

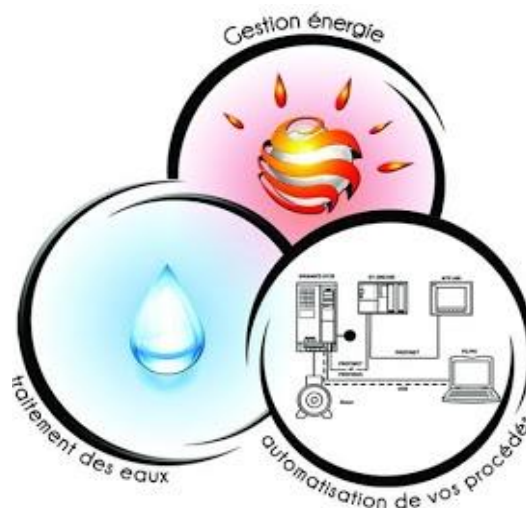


GTC Lampadaire-Revue de projet

- Brochard Matthieu
- Garagnoli Enzo
- Eeckeman Maxime
- Achor Mehdi



Pour l'entreprise API :



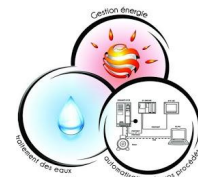
Académie d'Aix-Marseille

Session 2017-2018-Lycée A.BENOIT



Sommaire

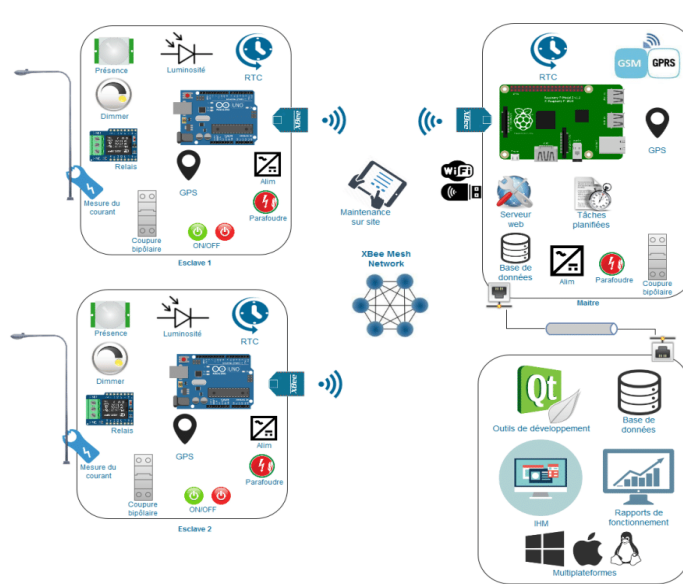
•	Partie commune :	
○	Fiche de modification n°1	4
○	Fiche de modification n°2	5
○	Entreprise API : Présentation	6
○	Pourquoi un tel projet ?	7
○	Présentation du projet	8
○	Prototype du projet	9
○	Diagramme d'exigence des besoins clients :	10
○	Fonctionnement du projet	11
•	Partie individuelle :	
○	Brochard Matthieu	
▪	Partie IR : Brochard Matthieu	12
▪	Diagramme de Gantt	13
▪	Plan des itérations	14
▪	Cas d'utilisation	15
▪	Partie Physique : Les ondes xbee	16-17
▪	Logiciel XCTU	18-19
▪	Activités de mise en œuvre : Itération 1	20
▪	Diagramme séquence interaction	21
▪	Itération 1	22-23
▪	Problèmes et solutions	24
▪	Activités de mise en œuvre : Itération 2	25
▪	Itération 2	26-30
▪	Problèmes et solutions	31
▪	Journal de bord personnel	32
○	Garagnoli Enzo	
▪	Partie : Garagnoli Enzo	33
▪	Diagramme de Gantt	34
▪	Diagramme des cas d'utilisation	35
▪	Diagramme de bloc structure	36
▪	Diagramme de séquence pour l'envoi d'ordre	37
▪	Plan des itérations	38
▪	Cas d'utilisation	39
▪	Activités de mise en œuvre	40
▪	Itération 1	41-42
▪	Problèmes et solutions	43-44
▪	Itération 2	45-48
▪	Problèmes et solutions	49-50
▪	Partie Physique : réseaux Wi-Fi	51-53
▪	Journal de bord personnel	54



Sommaire-suite

○ Eeckeman Maxime	
▪ Partie : Eeckeman Maxime	55
▪ Diagramme de Gantt	56
▪ Plan des parties	57
▪ Cas d'utilisation	58
▪ Activités de mise en œuvre	59-61
▪ Partie 1	62-64
▪ Partie 2	65-69
▪ Partie 3	70-71
▪ Partie Physique	72-73
▪ Problèmes et solutions	74
▪ Journal de bord personnel	75
○ Achor Mehdi	
▪ Partie : Achor Mehdi	76
▪ Les composants	77-78
▪ Description des composants	78-79
▪ Diagramme de gantt	79-80
▪ Cas d'utilisation	80-81
▪ Logiciel KICAD	81-82
▪ Mise en place des composants	83-88
▪ Communication (physique)	88-89
▪ Traitement de la partie dimmer	90-95
▪ Problèmes et solutions	96-97
▪ Journal de bord personnel	98-99
▪ Annexe	100-102
• Annexes	103-111

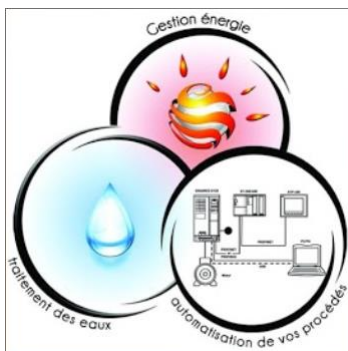
Fiche de modification n°1

2. Fiche de demande de modification.		Réf. FDM-GTC-01
Projet	Demandeur	Date
API GTC (Gestion Technique Centralisée) lampadaires	Société API	10/01/2018
Description de la modification		
Modifications apportées au cahier des charges à la demande MM. Bunel et Jullian, suite à la réunion du 5 janvier 2018 dans les locaux de la société API, en présence de M. Amaudric (Approvisionnement Electrique Avignon)		
 • Chaque lampadaire disposera d'un module esclave piloté par une carte Arduino (Uno ou mini). • Le module maître de chaque segment de lampadaires sera piloté par une carte Raspberry Pi, et sera fixé soit dans l'un des lampadaires du segment, soit sur un bâtiment proche. • Le module devra disposer d'une entrée logique pour y raccorder (bornier à vis) la sortie de type contact sec d'un parafoudre, dont le modèle sera choisi par la société API. • Le module devra disposer d'une entrée logique pour y raccorder (bornier à vis) la sortie de type contact sec d'un détecteur de présence, dont le modèle sera choisi par la société API. • Le module devra disposer d'une entrée analogique pour y raccorder (bornier à vis) un capteur de luminosité, dont le modèle sera choisi par la société API. • Les blocs d'éclairage seront de type panneau de LED, leur puissance sera au maximum de 140W, ceux-ci disposeront d'une entrée analogique (1V à 10V) permettant la gradation de l'intensité lumineuse du lampadaire. Le modèle de panneau de LED utilisé sera choisi par la société API. • Le module maître devra disposer d'une liaison GSM/GPRS de façon à palier toute défaillance du réseau Ethernet. • Tous les modules devront disposer d'un capteur GPS de façon à pouvoir établir une cartographie du réseau de lampadaires.		

