

P2022 : Sunshare  
Courjaud Melvin

## Dossier technique du projet - partie individuelle




### Table des matières

<b>1 - SITUATION DANS LE PROJET.....</b>	<b>3</b>
1.1 - RAPPEL DES TÂCHES PROFESSIONNELLES À RÉALISER.....	3
1.2 - PRÉSENTATION DE LA PARTIE PERSONNELLE.....	3
1.2.1 - Introduction.....	3
1.2.2 - Synoptique de la réalisation.....	5
1.2.3 - Diagramme de déploiement.....	6
1.2.4 - Aperçu de la maquette de test.....	7
1.2.5 - Diagramme de cas d'utilisation.....	9
1.2.6 - Diagramme de classe.....	10
1.2.7 - Diagramme de la base de données.....	11
<b>2 - RÉALISATION DE LA TÂCHE PROFESSIONNELLE RÉCUPÉRER LA TIC.....</b>	<b>12</b>
2.1 - PRÉSENTATION DE LA TÂCHE.....	12
2.1.1 - La TIC.....	12
2.1.1.1 - Première solution.....	13
2.1.1.2 - Seconde solution.....	13
2.1.1.3 - Matériel utilisé.....	14
2.1.2 - Bibliothèques logicielles.....	15
2.2 - CONCEPTION DÉTAILLÉE.....	16
2.2.1 - Classe Linky.....	16
2.2.2 - Constructeur.....	17
2.2.3 - Méthodes LancerAcquisition et on Lire.....	18
2.2.4 - Méthodes on_TerminerAcquisition et DecoderTrame.....	18
2.2.5 - Méthodes ExtraireEtiquette et ExtraireDate.....	20
2.3 - TESTS UNITAIRES.....	21
2.3.1 - Test unitaire du matériel µTéléinfo.....	21
2.3.1.1 - Procédure de test.....	21
2.3.1.2 - Rapport d'exécution.....	23
2.3.2 - Problèmes rencontrés.....	23
2.3.2.1 - Problème matériel.....	23
2.3.2.2 - Checksum non pris en compte.....	25
<b>3 - RÉALISATION DE LA TÂCHE PROFESSIONNELLE LIRE LES IMPULSIONS.....</b>	<b>26</b>
3.1 - PRÉSENTATION DE LA TÂCHE.....	26
3.1.1 - Le compteur SDM120D.....	26
3.1.2 - Le shield BoxEnergie.....	26
3.1.3 - Bibliothèque logicielles.....	27
3.2 - CONCEPTION DÉTAILLÉE.....	27

3.2.1 - Classe CompteurSDM.....	27
3.2.2 - Constructeur de la classe.....	28
3.2.3 - Méthode Lire.....	28
3.2.4 - Méthode Get_nProduction.....	29
3.3 - TESTS UNITAIRES.....	29
3.3.1 - Test unitaire du shield BoxEnergie.....	29
3.3.1.1 - Procédure de test.....	29
3.3.1.2 - Rapport d'exécution.....	30
3.3.2 - Problèmes rencontrés.....	30
<b>4 - RÉALISATION DE LA TÂCHE PROFESSIONNELLE STOCKER EN LOCAL.....</b>	<b>30</b>
4.1 - PRÉSENTATION DE LA TÂCHE.....	30
4.1.1 - Bibliothèques utilisées.....	30
4.2 - CONCEPTION DÉTAILLÉE.....	30
4.2.1 - Diagramme de classe.....	30
4.2.2 - Constructeur de la classe Gestionnaire.....	31
4.2.3 - Constructeur de la classe BaseDeDonnees.....	32
4.2.4 - Méthodes OuvrirBdD et FermerBdD.....	33
4.2.5 - Méthode on_EnvoyerIndicateurs.....	33
4.2.6 - Méthode EnvoyerRequete.....	34
4.2.7 - Méthode on_CalculerMoyennes.....	35
4.2.8 - Méthode EnvoyerRequeteMoy.....	36
4.2.9 - Méthode on_LancerRetention.....	37
4.3 - TESTS UNITAIRES.....	38
4.3.1 - Test unitaire du fichier d'initialisation.....	38
4.3.1.1 - Procédure de test.....	38
4.3.1.2 - Rapport d'exécution.....	39
4.3.2 - Problèmes rencontrés.....	39
4.3.3 - Test unitaire de l'envoi des indicateurs de soutirage et d'injection à la base de données.....	39
4.3.3.1 - Procédure de test.....	39
4.3.3.2 - Rapport d'exécution.....	40
4.3.4 - Problèmes rencontrés.....	40
4.3.5 - Test unitaire des calculs de moyennes.....	40
4.3.5.1 - Procédure de test.....	40
4.3.5.2 - Rapport d'exécution.....	41
4.3.6 - Problèmes rencontrés.....	41
4.3.6.1 - Fonctionnement.....	41
4.3.6.2 - Gestion de la date difficile.....	42
4.3.7 - Test unitaire de la suppression automatique.....	42
4.3.7.1 - Procédure de test.....	42
4.3.7.2 - Rapport d'exécution.....	44
<b>5 - BILAN DE LA RÉALISATION PERSONNELLE.....</b>	<b>44</b>
5.1 - STATUT DES FONCTIONS À CHARGE.....	44
5.2 - POINTS À AMÉLIORER.....	44
5.3 - POINTS POSITIFS.....	44

## 1 - Situation dans le projet

### 1.1 - Rappel des tâches professionnelles à réaliser

Fonction secondaire de l'application (Fpi)	Description	État d'avancement
Fp1	<b>Récupérer les indicateurs :</b> C'est la partie électronique/Hardware du projet pour lire les indexes de consommation dans le compteur Linky, et récupérer les données de production local en provenance de l'onduleur (énergie solaire ou autre).	
Fp2	<b>Stocker en local :</b> Permettre le stockage en local chez le client SunShare des données de productions solaire, de consommation locale et d'injection sur le réseau électrique. La durée de rétention des données pourra être programmable et une suppression automatique après une certaine durée est envisageable. Pour stocker les valeurs sur le long terme, des moyennes des données sont calculées sur 1 jour, 1 semaine, 1 mois et 1 an. Ces valeurs ne sont pas affectées par la suppression automatique des valeurs en temps réel.	
Fs1	<b>Lire les information de connexion dans un fichier :</b> Les informations de connexion à la base de données doivent être stockées dans un fichier .ini, pour permettre au client de déporter la base de données s'il le souhaite et de changer les informations de connexion.	

### 1.2 - Présentation de la partie personnelle

#### 1.2.1 - Introduction

Ce dossier présente les tâches que j'ai effectuées durant mon projet. Étudiant A, COURJAUD Melvin, je me suis principalement intéressé à la partie embarquée, c'est-à-dire, la carte Raspberry pi. Mes tâches se sont concentrés sur la récupération dans le Linky des indicateurs et de les envoyer dans la base de données du Raspberry pi.

La réalisation du projet s'est déroulée en plusieurs phases jusqu'à présent :

- Découverte du projet
  - Étudier le projet
  - Étude matérielle
  - Mise en place des outils de gestion de projet (Trello / Gitlab)
- Sprint 1
  - Analyse SysML
  - Installation et configuration de la carte Raspberry (OS, bibliothèques de développement)
- Sprint 2
  - Communication entre le Raspberry et le Linky
  - Traitement des données reçues pour en extraire les indicateurs
- Sprint 3
  - Envoi des indicateurs à la base de données

- Gestion d'un fichier .ini servant à la configuration de l'application Qt
- Envoi de requêtes SQL permettant de calculer les moyennes des indicateurs
- Envoi de requêtes SQL de rétention (suppression de données trop anciennes)
- Communication entre le Raspberry et le compteur SDM120D pour le contrôle de la production électrique locale

La première phase du projet à été consacrée à la découverte du cahier des charges, et du fonctionnement du matériel, qui nous était encore totalement inconnu.

Le premier sprint consiste en l'analyse SysML et l'installation du matériel.

L'analyse SysML permet l'analyse, la conception, la vérification et la validation du système qui devra être mis en place, par exemple, en retranscrivant le cahier des charges sous formes de schéma, et de même pour les fonctionnement des différentes fonctionnalités du projet.

En ce qui concerne l'installation du matériel, j'ai dû installer l'OS du Raspberry Pi (Raspberry Pi OS) qui est un système d'exploitation libre basé sur Debian.

J'ai aussi dû installer les bibliothèques de développement sur le Rpi (Raspberry Pi), tel que les bibliothèques de Qt, une API orientée objet et développée en C++, ainsi que la bibliothèque pigpio, développée en C, qui permet le contrôle du GPIO (General Purpose Input Output) du Rpi. Afin de rendre le développement de l'application plus agréable, j'ai mis en place la compilation croisée afin de ne pas avoir besoin de me connecter au Rpi par interface graphique et compiler les programmes à partir de mon PC de développement sous Linux.

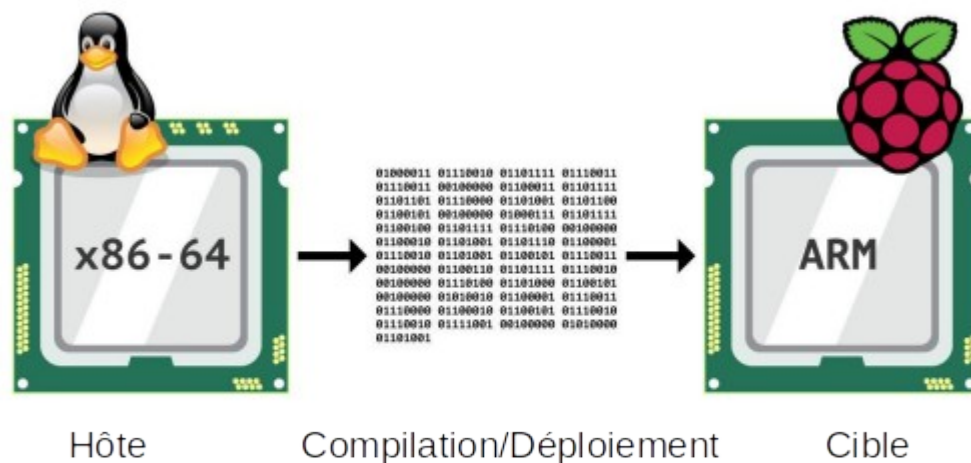


Figure 1: Image illustrant la compilation croisée

La compilation croisée fait référence aux chaînes de compilation capables de traduire un code source en code objet dont l'architecture processeur diffère de celle où la compilation est effectuée.

### 1.2.2 - Synoptique de la réalisation

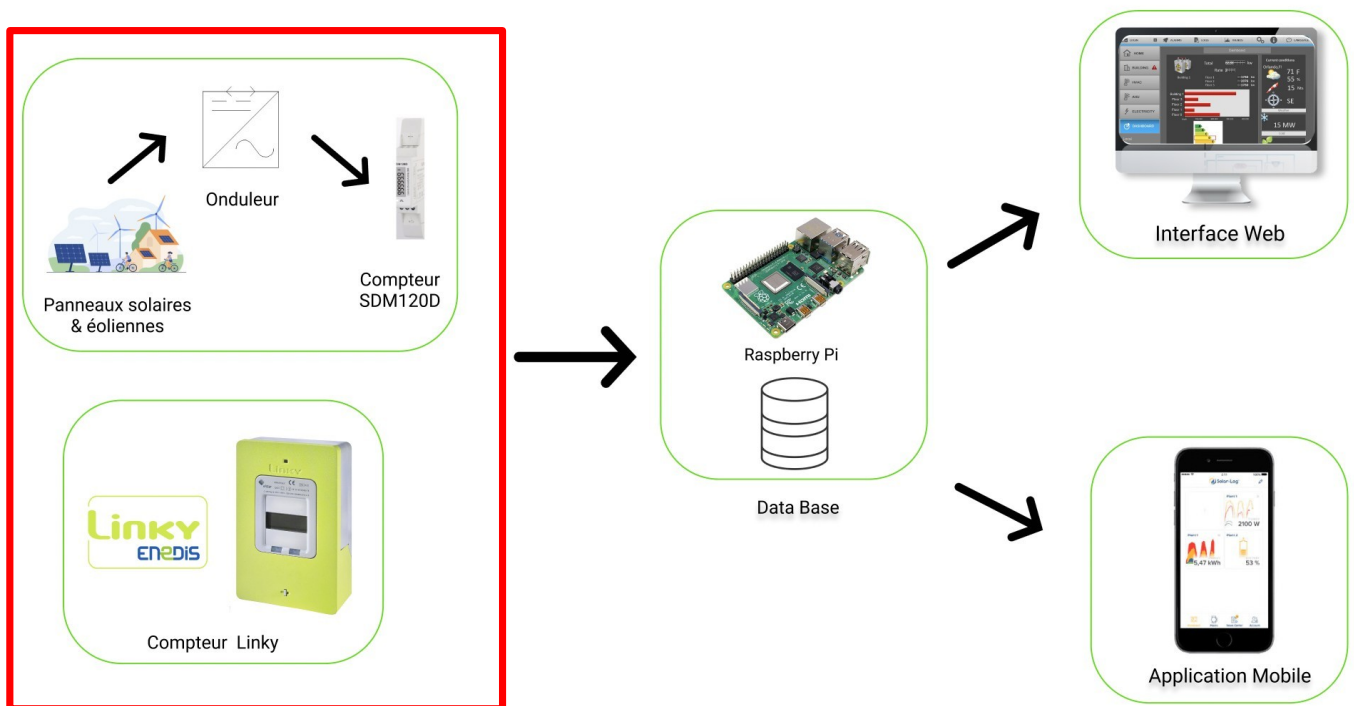


Figure 2: Synoptique général du système

Ma partie se concentre sur la collecte des indicateurs de puissance et de leur envoi dans la base de données (partie encadrée en rouge).

Les indicateurs sont les suivants :

- Le **soutirage** correspond à l'énergie vendue par Enedis et consommée par le particulier.
- L'**injection** est l'énergie produite localement et injectée sur le réseau Enedis.
- La **production** correspond à l'énergie produite et consommée localement.

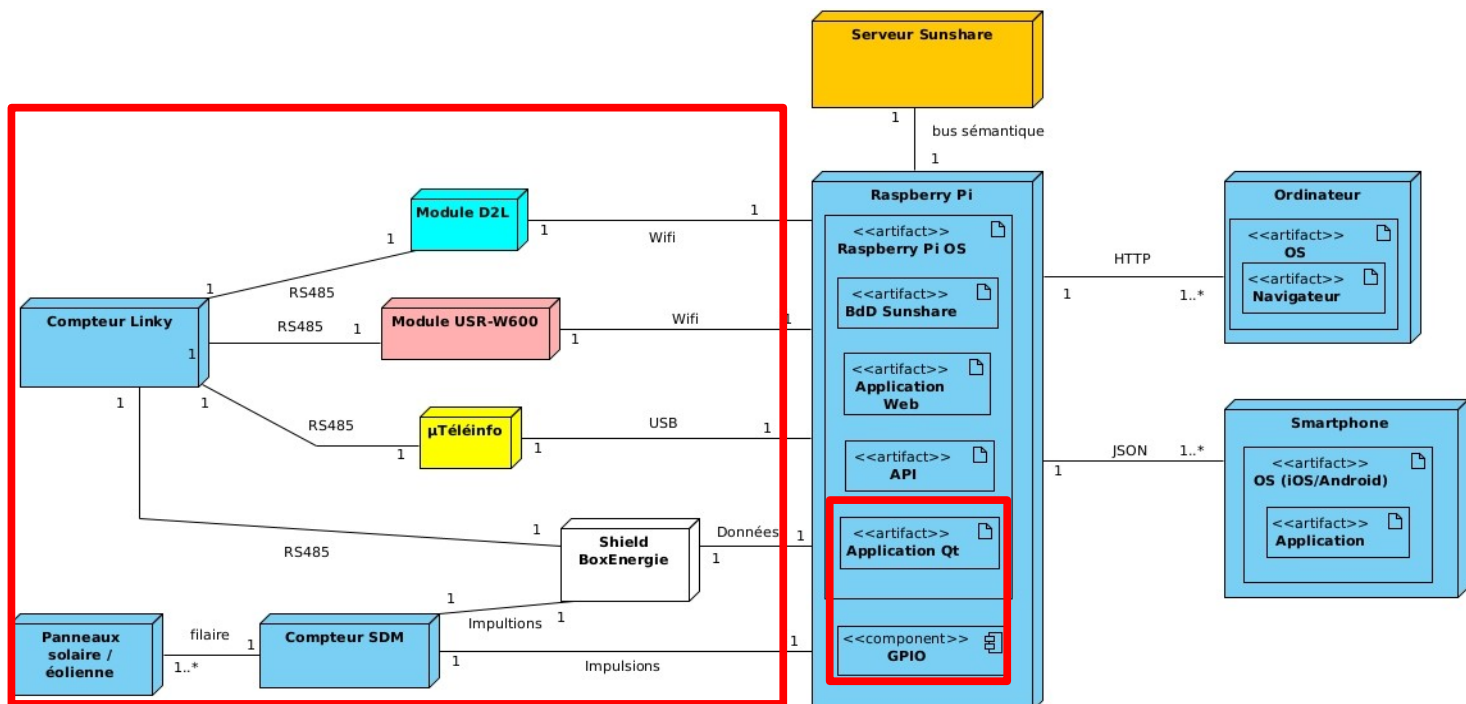
Le compteur SDM120D s'occupe de compter la production tandis que le Linky compte le soutirage et l'injection.

