accumulation de charges négatives(électrons)sous cette électrode.Cela va former un canal "inversé" de type N dans la zone P.Les électrons concentrés sous la grille vont assurer la conduction entre la source et le drain.(La source doit être connectée à la zone P pour en fixer le potentiel.)

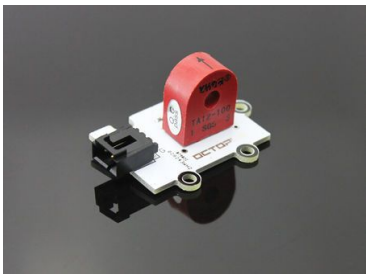
SPECIFICATIONS[a]		LIMITS				
SYMBOL	**PARAMETER**	**TYP[b]**	**MIN**	**MAX**	**UNIT**	**TEST CONDITIONS**
STATIC						
$V_{(BR)DSS}$	Drain-Source Breakdown Voltage	70	60		V	I_D = 100µA, V_{GS} = 0V
$V_{GS(th)}$	Gate-Threshold Voltage	2.3	0.6	2.5		V_{DS} = V_{GS}, I_D = 1mA
I_{GSS}	Gate-Body Leakage			±100	nA	V_{GS} = ±20V, V_{DS} = 0V
I_{DSS}	Zero Gate Voltage Drain Current			10	µA	V_{DS} = 48V, V_{GS} = 0V
				500		T_J = 125°C
$I_{D(ON)}$	On-State Drain Current[c]	1000	750		mA	V_{DS} = 10V, V_{GS} = 10V
$r_{DS(ON)}$	Drain-Source On-Resistance[c]	5		7.5	Ω	V_{GS} = 5V, I_D = 0.2A
		2.5		7.5		V_{GS} = 10V, I_D = 0.5A
		4.4		13.5		T_J = 125°C
g_{FS}	Forward Transconductance[c]	230	100		mS	V_{DS} = 10V, I_D = 0.5A
g_{OS}	Common Source Output Conductance[c]	1200			µS	V_{DS} = 10V, I_D = 0.2A
DYNAMIC						
C_{iss}	Input Capacitance	16		60	pF	V_{DS} = 25V, V_{GS} = 0V, f = 1MHz
C_{oss}	Output Capacitance	11		25		
C_{rss}	Reverse Transfer Capacitance	2		5		
SWITCHING						
t_{ON}	Turn-On Time	7		10	ns	V_{DD} = 15V, R_L = 23Ω, I_D = 0.6A V_{GEN} = 10V, R_G = 25Ω
t_{OFF}	Turn-Off Time	7		10		(Switching time is essentially independent of operating temperature)

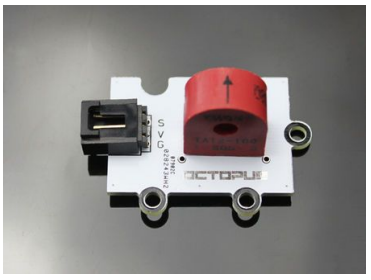
Notes:

a. T_A = 25°C unless otherwise noted.
b. For design aid only, not subject to production testing.
c. Pulse test; PW = ≤300µS, duty cycle ≤2%.

Capteur de courant

Dans notre projet on a besoin d'un capteur de courant qui nous permet de mesurer le courant pour savoir la consommation du lampadaire.
Pour ma part ce n'est pas moi qui m'en suis occupé, car ce n'était pas dans mon cahier des charges .
Je me suis quand même un peut pencher sur le sujet pour pas être dans l'inconnu.

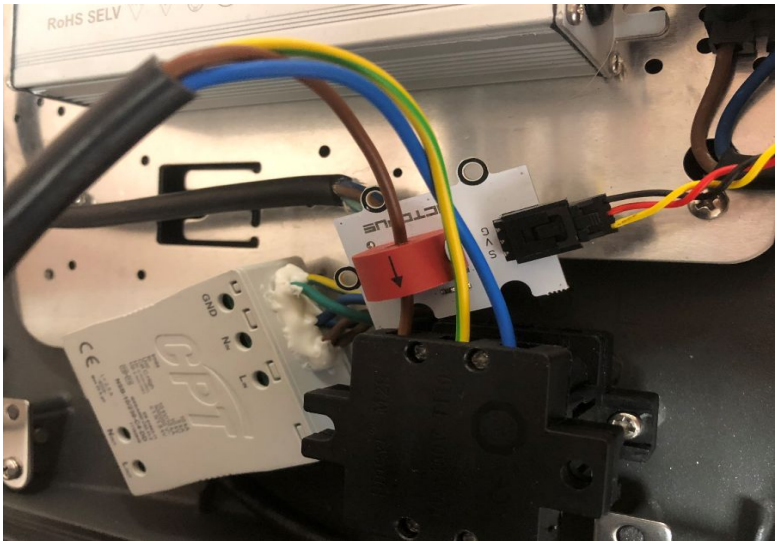




Branchement du capteur côté carte :

GND	VCC	Signal
GND	VCC	Signal

Branchement capteur côté lampadaire "+" de l'alimentation

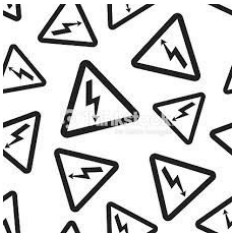


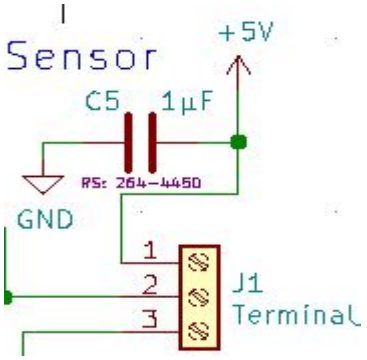
Model	Ratio	Input Currenton	Output Current	Sample Resistance	Sampling Voltage
TA12-100	1000:1	0~5A	0~5mA	200Ω	1V

Ce module capteur de courant permet de mesurer un courant alternatif et est basé sur le capteur TA12-100. Un seul des deux conducteurs doit passer dans le capteur, il délivre un signal analogique en fonction de l'intensité mesurée.

Pour le schémas structurel, ont a dû rajouter ceci :

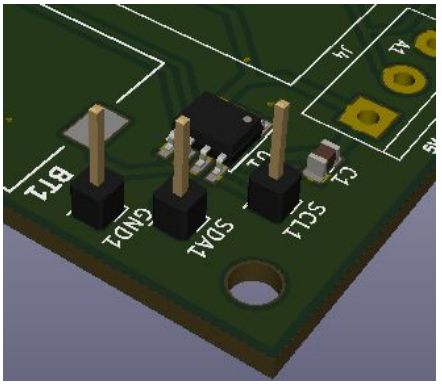
On a dû rajouter des condensateurs, ils servent à stabiliser une alimentation électrique (il se décharge lors des chutes de tension et se charge lors des pics de tension).

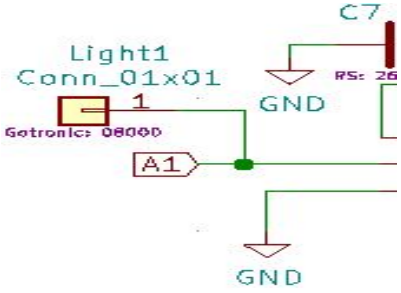




Ces condensateur qui doivent être au plus proche des sorties et qui peuvent générer de brefs pics de consommation sont tout à fait justifié pour éviter que ces pics se transmettent dans toutes les lignes d'alimentations.

Nous avons rajouté des pics pour accéder plus facilement aux valeurs recherchées, par exemple si le 5V ou le 12V est bien transmit, le SDA SCL … Ces pics sont des pics uniquement créés pour vérifier un niveau logique, une tension ou encore une masse.





Liste des composants

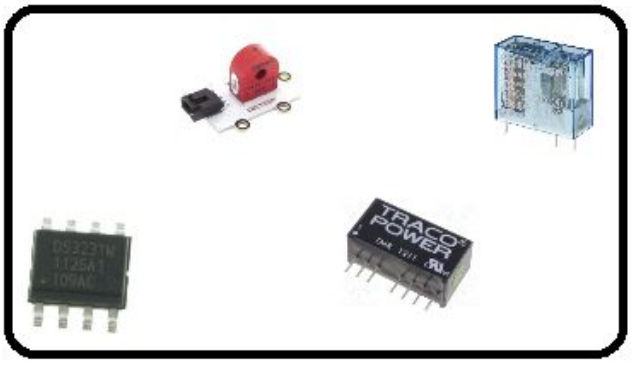
Référence	Valeur	Code Commande	Qty
12V1	Conn_01x01	Gotronic: 08000	1
5V1	Conn_01x01	Gotronic: 08000	1
A1	Arduino_MKR_WAN_1300	MKR Farnell:2851778	1
BT1	Battery	Farnell:1704232	1
C1	100nF	RS: 264-4422	1
C4	4.7 µF	RS : 723-6580	1
C2,C3,C5,C7,C8,C10	1µF	RS: 264-4450	6
CURRENT1	Conn_01x01	Gotronic: 08000	1
D1	Diode	RS :670-8823	1
F1	Polyfuse 500mA 0.25A	RS : 517-6607	1
GND1	Conn_01x01		1
J1 - J6	Terminal_01x03		5
J5, J8	Terminal_01x02		2
K1	FINDER-40.31	RS: 351-601	1
L1	3.3 µH	RS: 745-1174	1
LED1	LED		1
Light1	Conn_01x01	Gotronic: 08000	1
Presence1	Conn_01x01	Gotronic: 08000	1
Q1, Q2	2N7002	RS: 753-3134	2
R1	470k	RS : 223-2619	1
R4	10k	RS : 125-1192	1
R2 - R5	100k	RS : 901-3746	3
R9	330	RS : 679-2049	1
RELAIS1	Conn_01x01	Gotronic: 08000	1
SCL1	Conn_01x01	Gotronic: 08000	1
SDA1	Conn_01x01	Gotronic: 08000	1
U1	DS3231MZ	RS : 778-1484	1
U2	TMR_1211	RS : 433-8236	1

LoRa

THE THINGS NETWORK

Utilisateur

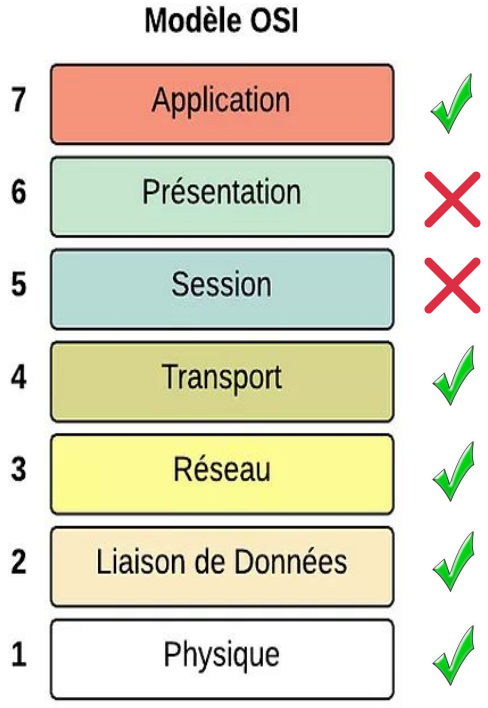
MKR 1300

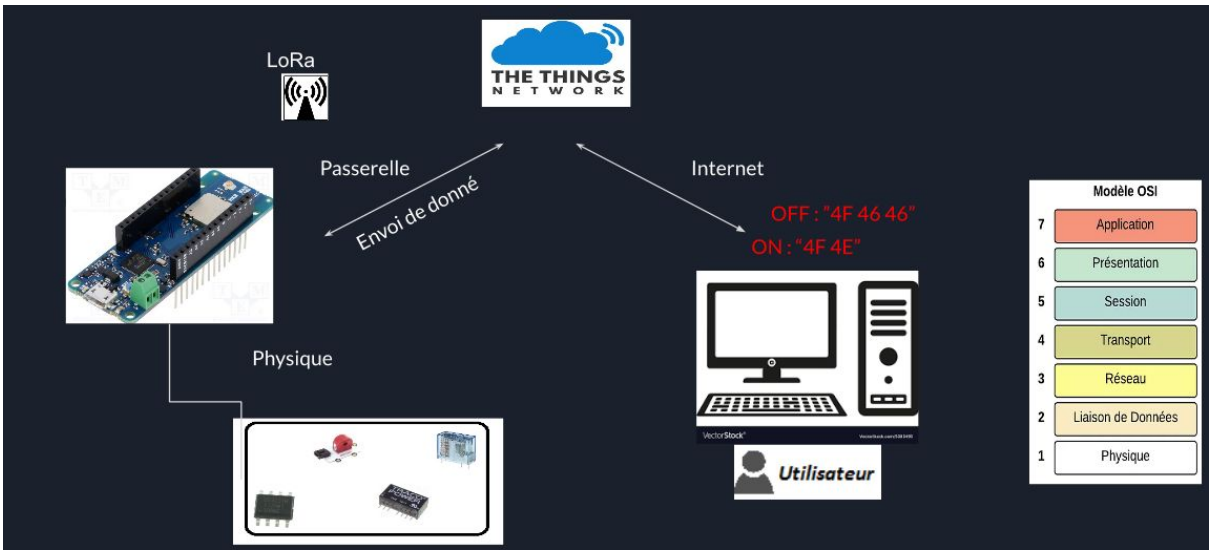


Carte Fille

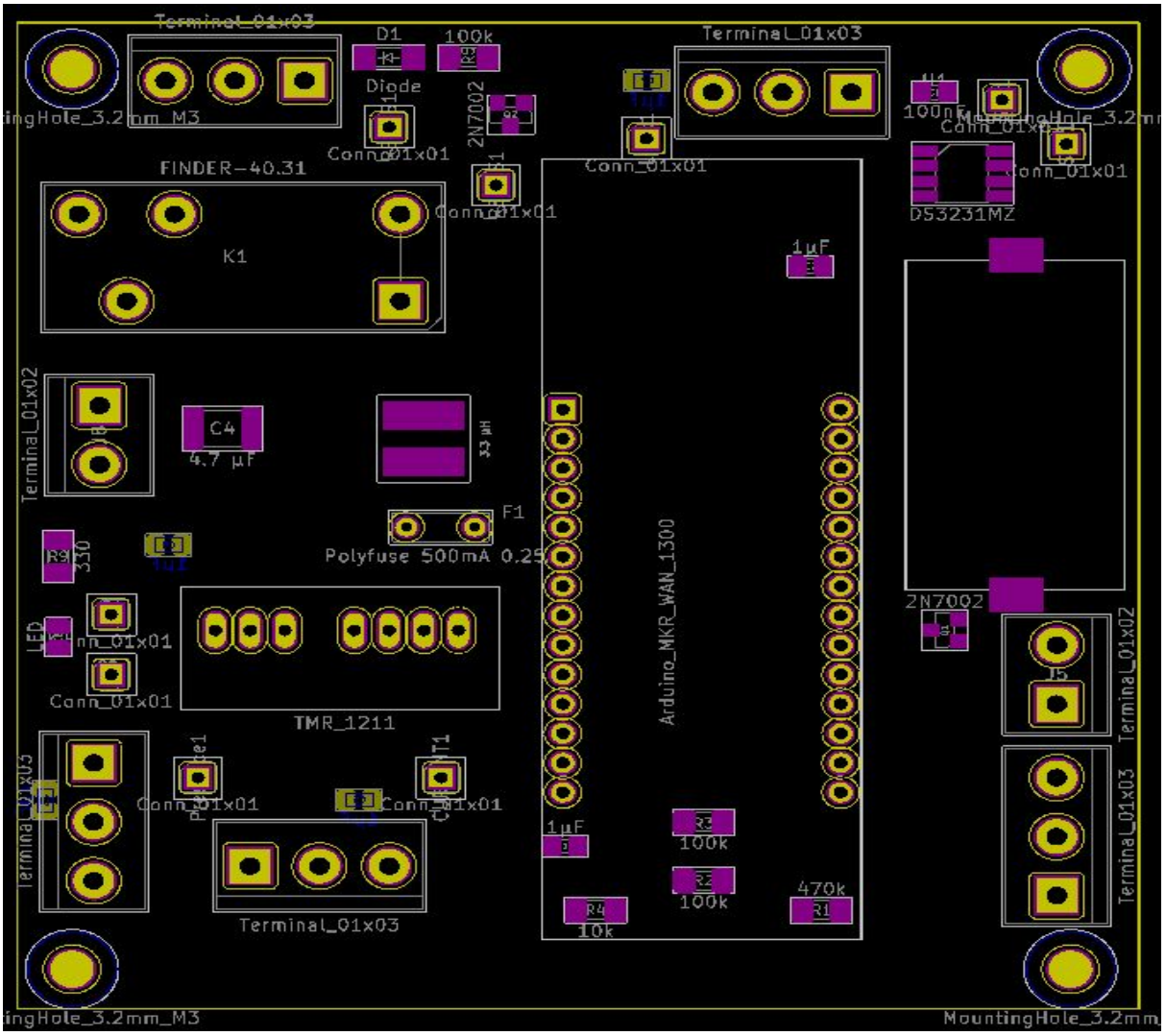
- Application: “ APP ”
- Transport ; ” TCP ”
- Réseaux : “ Adresse IP
- Liaison De données
- Physique