Fiche de modification n°2

2. Fiche de demande de modification.		Réf.
		FDM-GTC-02
Projet	Demandeur	Date
API GTC (Gestion Technique Centralisée) lampadaires	Société API	19/04/2018
Description de la modification		
Modifications et précisions apportées au cahier des charges à la demande MM. Bunel, Jullian et Rigal, suite à la réunion de projet du 18 avril 2018 au lycée Alphonse Benoit • La fréquence de récupération des données sera l'un des éléments de la BDD. • Retour des mesures en RAM Rpi tout les 10 à 15 minutes ou plus. Mais écriture sur SD Card Rpi beaucoup moins fréquemment. • Prise en compte des données par le technicien le matin. • Mesure des durées d'éclairement des LED pour maintenance préventive. • Pas de GPS sur l'esclave, uniquement sur RPi maître (une panne éventuelle d'un esclave sera localisée en sachant où est le maître). D'ici la fin de l'année scolaire le GPS n'est pas une priorité. • Le relais sera utilisé non plus pour commander le candélabre, mais pour une application autre de type éclairage festif, exemple : éclairage de Noël. • Faire remonter la température des processeurs Rpi. • Borniers à ressorts préférables à borniers à vis. • Coupure bipolaire conservée. • Boutons de test : mode manuel (marche forcée, arrêt forcé), mode automatique. • Relamping -> économie d'énergie et changement pour une technologie pérenne (Le relamping consiste à moderniser un système d'éclairage en remplaçant des lampes et sources lumineuses obsolètes et inappropriées) • Rpi équipées d'un dissipateur. • Utiliser une dénomination : Maître/Esclave ou Coordinateur/Lampadaire. • Refaire IHM config Coordinateur/Lampadaire. • Accès données prévoir 1 super administrateur/concepteur, 1 administrateur et 1 technicien/consultant. Mots de passe pour mise au point API API. • GTC (Gestion Technique Centralisée) aussi appelée supervision : c'est un visuel à un instant donné de ce qui fonctionne ou pas + historique des événements. • GMAO : Gestion Maintenance Assistée par Ordinateur : suivi de l'entretien, des pièces commandées, etc ...		

Entreprise API : Présentation



La société API est une microentreprise qui est sur plusieurs secteurs d'activité :

- Dans un premier temps elle pratique une activité industrielle ou elle intervient comme l'industrie du verre, l'agroalimentaire ou encore dans l'artisanat

Dans un second, ils pratiquent une activité énergétique avec la proposition de nombreux service telle que :

- Audit des réseaux d'énergie
- Mesures et bilan des consommations
- Préconisation des procédés de gestion, domotique, Immotique
- Étude et calcul d'installation photovoltaïque

L'activité de traitement d'eau est aussi une importante partie de l'entreprise et se spécialise dans les solutions de filtration membranaires de la micro filtration jusqu'à l'osmose inverse.

Le dernier secteur est leurs secteurs le plus récent qui est une activité de formateur depuis Novembre 2016.

Nous proposons donc des formations qui pourront être prises en charges par votre organisme OPCA dans le cadre du DIF ou du CFP.

API propose des formations dans les domaines de l'automatisme et l'informatique industrielle.

Pourquoi un tel projet ?

De nombreux d'appel d'offres ont été formulés auprès de l'entreprise notamment en île de la Réunion.

Celle-ci ont demandé de faire des lampadaires qui serait des candélabres plus économique (ici les lumières utilisées sont des LED disposée comme indiquée l'image sous dessous)



*Tête du lampadaire avec
la disposition des LED*

La solution proposée par l'entreprise est moins cher que les concurrents (Phillips etc...) Ensuite dans les demandes formulées, la question économique a été posée ainsi que l'autonomie soit augmentée, les réverbères pourront être contrôlables avec différents paramètres.

Une des raisons de ses demandes auprès de l'entreprise Api est en rapport avec de nouvelles législations datant de novembre 2012 voulant :

- La préservation des ressources de la planète et plus particulièrement les économies d'énergie et les émissions de CO2
- Exigences de sécurité, de continuité et qualité du service public
- Préoccupation majeure sur la dépense publique
- Possibilité d'appropriation par la collectivité

Présentation du projet

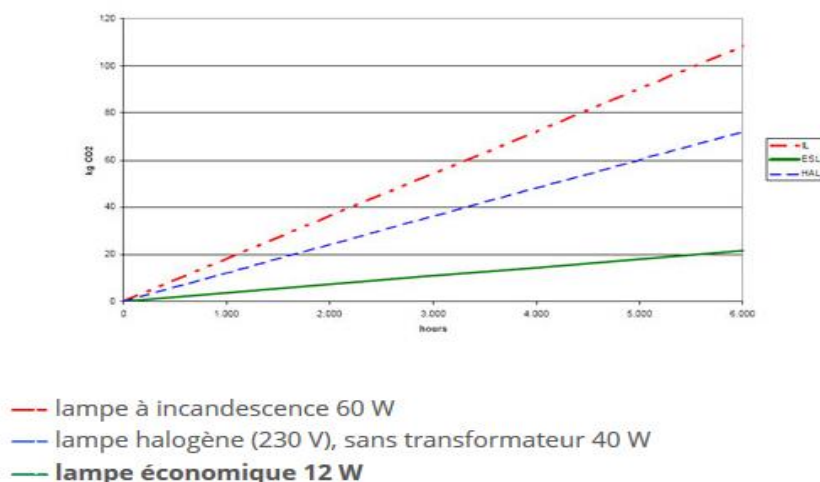
Le projet sur lequel nous travaillons est un système moderne permettant de contrôler un éclairage public grâce à une gestion technique centralisée (GTC).

Ce système va permettre d'éclairer des segments de rue grâce à un capteur de proximité avec un pourcentage de luminosité choisit au préalable.

Le système se présente de cette façon : un coordinateur envoie les ordres qu'il a reçu de la GTC aux lampadaires de son segment en utilisant un réseau maillé ZigBee constitué de modules xbee installés dans chaque lampadaires.

Le contexte de ce projet a avant tout deux objectif majeurs :

- 1) L'aspect économique, il y a plusieurs facteurs à prendre en compte :
 - Le rapport temps/prix : prendre des ampoules LED peut être avantageux en considérant qu'elles ont une durée de vie de + de 15000H ce qui est plus que le reste mais reste cher (10€ en moyenne)
 - Rendement lumineux : les LED ont un rendement lumineux qui est bien plus grand que la concurrence (100 lm/W pour les LED et 60 lm/W pour les lampe basse consommation)
 - Résistance aux cycles allumage/extinction
- 2) L'aspect écologique : 9 millions de lampes fonctionnent entre 3 500 et 4 300 heures par an pour une puissance installée d'environ 1 260 MW. L'utilisation de matériel de nouvelles génération est important car les anciennes générations de lampe consomment beaucoup et produisent beaucoup de Co2 comme pour les lampe à incandescence ou les lampes halogène.