Le Linky transmet la TIC (protocole de Télé-Information Client), qui contient des informations relatives à la consommation, par l'intermédiaire d'une interface unidirectionnelle (disponible uniquement en lecture) en liaison série RS485.

Le SDM120D envoie des impulsions correspondant aux mesures qu'il réalise.

### 1.2.3 - Diagramme de déploiement

Le diagramme de déploiement ci-dessous permet de détailler les différents acteurs physiques du système, les parties encadrées en rouges sont celles que je traite.



Dessin 1: Diagramme de déploiement

- Le **Raspberry Pi** s'occupe du traitement des informations reçues pour en extraire les indicateurs (soutirage, injection, production) et les stocker dans une base de données (l'**Application Qt** gère à la fois de la collecte des indicateurs et de leur envoi dans la base de données), ces données sont ensuite rendues accessibles par application web. L'API permet à l'application mobile de recevoir les données de la base de données.
- Pour collecter la TIC (soutirage, injection), 3 solutions sont possibles :
  - Le dongle USB **µTéléinfo** est un appareil spécialement conçu pour collecter la TIC du Linky, c'est une solution filaire et simple. Plusieurs **µTéléinfo** peuvent être connectés à un Rpi dans le cas où le particulier possède plusieurs Linkys.
  - Le module **USR-W600** est un convertisseur série (RS232 et RS485) vers wifi.
  - le module **D2L** est un émetteur radio conçu spécialement pour le Linky (un emplacement est prévu sur le compteur pour l'y insérer). L'inconvénient de cette solution est qu'elle nécessite un partenariat avec l'entreprise eeSmart pour pouvoir utiliser l'API pour exploiter les données recueillies.
- Le shield **BoxEnergie** élaboré particulièrement pour ce projet par les élèves de Polytech Nantes et financé par financement participatif (crowdfunding). Ce composant peut à la fois récupérer la TIC et les impulsions. Il se place sur le **GPIO** du Rpi. Cependant, son utilisation n'a pas pu être mise en œuvre à cause d'un dysfonctionnement : il se peut que l'utilisation du BoxEnergie nécessite un *driver*<sup>1</sup>, auquel je n'ai pas accès. Il se peut aussi qu'il y ait un problème de compatibilité avec notre marque de Linky qui diffère de celui de Polytech.

1 Pilote de périphérique



### 1.2.4 - Aperçu de la maquette de test

Voici la maquette de test utilisée tout au long du projet. Celle-ci simule une consommation locale.

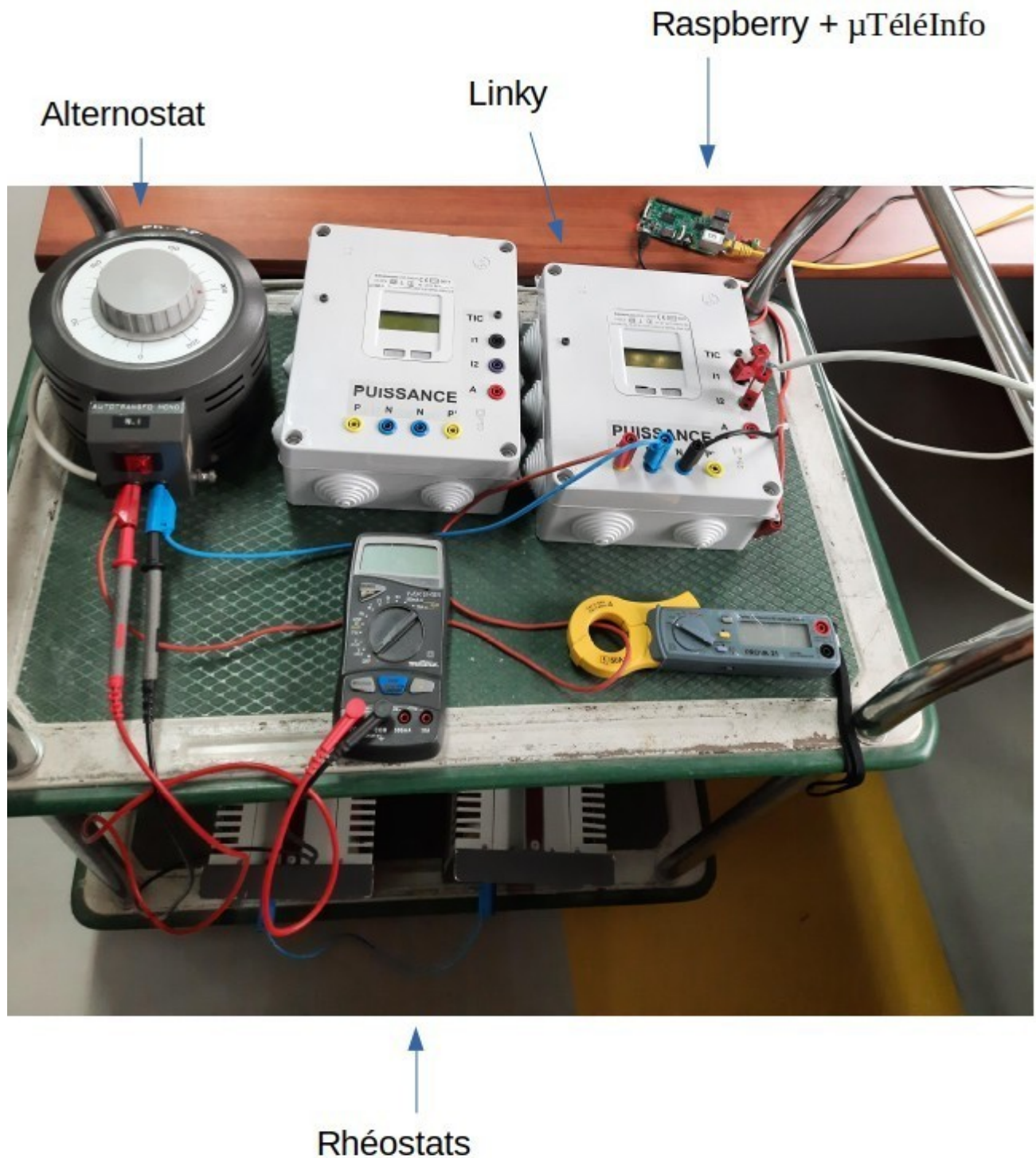


Figure 3: Maquette permettant de collecter la TIC du Linky à l'aide du µTéléInfo

L'alternostat est branché au secteur, il fournit une tension alternative variable. Afin de simuler une consommation locale, celui-ci est réglé sur 230V.

Les rhéostats sont des résistances variables, elles servent à limiter le courant passant dans le circuit.

Le Linky est placé entre l'alternostat et les rhéostats et fait les mesures électriques, il mesure le soutirage et l'injection.

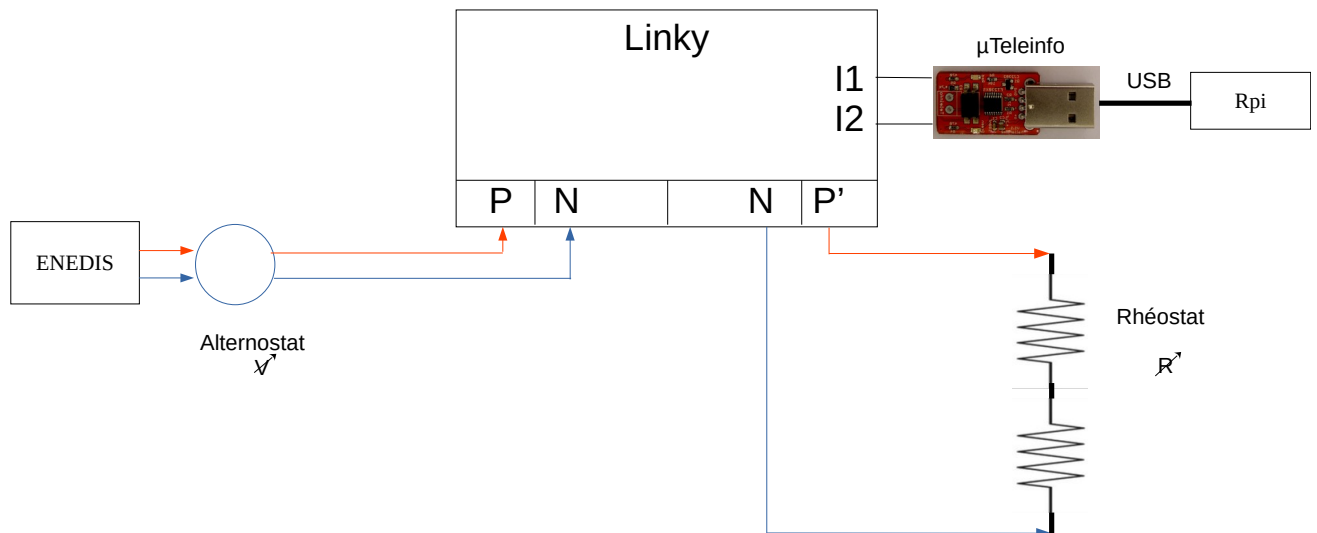


Figure 4: Schéma de la maquette simulant la consommation

Ce schéma simule une consommation avec les rhéostats pour pouvoir collecter le soutirage.

En ce qui concerne le Linky :

- **P** correspond à la phase en entrée du Linky et **P'** à la phase en sortie
- **N** au neutre
- **I1** et **I2** au circuit d'information de la TIC, une broche **A** est disponible pour servir d'alimentation, dans mon cas, c'est inutile car l'interface USB permet d'alimenter le µTéléinfo en 5V

Pour relever l'injection, il suffit d'inverser l'alternostat et les rhéostats :

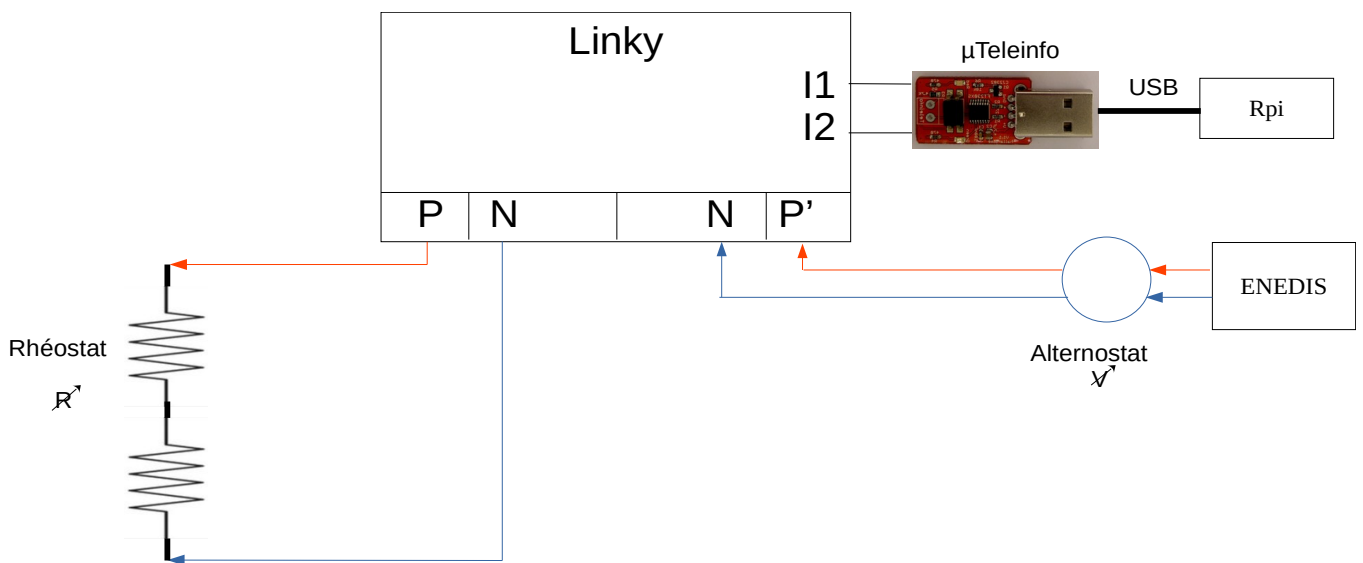


Figure 5: Schéma de la maquette simulant l'injection

Le Linky n'est pas associé au réseau CPL<sup>1</sup> d'Enedis, ce qui signifie qu'aucune données ne leur sera envoyé. Les tests peuvent être réalisés sans problèmes.

1 Courant Porteur en Ligne, permet de construire un réseau informatique sur un réseau électrique



### 1.2.5 - Diagramme de cas d'utilisation

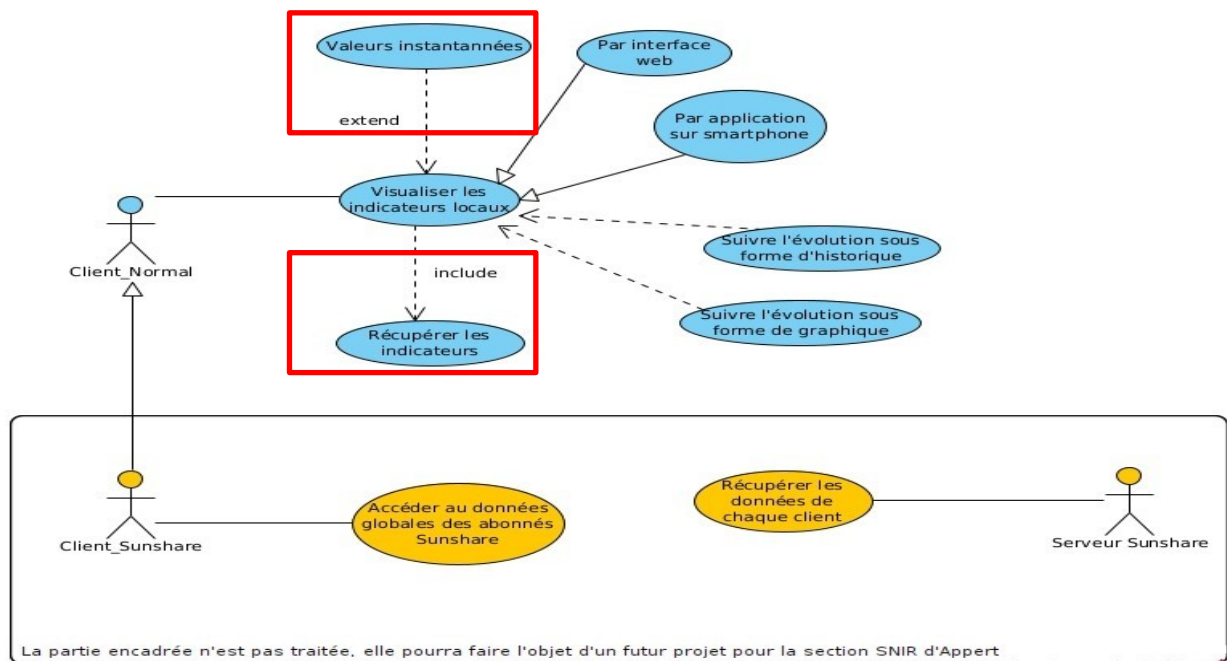


Figure 6: Diagramme des cas d'utilisation du système

Pour rappel, mes tâches sont les suivantes (le cas d'utilisation Récupérer les Indicateurs les regroupe) :

- Récupérer les indicateurs locaux
- Les stocker dans une base de données

Les indicateurs sont les suivants :

- Soutirage
- Injection
- Production

## 1.2.6 - Diagramme de classe

Voici le diagramme représentant la solution final du projet.

