

LoKy – OpenSource projet pour Linky avec LoRaWAN

Bonjour à tous,

Je suis [HO Vu Duy Bao](#) et je suis le stagiaire de M. [Jérôme FERRARI](#) au CNRS cette année – 2021. Aujourd'hui, je vous présente mon projet de fin d'études.

0. Introduction

Avant toute chose, ce projet est sous la licence CC BY-NC-SA 4.0

Ce projet est aussi supporté par les entités suivantes CNRS/UGA/G-INP – G2ELAB dans le cadre de l'IDEX EcoSesa et du futur observatoire pour la transition énergétique



Vous pourrez aussi récupérer tous les fichiers pour recréer votre LoKy (Code, PCB et STL) sur le gitlab du Gricad à l'adresse suivante:

<https://gricad-gitlab.univ-grenoble-alpes.fr/ferrarij/LoKy>

Si vous voulez participer activement à l'amélioration du projet mais aussi participer en tant que candidat aux expériences du laboratoire, n'hésitez pas à nous contacter soit par le biais du GitLab soit à l'adresse mail jerome.ferrari@g2elab.grenoble-inp.fr

Maintenant que les présentations sont faites, nous pouvons partir au coeur du sujet.

I. Fonctionnalités attendues

Actuellement, les données du linky accessibles via le portail d'Enedis ne donnent que les données des courbes de charge de la veille avec un pas de 30 min. Cependant, nous pouvons utiliser la prise TIC (Télé Information Client) pour les obtenir en temps réel. C'est pour cela que nous avons choisi de construire un système pouvant retranscrire ces données et les envoyer sur un serveur personnel et ainsi permettre aux utilisateurs ou aux organisations de mieux gérer leur consommation.

Pour ma part, j'étais en charge de construire la passerelle **LoKy** (Lora LinKy)- un *low-power* Lora end-node qui comprend les fonctions suivantes :

- Pouvoir s'alimenter de façon autonome depuis la prise TIC du Linky
- Pouvoir récupérer les informations transmises par le compteur **Linky** sur la sortie de *Télé-Information Client* ou **TIC**
- Pouvoir envoyer ces informations via **LoRa** vers un serveur d'application, auquel l'utilisateur pourra se connecter et visualiser ses données.

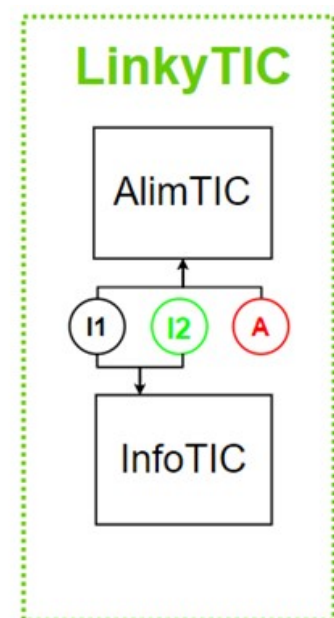
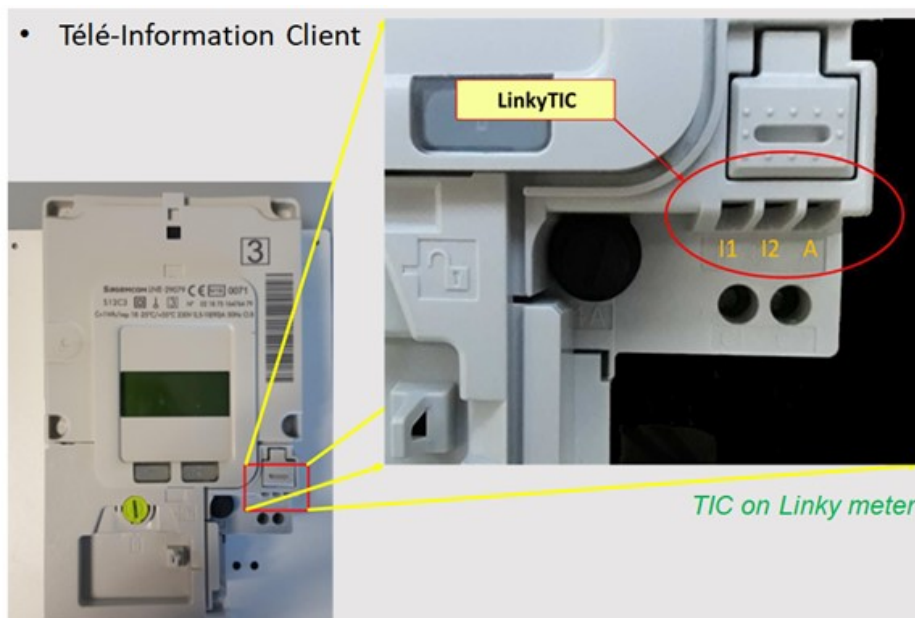
II. Linky et TIC

Tout d'abord, parlons du compteur Linky. Le compteur Linky pour avantage comparé aux anciens compteurs de fournir en plus de la sortie TIC une sortie de « puissance » afin d'alimenter un capteur de communication autonome.

Concernant les paramètres techniques du compteur Linky, vous pouvez les retrouver dans la [datasheet](#) officielle fournie par ENEDIS.

