

REVE : REcharge de Véhicule Électrique

Projet de fin d'année de BTS SNIR



Entreprise : EM-SOLAR, 15130 Ytrac

Etablissement : Lycée Jean Monnet Mermoz Aurillac

Professeur : Kurdzielewicz Michel

Etudiants : Agnel Dorian | Luka Mayonade | Lucas Serrée-Delpeux

SOMMAIRE

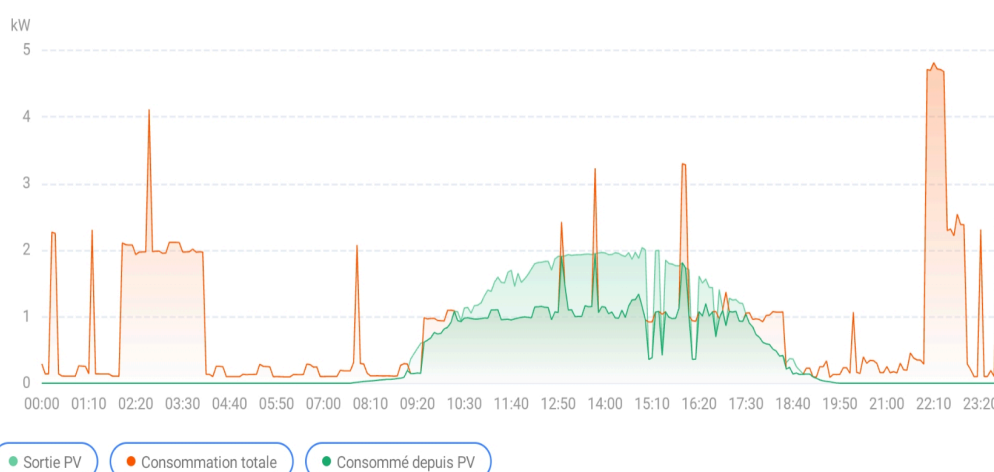
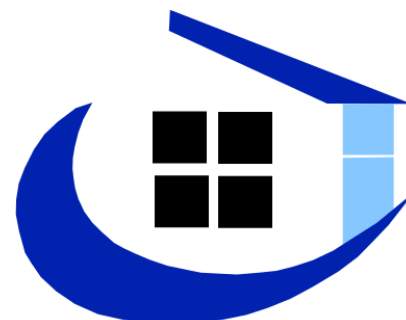
| | |
|---|----|
| I-Introduction..... | 4 |
| Présentation générale du système supportant le projet :..... | 4 |
| Projet :..... | 5 |
| Expression du besoin :..... | 5 |
| Solutions envisagées pour répondre aux besoins :..... | 7 |
| Synoptique correspondant :..... | 8 |
| Contraintes technologiques :..... | 8 |
| Architecture du système :..... | 9 |
| Architecture logicielle du système :..... | 12 |
| Répartition des tâches..... | 13 |
| Partie étudiant Luka..... | 14 |
| Introduction..... | 14 |
| Analyse..... | 14 |
| Développement..... | 17 |
| Tests..... | 18 |
| Conclusion..... | 19 |
| Partie étudiant Dorian..... | 20 |
| Introduction..... | 20 |
| Accès à la ZDC..... | 20 |
| MQTT..... | 22 |
| Exemple de la publication de la puissance:..... | 26 |
| Exemple de la réception du topic dépassement:..... | 26 |
| Tests..... | 28 |
| Difficultés rencontrées..... | 30 |
| Conclusion..... | 30 |
| Partie étudiant Lucas..... | 31 |
| Objectif..... | 31 |
| Analyse de la partie..... | 31 |
| Développement..... | 35 |
| Tests individuels..... | 38 |
| Test d'intégration..... | 38 |
| Conclusion..... | 38 |
| Conclusion générale..... | 39 |
| Annexes..... | 40 |
| Annexe 1 : Installation bibliothèque MQTT avec Paho..... | 40 |
| Annexe 2 : Documentation MQTT..... | 41 |
| Annexe 3 : Installation sur Linux du broker MQTT Mosquitto..... | 43 |
| Annexe 4 : Installation sur Windows du broker MQTT Mosquitto..... | 44 |
| Annexe 5 : Documentation ModBus..... | 45 |
| Annexe 6 : Documentation ModBus TCP pour WallBox..... | 46 |

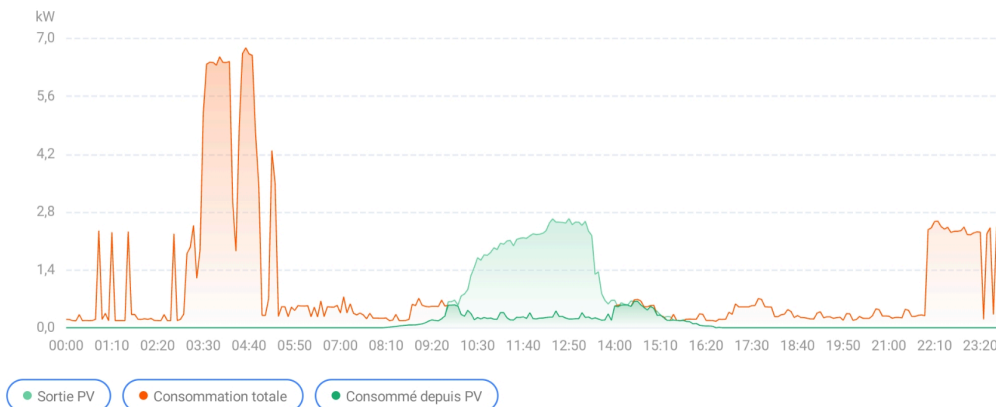
| | |
|---|----|
| Annexe 7 : Numéro d'esclave..... | 47 |
| Annexe 8 : codes Luka..... | 48 |
| Annexe 9 : codes Dorian..... | 62 |
| Annexe 10 : codes Lucas..... | 78 |
| Annexe 11 : kbhit.h..... | 87 |
| Annexe 12 : kbhit.cpp..... | 87 |
| Annexe 13 : config.ini..... | 88 |
| Annexe 14 : Configuration du client MQTT..... | 89 |
| Annexe 15 : Tests individuels Lucas Serrée-Delpeux..... | 90 |
| Annexe 16 : Test Unitaire Dorian..... | 93 |
| Annexe 17 : Test unitaire Mayonade Luka..... | 94 |
| Annexe 18 : Test Unitaire Serrée-Delpeux Lucas..... | 95 |

I-Introduction

Présentation générale du système supportant le projet :

Présentation générale du système supportant le projet : EM-SOLAR est une PME implantée à Ytrac (proche d'Aurillac) spécialisée dans la maintenance, le développement, la gestion et l'exploitation de tous types de production d'énergies renouvelables; le contrôle, le suivi, l'analyse de manière ponctuelle ou périodique, la réalisation d'audits et notamment la maîtrise de la consommation d'énergie de tous types de production d'énergies renouvelables. Une grande partie de son activité est orientée sur l'installation et la maintenance de petits parcs photovoltaïques destinés aux particuliers pour de l'autoconsommation. Le constat, fruit de l'expérience de l'entreprise, est que le pic de production correspond au moment où il y a le moins de demande au niveau des particuliers. L'excédent de production est donné au fournisseur d'énergie.





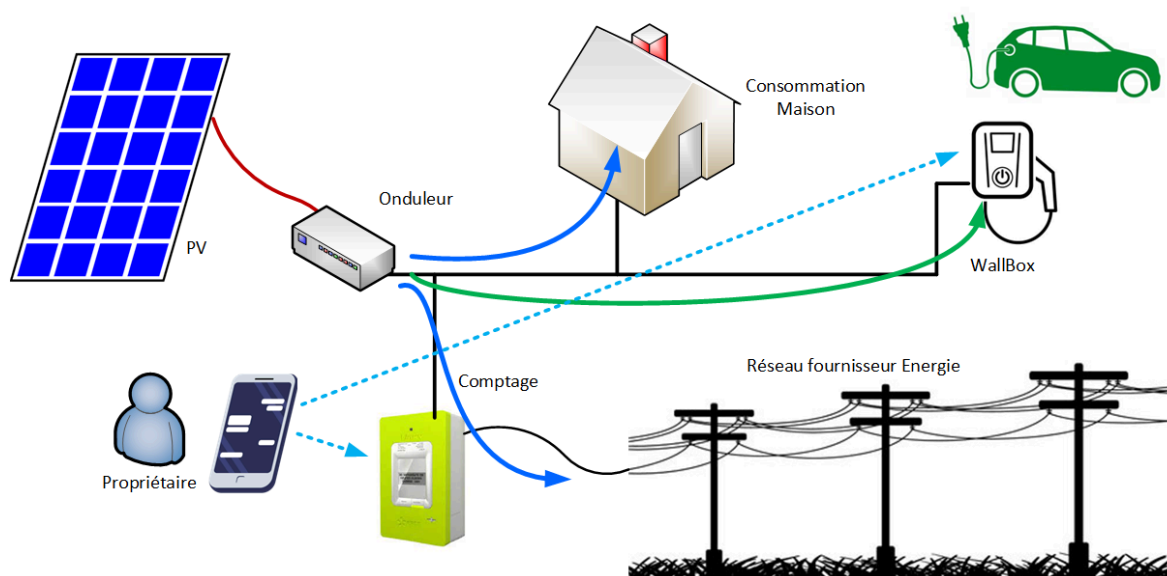
Comment optimiser ce surplus de production ?

- consommer pendant (pb de gestion)
- stocker sous forme de chaleur (chauffe eau ...)
- stocker dans batterie (coût élevé)

Projet :

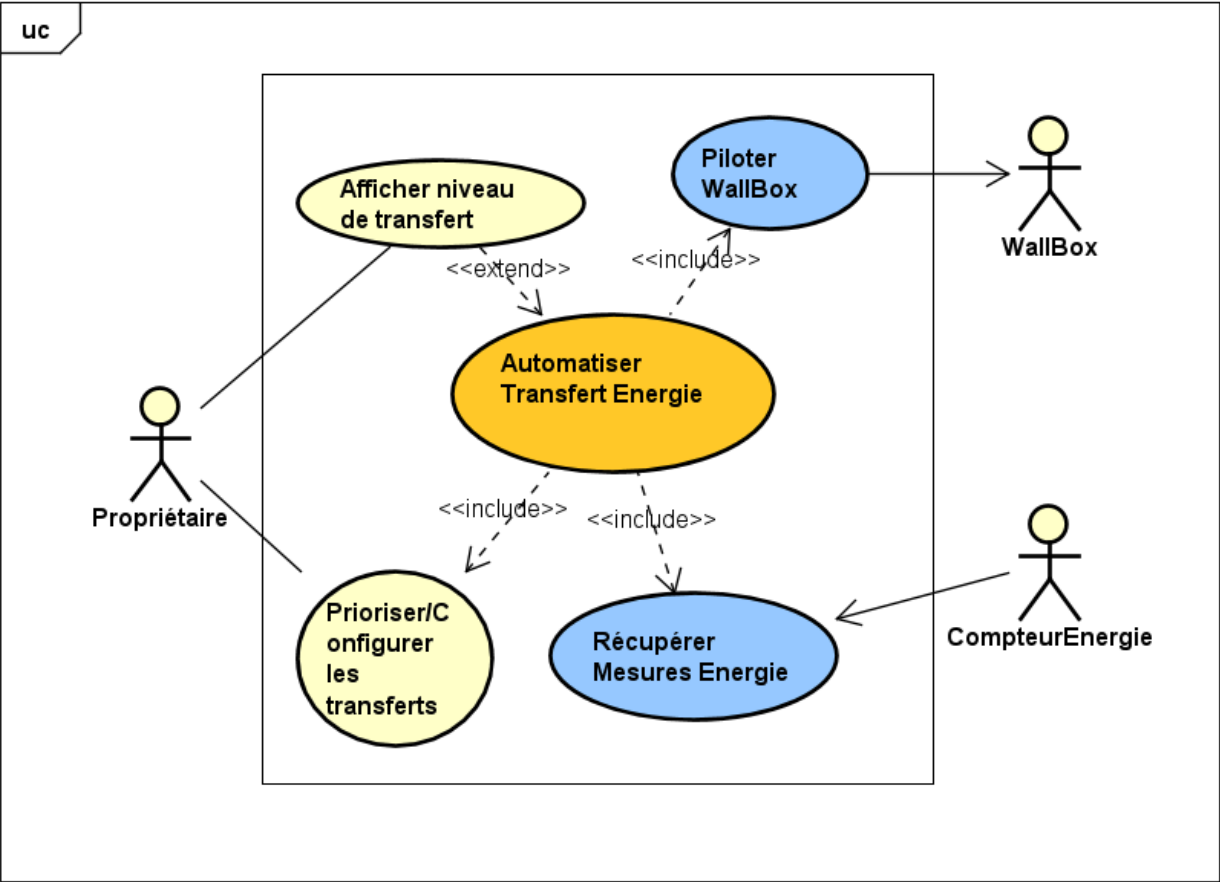
Une solution hybride peut être de stocker le surplus de production dans la batterie d'une voiture électrique de façon automatique.

Il nous a été demandé de proposer le prototype d'un système permettant de piloter automatiquement la wallbox d'un particulier pour que les surplus de production électrique issus d'une installation photovoltaïque permettent de recharger sa voiture électrique. L'utilisateur doit pouvoir visualiser les transferts d'énergie et les prioriser ...



Expression du besoin :

Diagramme des cas d'utilisation :



Description des acteurs/cas d'utilisation

| Acteur/use case | Rôle de cet acteur |
|-----------------|---|
| Propriétaire | Acteur qui doit configurer le système en fonction de ses besoins et de ses priorités. Il peut visualiser les niveaux de transfert d'énergie |
| WallBox | Acteur permettant de réguler la charge d'un véhicule électrique |
| CompteurEnergie | Acteur permettant une mesure le flux d'énergie et son sens |

Solutions envisagées pour répondre aux besoins :

La puissance instantanée envoyée ou reçue du réseau du fournisseur d'énergie est mesurée par un wattmètre, elle correspond à la différence entre la puissance absorbée par l'habitation et la puissance fournie par les panneaux photovoltaïques. Lorsqu'elle est négative c'est que les panneaux débitent sur le réseau. A ce moment là, il faut commander une wallbox pour recharger un véhicule électrique.

Différentes wallbox sont disponibles sur le marché avec des puissances de charge diverses. Nous devons en mettre une en œuvre dont la puissance de charge soit configurable à distance.

Ce prototype n'a pas vocation à être universel : ce sera un démonstrateur basé sur le matériel couramment installé par EM-SOLAR:

- Onduleurs HUAWEI (gamme SUN2000) et compteur d'énergie associé HUAWEI (gamme - Onduleurs HUAWEI (gamme SUN2000) et compteur d'énergie associé HUAWEI (gamme Smart Power Sensor), échange Modbus RTU sur RS485.

- WallBox GO-E (gamme Gemini Flex) : puissance de charge variable (courant pas de 1A), communication WIFI, Interfaces API publiques documentées : HTTP, MQTT, Modbus TCP.

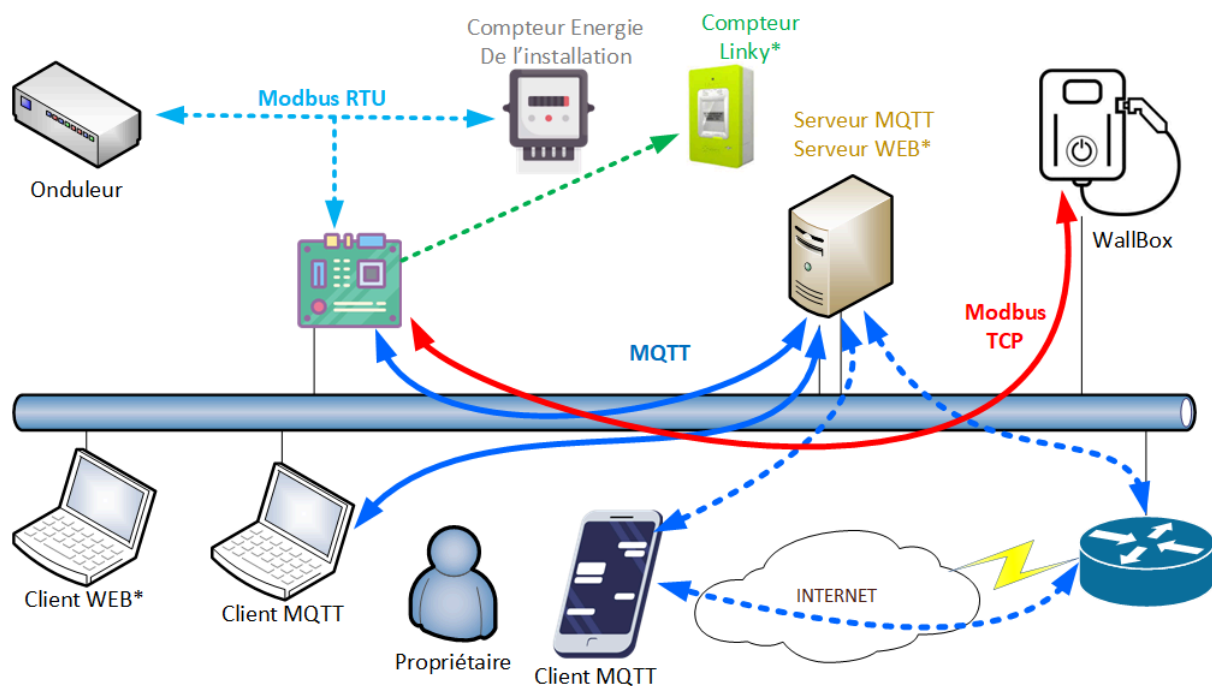
Il faut que notre système puisse observer les échanges entre l'onduleur et le compteur d'énergie pour y relever la valeur de la puissance envoyée sur le réseau électrique. Une option doit permettre de faire ce relevé à partir de compteur Linky.

Le propriétaire doit pouvoir avoir une vue d'ensemble des échanges, doit pouvoir prioriser ces échanges à partir d'interfaces clientes standards avec la possibilité de le faire à distance.




Les échanges d'informations doivent utiliser des protocoles standards :

Ce sont des données de petite taille qui sont échangées, le protocole MQTT sera privilégié. Une option doit fournir l'IHM du propriétaire sur une page WEB.