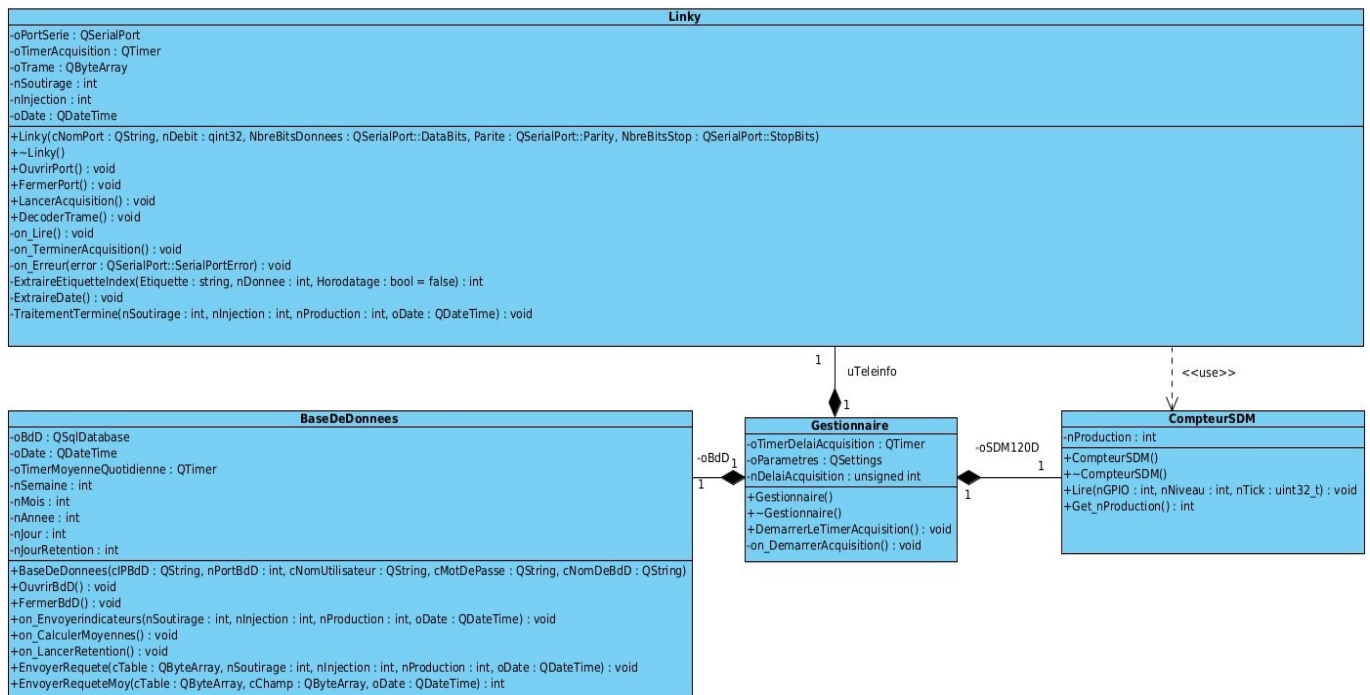


Figure 7: Diagramme de classe final

- **Gestionnaire** : gère le programme dans sa globalité, il permet par exemple de modifier le délai entre chaque acquisition, les paramètres de connexion à la base de données à partir d'un fichier *ini*.
- La classe **Linky** gère la collecte de la TIC et son traitement.
- La classe **CompteurSDM** permet de traiter les impulsions du compteur SDM120D.
- La classe **BaseDeDonnees** gère l'envoi de requêtes à la base de données.

Chacune de ces classes sera présentée dans les pages suivantes.

## Mise en forme :

Afin de repérer plus facilement le type des variables, un préfixe leur est attribué

- n : nombre entier
- c : chaîne de caractères
- o : objet (difficile à catégoriser)
- on\_ : slot (pour les méthodes)

### 1.2.7 - Diagramme de la base de données

Les indicateurs doivent être envoyés dans une base de données afin d'être exploités par le serveur web et l'application mobile

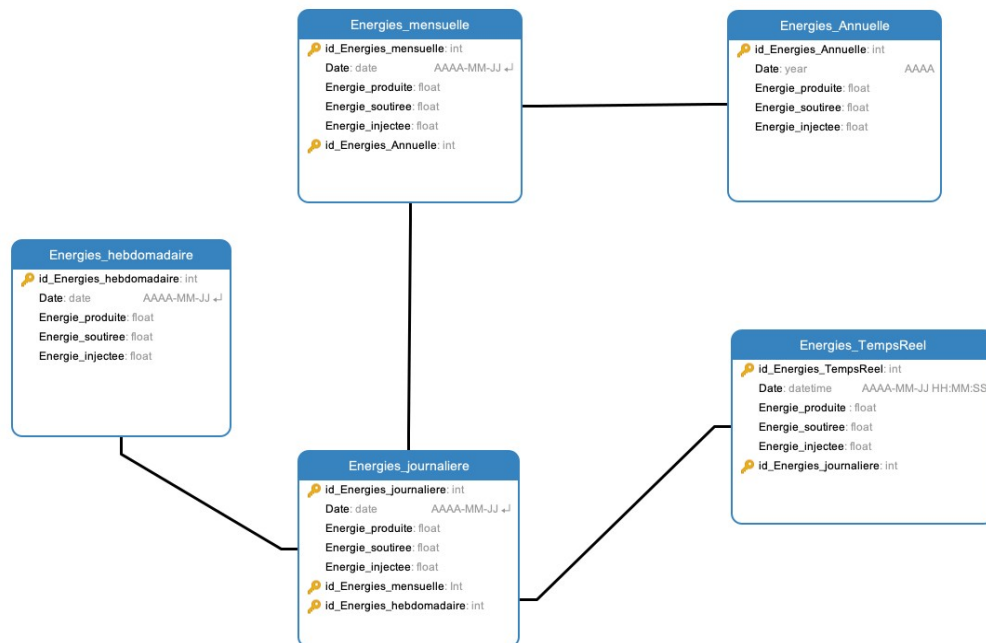


Figure 8: Schéma de la base de données

Les indicateurs sont en permanence envoyés dans la table **Energies\_TempsReel**, cette table contient les données en temps réel recueillies par l'application Qt.

Une moyenne des instantanées est ensuite réalisée pour être stocker dans les autres stables.

Afin de ne pas saturer la base de données, les données trop anciennes sont supprimées automatiquement de la table **Energies\_TempsReel**.

Le SGBD utilisé pour la base de données est MariaDB.

## 2 - Réalisation de la tâche professionnelle Récupérer la TIC

### 2.1 - Présentation de la tâche

La tâche *Récupérer la TIC* consiste en la collecte de la TIC du Linky et d'effectuer un traitement pour en extraire les indicateurs de soutirage et d'injection, mais également la date de ce prélèvement.

#### 2.1.1 - La TIC

Le Linky transmet un signal modulé en amplitude, ce qui signifie que le signal doit être traité avant d'être exploité par un programme.

Les caractéristiques du signal sont les suivantes :

- logique inverse : si la porteuse est présente alors le bit vaut « 0 », et si la porteuse est absente, le bit vaut « 1 »
- 9 600 bauds
- 1 bit de start correspondant à un "0" logique
- 7 bits pour représenter le caractère en ASCII (bits de donnée)
- 1 bit de parité, parité paire
- 1 bit de stop correspondant à un "1" logique.

Pour exploiter la TIC, il faut d'abord démoduler le signal : passer d'un signal sinusoïdale à un signal carré, mais il faut aussi inverser le signal (logique inverse).

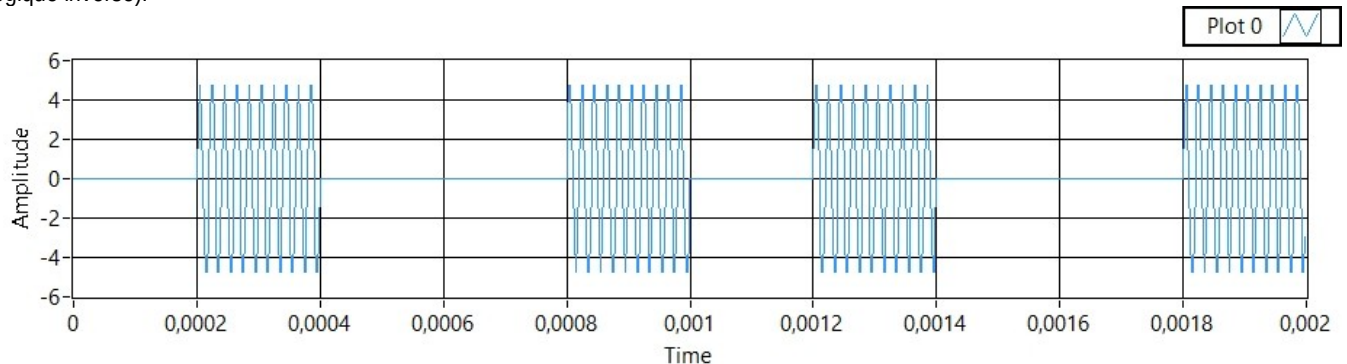


Figure 9: Signal modulé en entrée simulé avec LabView

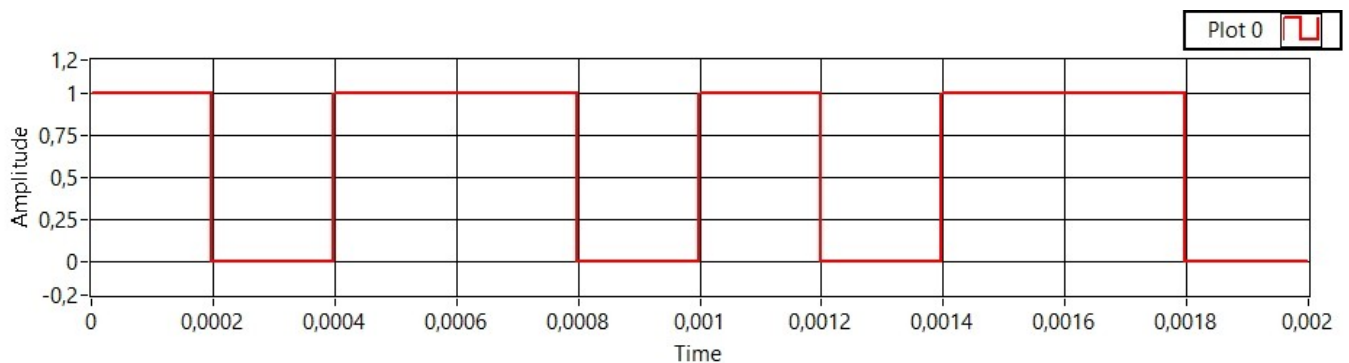


Figure 10: Signal carré en sortie simulé avec LabView

### 2.1.1.1 - PREMIÈRE SOLUTION

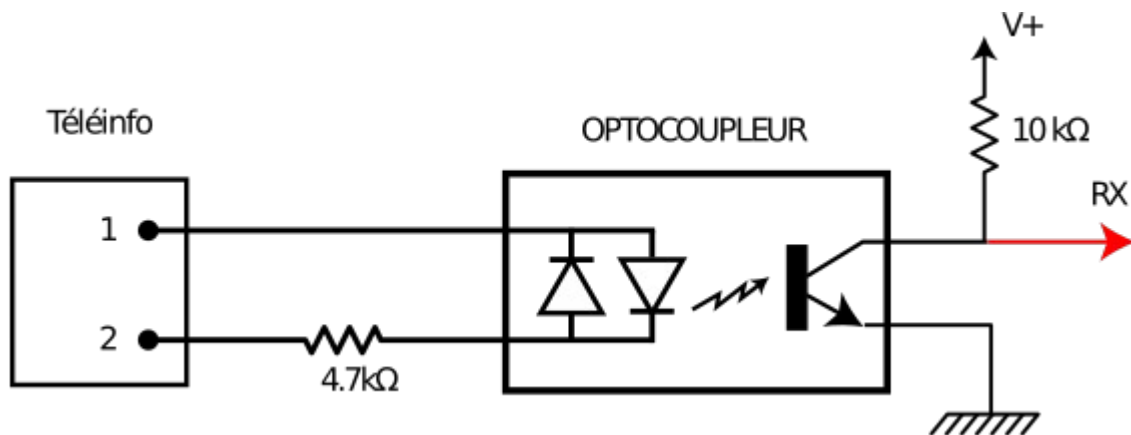


Figure 11: Première solution de démodulateur

Le signal modulé passe par un optocoupleur, composé de :

- Deux DELs en parallèles et dont le sens s'oppose, qui émettent de la lumière chaque fois qu'il y a un signal. Le fait de mettre deux diodes ainsi permet de collecter le signal entier : s'il n'y avait qu'une diode seule la partie positive ou négative serait récupérée.
- Un phototransistor, qui est un interrupteur laissant passer un signal seulement s'il capte de la lumière.

S'il y a un signal, les DELs sont allumées, le phototransistor est fermé, RX est ainsi relié à la masse et vaut 0 V.

S'il n'y a pas de signal, les DELs sont éteintes le phototransistor est ouvert, RX n'est plus relié à la masse, il vaut donc V+.

Cette solution pose un problème : si le bruit est trop important, les DELs peuvent s'allumer par erreur, ce qui peut avoir une incidence sur le signal en sortie.

### 2.1.1.2 - SECONDE SOLUTION

Une autre solution un peu plus poussée est possible :

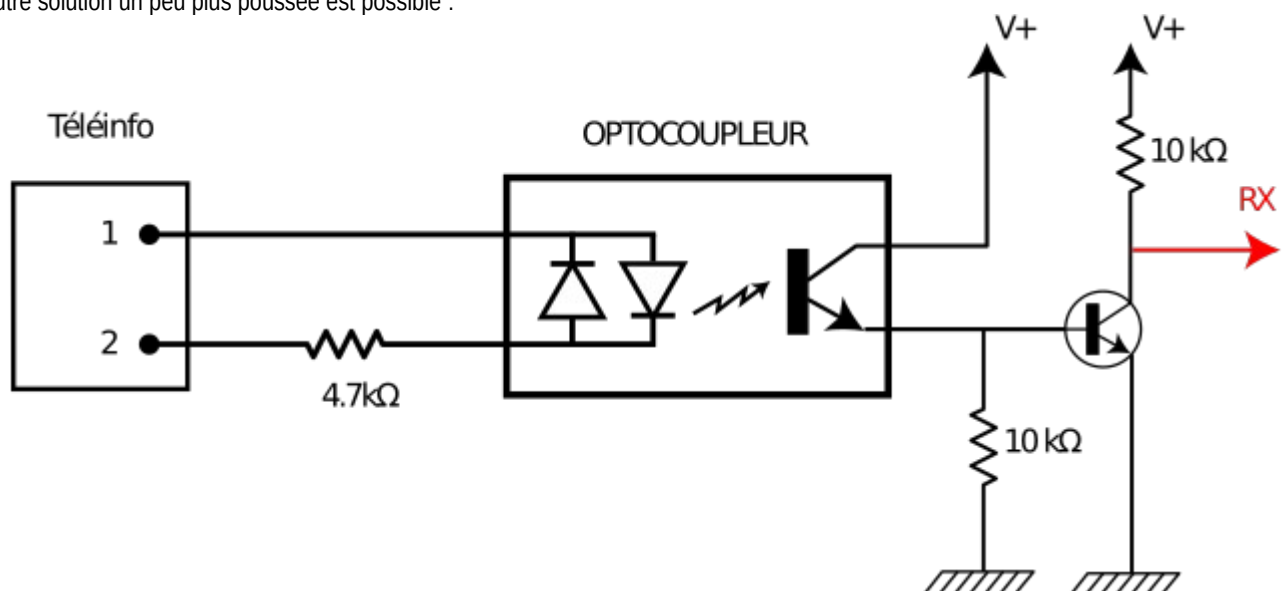


Figure 12: Seconde solution de démodulateur

La différence notable est le transistor placé en sortie.

S'il y a un signal, le phototransistor est fermé, la base du transistor est alimentée et celui-ci est fermé, alors RX est relié à la masse et vaut 0 V.

S'il n'y a pas de signal, le phototransistor est ouvert, la base du transistor n'est pas alimentée et celui-ci est ouvert, alors RX est relié à V+.

### 2.1.1.3 - MATÉRIEL UTILISÉ

Dans mon projet, j'utilise le dongle USB  $\mu$ Téléinfo créé par Charles-Henry Hallard (<https://hallard.me/>) dont le fonctionnement est le même que vu précédemment.