[Enedis-NOI-CPT_54E](#) [Download](#)

En ce qui concerne notre projet, l'essentiel à retenir est que le compteur Linky est équipé d'une sortie numérique de Télé-Information Client (ci-après appelée **TIC** ou **LinkyTIC**) qui dispose de 3 bornes **I1**, **I2** et **A** :



- **I1** : borne de commun
- **I2** : la borne d'information
- **A** : borne d'alimentation

a. Caractéristiques de l'alimentation TIC entre bornes A et I1

L'énergie fournie par le linky pour l'alimentation de module externe a les caractéristiques suivantes:

| Spécification | Niveaux |
|-------------------|--|
| Puissance fournie | ~130 mW |
| Tension | 6 Vrms \pm 10% à 50 kHz (12 V pic au maximum) |

Comme la puissance fournie est faible, nous avons décidé de nous orienter vers un arduino pro mini 3,3V couplé à un module LoRa RF95w pour effectuer la transmission de données car cette technologie consomme très peu d'énergie comparé à la technologie WiFi.

b. Circuit d'informations InfoTIC via les bornes I1 et I2

En ce qui concerne les données, la prise TIC diffuse en continu les paramètres de consommation électrique au fur et à mesure que la partie comptage les mets à jours. Cette sortie est asynchronous et les informations sont transmises en série sur la ligne.

Pour illustrer les informations contenues dans les trames, nous réutilisons certaines des images du blog <https://lucidar.me/fr/home-automation/linky-customer-tele-information/> qui pour ses travaux se sert du montage octocoupleur/mosfet inventé par Charles Hallard (<https://hallard.me/pitinfov12/>) que nous vous avons présenté dans notre article de 2019 (<https://miniprojets.net/index.php/2019/06/28/recuperer-les-donnees-de-son-compteur-linky/>)

Les signaux binaires sont transmis avec une porteuse modulée à 50kHz qui est configurée en 2 modes :

- **Mode historique** : le compteur permet de restituer des trames d'information au 1200 bauds.

Exemple en mode historique : **ADCO 021875164764 J**

| Exemple en mode historique de l'adresse d'identification du compteur | | | | | | |
|--|-----------|-----------|----------------|-----------|----------|-----------|
| LF (0x0A) | Etiquette | SP (0x20) | Donnée | SP (0x20) | Checksum | CR (0x0D) |
| | "ADCO" | " " | "021875164764" | " " | 'J' | |

- **Mode standard** : un nouveau mode qui est apparu sur les compteurs Linky avec la vitesse de 9600 bauds.

| Format d'un groupe d'information contenant une donnée horodatée | | | | | | | | |
|---|-----------|-----------|----------|-----------|--------|-----------|----------|-----------|
| LF (0x0A) | Etiquette | HT (0x09) | Horodate | HT (0x09) | Donnée | HT (0x09) | Checksum | CR (0x0D) |

Information avec l'horodate

| Format d'un groupe d'information contenant une donnée non horodatée | | | | | | |
|---|-----------|-----------|--------|-----------|----------|-----------|
| LF (0x0A) | Etiquette | HT (0x09) | Donnée | HT (0x09) | Checksum | CR (0x0D) |

Information sans l'horodate

En fonction de cela il faudra modifier le Baudrate de communication. Actuellement le firmware du LoKy est en 1200 bauds.

Maintenant que nous avons les données principales pour la partie Linky, nous allons passer à l'explication de la partie transmission via LoRa.

III. Présentation de LoRa et documentations utiles

Ici, nous allons juste effleurer le principe du LoRa et LoRaWAN mais vous pouvez aussi relire nos précédents articles ou lire ces excellents cours afin de parfaire vos connaissances :

- Article du Journal du net : [LoRa : comment fonctionne le réseau...](#)
- Cours de Sylvain Montagny : [LoRa et LoRaWAN pour l'Internet des Objets](#) – [Sylvain MONTAGNY](#)

Qu'est-ce que le LoRa ?



LoRa est une combinaison de deux concepts : **Long-Range spread spectrum modulation** (couche physique) et **LoRaWAN** (protocole réseau).

LoRaWAN – protocole réseau

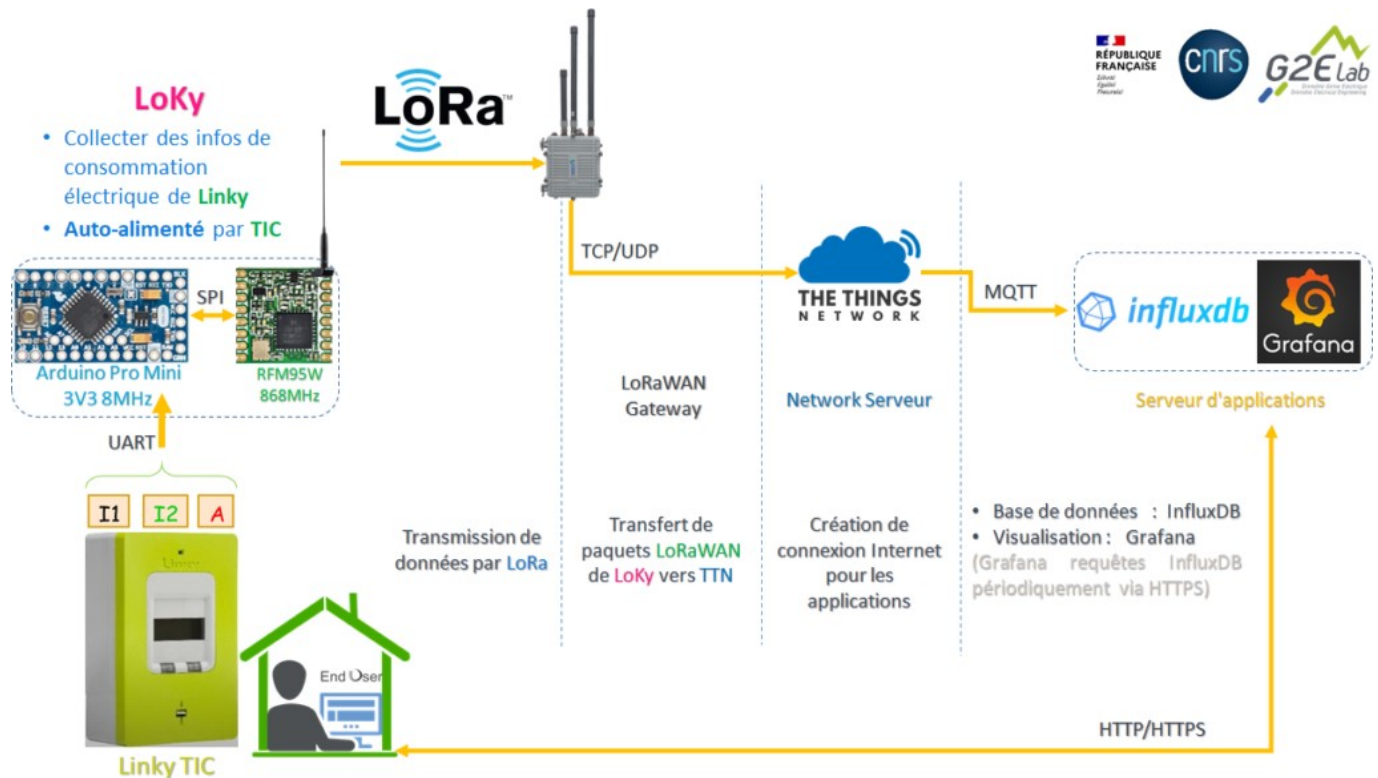
Défini par IBM & Actility avant d'être ouvert par la LoRaWAN-Alliance. Ce protocole fournit un réseau de type cellulaire (topologie de grande-étoile) avec ces paramètres :