Synoptique correspondant :

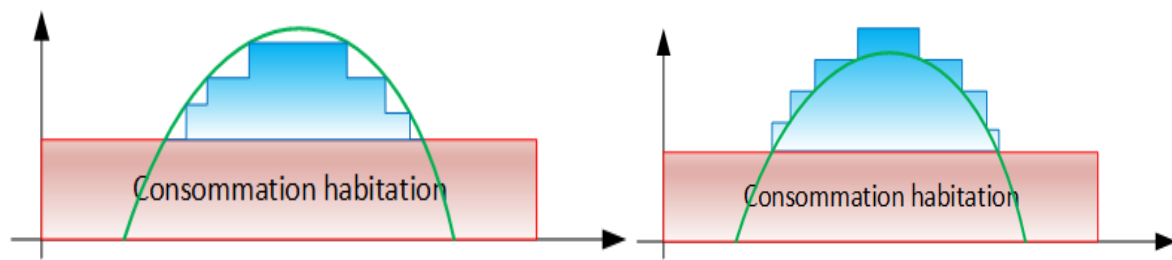


Contraintes technologiques :

- Système numérique d'acquisition et de traitement: micro-ordinateur de type Raspberry Pi 4 
- Application d'acquisition et de traitement à développer : QT 
- Broker MQTT/serveur WEB: Serveur local ou distant 
- Client MQTT graphique : application à installer
- les configurations dans des fichiers texte

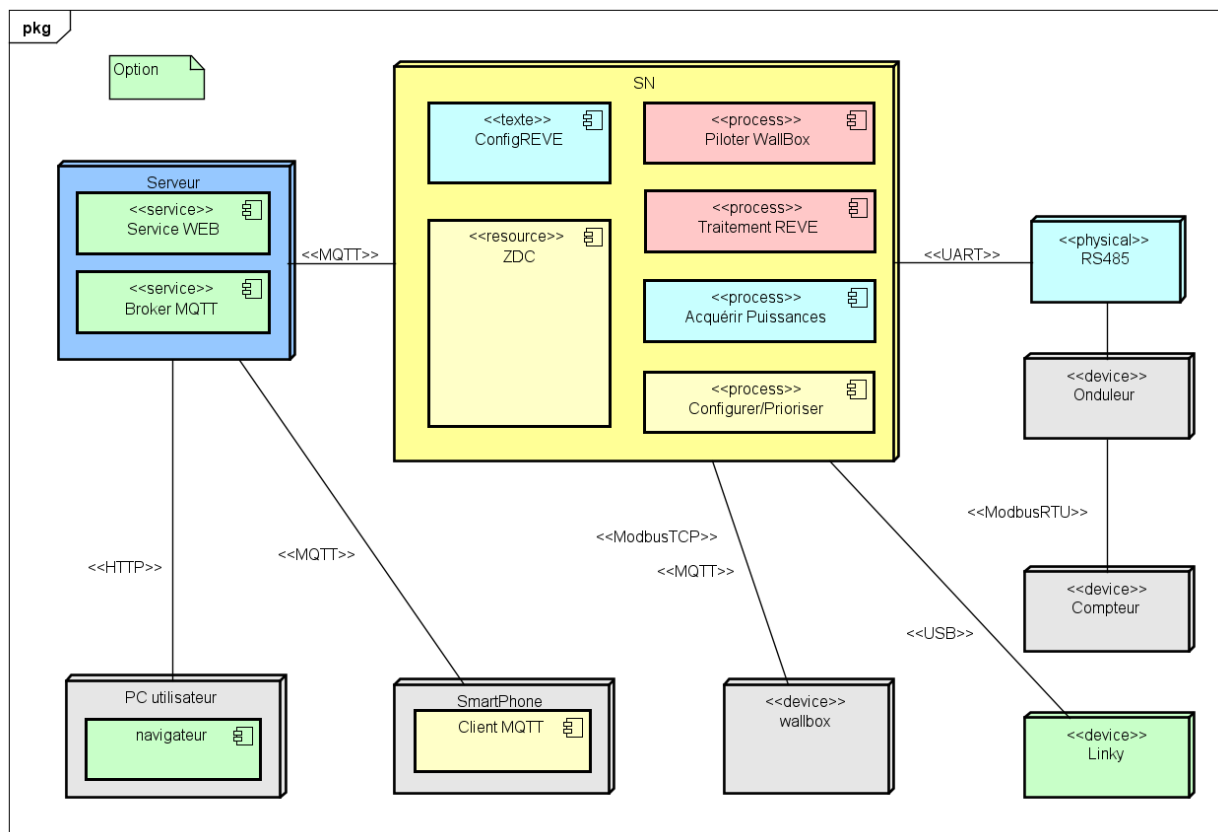
une priorité configurable par le propriétaire :

Un des choix de configuration du propriétaire est la possibilité de dépasser la production d'énergie pour recharger le véhicule ou non :



Architecture du système :

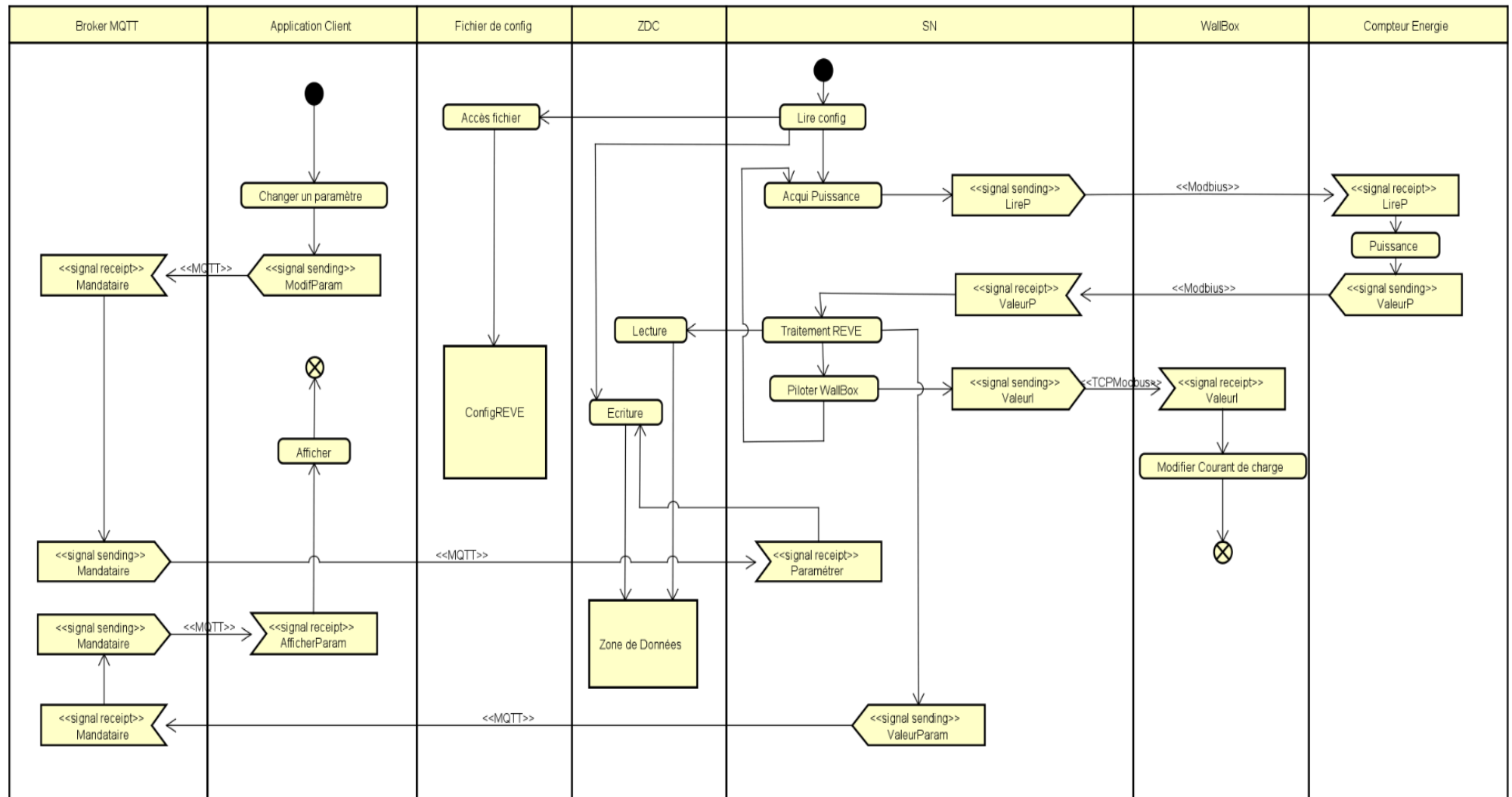
Diagramme de déploiement :



Description structurelle du système :

| Principaux constituants : | Caractéristiques techniques : |
|------------------------------|---|
| Client MQTT | Sur PC, tablette, smartphone du propriétaire pour configurer et prioriser les échanges au travers d'une application cliente MQTT standard |
| Navigateur | Sur PC, tablette, smartphone du propriétaire pour configurer et prioriser les échanges au travers d'une page WEB |
| Broker MQTT | Serveur MQTT public ou local |
| Service WEB | Serveur WEB public ou local |
| SN | Raspberry Pi V4 et ses processus et ses configurations |
| Process Traitement REVE | C++/QT, calcule la puissance à transférer au véhicule électrique en fonction de la production, de la consommation de la maison et de la configuration du propriétaire. |
| Process Piloter WallBox | C++/QT, configure la WallBox. |
| Process Acquérir Puissance | C++/QT, espionne la liaison ModbusRTU pour extraire les mesures de puissances échangées entre l'onduleur et le compteur d'énergie. Récupérer les mesures de puissances depuis Linky. |
| Process Configurer/Prioriser | C++/QT, remplit le ZDC en fonction des éléments de configuration et de priorisation fournis par le propriétaire en MQTT. Idem en HTTP. |
| ZDC | Zone de données commune à tous les processus : fichier, zone mémoire partagée ... |
| ConfigREVE | Fichier texte de configuration du système |

Diagramme d'activité :

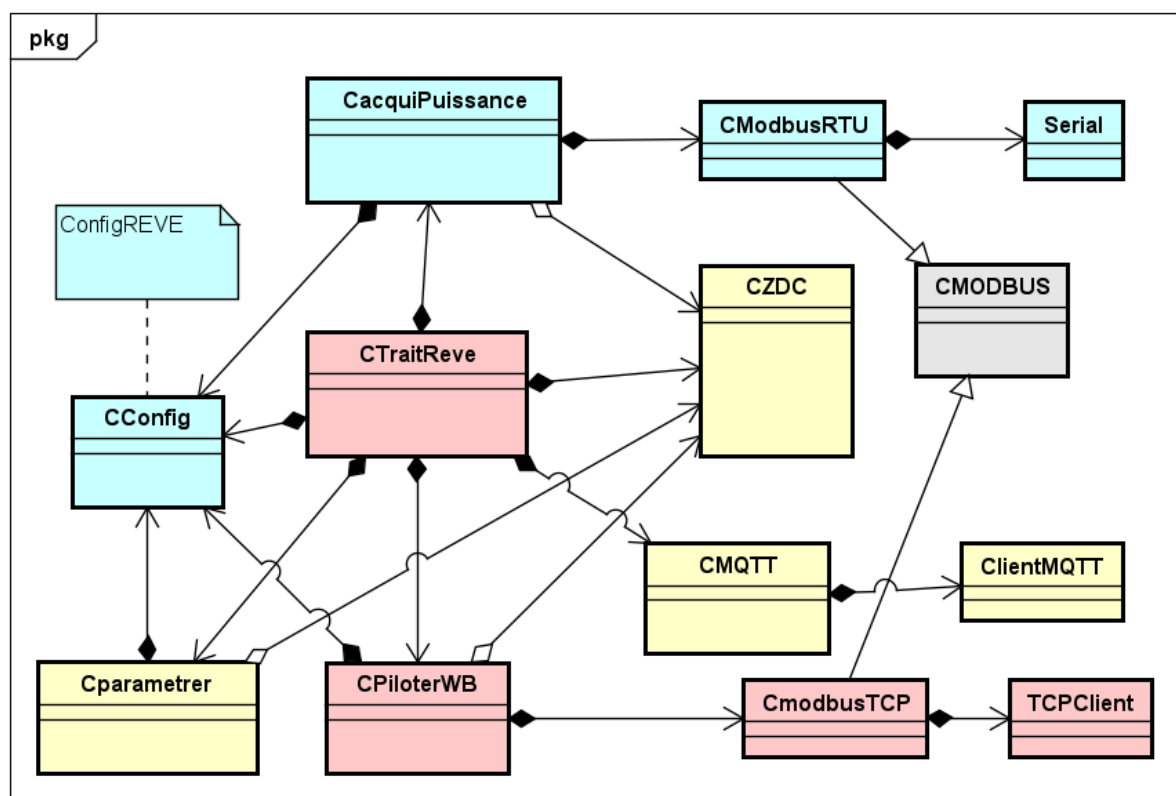


Après avoir rempli la zone de données commune par les éléments de configurations présents dans un fichier texte, le système numérique va acquérir la puissance consommée en espionnant les échanges modbus entre l'onduleur et le compteur d'énergie. Un traitement en sera fait en tenant compte des éléments de la zone de données communes pour piloter la wallbox.

Au travers du protocole MQTT, l'application client recevra des informations issues du traitement (niveau de charge, puissances, ...) et positionne des paramètres dans la zone de données commune comme l'autorisation de dépassement de production.

Architecture logicielle du système :

La structure logicielle du système peut être basée sur des processus (lourds ou légers) qui échangent au travers d'une zone de données commune ou être dans le même processus selon le diagramme suivant :

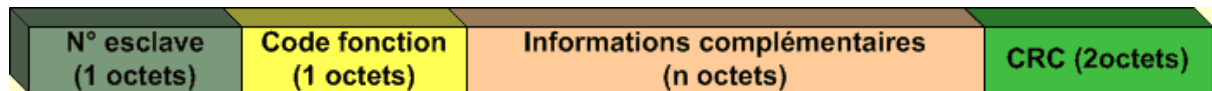


Répartition des tâches

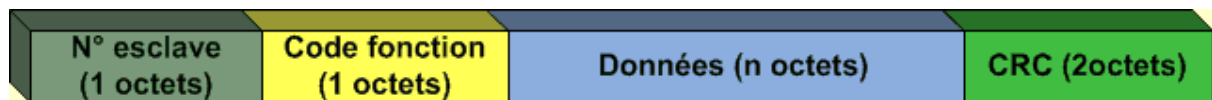
| | |
|---------------|---|
| Luka | <ul style="list-style-type: none">- Création du fichier CConfig- Création de classe CacquiPuissance |
| Dorian | <ul style="list-style-type: none">- Création de la Zone de Donnée Commune (CZDC)- Création de classe CParametrer- Installation et configuration du broker MQTT- Configuration du client MQTT |
| Lucas | <ul style="list-style-type: none">- Création du fichier CPiloterWB- Création du fichier CTraitReve |

Le format des requêtes et réponses des trames se présente comme ci-dessous.

Requête :



Réponse :



Voici le relevé de quelques échanges :

```

0B 03 20 06 00 02 2F 60
0B 03 04 3D 5E 69 3F 3F 3F
0B 03 20 06 00 02 2F 60
0B 03 04 3D 31 5B 57 76 3F
0B 03 20 06 00 02 2F 60
0B 03 04 3C 3F 3F 3F 3F 53
0B 03 20 00 00 22 3F 3F
0B 03 44 43 6E 3F 3F 3F 49 3F 5E 00 00 00 00 3D 76 3F 22 3D 76
3F 22 00 00 00 00 3F 36 7A 10 3F 36 7A 10 00 00 00 00 3E 40 3F
49 3E 40 3F 49 00 00 00 00 3F 3F 3F 0A 3F 3F 3F 0A 3F 3F 3F 0A
00 00 00 00 42 48 00 00 3F 3F
0B 03 20 06 00 02 2F 60
0B 03 04 3D 3F 3F 02 36 3F
0B 03 20 06 00 02 2F 60
0B 03 04 3D 63 53 3F 3F 3F
0B 03 20 06 00 02 2F 60
0B 03 04 3D 63 53 3F 3F 3F
0B 03 20 06 00 02 2F 60
0B 03 04 3D 5A 51 1A 3F 3F

```

En observant les échanges, on s'est rendu compte que c'est la fonction 3 qui est utilisée, voici comment nous la trouvons :

Request

| | | |
|-----------------------|---------|------------------|
| Function code | 1 Byte | 0x03 |
| Starting Address | 2 Bytes | 0x0000 to 0xFFFF |
| Quantity of Registers | 2 Bytes | 1 to 125 (0x7D) |

Response

| | | |
|----------------|--------------|--------|
| Function code | 1 Byte | 0x03 |
| Byte count | 1 Byte | 2 x N* |
| Register value | N* x 2 Bytes | |

*N = Quantity of Registers

Error

| | | |
|----------------|--------|----------------------|
| Error code | 1 Byte | 0x83 |
| Exception code | 1 Byte | 01 or 02 or 03 or 04 |

D'après la documentation ([Annexe 7](#)) du compteur d'énergie, son numéro d'esclave est le 11.

On peut ainsi décoder les éléments reçus, exemple des trames surlignées ci-dessus:

0B 03 20 06 00 02 2F 60 Requête:

| | | | | | |
|------------------|----------|----------|----------------|-----|----|
| 0B | 03 | 2006 | 0002 | 2F | 60 |
| Numéro d'esclave | Fonction | Registre | Nombre d'octet | CRC | |

0B 03 04 3D 5E 69 3F 3F 3F Réponse:

| | | | | | |
|------------------|----------|----------------|-----------------|-----|----|
| 0B | 03 | 04 | 3D5E693F | 3F | 3F |
| Numéro d'esclave | Fonction | Nombre d'octet | Valeur registre | CRC | |

valeur registre: \$3D5E693F => c'est un float sur 32 bits qui représente la puissance en KWatt soit : 1029597,503

0B 03 20 06 00 02 2F 60

0B 03 04 3D 31 5B 57 76 3F

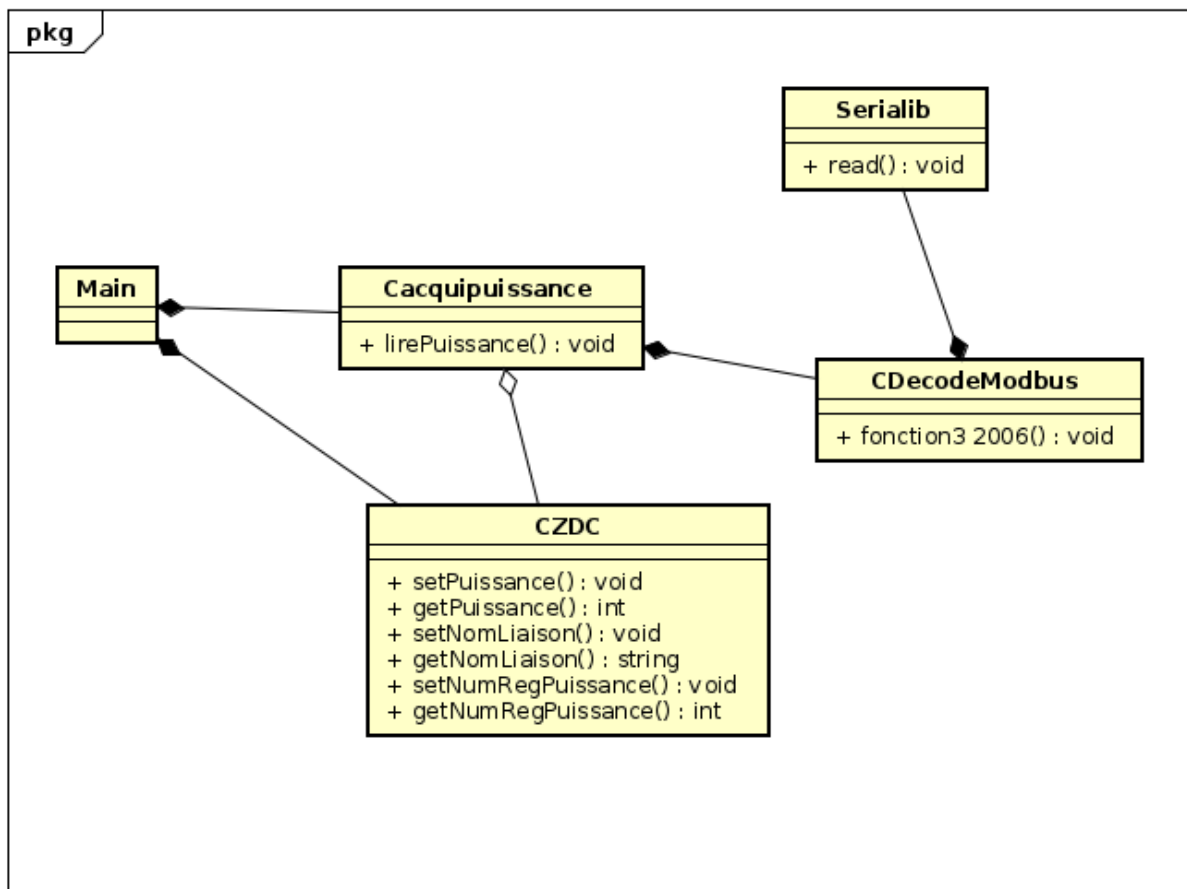
Après analyse c'est le registre numéro 2006 qui contient un float sur 32 bits image de la puissance.