802.3



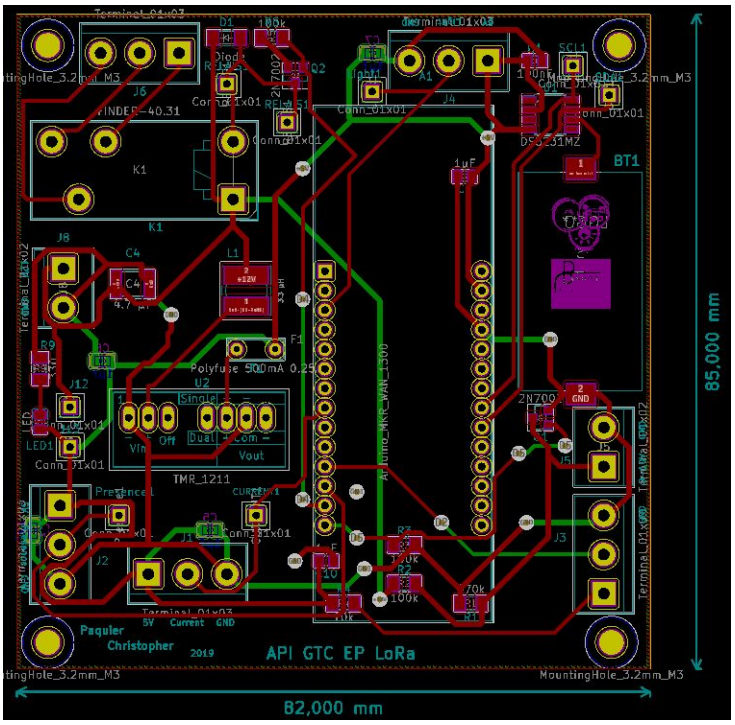


Pose des composants sur la carte avec kicad :

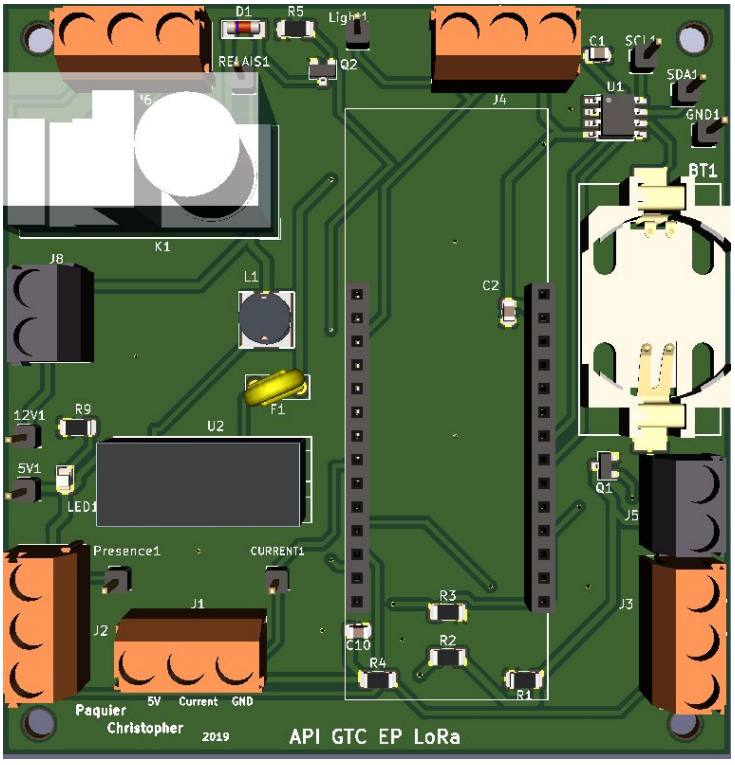


Routage de la carte shield sur kicad

:

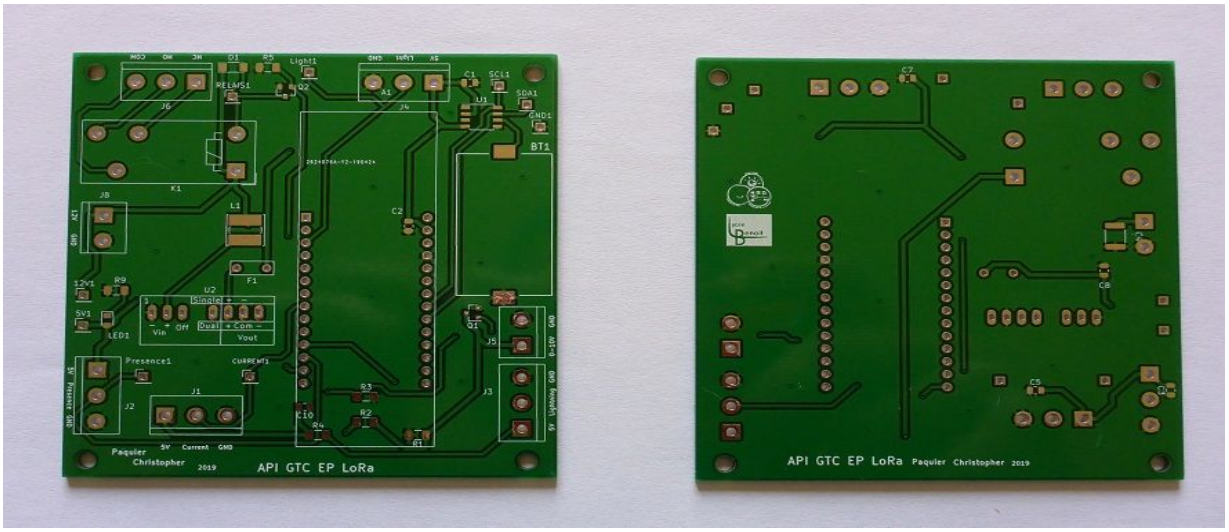


Prototype de la carte shield terminé:

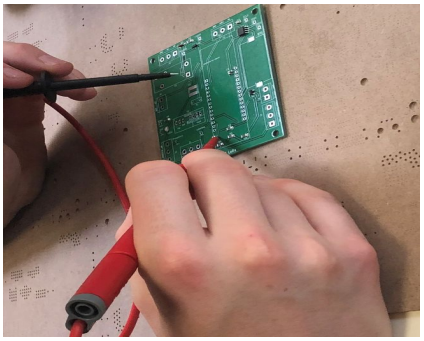


Réalisation de la carte fille

Récupéré auprès de mon professeur les **composants,** le **stencil** et le **pcb** , nécessaire pour réaliser la carte. Le stencil et le PCB ont été commandés sur “JLC PCB”
j’ai dû leur fournir mon fichier gerber pour qu’il réalise la carte.



1) Vérification du routage à l'aide d'un multimètre

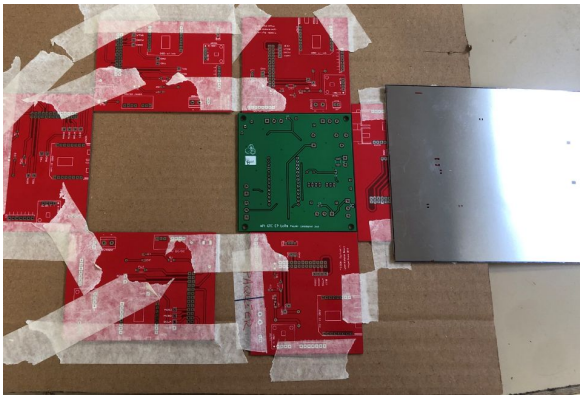


Stencil : Une feuille en alu servant à déposer une pâte à braser sur la surface des composant CMS à souder sur le pcb.

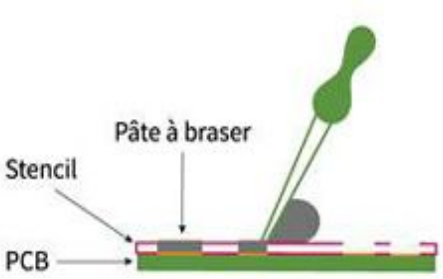
- **L'avantage** c'est qu'une fois le stencil mis en place, le dépôt de pâte à braser est rapide et précis.
- **Les inconvénients** sont que la fabrication du stencil représentent un coût important car ils nécessitent un laser de grande précision.

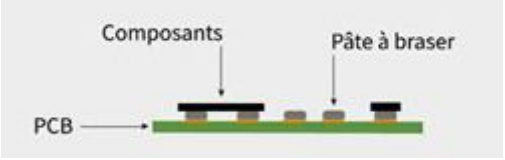
2) Fabrication du socle

J'ai réalisé un socle pour le pcb de sorte que le stencil recouvre bien les parties des composants cms à souder, et que le pcb ne bouge pas pendant l'opération.



3) Pose de la pâte à braser ainsi que des composants “cms”

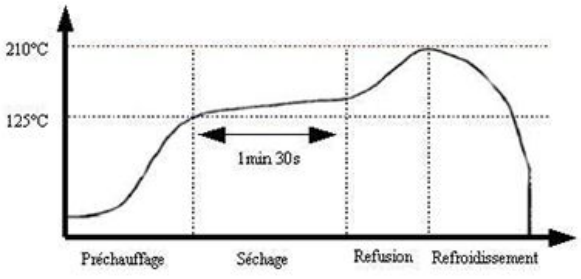




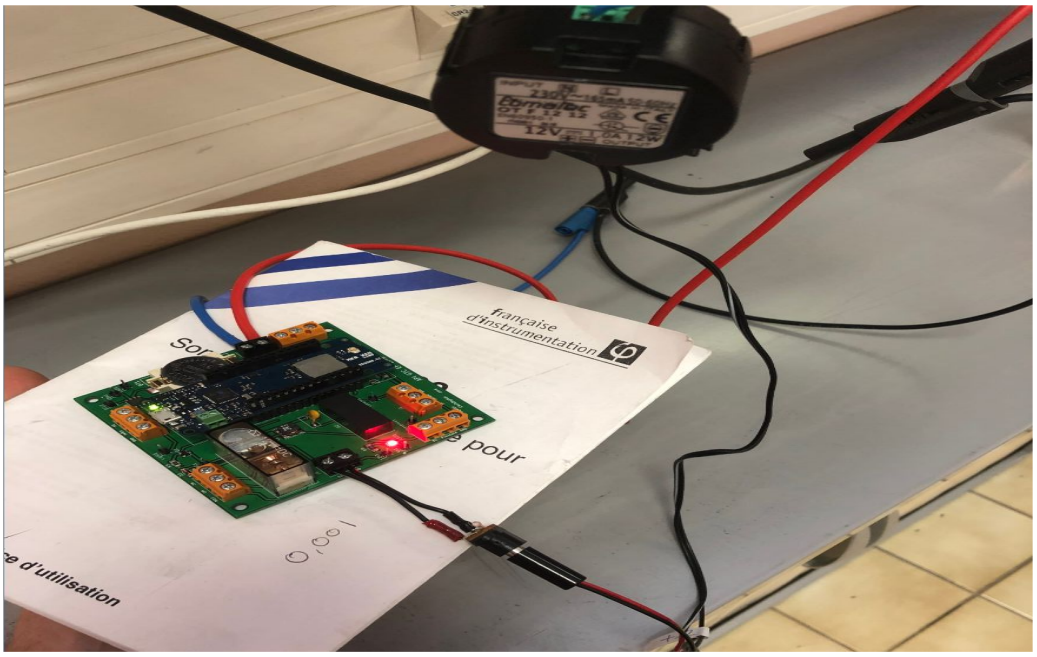
4) Mise au four de la carte

La pâte à braser a besoin de deux etat de température pour bien souder les composants.

Voici le réglage du four



Câblage de la carte sur le lampadaire

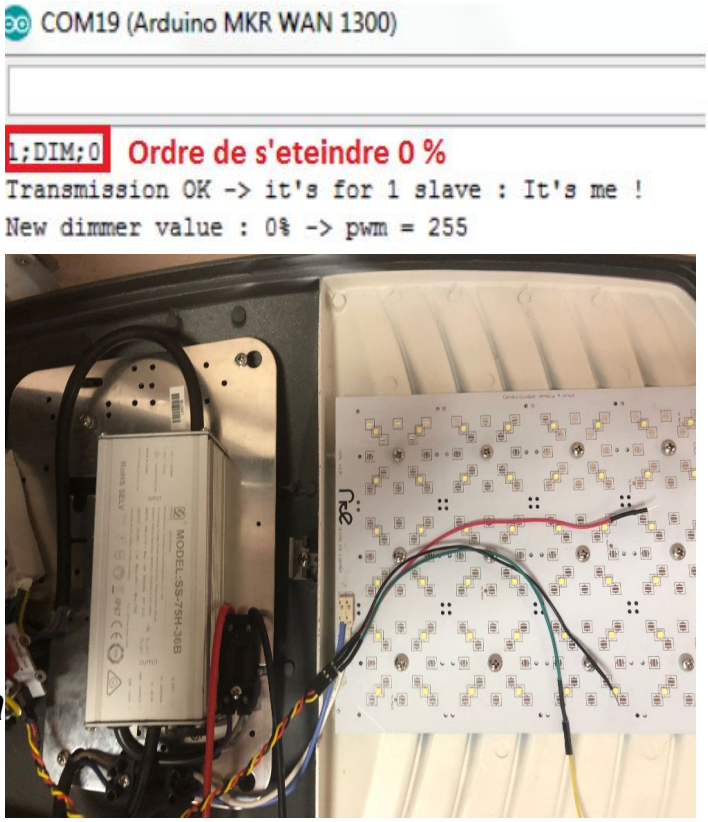


j'ai utilisé une Alimentation à découpage Comatec :

Input	Output
230 Vac	12 Vdc – 1 A
50/60 Hz	12 W

Elle permet de convertir le courant entrant "230V" en courant sortant "12V".
Car j'ai besoin de 12V pour alimenter la carte fille

Test du "Dimmer"



Conclusion

Ce projet m'a permis d'approfondir mes connaissances dans le choix des composants a utilisé pour réaliser ma carte fille .
j'ai appris à utiliser kicad pour le routage de la carte .
pour finir j'ai pu voir que le travail en équipe et très important pour arriver à terme de notre projet.
On a réussi avec toutes nos compétences réunies à mener à bien notre projet. Ma carte et fonctionnelle.
J'ai réalisé deux cartes une pour le client et une pour le bts.