- Spreading Factor (SF) – Programmable SF : 7, 8, 9, 10, 11, 12. Plus le SF est élevé, il y a plus d'informations transmises par bit, donc un gain de traitement plus élevé
- Bandwidth (BW) : Programmable signal BW : 125 kHz, 250 kHz, 500 kHz. Pour une SF, un BW plus étroit est une sensibilité de réception accrue ; cependant, augmentation du ToA
- Fréquence Sub-GHz sans licence : en France (Europe) est EU 863-870 MHz.

LoRa est fortement utile car en effet LoRaWAN est un protocole de télécommunication permettant une communication à bas débit pour des objets à faible consommation électrique communiquant selon la technologie LoRa et connectés à Internet grâce à de multiples passerelles.

IV. Architecture réseau pour le projet LoKy

Afin de mieux comprendre les différentes strates pour l'envoi des données, je vous propose ci-dessous une visualisation de l'architecture du projet depuis le Linky jusqu'à la visualisation des données sur un dashboard Grafana :



Pour simplifier, le LoKy décodera les données du compteur Linky et les transmettra par LoRa aux Gateways. Ensuite, les gateways transmettront les paquets aux serveurs TheThingsNetwork (TTN).

Depuis TTN, nous pouvons récupérer les données grâce à des API puis de les enregistrer dans InfluxDB via MQTT afin de les afficher sur Grafana. Nous avons expliqué cette architecture dans l'article <https://miniprojets.net/index.php/2019/07/23/monitoring-dune-serre-du-capteur-jusquau-serveur/>

V. Logiciels et composants utilisés pour LoKy

a. Outils et Logiciels

Dans ce projet, les logiciels et les outils suivants ont été utilisés:

- KiCad : [v5.1.5](#)
- Arduino : [IDE](#)
- Node-RED : [instruction](#)
- InfluxDB : [v1.8](#)
- Grafana : [v6.6](#)

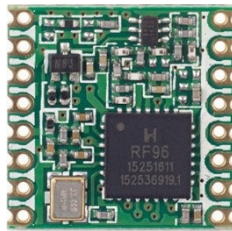
Voici les principaux composants de **LoKy** :

b. Composants

Arduino Pro Mini 3.3V 8MHz



Module LoRa RFM95W



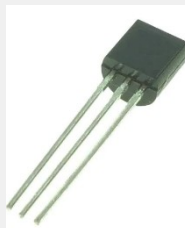
Optocoupleur PC814



Super Condensateur 5V 0.47F



1 regulateur MCP1702-4002E



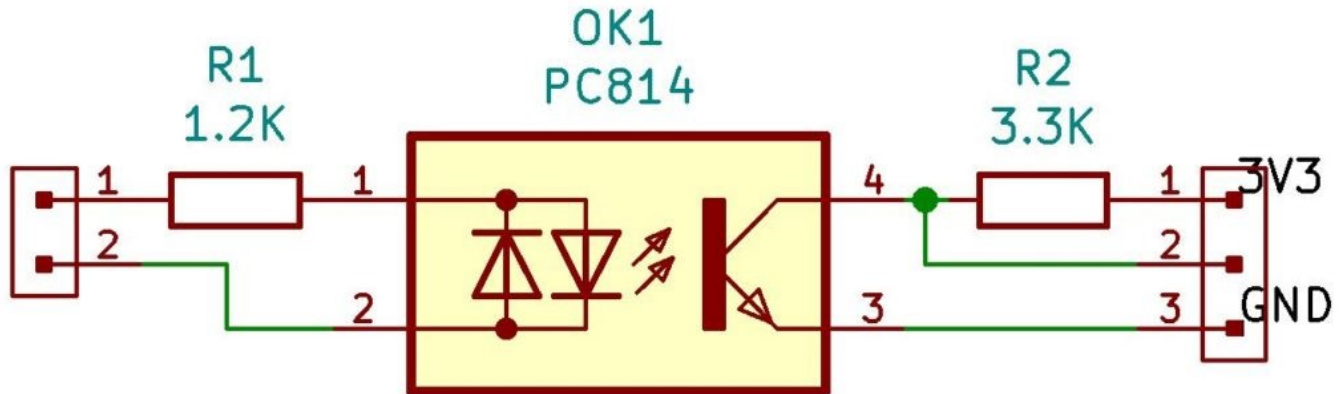
... et bien sûr d'autres composants communs comme les résistances, les condensateurs, les diodes que vous pouvez trouver dans le schématique.