Source de l'image :

<http://www.lampe-de-luminothérapie.com/comparatif-emissions-de-co2-de-3-categories-de-lampes/>

Prototype du projet

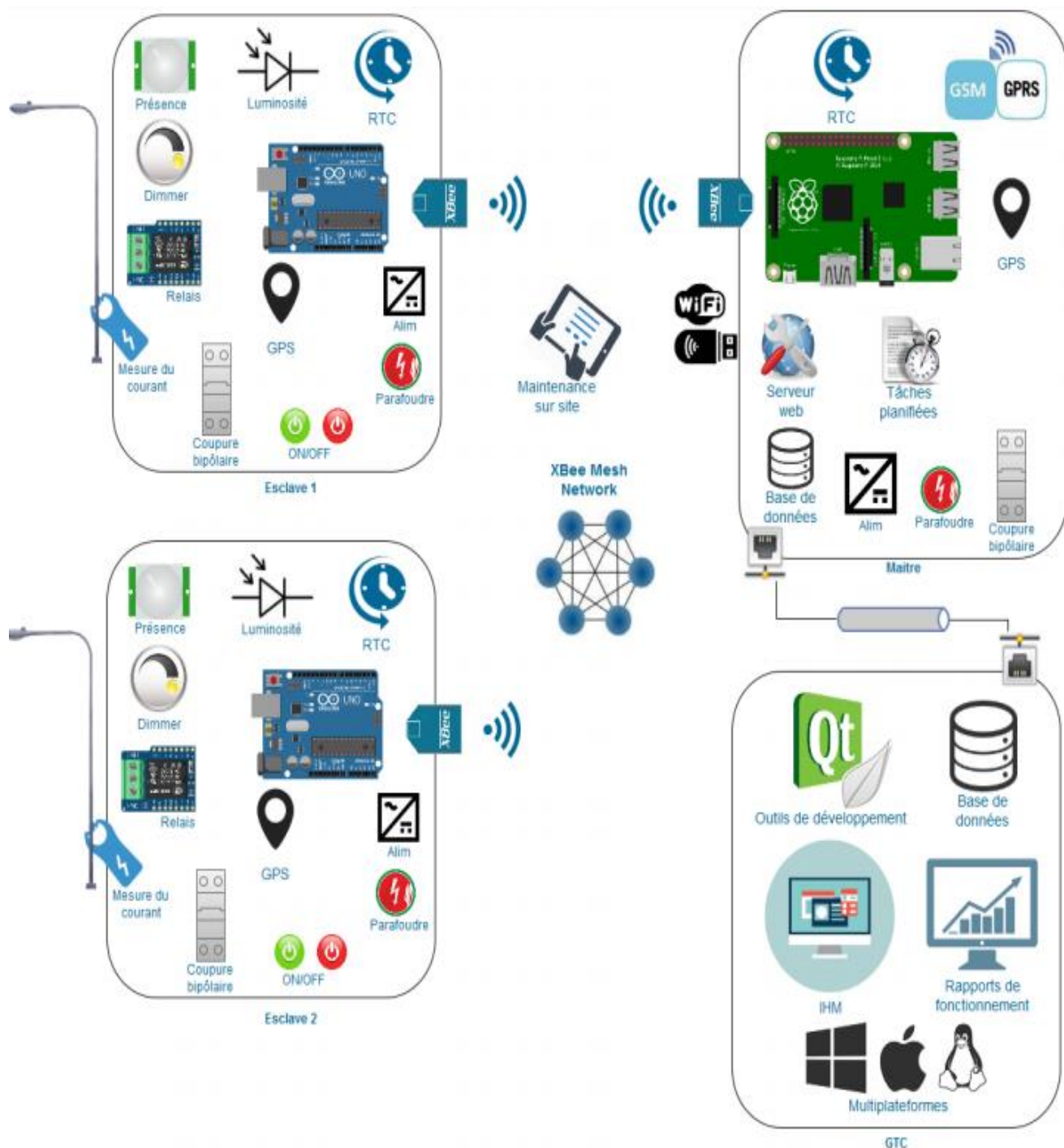
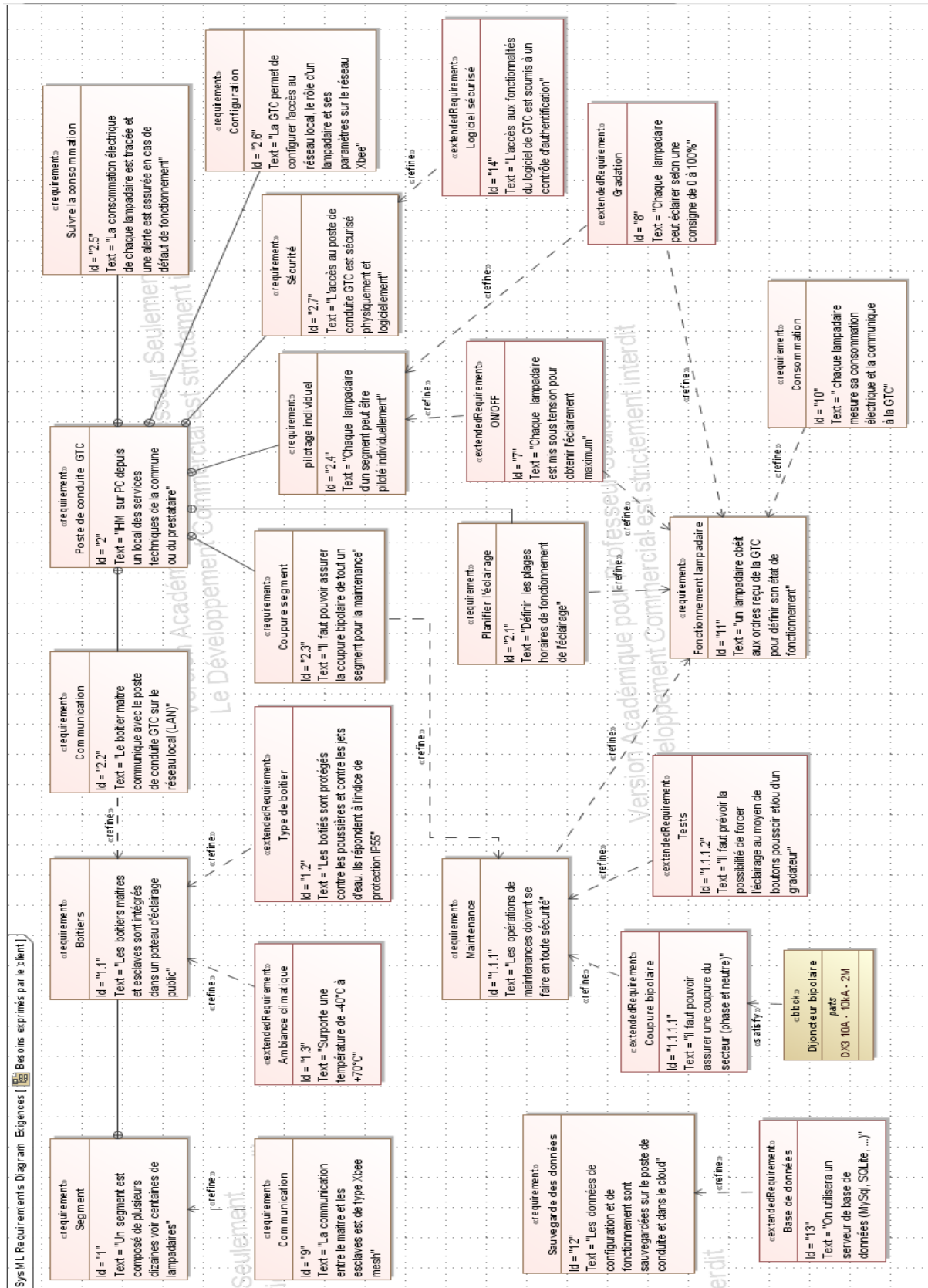