ANNEXE :

Fiche de Réunion :

10/01/19
durée: 1 heure

Travail réalisé	mettre au point le dossier et le diapo, explication du drive, recherche de logo, gantt, début élaboration protocole
Travail à faire	améliorer les fiches de réunion, trouver un logo définitif, déterminer le protocole, finir le gantt

24/01/19
durée: 25 minutes

Travail réalisé	trouver un logo, finir le gantt, TTN commencé, horloge partiel (marche, à adapter), dimmer partiel (LoRa à faire), installation en cours MQTT/Qt, installation en cours InfluxDB/Telegraf/Grafana
Travail à faire	diagrammes de séquences/bloc interne, préparer diapo/dossier, établir le protocole, installation MQTT/Qt/InfluxDB/Telegraf/Grafana

07/02/19
durée: 30 minutes

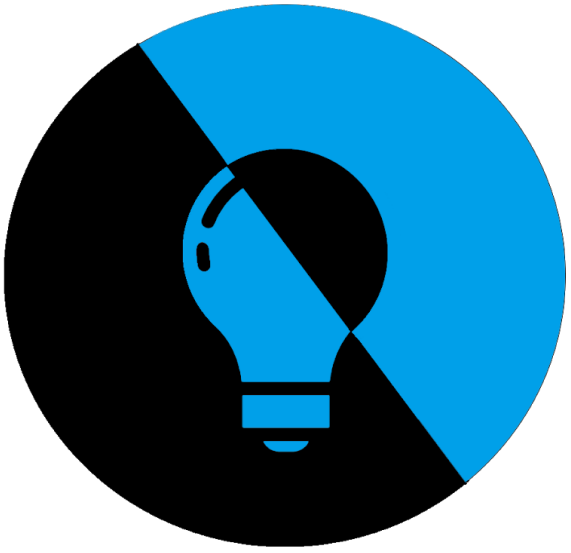
Travail réalisé durant la réunion	mise au point sur le LoRa, correction d'erreurs
Travail à faire	Telegraf, communication entre lampadaire et TTN (LoRa)

08/02/19
durée: 30 minutes

Travail réalisé	test de communication entre TTN et la MKR (vice versa), définition du protocole de communication dans chaque sens (à faire valider par M. Silanus)
Travail à faire	mettre en place le protocole LoRa

1. Fiche de rapport de lecture croisée.		*Réf.* FRLC-XX
Projet API GTC EP LoRa	*Auteur* TILLIOLE DUNCAN	*Date* 25/01/2019

Produit Diagramme de Séquence	Nom du document : Consommation du lampadaire
Lecteur(s) PAQUIER CHRISTOPHER	*Contrôleur des corrections* PAQUIER CHRISTOPHER
Liste des erreurs détectées	
R.A.S	



Dossier de projet

Partie EC2 : Tilliole Duncan

Professeur référent :

Silanus Marc

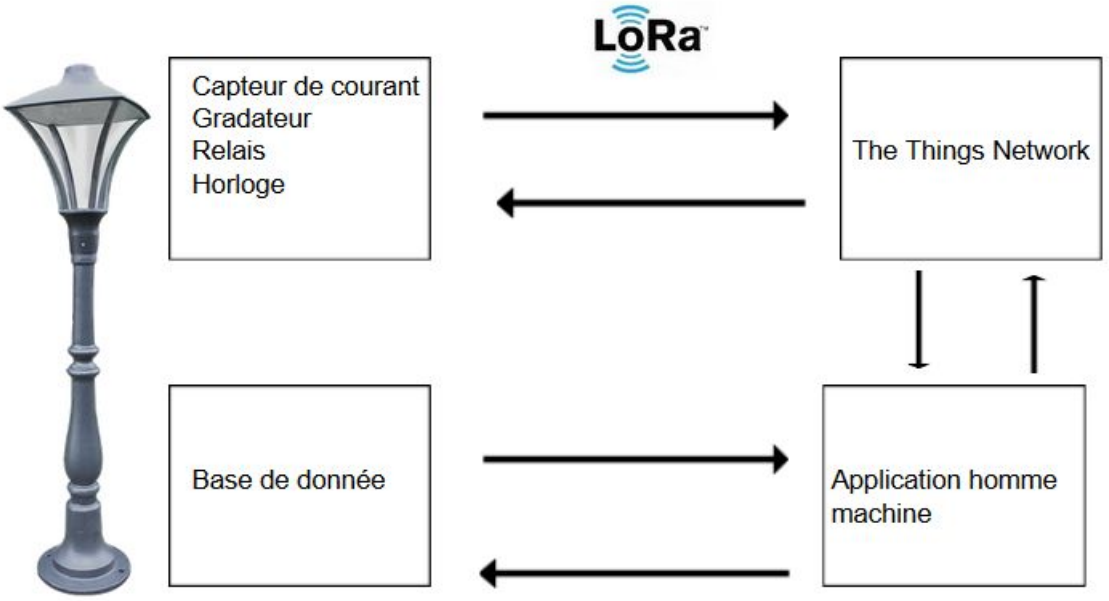
Le commanditaire :

API, 402 avenue des Lacs 84270 VEDENE

API GTC Lampadaire LoRaWan

Introduction

Diagramme de bloc explicatif du projet



Le projet consiste à créer un lampadaire automatique. Nous avons utilisé la technologie de communication LoRa pour mener à bien ce projet. Cette communication aura lieu entre la carte MKR placé dans le lampadaire jusqu'à The Things network, une interface expliqué ci-après.

Ma partie consiste à gérer le gradateur de tension, le relais ainsi que le capteur de courant. J'ai examiné ces parties avec des montages break-out et j'ai ensuite construit une carte électronique possédant l'intégralité des systèmes demandés.

Introduction

Activités EC2	**203 hr**	**Mer 09/01/1**	**Mer 12/06/1**		
Installation des logiciels et des bibliothèques	4 hr	Mer 09/01/19	Jeu 10/01/19	2	EC2
Lire et comprendre les ressources	2 hr	Jeu 10/01/19	Jeu 10/01/19	40	EC2
Diagramme de Gantt	2 hr	Jeu 10/01/19	Ven 11/01/1	41	EC2
Diagrammes de séquences	6 hr	Ven 11/01/19	Mer 16/01/19	42	EC2
Faire schéma de câblage rapide avec Fritzing	2 hr	Mer 23/01/19	Mer 23/01/19	43	EC2
Test capteur + dimmer	**22 hr**	**Mer 23/01/1**	**Ven 01/02/1**		
Comprendre l'intégralité du schéma	2 hr	Mer 23/01/19	Mer 23/01/19	44	EC2
Tester la mesure du courant	6 hr	Jeu 24/01/19	Jeu 24/01/19	46	EC2
Aborder la communication LoRa	4 hr	Ven 25/01/19	Mer 30/01/19	47	EC2
DIMMER	6 hr	Mer 30/01/1	Jeu 31/01/19	48	EC2
Relais	4 hr	Jeu 31/01/19	Ven 01/02/1	49	EC2
Production d'une carte fille + production des documents	**109 hr**	**Mer 20/03/19**	**Mer 12/06/19**		
Concevoir un circuit imprimé qui réponde au cahier des charges	50 hr	Mer 20/03/19	Ven 26/04/19	50	EC2
document SysML a complétés et a adapter	8 hr	Jeu 02/05/19	Ven 03/05/19	52	EC2
Développer une application	50 hr	Jeu 09/05/19	Mer 12/06/19	53	EC2

Diagramme de gantt partie tilliole

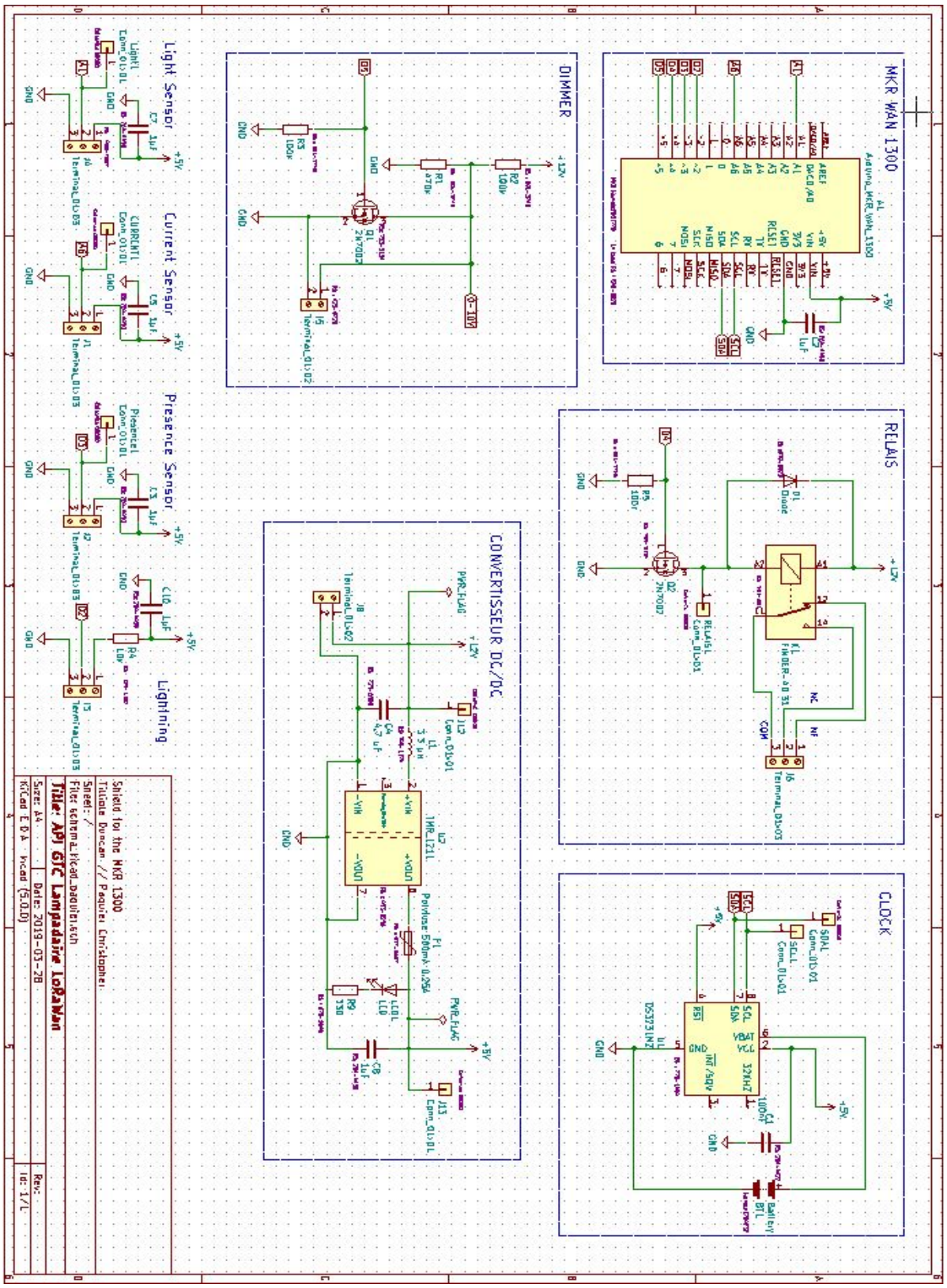
Sur le cahier des charges, plusieurs tâches m'ont été données :

• Tester et valider les éléments proposés sur le synoptique du prototype et sur le site du projet.

• Concevoir/Réaliser/Tester un shield Arduino intégrant les éléments retenus, y compris ceux de l'étudiant 3 EC pour fournir une solution complète.

• Développer une application (la plus avancée possible) permettant de mettre en évidence les différentes fonctionnalités du boitier Lampadaire.

Schéma structurel complet de la carte Shield



Partie Capteur de courant :

Renseignement sur le capteur de courant : Octopus TA12-100

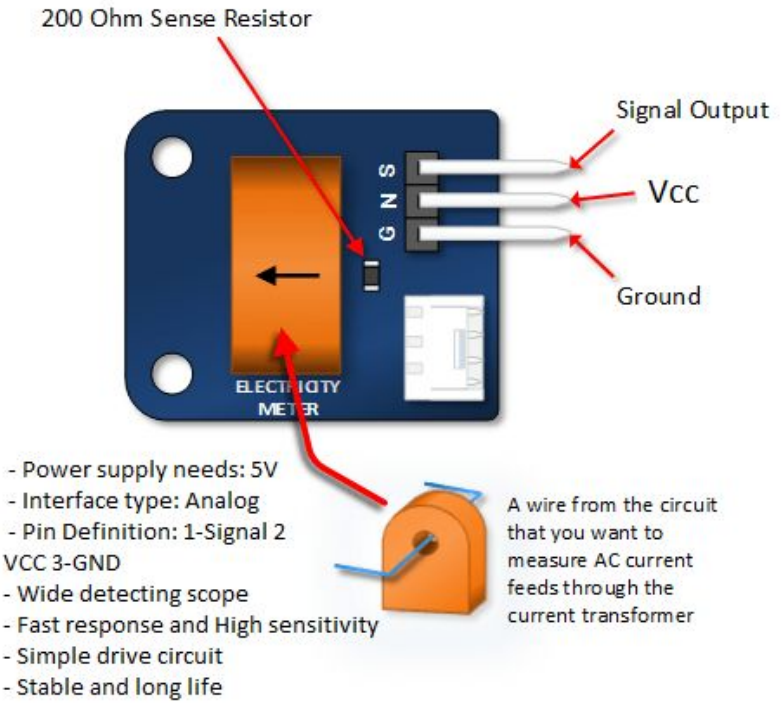
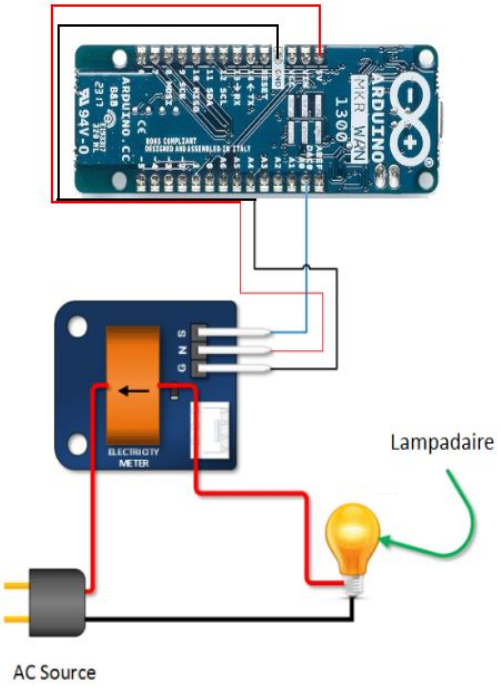


Schéma de câblage pour le capteur de courant



Nouvel environnement Arduino

Pour programmer via la carte MKR WAN 1300, j'ai dû aller dans le gestionnaire de carte (sur le logiciel Arduino) pour installer la dernière version de la MKR WAN 1300. Après ça j'ai installé la bibliothèque pour avoir accès aux programmes tels que "First Configuration" ; "LoraSendandReceive" qui servira à la communication LoRa.

Programme Arduino pour la mesure du courant

Après avoir lu la documentation du capteur, j'ai remarqué que la sortie Source doit être branchée à une entrée analogique de la carte MKR 1300. Donc le signal délivré par le capteur est Analogique.

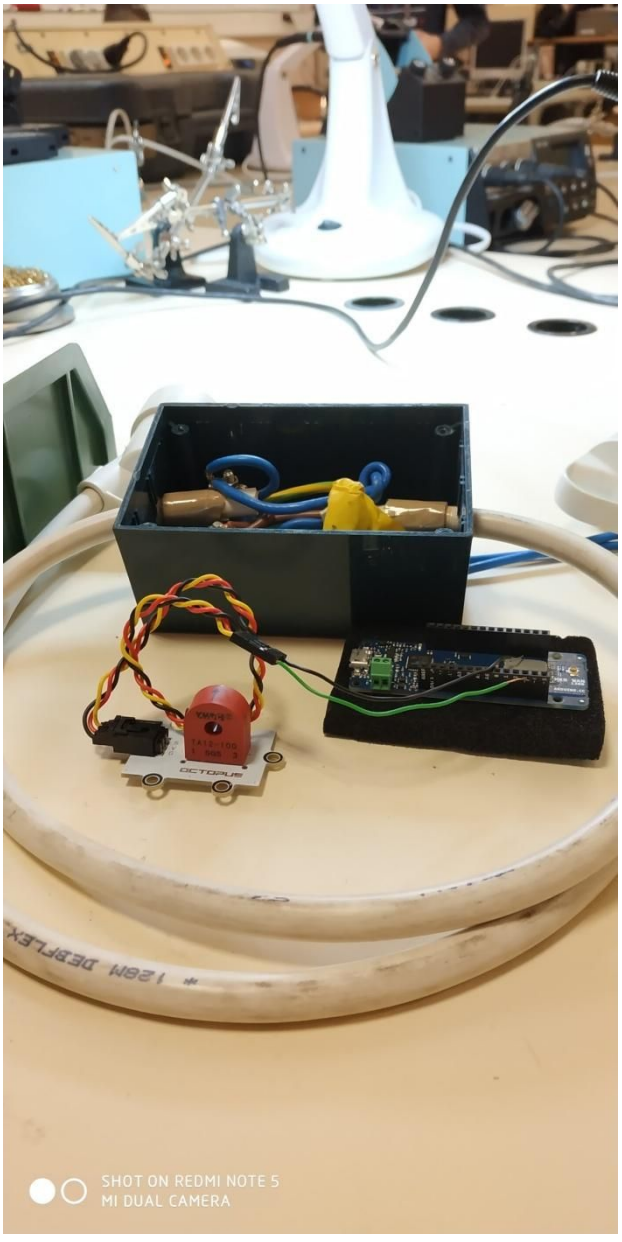
J'ai choisi la broche à utiliser sur le programme arduino (La broche A0) et j'ai ensuite utilisé la fonction "analogRead()" avec plusieurs variables crées pour faciliter la compréhension du programme.