0B 03 20 06 00 02 2F 60

Développement

Il faut lire en continu les trames reçues sur une liaison série, repérer dans cette trame la valeur du registre pour pouvoir renseigner la zone de données communes.

La méthode `cacquipuissance()` sera appelée cycliquement, un tableau d'octets reçus sur la liaison série sera traité par la méthode `cdecodemodbus()` de fonction3.



Voici l'algorithme de la partie lecture liaison série et remplissage du tableau d'octets :

repeter jusqu'à rep != 1

lire caractere octet

stocker resultat dans rep

si rep = 1

Ajouter octet au tableau trame à l'index 'i'

```
'i' = 'i'++
```

afficher nombre de caractère ('i')

Et, ci-dessous l'algorithme du décodage des trames :

```
i, fin = 0
```

repete **jusqu'à** i = trame ou fin = 1

Si sequence trame[i] == 0x0b && trame[i + 1] == 0x03 &&
trame[i + 2] == 0x04 est trouvée

```
fin = 1
```

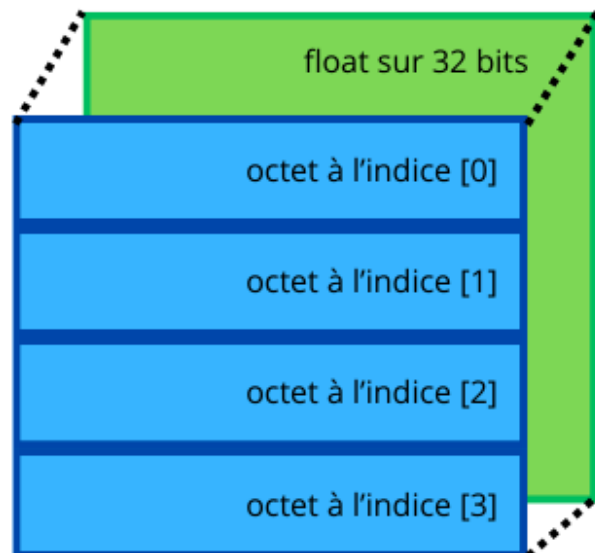
```
i = i++
```

afficher i

Principe mis en place pour lire un float à partir des 4 octets reçus:

L'union P permettra de
représenter un float en
parallèle du tableau de 4
octets.

```
union P
{
    unsigned char octet[4];
    float valeur;
};
```



Tests

Après avoir construit le code, nous avons pu le faire fonctionner.
Voici les traces reçues une fois lancé:

```
.pi@rasp45:~/Cible/REVE $ sudo ./recupTrame
PROGRAMME du 17 Mai 8:26 time out 500
nom liaison /dev/ttyUSB0
dans CDecodemodbus /dev/ttyUSB0
ouverture serie OK
Valeur obtenue: 3f b7 96 ad
Puissance en watt= 1434.29
Puissance dans ZDC = 59a

.Valeur obtenue: 3f b7 96 ad
Puissance en watt= 1434.29
Puissance dans ZDC = 59a

.Valeur obtenue: 3f b7 96 ad
Puissance en watt= 1434.29
Puissance dans ZDC = 59a

.Valeur obtenue: 3f b7 96 ad
Puissance en watt= 1434.29
Puissance dans ZDC = 59a
```

Nous pouvons voir les quatre première ligne qui sont informatives, où l'on retrouve si la liaison série est accessible, son nom et la date de la dernière modification.

```
PROGRAMME du 17 Mai 8:26 time out 500
nom liaison /dev/ttyUSB0
dans CDecodemodbus /dev/ttyUSB0
ouverture serie OK
```

Ensuite, nous retrouvons la valeur en hexadécimal des 4 octets reçus image de la puissance. On en extrait un réel en watt. Ici le résultat donne 1434.29 Watt. Nous avons également la puissance mise sous forme d'entier dans la zone de données commune en hexadécimal 59A.

```
.Valeur obtenue: 3f b7 96 ad
Puissance en watt= 1434.29
Puissance dans ZDC = 59a
```

(voir [Annexe 17](#) pour test unitaire)

Conclusion

Désormais, j'ai atteint la fonctionnalité désirée sur cette partie, j'ai réussi à récupérer des trames et à les analyser pour en extraire la puissance reçue.

J'ai aimé travailler sur cette partie avec l'espion ModBus car il y avait beaucoup de questionnements qui m'ont permis d'approfondir mes connaissances et de découvrir de nouveaux éléments. J'ai particulièrement aimé travailler sur la partie décodage de trame où il fallait chercher dans la trame ce qu'on voulait récupérer.

Partie étudiant Dorian

Introduction

Ma partie consiste à permettre, au travers du protocole MQTT, à l'application client de recevoir des informations issues du traitement (niveau de charge, puissances, ...) et à positionner des paramètres dans la zone de données commune comme l'autorisation de dépassement de production.

C'est moi qui me charge de créer cette zone de données communes.

Pour la mise en œuvre de ces éléments, je dois mettre en place les accès à la zone de données communes et un service MQTT (voir [Annexe 2](#)).

Accès à la ZDC

analyse:

Cette zone de données commune contiendra toutes les informations qui doivent être utilisées par les autres éléments du projet.

Tableau de ces informations :

| nom | type | remarque |
|-----------------|----------------------|--|
| puissance | entier | watt |
| numregPuissance | entier | |
| auDepassement | bool | Activer ou désactiver |
| idTrame | entier | |
| courant_WB | entier | |
| port_serie | chaîne de caractères | liaison série |
| ip_broker | chaîne de caractères | Adresse ip du broker MQTT |
| port_broker | entier | Port du broker MQTT |
| identifiant | chaîne de caractères | identifiant de la connection au broker |

| | | |
|-------------------|----------------------|--|
| motDePasse | chaîne de caractères | mot de passe du broker |
| clientid | chaîne de caractères | identifiant du client connecté au broker |
| ip_WallBox | chaîne de caractères | adresse ip de la WallBox |
| port_WallBox | entier | port de la WallBox |
| esclave_WallBox | entier | |
| topic_depassement | chaîne de caractères | topic pour le dépassement |
| topic_puisAqoise | chaîne de caractères | topic pour la puissance acquise |

Tous ces éléments seront des attributs privés d'une classe "CZDC". Il faudra lui rajouter des accesseurs publics pour accéder à ces attributs.

Des attributs de cette classe sont issus d'un fichier de configuration pour être visible par tous. Ce remplissage sera assuré par une classe "Cparametrer" qui doit connaître notre zone de données commune.

Développement :

Exemple de l'attribut "ip_broker" et de ses accesseurs :

czdc.h

```
#ifndef CZDC_H
#define CZDC_H

#include <string>
using namespace std;

class CZDC
{
private:
    string ip_broker;
...
public :
    void set_ipBroker(string ipBroker);
    string get_ipBroker();
...
}
```

```
};

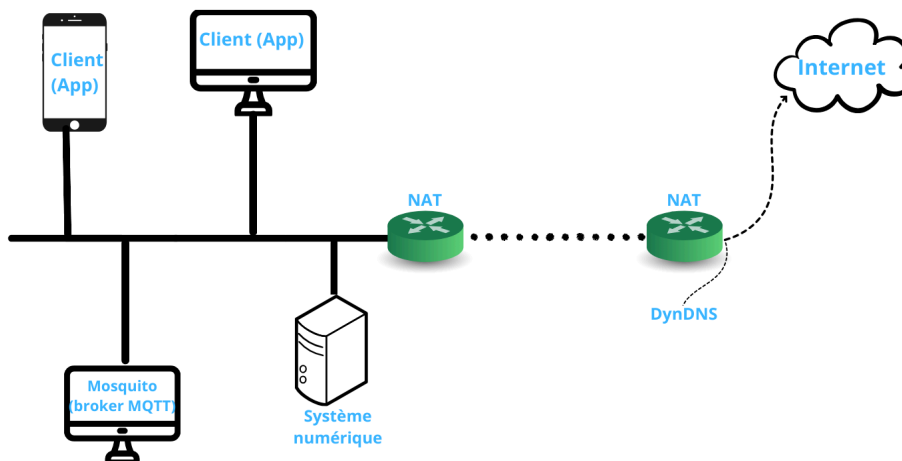
#endif // CZDC_H

czdc.cpp
#include "czdc.h"

// Broker MQTT
void CZDC::set_ipBroker(string ip_broker) // ip du broker
{
    this->ip_broker=ip_broker;
}
string CZDC::get_ipBroker()
{
    return ip_broker;
}
}
```

MQTT

Pour que le client ait accès aux différentes données de l'infrastructure tel que la puissance reçue par la WallBox par exemple, je dois configurer le service MQTT. Pour utiliser ce service, nous avons configuré le broker (serveur) MQTT Mosquito en local (voir installation [Annexe 3](#) pour Linux ou [Annexe 4](#) pour Windows), accessible à partir d'internet, à l'extérieur du réseau local. Il existe plusieurs broker, gratuit ou payant mais nous avons choisis Mosquito car il est gratuit et il fonctionne correctement, ce qui n'est pas le cas de tous.



Grâce au protocole MQTT, le client peut utiliser une application sur son téléphone afin de définir s'il veut dépasser la consommation électrique créée par le panneau solaire pour alimenter la WallBox ou non.

Au niveau de notre système numérique, on doit aussi être abonné et publier différents topics.

C'est grâce à la bibliothèque Paho que nous avons intégrée à notre projet C++ (voir [Annexe 1](#)) que nous pouvons faire ça.

Configuration d'un client MQTT

Liste de topics à utiliser:

| nom topic | message | Rôle | Publish/Sub scribe vis à vis du SN |
|---------------------|--|--|--|
| depassement | on / off | Autoriser le dépassement de la consommation électrique créée par le panneau solaire pour alimenter la WallBox ou non | Subscribe |
| puisAquis | chaîne de caractère image d'un entier | Puissance reçus en Watt | Publish |
| autTransfertWallbox | on / off | On transfère l'énergie vers la WallBox ou non | Subscribe |
| envoiWallbox | chaîne de caractère image d'un entier | Précise la puissance qui est envoyé à la WallBox en watt | Publish |
| charge | on / off | Etat de charge du véhicule | Publish |

Pour configurer un client MQTT sur une application client, vous pouvez utiliser l'application "IoT MQTT Panel" pour Android, ou "MQTT Explorer" pour PC. Dans chaque applications vous devrez ajouter une connection en paramétrant le nom que vous voulez donner à la connection, un client ID qui peut être unique, l'adresse du broker MQTT, le port de connection, et pour finir le protocole utiliser pour la connection (TCP ou Websocket). Avec l'application sur Android, vous pouvez créer un Dashboard afin d'y faire apparaître des boutons ou des boîtes de dialogues ce qui peut servir pour afficher des informations tels qu'une température par exemple ou pour d'autoriser quelque chose, dans notre cas un bouton permet d'autoriser le dépassement de la consommation électrique créée par le panneau solaire pour alimenter

la WallBox ou non. Cependant, avec MQTTExplorer, on ne peut pas créer de Dashboard, on doit publier directement en précisant le topic ainsi que le payload (message).

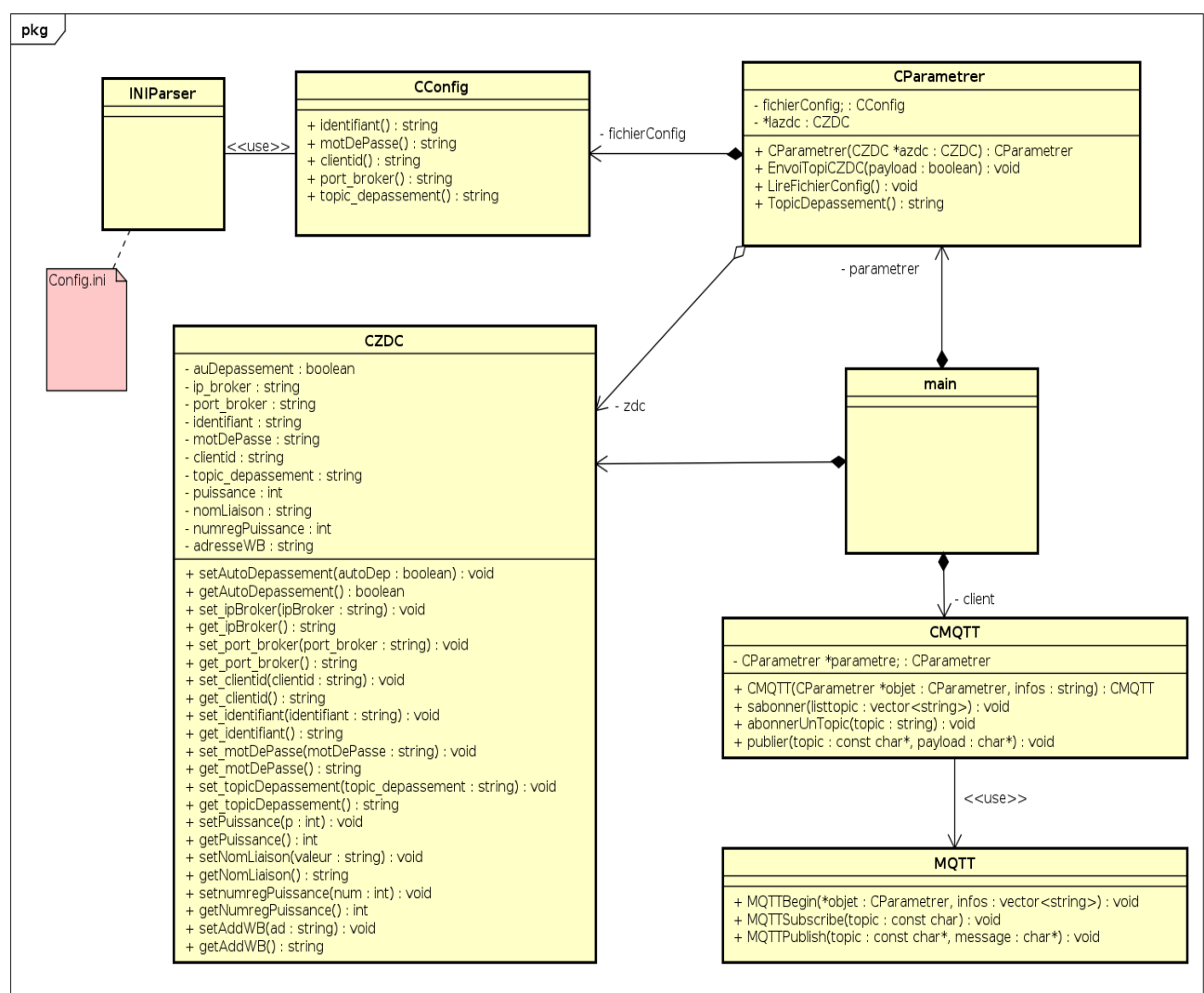
Configuration des clients MQTT pour le projet voir [Annexe 14](#).

Développement:

Au travers d'une classe "Cparametrer" les informations du fichier de configuration sont transférées dans la zone de données communes.

Cette classe sera aussi utilisée au moment à la réception du topic "dépassement" pour mettre l'état du topic dans la zone de données communes afin que d'autres classes puissent l'utiliser.

Diagramme de classes :



Paho nous fournit des fonctions de base pour les accès MQTT (fichier mqtt.h et .cpp):

```
void MQTTBegin(CParametrer *objet,vector<string> infos);
void MQTTSubscribe(const char* topic);
void MQTTPublish(const char* topic, char* message);
int  msgarrvd(void *context,  char *topicName,  int  topicLen,
MQTTClient_message *message)
```

Si le rôle des fonctions MQTTBegin (initialisation), MQTTSubscribe (Abonner), MQTTPublish (publier) est assez clair, il faut se pencher sur la fonction msgarrvd qui est déclencher à la réception d'un topic auquel on est abonné.

Pour que cette fonction appelle la méthode EnvoiTopicDepassement_CZDC() de Cparametrer, il faut faire passer l'adresse de l'objet issu de Cparametrer à la fonction MQTTBegin.

Nous allons encapsuler ces fonctions dans une classe CMQTT avec les méthodes suivantes:

```
CMQTT(CParametrer *objet, vector<string> infos);
~CMQTT();
void sabonner(vector<string> listtopic);
void abonnerUnTopic(string topic);
void publier(const char* topic, char* payload);
```

Le **constructeur** reçoit toutes les informations de connexions au broker dans un vecteur de chaîne de caractère nommée "infos" qui sont utilisées pour initialiser la connexion au broker en appelant à la fin la fonction " MQTTBegin(objet,infos);". On en profite pour faire passer l'adresse d'un objet de type Cparametrer pour pouvoir sa méthode EnvoiTopicDepassement_CZDC().

La méthode "**sabonner()**" permet de s'abonner à une liste de topic. Par exemple, cela nous sert pour nous abonner au topic "depassement" et en même temps au topic "puisAquis".

Tandis que la méthode "**abonnerUnTopic()**" permet seulement de s'abonner à un topic ce qui peut nous servir pour tester le fonctionnement d'un topic.

La méthode "**publier()**" permet de publier un message sous forme d'une chaîne de caractère (payload) en fonction d'un topic.

Exemple de la publication de la puissance:

Lorsque le programme principal est lancé, on rentre dans une boucle qui lit la puissance acquise et on l'enregistre dans la zone de données communes. Si elle a changé alors on la publie sur le topic "puisAquis".