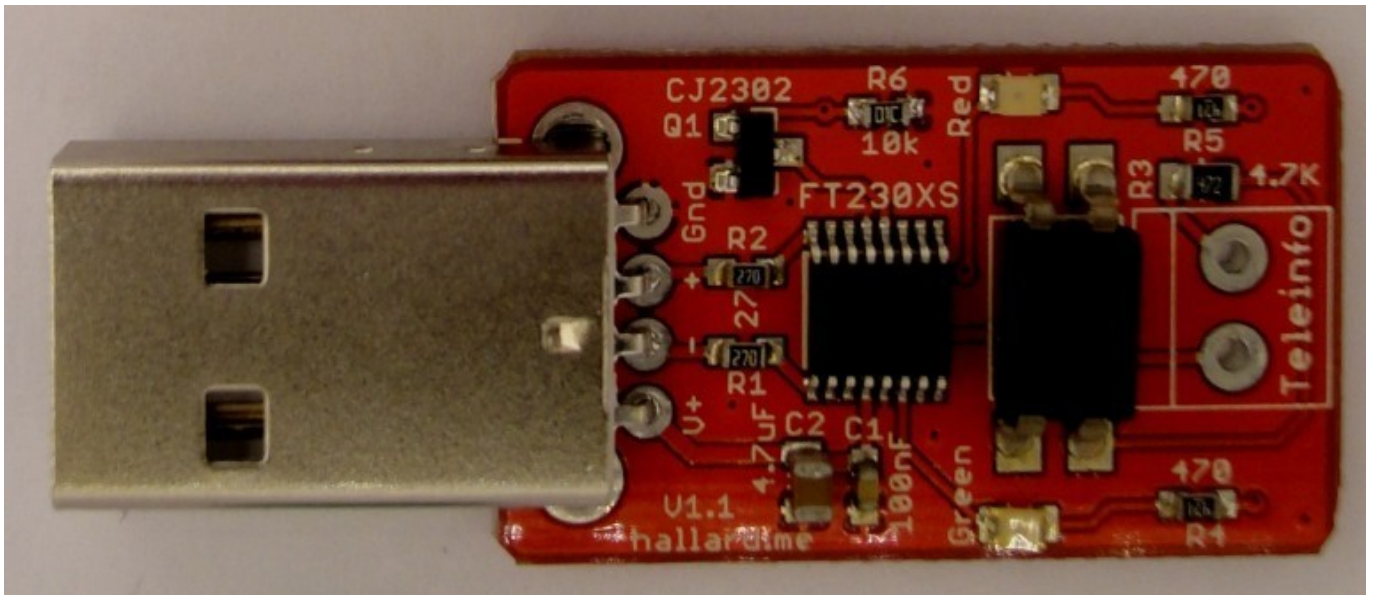


Figure 13:  $\mu$ Téléinfo

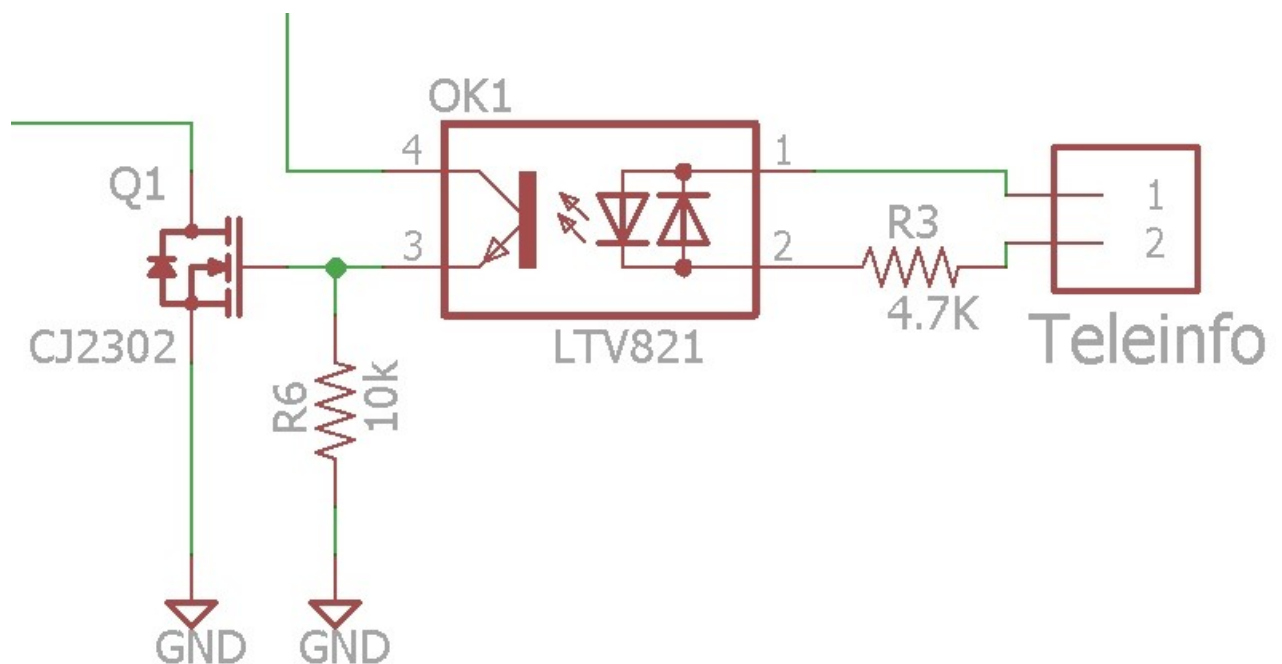


Figure 14: Schéma électrique du  $\mu$ Téléinfo



Après avoir fait des tests, je remarque que les signaux recueillis correspondent aux signaux simulés sur Labview. Cela signifie que le  $\mu$ Téléinfo fait bien son travail.

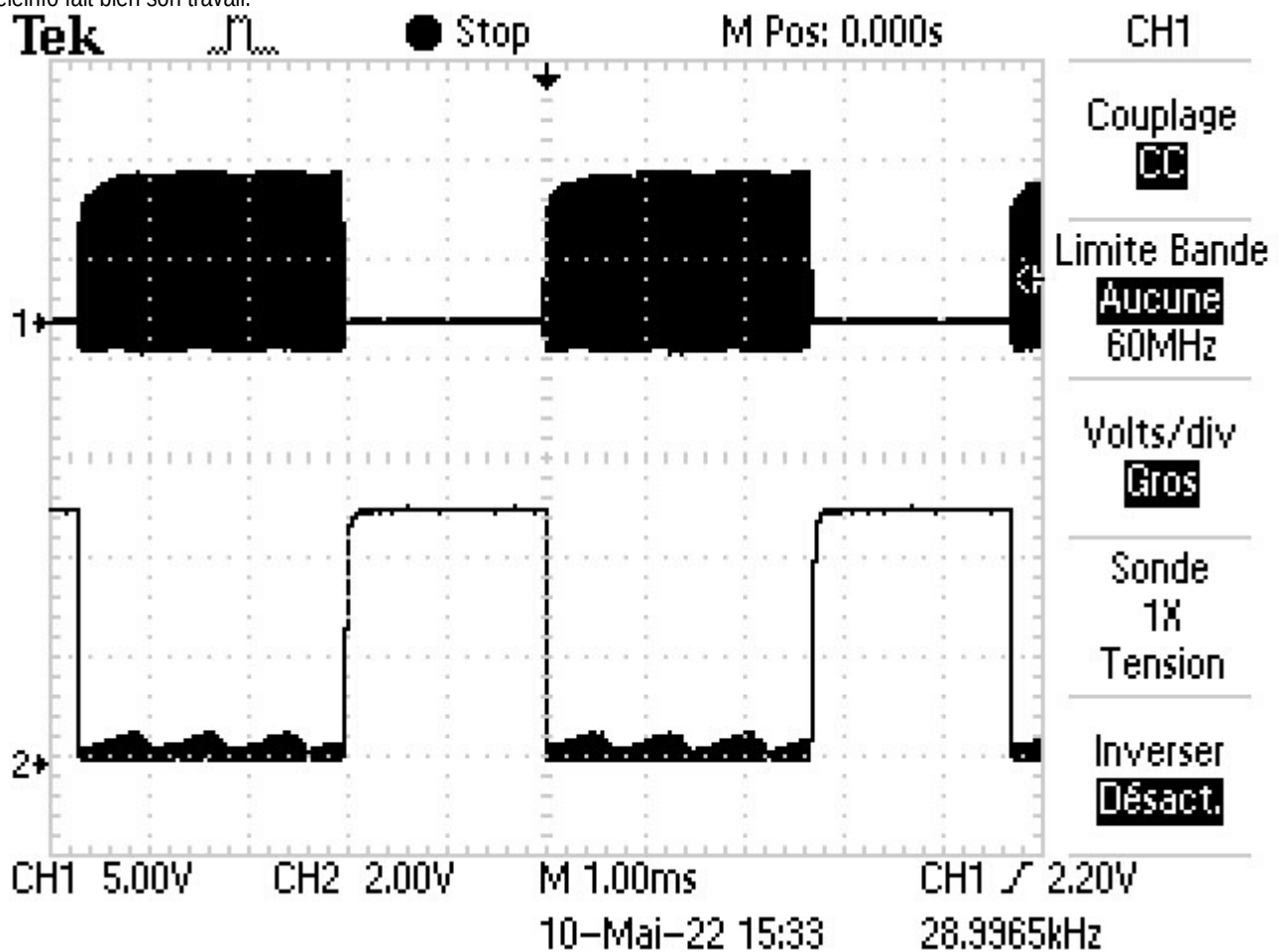


Figure 15: Signaux mesurés avec un oscilloscope

### 2.1.2 - Bibliothèques logicielles

Les bibliothèques utilisées dans cette tâche sont issues de l'API Qt. Qt est multi-plateforme, ce qui signifie que ses bibliothèques peuvent aussi bien être utilisées sur des ordinateurs de bureau (Windows, Mac, Linux) mais aussi mobile (Android, iOS). La collecte de la TIC se fait à l'aide du  $\mu$ Téléinfo, qui est un dongle USB, cela signifie donc que ce programme est compatible avec beaucoup de matériel, et donc que cette tâche peut être réalisée par n'importe quel appareil muni d'un port USB.

## 2.2 - Conception détaillée

### 2.2.1 - Classe Linky

Linky
<pre> -oPortSerie : QSerialPort -oTimerAcquisition : QTimer -oTrame : QByteArray -nSoutirage : int -nInjection : int -oDate : QDateTime  +Linky(cNomPort : QString, nDebit : qint32, NbreBitsDonnees : QSerialPort::DataBits, Parite : QSerialPort::Parity, NbreBitsStop : QSerialPort::StopBits) +~Linky() +OuvrirPort() : void +FermerPort() : void +LancerAcquisition() : void +DecoderTrame() : void -on_Lire() : void -on_TerminerAcquisition() : void -on_Erreur(error : QSerialPort::SerialPortError) : void -ExtraireEtiquetteIndex(Etiquette : string, nDonnee : int, Horodatage : bool = false) : int -ExtraireDate() : void -TraitementTermine() : void </pre>

Figure 16: Classe Linky

Les attributs sont les suivants :

- **oPortSerie** : permet la lecture/écriture sur un port série
- **oTimerAcquisition** : permet de limiter la durée d'acquisition de la TIC
- **oTrame** : chaîne de caractère possédant des fonctionnalités avancées (pour le traitement). Elle stock les données lu par le port série
- **nSoutirage** : contient l'index de soutirage
- **nInjection** : contient l'index d'injection
- **oDate** : contient la date

Les méthodes :

- **OuvrirPort** : permet d'ouvrir le port série
- **FermerPort** : ferme le port série
- **LancerAcquisition** : lance l'acquisition de la TIC
- **DecoderTrame** : démarre le traitement de la TIC
- **on\_Lire** : slot appelé à chaque fois qu'une donnée est lisible
- **on\_TerminerAcquisition** : slot appelé lorsque l'acquisition est terminée
- **on\_Erreur** : slot appelé chaque fois qu'il y a une erreur sur le port série
- **ExtraireEtiquette** : collecte un indicateur en fonction du nom de son étiquette et du nombre de caractères de données de cet indicateur
- **ExtraireDate** : extrait la date contenue dans la TIC
- **TraitementTermine** : signal émis à la fin du traitement pour envoyer les données à la classe *BaseDeDonnees*

## 2.2.2 - Constructeur

Le constructeur permet d'initialiser la classe Linky

```
Linky::Linky(QString cNomPort, qint32 nDebit, QSerialPort::DataBits NbreBitsDonnees,
             QSerialPort::Parity Parite, QSerialPort::StopBits NbreBitsStop, QObject *parent) :
    QObject(parent),
    oPortSerie(cNomPort),
    nSoutirage(0),
    nInjection(0),
    cDate(0)
{
    // Test si le port est bien branché
    QSerialPortInfo InfoPort(cNomPort);
    if(InfoPort.isNull())
    {
        cout << "Dongle USB non branche.\n";
        exit(-1);
    }

    /** Le port série est initialisé ici
     * S'il y a une erreur, celui-ci est inutilisable, c'est pourquoi le programme est quitté si c'est
     le cas
     */
    if (!this->oPortSerie.setBaudRate(nDebit)) {
        cout << "Le debit est incorrect\n";
        exit(-1);
    }
    if (!this->oPortSerie.setDataBits(NbreBitsDonnees)) {
        cout << "Le nombre de bits de donnee est incorrect\n";
        exit(-1);
    }
    if (!this->oPortSerie.setParity(Parite)) {
        cout << "La parite est incorrecte\n";
        exit(-1);
    }
    if (!this->oPortSerie.setStopBits(NbreBitsStop)) {
        cout << "Nombre de bits de stop incorrect\n";
        exit(-1);
    }

    oTimerAcquisition.setInterval(nDureeAcquisition);

    // signal émis à chaque fois qu'une donnée est lisible : il faut la lire
    connect(&oPortSerie, &QSerialPort::readyRead,
            this, &Linky::on_Lire );
    // signal émis en cas d'erreur sur le port série
    connect(&oPortSerie, &QSerialPort::errorOccurred,
            this, &Linky::on_Erreur);
    // signal émis lorsque le timer d'acquisition prend fin : l'acquisition est terminée
    connect(&oTimerAcquisition, SIGNAL(timeout() ),
            this, SLOT(on_TerminerAcquisition() ) );
}
```

L'intervalle entre chaque acquisition est initialisée dans le constructeur à l'aide d'une constante. L'intervalle est de 3,5 secondes (variable nDureeAcquisition) car cela permet de collecter 2 fois la TIC, cela permet d'éviter les erreurs lors de la lecture des étiquettes lorsqu'elles sont collectées en fin et début de trame car elles ont des chances d'être coupées.

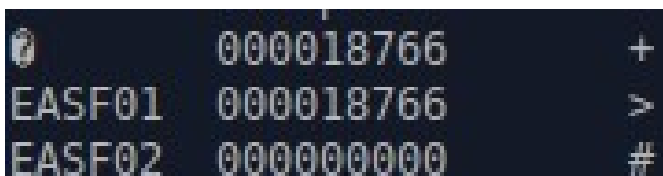


Figure 17: La première étiquette lue est illisible et donc inexploitable

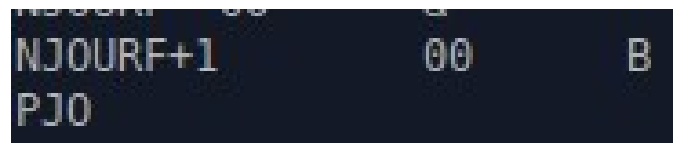


Figure 18: La dernière étiquette est incomplète et donc inexploitable

### 2.2.3 - Méthodes LancerAcquisition et on Lire

Méthode permettant de lancer l'acquisition. Le fait d'ouvrir le port série rend les données lisibles, l'objet oPortSerie envoie alors en permanence le signal *readyRead* (voir connect du constructeur) et le slot on\_Lire est appelé.

```
void Linky::LancerAcquisition()
{
    // ouvre le port série pour acquérir les données
    this->OuvrirPort();

    cout << "Debut de l'acquisition\n";

    // démarre le timer d'acquisition
    oTimerAcquisition.start();
}
```

Le slot on\_Lire ajoute le dernier caractère lu à la fin de la chaîne de caractères.

```
void Linky::on_Lire()
{
    // place le dernier bit lu à la fin du tableau
    oTrame.append(oPortSerie.readAll());

    if (!oTimerAcquisition.isActive())
    {
        oTimerAcquisition.start();
    }
}
```

### 2.2.4 - Méthodes on TerminerAcquisition et DecoderTrame

Le slot on\_TerminerAcquisition est appelé quand le timer envoie le signal timeout. Il ferme le port série, ainsi plus aucune donnée n'est en attente de lecture : le slot on\_Lire n'est plus appelé. Par défaut, le timer est rappelé automatiquement après avoir lancé le signal *timeout*, c'est pourquoi il faut le stopper avec la méthode *stop*.

```
void Linky::on_TerminerAcquisition()
{
    // comme l'acquisition est terminée, il faut fermer le port, sinon les
    // ressources de la machine sont utilisées pour rien
    FermerPort();
    // on arrête le timer
    oTimerAcquisition.stop();

    cout << "Fin de l'acquisition\n";

    // affiche un message d'erreur si aucune donnée n'a été lue
    if (oTrame.isEmpty()) {
        cout << "Aucune donnée n'est disponible.\n";
    } else {
        // cout << "Données récupérées :\n" << oTrame.toStdString() << "\n\n";
    }

    // après l'acquisition de la TIC, il faut extraire les données de la TIC et les
    // stocker dans les attributs de la classe
    DecoderTrame();
}
```

La méthode DecoderTrame est appelée après la fin de l'acquisition

```
void Linky::DecoderTrame()
{
    cout << "Traitement de la TIC.\n";

    if(ExtraireEtiquette("VTIC",2) == 2)
    {
        // extraction des étiquettes pour en extraire les valeurs et les stocker
        dans l'attribut correspondant
        this->nSoutirage = ExtraireEtiquette("SINSTS", 5);
        this->nInjection = ExtraireEtiquette("SINSTI", 5);
        ExtraireDate();// stock la date dans l'attribut cDate

        // traitement fini : on libère l'espace mémoire utilisé par la chaîne de
        caractères comportant la trame complète
        this->oTrame.clear();

        // le traitement est terminé : un signal est émis à la classe BaseDeDonnees
        avec les indicateurs
        emit TraitementTermine(this->nSoutirage, this->nInjection,
        CompteurSDM::Get_nProduction(), this->oDate);
    } else {
        cout << "Traitement impossible\n";
    }
    cout << "\n";
}
```

Le test avec l'étiquette VTIC sert seulement à vérifier si la TIC est lisible. Après la collecte des différentes informations, les données sont envoyées à la classe BaseDeDonnees grâce à au signal TraitementTermine.