Diagramme d'exigence des besoins clients :





Fonctionnement du projet :

Le logiciel GTC permet d'ajouter, modifier ou supprimer un utilisateur, configurer un segment et configurer un lampadaire. Il peut envoyer des ordres à chaque coordinateur tels qu'allumer ou éteindre un lampadaire seul ou un segment complet, il peut planifier les plages horaires où les lampadaires doivent fonctionner et aussi enregistrer l'historique des consommations.

Pour communiquer avec les segments la GTC et chaque coordinateur ont une Base De Données (BDD).

Les BDD communiquent de cette façon : La GTC envoie un ordre à un coordinateur, l'ordre est écrit dans la BDD du coordinateur qui va y lire à intervalle régulier sa table Ordre et exécuter l'ordre reçu. Si l'ordre reçu est une lecture de consommation, la GTC copie les données dans sa table historique.

Pour communiquer entre eux, les lampadaires et les coordinateurs utilisent des modules xbee qui permettent de communiquer à distance et sans fil sur un réseau ZigBee. Les trames de données transmises par liaison ZigBee sont chiffrées nativement en AES 128, ce qui permet d'assurer la sécurité des données.

Les lampadaires seront équipés de capteurs de présence qui permettront d'augmenter la luminosité automatiquement si une personne passe devant un lampadaire et ainsi permettront d'économiser de l'énergie.

Partie IR: Brochard Matthieu

Ma partie dans le projet concerne la confection du logiciel de la gestion technique centralisée (GTC) :

Je dois créer un logiciel grâce à Qt creator qui permet de configurer les modules Xbee afin d'établir la communication entre les lampadaires et le coordinateur des segments. Le logiciel devra permettre d'ajouter, modifier ou supprimer un utilisateur, configurer un segment, configurer un lampadaire et planifier les plages horaires où les lampadaires doivent fonctionner. Le logiciel devra aussi permettre d'envoyer des ordres à chaque maître de segments tels que contrôler un ou plusieurs lampadaires d'un segment connecté (allumer, éteindre) et enregistrer l'historique des consommations.

Je dois répondre au cas d'utilisation "Configurer", "Contrôler les lampadaires", "Générer les rapports de fonctionnement", "Archiver les mesures de consommation".

Schéma de ma partie :



Diagramme de Gantt

Diagramme de Gantt provisoire:

▲ Tâches Matthieu	34,71 jours	Ven 19/01/18	Ven 15/06/18	
mise en page IHM	20 hr	Ven 19/01/18	Mer 31/01/18	7
coder classe lampadaire	40 hr	Mer 31/01/18	Jeu 15/03/18	9
configuration des cartes esclaves	20 hr	Jeu 15/03/18	Ven 23/03/18	10
Creation de la BDD GTC	70 hr	Ven 23/03/18	Jeu 10/05/18	11
integration de la BDD dans le logiciel	70 hr	Jeu 10/05/18	Jeu 07/06/18	12
IR1 IT1	60 hr	Ven 19/01/18	Lun 12/03/18	
IR1 IT2	118 hr	Mar 13/03/18	Jeu 10/05/18	14
IR1 IT3	89 hr	Ven 11/05/18	Ven 15/06/18	15

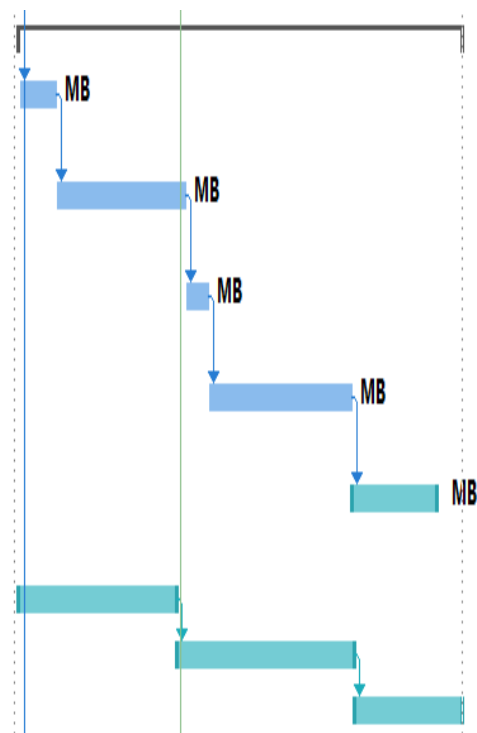
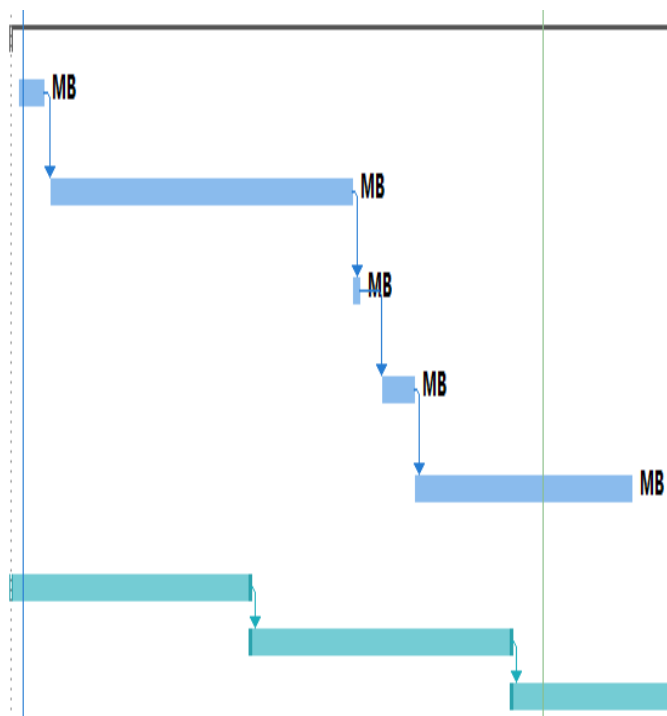


Diagramme de Gantt final :