Principe utilisé pour cela:

On enregistre la valeur actuelle de la puissance mesurée et on la compare à la nouvelle:

(extrait du programme principal)

```
...
do{
    ...
    int old_puissance=zdc.getPuissance();

    // acquisition de la puissance => remplissage ZDC
    acquiP.LirePuissance(zdc.getNumregPuissance());

    int new_puissance=zdc.getPuissance();
    cout << "Puissance lue dans ZDC : " << new_puissance << endl;

    // publier des éléments (qui sont dans la ZDC) que si il y a changement
    ...
    if (old_puissance != new_puissance)
    {
        sprintf(puissance_char, "%d", new_puissance); //convertit l'entier
        puissance de la zdc en char*
        client.publier(zdc.get_topic_puiAquis().c_str(), puissance_char);
    }
    ...
} while(fin==0);
...
```

Exemple de la réception du topic depassement:

A la réception d'un topic auquel on est abonné, la fonction msgarrvd() est appelée. On positionne l'autorisation de dépassement dans la ZDC au travers de la méthode EnvoiTopicDepassement_CZDC() de Cparametrer uniquement pour le topic depassement:

(extrait de la fonction msgarrvd())

```
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message
*message)
{
    bool etat = false;
    char* payload;
    string messageArvd;
    string topic;

    printf("\n\n-----Message arrived-----\n");
```

```

printf("Topic : %s\n", topicName);

payload = static_cast<char*>(message->payload);
messageArvd=payload;

topic = topicName;

cout<<"Payload : "<<payload<<endl;

cout<<"Vrai trame a traiter "<<payload<<" , du topic "<<topicName<<endl;
cout<<"-----"<<endl;

// pour le topic depassement : on rempli la zdc
if (topic==parametre->TopicDepassement())
{
    if(messageArvd=="on")etat=1; //si message reçu on alors etat=1 (allume)
    else etat=0; //si message reçu off alors etat=0 (eteind)
    parametre->EnvoiTopicDepassement_CZDC(etat);
}

...
return

```

Tests

Après avoir créé toutes mes classes, J'ai réalisé des tests afin d'évaluer leur fonctionnement. Comme nous pouvons le voir sur la capture d'écran ci-dessous, on reçoit bien l'information que l'état du bouton de dépassement de la consommation associé au topic "dépassement" à changer et ce changement est bien pris en compte dans la ZDC. Nous pouvons voir que la connexion entre le client et le service MQTT fonctionne.

```
----- Infos connection au broker MQTT -----
# Identifiant => client
# IP du broker MQTT => 172.16.10.1
# Mot de passe => client
# Topic actuellement abonné => depassement
# Client id => client1
après lecture vector ...

Abonnement au topic : 'depassement' pour le client 'client1' utilisant QoS1
Subscribing to topic depassement for client 1 using QoS1634755940
Subscribing to topic puisAquis for client 1 using QoS1936291184

----- DEBUT -----
...

Etat du topic 'depassement' dans la ZDC : 0

...
Réception du topic dépassement
-----Message arrived-----
Topic : depassement
Payload : on
Vrai trame a traiter on , du topic depassement
...
Visualisation de l'état du topic "depassement" dans la ZDC
Etat du topic 'depassement' dans la ZDC : 1
...
Réception du topic dépassement
-----Message arrived-----
Topic : depassement
Payload : off00
Vrai trame a traiter off00 , du topic depassement
...
Visualisation de l'état du topic "depassement" dans la ZDC
Etat du topic 'depassement' dans la ZDC : 0

....
Fin du programme
```

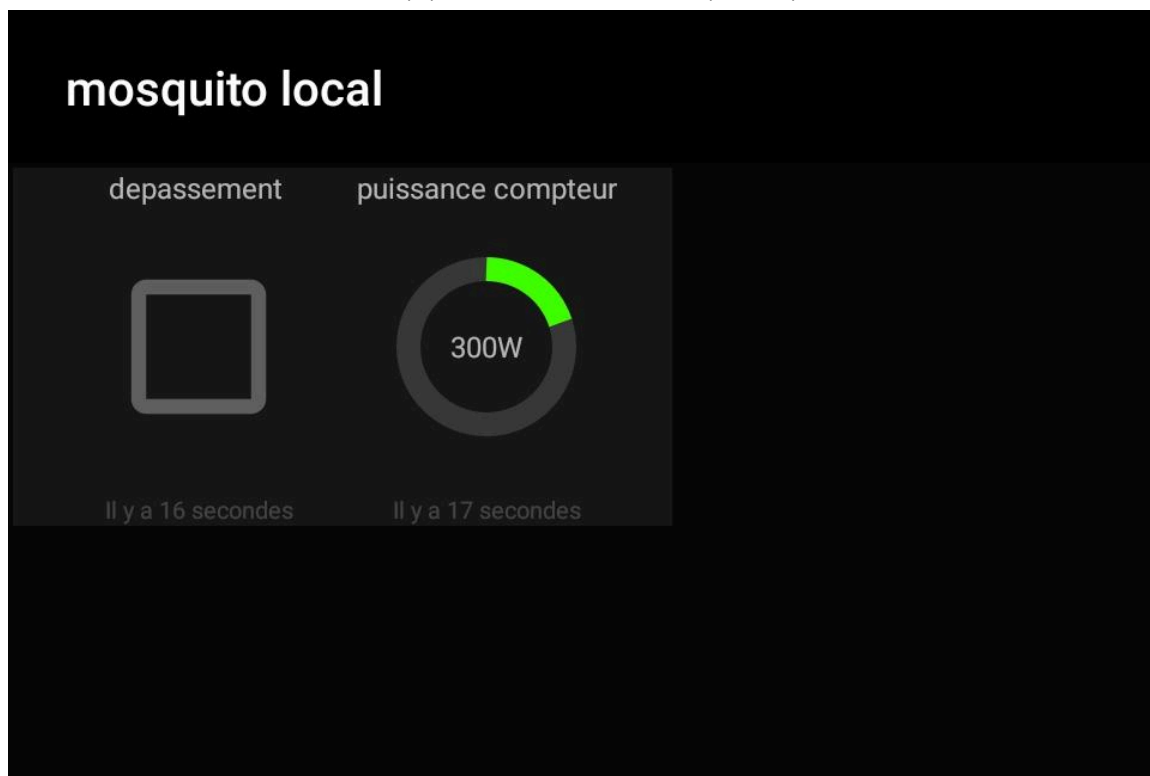
Ensuite, le fonctionnement de l'affichage de la puissance du compteur est validé, puisque sur la capture d'écran ci-dessous nous voyons bien que lorsque le topic "puisAquis" est modifié, nous recevons le payload et ça modifi la jauge sur l'application client :

```
----- Infos connection au broker MQTT -----  
# Identifiant => client  
# IP du broker MQTT => 172.16.10.1  
# Mot de passe => client  
# Topic actuellement abonné => puisAquis  
# Client id => client1  
après lecture vector ...  
  
Abonnement au topic : 'puisAquis' pour le client 'client1' utilisant QoS1  
Subscribing to topic puisAquis for client 1 using QoS1936291184  
  
----- DEBUT -----  
.....published to puisAquis  
payload : 300  
.Message with token value 3 delivery confirmed
```

Topic auquel on est abonné

Publication du payload "300" sur le topic "puisAquis"

Vue de la tablette sur l'application client après publication :



Voir test unitaire [Annexe : 16](#)

Difficultés rencontrées

Durant le développement de ma partie, j'ai eu des difficultés pour l'installation de la bibliothèque Paho ainsi que pour l'inclure dans le projet. Sur le github de la bibliothèque, les instructions données pour l'installation ne marchent pas complètement et j'ai dû trouver d'autres instructions pour l'installation et l'inclusion au projet (voir [Annexe 1](#)).

Conclusion

Pour ma part, mes classes fonctionnent comme on peut le voir à travers les tests et elles remplissent leurs fonctions principales pour l'utilisateur tout en respectant le cahier des charges.

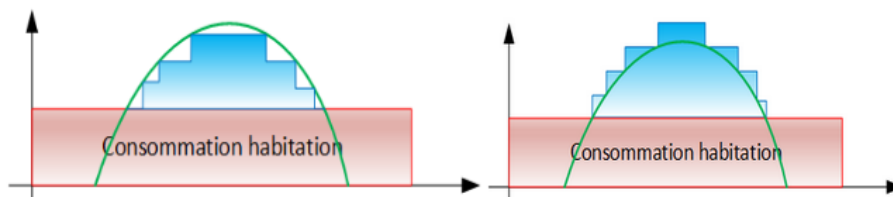
Partie étudiant Lucas

Objectif

Quand le système observe une surproduction d'énergie dans le foyer, le programme doit être capable de communiquer avec la WallBox pour transférer le surplus dans la batterie du véhicule électrique.

La transmission d'énergie doit être contrôlée en fonction de la valeur du surplus d'énergie, en d'autres termes il ne suffit pas de simplement allumer ou éteindre la WallBox.

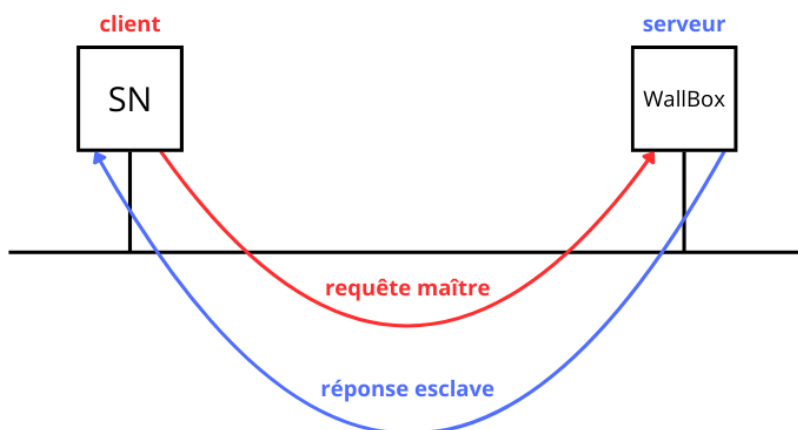
On doit également pouvoir autoriser ou non le dépassement d'énergie :



Analyse de la partie

Communication WallBox : ([Annexe 6](#))

C'est au travers du protocole TCP ModBus qu'on peut commander la WallBox accessible à son adresse IP et au port 502. TCP ModBus fonctionne sur le principe du maître-esclave, où notre programme sera un client de la WallBox (Serveur).

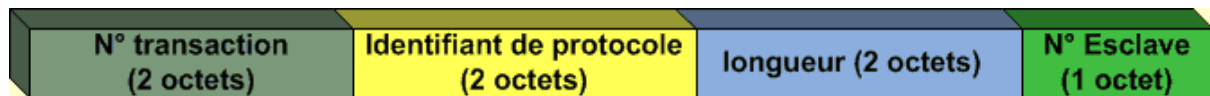


La valeur d'intensité de charge de la WallBox est configurable au travers du registre 299, c'est une valeur comprise entre 6 et 32 Ampères.

On doit également pouvoir autoriser ou non la charge en manipulant le paramètre correspondant au registre 200.

Pour écrire dans ces registres, on utilisera la fonction 6 du protocole TCP ModBus. (This function code is used to write a single holding register in a remote device.)

La première partie de l'entête TCP/ModBus se divise en 4 champs :



| Fields | Length | Description - | Client | Server |
|------------------------|---------|--|--------------------------------------|--|
| Transaction Identifier | 2 Bytes | Identification of a MODBUS Request / Response transaction. | Initialized by the client | Recopied by the server from the received request |
| Protocol Identifier | 2 Bytes | 0 = MODBUS protocol | Initialized by the client | Recopied by the server from the received request |
| Length | 2 Bytes | Number of following bytes | Initialized by the client (request) | Initialized by the server (Response) |
| Unit Identifier | 1 Byte | Identification of a remote slave connected on a serial line or on other buses. | Initialized by the client | Recopied by the server from the received request |

- N° transaction : nombre sur 2 octets à incrémenter à chaque envoi
- Identifiant de protocole : nombre sur deux octets (Ici, zero)
- Longueur : nombre d'octets qui suivent ce champ.
- d'un Numéro d'esclave (On possède un seul esclave, on choisira simplement 0

La partie suivante est :

| Code fonction (1 octets) | Informations complémentaires (n octets) |
|-----------------------------|--|
|-----------------------------|--|

où :

| | | |
|------------------|---------|------------------|
| Function code | 1 Byte | 0x06 |
| Register Address | 2 Bytes | 0x0000 to 0xFFFF |
| Register Value | 2 Bytes | 0x0000 to 0xFFFF |

Les champs contiennent :

- Un code de fonction, ici la valeur 6sdgdgsdsgsdg
- l'adresse du registre sur 2 octets
- la valeur du registre sur 2 octets

Exemples de trame à envoyer:

- pour autoriser la charge :

(Valeurs en Hexadécimal)

| N° transaction | ID Protocole | Longueur | N° Esclave | Code fonction | Adresse registre | Valeur registre |
|---------------------------------------|-------------------------------|--------------------------------------|------------|---------------------------|------------------------------|--------------------------------|
| 0021 | 0000 | 0006 | 01 | 06 | 00C8 | 0000 |
| nombre à incrémenter à chaque requête | Protocole utilisé 0=ModBus | Nombre d'octets qui suivent ce champ | numéro 1 | écriture dans un registre | Adresse du registre ALLOW | 0 = charge 1 = arrêt charge |

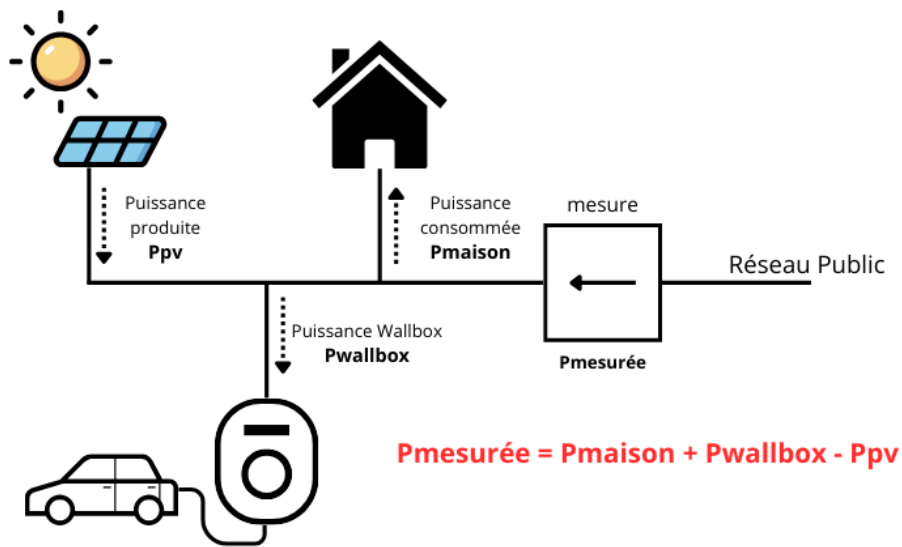
- pour changer le courant de charge :

(Valeurs en Hexadécimal)

| N° transaction | ID Protocole | Longueur | N° Esclave | Code fonction | Adresse registre | Valeur registre |
|---------------------------------------|-------------------------------|--------------------------------------|------------|---------------------------|--|----------------------|
| 0022 | 0000 | 0006 | 01 | 06 | 012B | 0006 |
| nombre à incrémenter à chaque requête | Protocole utilisé 0=ModBus | Nombre d'octets qui suivent ce champ | numéro 1 | écriture dans un registre | Adresse du registre AMPERE_VOLATILE | De 6 à 32 Ampères |

Traitement

Le bilan des puissances mises en oeuvre est le suivant:



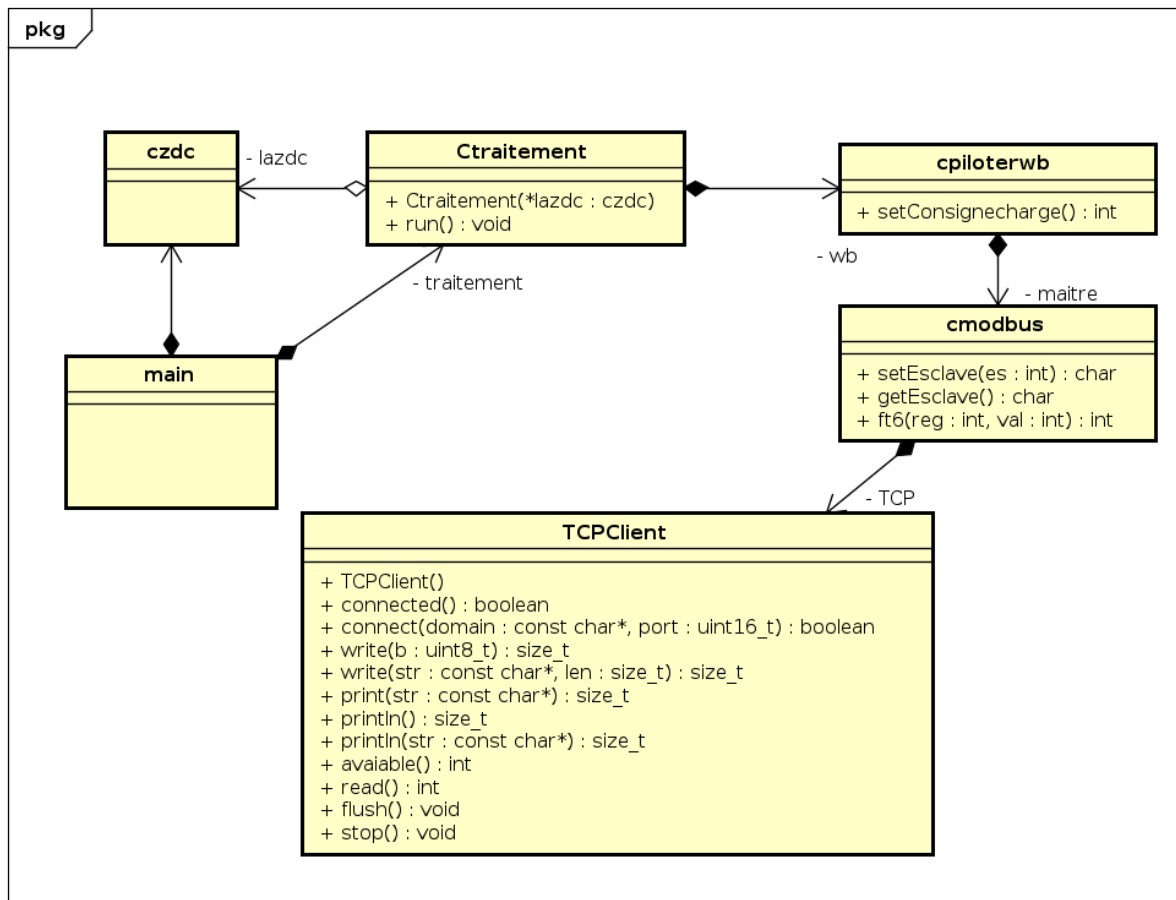
Quand l'habitation produit plus qu'elle ne consomme, la puissance mesurée est négative:

L'yc courant de charge minimum de la WallBox étant de 6 Ampères, pour enclencher la charge, il faut avoir une puissance réinjectée dans le réseau qui soit inférieure à -6×230

Puis il faut tenir compte de la puissance de charge selon de diagramme suivant:

Développement

Diagramme de classe:

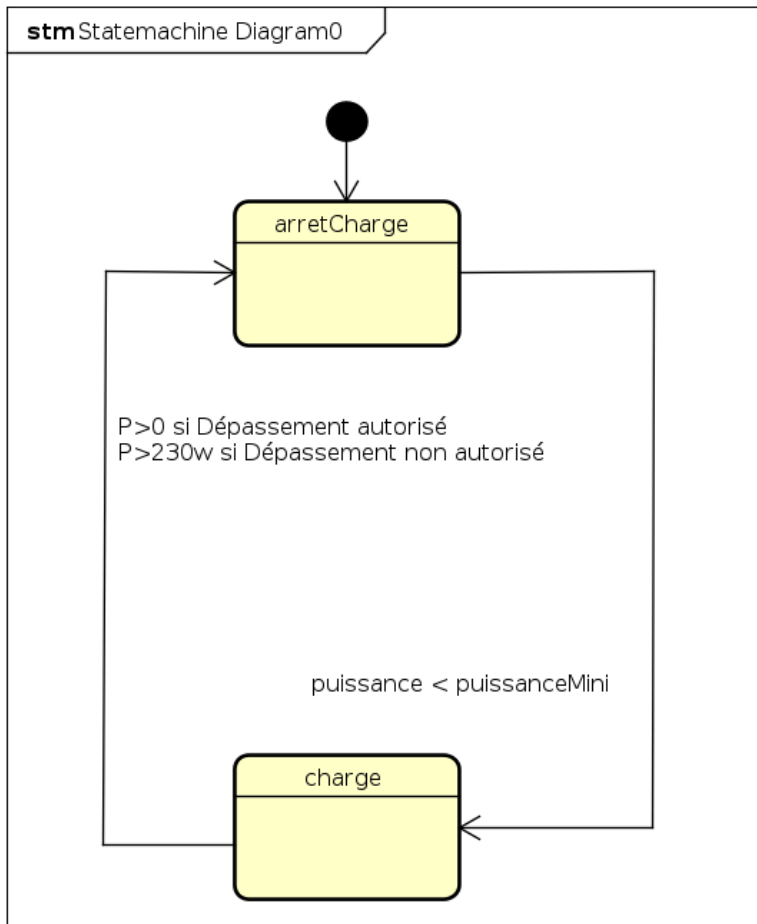


la méthode run de traitement doit être appelée cycliquement après l'acquisition de la puissance:

Elle répondra au diagramme Etat-Transition suivant pour lequel dans l'état:

arretCharge: la charge est arrêtée sur la wallbox (protocole TCPModbus), la zone de données commune est remplie en conséquence.

charge: la charge est configurée sur la wallbox en fonction de la puissance disponible, la zone de données commune est remplie en conséquence.



extrait du code pour cette méthode run():

```

enum etat{charge, arretCharge};
...
    int puissance = lazdc->getPuissance();
this->depassement=lazdc.getdepassement();
...

switch(etatCharge)    // faire évoluer machine à état
{
case arretcharge: if(puissance < -PCMini) etatCharge=charge;break;
case  charge:    if(puissance  >  (Tension*(int)this->depassement))
etatCharge=arretcharge;break;    // en fonction du dépassement
}

switch(etatCharge)    // que faire dans chaque état
{
case arretcharge:
this->PuissanceCharge=0;
this->courantCharge=0;
if(!this->arretenvoye){

```

```

        wb.setConsignecharge(this->courantCharge);        // arreter WB et
mettre puissance charge à 0
        this->arretenvoye=true;
    }

    // voir pour envoyer qu'une seule fois ...
    break;
    case charge:
    // calcul puissance de charge et commander WB
        nbpasPuissance=-puissance/(Tension*1);

        // en fonction du dépassement
        if(this->depassement==true)
        {
            nbpasPuissance=nbpasPuissance+1;
        }
        courantCharge=courantCharge+nbpasPuissance;
        this->PuissanceCharge=(courantCharge)*Tension;
        cout << "courantCharge : " << courantCharge << endl;
        // si puissance charge a changée alors piloter WB
        if(nbpasPuissance!=0)=
        {
            wb.setConsignecharge(courantCharge);
            this->arretenvoye=false;
        }
    break;
}

cout << "Nbpas puissance = " << nbpasPuissance << endl;
cout << "PuissanceCharge = " << this->PuissanceCharge << endl;
switch(etatCharge){
case 0: cout << "EtatCharge = 0 (Allumé)" << endl << endl;
        break;
case 1: cout << "EtatCharge = 1 (Eteint)" << endl << endl;
        break;
}
}

```

Tests individuels

Les observations sont vérifiables [Annexe 15](#).
Voir [Annexe 18](#) pour le test Unitaire.