Tous les schémas dont vous avez besoin seront disponibles sur GitLab de LoKy.

VI. Prototypages des différents étages électroniques

a. Le décodage d'InfoTIC

Avec toutes les spécifications données par la datasheet, on doit démoduler les signaux ASK de InfoTIC. Le moyen le plus efficace est de profiter d'un optocoupleur pour effectuer ce travail :



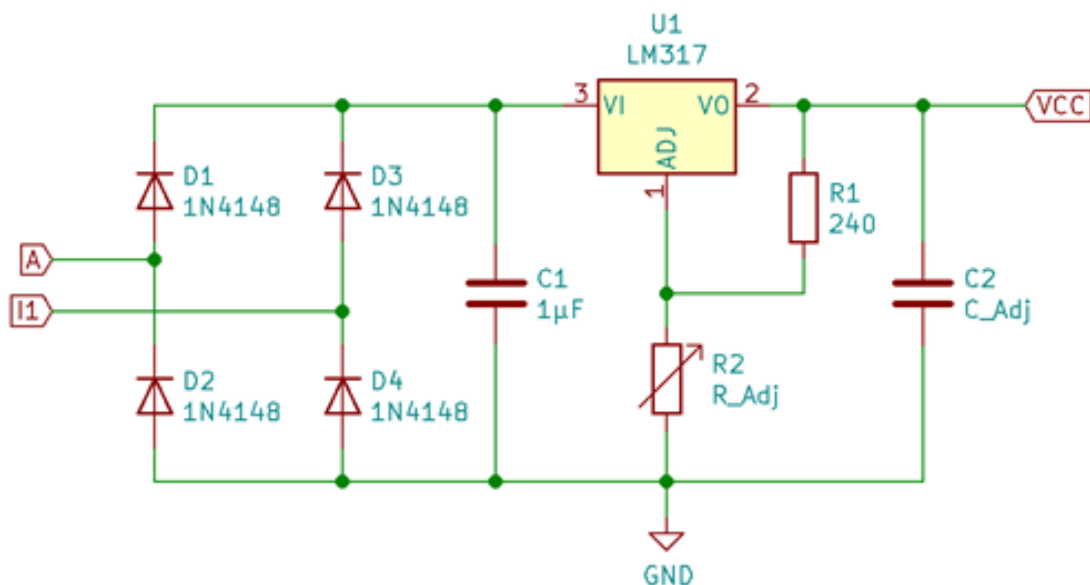
Ce schématique est familier à quiconque a une fois travaillé avec le **TIC** que ce soit le linky ou la génération précédente. Pour les personnes peu familières avec ce moyen de démodulation, vous pouvez en savoir plus à ce sujet dans cet [article](#) de Charles.

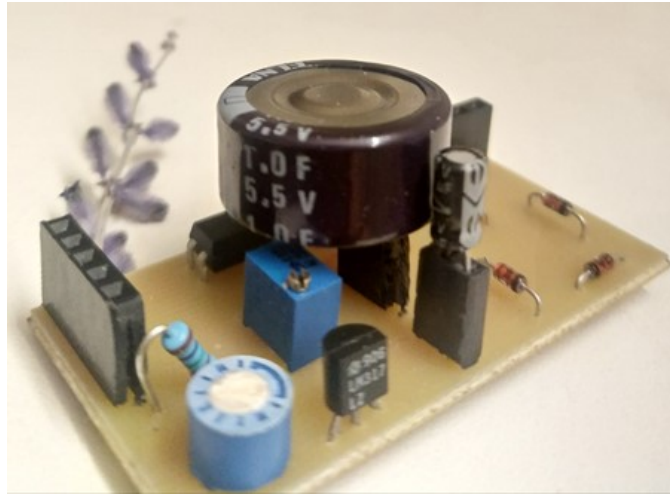
b. Le stockage de l'énergie avec AlimTIC

Cette partie de LoKy est améliorée par rapport à l'ancien article <https://miniprojets.net/index.php/2019/06/28/recuperer-les-donnees-de-son-compteur-linky/> et aussi lors de mon ancien stage non seulement dans les performances mais aussi dans la dimension.

Dans la première version de Aymeric, un simple 7805 avait été testé sans succès.

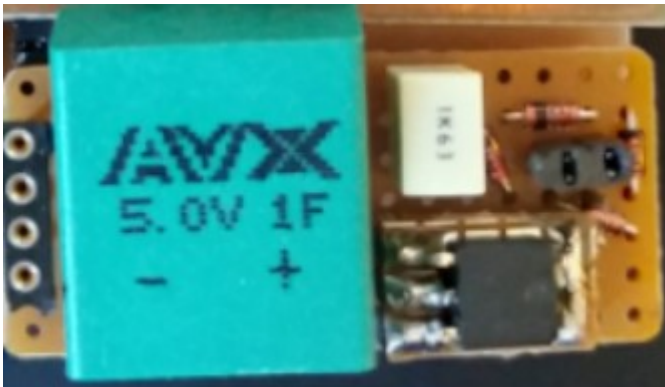
Dans la seconde version, j'avais utilisé un régulateur réglable LM317