▲ Tâches Matthieu	35,86 jours?	Jeu 18/01/18	Ven 15/06/18	
mise en page IHM	10 hr	Ven 19/01/18	Jeu 25/01/18	
coder port série	85 hr	Ven 26/01/18	Mer 04/04/18	9
Creation de la BDD GTC	12 hr	Jeu 05/04/18	Ven 06/04/18	10
integration de la BDD dans le logiciel	20 hr	Mer 11/04/18	Mer 18/04/18	11
Coder envoi des ordres sur la BDD	80 hr?	Jeu 19/04/18	Mer 06/06/18	12
IR1 IT1	63 hr	Jeu 18/01/18	Lun 12/03/18	
IR1 IT2	118 hr	Mar 13/03/18	Jeu 10/05/18	14
IR1 IT3	89 hr	Ven 11/05/18	Ven 15/06/18	15



Plan des itérations

Tâche	Itération	Pourquoi?
Faire la mise en page de l'IHM	1	En premier car essentiel au fonctionnement du logiciel et permet de mettre en place toute les choses à faire.
Coder la connexion par port série avec le module xbee	1	Le code le plus important de la partie qui permet de communiquer avec les modules xbee pour configurer l'ID, le réseaux et le rôles (maître/esclaves) ainsi que de se connecter à la BDD.
Création de la BDD	2	La BDD sert à envoyer des ordres au maitres et à stocker les mesures de consommation des lampadaire .
Communiquer avec la BDD	2	Impossible à faire avant car il faut que les modules Xbee soit configurable et que la BDD soit créé/accessible.
Communiquer avec les segments pour envoyer des ordres	3	A faire quand la BDD et les modules communiquent.
Archiver les mesures de consommation sur la BDD	3	Possible quand tous est terminer.



Cas d'utilisation

Cas d'utilisation : Configurer

Scénario nominal : Initialiser le système

Il faut que le logiciel puisse configurer les modules xbee avec les différents paramètres :ID, JV, CE, DH, DL, NI

Cas d'utilisation : Authentifier

Scénario nominal : S'authentifier avant d'utiliser le logiciel

La connexion à la BDD via le logiciel doit être sécurisé par identifiant et mot de passe.

Cas d'utilisation : Contrôler les lampadaires

Scénario nominal : Contrôler des lampadaires d'un segment connecté

Le logiciel doit pouvoir envoyer des ordres à tous les lampadaires d'un segment.

Cas d'utilisation : Contrôler individuellement un lampadaire

Scénario nominal : Contrôler le fonctionnement d'un seul lampadaire

Le logiciel doit pouvoir envoyer des ordres à un lampadaires d'un segment.

Cas d'utilisation : Planifier les plages d'éclairage

Scénario nominal : Planification du mode automatique

Le logiciel doit permettre de choisir les horaires ou les lampadaires s'allument et s'éteignent.

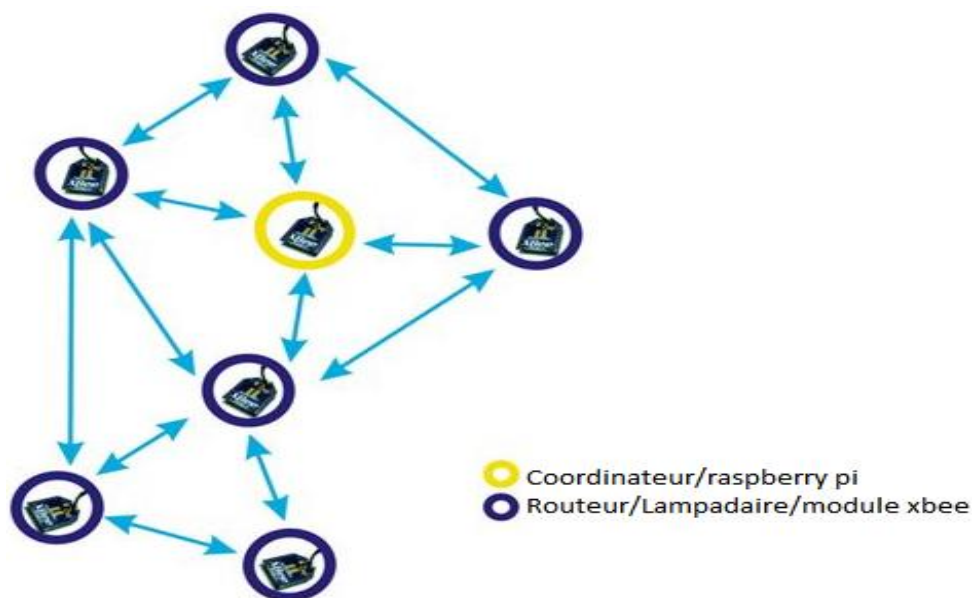
Partie Physique : Les ondes xbee

Pour ce projet, nous utilisons des module XBee, ceux-ci sont implantés sur une carte Arduino ou un Shield.



Ces modules sont capables de communiquer avec les fréquences 2.4Ghz qui sont ce que l'on appelle les Bandes ISM (Bande Industrielle, scientifique et médicale). Ces bandes peuvent être utilisées dans un espace réduit pour des applications industrielles, scientifiques, médicales ou domestiques.

Ici, elles ont pour objectif de transmettre des informations nécessaires aux autres Xbee grâce au système Mesh qui relie des Xbee entre eux.



Le réseaux Mesh est fiable car si un des modules ne marche plus les autres peuvent continuer de communiquer entre eux directement ou en passant par un autre module. Le réseaux Mesh marche quand les différents modules sont éparpillés et non pas dans une ligne.

Pour communiquer entre appareils, nous avons besoin d'antennes.
Celle que nous avons utilisée est l'antenne RP SMA Antenna 2dBi – LM251

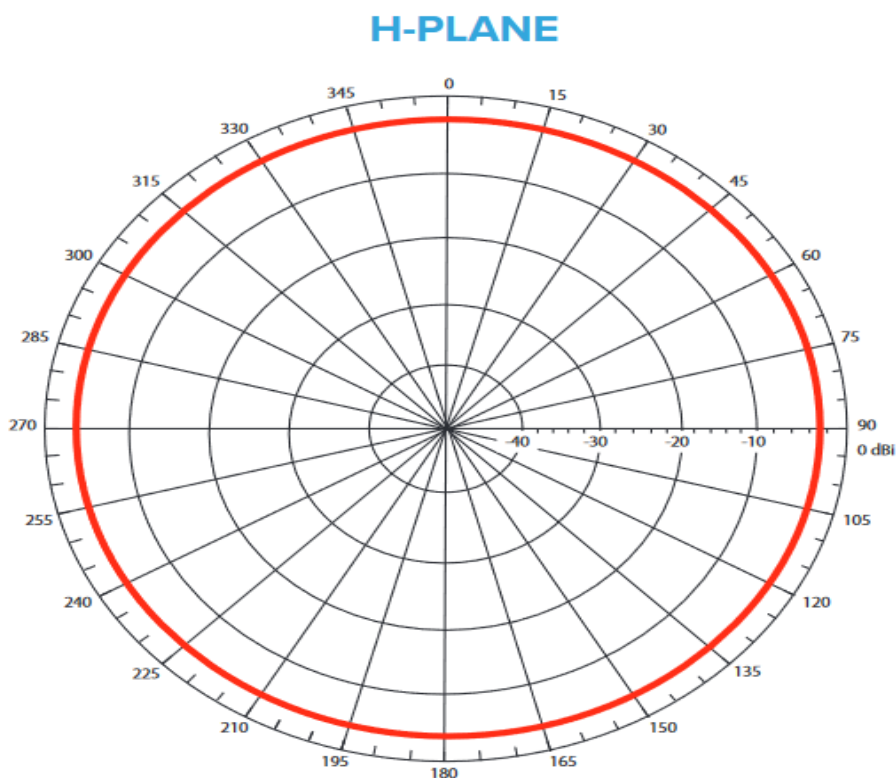
Ces caractéristiques sont les suivantes :