Test d'intégration

Toujours en cours...

Conclusion

D'un point de vue avancement du projet, ma partie personnelle est fonctionnelle.

J'ai personnellement apprécié travailler sur cette partie du projet, celle-ci m'a posée problème à plusieurs reprises, ces problèmes ont été réglés après de lentes mais intéressantes recherches, ce a grandement enrichi mes connaissances.

Conclusion générale

En conclusion, nous apprécions le projet qui nous a été attribué. Ce dernier n'est pas totalement fini, mais étudier le protocole MQTT ainsi que le protocole ModBus, a pu développer nos connaissances. Cela nous a permis d'apprendre plus sur les bibliothèques et le monde de la programmation. La phase d'intégration est un travail complexe et coopératif qui nous a appris à travailler en équipe, à gérer notre temps et de mettre en œuvre nos compétences acquises durant notre formation.

Annexes

Annexe 1 : Installation bibliothèque MQTT avec Paho

Liens documentation github :

<https://github.com/eclipse/paho.mqtt.cpp>

Commande d'installation sous Linux :

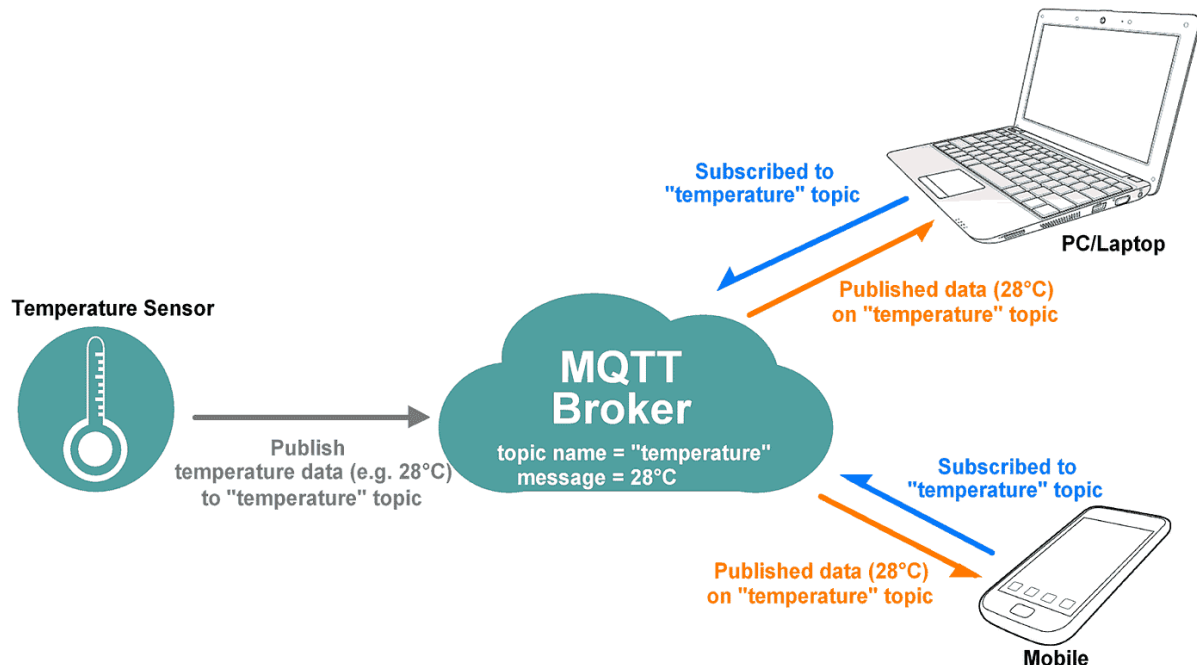
1. `sudo apt-get install build-essential git gcc make cmake cmake-gui cmake-curses-gui libssl-dev doxygen graphviz`
2. `git clone https://github.com/eclipse/paho.mqtt.cpp`
3. `cd paho.mqtt.cpp`
4. `sudo ./install_paho_mqtt_c.sh`
5. `sudo cmake -DCMAKE_INSTALL_PREFIX=install -DPAHO_ENABLE_TESTING=OFF`
6. `sudo make install`

On doit inclure le dossier `"/paho.mqtt.cpp/paho.mqtt.c/src/"` au `.pro` du projet:

`INCLUDEPATH += /home/<l'utilisateur>/paho.mqtt.cpp/paho.mqtt.c/src/`

Annexe 2 : Documentation MQTT

Le protocole MQTT est devenu une norme pour la transmission de données IoT car il est facile à implémenter et peut communiquer efficacement les données IoT.



Topic MQTT

Un topic est une chaîne codée en UTF-8 qui sert de base à l'acheminement des messages dans le protocole MQTT. Un sujet est généralement divisé en niveaux et séparé par une barre oblique '/' entre les niveaux. Ceci est similaire aux chemins d'URL, par exemple :

myhome/groundfloor/livingroom/temperature

Ce thème représente la température dans le salon d'une maison située au rez-de-chaussée.

MQTT Wildcards

Les wildcards MQTT sont un type spécial de topic qui ne peut être utilisé que pour l'abonnement et non pour la publication. Les clients peuvent s'abonner à un topic wildcard pour recevoir des messages de plusieurs topics correspondants, ce qui évite de devoir s'abonner à chaque topic individuellement. MQTT prend en charge deux types de caractères génériques : + (à un niveau) et # (à plusieurs niveaux).

Abonnement (subscribe)

Un broker MQTT est une entité intermédiaire qui permet aux clients MQTT de communiquer. Plus précisément, un broker MQTT reçoit les messages publiés par les clients, filtre les messages par sujet et les distribue aux abonnés.

Publication (publish)

Dans MQTT, un client peut publier des messages immédiatement lorsqu'il se connecte à un broker. Les messages sont filtrés en fonction de sujets et chaque message doit contenir un sujet que le broker peut utiliser pour transmettre le message aux clients intéressés. La charge utile de chaque message comprend les données à transmettre au format octet, et le client expéditeur peut choisir d'envoyer tout type de données, y compris du texte, des chiffres, des images, des données binaires et même du XML ou JSON à part entière.

Annexe 3 : Installation sur Linux du broker MQTT Mosquitto

Étape 1 : Mise à jour des dépôts

Avant d'installer Mosquitto, assurez-vous que votre liste de paquets est à jour. Exécutez la commande suivante :

```
sudo apt update
```

Étape 2 : Installation de Mosquitto

Installez Mosquitto et les utilitaires clients :

```
sudo apt install mosquitto mosquitto-clients
```

Étape 3 : Démarrage et activation de Mosquitto

Pour démarrer Mosquitto et l'activer au démarrage du système, utilisez les commandes suivantes :

```
sudo systemctl start mosquitto
```

```
sudo systemctl enable mosquitto
```

Étape 4 : Configuration de Mosquitto

Le fichier de configuration de Mosquitto se trouve dans `/etc/mosquitto/mosquitto.conf`. Pour une configuration de base, éditez ce fichier avec votre éditeur de texte préféré :

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Ajoutez ou modifiez les lignes suivantes pour configurer Mosquitto :

```
# Autoriser les connexions anonymes (non sécurisé)  
allow_anonymous true
```

```
# Définir le port d'écoute  
listener 1883
```

Après avoir modifié le fichier de configuration, redémarrez Mosquitto :

```
sudo systemctl restart mosquitto
```

Annexe 4 : Installation sur Windows du broker MQTT Mosquitto

Étape 1 : Téléchargement de Mosquitto

Accédez à la page de téléchargement de Mosquitto et téléchargez l'installateur pour Windows depuis [ce lien](#) et exécutez l'installateur téléchargé.

Étape 2 : Configuration de Mosquitto

Le fichier de configuration par défaut se trouve dans C:\Program Files\mosquitto\mosquitto.conf. Ouvrez ce fichier avec un éditeur de texte comme le Bloc-notes et configurez les options souhaitées, par exemple :

```
# Autoriser les connexions anonymes (non sécurisé)
allow_anonymous true
```

```
# Définir le port d'écoute
listener 1883
```

Enregistrez le fichier après avoir effectué les modifications.

Étape 3 : Démarrage de Mosquitto

Pour démarrer le service Mosquitto, ouvrez une invite de commande en tant qu'administrateur et exécutez :

```
net start mosquitto
```

Pour arrêter le service, utilisez :

```
net stop mosquitto
```

Vérification de l'installation

Pour vérifier que Mosquitto fonctionne correctement, vous pouvez utiliser les clients Mosquitto. Par exemple, pour publier un message sur le sujet test :

```
mosquitto_pub -h localhost -t test -m "Hello MQTT"
```

Et pour vous abonner au même sujet :

```
mosquitto_sub -h localhost -t test
```

Annexe 5 : Documentation ModBus

Lien documentation :

<http://btssnir15.go.yj.fr/Cours/ModBus/modbus.html>

Ressources utilisées :

Quelques codes de fonction publiques:

01 (01 hex) Read Discrete Output Coils
05 (05 hex) Write single Discrete Output Coil
15 (0F hex) Write multiple Discrete Output Coils
02 (02 hex) Read Discrete Input Contacts
04 (04 hex) Read Analog Input Registers
03 (03 hex) Read Analog Output Holding Registers
06 (06 hex) Write single Analog Output Holding Register
16 (10 hex) Write multiple Analog Output Holding Registers

Annexe 6 : Documentation ModBus TCP pour WallBox

Lien de la documentation GitHub :

<https://github.com/goecharger/go-eCharger-API-v1/blob/master/go-eCharger%20Modbus%20TCP%20API%20v1%20EN.md>

Ressources utilisées :

Dans la partie "Connection"

Connection

The Modbus TCP connection goes as follows:

| Connection | Port |
|------------|------|
| WLAN | 502 |

The go-e Charger has the unitId of 1.

Dans le tableau de la partie "ModBus Register"

Lignes utilisées :





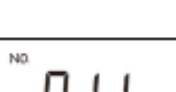
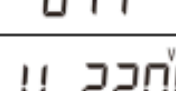
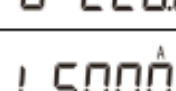
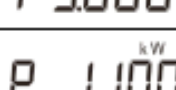
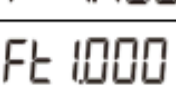

(Valeurs en décimal)

| | | | | | |
|-----|-----------------|------------------|-----------------------|---|--|
| 200 | ALLOW | Holding Register | unsigned integer (16) | 1 | allow_charging: PWM Signal is allowed to abut 0: no 1: yes |
| 299 | AMPERE_VOLATILE | Holding Register | unsigned integer (16) | 1 | Amps Value for the PWM signaling in whole amps of 6-32A Is not saved in the EEPROM and is reset to the value last saved in the EEPROM during the next boot process. For energy control |

Annexe 7 : Numéro d'esclave

The instrument has button display and backlight function, please see button displayed items in table2-2.

a. 2-2Button displayed items

| No. | Content | Description |
|-----|---|---|
| 1 |  | Current combined active energy =0.20 kWh |
| 2 |  | Current positive active energy Imp = 1.20 kWh |
| 3 |  | Current reverse active energy Exp = 1.00 kWh |
| 4 |  | n.1. data format to be eight bits,none parity bit and one stop bit. 9600: baud rate to be 9600 bps 4800: baud rate to be 4800 bps |
| 5 |  | Comm.Add=11 |
| 6 |  | Voltage U=220.0 V |
| 7 |  | Current I=5.000 A |
| 8 |  | Active power P=1.100 kW |
| 9 |  | Power factor Ft=1.000 |
| 10 |  | Frequency F=50.00 Hz |

Note: Backlight closed without button operation for sixty seconds.

Note: The combined active energy default by the factory is equal to positive active energy.

- Parameter setting function

The sensor can set the communication address and baud rate through buttons.

Setting method please see figure 2-4: Long press the button 3s, the sensor will automatically enter into the communication address setting interface,

Annexe 8 : codes Luka

cacquipuissance.h

```
#ifndef CAQUIPUISSANCE_H
#define CAQUIPUISSANCE_H

#include "czdc.h"
#include "cdecodemodbus.h"

class CaquiPuissance
{
public:
    CaquiPuissance(CZDC *zdc);
    void LirePuissance(int numReg);
private:
    CZDC *lazdc;
    CDecodeModbus modbus;
};

#endif
```

cacquipuissance.cpp

```
#include "cacquipuissance.h"

CaquiPuissance::CaquiPuissance(CZDC *zdc): modbus(zdc->getNomLiaison())
{
    lazdc=zdc;    // pour pointer sur "notrezdc" crée par le main
}

void CaquiPuissance::LirePuissance(int numReg)
{
    int p=modbus.fonction3(numReg);
    if(p!=-111111)    lazdc->setPuissance(p);
}
```


cdecodemodbus.h

```
#ifndef CDECODEMODBUS_H
#define CDECODEMODBUS_H

#include "serialib.h"

using namespace std;

class CDecodeModbus
{
public:
    CDecodeModbus(string nom);

    int fonction3(int numreg);
private:
    serialib liaison;
    char ouvertureOK;
    int trame;
};

#endif // CDECODEMODBUS_H
```

cdecodemodbus.cpp

```
#include "cdecodemodbus.h"
#include "czdc.h"

CDecodeModbus::CDecodeModbus(std::string nom)
{
    // liaison.Open(nom,9600);
    cout << "dans CDecodemodbus "<<nom<<endl;
    char ret=liaison.Open(nom.c_str(),9600);
    if (ret==1) cout << "ouverture serie OK"<<endl; else cout <<
"OUverture PAS OK ;)"<<endl;
}

int CDecodeModbus::fonction3(int numreg)
{
    CZDC zdc;

    // appel methode read de l'objet liaison => reception de trame
    // capture de TOUS les octets de la trame ...
    /*!
    \brief Read an array of bytes from the serial device (with timeout)
    \param Buffer : array of bytes read from the serial device
    \param MaxNbBytes : maximum allowed number of bytes read
    \param TimeOut_ms : delay of timeout before giving up the reading
    \return 1 success, return the number of bytes read
    \return 0 Timeout reached
    \return -1 error while setting the Timeout
    \return -2 error while reading the byte
    */

    union P
```

```

{
unsigned char octet[4];
float valeur;

};

P puissance;
char trame[200];
char octet, rep;
int i = 0;
unsigned char octetHH, octetHB, octetBH, octetBB;
//int rep=liaison.Read(trame, sizeof(trame), 1000);

do
{
rep=liaison.ReadChar(&octet,500);
if (rep==1) // return 1 success, return the number of bytes read
{
    //cout << "trame qq chose recu: " << (int)octet <<endl;
    trame[i++] = octet;
    // à mettre dans un tableau
}

}while (rep==1);

// il faut chercher dans cette trame les 32 bits (4 octets) qui sont à
l'adresse numreg dans la réponse

int fin=0;
i=0;
do {
    if (trame[i] == 0x0b && trame[i + 1] == 0x03 && trame[i + 2] ==
0x04) {
        fin=1;
    }
    i++;
    //cout << i; cout.flush();

} while (i < sizeof(trame) && fin!=1);

if(fin==1){

octetHH = trame[i+3-1];
octetHB = trame[i+4-1];
octetBH = trame[i+5-1];
octetBB = trame[i+6-1];
cout << hex << "Valeur obtenue: " << (int)octetHH << " " <<
(int)octetHB << " " << (int)octetBH << " " << (int)octetBB << endl;
puissance.octet[0]=octetBB;
puissance.octet[1]=octetBH;
puissance.octet[2]=octetHB;
puissance.octet[3]=octetHH;
puissance.valeur=puissance.valeur*1000;
cout << "Puissance en watt= " << puissance.valeur<< endl;
return (int) puissance.valeur;

} else cout<<"valeur non trouvée"<<endl;
return -111111;

```

```
}
```

czdc.h

```
#ifndef CZDC_H
#define CZDC_H

#include <string>
using namespace std;

class CZDC
{
public:
    CZDC();
    void setPuissance(int p);
    int  getPuissance();
    string getNomLiaison();
    void setNomLiaison(string valeur);
    int  getNumregPuissance();
    void setnumregPuissance(int);
private:
    int puissance;
    string nomLiaison;
    int numregPuissance;
};

#endif // CZDC_H
```

czdc.cpp

```
#include "czdc.h"

CZDC::CZDC()
{

}

void CZDC::setPuissance(int p)
{
    this->puissance=p;
}

int CZDC::getPuissance()
{
    return this->puissance;
}

string CZDC::getNomLiaison()
{
    return this->nomLiaison;
}

void CZDC::setNomLiaison(string valeur)
{
    this->nomLiaison=valeur;
}
```

```

int CZDC::getNumregPuissance()
{
    return this->numregPuissance;
}

void CZDC::setnumregPuissance(int num)
{
    this->numregPuissance=num;
}