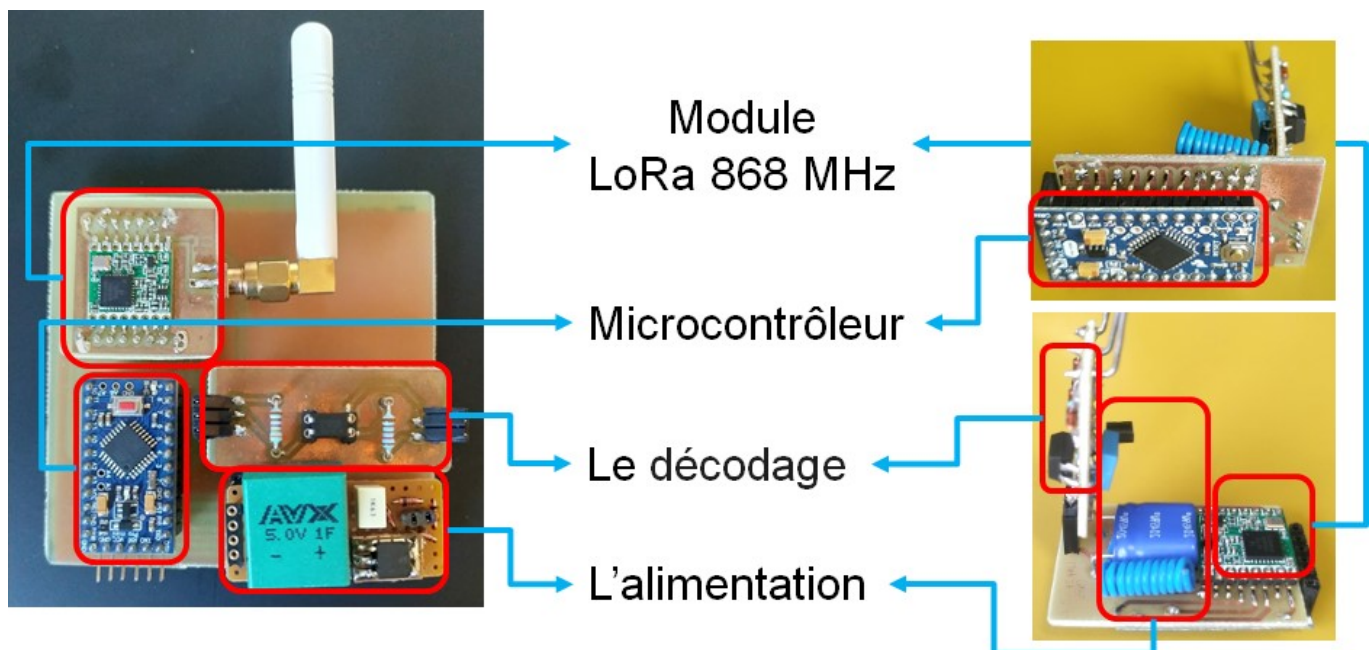
Après les tests, nous avons décidé d'utiliser un régulateur fixe 3.5V et un super condensateur 5V 0.47~1 Farad.*

* **Note** : La **résistance série équivalente** ou **ESR** du super condensateur doit être inférieure à 300mOhms.