La documentation complète des étiquettes est fournie par Enedis (page 18-21) : <https://www.enedis.fr/media/2035/download>

Les trames qui nous intéressent sont les suivantes :

Donnée Restituée	Étiquette	Horodate	Nombre de caractères de la donnée	Unité donnée	Triphasé seulement	Producteur seulement
Adresse Secondaire du Compteur	ADSC		12	Sans		
Version de la TIC	VTIC		2	Sans		
Date et heure courante	DATE	✓	0	Sans		
Puissance app. Instantanée soutirée	SINSTS		5	VA		
Puissance app. Instantanée injectée	SINSTI		5	VA		✓

Figure 19: Tableau comportant les étiquettes utilisées dans le projet

## 2.2.5 - Méthodes ExtraireEtiquette et ExtraireDate

Les informations sont envoyées dans ces formats :

Table 6 - Information systématiquement horodatée par le compteur

Format d'un groupe contenant une donnée horodatée								
< LF > (0x0A)	Etiquette	< HT > (0x09)	Horodate	< HT > (0x09)	Donnée	< HT > (0x09)	Checksum	< CR > (0x0D)
	Zone contrôlée par la checksum							

Table 7 - Information sans horodate

Format d'un groupe contenant une donnée non horodatée						
< LF > (0x0A)	Etiquette	< HT > (0x09)	Donnée	< HT > (0x09)	Checksum	< CR > (0x0D)
	Zone contrôlée par la checksum					

Figure 20: Format des trames

## Données horodatées ou non

Certaines données sont horodatées, comme l'étiquette DATE qui permet de collecter l'heure courante, son format est le suivant :

Le format utilisé pour les horodates est **S**AAMMMJJ**h**hmm**s**, c'est-à-dire **S**aison, **A**nnée, **M**ois, **J**our, **h**eure, **m**inute, **s**econde.

La date utilise 13 caractères au total.

## ExtraireEtiquette

Cette méthode permet de récupérer la valeur d'une étiquette à partir de son nom, du nombre de caractères de données et si elle est horodatée. Le paramètre bHorodatage a pour valeur par défaut false (faux).

```
int Linky::ExtraireEtiquette(QByteArray cEtiquette, int nDonnee, bool bHorodatage)
{
    int nRetour (0); // variable qui contiendra la valeur de l'étiquette
    int nIndex (0); // numéro de case de l'attribut oTrame (chaîne de caractères)

    // Première étape, se placer au début de l'étiquette qui nous intéresse
    nIndex = this->oTrame.indexOf(cEtiquette, 0); // 0 correspond au premier caractère
    // on passe le nom de l'étiquette + caractère '\t'
    nIndex += cEtiquette.length() + 1;
    // Si l'étiquette est horodatée
    if(bHorodatage)
    {
        nIndex += 13; // Les prochains caractères correspondent à la date, ils ne nous
        intéressent pas
    }
    // Extraction de la valeur chiffrée
    nRetour = this->oTrame.mid(nIndex, nDonnee).toInt();

    cout << cEtiquette.toString() << " : " << nRetour << "\n";

    return nRetour;
}
```



La méthode « toInt() » de la classe QByteArray (ici l'objet oTrame) permet de convertir la valeur de retour de la procédure « mid() » en nombre (int).

La ligne « cout <<... » permet d'afficher le nom de l'étiquette (SINSTS, SINSTI...) suivi de sa valeur.

## ExtraireDate

Cette méthode a pour objectif de collecter la date de la TIC, elle est rattachée à l'étiquette DATE. Il faut ensuite extraire les caractères correspondant à l'horodatage et en changer le format pour pouvoir l'envoyer à la base de données. Le format compris par la base de données est AAAA-MM-JJ HH:MM:SS

```
void Linky::ExtraireDate()
{
    int nIndex (0);
    QTime Heure;
    QDate Date;

    nIndex = this->oTrame.indexOf("DATE", 0);

    nIndex += 4 + 1 +1; // on zappe le nom de l'étiquette, caractère '\t' et le caractère de saison

    // on récupère l'année, le mois et le jour
    Date.setDate(2000 + this->oTrame.mid(nIndex, 2).toInt(),
                this->oTrame.mid(nIndex + 2, 2).toInt(), this->oTrame.mid(nIndex + 4, 2).toInt());
    // on récupère l'heure, les minutes et les secondes
    Heure.setHMS(this->oTrame.mid(nIndex + 6, 2).toInt(),
                 this->oTrame.mid(nIndex + 8, 2).toInt(), this->oTrame.mid(nIndex + 10, 2).toInt());

    // on les applique dans l'attribut
    this->oDate.setDate(Date);
    this->oDate.setTime(Heure);

    cout << "Date : " << this->oDate.toString(FormatDate).toStdString() << "\n";
}
```

## 2.3 - Tests unitaires

### 2.3.1 - Test unitaire du matériel µTéléinfo

Ce test a pour objectif de récupérer les données envoyées par le Linky (la TIC) et d'en faire le traitement pour en extraire les indicateurs (soutirage et injection) ainsi que la date de prélèvement de ces données.

#### 2.3.1.1 - PROCÉDURE DE TEST

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U1.0	Compilation	cd ~/Documents/RecupererTIC/_build qmake ../RecupererTIC.pro make ./RecupererTIC
		Le projet compile sans avertissement
U1.1	Linky::Linky()	L'objet est construit sans problème

Id.	Méthode testée Description Sommaire	Procédure de test								
		Résultats attendus								
		<div>La communication entre le Linky et le Raspberry Pi est opérationnelle</div> <table><tr><td>Vitesse :</td><td>9600 bauds</td></tr><tr><td>Parité :</td><td>Paire</td></tr><tr><td>Bit de stop :</td><td>1 bit</td></tr><tr><td>Bits de données :</td><td>7 bits</td></tr></table>	Vitesse :	9600 bauds	Parité :	Paire	Bit de stop :	1 bit	Bits de données :	7 bits
Vitesse :	9600 bauds									
Parité :	Paire									
Bit de stop :	1 bit									
Bits de données :	7 bits									
U1.2	void Linky::LancerAcquisition()	<div>La méthode demande la réception de la trame, le port série est ouvert, et le timer d'acquisition est démarré</div> <div>La trame est réceptionnée sans erreur</div>								
U1.3	void Linky::on_TerminerAcquisition()	<div>Déconnecte le port série, stop le timer et affiche la trame</div> <div>Le port série est libéré</div> <div><div>Trame entière :</div><div>ADSC 811875704707 D VTIC 02 J DATE e190819144809 4 NGTF BASE &lt; LTARF BASE F EAST 000011918 # EASF01 000011918 6 EASF02 000000000 # EASF03 000000000 \$ EASF04 000000000 % EASF05 000000000 &amp; EASF06 000000000 ' EASF07 000000000 ( EASF08 000000000 ) EASF09 000000000 * EASF10 000000000 " EASD01 000011918 4 EASD02 000000000 ! EASD03 000000000 " EASD04 000000000 # EAIT 000001576 X ERQ1 000000317 F ERQ2 000000004 @ ERQ3 000000000 = ERQ4 000000000 &gt; IRMS1 003 1 URMS1 232 A PREF 03 B PCOUP 03 \ SINSTS 00809 W SMAXSNe190819144757 00810 + SMAXSN-1 e190812192459 00820 E SINSTI 00000 &lt; SMAXIN e190819143346 00000 Q SMAXIN-1 e190812221505 00750 . UMOY1 E190819144000 000 0 STGE 000B0111 : MSG1 PAS DE MESSAGE &lt; PRM 14200000000022 L RELAIS 000 B NTARF 01 N NJOURF 00 &amp; NJOURF+1 00 B PJOURF+1 00008001 NONUTILE NONUTILE NONUTILE NONUTILE NONUTILE NONUTILE NONUTILE NONUTILE NONUTILE NONUTILE 9</div></div> <div>Les valeurs reçues peuvent différer de ce test</div>								
U1.4	void Linky::DecoderTrame()	Fait le traitement de la TIC pour en extraire les indicateurs								

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
		<b>Les indicateurs sont identifiés :</b> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> VTIC : 2  SINSTS : 809  SINSTI : 0  DATE : 2019-08-19 14:48:09 </div>

### 2.3.1.2 - RAPPORT D'EXÉCUTION

Id.	OK	!OK	Observations
U1.0	*		Pas d'avertissement
U1.1	*		La connexion est bonne
U1.2	*		La demande de réception de la trame a été prise en compte
U1.3	*		Port série libéré
U1.4	*		Les étiquettes des indicateurs sont identifiées

### 2.3.2 - Problèmes rencontrés

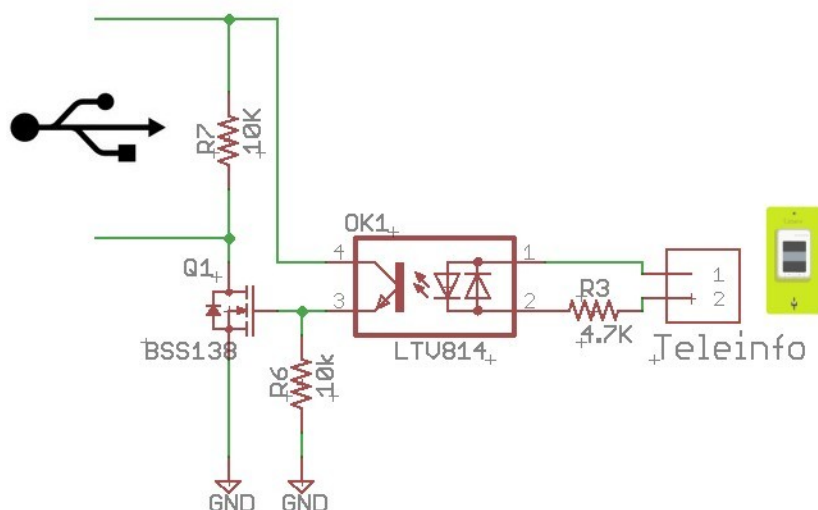
#### 2.3.2.1 - PROBLÈME MATÉRIEL

Il existe deux types de TIC : l'historique et la standard. Les données envoyées sont différentes (notamment le nom des étiquettes), la TIC standard est plus complète et leurs débits sont différents :

- Pour l'historique : 1200 bauds
- Pour la standard : 9600 bauds

La TIC historique ne permet pas de collecter l'injection, alors que la standard le peut, la TIC historique ne peut donc pas être utilisée dans ce projet.

Cependant, le dongle USB  $\mu$ Téléinfo n'est compatible qu'avec la TIC historique, pour la rendre compatible, il faut faire une modification de l'un de ses composants.



D'après l'inventeur du  $\mu$ Téléinfo (son blog disponible ici : <https://community.ch2i.eu/topic/1053/point-resistances-utéléinfo-et-linky-mode-standard>), il faut remplacer R6 (10 k $\Omega$ ) par une résistance de 3,3 k $\Omega$ , la solution retenue est de souder une résistance de 4,7 k $\Omega$  en parallèle :

$$\frac{1}{R} = \frac{1}{10} + \frac{1}{4,7}$$

$$\frac{1}{R} = 0,3127$$

$$R = \frac{1}{0,3127}$$

$$R = 3,197 \text{ k}\Omega$$

La soudure sur un composant CMS étant délicate car il y a un risque de griller d'autres composants, il a finalement été retenu de scotcher la résistance.

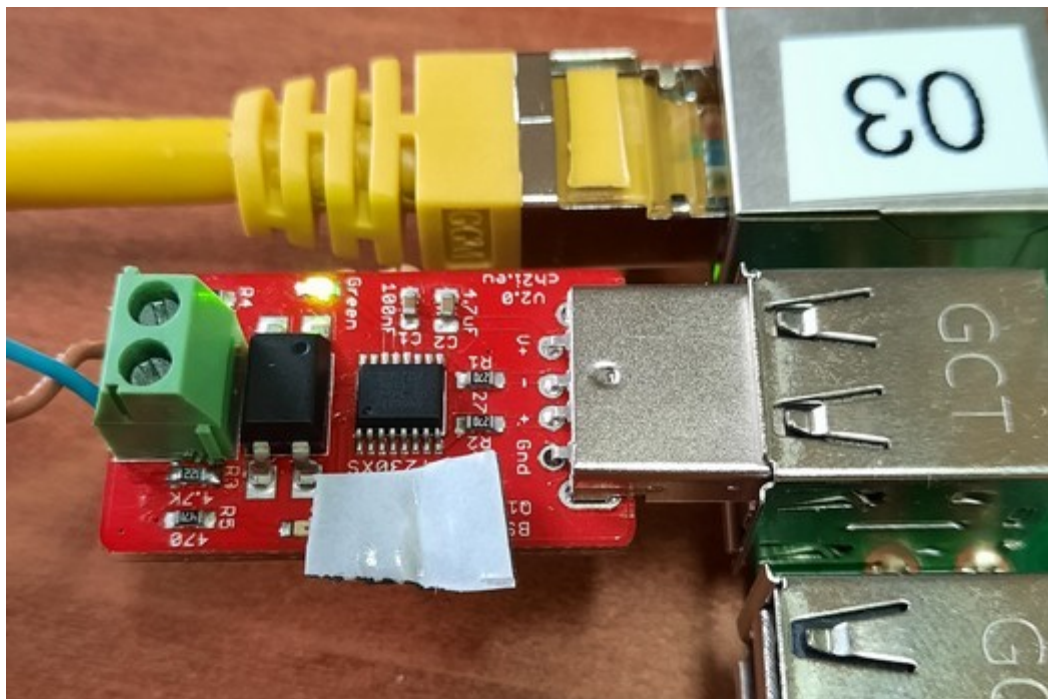


Figure 21: La résistance est scotchée à R6, on peut la voir dépasser sous la carte, près de la DEL verte

### 2.3.2.2 - CHECKSUM NON PRIS EN COMPTE

Le checksum (somme de contrôle) est un calcul effectué pour vérifier d'éventuelles erreurs de transmission.

Table 6 - Information systématiquement horodatée par le compteur

Format d'un groupe contenant une donnée horodatée								
< LF > (0x0A)	Etiquette	< HT > (0x09)	Horodate	< HT > (0x09)	Donnée	< HT > (0x09)	Checksum	< CR > (0x0D)
	Zone contrôlée par la checksum							

Table 7 - Information sans horodate

Format d'un groupe contenant une donnée non horodatée						
< LF > (0x0A)	Etiquette	< HT > (0x09)	Donnée	< HT > (0x09)	Checksum	< CR > (0x0D)
	Zone contrôlée par la checksum					

Le principe de calcul de la Checksum est le suivant :

- calcul de la somme « S1 » de tous les caractères allant du début du champ « Etiquette » jusqu'au délimiteur (inclus) entre les champs « Donnée » et « Checksum »
- cette somme déduite est tronquée sur 6 bits (cette opération est faite à l'aide d'un ET logique avec 0x3F)
- pour obtenir le résultat checksum, on additionne le résultat précédent S2 à 0x20

Cependant, pour appliquer le checksum, il faut récupérer la valeur de chaque caractère ASCII en hexadécimale, malheureusement, la classe QByteArray de Qt (classe dans laquelle est stockée la trame TIC) ne le permet pas.

## Solution

Afin de pallier ce problème, j'effectue un test de l'étiquette VTIC qui doit avoir la valeur 2 : la version de la TIC doit être la seconde.

Le défaut de cette solution est que mon programme n'est compatible qu'avec la TIC standard mais pas avec la TIC historique.