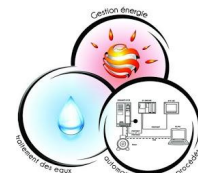
Fréquence : 2.4GHz
Impédance de 50 Ohm
Polarisation linéaire vertical
Gain de 2 dBi (10Log G)
G : 1,58 ($10^{\text{dBi}/10}$)



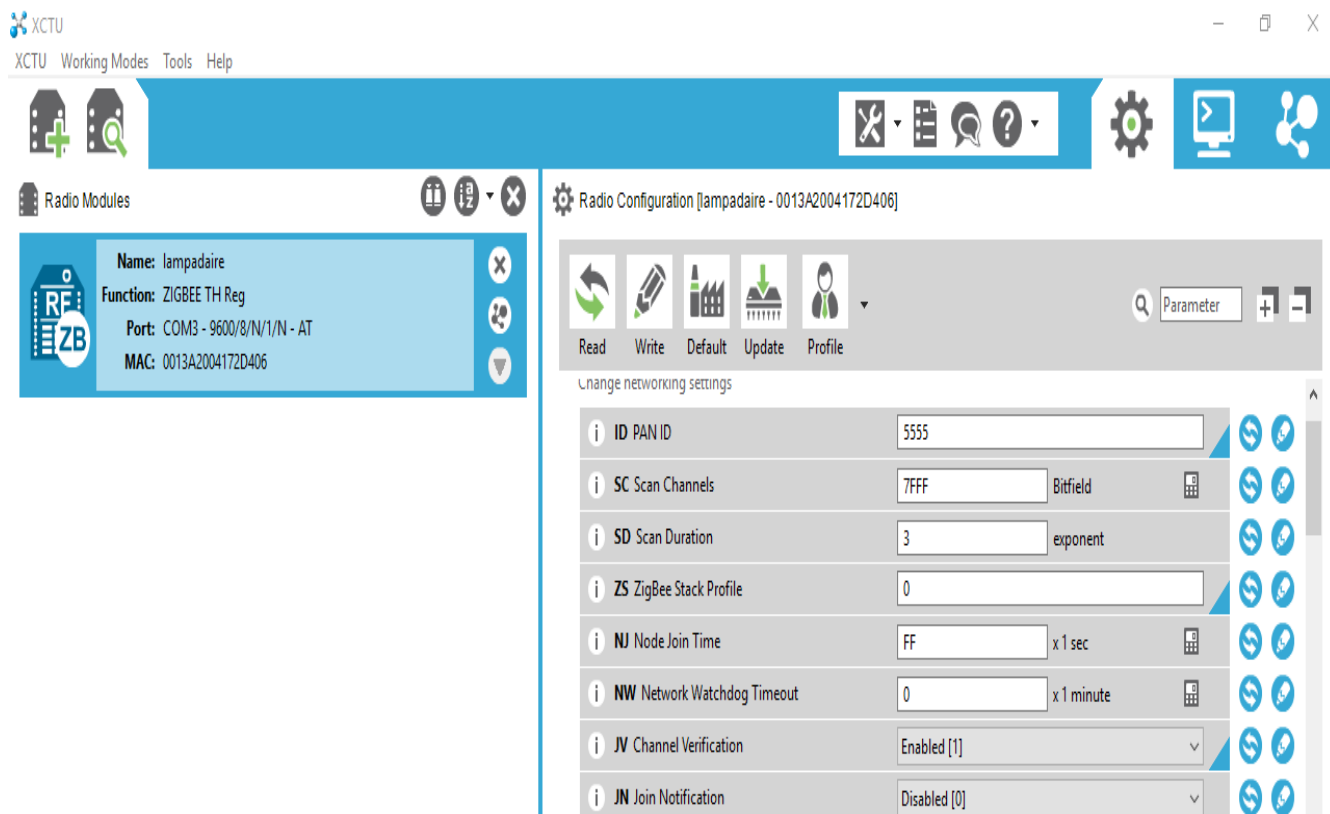
Étant donné que nous utilisons le protocole ZigBee pour la communication, cela peut avoir une zone de conflit avec le wifi et donc un mauvais transport des trames.

Cette antenne est omnidirectionnelle, c'est-à-dire elle envoie les informations dans toutes les directions comme le montre ce plan de rayonnement :





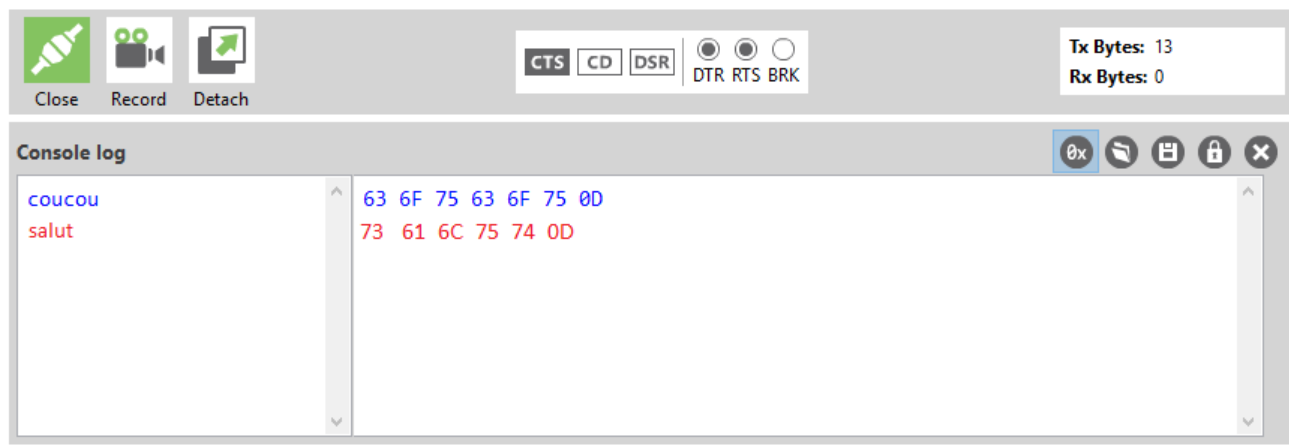
Logiciel XCTU



Le logiciel XCTU permet de configurer les paramètres des modules Xbee et de les tester grâce à un terminal. Le logiciel GTC est inspiré de XCTU.

Les modules qui sont sur le même réseau peuvent communiquer dans le terminal et envoyer des messages.