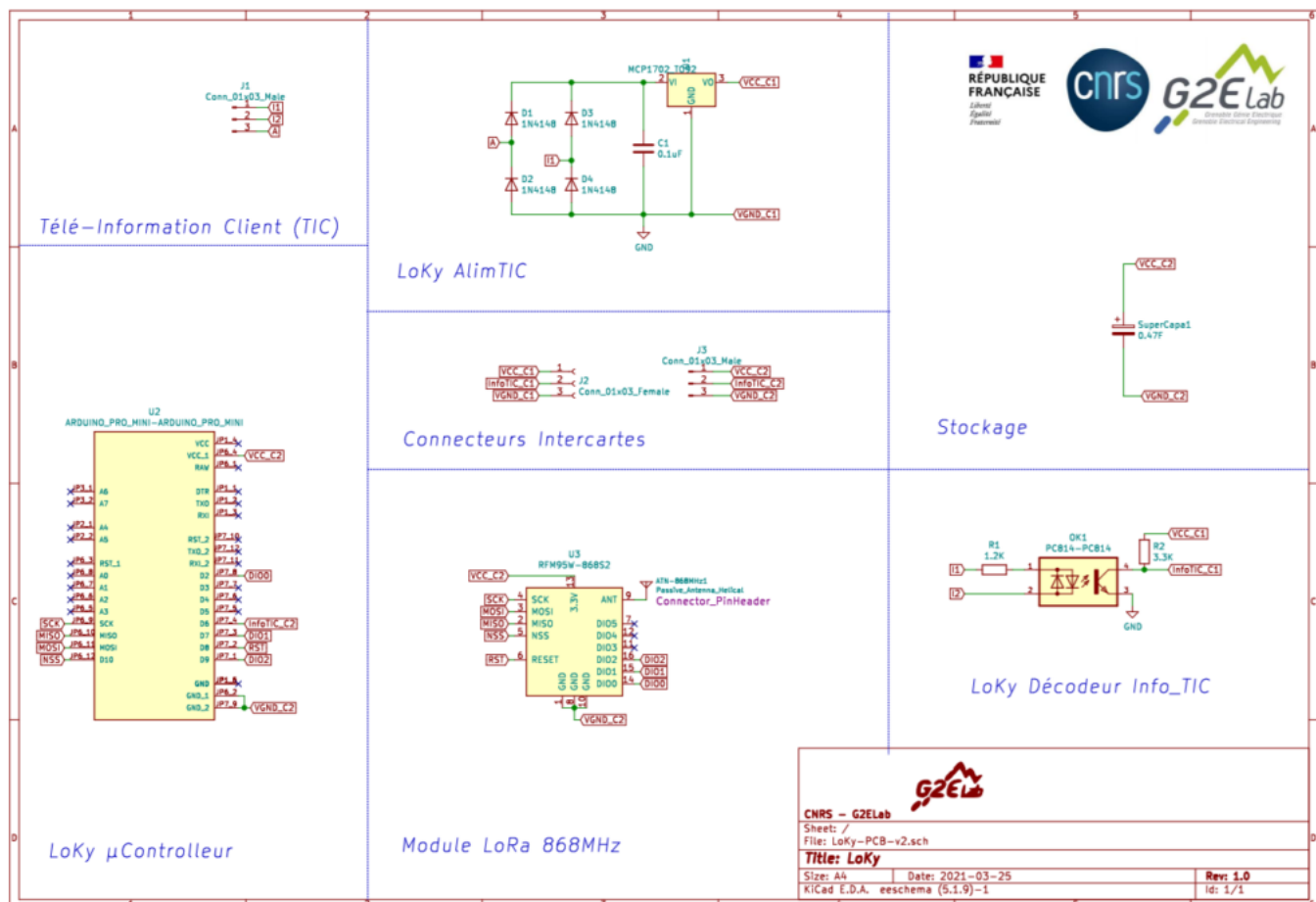


Ci-dessus, vous pouvez voir les version 2.1 et 2.2 de la partie stockage du LoKy.

Voici les prototypes de la carte de débogage et la version courant de LoKy à travers les différentes étapes du projet :

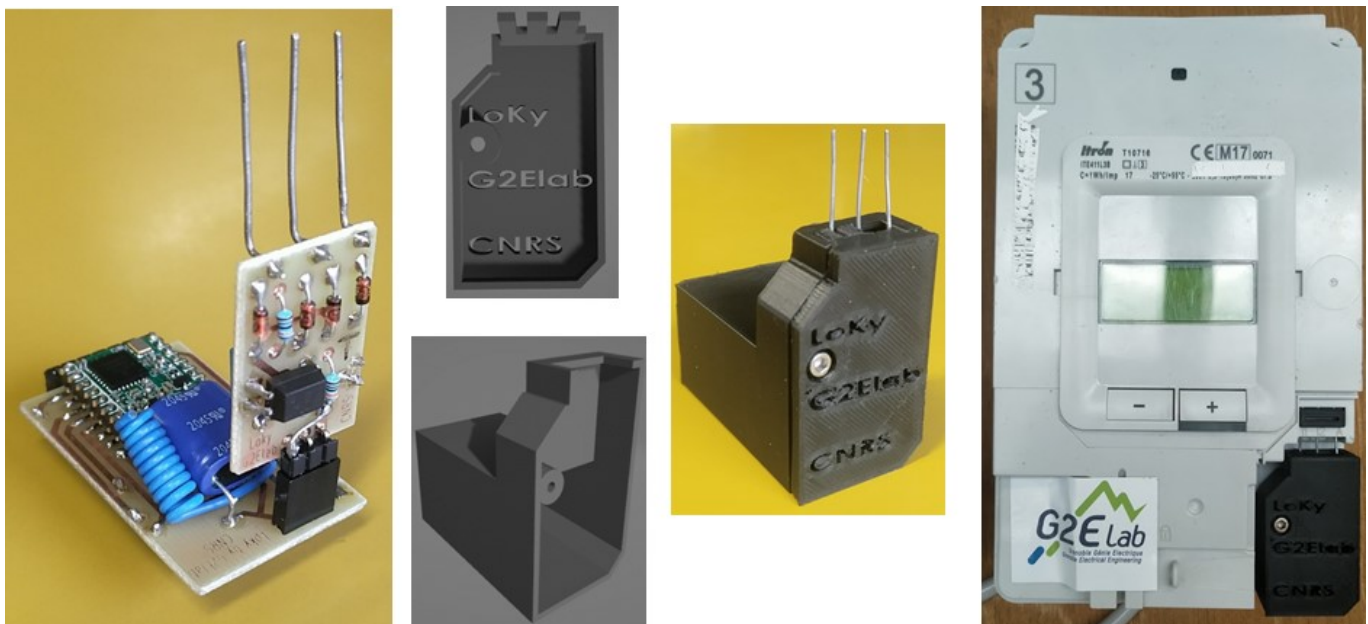


c. Le schématique final retenu



d. Intégration finale

Avec l'aide de Jérôme et le redesign des parties électroniques, le boîtier du LoKy s'intègre parfaitement dans l'espace TIC du Linky, comme vous pouvez le voir ci-dessous :