```

serialib.h

```

#ifndef SERIALIB_H
#define SERIALIB_H

// Used for TimeOut operations
#include <sys/time.h>

// Include for Linux
#ifdef __linux__
#include <stdlib.h>
#include <sys/types.h>
#include <sys/shm.h>
#include <termios.h>
#include <string.h>
#include <iostream>
// File control definitions
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#endif

class serialib
{
public:
    // Constructor of the class
    serialib    ();

    // Destructor
    ~serialib    ();
    // _____
    // ::: Configuration and initialization :::
    // Open a device
    char  Open          (const char *Device,const unsigned int Bauds);

    // Close the current device
    void  Close();
    // _____
    // ::: Read/Write operation on characters :::

    // Write a char
    char  WriteChar    (char);

    // Read a char (with timeout)
    char  ReadChar     (char *pByte,const unsigned int TimeOut_ms=NULL);

    // _____

```

```

// ::: Read/Write operation on strings :::
// Write a string
char WriteString (const char *String);
// Read a string (with timeout)
int ReadString (char *String,
                char FinalChar,
                unsigned int MaxNbBytes,
                const unsigned int TimeOut_ms=NULL);

// _____
// ::: Read/Write operation on bytes :::

// Write an array of bytes
char Write (const void *Buffer, const unsigned int NbBytes);

// Read an array of byte (with timeout)
int Read (void *Buffer, unsigned int MaxNbBytes, const unsigned
int TimeOut_ms=NULL);

// _____
// ::: Special operation :::

// Empty the received buffer
void FlushReceiver();

// Return the number of bytes in the received buffer
int Peek();

private:
    // Read a string (no timeout)
    int ReadStringNoTimeOut (char *String, char FinalChar, unsigned int
MaxNbBytes);

#ifdef __linux__
    int fd;
#endif

};

// Class TimeOut
class TimeOut
{
public:
    // Constructor
    TimeOut();

    // Init the timer
    void InitTimer();

    // Return the elapsed time since initialization
    unsigned long int ElapsedTime_ms();

private:
    struct timeval PreviousTime;
};

#endif // SERIALIB_H

```

serialib.cpp

```
#include "serialib.h"

// Class constructor
serialib::serialib()
{}

// Class desctructor
serialib::~serialib()
{
    Close();
}

// _____
// ::: Configuration and initialization :::

/*!
    \brief Open the serial port
    \param Device : (/dev/ttyS0, /dev/ttyACM0, /dev/ttyUSB0 ... for linux)
    \param Bauds : Baud rate of the serial port.

    \n Supported baud rate for Linux :\n
        - 110
        - 300
        - 600
        - 1200
        - 2400
        - 4800
        - 9600
        - 19200
        - 38400
        - 57600
        - 115200

    \return 1 success
    \return -1 device not found
    \return -2 error while opening the device
    \return -3 error while getting port parameters
    \return -4 Speed (Bauds) not recognized
    \return -5 error while writing port parameters
    \return -6 error while writing timeout parameters
*/
char serialib::Open(const char *Device,const unsigned int Bauds)
{
#ifdef __linux__
    struct termios options;
    Structure with the device's options //
```

```

        // Open device
        fd = open(Device, O_RDWR | O_NOCTTY | O_NDELAY);           //
Open port
        if (fd == -1) return -2;                                   //
If the device is not open, return -1
        fcntl(fd, F_SETFL, FNDELAY);                               //
Open the device in nonblocking mode

        // Set parameters
        tcgetattr(fd, &options);                                   //
Get the current options of the port
        bzero(&options, sizeof(options));                           //
Clear all the options
        speed_t      Speed;
        switch (Bauds)                                             //
Set the speed (Bauds)
        {
            case 110 : Speed=B110; break;
            case 300 : Speed=B300; break;
            case 600 : Speed=B600; break;
            case 1200 : Speed=B1200; break;
            case 2400 : Speed=B2400; break;
            case 4800 : Speed=B4800; break;
            case 9600 : Speed=B9600; break;
            case 19200 : Speed=B19200; break;
            case 38400 : Speed=B38400; break;
            case 57600 : Speed=B57600; break;
            case 115200 : Speed=B115200; break;
            default : return -4;
        }
        cfsetispeed(&options, Speed);                               //
Set the baud rate at 115200 bauds
        cfsetospeed(&options, Speed);
        options.c_cflag |= ( CLOCAL | CREAD | CS8);               //
Configure the device : 8 bits, no parity, no control
        options.c_iflag |= ( IGNPAR | IGNBRK );
        options.c_cc[VTIME]=0;                                     //
Timer unused
        options.c_cc[VMIN]=0;                                       //
At least on character before satisfy reading
        tcsetattr(fd, TCSANOW, &options);                           //
Activate the settings
        return (1);                                                 //
Success
#endif
}

/*!
    \brief Close the connection with the current device
*/
void serialib::Close()
{
#ifdef __linux__
    close (fd);
#endif
}

```

```

//
// :: Read/Write operation on characters ::

/*!
    \brief Write a char on the current serial port
    \param Byte : char to send on the port (must be terminated by '\0')
    \return 1 success
    \return -1 error while writting data
*/
char serialib::WriteChar(const char Byte)
{
#ifdef __linux__
    if (write(fd,&Byte,1)!=1) //
Write the char
    return -1; // Error
while writting
    return 1; //
Write operation successfull
#endif
}

//
// :: Read/Write operation on strings ::

/*!
    \brief Write a string on the current serial port
    \param String : string to send on the port (must be terminated by
'\0')
    \return 1 success
    \return -1 error while writting data
*/
char serialib::WriteString(const char *String)
{
#ifdef __linux__
    int Lenght=strlen(String); //
Lenght of the string
    if (write(fd,String,Lenght)!=Lenght) //
Write the string
    return -1; // error
while writing
    return 1; //
Write operation successfull
#endif
}

//
// :: Read/Write operation on bytes ::

/*!

```



```

        \brief Write an array of data on the current serial port
        \param Buffer : array of bytes to send on the port
        \param NbBytes : number of byte to send
        \return 1 success
        \return -1 error while writting data
    */
char seriallib::Write(const void *Buffer, const unsigned int NbBytes)
{
#ifdef __linux__
    if (write (fd, Buffer, NbBytes) != (ssize_t)NbBytes)
        // Write data
        return -1; // Error
    while writing
        return 1; //
    Write operation successfull
#endif
}

/*!
    \brief Wait for a byte from the serial device and return the data read
    \param pByte : data read on the serial device
    \param TimeOut_ms : delay of timeout before giving up the reading
        If set to zero, timeout is disable (Optional)
    \return 1 success
    \return 0 Timeout reached
    \return -1 error while setting the Timeout
    \return -2 error while reading the byte
    */
char seriallib::ReadChar(char *pByte, unsigned int TimeOut_ms)
{
#ifdef __linux__
    TimeOut Timer; // Timer
    used for timeout
    Timer.InitTimer(); //
    Initialise the timer
    while (Timer.ElapsedTime_ms() < TimeOut_ms || TimeOut_ms == 0) //
    While Timeout is not reached
    {
        switch (read(fd, pByte, 1)) { // Try to
        read a byte on the device
        case 1 : return 1; // Read
        successfull
        case -1 : return -2; // Error
        while reading
        }
        }
        return 0;
    }
#endif

/*!
    \brief Read a string from the serial device (without Timeout)
    \param String : string read on the serial device
    \param FinalChar : final char of the string

```

```

        \param MaxNbBytes : maximum allowed number of bytes read
        \return >0 success, return the number of bytes read
        \return -1 error while setting the Timeout
        \return -2 error while reading the byte
        \return -3 MaxNbBytes is reached
    */
int serialib::ReadStringNoTimeOut(char *String,char FinalChar,unsigned int
MaxNbBytes)
{
    unsigned intNbBytes=0; // Number
of bytes read
    char ret; //
Returned value from Read
    while (NbBytes<MaxNbBytes) //
While the buffer is not full
    { //
Read a byte with the restant time
    ret=ReadChar(&String[NbBytes]);
    if (ret==1) // If a
byte has been read
    {
        if (String[NbBytes]==FinalChar) //
Check if it is the final char
        {
            String [++NbBytes]=0; // Yes :
add the end character 0
            return NbBytes; // Return
the number of bytes read
        }
        NbBytes++; //
If not, just increase the number of bytes read
    }
    if (ret<0) return ret; // Error
while reading : return the error number
    }
    return -3; //
Buffer is full : return -3
}

/*!
    \brief Read a string from the serial device (with timeout)
    \param String : string read on the serial device
    \param FinalChar : final char of the string
    \param MaxNbBytes : maximum allowed number of bytes read
    \param TimeOut_ms : delay of timeout before giving up the reading
(optional)
    \return >0 success, return the number of bytes read
    \return 0 timeout is reached
    \return -1 error while setting the Timeout
    \return -2 error while reading the byte
    \return -3 MaxNbBytes is reached
*/
int serialib::ReadString(char *String,char FinalChar,unsigned int
MaxNbBytes,unsigned int TimeOut_ms)
{
    if (TimeOut_ms==0)
        return ReadStringNoTimeOut(String,FinalChar,MaxNbBytes);

    unsigned intNbBytes=0; // Number
of bytes read

```

```

        char        ret;                                //
Returned value from Read
        Timeout     Timer;                              // Timer
used for timeout
        long int    TimeoutParam;
        Timer.InitTimer();                              //
Initialize the timer

        while (NbBytes<MaxNbBytes)                      //
While the buffer is not full
        {                                                //
Read a byte with the restant time
        TimeoutParam=Timeout_ms-Timer.ElapsedTime_ms(); //
Compute the Timeout for the call of ReadChar
        if (TimeoutParam>0)                             // If the
parameter is higher than zero
        {
                ret=ReadChar(&String[NbBytes],TimeoutParam); //
Wait for a byte on the serial link
                if (ret==1)                                //
If a byte has been read
                {

                        if (String[NbBytes]==FinalChar)    // Check
if it is the final char
                        {
                                String [++NbBytes]=0;      // Yes :
add the end character 0
                                return NbBytes;            // Return
the number of bytes read
                        }
                        NbBytes++;                          // If
not, just increase the number of bytes read
                }
                if (ret<0) return ret;                     //
Error while reading : return the error number
        }
        if (Timer.ElapsedTime_ms()>Timeout_ms) {         //
Timeout is reached
                String[NbBytes]=0;                        //
Add the end caracter
                return 0;                                  //
Return 0
        }
        }
        return -3;                                        //
Buffer is full : return -3
}

/*!
\brief Read an array of bytes from the serial device (with timeout)
\param Buffer : array of bytes read from the serial device
\param MaxNbBytes : maximum allowed number of bytes read
\param Timeout_ms : delay of timeout before giving up the reading
\return 1 success, return the number of bytes read
\return 0 Timeout reached
\return -1 error while setting the Timeout
\return -2 error while reading the byte
*/

```

```

int serialib::Read (void *Buffer,unsigned int MaxNbBytes,unsigned int
TimeOut_ms)
{
#ifdef __linux__
    Timeout Timer; //
Timer used for timeout
    Timer.InitTimer(); //
Initialise the timer
    unsigned int NbByteRead=0;
    while (Timer.ElapsedTime_ms()<TimeOut_ms || TimeOut_ms==0) //
While Timeout is not reached
    {
        unsigned char* Ptr=(unsigned char*)Buffer+NbByteRead; //
Compute the position of the current byte
        int Ret=read(fd, (void*)Ptr,MaxNbBytes-NbByteRead); // Try to
read a byte on the device
        if (Ret==-1) return -2; // Error
while reading
        if (Ret>0) { // One or
several byte(s) has been read on the device
            NbByteRead+=Ret; //
Increase the number of read bytes
            if (NbByteRead>=MaxNbBytes) //
Success : bytes has been read
                return 1;
        }
    }
    return 0; //
Timeout reached, return 0
#endif
}

```

```

// _____
// ::: Special operation :::

```

```

/*!
    \brief Empty receiver buffer (UNIX only)
*/

```

```

void serialib::FlushReceiver()
{
#ifdef __linux__
    tcflush(fd,TCIFLUSH);
#endif
}

```

```

/*!
    \brief Return the number of bytes in the received buffer (UNIX only)
    \return The number of bytes in the received buffer
*/
int serialib::Peek()
{

```

```

        int Nbytes=0;
#ifdef __linux__
        ioctl(fd, FIONREAD, &Nbytes);
#endif
        return Nbytes;
    }

// *****
// Class TimeOut
// *****

/#!/
    \brief      Constructor of the class TimeOut.
*/
// Constructor
TimeOut::TimeOut()
{}

/#!/
    \brief      Initialise the timer. It writes the current time of the
day in the structure PreviousTime.
*/
//Initialize the timer
void TimeOut::InitTimer()
{
    gettimeofday(&PreviousTime, NULL);
}

/#!/
    \brief      Returns the time elapsed since initialization. It write
the current time of the day in the structure CurrentTime.
                Then it returns the difference between CurrentTime and
PreviousTime.
    \return     The number of microseconds elapsed since the functions
InitTimer was called.
*/
//Return the elapsed time since initialization
unsigned long int TimeOut::ElapsedTime_ms()
{
    struct timeval CurrentTime;
    int sec,usec;
    gettimeofday(&CurrentTime, NULL); //
Get current time
    sec=CurrentTime.tv_sec-PreviousTime.tv_sec; //
Compute the number of second elapsed since last call
    usec=CurrentTime.tv_usec-PreviousTime.tv_usec; //
Compute
    if (usec<0) { //
If the previous usec is higher than the current one
        usec=1000000-PreviousTime.tv_usec+CurrentTime.tv_usec; //
Recompute the microseconds
        sec--; //
Substract one second
    }
    return sec*1000+usec/1000;
}

```

Annexe 9 : codes Dorian

REVE_FINAL.pro

```
TEMPLATE = app
CONFIG += console c++17
CONFIG -= app_bundle
CONFIG -= qt

SOURCES += \
    CConfig.cpp \
    CParametrer.cpp \
    INIParser.cpp \
    TCPClient.cpp \
    caquipuissance.cpp \
    cdecodemodbus.cpp \
    cmodbus.cpp \
    cmqtt.cpp \
    cpiloterwb.cpp \
    ctraitement.cpp \
    czdc.cpp \
    kbhit.cpp \
    main.cpp \
    mqtt.cpp \
    serialib.cpp

DISTFILES += \
    config.ini

HEADERS += \
    CConfig.h \
    CParametrer.h \
    INIParser.h \
    TCPClient.h \
    caquipuissance.h \
    cdecodemodbus.h \
    cmodbus.h \
    cmqtt.h \
    cpiloterwb.h \
    ctraitement.h \
    czdc.h \
    kbhit.h \
    mqtt.h \
    serialib.h

LIBS += -L. -lpaho-mqtt3cs -lpthread
INCLUDEPATH += /home/pi/paho.mqtt.cpp/paho.mqtt.c/src/
INCLUDEPATH += /home/btssnir/paho.mqtt.cpp/paho.mqtt.c/src/

target.path=/home/pi/Cible/REVE
TARGET=REVE_FINAL
INSTALLS += target
```

czdc.h

```
#ifndef CZDC_H
#define CZDC_H
```

```

#include <string>
using namespace std;

class CZDC
{
public:
    CZDC();
    void setPuissance(int p);
    int getPuissance();
    int getNumregPuissance();
    void setnumregPuissance(int num);
    bool getAutoDepassement();
    void setAutoDepassement(bool autoDep);
    int get_idTrame();

    // Liaison série
    string getNomLiaison();
    void setNomLiaison(string valeur);

    // broker MQTT
    void set_ipBroker(string ipBroker);
    string get_ipBroker();

    void set_port_broker(string port_broker);
    string get_port_broker();

    void set_clientid(string clientid);
    string get_clientid();

    void set_identifiant(string identifiant);
    string get_identifiant();

    void set_motDePasse(string motDePasse);
    string get_motDePasse();

    // WallBox
    void setAddWB(string ipWB);
    string getAddWB();

    void set_port_WallBox(int port_WallBox);
    int get_port_WallBox();

    void set_esclave_WallBox(int esclave);
    int get_esclave_WallBox();

    void set_courant_WallBox(int puis);
    int get_courant_WallBox();

    // Topics MQTT
    void set_topicDepassement(string topic_depassement);
    string get_topicDepassement();

    void set_topic_puiAqaise(string topic_puissance_aqaise);
    string get_topic_puiAqaise();

```

```

private:
    int puissance=0;
    int numregPuissance;
    bool auDepassement=0;
    int idTrame=0;
    int courant_WB;

    string port_serie;

    string ip_broker;
    string port_broker;
    string identifiant;
    string motDePasse;
    string clientid;

    string ip_WallBox;
    int port_WallBox;
    int esclave_WallBox;

    string topic_depassement;
    string topic_puisAquise;
};

#endif // CZDC_H

```

czdc.cpp

```

#include "czdc.h"

CZDC::CZDC()
{
}

// Liaison série
void CZDC::setNomLiaison(string liaison)
{
    this->port_serie=liaison;
}
string CZDC::getNomLiaison()
{
    return port_serie;
}

// Broker MQTT

```



```

void CZDC::set_ipBroker(string ip_broker) // ip du broker
{
    this->ip_broker=ip_broker;
}
string CZDC::get_ipBroker()
{
    return ip_broker;
}

void CZDC::set_port_broker(string port_broker) // port du broker
{
    this->port_broker=port_broker;
}
string CZDC::get_port_broker()
{
    return ip_broker;
}

// client MQTT
void CZDC::set_identifiant(string identifiant) // identifiant du client
{
    this->identifiant=identifiant;
}
string CZDC::get_identifiant()
{
    return identifiant;
}

void CZDC::set_motDePasse(string motDePasse) // mot de passe du client
{
    this->motDePasse=motDePasse;
}
string CZDC::get_motDePasse()
{
    return motDePasse;
}

void CZDC::set_clientid(string clientid) // id du client sur le broker
{
    this->clientid=clientid;
}
string CZDC::get_clientid()
{
    return clientid;
}

// WallBox
void CZDC::setAddWB(string ip)
{
    this->ip_WallBox=ip;
}
string CZDC::getAddWB()
{
    return ip_WallBox;
}

```

```

void CZDC::set_esclave_WallBox(int esclave)
{
    this->esclave_WallBox=esclave;
}
int CZDC::get_esclave_WallBox()
{
    return esclave_WallBox;
}

void CZDC::set_port_WallBox(int port)
{
    this->port_WallBox=port;
}
int CZDC::get_port_WallBox()
{
    return port_WallBox;
}

void CZDC::set_courant_WallBox(int puis)
{
    this->courant_WB=puis;
}
int CZDC::get_courant_WallBox()
{
    return courant_WB;
}

}

// Topics MQTT
void CZDC::setAutoDepassement(bool autoDep)
{
    this->auDepassement=autoDep;
}
bool CZDC::getAutoDepassement()
{
    return auDepassement;
}

void CZDC::set_topicDepassement(string Depassement) // id du client sur le broker
{
    this->topic_depassement=Depassement;
}
string CZDC::get_topicDepassement()
{
    return topic_depassement;
}

void CZDC::set_topic_puiAqaise(string topic_puissance_aqaise) // id du client sur le broker
{
    this->topic_puisAqaise=topic_puissance_aqaise;
}
string CZDC::get_topic_puiAqaise()
{
    return topic_puisAqaise;
}

void CZDC::setPuissance(int p)

```

```

{
    this->puissance=p;
}
int CZDC::getPuissance()
{
    return puissance;
}

void CZDC::setnumregPuissance(int reg)
{
    this->numregPuissance=reg;
}
int CZDC::getNumregPuissance()
{
    return numregPuissance;
}

int CZDC::get_idTrame()
{
    idTrame++;
    return idTrame-1;
}