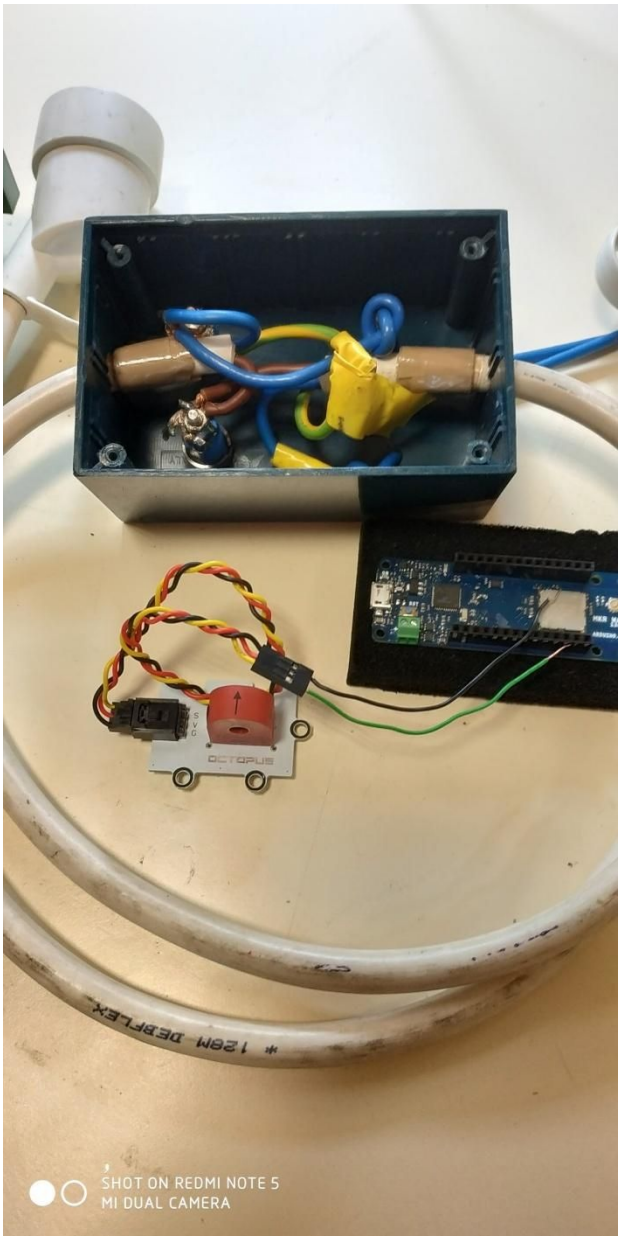
Programme Arduino réalisé :

```
#define CURRENT_PIN A0
#include <Wire.h>

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  pinMode(CURRENT_PIN, INPUT);
}

void loop()
{
int sensorValue;
sensorValue = analogRead(CURRENT_PIN);
Serial.println(sensorValue);
delay(1000);
}
```

Après avoir fait ce programme ainsi que le schéma de câblage sur Fritzing, j'ai voulu mettre en oeuvre ce capteur. A ce moment, la tête du lampadaire n'était pas encore à disposition, j'ai donc essayé de trouver un moyen de tester le capteur sans la tête du lampadaire.

Test du capteur de courant Octopus TA12-100

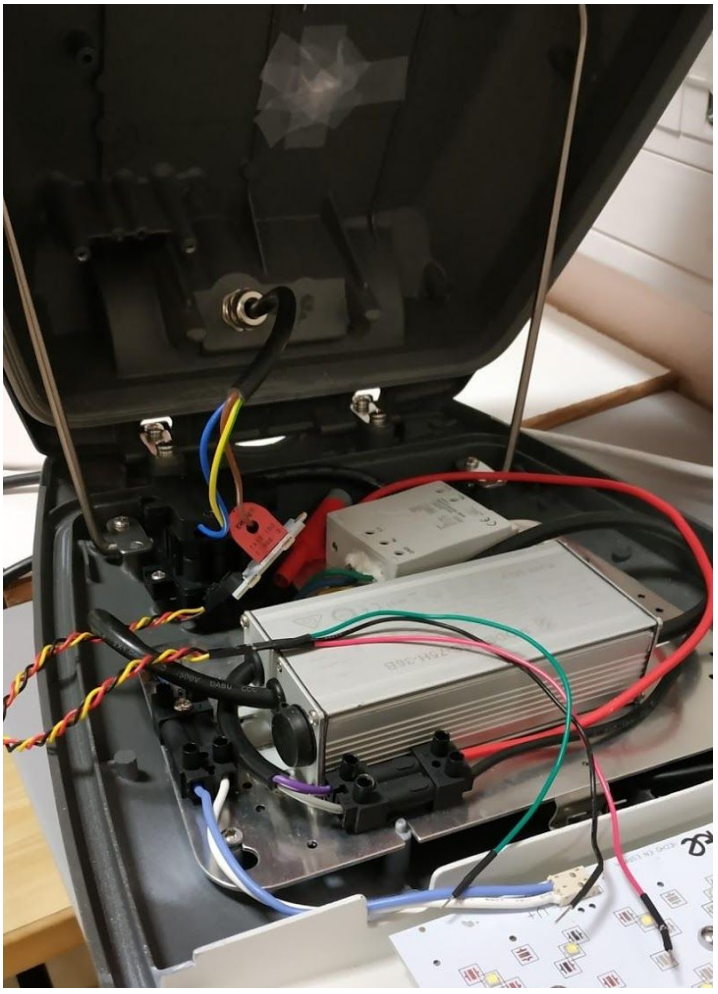




J'ai commencé par le programme Arduino. A l'aide du programme de M. Silanus, j'ai créé un programme qui permet uniquement d'afficher la mesure du courant.
Après avoir pris connaissance de la documentation du capteur TA12-100, j'ai commencé à câbler le capteur sur la MKR 1300. Pour réaliser la mesure d'un courant j'ai dû commencer par dévisser un boitier qui donne une tension représentative de la tension secteur.

Pour continuer je devais dessouder l'un des câbles se trouvant à l'intérieur du boitier mais par manque de temps à la fin d'une séance je n'ai pas réalisé ce travail. Le lampadaire fut livré à la séance suivante, je suis donc passé directement sur la mesure de courant du lampadaire.

Mise en œuvre et expérimentation du capteur de courant



Le capteur de courant à déjà été utilisé auparavant, nous n'avons donc pas eu à le mettre sur le lampadaire.
Pour tester la mesure du courant j'ai donc suivi le schéma Fritzing. Après avoir mis sous tension le lampadaire, la tension représentative du courant relevé aurait dû être affichée. Hors, aucune tension ne fut affichée sur le moniteur série (arduino). Toutes les 1 seconde, le programme nous renvoyait 0V.

Pour remédier à ce problème nous avons consulter la documentation du capteur plus profondément mais nous n'avons eu aucun résultat.

Nous avons ensuite pensé à un défaut capteur, c'est à dire, si le capteur avait un dysfonctionnement. On a donc changé le capteur par un autre mais le problème a persisté.

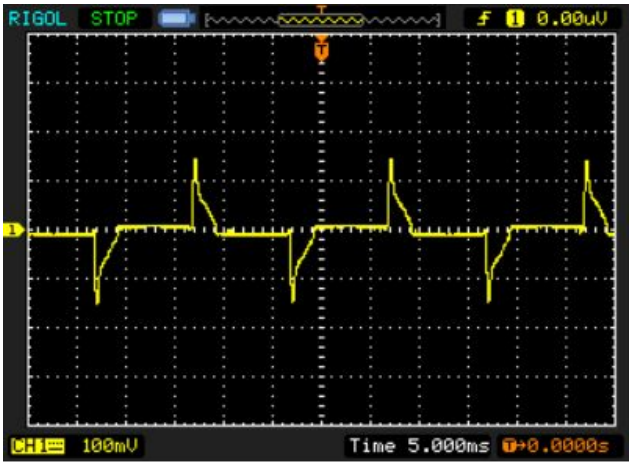
Après ça, à l'aide d'un multimètre, on a donc regardé si la Source du capteur envoyer un signal à la MKR et si le capteur était bien alimenté.

L'alimentation de 5V était bien émise jusqu'au capteur, donc le capteur était bien alimenté mais la source ne délivrait aucune tension. Pourtant le lampadaire était allumé.

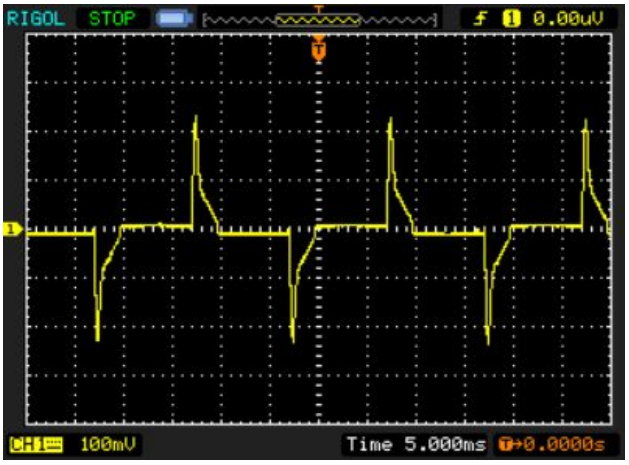
Mise en œuvre et expérimentation du capteur de courant

Après quelques semaines nous avons réessayé l'expérience avec un nouveau programme et un oscilloscope branché sur le signal délivré par le capteur.

Valeur intermédiaire de la tension représentative du courant :



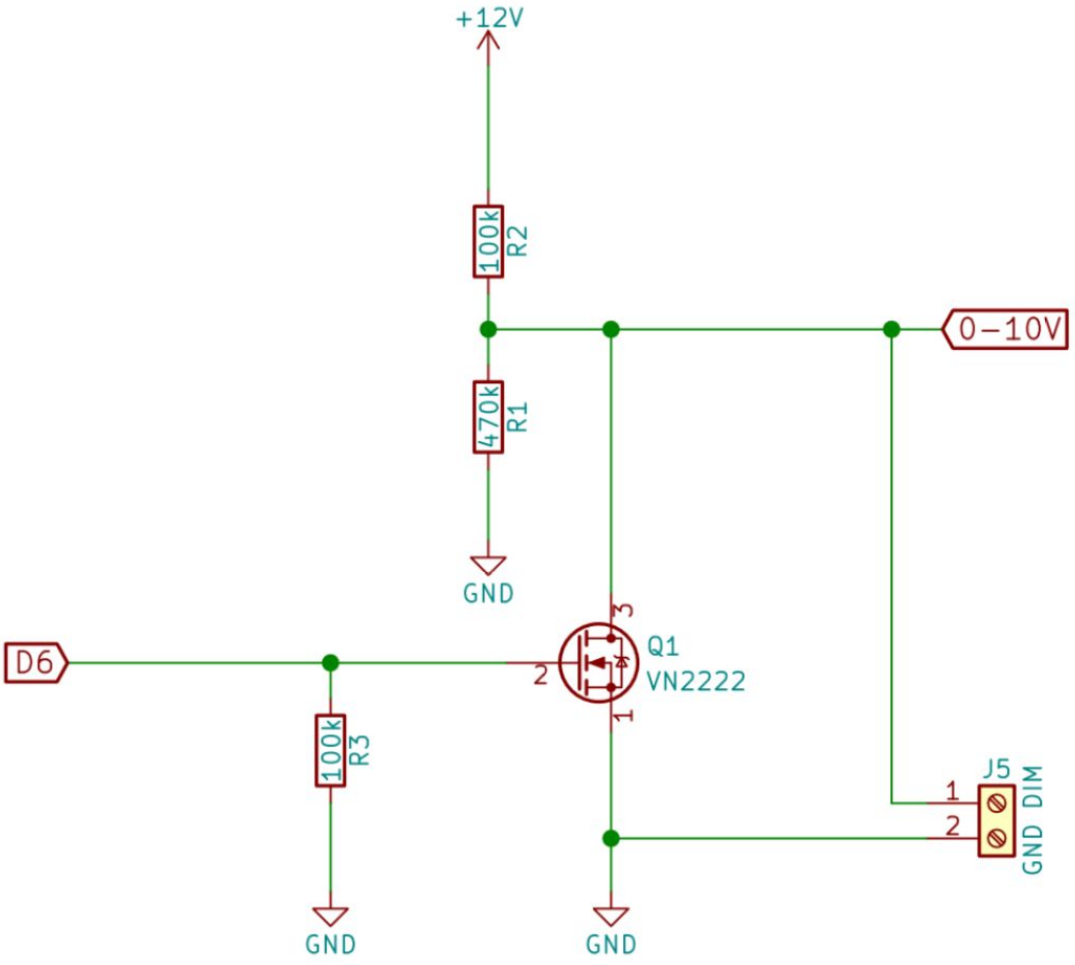
Valeur maximale de la tension représentative du courant :



L'oscillateur montre un signal alternatif qui peut varier jusqu'à environ 200 mV et -200 mV. Sur le programme arduino, la tension relevé a été convertie et correspond à la valeur du courant relevé.

Partie DIMMER :

Analyse du schéma structurel du DIMMER



Ce schéma correspond au DIMMER (il sert à graduer la lumière du lampadaire, de 0 à 100%).

Dans l'expérimentation;
• L'alimentation 12V va être utilisée pour un générateur de tension.
• La sortie D6 est une sortie digital Arduino, elle sera donc connecté à la MKR 1300.
• J5 sert à relier le lampadaire au circuit.

Si on sature le transistor Q1 par une tension de 5V sortant de D6, il se comportera comme un fil. Le 10V sera amené à la masse et J5 recevra 0V.
A l'inverse, si on bloque le transistor Q1 par une tension de 0V, il se comportera comme un interrupteur ouvert, le 10V sera envoyé sur J5.

Modulation de largeur d'impulsion (PWM) en correspondance avec le DIMMER

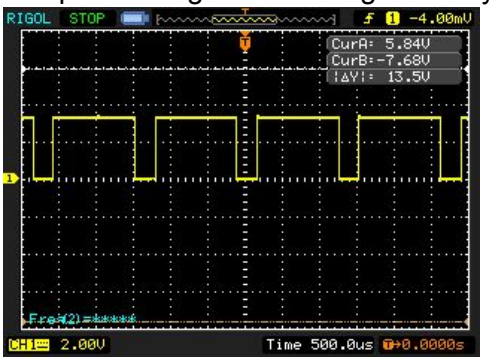
Le signal PWM doit subir un filtrage passe-bas pour en extraire le signal de consigne. Le signal PWM est un signal carré dont le rapport cyclique est $\alpha=U_0/m$, où m est la valeur maximale de la porteuse. La moyenne de ce signal carré est précisément égale à U_0.

Lorsque la consigne est lentement variable par rapport à la porteuse, il faudra appliquer un filtrage passe-bas pour restituer les variations de basses fréquences de la consigne. En pratique, le signal PWM est utilisé pour commander un circuit de puissance travaillant en commutation, et le filtrage passe-bas est assuré par une bobine en série avec la charge. Pour commander un pont en H, il faudra aussi disposer du signal complémentaire, obtenu avec une porte NON.

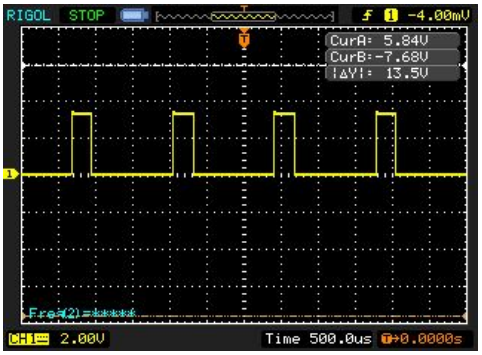
La modulation de largeur d'impulsion sert à gérer plusieurs systèmes électroniques différents tels que :

- Variateur de vitesse pour perceuse
- Alimentation à découpage (la variation du rapport cyclique est faite automatiquement par le circuit de régulation).
- Amplification audio en classe D

Cette modulation est aussi utilisée dans les gradateurs de lumière sous tension continue et c'est notre cas actuel, nous voulons graduer la lumière du lampadaire.
Voici plusieurs prises à l'oscilloscope qui montrent le changement du rapport cyclique comparé au signal de consigne envoyé :



Modulation de largeur d'impulsion à 20 %



Modulation de largeur d'impulsion à 80 %

Programme Arduino pour le DIMMER ON/OFF

```
void loop()
{
  if (Serial.available() > 0)
  {
            //End Of Frame is \n
            String messageFromLoRa = Serial.readStringUntil('\n');
            Serial.println(messageFromLoRa);
            Serial.print("Transmission OK -> it's for ");  // No error transmission

            // Test for receiver address
            String messageFor = getAddress(messageFromLoRa);
            if ( messageFor == slaveAddress || messageFor == diffusionAddress) // It's for me ?
            {
            Serial.println(" slave : It's me !");
            // look for order
            String order = getOrder(messageFromLoRa);
            if (order == "DIM")
            {
            // look for data
            String data = getData(messageFromLoRa);
            dimmer(data);
            }
            }
  }
}
```

Programme provenant en partie du programme de M. Silanus

Le programme du DIMMER est long, j'ai donc pris pour exemple la partie la plus importante ;

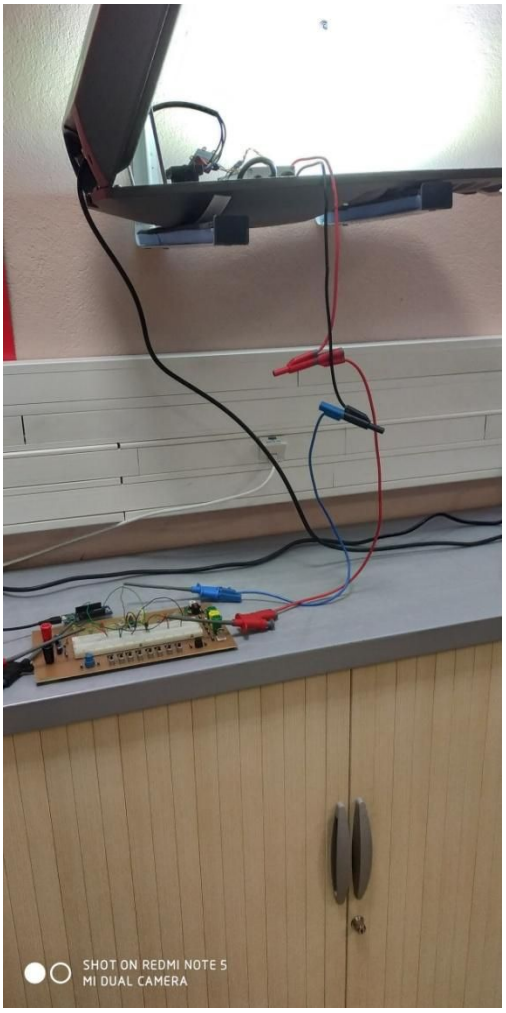
Dans un premier temps, dans la boucle if, on voit l'utilisation de la fonction Serial.readStringUntil() qui va servir à lire ce que l'utilisateur a écrit dans le moniteur série.