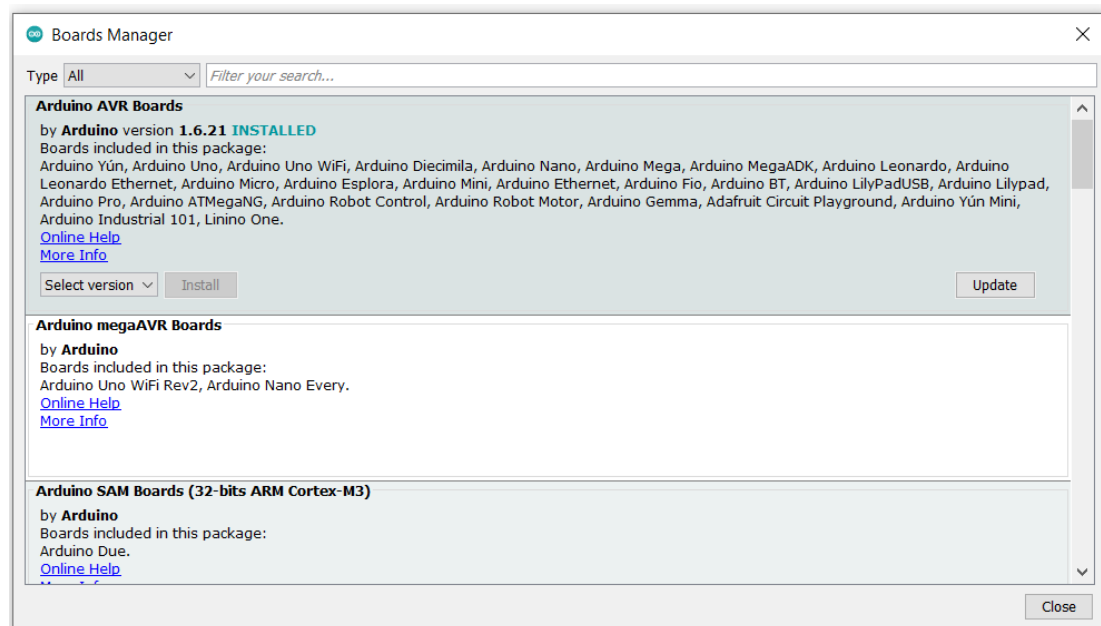
VII. Firmware de LoKy

Maintenant, nous passons à la partie la plus importante qui permet de coder le lien entre le linkyTIC et le réseau LoRaWAN.

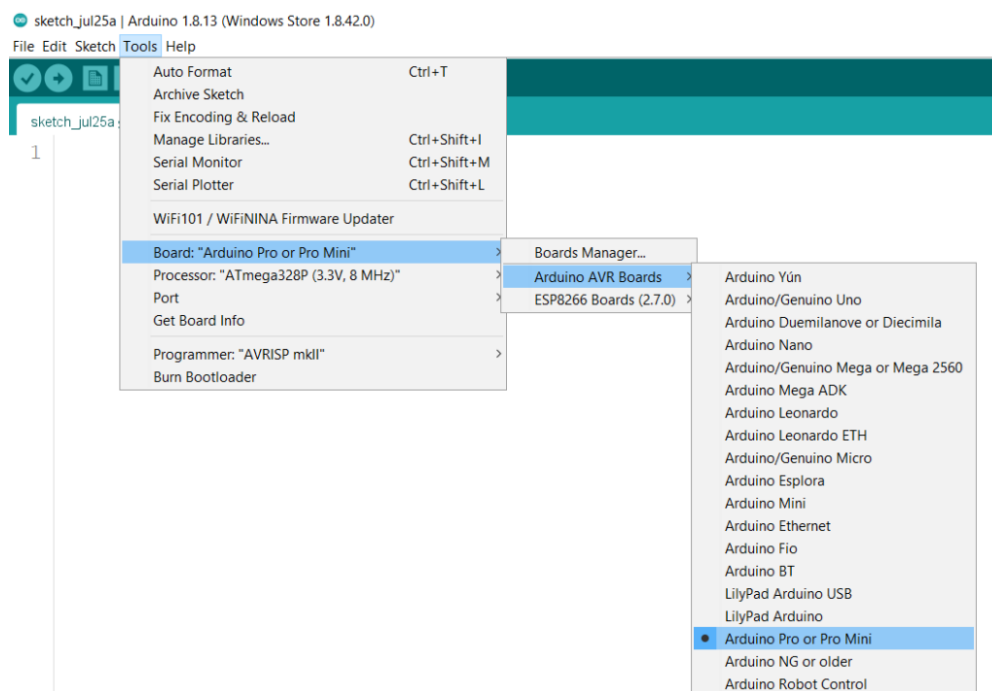
a. Configuration Arduino IDE

Pour nous assurer que tout fonctionne bien, nous avons besoin de configurer Arduino IDE de la façon suivante

- Installer la version 1.6.21 de Arduino AVR Boards dans Boards Manager (les versions plus récentes ont des bugs inconnus pour la bibliothèque de LoRa)



- Board setup :



- Sélectionnez le processeur: «ATmega328P (3,3 V, 8 MHz)» et le port qui se connecte à votre ProMini
- [Installez les bibliothèques](#) pour Arduino IDE.

* Bibliothèques du projet

Pour un souci de continuité et de stabilité, les bibliothèques sont aussi disponibles sur le GitLab du LoKy, mais vous pouvez les retrouver sur leur sites officiels :

- Arduino-LMiC : [GitHub](#) (Lire la suite à récapitulatif des bibliothèques LoRaWAN [\[HowTo\]](#))
- Low-Power pour l'Arduino : [Rocketscream](#)

b. Configuration de TTN ver.3

Dans cette section, vous trouverez les instructions pour créer des applications et des devices sur TTN version 3 (TTS) pour cela je vous conseille cette vidéo en attendant un futur article.

... ou ce lien <https://www.thethingsindustries.com/docs/devices/adding-devices/>.