### 3 - Réalisation de la tâche professionnelle Lire les impulsions

#### 3.1 - Présentation de la tâche

La tâche Lire les impulsions consiste à lire les impulsions émises par le compteur SDM120D afin d'en déduire l'index de production.

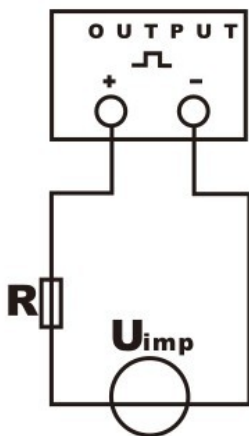
##### 3.1.1 - Le compteur SDM120D

Le SDM120D est un compteur d'énergie numérique à écran LCD. Son manuel d'installation est disponible à l'adresse suivante : <https://docs.vektu.nl/media/eastron/sdm120.pdf>



Figure 22: Compteur SDM120D

Celui-ci met à disposition un circuit transmettant des impulsions :



ATTENTION: Pulse output must be fed as shown in the wiring diagram below. Scrupulously respect polarities and the connection mode. Opto-coupler with potential-free SPST-NO Contact.

Contact range: 5~27VDC Max. current  
Input: 27mA DC.

Figure 23: Caractéristiques techniques du circuit transmettant les impulsions

Il répond à la norme EN 62 053-31 (Tension 12-27V DC/ courant <27mA).

##### 3.1.2 - Le shield BoxEnergie

Le shield BoxEnergie est une carte d'extension pour le Rpi conçue par des élèves de l'école Polytech Nantes. Le BoxEnergie se place sur le GPIO (General Purpose Input Output) du Rpi. Le signal transmis est un signal ToR (Tout ou Rien).

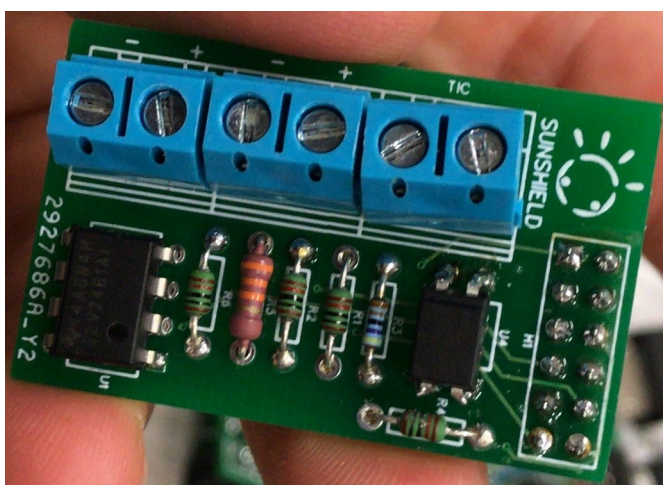


Figure 24: Shield BoxEnergie

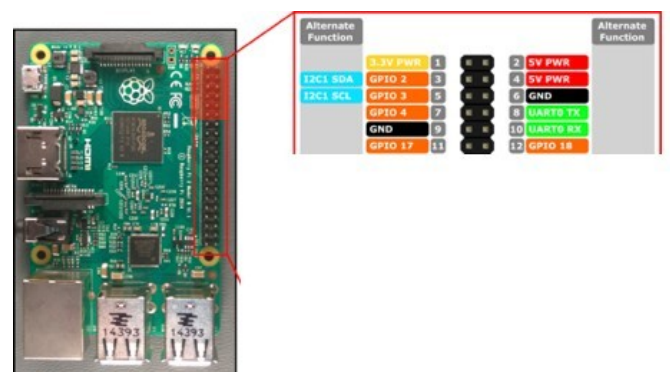
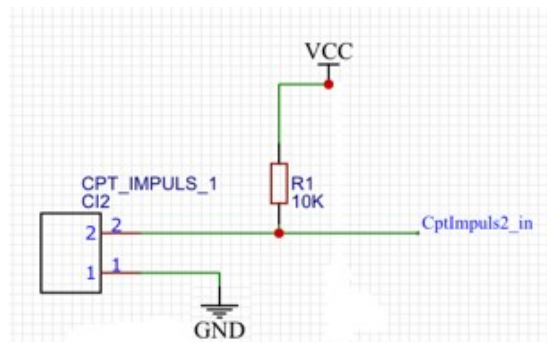


Figure 25: Emplacement d'installation sur le Rpi



Voici le schéma électronique du BoxEnergie :



Le circuit supporte un courant maximal de 27 mA, c'est pourquoi il faut placer une résistance en série afin de limiter le courant. Le GPIO fournit une tension de 3.3V, le calcul de valeur de la résistance peut se faire ainsi :

$$R_{min} = \frac{U}{I} = \frac{3,3}{27 * 10^{-3}} = 122,22 \Omega$$

D'après ce calcul, une résistance de 10 kΩ est largement suffisante.

Cette résistance est en pull up :

- S'il n'y a pas d'impulsion, la sortie vaut VCC (ici 3,3V)
- S'il y a une impulsion, la sortie est reliée à la masse (GND) et vaut donc 0V

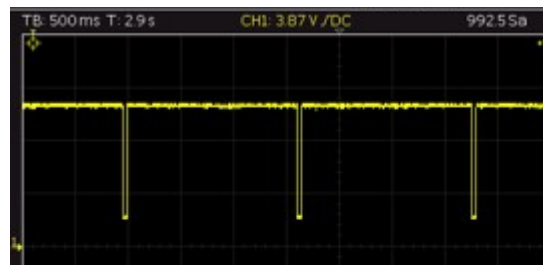


Figure 26: Signal en sortie du BoxEnergie mesuré à l'aide d'un oscilloscope

Les impulsions sont lisibles et donc exploitables.

### 3.1.3 - Bibliothèque logicielles

La bibliothèque utilisée est pigpio (<https://abyz.me.uk/rpi/pigpio/>), comme son nom l'indique, elle permet de contrôler le GPIO du raspberry pi. Elle est très simple à intégrer au programme et est disponible en plusieurs langage : C, Java, Ruby...

## 3.2 - Conception détaillée

### 3.2.1 - Classe CompteurSDM

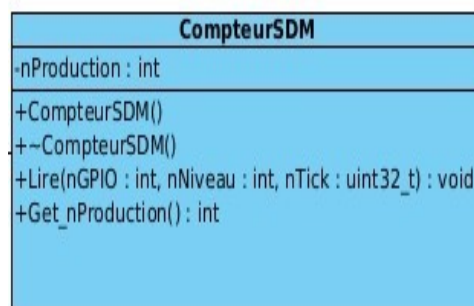


Figure 27: Classe CompteurSDM

La classe ne possède qu'un unique attribut :

- **nProduction**, qui contient l'index de production.

Les méthodes sont les suivantes :

- **Lire**, permet d'incrémenter l'attribut **nProduction** à chaque fois qu'une impulsion est détectée
- **Get\_nProduction**, qui renvoie l'indicateur de production

### 3.2.2 - Constructeur de la classe

Le constructeur permet d'initialiser la classe CompteurSDM.

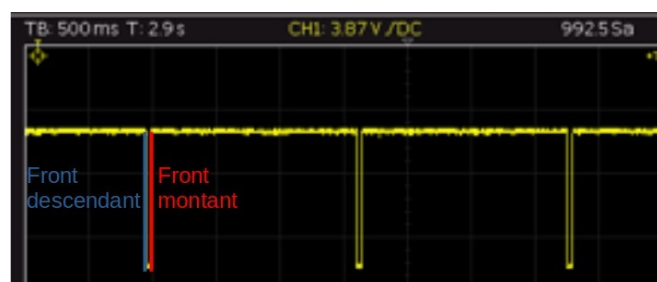
```
CompteurSDM::CompteurSDM()  
{  
    if(gpioInitialise() < 0)  
    {  
        cout << "PIGPIO n'a pas pu etre initialise.\n";  
        exit(-1);  
    }  
  
    if(gpioSetAlertFunc(4, Lire) < 0 )  
    {  
        cout << "L'alerte n'est pas disponible";  
        exit(-1);  
    }  
}
```

La méthode *gpioSetAlertFunc* permet d'appeler la méthode Lire chaque fois qu'il y a un changement d'état sur la broche 4 du GPIO du Rpi, ce qui permet de détecter les impulsions.

### 3.2.3 - Méthode Lire

```
void CompteurSDM::Lire(int gpio, int niveau, uint32_t tick)  
{  
    if (gpio == 4 && niveau == FALLING_EDGE) {  
        nProduction++;  
  
        cout << "Compteur : " << nProduction << " Wh\n";  
    }  
}
```

La méthode *Lire* est appelée à chaque changement d'état, cependant, une impulsion se compose de deux changements d'état, (passage à état bas et passage à état haut), ici, le programme ne détecte que le passage à l'état bas.



### 3.2.4 - Méthode Get\_nProduction

Cette méthode retourne simplement l'attribut de classe **nProduction**.

```
int CompteurSDM::Get_nProduction()
{
    return nProduction;
}
```

**nProduction** est un attribut statique (la méthode *Get\_nProduction* l'est donc également) ce qui permet d'appeler cette méthode sans avoir à créer d'objet.

Pour rappel, c'est la classe Linky qui émet le signal d'envoi des données dans la base de données.

```
emit TraitementTermine(this->nSoutirage, this->nInjection,
CompteurSDM::Get_nProduction(), this->oDate);
```

Le signal est donc émis sans avoir à créer d'instance **CompteurSDM**.

## 3.3 - Tests unitaires

### 3.3.1 - Test unitaire du shield BoxEnergie

Ce test a pour objectif de lire les impulsions émises par le compteur SDM120D en utilisant le

#### 3.3.1.1 - PROCÉDURE DE TEST

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U2.0	Compilation et exécution	cd ~/Documents/LectureCompteur/_build qmake ../LectureCompteur.pro make sudo ./LectureCompteur
		Le projet compile sans avertissement
U2.1	CompteurSDM::CompteurSDM()	La classe est initialisée sans problème
		Aucune erreur ne s'affiche
U2.2	CompteurSDM::Lire()	Les impulsions sont comptabilisées
		<div> Compteur : 1 Wh  Compteur : 2 Wh  Compteur : 3 Wh  Compteur : 4 Wh  Compteur : 5 Wh  Compteur : 6 Wh  Compteur : 7 Wh  Compteur : 8 Wh  Compteur : 9 Wh  Compteur : 10 Wh  Compteur : 11 Wh  Compteur : 12 Wh  Compteur : 13 Wh </div> <p>La valeur affichée est incrémentée à chaque fois que la DEL du compteur SDM120D émet de la lumière.</p>

### 3.3.1.2 - RAPPORT D'EXÉCUTION

Id.	OK	!OK	Observations
U2.0	*		Pas d'avertissement
U2.1	*		Aucune erreur n'est affichée
U2.2	*		Le compte s'affiche bien

### 3.3.2 - Problèmes rencontrés

Aucun problème rencontré.

## 4 - Réalisation de la tâche professionnelle Stocker en local

### 4.1 - Présentation de la tâche

La tâche *Stocker en local* consiste à recevoir les indicateurs de la classe *Linky* et de les envoyer dans la base de données avec des requêtes SQL.

#### 4.1.1 - Bibliothèques utilisées

Les bibliothèques utilisées sont issues de l'API de Qt.

### 4.2 - Conception détaillée

#### 4.2.1 - Diagramme de classe

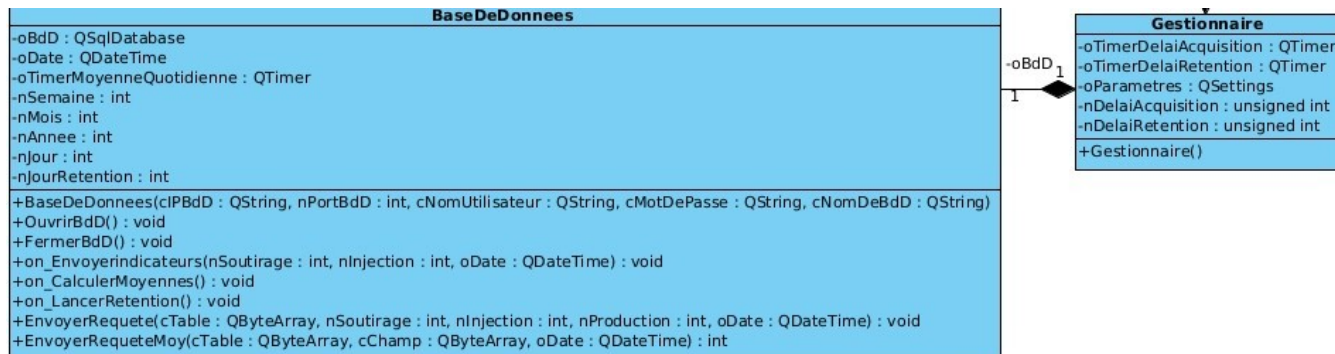


Figure 28: Diagramme de classe pour la communication avec la base de données

La classe **Gestionnaire** envoie les données de configuration à la classe **BaseDeDonnees** avec son attribut **oParametres**.

Les attributs de la classe **BaseDeDonnees** sont les suivants :

- **oBdD** gère les interactions avec la base de données
- **cDate** stocke la date pour l'utiliser lors du calcul des moyennes / rétention
- **oTimer** est un timer lancé quotidiennement, il permet de délimiter l'intervalle entre chaque calcul
- **nSemaine** stock le nombre de jours avant de faire la moyenne hebdomadaire
- **nMois** idem pour la moyenne mensuelle
- **nAnnee** idem pour la moyenne annuelle
- **nJour** stocke le nombre de jours de rétention des données avant l'effacement automatique
- **nJoursRetention** stock le nombre de jours avant la rétention paramétré par l'utilisateur

## 4.2.2 - Constructeur de la classe Gestionnaire

La classe Gestionnaire est le cerveau du programme, elle gère les interactions entre les différentes classes.

L'objet **oParametres** appartient à la classe QSettings de Qt, il permet de lire le contenu d'un fichier (ici au format ini) afin de configurer la connexion à la base de données. Les fichiers sont divisés en sections (ici Connexion et Delais) contenant des paramètres (IP, Port, Acquisition...).

La méthode *connect* permet de connecter l'objet **µTéléinfo** à l'objet **oBdd** : chaque fois que **µTéléinfo** envoie le signal *TraitementTermine*, **oBdd** reçoit les indicateurs dans le slot *on\_EnvoyerIndicateurs*.

```
Gestionnaire::Gestionnaire(QObject *parent):
    QObject(parent),
    // le fichier de configuration doit être placé dans le même dossier que l'exécutable
    oParametres("Sunshare.ini", QSettings::IniFormat),

    /* Rappel des paramètres du port série :
     * 9600 baud
     * 7 bits de donnée
     * Parité paire
     * 1 bit de stop
     */
    uTeleInfo("ttyUSB0", QSerialPort::Baud9600, QSerialPort::Data7, QSerialPort::EvenParity,
    QSerialPort::OneStop, this),

    // les paramètres de la base de donnée sont stockés dans le fichier .ini
    Bdd(oParametres.value("Connexion/IP", "127.0.0.1").toString(),
    oParametres.value("Connexion/Port", 3306).toInt(),
    oParametres.value("Connexion/Utilisateur", "admin").toString(),
    oParametres.value("Connexion/mdp", "admin").toString(),
    oParametres.value("Connexion/NomBdd", "SunshareBDD").toString(),
    oParametres.value("Delais/Retention").toInt() ,
    this),

    // les attributs prennent les valeurs contenues dans le fichier .ini
    nDelaiAcquisition(oParametres.value("Delais/Acquisition", 60).toInt() )// valeur par
    défaut : 60 secondes
{

    oTimerAcquisition.setInterval(nDelaiAcquisition * 1000);// conversion en secondes

    // signal émis pour démarrer les acquisitions
    connect(&oTimerAcquisition, SIGNAL(timeout() ),
            this,          SLOT(on_DemarrerAcquisition() ) );

    // signal émis lors de la fin des acquisitions

    connect(&uTeleInfo, &Linky::TraitementTermine ,
            &Bdd,          &BaseDeDonnees::on_Envoyerindicateurs );

}
```

### 4.2.3 - Constructeur de la classe BaseDeDonnees

Les paramètres de connexion à la base de données sont initialisés ici.