```

CParametrer.h

```

#ifndef CPARAMETRER_H
#define CPARAMETRER_H
#include "czdc.h"

#include "CConfig.h"

class CParametrer
{
public:
    CParametrer(CZDC *azdc);

    void EnvoiTopicDepassement_CZDC(bool payload);
    void LireFichierConfig();
    void EnvoiTopicPuisAquis(int puis);

    string TopicDepassement();
    string TopicPuisAquis();

private:
    CConfig fichierConfig;
    CZDC *lazdc;
};

#endif // CPARAMETRER_H

```

CParametrer.cpp

```
#include "CParametrer.h"
#include "CConfig.h"
#include "czdc.h"
#include <iostream>

using namespace std;

CParametrer::CParametrer(CZDC *azdc)
{
    // l'attribut zdc prend l'adresse azdc
    lazdc=azdc;
}

void CParametrer::LireFichierConfig()
{
    // va chercher les infos dans le config.ini
    string ip_broker=fichierConfig.ip_broker();
    string port_broker=fichierConfig.port_broker();
    string clientid_broker=fichierConfig.clientid_broker();
    string identifiant_broker=fichierConfig.identifiant_broker();
    string motDePasse_broker=fichierConfig.motDePasse_broker();

    string ip_WallBox=fichierConfig.ip_WallBox();
    char esclave_WallBox=fichierConfig.esclave_WallBox();
    int port_WallBox=fichierConfig.port_Wallbox();

    string topic_depassement= fichierConfig.topic_depassement();
    string topic_puisAquis = fichierConfig.topic_puisAquis();

    string nomLiaison = fichierConfig.port_serie();

    // ajoute dans la zdc les infos du config.ini récupérer précédement
    lazdc->set_ipBroker(ip_broker);
    lazdc->set_port_broker(port_broker);
    lazdc->set_identifiant(identifiant_broker);
    lazdc->set_motDePasse(motDePasse_broker);
    lazdc->set_clientid(clientid_broker);
    lazdc->setAddWB(ip_WallBox);
    lazdc->set_port_WallBox(port_WallBox);
    lazdc->set_esclave_WallBox(esclave_WallBox);
    lazdc->setNomLiaison(nomLiaison);

    lazdc->set_topicDepassement(topic_depassement);
    lazdc->set_topic_puiAquis(topic_puisAquis);
}

void CParametrer::EnvoiTopicDepassement_CZDC(bool payload)
{
    lazdc->setAutoDepassement(payload);
}

void CParametrer::EnvoiTopicPuisAquis(int puis)
```

```

{
    lazdc->setPuissance(puis);
}

string CParametrer::TopicDepassement()
{
    string topic=lazdc->get_topicDepassement();
    return topic;
}

string CParametrer::TopicPuisAquis()
{
    string topic=lazdc->get_topic_puiAquise();
    return topic;
}

```

mqtt.h

```

#ifndef LINUX_MQTT
#define LINUX_MQTT

#define QOS                1
#define TIMEOUT             5000L
#define KEEP_ALIVE         20

#include <MQTTClient.h>
#include "CParametrer.h"
#include <iostream>
#include <vector>

using namespace std;

void MQTTBegin(CParametrer *objet,vector<string> infos);
void MQTTSubscribe(const char* topic);
void MQTTPublish(const char* topic, char* message);

#endif /* LINUX_MQTT */

```

mqtt.cpp

```

#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include "mqtt.h"

MQTTClient monclient;
vector<string> infosBroker;
CParametrer *parametre;

using namespace std;

```

```

volatile MQTTClient_deliveryToken deliveredtoken;

//confirme le message
void delivered(void *context, MQTTClient_deliveryToken dt)
{
    printf("Message with token value %d delivery confirmed\n", dt);
    deliveredtoken = dt;
}

//affiche le message envoyé par le client a un topic précis
int msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message
*message)
{
    bool etat = false;
    char* payload;
    string messageArvd;

    string topic;

    printf("\n\n-----Message arrived-----\n");
    printf("Topic : %s\n", topicName);

    payload = static_cast<char*>(message->payload);
    messageArvd=payload;

    topic = topicName;

    cout<<"Payload : "<<payload<<endl;

    cout<<"Vrai trame a traiter "<<payload<<" , du topic "<<topicName<<endl;
    cout<<"-----"<<endl;

    // pour le topic depassement : on remplit la zdc
    if (topic==parametre->TopicDepassement())
    {
        if(messageArvd=="on")etat=1; //si message reçu on alors etat=1 (allume)
        else etat=0; //si message reçu off alors etat=0 (eteind)
        parametre->EnvoiTopicDepassement_CZDC(etat);
    }

    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    return 1;
}

//s'affiche si la connexion est perdue
void connlost(void *context, char *cause)
{
    printf("\nConnection lost\n");
    printf("    cause: %s\n", cause);
}

//s'abonne aux topics
void MQTTSubscribe(const char* topic)
{
    printf("Subscribing to topic %s for client %d using QoS%d\n\n", topic,
QOS);
}

```

```

MQTTClient_subscribe(monclient, topic, QOS); // monclient souscrit au topic
}

void MQTTBegin(CParametrer*objet, vector<string> brokerInfo)
{
    infosBroker=brokerInfo; // remplissage variables globales
    parametre=objet;

    // MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;

    //initié connexion en ssl
    MQTTClient_SSLOptions ssl_opts = MQTTClient_SSLOptions_initializer;
    ssl_opts.enableServerCertAuth = 0;
    //déclarer des valeurs pour les options ssl, ici nous n'utilisons que celles
nécessaires pour TLS, mais on peut éventuellement en définir beaucoup plus
    ssl_opts.verify = 1;
    ssl_opts.CApath = NULL;
    ssl_opts.keyStore = NULL;
    ssl_opts.trustStore = NULL;
    ssl_opts.privateKey = NULL;
    ssl_opts.privateKeyPassword = NULL;
    ssl_opts.enabledCipherSuites = NULL;

    int rc;
    int ch;

    //création du client
    MQTTClient_create(&monclient, brokerInfo[1].c_str(), brokerInfo[4].c_str(),
        MQTTCLIENT_PERSISTENCE_NONE, NULL);
    //connexion en ssl
    conn_opts.ssl = &ssl_opts;

    //paramètre de la connexion (mdp, username...)
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;
    conn_opts.username = brokerInfo[0].c_str();
    conn_opts.password = brokerInfo[2].c_str();
    MQTTClient_setCallbacks(monclient, NULL, connlost, msgarrvd, delivered);

    //vérifie si la connexion a réussi ou pas
    if ((rc = MQTTClient_connect(monclient, &conn_opts)) != MQTTCLIENT_SUCCESS)
    {
        printf("Failed to connect, return code %d\n", rc);
        exit(EXIT_FAILURE);
    }

    printf("Abonnement au topic : '%s' pour le client '%s' utilisant
QoS%d\n\n",brokerInfo[3].c_str(), brokerInfo[4].c_str(), QOS);
    MQTTClient_subscribe(monclient,brokerInfo[3].c_str(), QOS);

    return ;
}

void MQTTPublish(const char* topic, char* message)
{
    int payloadlen = strlen(message);
    MQTTClient_deliveryToken dtt;
    MQTTClient_publish(monclient, topic, payloadlen, message, QOS, 0, &dtt);
}

```

```
printf("published to %s \n", topic);
}
```

cmqtt.h

```
#ifndef CMQTT_H
#define CMQTT_H

#include "mqtt.h"
#include <vector>
//#include "CParametrer.h"

using namespace std;

class CMQTT
{
public:
    CMQTT(CParametrer *objet, vector<string> infos);
    ~CMQTT();
    void sabonner(vector<string> listtopic);
    void abonnerUnTopic(string topic);
    void publier(const char* topic, char* payload);
private:

};

#endif // CMQTT_H
```

cmqtt.cpp

```
#include "cmqtt.h"
#include <iostream>

using namespace std;

//permet de se connecter au broker
CMQTT::CMQTT(CParametrer *objet, vector<string> infos)
{
    cout<<endl<<"----- Infos connection au broker MQTT
    -----"<<endl<<endl;

    //affiche la liste du broker
    cout<<"# Identifiant => "<<infos[0]<<endl;
    cout<<"# IP du broker MQTT => "<<infos[1]<<endl;
    cout<<"# Mot de passe => "<<infos[2]<<endl;
    cout<<"# Topic actuellement abonné => "<<infos[3]<<endl;
    cout<<"# Client id => "<<infos[4]<<endl;
    cout << " après lecture vector ..."<<endl<<endl;
    MQTTBegin(objet,infos);
}
```



```

//destructeur
CMQTT::~CMQTT()
{
}

//permet de s'abonner aux topics
void CMQTT::sabonner(vector<string> listtopic)
{
    for(unsigned int i=0;i<listtopic.size();i++)
    {
        MQTTSubscribe(listtopic[i].c_str());
    }
}

void CMQTT::abonnerUnTopic(string topic)
{
    MQTTSubscribe(topic.c_str());
}

void CMQTT::publier(const char* topic, char *payload)
{
    MQTTPublish(topic, payload);
}

```

INIParser.h

```

#ifndef INIPARSER_H
#define INIPARSER_H

#include <fstream>
#include <sstream>
#include <string>
#include <map>

class INIParser
{
private:
    std::map<std::string, std::map<std::string, std::string> > ini;
    std::string FileName;
    bool AutoSave;

public:
    INIParser(const std::string &, bool=false);
    ~INIParser()
    {
        if(AutoSave) save();
        ini.clear();
    };
    // permet de récupérer une valeur, et retourne une valeur par
    // défaut si la clef n'est pas trouvée
    template <class T> T GetValue(const std::string &, const
std::string &, const T &);
    int GetValue(const std::string &, const std::string &, const std::string
&);
    // permet d'enregistrer une valeur, ou dans modifier une
    // existante.

```

```

        template <class T> void SetValue(const std::string &, const
std::string &, const T &);
        // enregistre la map dans un fichier.
        bool save(std::string="");
};

#endif // INIPARSER_H

```

INIParser.cpp

```

#include "INIParser.h"

template <class T> T INIParser::GetValue(const std::string &Section, const
std::string &clef, const T &defaultValue)
{
    std::map<std::string, std::map<std::string, std::string> >::iterator
_it=ini.find(Section);

    if(_it != ini.end()) // si la valeur Section n'est pas trouvé
    {
        std::map<std::string, std::string>::iterator it=_it->second.find(clef);

        if(it != _it->second.end()) // si la valeur clef n'est pas trouvé
        {
            // permet la conversion de la valeur en type std::string dans le type
demandé
            T val;
            std::istringstream iss(it->second);
            iss >> val;
            return val;
        }
        else
            return defaultValue;
    }
    else
        return defaultValue;
}

// spécialisation de la fonction pour la class std::string
template <> std::string INIParser::GetValue<std::string>(const std::string
&Section, const std::string &clef, const std::string &defaultValue)
{
    std::map<std::string, std::map<std::string, std::string> >::iterator
_it=ini.find(Section);

    if(_it != ini.end())
    {
        std::map<std::string, std::string>::iterator it=_it->second.find(clef);

        if(it != _it->second.end())
            return it->second;
        else

```

```

        return defaultValue;
    }
    else
        return defaultValue;
}

int INIParser::GetValue(const std::string &Section, const std::string &clef,
const std::string &defaultValue)
{
    std::map<std::string, std::map<std::string, std::string> >::iterator
_it=ini.find(Section);

    if(_it != ini.end())
    {
        std::map<std::string, std::string>::iterator it=_it->second.find(clef);

        if(it != _it->second.end())
        {
            std::string rep=it->second;
            int i = std::stoi(rep);
            return i;
        }
        else
        {
            int i = std::stoi(defaultValue);
            return i;}
    }
    else
    {
        int i = std::stoi(defaultValue);
        return i;}
}

template <class T> void INIParser::SetValue(const std::string &Section, const
std::string &clef, const T &Value)
{
    std::ostringstream oss;
    // conversion de la valeur en std::string
    oss << Value;
    // enregistrement de la valeur dans la map
    ini[Section][clef] = oss.str();
}

INIParser::INIParser(const std::string &filename, bool autoSave)
{
    std::string section, line, clef, valeur;
    size_t pos;

    AutoSave = autoSave;
    FileName = filename;

    std::ifstream file(filename.c_str());

    while (!file.eof())
    {
        std::getline (file, line);

        // suppression des commentaires
        pos = line.find_first_of(';');

```

```

        if(pos != std::string::npos)        line.erase        (line.begin()        +        pos,
line.end());

// continue si la ligne n'ai pas vide
if(!line.empty())
{
    // test si la ligne coorespond a une section
    pos = line.find_first_of('[');
    if(pos != std::string::npos)
    {
        line.erase(line.begin(), line.begin() + pos+1);
        line.erase(line.begin() + line.find_first_of (']'), line.end());
        section = line;
    }
    else // sinon elle coorespond a une clef
    {
        pos = line.find_first_of('=');
        // si le '=' a bien été trouvé
        if(pos != std::string::npos)
        {
            clef = line.substr (0, pos);
            valeur = line.substr (pos+1);

            // permet la suppression de tout les espaces dans la clef
            while(std::string::npos != (pos = clef.find_first_of(' ')))
                clef.erase(pos);

            ini[section][clef] = valeur;
        }
    }
}

file.close();
}

bool INIParser::save(std::string filename)
{
    std::map<std::string, std::map<std::string, std::string> >::iterator _it;
    std::map<std::string, std::string>::iterator it;

    std::ofstream file;

    if(filename == "")
        file.open(FileName.c_str());
    else
        file.open(filename.c_str());

    if(!file.is_open())    return false;

    // ecriture de la map dans le fichier demandé
    for(_it=ini.begin(); _it!=ini.end(); ++_it)
    {
        file << "[" << _it->first << "]" << std::endl;

        for(it=_it->second.begin(); it!=_it->second.end(); ++it)
            file << it->first << "=" << it->second << std::endl;
    }

    file.close();
}

```

```
    return true;  
}
```

Annexe 10 : codes Lucas

TCPClient.h

```
#include <cstdint>
#include <stdint.h>

class TCPClient {
public:
    TCPClient();
    bool connected();
    bool connect(const char* domain, uint16_t port);
    size_t write(uint8_t b);
    size_t write(const char* str, size_t len);
    size_t print(const char* str);
    size_t println();
    size_t println(const char* str);
    int available();
    int read();
    void flush();
    void stop();

protected:
    int sockfd;
};
```

TCPClient.cpp

```
#include <sys/ioctl.h>
#include <netdb.h>
#include <string.h>
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include "TCPClient.h"

TCPClient::TCPClient() : sockfd(-1) { }

bool TCPClient::connected() {
    return sockfd >= 0;
}

bool TCPClient::connect(const char* domain, uint16_t port) {
    struct addrinfo hints;
    struct addrinfo *res;

    // Specify TCP stream socket (don't mind if ipv4 or ipv6)
    memset(&hints, 0, sizeof hints);
    hints.ai_family = AF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;

    // Specify host and port
    char portString[5];
    sprintf(portString, "%d", port);
```

```

    getaddrinfo(domain, portString, &hints, &res);

    // Create the socket
    if((sockfd = socket(res->ai_family, res->ai_socktype,
res->ai_protocol)) < 0) {
        return false;
    }

    // Connect the socket
    if(::connect(sockfd, res->ai_addr, res->ai_addrlen) < 0) {
        return false;
    }

    // Cleanup
    freeaddrinfo(res);

    return true;
}

size_t TCPClient::write(uint8_t b) {
    return write((char *)&b, 1);
}

size_t TCPClient::write(const char* str, size_t len) {
    return ::send(sockfd, str, len, 0);
}

size_t TCPClient::print(const char* str) {
    return write(str, strlen(str));
}

size_t TCPClient::println() {
    return print("\n");
}

size_t TCPClient::println(const char* str) {
    size_t bytes = 0;
    bytes += print(str);
    bytes += print("\n");

    return bytes;
}

int TCPClient::available() {
    uint8_t buf;
    ::recv(sockfd, &buf, 1, MSG_PEEK);

    return buf > 0;
}

int TCPClient::read() {
    uint8_t buf;
    ::recv(sockfd, &buf, 1, 0);
    return buf;
}

void TCPClient::flush() {
    while(available()) read();
}

```

```

void TCPClient::stop() {
    if (sockfd != -1) {
        while (close(sockfd) < 0) {
            usleep(10000);
        }
        sockfd = -1;
    }
}

```

cmodbus.h

```

#ifndef CMODBUS_H
#define CMODBUS_H
#define MASQUE 0x00FF
#define PROTOCOLE 0x0000
#define FONCTION6 0x06
#define OFFSETTAILLE 6
#include "TCPClient.h"
#include <iostream>
using namespace std;
class CModbus
{
public:
    CModbus(string ad,int port, char es);
    void setEsclave(char es);
    char getEsclave();
    void ft3(int a);
    int ft6(int reg,int val);

    TCPClient TCP;
private:
    uint16_t portsocket = 502;
    char esclave;
    int numtransaction;

};

#endif // CMODBUS_H

```

cmodbus.cpp

```

#include "cmodbus.h"
#include <cstring>

CModbus::CModbus(string ad, int port, char es)
{
    const char* adresse =ad.c_str();
    if((TCP.connect(adresse,port))==true)
        cout << "Connexion réussie !" << endl;
    else
        cout << "Erreur de connexion" << endl;

    this->esclave=es;
    this->numtransaction=0;
}

```



```

void CModbus::ft3(int a)
{
    //const char* trame ="0003
    //TCP.write(trame, 12);
}

int CModbus::ft6(int reg, int val)
{
    char hautTransaction=this->numtransaction>>8;char
basTransaction=this->numtransaction&MASQUE;
//Identificateur
    char hautPro=PROTOCOLE>>8; char basPro=PROTOCOLE&MASQUE; //Numéro
de Protocole
    char hautReg=reg>>8;char basReg=reg&MASQUE; //Numéro
de Registre
    char hautVal=val>>8;char basVal=val&MASQUE; //Valeur
du Registre

    char
trame[]={hautTransaction,basTransaction,hautPro,basPro,0x00,0x00,this->esclav
e,FONCTION6,hautReg,basReg,hautVal,basVal};
    int taille = sizeof(trame)-OFFSETTAILLE;
    char hautTai=taille>>8;char basTai=taille&MASQUE;
    trame[4]=hautTai; trame[5]=basTai;
    int rep=TCP.write(trame,sizeof(trame));

    this->numtransaction = this->numtransaction + 1;

    if(rep==sizeof(trame))return 1;
    else return -1;
}

```

cpiloterwb.h

```

#ifndef CPILOTERWB_H
#define CPILOTERWB_H
#include <string>
#include "cmodbus.h"
#define REGAMPEREVOLATILE 299
#define REGALLOW 200
#define CONSIGNEMINIWB 6

class CPiloterWB
{
public:
    CPiloterWB(string ad, int por, char es);
    int setConsignecharge(int consigne);
private:
    CModbus maitre;
};

#endif // CPILOTERWB_H

```

cpiloterwb.cpp

```
#include "cpiloterwb.h"

CPiloterWB::CPiloterWB(string ad, int port, char es):maitre(ad,port,es)
{
    // creer un objet de type client modbusTCP
}

int CPiloterWB::setConsignecharge(int consigne)
{
    // appel fonctionxxxx modbus en passant la consigne
    //.....
    //Modbus.ft6(consigne);

    if(consigne>=CONSIGNEMINIWB)
    {
        int val = maitre.ft6(REGAMPEREVOLATILE, consigne);
        val=maitre.ft6(REGALLOW,1);
        return val;
    }
    else
    {
        int val=maitre.ft6(REGALLOW,0);
        return val;
    }
}
```

ctraitement.h

```
#ifndef CTRAITEMENT_H
#define CTRAITEMENT_H
#define Tension 230
#define CourantMini 6
#define CourantMaxi 32
#define PCMini Tension * CourantMini
#define PCMaxi Tension * CourantMaxi

#include "czdc.h"
#include "cpiloterwb.h"

enum etat{charge,arretcharge};

class Ctraitement
{
public:
    Ctraitement(CZDC *zdc);
    void run();
private:
    CZDC *lazdc;
    CPiloterWB wb;
    int id;
    int PuissanceCharge;
    etat etatCharge;
    bool depassement;
```

```

        int courantCharge;
        bool arretenvoye;
    };

#endif // CTRAITEMENT_H