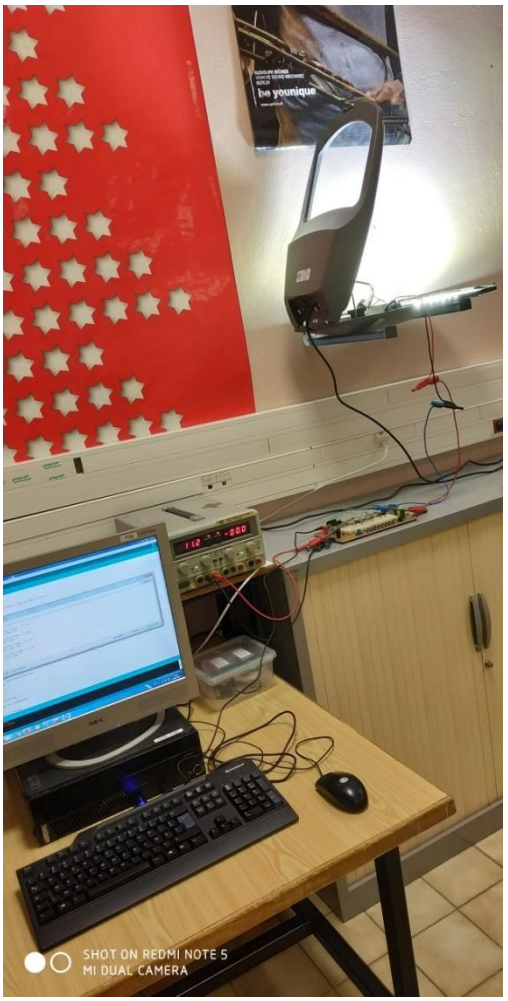
Ensuite on initialise une variable qui est égale à une fonction écrite à la suite du programme. La fonction getAdress() sert à lire et écrire l'adresse de l'esclave.

La seconde boucle IF permet d'envoyer l'ordre demandé au dimmer, si le message envoyé dans le moniteur série correspond bien à l'adresse de l'esclave on rentrera dans cette boucle et on lira la donnée qui suit l'adresse de l'esclave, c'est à dire "DIM".

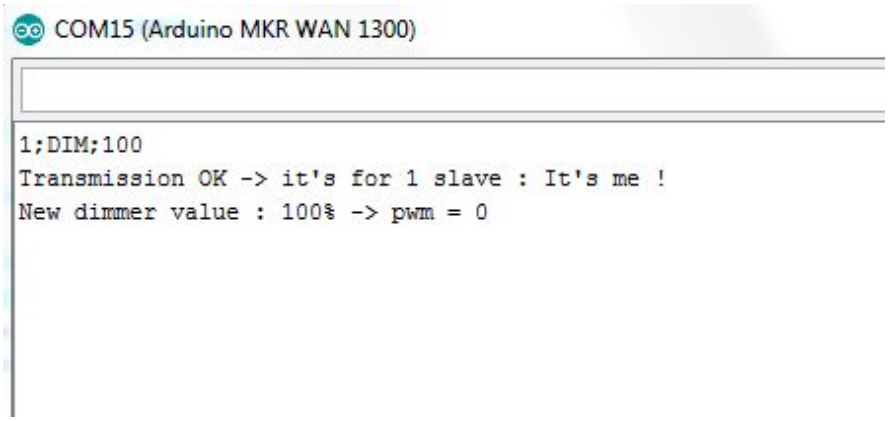
Si l'utilisateur écrit "1;DIM" (1 = adresse de l'esclave), il rentrera dans la dernière boucle IF qui servira à envoyer la donnée demandé par l'utilisateur au dimmer. Il devra écrire une valeur qui varie de 0 jusqu'à 100, et non de 0 à 255 car une fonction prend en compte ce changement pour la clarté du programme. *L'ordre envoyé doit être :* ***[Adress;DIM;Data]***

Photo des essais du DIMMER





J’ai câblé les éléments intégrés dans le schéma d’implantation, j’ai mis en fonctionnement le lampadaire et enfin, j'ai testé le programme vu à la page précédente.



Le programme a bien fonctionné ;

L’ordre envoyé était 1;DIM;100 et j’ai remarqué sur le lampadaire le changement de luminosité.

La technologie de communication utilisée

Description de la technologie LoRa :

En France, deux réseaux sont particulièrement mis en avant : SigFox et LoRa. Le réseau LoRa signifie Long Range ou « longue portée » en français. Il s'agit d'une technologie qui permet aux objets connectés d'échanger des données de faible taille en bas débit. Premier impact, cela permet de réduire la consommation énergétique des appareils, leur conférant jusqu'à 10 ans d'autonomie. Cette technologie utilise à la fois les fréquences radio libre 868 MHz et Internet.

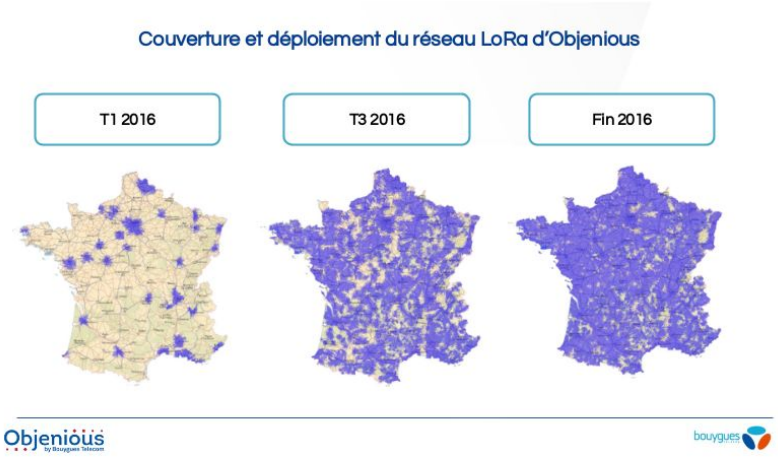
LoRaWAN : un protocole multifonctionnel :

La première classe permet à un objet d'envoyer des informations vers une antenne puis d'en recevoir immédiatement après l'envoi. Si le serveur veut envoyer des informations à l'objet, il devra attendre le prochain cycle d'envoi. C'est le mode le moins gourmand en énergie.

La seconde classe permet à l'objet connecté de recevoir des données à des intervalles réguliers et paramétrés à l'avance.

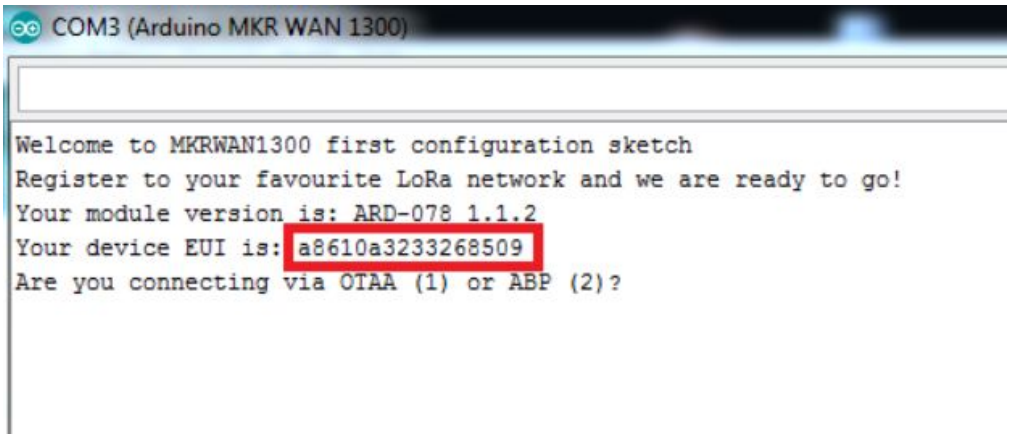
La dernière classe permet au récepteur de recevoir des données en continu. Ce dernier s'avère le plus énergivore.

Pour le projet API GTC EP LoRa, nous utilisons la première classe du protocole pour consommer le moins d'électricité possible.



Approche sur la communication LoRa

Les données envoyées par la technologie LoRa sont envoyées sur internet, sur le site The Things Network. Pour acquérir les données envoyées il faut au préalable inscrire l'objet qui communique dans l'espace personnel de The Things Network, après ça, l'échange peut-être fait.



Dans les programmes d'exemples de la carte MKR 1300, j'ai utilisé le programme First Configuration pour connaître son dispositif EUI. Dans la communication LoRa, Je vais en avoir besoin pour m'inscrire et communiquer avec The Things Network.

Ce device EUI représente en quelque sorte une adresse MAC, elle ne change pas et définie le "nom" de la carte.

Après avoir écrit mon device EUI, je l'ai communiqué aux élèves d'IR qui m'ont inscrit dans The Things Network, ils m'ont donnés un code en retour à inscrire sur le programme de ma carte :

```
// Replace with keys obtained from TheThingsNetwork console
#define SECRET_APP_EUI "70B3D57ED00169C0"
#define SECRET_APP_KEY "1AAA136C4E7AA4DF04C45D7F63E732F5"
```

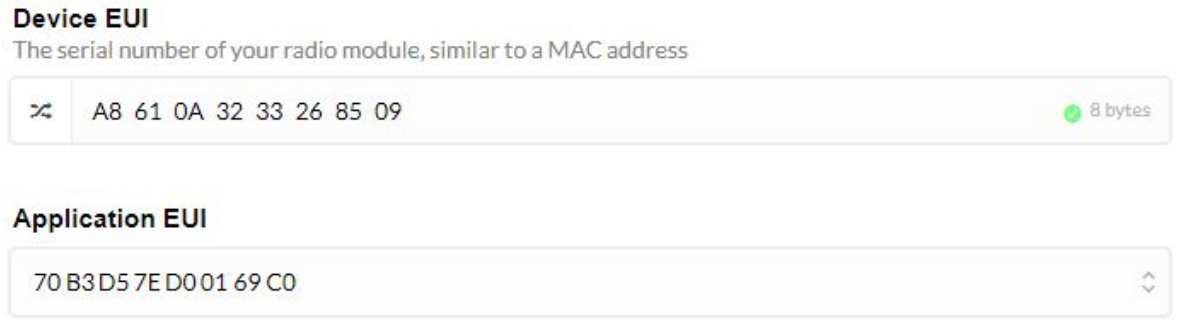
Cette partie du programme va me servir à communiquer avec the things network.

Ensuite, j'ai commencé à lire le programme donné en exemple intitulé "LoraSendAndReceive" pour comprendre comment envoyer des données aux élèves d'IR.

Dans la suite du rapport The Things Network sera aussi appellé « TTN ».

Connexion à The Things Network

Je n'arrivais pas à communiquer avec TTN alors que les élèves d'IR avaient rentré mon Device EUI ainsi que mon application EUI. J'ai donc pris en main The Things network pour retracer les erreurs et j'ai remarqué que mon Device EUI n'était pas exactement celui écrit par les élèves d'IR, j'ai donc réécrit exactement le Device de la carte et la communication fut une réussite.



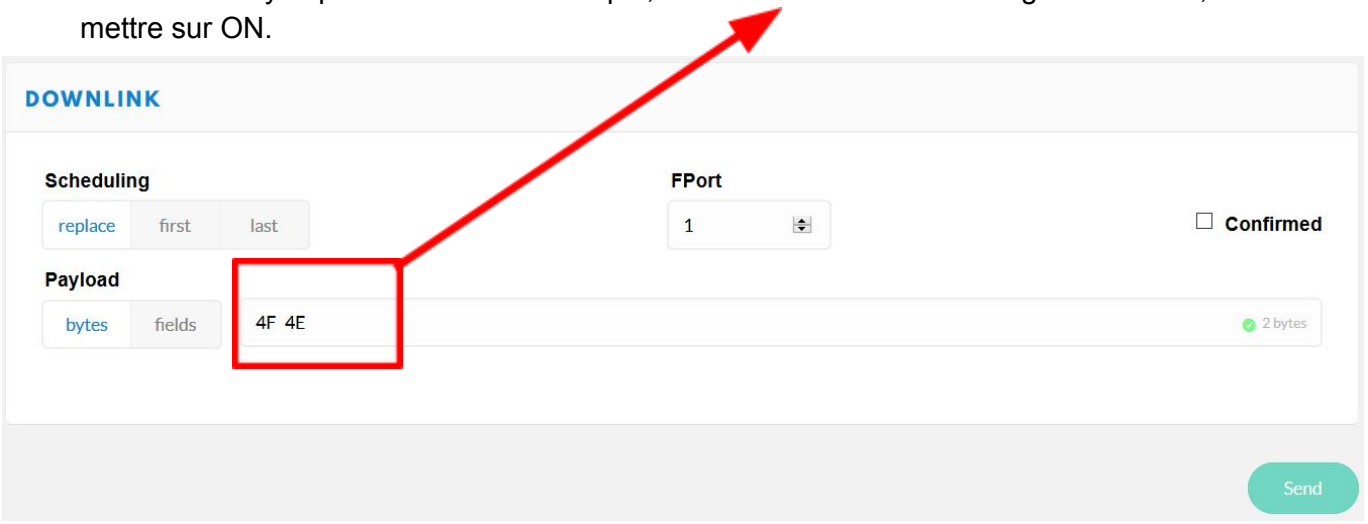
On peut voir que la carte était connectée sur le réseau TTN il y a 1 mois, la configuration de la carte est une réussite.

Création d'un protocole

Pour envoyer des ordres de TTN aux lampadaires nous avons dû créer un protocole à travers la communication LoRa. Nous avons organisé une réunion pour créer le protocole.

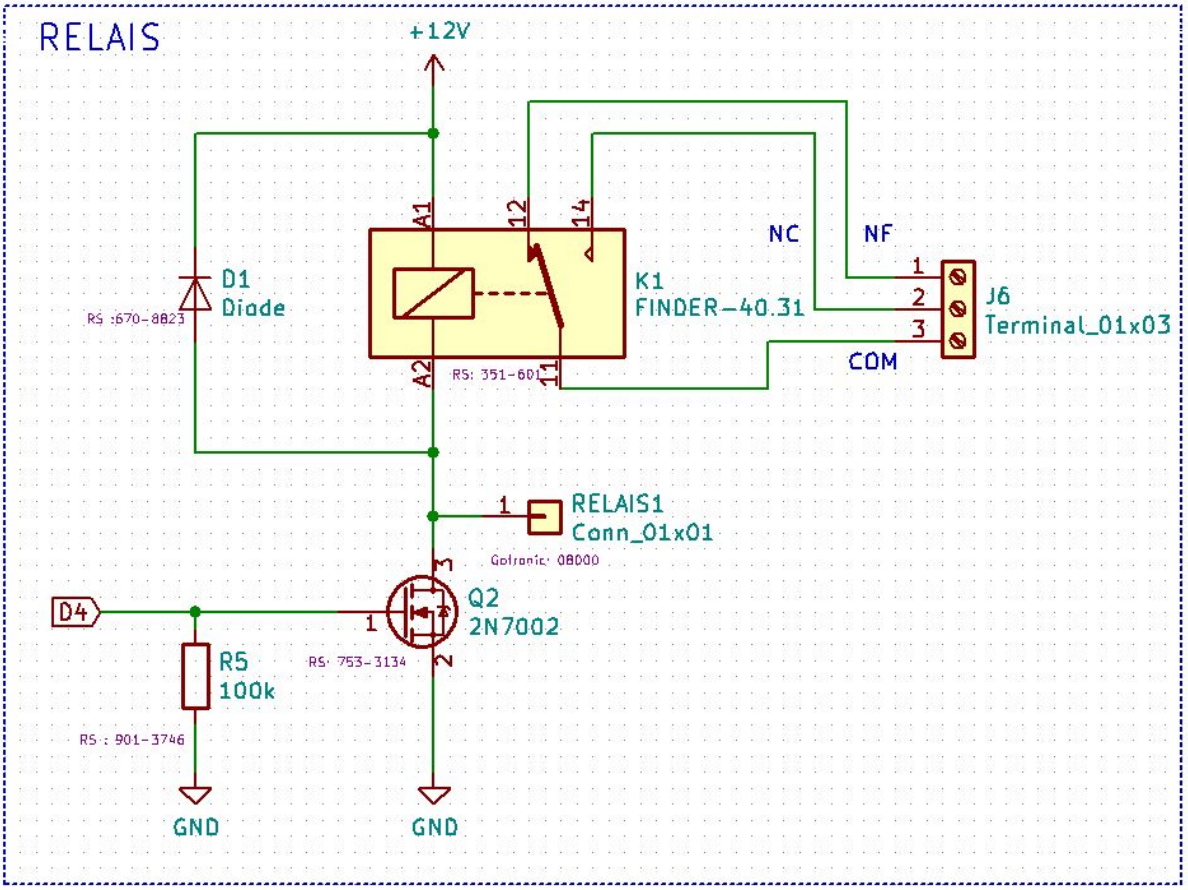
TTN -> MKR		
Commande Ascii	**ORDRE demandé**	**Explications :**
30 / 31 30 30	DIM	gradateur, de 0 à 100%
4F 4E	ON	allumer lampadaire
4F 46 46	OFF	éteindre lampadaire
41 55 54 4F	AUTO	passe en mode auto
67 [2] [2] [2] [2]	CONF	
MKR -> TTN		
Code sur [XX] octects	**Envoie automatique**	**Explications :**
XX XX XX XX	Courant	Envoie du courant à chaque trame
XX	Etat	Envoie de l'état à chaque trame

Ce protocole défini tous les ordres que le réseau TTN peut envoyer à la MKR et les données envoyés par la MKR. Par exemple, la commande « 4F 4E » va gérer le relais, le mettre sur ON.