Ces paramètres comprennent :

- L'adresse IP de l'appareil qui héberge la base de données. Comme c'est sur un même RPI que sont collectées et stockées les données, cette adresse correspond à une adresse de loopback (127,0,0,1)
- Le numéro de port d'accès en TCP au SGBD
- Le nom d'utilisateur pour la base de données
- Le mot de passe rattaché à l'utilisateur
- Le nom de la base de données

Le driver de la base de données est initialisé avec la méthode `QsqlDatabase::addDatabase("QMYSQL")`, MariaDB est un embranchement communautaire de MySQL, c'est pourquoi il est nécessaire de charger ce driver. La lettre « Q » spécifie l'appartenance du driver à l'API de Qt.

```
BaseDeDonnees::BaseDeDonnees(QString cIPBdD, int nPortBdD, QString cNomUtilisateur,
                              QString cMotDePasse, QString cNomDeBdD,
                              int nNombreJoursRetention, QObject *parent):
    QObject(parent),
    oBdD(QsqlDatabase::addDatabase("QMYSQL")),

    nSemaine(0),
    nMois(0),
    nAnnee(0),
    nJour(0),
    nJoursRetention(nNombreJoursRetention)
{
    oBdD.setHostName(nIPBdD);
    oBdD.setPort(nPortBdD);

    oBdD.setUserName(cNomUtilisateur);
    oBdD.setPassword(cMotDePasse);

    oBdD.setDatabaseName(cNomDeBdD);

    oTimer.setInterval(1000 * 60 * 60 * 24); // 1 jour

    connect(&oTimer, SIGNAL(timeout() ),
            this,    SLOT(on_CalculerMoyennes() ));

    connect(&oTimer, SIGNAL(timeout() ),
            this,    SLOT(on_LancerRetention() ));
    // le timer est démarré directement
    oTimer.start();

    // affichage du nombre de jours de rétention
    cout << "Retention au bout de " << this->nJoursRetention << " jour" << (this->nJoursRetention>1?"s":"" ) << "\n";
}
```



#### 4.2.4 - Méthodes OuvrirBdD et FermerBdD

Ces méthodes permettent de se connecter et se déconnecter à la base de données

```
void BaseDeDonnees::OuvrirBdD()
{
    if(oBdD.open())
    {
        cout << "Connexion a " << this->oBdD.hostName().toStdString() << " reussie
!\n";
    } else {
        cout << "Erreur lors de la connexion a la BdD : " <<
oBdD.lastError().text().toStdString() << "\n";
        exit(-1);
    }
}
```

```
void BaseDeDonnees::FermerBdD()
{
    cout << "Deconnexion !\n\n";

    oBdD.close();
}
```

#### 4.2.5 - Méthode on EnvoyerIndicateurs

Cette méthode permet d'envoyer les indicateurs à la base de données reçu de la classe Linky.

```
void BaseDeDonnees::on_Envoyerindicateurs(int nSoutirage, int nInjection, QByteArray
cDate)
{
    OuvrirBdD();

    this->cDate = cDate;
    cout << "Envoi des indicateurs\n";

    EnvoyerRequete("Energies_TempsReel", nSoutirage, nInjection, 0, cDate);

    FermerBdD();
}
```

#### 4.2.6 - Méthode EnvoyerRequete

Cette méthode permet d'envoyer les indicateurs dans la base de données en spécifiant la table dans laquelle doit être stockée les données, les indicateurs et la date.

```
void BaseDeDonnees::EnvoyerRequete(QByteArray cTable, int nSoutirage, int nInjection, int
nProduction, QDateTime oDate)
{
    QSqlQuery oRequete (this->oBdD);
    // envoi des valeurs dans la table
    if(!oRequete.prepare("INSERT INTO " + cTable + "(energie_soutiree, energie_injectee,
energie_produite, date) "
                        "VALUES( ?, ?, ?, ?)" )
    {
        cout << "La requete n'a pas pu etre preparee : " <<
oRequete.lastError().text().toString() << "\n";
    }
    oRequete.bindValue(0, nSoutirage);
    oRequete.bindValue(1, nInjection);
    oRequete.bindValue(2, nProduction);
    oRequete.bindValue(3, oDate);

    if(!oRequete.exec() )
    {
        cout << "Erreur lors de l'envoi de la requete : " <<
oRequete.lastError().text().toString() << "\n";
    } else {
        cout << "Requete envoyee.\n";
    }
}
```

Cette méthode utilise la requête SQL **INSERT INTO**, elle permet d'insérer de nouvelles données dans une table.

La méthode *bindValue* permet de remplacer un « ? » de la requête SQL par la valeur d'une variable. Par exemple, le premier « ? » (numéro 0) prend la valeur de la variable nSoutirage.

#### 4.2.7 - Méthode on\_CalculerMoyennes

Cette méthode permet de lancer les calculs des moyennes et les envoie dans la base de données avec la méthode *EnvoyerRequete*. Cette méthode gère toutes les moyennes en même temps. Afin de ne pas trop rallonger le rapport, seul la moyenne quotidienne sera étudiée car le principe est le même.

```
void BaseDeDonnees::on_CalculerMoyennes()
{
    cout << "Calcul des moyennes. " << this->oDate.toString(FormatDate).toStdString() << "\n";
    // variable qui contient la date à partir de laquelle commence la moyenne
    QDateTime oDate;
    // variables qui contiennent les moyennes
    int nSoutirageMoy (0);
    int nInjectionMoy (0);
    int nProductionMoy (0);

    // 1 jour passe : on incrémente les comptes
    this->nSemaine++;
    this->nMois++;
    this->nAnnee++;

    ConnexionBdD();

    //*****

    cout << "Moyenne journaliere\n";
    oDate = this->oDate.addDays(-1); // On prend la date par rapport à la veille

    cout << "La moyenne sera faite de " << oDate.toString(FormatDate).toStdString() << " a
aujourd'hui.\n";

    // récupération de la moyenne de soutirage
    nSoutirageMoy = EnvoyerRequeteMoy("Energies_TempsReel", "energie_soutiree", oDate);
    // récupération de la moyenne d'injection
    nInjectionMoy = EnvoyerRequeteMoy("Energies_TempsReel", "energie_injectee", oDate);
    // récupération de la moyenne de production
    nProductionMoy = EnvoyerRequeteMoy("Energies_TempsReel", "energie_produite", oDate);

    // envoi des valeurs dans la table journalière
    EnvoyerRequete("Energies_Journaliere", nSoutirageMoy, nInjectionMoy, nProductionMoy,
this->oDate);

    //*****

    FermerBdD();
}
```

Pour afficher une date utilisant la classe *QdateTime*, il faut préciser un format (variable *FormatDate*), le format que j'utilise est le suivant : AAAA-MM-JJ hh:mm:ss. La classe *QdateTime* gère très bien la date : si l'on décrémente le jour alors que nous sommes le dernier mois, le mois est alors décrémente et le jour devient le dernier de celui-ci.

Les attributs **nSemaine**, **nMois**, **nAnnee** sont incrémentés à chaque fois que le slot *on\_CalculerMoyennes* (celle-ci est appelée tous les jours). Le calcul des moyennes hebdomadaires n'est lancé que lorsque **nSemaine** a atteint 7, la moyenne mensuelle lorsque **nMois** a atteint 30 et la moyenne annuelle lorsque **nAnnee** a atteint 365. Si les calculs ont bien été réalisés, Les attributs sont réinitialisés afin de réaliser les prochains calculs.

#### 4.2.8 - Méthode EnvoyerRequeteMoy

Cette méthode est utilisée à plusieurs reprises par le slot *on\_CalculerMoyennes* et permet d'envoyer une requête à la base de données afin de récupérer la moyenne d'un champ dans une table précise.

```
int BaseDeDonnees::EnvoyerRequeteMoy(QByteArray cTable, QByteArray cChamp, QDateTime
oDate)
{
    int nMoyenne (0);
    QSqlQuery oRequete (this->oBdD);

    // récupération de la moyenne pour une date supérieur à celle donnée : on fait
    la moyenne de cDate à aujourd'hui
    oRequete.prepare("SELECT AVG(" + cChamp + ") FROM " + cTable + " WHERE date
    > ?");
    oRequete.bindValue(0, oDate);

    if(!oRequete.exec() )
    {
        cout << "Erreur lors de l'envoi de la requete : " <<
oRequete.lastError().text().toString() << "\n";
    }
    while(oRequete.next())
    {
        nMoyenne = oRequete.value(0).toInt();
    }
    cout << "Valeur de " << cChamp.toString() << " : " << nMoyenne << "\n";
    return nMoyenne;
}
```

La requête SQL utilisée est **SELECT AVG**. Elle renvoie directement la valeur moyenne.

La valeur de retour de la méthode est la moyenne calculée par la requête.

#### 4.2.9 - Méthode on\_LancerRetention

Cette méthode gère la rétention des données.

```
void BaseDeDonnees::on_LancerRetention()
{
    QDateTime oDate;

    this->nJour++;

    // rétention : suppression des anciennes données
    if(this->nJour == this->nJoursRetention) // si on atteint le nombre de jours pour lancer
la rétention
    {
        cout << "Debut de la retention\n";
        oDate = this->oDate.addDays( - this->nJoursRetention); // on décrémente du nombre
paramétré par l'utilisateur

        cout << "Les donnees anterieurs a " << oDate.toString(FormatDate).toString() << "
seront supprimees.\n";

        ConnexionBdD();

        QSqlQuery oRequete (this->oBdD); // objet permettant d'envoyer des requêtes SQL
        if(!oRequete.prepare("DELETE FROM Energies WHERE date < ?") ) // on détruit les lignes
comportant les dates antérieurs
        {
            cout << "La requete n'a pas pu etre preparee : " <<
oRequete.lastError().text().toString() << "\n"; // s'il y a un problème (syntaxe)
        } else {
            cout << "Requete envoyee.\n\n";
        }
        oRequete.bindValue(0, oDate);

        if(!oRequete.exec() )
        {
            cout << "Erreur lors de l'envoi de la requete : " <<
oRequete.lastError().text().toString() << "\n";
        }

        FermerBdD();
    }
}
```

La requête SQL utilisée est **DELETE**, qui permet de supprimer des données d'une base de données.

La table « Energies » est une table fictive utilisée uniquement pour tester ce code, afin d'éviter de détruire de réels valeurs. Celle-ci sera modifiée en « Energies\_TempsReel » lors de la livraison au client.

## 4.3 - Tests unitaires

### 4.3.1 - Test unitaire du fichier d'initialisation

Ce test a pour but de lire la configuration de la connexion à la base de données à partir d'un fichier .ini.

#### 4.3.1.1 - PROCÉDURE DE TEST

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U2.0	Compilation et exécution	<pre>cd ~/Documents/TestQsetting/_build qmake ../TestQsetting.pro make ./TestQsetting</pre> <p>Le projet compile sans avertissement</p>
U2.1	cat Sunshare.ini	<p>Le contenu du fichier d'initialisation est affiché</p> <pre>[Connexion] # Adresse IP IP=127.0.0.1 # Numéro de port Port=3306 # Nom d'utilisateur Utilisateur=admin # Mot de passe mdp=admin # Nom de la base de données NomBdD=SunshareBDD  [Delais] # Délai d'acquisition en secondes Acquisition=60 # Délai avant la suppression des anciennes données en jours Retention=20</pre>
U2.2	Gestionnaire::Gestionnaire()	<p>Le contenu du fichier Sunshare.ini est lu</p> <pre>Test de QSettings Groupe [Connexion] IP de la BdD : 127.0.0.1 Port : 3306 Nom d'utilisateur : admin Mot de passe : admin Mot de passe : SunshareBDD Groupe [Delais] Délai d'acquisition : 60 Délai de retention : 20</pre>

### 4.3.1.2 - RAPPORT D'EXÉCUTION

Id.	OK	!OK	Observations
U2.0	*		Pas d'avertissement
U2.1	*		Le contenu du fichier s'affiche
U2.2	*		Le contenu du fichier est lu par le programme

### 4.3.2 - Problèmes rencontrés

Aucun problème rencontré.

### 4.3.3 - Test unitaire de l'envoi des indicateurs de soutirage et d'injection à la base de données

Ce test a pour but d'envoyer les indicateurs collectés dans la TIC et de les envoyer dans la base de données.

#### 4.3.3.1 - PROCÉDURE DE TEST

Id.	Méthode testée Description Sommaire	Procédure de test								
		Résultats attendus								
U3.0	Compilation et exécution	cd ~/Documents/TestBdD/_build qmake ../TestBdD.pro make ./TestBdD								
		Le projet compile sans avertissement								
U3.1	Gestionnaire::Gestionnaire() Linky::Linky() BaseDeDonnees::BaseDeDonnes()	Les objets sont construits sans problème								
		Les informations de la base de données sont correctement renseignées <table><tr><td>IP :</td><td>127.0.0.1</td></tr><tr><td>Port :</td><td>3306</td></tr><tr><td>Utilisateur :</td><td>admin</td></tr><tr><td>Mot de passe :</td><td>admin</td></tr><tr><td>Nom de la base de données :</td><td>SunshareBDD</td></tr></table>	IP :	127.0.0.1	Port :	3306	Utilisateur :	admin	Mot de passe :	admin
IP :	127.0.0.1									
Port :	3306									
Utilisateur :	admin									
Mot de passe :	admin									
Nom de la base de données :	SunshareBDD									
U3.2	void Linky::TraitementTermine()	L'intégration 1 est déjà fonctionnelle, les indicateurs sont envoyés à la classe BaseDeDonnees								
		La classe BaseDeDonnees reçoit les données <div>Debut de l'acquisition Fin de l'acquisition Traitement de la TIC. VTIC : 2 SINSTS : 808 SINSTI : 0 DATE : 2019-08-19 14:53:50</div> <p>Les valeurs reçues peuvent différer de ce test</p>								
U3.3	void BaseDeDonnees::ConnexionBdD()	La méthode demande une connexion à la base de données								
		La connexion se fait sans erreur <div>Connexion a 127.0.0.1 reussie !</div>								
U3.4	void BaseDeDonnees::on_EnvoyerIndicateurs()	Une requête SQL est envoyée à la base de données								



Id.	Méthode testée Description Sommaire	Procédure de test																				
		Résultats attendus																				
		<div>La base de données reçoit les données (voir avec phpMyAdmin)</div> <div><div>Table Energies TempsReel avant l'envoi :</div><table><tr><th>id_Energies_TempsReel</th><th>date</th><th>energie_produite</th><th>energie_soutiree</th><th>energie_injectee</th></tr><tr><td colspan="5">Opérations sur les résultats de la requête</td></tr></table></div> <div><div>Table Energies TempsReel après l'envoi :</div><table><tr><th>id_Energies_TempsReel</th><th>date</th><th>energie_produite</th><th>energie_soutiree</th><th>energie_injectee</th></tr><tr><td>96</td><td>2019-08-19 14:53:50</td><td>0</td><td>808</td><td>0</td></tr></table></div>	id_Energies_TempsReel	date	energie_produite	energie_soutiree	energie_injectee	Opérations sur les résultats de la requête					id_Energies_TempsReel	date	energie_produite	energie_soutiree	energie_injectee	96	2019-08-19 14:53:50	0	808	0
id_Energies_TempsReel	date	energie_produite	energie_soutiree	energie_injectee																		
Opérations sur les résultats de la requête																						
id_Energies_TempsReel	date	energie_produite	energie_soutiree	energie_injectee																		
96	2019-08-19 14:53:50	0	808	0																		
U3.5	Void BaseDeDonnees::FermerBdD()	<div>La méthode demande la fermeture de la connexion à la base de données</div> <div>La connexion à la base de données est fermée</div>																				

#### 4.3.3.2 - RAPPORT D'EXÉCUTION

Id.	OK	!OK	Observations
U3.0	*		Pas d'avertissement
U3.1	*		La connexion est bonne
U3.2	*		La demande de réception de la trame a été prise en compte
U3.3	*		La connexion à la base de données est réussie
U3.4	*		Les données sont présentes dans la base de données
U3.5	*		La connexion à la base de données est fermée

#### 4.3.4 - Problèmes rencontrés

Aucun problème rencontré.

#### 4.3.5 - Test unitaire des calculs de moyennes

Ce test met en valeur le fonctionnement des calculs des moyennes journalières, hebdomadaires, mensuelles et annuelles des indicateurs.