```

ctraitement.cpp

```
#include "ctraitement.h"
```

```

Ctraitement::Ctraitement(CZDC *zdc): wb(zdc->getAddWB(), zdc->getPort(),
zdc->getEsclave())
{
    lazdc=zdc;
    this->PuissanceCharge=0;                this->etatCharge=arretcharge;
    this->depasement=false; this->courantCharge=0;
    this->arretenvoye=false;
}

void Ctraitement::run()
{
    int nbpasPuissance = 0;
    int puissance = lazdc->getPuissance();
    //this->depasement=lazdc.getdepasss

    //pour tests:
    puissance=puissance+this->PuissanceCharge;

    cout << "Puissance moins puissanceCharge: " << puissance<<endl;

    switch(etatCharge) // faire évoluer machine à étét
    {
    case arretcharge: if(puissance < -PCMini) etatCharge=charge;break;
    case charge: if(puissance > (Tension*(int)this->depasement))
    etatCharge=arretcharge;break; // en fonction du dépassement
    }

    switch(etatCharge) // que faire dans chaque état
    {
    case arretcharge:
    this->PuissanceCharge=0;
    this->courantCharge=0;
    if(!this->arretenvoye){
        wb.setConsignecharge(this->courantCharge); // arreter WB et
mettre puissance charge à 0
        this->arretenvoye=true;
    }

    // voir pour envoyer qu'une eule fois ...
    break;
    case charge:
    // calcul puissance de charge et commander WB
        nbpasPuissance=-puissance/(Tension*1); // voir signe
    ???

    // en fonction du dépassement
    if(this->depasement==true)

```

```

        {
            nbpasPuissance=nbpasPuissance+1;
        }
        courantCharge=courantCharge+nbpasPuissance;
        this->PuissanceCharge=(courantCharge)*Tension;
        cout << "courantCharge : " << courantCharge << endl;
        // si puissance charge a changée alors piloter WB
        if(nbpasPuissance!=0)=
        {
            wb.setConsignecharge(courantCharge);
            this->arretenvoye=false;
        }

        break;
    }

    cout << "Nbpas puissance = " << nbpasPuissance << endl;
    cout << "PuissanceCharge = " << this->PuissanceCharge << endl;
    switch(etatCharge){
        case 0: cout << "EtatCharge = 0 (Allumé)" << endl << endl;
                break;
        case 1: cout << "EtatCharge = 1 (Eteint)" << endl << endl;
                break;
    }
}

```

czdc.h

```

#ifndef CZDC_H
#define CZDC_H

#include <string>
using namespace std;

class CZDC
{
public:
    CZDC();
    void setPuissance(int p);
    int getPuissance();
    string getNomLiaison();
    void setNomLiaison(string valeur);
    int getNumregPuissance();
    void setnumregPuissance(int);
    void setAddWB(string ad);
    string getAddWB();
    void setPort(int port);
    int getPort();
    void setEsclave(int esclave);
    char getEsclave();
    void setId(int id);
    int getId();
    void setPuissanceCharge(int puissanceCharge);
    int getPuissanceCharge();
    void setChargeWB(bool chargeWB);
    bool getChargeWB();

private:

```

```

        int puissance;
        string nomLiaison;
        int numregPuissance;
        string adresseWB;
        int port;
        char esclave;
        int id;
        int puissanceCharge;
        bool chargeWB;
};

#endif // CZDC_H

```

czdc.cpp

```

#include "czdc.h"

CZDC::CZDC()
{

}

void CZDC::setPuissance(int p)
{
    this->puissance=p;
}

int CZDC::getPuissance()
{
    return this->puissance;
}

string CZDC::getNomLiaison()
{
    return this->nomLiaison;
}

void CZDC::setNomLiaison(string valeur)
{
    this->nomLiaison=valeur;
}

int CZDC::getNumregPuissance()
{
    return this->numregPuissance;
}

void CZDC::setnumregPuissance(int num)
{
    this->numregPuissance=num;
}

void CZDC::setAddWB(string ad)
{
    this->adresseWB=ad;
}

string CZDC::getAddWB()

```

```

{
    return this->adresseWB;
}

void CZDC::setPort(int port)
{
    this->port=port;
}

int CZDC::getPort()
{
    return port;
}

void CZDC::setEsclave(int esclave)
{
    this->esclave=esclave;
}

char CZDC::getEsclave()
{
    return esclave;
}

void CZDC::setId(int id)
{
    this->id=id;
}

int CZDC::getId()
{
    return id;
}

int CZDC::getPuissanceCharge()
{
    return puissanceCharge;
}

void CZDC::setPuissanceCharge(int puissanceCharge)
{
    this->puissanceCharge=puissanceCharge;
}

void CZDC::setChargeWB(bool chargeWB)
{
    this->chargeWB=chargeWB;
}

bool CZDC::getChargeWB()
{
    return chargeWB;
}

```

Annexe 11 : kbhit.h

```
#ifndef _KBHIT_H
#define _KBHIT_H

#include <termios.h>
#include <unistd.h>
#include <stdlib.h>

class Keyboard
{
private:
    struct termios initial_settings, new_settings;
public:
    Keyboard();
    ~Keyboard();
    void Initialisation();
    void Arret();
    int kbhit();
    int getch();
};

#endif/* _KBHIT_H */
```

Annexe 12 : kbhit.cpp

```
#include "kbhit.h"

int Keyboard::getch()
{
    char ch;
    read(0,&ch,1);
    return ch;
}

Keyboard::Keyboard()
{
    Keyboard::Initialisation();
}

Keyboard::~~Keyboard()
{
    Keyboard::Arret();
}

int Keyboard::kbhit()
{
    char ch;
    int nread;
    new_settings.c_cc[VMIN]=0;
    tcsetattr(0,TCSANOW, &new_settings);
    nread = read(0,&ch,1);
    new_settings.c_cc[VMIN]=1;
    tcsetattr(0,TCSANOW, &new_settings);
    if (nread == 1)
    {
        return ch;
    }
}
```

```

    }
    return 0;
}

void Keyboard::Initialisation()
{
    tcgetattr(0,&initial_settings);
    new_settings = initial_settings;
    new_settings.c_lflag &= ~ICANON;
    new_settings.c_lflag &= ~ECHO;
    new_settings.c_lflag &= ~ISIG;
    new_settings.c_oflag &= ~NL0;
    new_settings.c_oflag &= ~CR0;
    new_settings.c_oflag &= ~TAB0;
    new_settings.c_oflag &= ~BS0;
    new_settings.c_cc[VMIN] = 1;
    new_settings.c_cc[VTIME] = 0;
    cfsetospeed (&new_settings, B230400);
    cfsetispeed(&new_settings, 0);
    tcsetattr(0, TCSANOW, &new_settings);
}

void Keyboard::Arret()
{
    tcsetattr(0, TCSANOW, &initial_settings);
}

```

Annexe 13 : config.ini

```

[Liaison_série]
port_serie=/dev/ttyUSB0
debit=9600

[connexion_BrokerMQTT]
ip=172.16.10.1
port=1883
clientid=client1
id=client
mdp=client

[connexion_WallBox]
ip=172.16.11.1
port=502
esclave=0

[topic]
topic_depassement=depassement
topic_puissanceAquis=puisAquis
topic_autTransfWallbox_Voiture=autTransfertWallbox
topic_charge=charge
topic_energieEnvoyerWallbox=envoiWallbox

```


Annexe 14 : Configuration du client MQTT

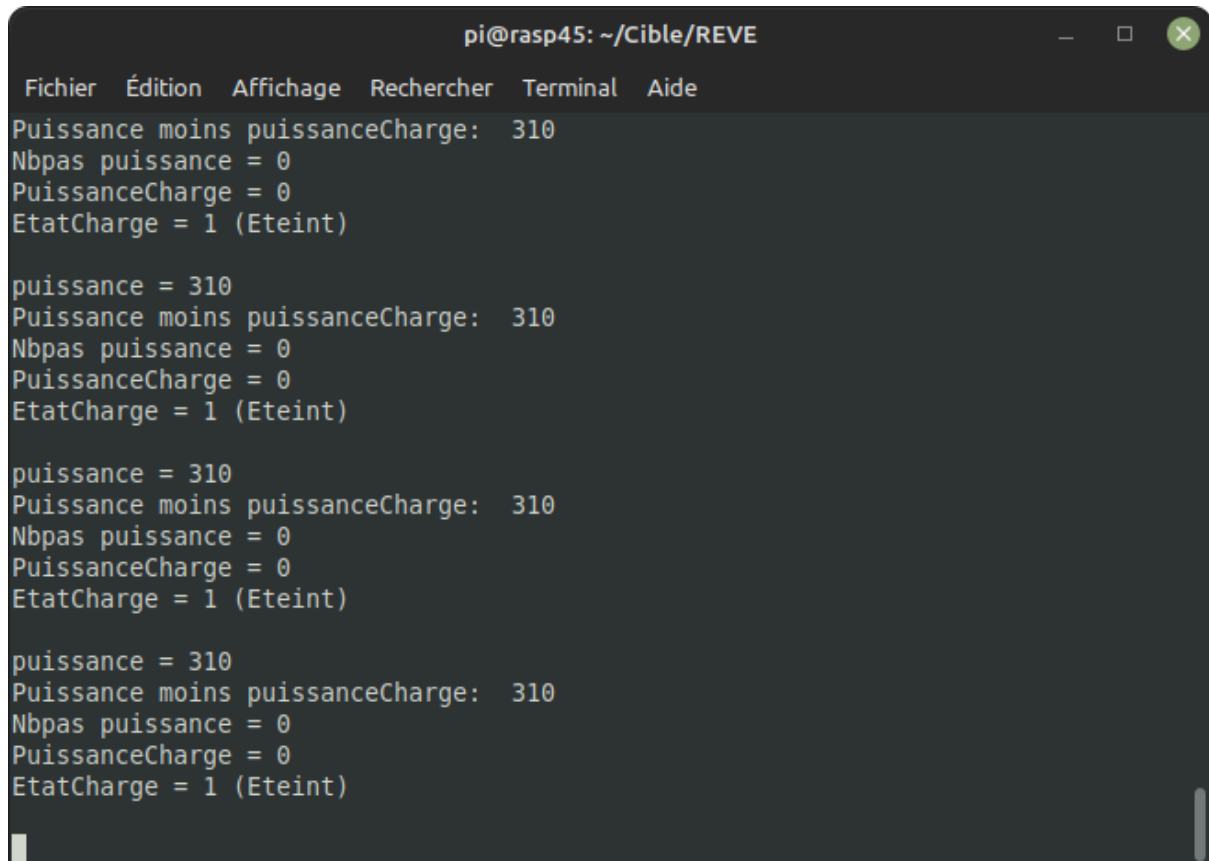
Configuration de la connexion :

| | |
|----------------------|---|
| Adresse IP du broker | 172.16.10.1 en local lgt-lycee-des-metiers-jean-mon.pro.dns-orange.fr pour accès distant |
| Port | 1883 |
| Client id | doit être unique |
| Identifiant | client |
| Mot de passe | client |

Configuration des topics :

| Topics | Nom des topics | type d' élément | Valeur |
|--------------------------------|---------------------|-----------------|--------------------------|
| topic_depassement | depassement | commutateur | on off |
| topic_puissanceAquise | puisAquis | gauge | mini: -3000 max: 9000 |
| topic_autTransfWallbox_Voiture | autTransfertWallbox | commutateur | on off |
| topic_energieEnvoyerWallbox | envoiWallbox | gauge | mini: 0 max: 7360 |
| topic_charge | charge | commutateur | on off |

Annexe 15 : Tests individuels Lucas Serrée-Delpeux



```
pi@rasp45: ~/Cible/REVE
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
Puissance moins puissanceCharge: 310
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = 310
Puissance moins puissanceCharge: 310
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = 310
Puissance moins puissanceCharge: 310
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = 310
Puissance moins puissanceCharge: 310
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)
```

Au lancement du programme, la puissance de départ pour les tests individuels est de 310W.

La valeur est positive, il n'y a pas de surproduction d'énergie, la WallBox n'a donc pas lieu de charger le véhicule électrique, on remarque bien que la WallBox est en état 1 : arretcharge.

```
pi@rasp45: ~/Cible/REVE
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
Puissance moins puissanceCharge: -1300
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = -1300
Puissance moins puissanceCharge: -1300
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = -1300
Puissance moins puissanceCharge: -1300
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = -1300
Puissance moins puissanceCharge: -1300
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)
```

La puissance est diminuée jusqu'à atteindre -1300, il y a donc une surproduction d'énergie, cependant, -1300 est toujours supérieur à la puissance minimale ($-6 * 230 = -1380W$).
On voit que la WallBox est toujours bien éteinte.

```
pi@rasp45: ~/Cible/REVE
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
puissance moins puissanceCharge: -1300
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = -1300
Puissance moins puissanceCharge: -1300
Nbpas puissance = 0
PuissanceCharge = 0
EtatCharge = 1 (Eteint)

puissance = -1530
Puissance moins puissanceCharge: -1530
courantCharge : 6
Nbpas puissance = 6
PuissanceCharge = 1380
EtatCharge = 0 (Allumé)

puissance = -1530
Puissance moins puissanceCharge: -150
courantCharge : 6
Nbpas puissance = 0
PuissanceCharge = 1380
EtatCharge = 0 (Allumé)

puissance = -1760
Puissance moins puissanceCharge: -380
courantCharge : 7
Nbpas puissance = 1
PuissanceCharge = 1610
EtatCharge = 0 (Allumé)

puissance = -1760
Puissance moins puissanceCharge: -150
courantCharge : 7
Nbpas puissance = 0
PuissanceCharge = 1610
EtatCharge = 0 (Allumé)

puissance = -1990
Puissance moins puissanceCharge: -380
courantCharge : 8
Nbpas puissance = 1
PuissanceCharge = 1840
EtatCharge = 0 (Allumé)
```

(La puissance continue à diminuer progressivement)

Finalement, quand la puissance minimale est dépassée, on note que comme prévu, la WallBox autorise la charge (Etat à 0) et commence à 6 Ampères.

Ensuite la valeur Puissance moins puissance charge est calculée grâce à la valeur NbPas puissance qui augmente au nombre de pas franchis en un cycle.

La puissance diminue continuellement, lorsque que celle-ci devient inférieure à -230W.

Annexe 16 : Test Unitaire Dorian

| | | | | |
|--|--|--|--|-------------------|
| <input type="checkbox"/> Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale | | | | |
| <input checked="" type="checkbox"/> Test unitaire | | | | |
| <input type="checkbox"/> Test d'intégration | | | | |
| Objectif : Pouvoir stocker l'énergie du panneau solaire qui n'est pas consommé par la maison, dans la batterie de la voiture électrique. | | | | |
| Éléments à tester : Classe CMQTT et CParametrer du projet REVE | | | | |
| Conditions de réalisation : - Serveur MQTT Configuré - Client MQTT Configuré | | | | |
| Initialisation : | | | | |
| Scénario | | | | |
| ID | Démarche / Opération | Données en entrée / Condition initiale | Comportement / Résultat attendu | Status : OK / NOK |
| 1 | Mettre à 1 le bouton lié à l'autorisation de dépassement sur l'application cliente | | Affichage du nom et du payload du topic | OK |
| 2 | | | Enregistrement du payload du topic reçu dans la Zone de Donnée Commune | OK |
| 3 | Mettre à 0 le bouton lié à l'autorisation de dépassement sur l'application cliente | | Affichage du nom et du payload du topic | OK |
| 4 | | | Enregistrement du payload du topic reçu dans la Zone de Donnée Commune | OK |
| 5 | IHM application : touche 'e' | Valeur de la puissance 300 | Affichage du nom et du payload du topic envoyé | OK |
| | | | sur l'application cliente, la gauge liée à la puissance prend 300 | OK |

| | | | |
|-----------------|--|--------------------------|-------------|
| Rapport de test | | Testé par : Agnel Dorian | Le : |
| Commentaire : | | | |
| | | Approbation : | Signature : |

Annexe 17 : Test unitaire Mayonade Luka

| | | | | |
|--|------------------------|--|--|--------------------------|
| <input type="checkbox"/> Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale | | | | |
| <input checked="" type="checkbox"/> Test unitaire | | | | |
| <input type="checkbox"/> Test d'intégration | | | | |
| Objectif : Stocker le surplus d'énergie dans une batterie de voiture électrique reçu par le panneau photovoltaïque | | | | |
| Éléments à tester : Classe <u>cacquisition</u> Stockage du surplus d'énergie | | | | |
| Conditions de réalisation : - Simulateur du compteur d'énergie opérationnel et en service - Connexion sur l'interface série ttyUSB0 | | | | |
| Initialisation : | | | | |
| Scénario | | | | |
| ID | Démarche / Opération | Données en entrée / Condition initiale | Comportement / Résultat attendu | Status : OK / <u>NOK</u> |
| 1 | Lancement du programme | | Récupération des octets reçus et placement dans un tableau d'octet | OK |
| 2 | | | Décodage de la trame et visualisation des 4 octets | OK |
| 3 | | | Visualisation du flottant correspondant | OK |
| 4 | | | Écriture dans de l'entier correspondant dans la zone de donnée commune | OK |
| 5 | | | | |
| | | | | |

| | | |
|-------------------------|----------------------|----------------------------------|
| Rapport de test Le : | | Testé par : <u>Mayonade Luka</u> |
| Commentaire : | | |
| | Approbation : | Signature : |

Annexe 18 : Test Unitaire Serrée-Delpeux Lucas

- ☐ Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
☒ Test unitaire
☐ Test d'intégration

Objectif : Stocker un surplus d'énergie électrique dans la batterie d'un véhicule électrique.

Éléments à tester : Programme en C++ communiquant avec une WallBox.

Conditions de réalisation :

- Programme capable de communiquer avec la Wallbox
- Programme capable de récupérer les informations sur l'énergie transmise sur le réseau élec.
- Programme permettant de contenir les informations importantes.

Initialisation : Mesure de la puissance sur le réseau du domicile

| Scénario | | | | |
|----------|------------------------|---|--|-------------------|
| ID | Démarche / Opération | Données en entrée / Condition initiale | Comportement / Résultat attendu | Status : OK / NOK |
| 1 | Lancement du programme | Il n'y a pas de surplus d'énergie dans le domicile. (Puissance mesurée dans la zone de données commune > 0) | La WallBox reste en état d'arrêt. Aucune énergie est transmise. | OK |
| 2 | | Il y a un surplus d'énergie dans le domicile. Le surplus n'est pas suffisant. (-1380 < Puissance mesurée dans la zone de données commune < 0) | La WallBox reste en état d'arrêt. Aucune énergie est transmise. | OK |
| 3 | | Il y a un surplus d'énergie dans le domicile. (Puissance mesurée dans la zone de données commune = -1760) | La WallBox autorise la charge et la valeur d'Intensité est modifiée en conséquence : 7 Ampères | OK |
| | | | | |

| | | |
|---|---------------|-------------|
| Rapport de test | Testé par : | Le : |
| Commentaire : | | |
| | Approbation : | Signature : |