Pour envoyer un ordre de TTN, nous devons écrire cette ordre en code ASCII. Nous n'avons pas choisie ce code, il nous l'a été imposé par The Things Network.

Partie Relais

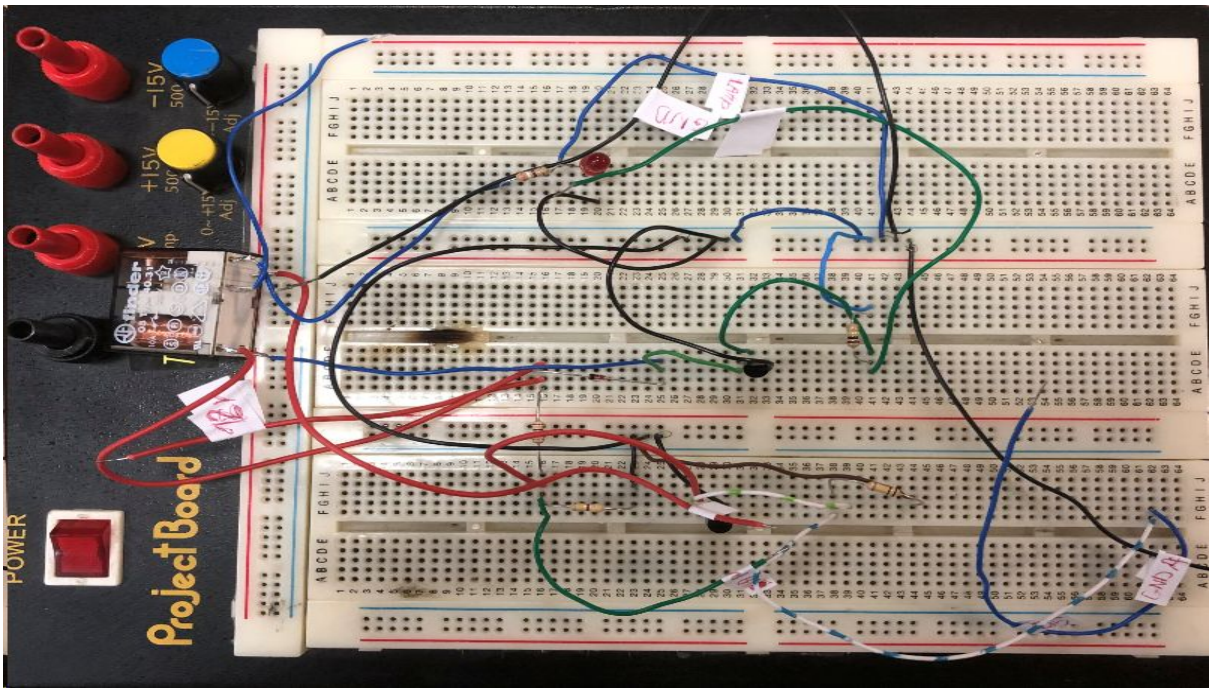


La partie Relais est un forçage ON/OFF, cette partie assure un interrupteur permettant par exemple l'allumage de guirlandes de noël.

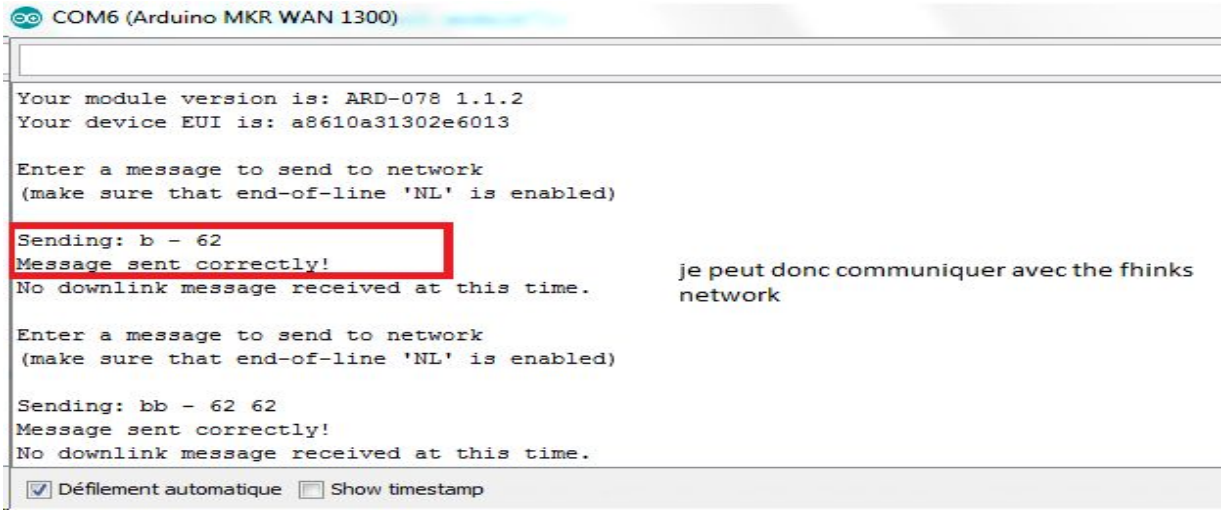
J'ai encore utilisé le transistor VN2222 pour câbler cette partie, sur ce schéma si le transistor est saturé, le 0V sera envoyé au relais FINDER 40.31. Inversement, si le transistor est bloqué, le relais recevra du 12V.

Christopher Paquier avait câblé cette partie sur une platine d'essais en ayant produit le programme qui le fait fonctionner. J'ai câblé la partie relais sur ma platine d'essais avec la partie DIMMER pour pouvoir gérer le relais ainsi que le dimmer en même temps.

Câblage DIMMER + RELAIS avec la communication LoRa



Après avoir câblé la partie RELAIS sur ma platine j'ai téléversé le programme Arduino dans la carte MKR pour que le relais ainsi que le dimmer fonctionnent simultanément.
Ensuite, comme ma carte est enregistrée dans le réseau TTN j'ai essayé le programme SendAndReceive pour communiquer avec celui-ci. Cette démarche fut une réussite et j'ai donc fusionné le programme SendAndReceive avec celui du relais / dimmer.



Après quelques essais, j'ai réussi à communiquer un ordre du réseau TTN à la MKR. Je peux à présent contrôler le dimmer ainsi que le relais via la plateforme internet, The Things Network

V_{in}, une entrée Arduino qui peut alimenter la carte ?

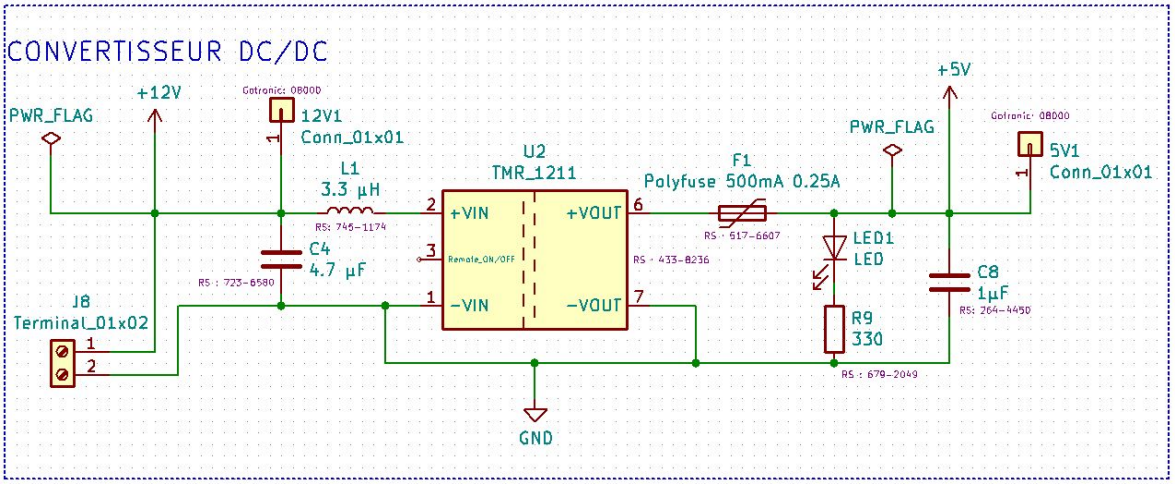
Pour une question de confort, nous avons cherché à savoir si une broche de l'Arduino pouvait alimenter celle-ci.

Dans le futur du projet, nous allons créer une carte shield qui sera connectée à la carte MKR, donc au lieu d'avoir un fil d'alimentation en l'air, nous aurions préféré que la carte shield alimente directement la MKR.

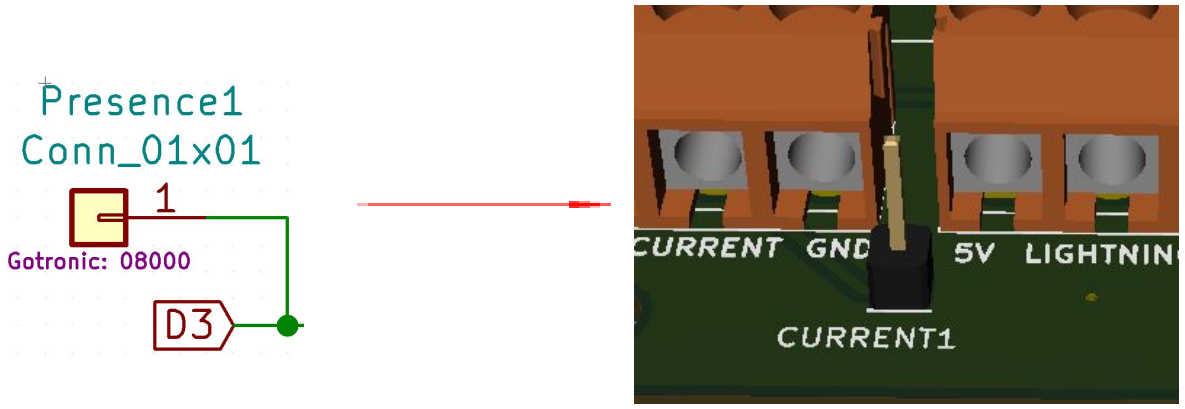
Nous avons regardé sur plusieurs sites, dont le site officiel d'Arduino et nous avons trouvé l'information recherché sur ce site : « http://www.mauroalfieri.it/elettronica/arduino-mkr-wan-1300-mkr-gsm-1400.html »

La MKR 1300 Wan et la MKR 1400 possèdent énormément de similitudes mais sont différentes sur certains points comme l'alimentation de V_{in}.
Comme expliqué dans le lien ci-joint, la MKR1400 peut être alimentée sous 12V et la MKR1300 WAN peut être alimentée au maximum sous 5V.

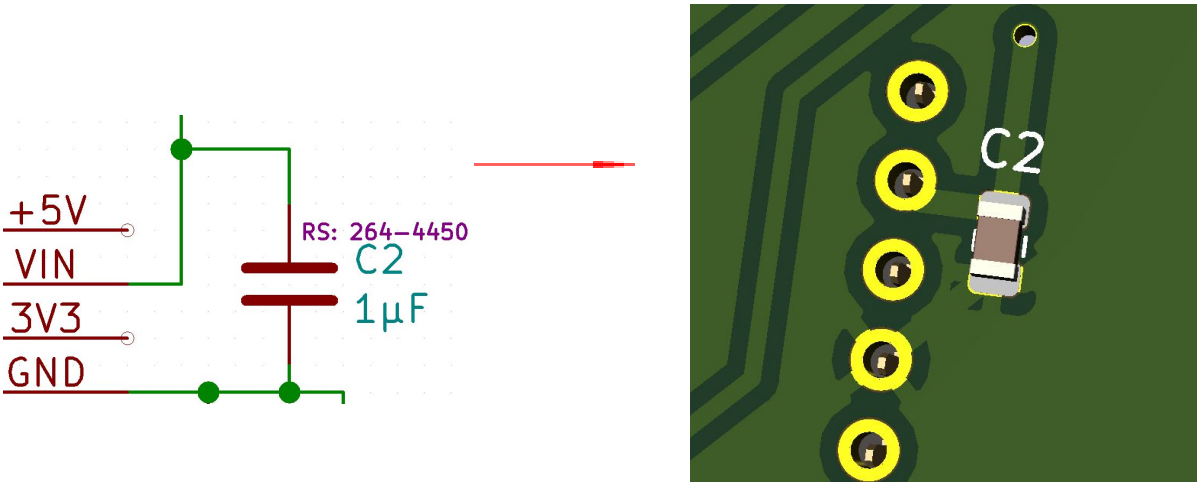
La tension qui arrive aux bornes du shield sera de 12V hors si nous voulons alimenter la MKR avec le shield, un conflit va avoir lieu.
Nous avons donc rajouté un convertisseur DC/DC 12-5V pour alimenter la MKR sous 5V.



Les points de test et les condensateurs de stabilisation



Nous avons rajouté des pics pour accéder plus facilement aux valeurs recherchées, par exemple si le 5V ou le 12V est bien transmit, le SDA SCL …
Ces pics sont des pics uniquement créé pour vérifier un niveau logique, une tension ou encore une masse.



Plusieurs condensateurs ont été rajouté au schéma structurel, ils servent à stabiliser une alimentation électrique (il se décharge lors des chutes de tension et se charge lors des pics de tension)

Dans le routage du shield, les condensateurs seront le plus rapproché possible des alimentations électrique pour être plus efficace dans leur fonction.

Explicatif du schéma structurel

La partie Dimmer, Relais ainsi que le convertisseur DC/DC ont été expliqués auparavant mais d'autres figures sur le schéma pour la première fois ;

• La partie MKR WAN 1300 désigne les broches femelles qui viendront s'insérer sur la carte MKR.

• La partie CLOCK désigne une horloge en temps réel expliqué par Christopher Paquier. Cette horloge va servir a gradué la lumière en fonction du temps, le lampadaire peut devenir autonome.

• La partie Capteur, les 4 sous parties en bas du schéma structurel désigne les composants issus des différentes parties capteurs, exemple, le capteur de courant.