##### 4.3.5.1 - PROCÉDURE DE TEST

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U4.0	Compilation et exécution	<pre>cd ~/Documents/TestMoyBdD/_build qmake ../TestMoyBdD.pro make ./TestMoyBdD</pre> <p>Le projet compile sans avertissement</p>
U4.1	void BaseDeDonnees::on_CalculerMoyennes()	Le programme lance des requêtes pour collecter les moyennes

Id.	Méthode testée Description Sommaire	Procédure de test																														
		Résultats attendus																														
		<div>Les moyennes sont collectées :</div> <div><pre>pi@raspberrypi:/opt/CalculMoyBdD/bin \$ ./CalculMoyBdD Calcul des moyennes. 2019-08-09 20:37:27 Connexion a 127.0.0.1 reussie ! Moyenne journaliere La moyenne sera faite de 2019-08-8 20:37:27 a aujourd'hui. Valeur de energie_soutiree : 655 Valeur de energie_injectee : 0 Valeur de energie_produite : 0 Requete envoyee.</pre></div> <div>Les valeurs reçues peuvent différer de ce test</div>																														
U4.2	Réception dans la base de données	<div>La base de données reçoit les données</div> <table><tr><th>id_Energies_Journaliere</th><th>date</th><th>energie_produite</th><th>energie_soutiree</th><th>energie_injectee</th></tr><tr><td>52</td><td>2019-08-09</td><td>0</td><td>655</td><td>0</td></tr></table>	id_Energies_Journaliere	date	energie_produite	energie_soutiree	energie_injectee	52	2019-08-09	0	655	0																				
id_Energies_Journaliere	date	energie_produite	energie_soutiree	energie_injectee																												
52	2019-08-09	0	655	0																												
U4.3	Les valeurs correspondent	<div>La moyenne calculée est bien la bonne</div> <table><tr><th>id_Energies_TempsReel</th><th>date</th><th>energie_produite</th><th>energie_soutiree</th><th>energie_injectee</th></tr><tr><td>93</td><td>2019-08-09 20:36:46</td><td>0</td><td>583</td><td>0</td></tr><tr><td>92</td><td>2019-08-09 20:36:35</td><td>0</td><td>631</td><td>0</td></tr><tr><td>91</td><td>2019-08-09 20:36:25</td><td>0</td><td>682</td><td>0</td></tr><tr><td>90</td><td>2019-08-09 20:36:16</td><td>0</td><td>722</td><td>0</td></tr><tr><td>89</td><td>2019-08-05 21:37:02</td><td>0</td><td>718</td><td>0</td></tr></table> <div><math display="block">\frac{583+631+682+722}{4}=654,5</math></div>	id_Energies_TempsReel	date	energie_produite	energie_soutiree	energie_injectee	93	2019-08-09 20:36:46	0	583	0	92	2019-08-09 20:36:35	0	631	0	91	2019-08-09 20:36:25	0	682	0	90	2019-08-09 20:36:16	0	722	0	89	2019-08-05 21:37:02	0	718	0
id_Energies_TempsReel	date	energie_produite	energie_soutiree	energie_injectee																												
93	2019-08-09 20:36:46	0	583	0																												
92	2019-08-09 20:36:35	0	631	0																												
91	2019-08-09 20:36:25	0	682	0																												
90	2019-08-09 20:36:16	0	722	0																												
89	2019-08-05 21:37:02	0	718	0																												

#### 4.3.5.2 - RAPPORT D'EXÉCUTION

Id.	OK	!OK	Observations
U4.0	*		Pas d'avertissement
U4.1	*		Les calculs sont réalisés
U4.2	*		La base de données reçoit bien les données
U4.3	*		Les valeurs sont bien les bonnes

#### 4.3.6 - Problèmes rencontrés

##### 4.3.6.1 - FONCTIONNEMENT

Dans l'idéal, les calculs des moyennes doivent se faire au plus tard dans la journée, ainsi, pour la moyenne quotidienne, le calcul doit être fait vers 23h, pour bien prendre en compte les valeurs de la journée entière.

La difficulté avec cette manière de faire est lors de la moyenne annuelle : la méthode utilisée pour créer l'intervalle de temps est **void setInterval(int msec)**, qui prend donc l'intervalle de temps en millisecondes, sauf que convertir 1 an en millisecondes dépasse largement la limite de la plage de données ( $2.147.483.647$  ou  $2^{31}-1$ ) et limite ainsi le nombre de jours maximal pour le timer à 24 jours.

$$\frac{2^{31} - 1}{24 * 60^2 * 1000} \simeq 24,85 \text{ jours}$$

Il faudrait alors que de nouveaux timers soient lancés dès que d'autres se terminent, ce qui est complexe à gérer.

La solution retenue pour mon programme est donc de calculer les moyennes par rapport à la date exacte de la veille. Par exemple, pour la moyenne quotidienne, si l'on est le 13 mai 2022 à 10:30:12, la moyenne commencera à partir du 12 mai 2022, à la même heure. Cette manière de faire pose problème dans le cas où il y aurait une coupure de courant ou que le programme doit être arrêté, le timer est remis à 0 lors du lancement du programme. Un autre problème étant qu'une partie des valeurs utilisées dans la moyenne provient de la veille, ce qui ne correspond pas vraiment à une « moyenne quotidienne ».

#### 4.3.6.2 - GESTION DE LA DATE DIFFICILE

La gestion de la date est assez complexe : par exemple, pour savoir quel jour était la veille, il faut soustraire 1 au nombre de jours. Cependant, il faut vérifier si l'on est le premier jour du mois, ou encore le premier mois de l'année, le tout dans une chaîne de caractères dont il faut convertir en nombre pour faire le moindre calcul ou encore des conditions.

Une alternative intéressante aurait été d'utiliser le timestamp, un système d'horodatage mis en place depuis le 1er janvier 1970, celui-ci compte les secondes écoulées depuis cette date. Celui-ci a pour avantage de comparer facilement les dates en faisant une simple opération de différence.

1398729600		4/29/2014
1389916800		1/17/2014
1433980800		6/11/2015
1392854400	→	2/20/2014
1430438400		5/1/2015
1432857600		5/29/2015
1428624000		4/10/2015
1395014400		3/17/2014

Figure 29: Exemples de timestamp

MariaDB gère très bien le timestamp, et Qt de même avec les classes Qdate, Qtime.

Cependant, un problème du timestamp se pose étant donné que l'OS utilisé dans ce projet est Raspbian OS version 32 bits, ce qui signifie que le timestamp maximal pouvant être lu est  $2.147.483.647$  ( $2^{31}-1$ ), correspondant à la date du 19 janvier 2038 à 3 h 14 min 7 s. Dépassé cette date, le Rpi ne pourra plus déchiffrer le timestamp, ce qui posera problème à long terme.

#### 4.3.7 - Test unitaire de la suppression automatique

Afin d'éviter de supprimer de réels données, ce test est réalisé dans une table contenant uniquement de fausses valeurs.

##### 4.3.7.1 - PROCÉDURE DE TEST

Id.	Méthode testée Description Sommaire	Procédure de test
		Résultats attendus
U5.0	Compilation	cd ~/Documents/TestRetention/_build qmake TestRetention.pro make ./TestRetention.pro
		Le projet compile sans avertissement

Id.	Méthode testée Description Sommaire	Procédure de test																																																																																																																																																																
		Résultats attendus																																																																																																																																																																
U5.1	void BaseDeDonnees::on_LancerRetention()	La méthode envoie une requête SQL pour supprimer les anciennes données																																																																																																																																																																
		Aucun problème n'est affiché <div><pre>pi@raspberrypi:/opt/CalculMoyBdD/bin \$ ./CalculMoyBdD Retention au bout de 1 jour Calcul des moyennes. 2022-03-23 11:55:00 Connexion a 127.0.0.1 reussie !  Debut de la retention Les donnees anterieurs a 2022-03-22 11:55:00 seront supprimees. Requete envoyee.</pre></div>																																																																																																																																																																
U5.2	Suppression des données	Seul les données trop anciennes sont supprimées (voir phpMyAdmin)																																																																																																																																																																
		<div>Avant la suppression :</div> <div>La partie encadrée en rouge sera supprimée</div> <table><tr><th>id</th><th>id_utilisateur</th><th>date</th><th>energie_produite</th><th>energie_soutiree</th><th>energie_injectee</th></tr><tr><td>32</td><td>NULL</td><td>2022-03-25 09:29:02</td><td>30.2</td><td>1.2</td><td>21.2</td></tr><tr><td>31</td><td>NULL</td><td>2022-03-25 09:28:35</td><td>30.2</td><td>20.2</td><td>21.2</td></tr><tr><td>28</td><td>NULL</td><td>2022-03-22 12:11:16</td><td>20</td><td>121</td><td>220</td></tr><tr><td>27</td><td>NULL</td><td>2022-03-22 12:11:00</td><td>20</td><td>121</td><td>220</td></tr><tr><td>26</td><td>NULL</td><td>2022-03-22 11:56:40</td><td>50</td><td>100</td><td>20</td></tr><tr><td>25</td><td>NULL</td><td>2022-03-22 11:51:55</td><td>50</td><td>100</td><td>20</td></tr><tr><td>24</td><td>NULL</td><td>2022-03-22 11:03:41</td><td>50</td><td>100</td><td>20</td></tr><tr><td>23</td><td>NULL</td><td>2022-03-22 11:03:00</td><td>50</td><td>100</td><td>20</td></tr><tr><td>22</td><td>NULL</td><td>2022-03-18 10:33:59</td><td>12.21</td><td>44.1</td><td>45.1</td></tr><tr><td>21</td><td>NULL</td><td>2022-03-08 15:24:02</td><td>10.2</td><td>25.1</td><td>11.3</td></tr><tr><td>20</td><td>NULL</td><td>2022-03-08 15:22:02</td><td>66.2</td><td>71.1</td><td>42.3</td></tr><tr><td>19</td><td>NULL</td><td>2022-03-08 15:21:23</td><td>36.2</td><td>21.1</td><td>22.3</td></tr><tr><td>18</td><td>NULL</td><td>2022-03-08 15:16:15</td><td>66.2</td><td>51.1</td><td>12.3</td></tr><tr><td>17</td><td>NULL</td><td>2022-03-08 15:14:45</td><td>56.2</td><td>21.1</td><td>22.3</td></tr><tr><td>16</td><td>NULL</td><td>2022-03-08 14:50:59</td><td>16.2</td><td>31.1</td><td>12.3</td></tr><tr><td>15</td><td>NULL</td><td>2022-03-04 17:26:57</td><td>22.4</td><td>45.8</td><td>24.5</td></tr><tr><td>13</td><td>NULL</td><td>2022-02-25 14:33:57</td><td>64.5</td><td>34.5</td><td>45.2</td></tr><tr><td>12</td><td>NULL</td><td>2022-02-25 14:32:48</td><td>54.5</td><td>24.5</td><td>35.2</td></tr><tr><td>11</td><td>NULL</td><td>2022-02-22 13:57:53</td><td>39.4</td><td>16.1</td><td>46.1</td></tr><tr><td>10</td><td>NULL</td><td>2022-02-22 13:56:45</td><td>62.4</td><td>54.1</td><td>42.1</td></tr></table> <div>Après la suppression :</div> <table><tr><th>id</th><th>id_utilisateur</th><th>date</th><th>energie_produite</th><th>energie_soutiree</th><th>energie_injectee</th></tr><tr><td>32</td><td>NULL</td><td>2022-03-25 09:29:02</td><td>30.2</td><td>1.2</td><td>21.2</td></tr><tr><td>31</td><td>NULL</td><td>2022-03-25 09:28:35</td><td>30.2</td><td>20.2</td><td>21.2</td></tr><tr><td>28</td><td>NULL</td><td>2022-03-22 12:11:16</td><td>20</td><td>121</td><td>220</td></tr><tr><td>27</td><td>NULL</td><td>2022-03-22 12:11:00</td><td>20</td><td>121</td><td>220</td></tr><tr><td>26</td><td>NULL</td><td>2022-03-22 11:56:40</td><td>50</td><td>100</td><td>20</td></tr></table> <div>ction :  Éditer  Copier  Supprimer  Exporter</div>	id	id_utilisateur	date	energie_produite	energie_soutiree	energie_injectee	32	NULL	2022-03-25 09:29:02	30.2	1.2	21.2	31	NULL	2022-03-25 09:28:35	30.2	20.2	21.2	28	NULL	2022-03-22 12:11:16	20	121	220	27	NULL	2022-03-22 12:11:00	20	121	220	26	NULL	2022-03-22 11:56:40	50	100	20	25	NULL	2022-03-22 11:51:55	50	100	20	24	NULL	2022-03-22 11:03:41	50	100	20	23	NULL	2022-03-22 11:03:00	50	100	20	22	NULL	2022-03-18 10:33:59	12.21	44.1	45.1	21	NULL	2022-03-08 15:24:02	10.2	25.1	11.3	20	NULL	2022-03-08 15:22:02	66.2	71.1	42.3	19	NULL	2022-03-08 15:21:23	36.2	21.1	22.3	18	NULL	2022-03-08 15:16:15	66.2	51.1	12.3	17	NULL	2022-03-08 15:14:45	56.2	21.1	22.3	16	NULL	2022-03-08 14:50:59	16.2	31.1	12.3	15	NULL	2022-03-04 17:26:57	22.4	45.8	24.5	13	NULL	2022-02-25 14:33:57	64.5	34.5	45.2	12	NULL	2022-02-25 14:32:48	54.5	24.5	35.2	11	NULL	2022-02-22 13:57:53	39.4	16.1	46.1	10	NULL	2022-02-22 13:56:45	62.4	54.1	42.1	id	id_utilisateur	date	energie_produite	energie_soutiree	energie_injectee	32	NULL	2022-03-25 09:29:02	30.2	1.2	21.2	31	NULL	2022-03-25 09:28:35	30.2	20.2	21.2	28	NULL	2022-03-22 12:11:16	20	121	220	27	NULL	2022-03-22 12:11:00	20	121	220	26	NULL	2022-03-22 11:56:40	50
id	id_utilisateur	date	energie_produite	energie_soutiree	energie_injectee																																																																																																																																																													
32	NULL	2022-03-25 09:29:02	30.2	1.2	21.2																																																																																																																																																													
31	NULL	2022-03-25 09:28:35	30.2	20.2	21.2																																																																																																																																																													
28	NULL	2022-03-22 12:11:16	20	121	220																																																																																																																																																													
27	NULL	2022-03-22 12:11:00	20	121	220																																																																																																																																																													
26	NULL	2022-03-22 11:56:40	50	100	20																																																																																																																																																													
25	NULL	2022-03-22 11:51:55	50	100	20																																																																																																																																																													
24	NULL	2022-03-22 11:03:41	50	100	20																																																																																																																																																													
23	NULL	2022-03-22 11:03:00	50	100	20																																																																																																																																																													
22	NULL	2022-03-18 10:33:59	12.21	44.1	45.1																																																																																																																																																													
21	NULL	2022-03-08 15:24:02	10.2	25.1	11.3																																																																																																																																																													
20	NULL	2022-03-08 15:22:02	66.2	71.1	42.3																																																																																																																																																													
19	NULL	2022-03-08 15:21:23	36.2	21.1	22.3																																																																																																																																																													
18	NULL	2022-03-08 15:16:15	66.2	51.1	12.3																																																																																																																																																													
17	NULL	2022-03-08 15:14:45	56.2	21.1	22.3																																																																																																																																																													
16	NULL	2022-03-08 14:50:59	16.2	31.1	12.3																																																																																																																																																													
15	NULL	2022-03-04 17:26:57	22.4	45.8	24.5																																																																																																																																																													
13	NULL	2022-02-25 14:33:57	64.5	34.5	45.2																																																																																																																																																													
12	NULL	2022-02-25 14:32:48	54.5	24.5	35.2																																																																																																																																																													
11	NULL	2022-02-22 13:57:53	39.4	16.1	46.1																																																																																																																																																													
10	NULL	2022-02-22 13:56:45	62.4	54.1	42.1																																																																																																																																																													
id	id_utilisateur	date	energie_produite	energie_soutiree	energie_injectee																																																																																																																																																													
32	NULL	2022-03-25 09:29:02	30.2	1.2	21.2																																																																																																																																																													
31	NULL	2022-03-25 09:28:35	30.2	20.2	21.2																																																																																																																																																													
28	NULL	2022-03-22 12:11:16	20	121	220																																																																																																																																																													
27	NULL	2022-03-22 12:11:00	20	121	220																																																																																																																																																													
26	NULL	2022-03-22 11:56:40	50	100	20																																																																																																																																																													

#### 4.3.7.2 - RAPPORT D'EXÉCUTION

Id.	OK	!OK	Observations
U5.0	*		Pas d'avertissement
U5.1	*		Le programme s'exécute sans problème
U5.2	*		Les données trop anciennes sont bien supprimées

### 5 - Bilan de la réalisation personnelle

#### 5.1 - Statut des fonctions à charge

Application Qt	Lecture des indicateurs	Lecture du soutirage et injection	Terminé
		Lecture de la production	Terminé
	Base de données	Envoi des indicateurs	Terminé
		Lecture de la configuration dans un fichier	Terminé
		Calcul des moyennes	Terminé
		Suppression des anciennes données	Terminé

#### 5.2 - Points à améliorer

Le moment du calcul des moyennes n'est pas idéal, il faudrait mettre au point un moyen de déclencher les calculs au plus tard dans la journée.

Le BoxEnergie est censé interpréter les informations transmises par le Linky (TIC), cependant, celui-ci ne serait pas compatible avec le Linky mis à ma disposition qui est de la marque Sagemcom.

#### 5.3 - Points positifs

Application des méthodes agiles.

Richesse de la gestion et du travail d'équipe.

Approfondissement et amélioration des connaissances dans le développement d'application en C++ et de l'API Qt.