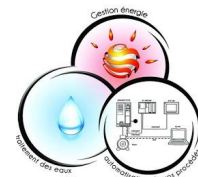
lampadaire - 0013A2004172D406





Voici un tableau qui regroupe les paramètres essentiels à la configuration des modules :

Paramètres	Coordinateur	Routeur1	Routeur2
ID	5555	5555	5555
JV(activer si le peripherique est un lampadaire)	[0]	Enable [1]	Enable [1]
CE(activer si le peripherique est un coordinateur)	Enable [1]	[0]	[0]
DH(premier bits de l'adresse de destination)	0	0	0
DL(dernier bits de l'adresse de destination)	FFFF	0	0
NI (nom du module)	COORD	ROUTEUR1	ROUTEUR2



Activités de mise en œuvre : Itération 1

J'ai créé un logiciel de test où les valeurs sont codées en dur, c'est-à-dire que j'ai inscrit les valeurs basic des modules xbee directement dans le code pour ne pas avoir à aller les récupérer grâce à la liaison séries.

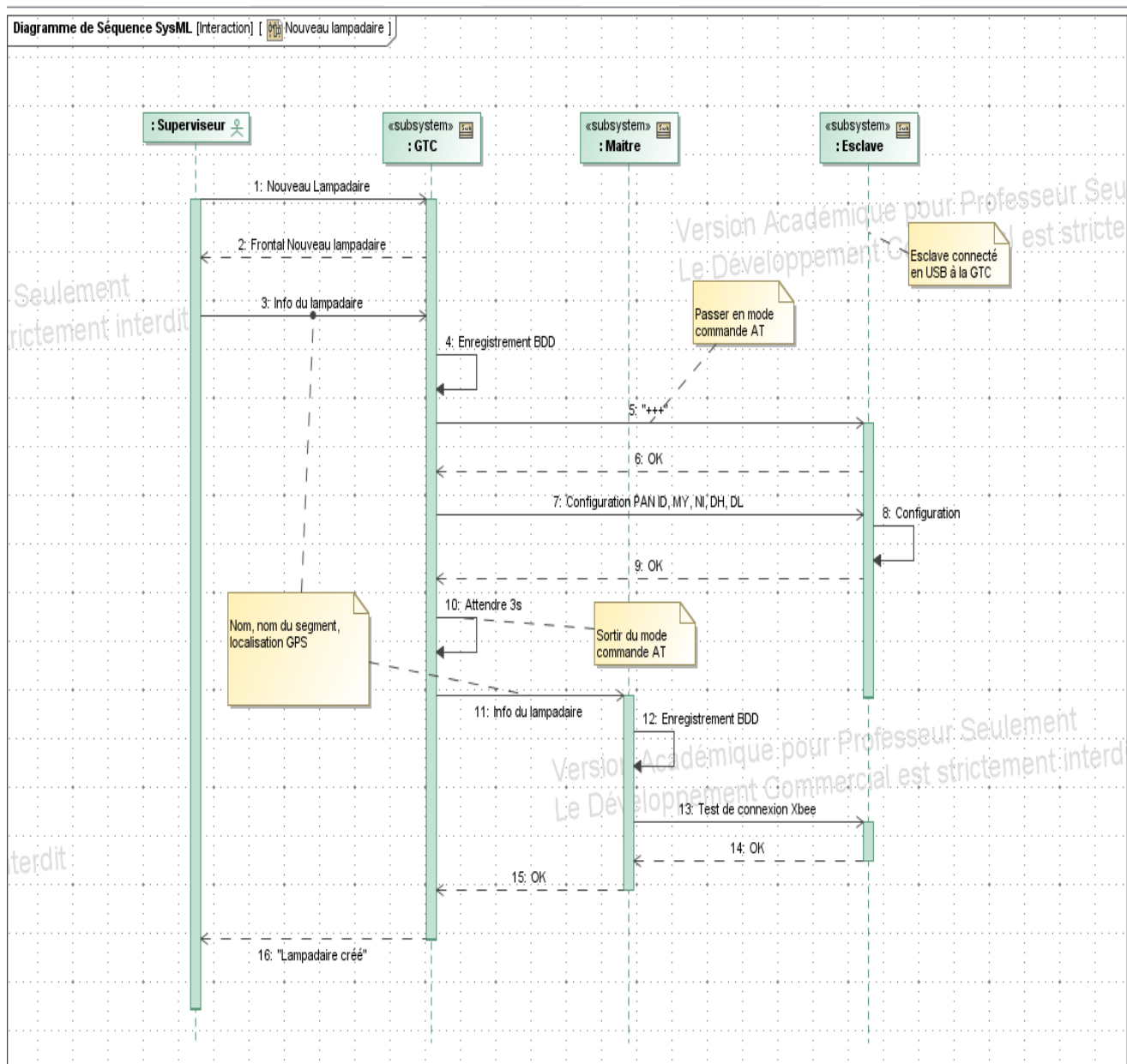
Cela ma permit de tester certaines parties de mon code avant d'avoir fait la partie de liaison séries car celle-ci est plus dur et plus longue.

```
void CIhmLampadaire::on_PbLire_clicked()
{
    ui->lineId->setText("5555");
    ui->BoxJv->setCurrentText("Disabled [0]");
    ui->BoxCe->setCurrentText("Enabled [1]");
    ui->LineDh->setText("0");
    ui->LineDl->setText("ffff");
    ui->LineNi->setText("Maître");
    QMessageBox::information(this,"Lecture","Les paramètres ont été mis à jour");
}
```

L'extrait du code est celui du bouton lire qui permet de charger les données du module xbee qui sont ici coder en dur :

Ancienne mise en page

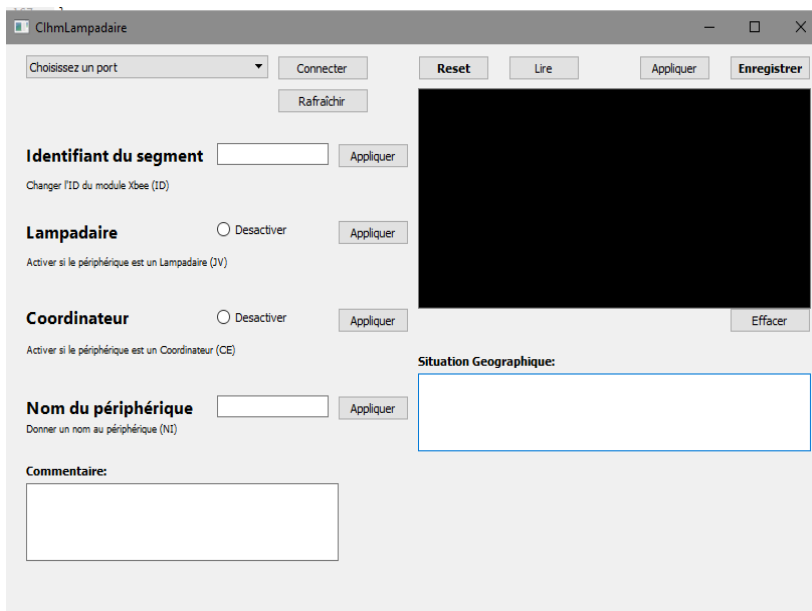
Diagramme séquence interaction



Itération 1

J'ai créé un logiciel sur Qt Creator qui permet de configurer les modules xbee via une liaison série. J'ai utilisé l'outil Qt Designer qui est intégré dans Qt Creator pour faire la mise en page graphique du logiciel grâce à des widgets.