Schéma de routage via le logiciel Kicad

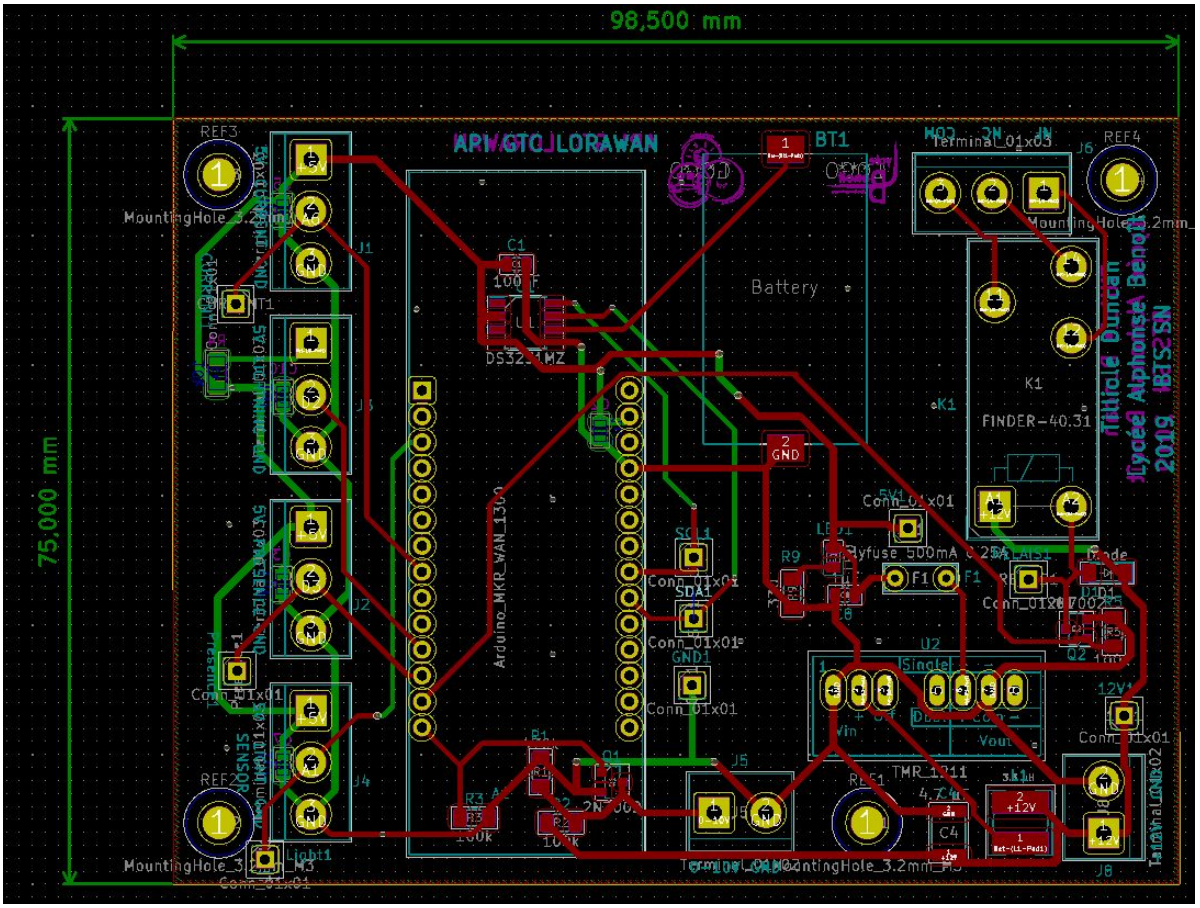
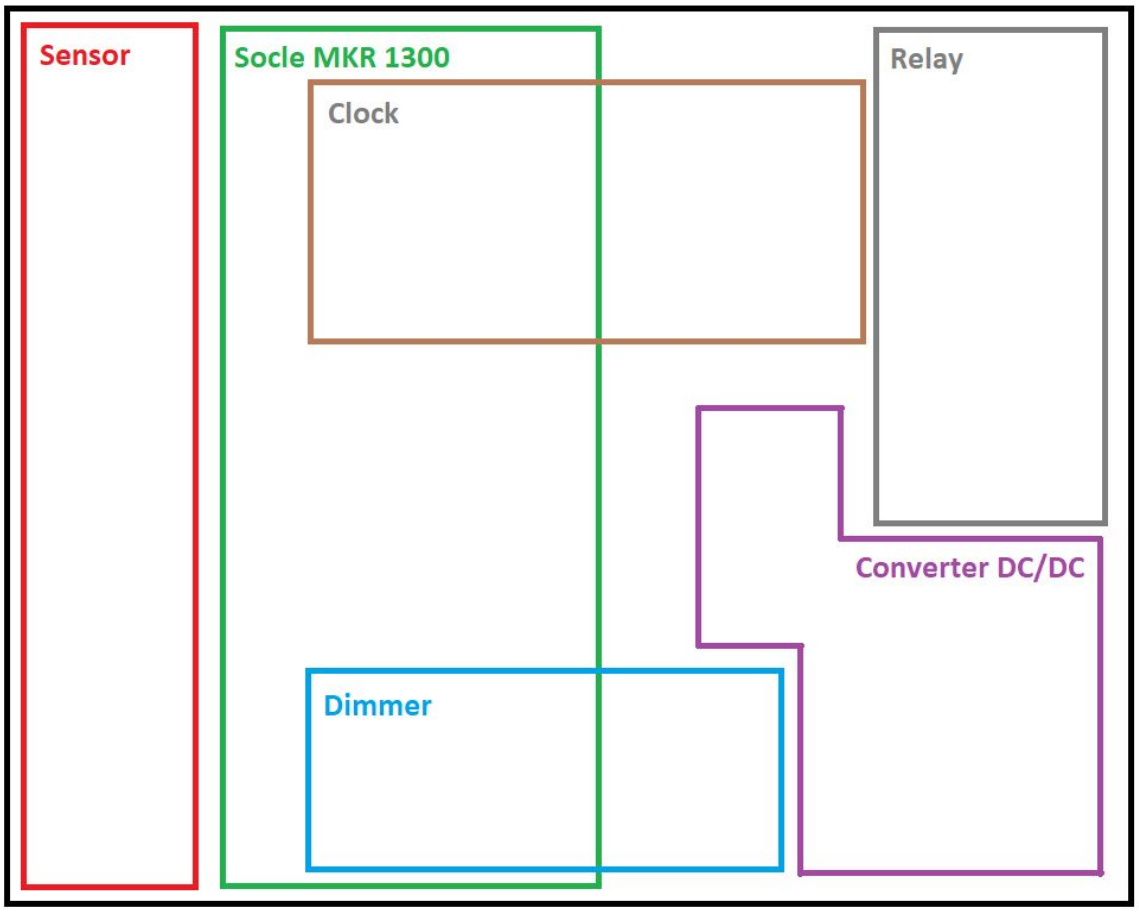
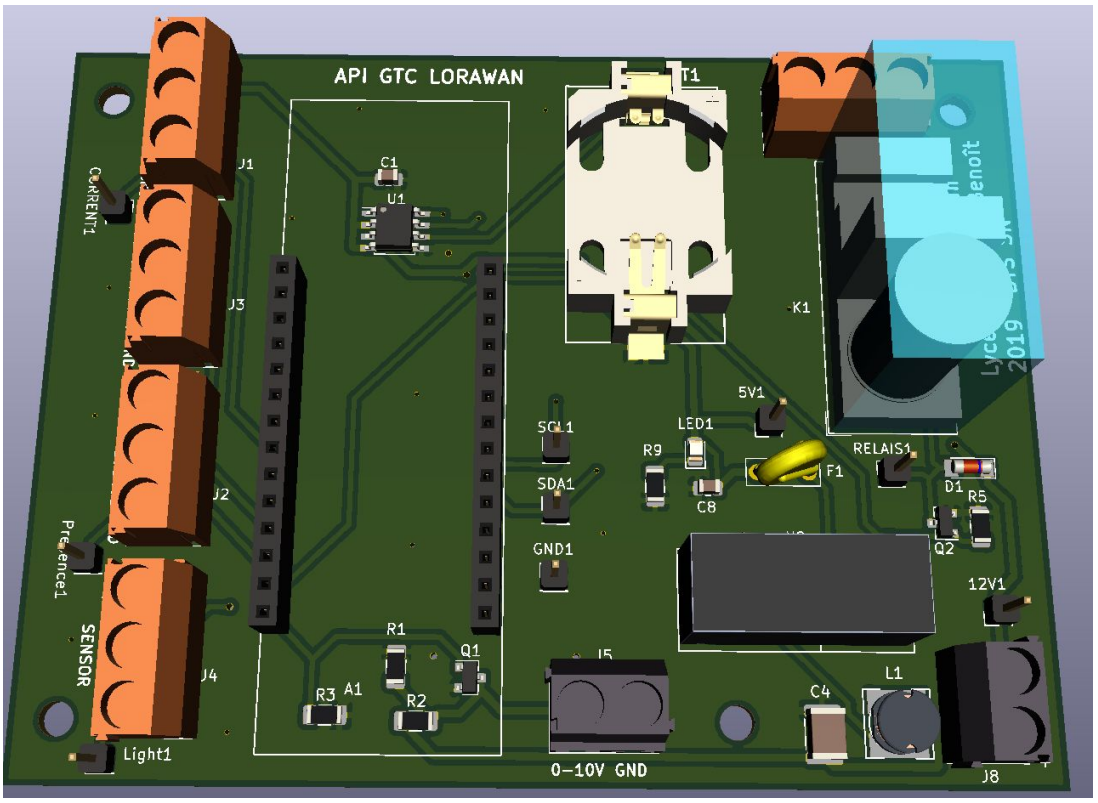


Schéma explicatif du routage de la carte

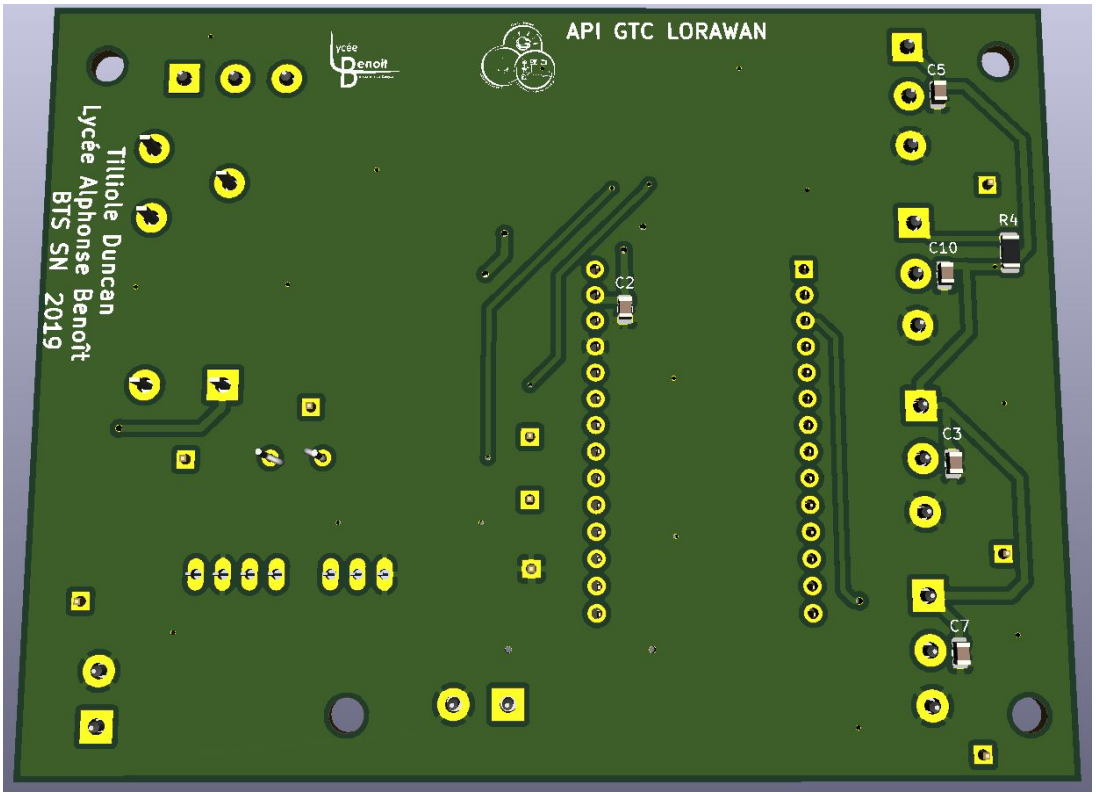


La disposition des composants n'était pas spécifiée dans le cahier des charges, j'ai donc choisie ces différents emplacements en respectant les besoins de certains composants, comme par exemple pour les condensateurs vu précédemment.

Schéma 3D du shield



Vue de l'avant



Vue de l'arriere

Introduction : API CARTE

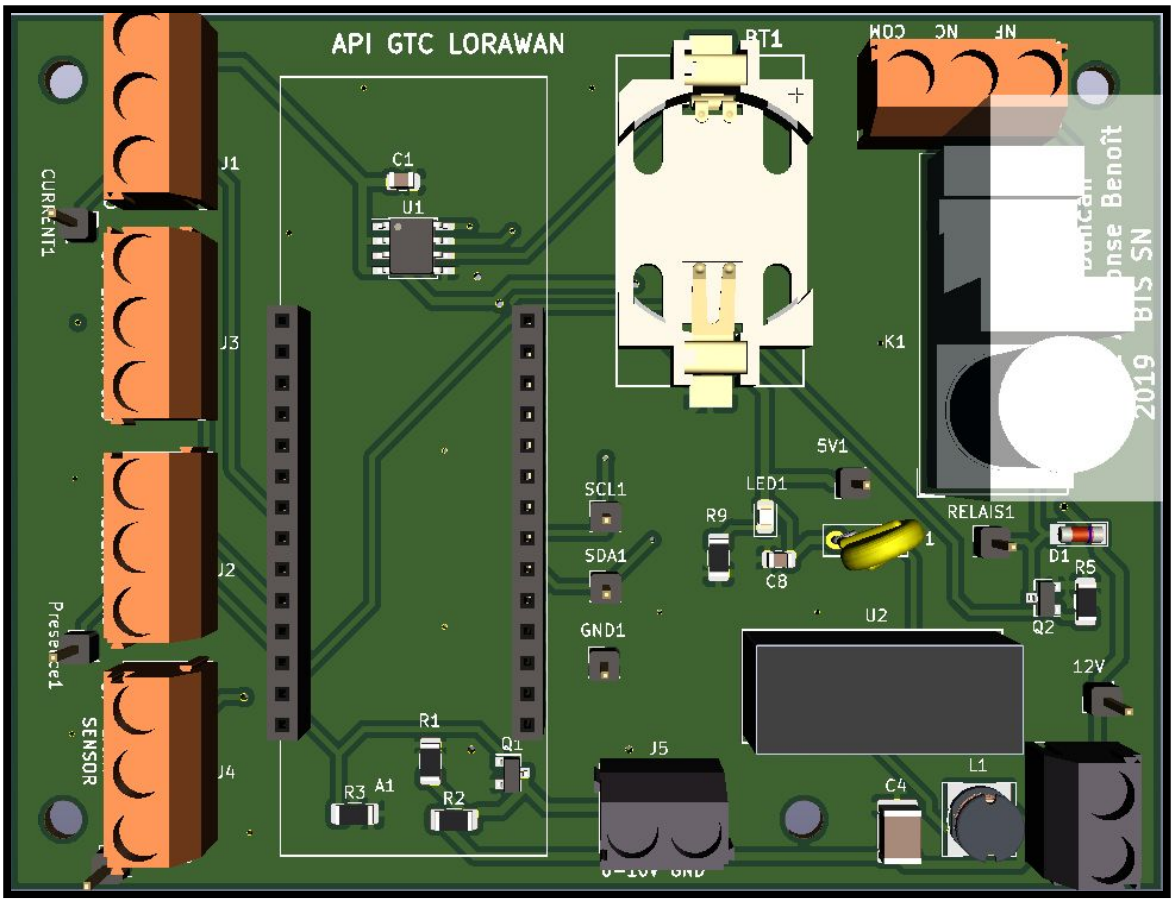


Table des matières

1/ Partie Soudure

• 41 éléments à souder •

1.1/ Nomenclature des composants

Référence	Valeurs	Code commande	Nombre
A1	Arduino_MKR_WAN_1300	Farnell:2851778	1
BT1	Battery	Farnell:1704232	1
C1	100nF	RS: 264-4422	1
C4	4.7 µF	RS : 723-6580	1
> C2 - C10	1µF	RS: 264-4450	8
CURRENT1	Conn_01x01	Gotronic: 08000	1
D1	Diode	RS :670-8823	1
F1	Polyfuse 500mA 0.25A	RS : 517-6607	1
> J1 - J6	Terminal_01x03		5
> J5, J8	Terminal_01x02		2
> J12, J13	Conn_01x01	Gotronic: 08000	2
K1	FINDER-40.31	RS: 351-601	1
L1	3.3 µH	RS: 745-1174	1
LED1	LED		1
Light1	Conn_01x01	Gotronic: 08000	1
Presence1	Conn_01x01	Gotronic: 08000	1
> Q1, Q2	2N7002	RS: 753-3134	2
R1	470k	RS : 901-3746	1
R4	10k	RS : 125-1192	1
> R2 - R5	100k	RS : 901-3746	3
R9	330	RS : 679-2049	1
RELAIS1	Conn_01x01	Gotronic: 08000	1
SCL1	Conn_01x01	Gotronic: 08000	1
SDA1	Conn_01x01	Gotronic: 08000	1
U1	DS3231MZ	RS : 778-1484	1
U2	TMR_1211	RS: 433-8258	1

1.2/ Ordre de câblage des composants CMS côté TOP

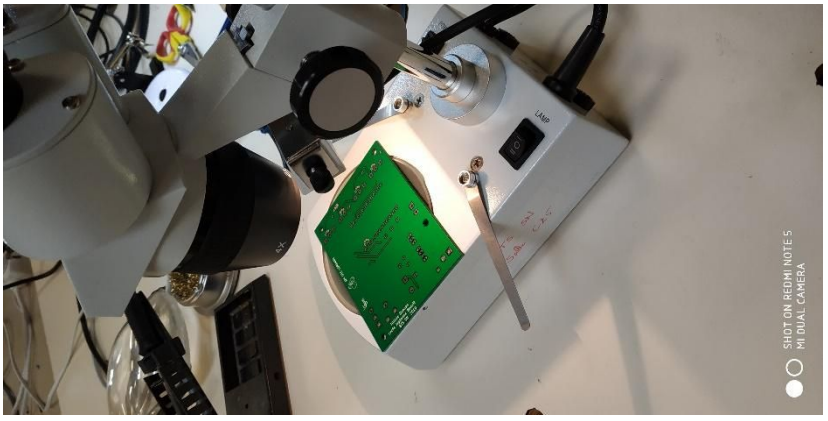
• Les résistances -> **R1 ; R2 ; R3 ; R5 ; R9**
• Les condensateurs -> **C4 ; C8 ; C1** et la bobine **L1**
• Les transistors **Q1** et **Q2**
• La diode **D1**, la led **LED1** ainsi que l'horloge **U1**
• La batterie **BT1**

Nous avons fait une soudure à la pâte à braser.