Une fois cela fait, allez dans l'onglet « End devices » et ajustez les clés de device au format suivant

Activation information

| | |
|-------------|--|
| AppEUI | 0x73, 0x19, 0x32, 0x00, 0x03, 0x3... lsb <> 📄 |
| DevEUI | 0x68, 0x07, 0x32, 0x86, 0x03, 0x3... lsb <> 📄 |
| Root key ID | n/a |
| AppKey | 0xF8, 0xDD, 0x8D, 0x74, 0x21... msb <> 📄 🔍 |
| NwkKey | n/a |

c. Configuration firmware

Une fois que vous aurez récupéré le code sur notre Gitlab <https://gricad-gitlab.univ-grenoble-alpes.fr/ferrarij/LoKy>

Allez dans le dossier firmware et prenez la dernière version (à l'heure actuelle, c'est la V4)

Avec l'IDE Arduino, ouvrez le fichier LoKy-main\LoKy_Firmware\src\V4\loky_node-1_v4.ino et le fichier LoKy-main\LoKy_Firmware\src\V4\Device_Keys.h

- Nous devons d'abord remplacer les clés (que nous avons généré dans la section VII.b) dans le fichier « Device_Keys.h ».
- Attention! Dans la dernière version du firmware l'étape qui suit n'est plus nécessaire car les 2 options sont inclusent
- Selon l'option tarifaire de votre Linky, nous pouvons utiliser BASE ou HCHP pour notre LoKy en commentant et décommentant ligne 11 or 12 du main-sketch.

```
10 /* Linky option tarifaire */
11 #define Linky_HCHP true
12 // #define Linky_BASE true
```

- Uploader le sketch dans votre Arduino et vérifiez le fonctionnement de LoKy à partir du moniteur série sur Arduino IDE.
- Enfin, on peut brancher le LoKy sur le compteur et voilà, on est prêt pour la dernière étape !

VII. Server d'application

a. Décodeur payload sur TTN

Allez maintenant dans la partie Payload formatters de TTN

Overview

End devices

Live data

Payload formatters

Uplink

Downlink

Integrations

Collaborators

API keys

General settings

Default uplink payload formatter

You can use the "Payload formatter" tab of individual end devices to test uplink payload formatters.

Setup

Formatter type*
Javascript

Formatter parameter*

```
1 function LoKyDecode(bytes) {  
2   var data_types = {  
3     0 : {'size': 2, 'index': 'VccTIC (in V)', 'divisor': 1000 },  
4     4 : {'size': 5, 'index': 'HCHC (in KWh)', 'divisor': 1000 },  
5     5 : {'size': 5, 'index': 'HCHP (in KWh)', 'divisor': 1000 },  
6     6 : {'size': 2, 'index': 'PTEC', 'divisor': 1 },  
7     7 : {'size': 2, 'index': 'IINST (in A)', 'divisor': 1 },  
8     8 : {'size': 5, 'index': 'BASE (in KWh)', 'divisor': 1000 },  
9     9 : {'size': 3, 'index': 'PAPP (in VA)', 'divisor': 1 },  
10  };  
11  };  
12    
13  function arrayToDecimal(stream, divisor) {  
14    var value = 0;  
15    for (var i = 0; i < stream.length; i++) {
```

Changez le décodage par défaut en Javascript et collez le contenu du fichier LoKy-main\LoKy_TTN_Decoder\LoKy_decoder_v2.js et enregistrez les modifications.

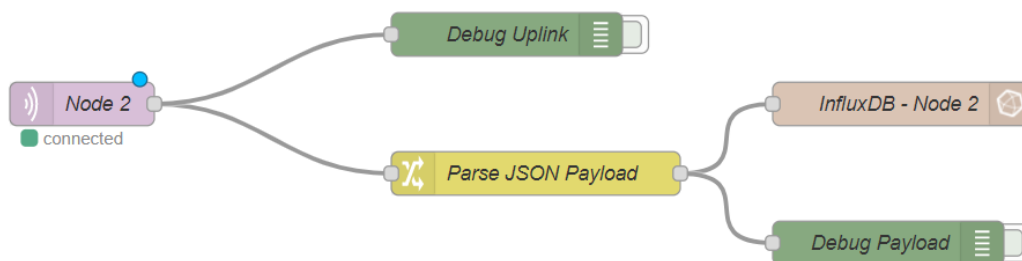
b. Integration MQTT avec le trio Node-RED, InfluxDB et Grafana

En suivant les instructions de TTN, nous pouvons facilement configurer le stockage des données dans le serveur de l'application.


Par rapport à la base de données InfluxDB et aux dashboard Grafana, vous pouvez revenir à la **partie 8** cet article de M. Jérôme :

[Monitoring d'une serre, du capteur jusqu'au serveur via le trio Lora, InfluxDB, Grafana](#)

Enfin, dans NodeRED, vous devez créer un « flow » comme celui-ci, nous vous l'expliquerons dans un prochain article de façon plus détaillé:




Le « **Parse JSON payload** » ne pourrait pas être plus simple.

General / LoKy ★ 

2021-06-03 19:10:04 to 2021-06-04 19:37:25 < > 🔍 ↺ ↻


BASE




node-2.mean

BASE

7872.799 MWh




PAPP



130 VA


node-1.mean

IINST (A)



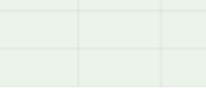
node-1.mean node-2.mean

PAPP (kVA)



node-1.mean node-2.mean

VccTIC




node-1.mean node-2.mean

PAPP

2021-06-04 08:53:00

node-2.mean: 7867.1 MWh

node-2.mean



1.20 kVA

Conclusion

Grâce à l'enthousiasme de Jérôme, j'ai pu terminer mon projet plus tôt que prévu. Cependant, ce n'est que le premier succès de ce projet, nous avons encore beaucoup de choses à mettre à niveau dans l'avenir. Alors n'hésitez pas à nous envoyer vos retours ainsi que les problèmes que vous rencontrez, nous apprécions votre contribution à [LoKy](#).

Et à bientôt pour de nouveaux articles



Ho Vu Duy Bao et Jérôme Ferrari – _Grenoble, Original Février-2021_ – Last Update Juillet 2021