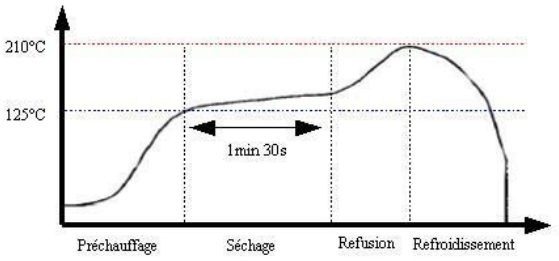
Dans un premier temps j'ai positionné la carte de telle sorte que l'on peut voir les empreintes des cms quand je replie le stencil sur elle-même.

Ensuite j'ai ajouté de la pâte à braser sur les empreintes que l'on voit grâce au stencil et j'ai raclé pour avoir la même hauteur de pâte sur chaque empreinte.

J'ai ensuite pu placer tous les CMS sur leurs empreintes respectives.



Après avoir mis les composants j'ai regardé à la loupe s'ils étaient bien placés et s'il y avait assez de pâte à braser.



Après avoir vérifié les soudures à la loupe j'ai mis ma carte dans un four à refusion. On peut voir la courbe de chauffe du four à refusion, il y a un temps destiné au préchauffage, au séchage, à la refusion et au refroidissement de la carte.
Après avoir soudé les composants côté TOP, j'ai reproduit toutes ces étapes côté BOTTOM

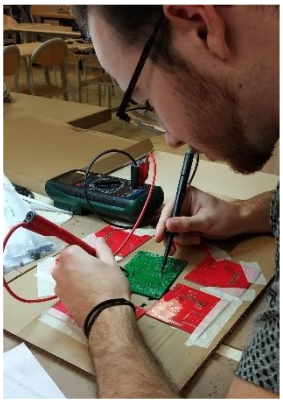
1.3/ Ordre de câblage des composants CMS côté BOTTOM

• Les condensateurs sur la face arrière -> **C2 ; C5 ; C10 ; C3 ; C7** ainsi que la résistance **R4**

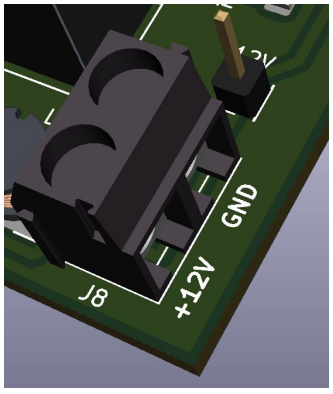
1.4/ Ordre de câblage des composants traversants côté TOP

• Les points de test -> **12V ; 5V1 ; GND1 ; SDA1 ; SCL1 ; PRESENCE1 ; CURRENT1 ; LIGHT1 ; RELAIS1**
• Le relais **K1**, le convertisseur **U2** et le polyfuse **F1**,
• Les borniers -> **J1 ; J2 ; J3 ; J4 ; J5 ; J6 ; J8**
• Le socle **A1** destiné à la MKR 1300 Wan

2/ Les alimentations

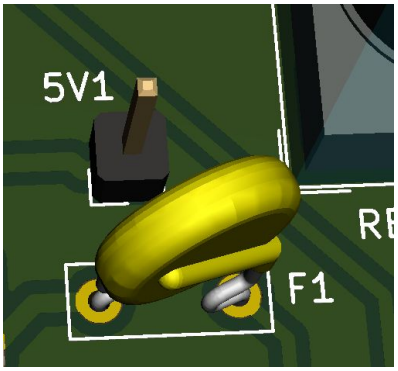


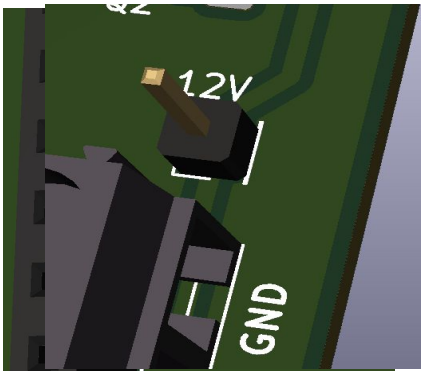
• Vérifier la continuité de toutes les soudures au préalable à l'aide d'un multimètre.



• Alimenter la carte en 12V par le bornier J8.

• Vérifier les points de test 12V & 5V & GND et entrée V_{in} à l'aide d'un multimètre.



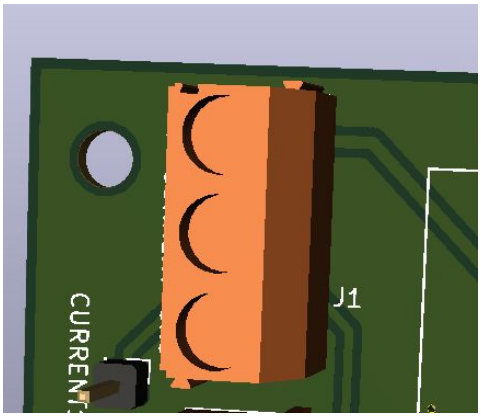


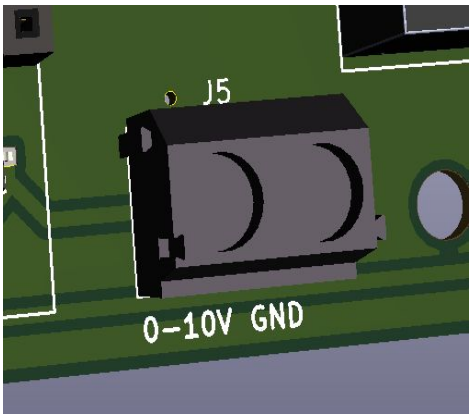
• Mettre la carte Arduino dans le socle MKR.

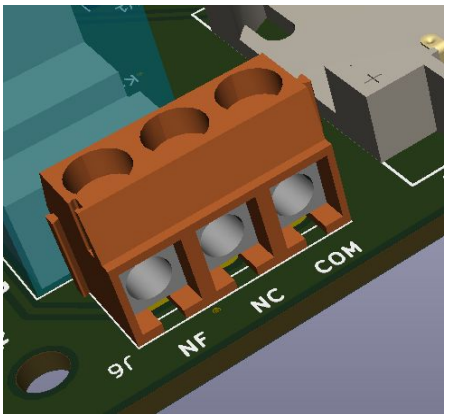
3/ La programmation

*Pour faire cette partie vous aurez besoin du document « **Arduino_Projet_TILLIOLE** »*

• Brancher les borniers J1 & J5 & J6 au lampadaire ainsi qu'au capteur de courant.







3.1/ Partie Courant

si l'erreur provient du programme ou de la carte.

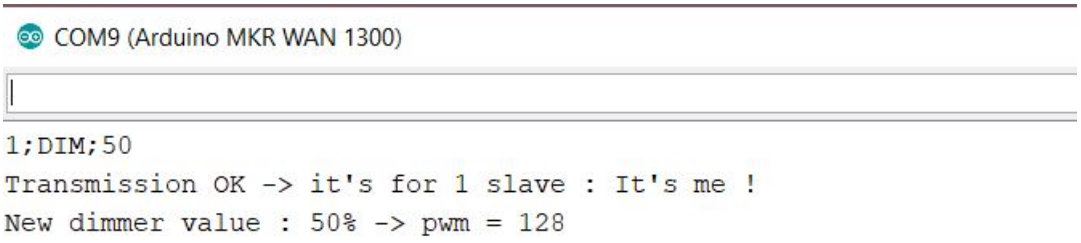
• Téléverser le programme « **prog_Courant** ».

• Allumer le lampadaire
• Lancer le moniteur série et écrire « AMP ». Vérifier si la valeur reçue correspond bien au courant qui circule dans le lampadaire.

Si le programme ne fonctionne pas, brancher un multimètre sur le point de test « CURRENT » pour savoir

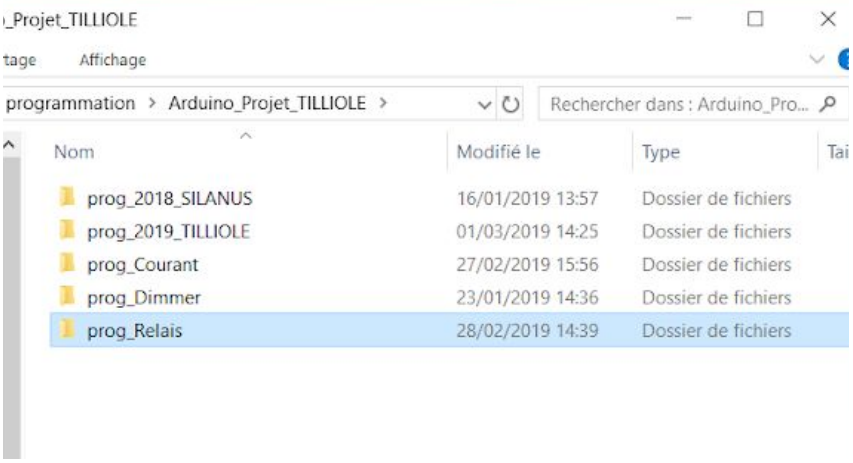
3.2/ Partie Graduateur

• Téléverser le programme « **prog_Dimmer** ».
• Lancer le moniteur série et écrire « 1;DIM;50 » comme montrer ci-dessous.



•Vérifier si la luminosité du lampadaire est descendue de moitié.
Le programme du graduateur est en pourcentage, pour faire varier ce pourcentage, il faut changer le texte souligner : « 1;DIM;50 ».

3.3/ Partie Relais



• Téléverser le programme « **prog_Relais** ».
• Lancer le moniteur série et écrire « ON » ou « OFF ».

Si le programme ne fonctionne pas, brancher un multimètre sur le point de test « RELAIS » pour savoir si l'erreur provient du programme ou de la carte.

3.4/ Partie Horloge

• Téléverser le programme « **prog_Horloge** ».
• Lancer le moniteur série et voir si la date et l'heure affichées sont corrects.

Pour l'horloge, le programme va s'effectué automatiquement, il n'y a pas besoin d'écrire quelque chose.

3.5/ Partie LoRa

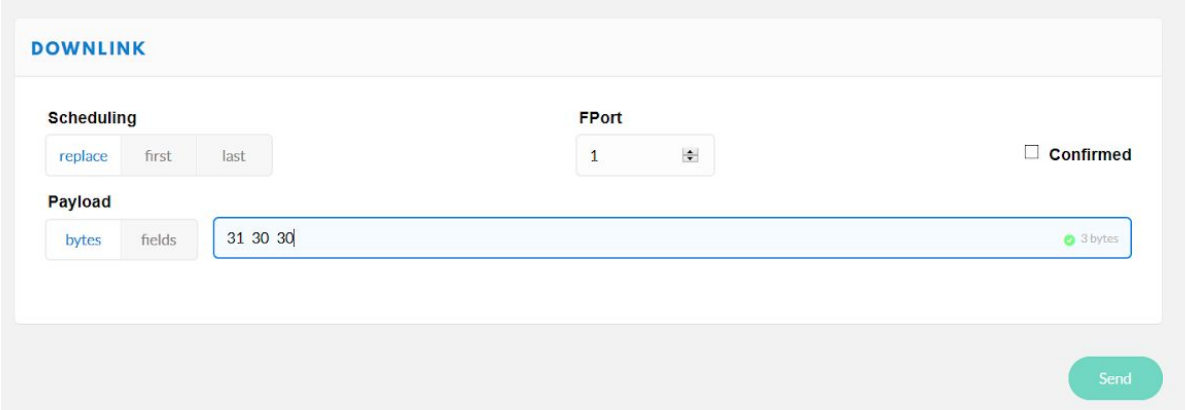
• Téléverser le programme « **prog_2019_TILLIOLE** ».
• Se connecter à The Things Network grâce au navigateur de recherche.

Récapitulatif du mode de communication :
Pour que TTN communique avec la carte il faut que la carte envoie une donnée. Si aucune donnée n'est envoyée de la carte à TTN, l'ordre de TTN ne pourra être transmis.

---

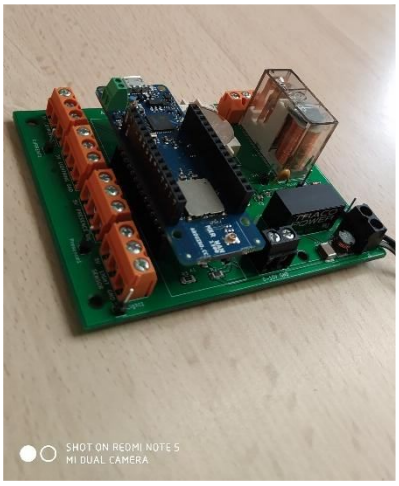
*Ouvrir le fichier « Protocole LoRa » fournie dans le document « **Arduino_Projet_TILLIOLE »***

• Dans le fichier « Protocole LoRa » se trouve le code ascii correspondant aux ordres pouvant être envoyés par TTN, tester tous les ordres un à un pour vérifier la compatibilité avec la carte.



L'ordre doit être envoyé un premier. Ensuite il faut agir sur le moniteur série du programme en envoyer un message à TTN pour recevoir l'ordre.

Création de plusieurs API cartes



Après avoir soudé tous les composants de ma carte je l'ai testé via un multimètre et j'ai pu en conclure qu'aucune anomalie n'a été détectée.
Après ça j'ai téléversé tous les programmes un par un et ma carte a réussi à contrôler toutes les fonctions demandées.

Je peux en conclure que la carte fille demandé dans le cahier des charges a été construite et validé.

Cette carte va être envoyée à l'entreprise API. Cependant je dois en construire une pour le lycée et j'ai pris la décision d'en construire une troisième pour mes intérêts personnels (entretien d'embauche, partie pratique…).

Conclusion

Ce projet m'a permis d'apprendre de nouvelles choses, tels que la création de carte électroniques, lire des schémas structurels, connaître de nouveaux composants au sein d'un système (relais, transistors, résistances de pull up….) mais aussi de consolider mes connaissances en programmation embarqués, de travailler en équipe, de rédiger des documents administratifs…

Pour la suite du projet, il me reste à construire 2 cartes et faire un programme qui va permettre l'automatisation du lampadaire. Pour l'instant le programme final permet de commander tous les ordres à partir de TTN mais un message doit être envoyé de la carte à TTN pour que ce principe fonctionne, en d'autres termes, je vais programmer l'envoie de la mesure de courant vers TTN toutes les 15 minutes.

ANNEXE

Fiche de Réunion :

10/01/19
Durée: 1 heure

Travail réalisé	mettre au point le dossier et le diapo, explication du drive, recherche de logo, gantt, début élaboration protocole
Travail à faire	améliorer les fiches de réunion, trouver un logo définitif, déterminer le protocole, finir le gantt

24/01/19
Durée: 25 minutes

Travail réalisé	trouver un logo, finir le gantt, TTN commencé, horloge partiel (marche, à adapter), dimmer partiel (LoRa à faire), installation en cours MQTT/Qt, installation en cours InfluxDB/Telegraf/Grafana
Travail à faire	diagrammes de séquences/bloc interne, préparer diapo/dossier, établir le protocole, installation MQTT/Qt/InfluxDB/Telegraf/Grafana

07/02/19
Durée: 30 minutes

Travail réalisé durant la réunion	mise au point sur le LoRa, correction d'erreurs
Travail à faire	Telegraf, communication entre lampadaire et TTN (LoRa)

08/02/19
durée: 30 minutes

Travail réalisé	test de communication entre TTN et la MKR (vice versa), définition du protocole de communication dans chaque sens (à faire valider par M. Silanus)
Travail à faire	mettre en place le protocole LoRa

ANNEXE

• **Fiche de rapport de lecture croisée.**		***Réf.*** **FRLC-XX**
Projet	***Auteur***	***Date***
API GTC EP LoRa	Tilliole Duncan	25/01/2019
Produit		***Nom du document***
Diagramme de séquence		diagramme Consommation d'un Segment
Lecteur(s)		***Contrôleur des corrections***
Paquier Christopher		Tilliole Duncan

Liste des erreurs détectées

-Erreur d'orthographe "The Thinks Network" -----> "The Things Network"

-Oublie du renvoie de la BDD à la GTC