Nouvelle mise en page :



Le logiciel est conçu pour des techniciens qui travailleront sur les lampadaires c'est pour cela qu'il fallait que la mise en page soit compréhensible et simple à utiliser.

```
{
    ui->setupUi(this);
    QList<QSerialPortInfo> listPsi; // liste des ports série existant
    listPsi = QSerialPortInfo::availablePorts(); // récupère les ports série disponibles
    ui->boxPorts->addItem("Choisissez un port");
    for(int i=0 ; i<listPsi.size() ; i++)
    {
        ui->boxPorts->addItem(listPsi.at(i).portName() + "-" + listPsi.at(i).description());
        qDebug() << "Port disponible : " << listPsi.at(i).portName() + "-" << listPsi.at(i).description();
    }
    mPs = new QSerialPort(this);

    mPs->setBaudRate(QSerialPort::Baud9600);
    mPs->setDataBits(QSerialPort::Data8);
    mPs->setParity(QSerialPort::NoParity);
    mPs->setStopBits(QSerialPort::OneStop);
    mPs->setFlowControl(QSerialPort::NoFlowControl);

    connect(mPs, &QSerialPort::readyRead, this, &CIhmLampadaire::readData);
}
```

J'utilise la classe QserialportInfo pour vérifier les ports disponibles, la liste de port est ensuite ajoutée à un menu déroulant.



```
void CIhmLampadaire::on_pbConnecter_clicked()
{
    // ouverture de la liaison data

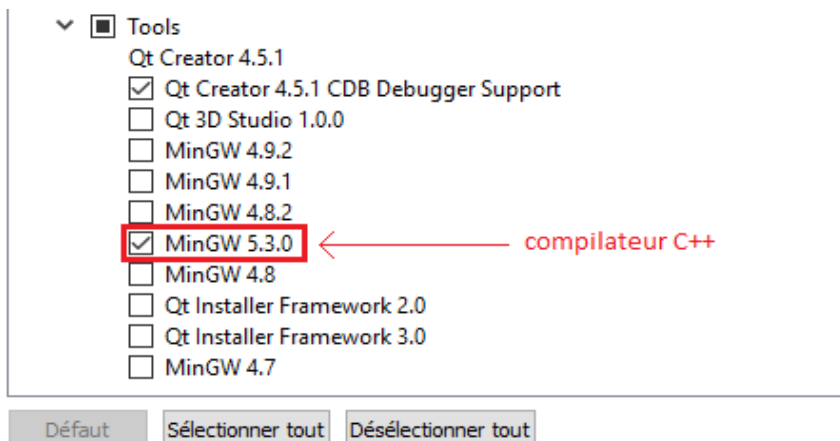
    int selection = this->ui->boxPorts->currentIndex();
    if ( selection != 0 ) {
        QString portname;
        QStringList listport = ui->boxPorts->itemText(selection).split('-');
        portname= listport[0];
        mPs->setPortName(portname);
        if (ui->pbConnecter->text()=="Connecter"){
            if (mPs->open(QIODevice::ReadWrite)){
                ui->txtConsole->append("Connecter au port: "+mPs->portName());
                ui->pbConnecter->setText("Déconnecter");
            } else{
                ui->txtConsole->append("Impossible de se connecter au port: "+mPs->portName());
            }
        }else{
            mPs->close();
            ui->pbConnecter->setText("Connecter");
            ui->txtConsole->append("Déconnecter du port: "+mPs->portName());
        }
    }
}
```

L'extrait du code est celui du bouton connecter qui permet de se connecter au port série sélectionné auparavant. Le code permet aussi de changer l'aspect du bouton en écrivant « Connecter » quand on n'est pas connecté à un port et « Déconnecter » quand on a déjà cliqué sur le bouton.

Problèmes et solutions

Problème : Lors de l'installation de Qt Creator sur Windows, aucun kit installer.

Solution : Il fallait installer un compilateur en cochant l'option dans l'installateur de Qt Creator.



Problème : Le logiciel ne détecte pas les module xbee.

Solution : La classe qserialport n'était pas bien instancier.

Problème : Impossible d'enregistrer la configuration envoyée sur les modules via l'application.

Solution : Il fallait utiliser des slots pour que la configuration prenne en compte les configurations.

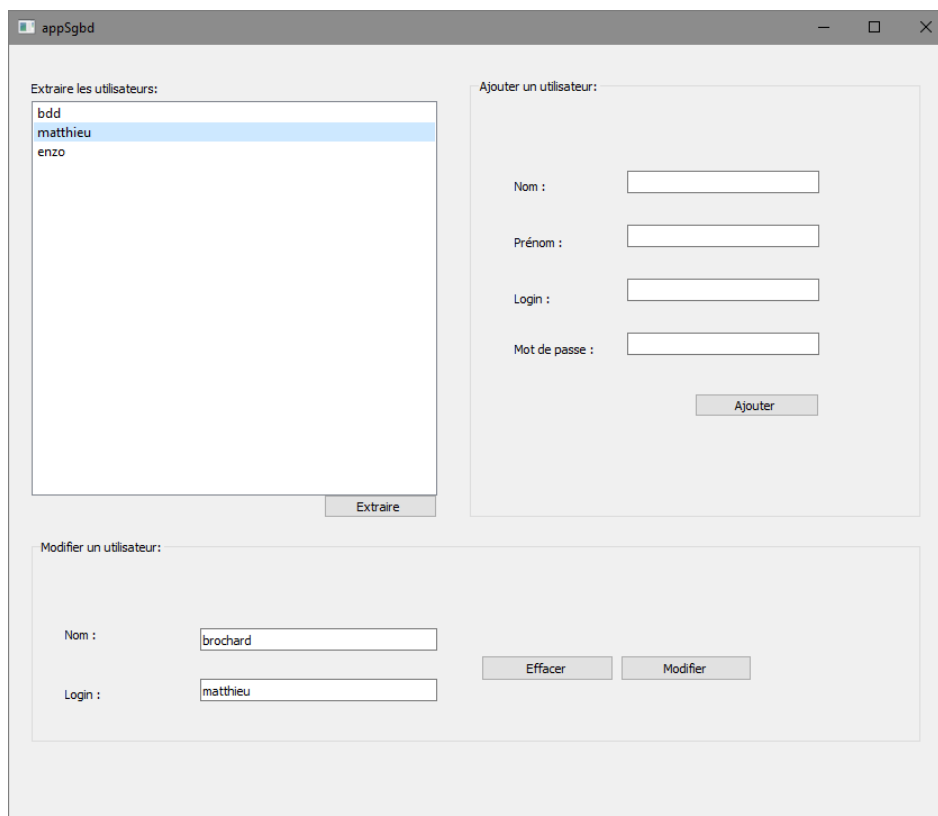
Activités de mise en œuvre : Itération 2

Pour bien comprendre le fonctionnement des Base de donnée j'ai suivi un cours de Mr. Antoine sur chamilo qui m'a fait crée une BDD de test appelée bddFormation source :(<https://lc.cx/mNNu>)

Dans cette BDD j'ai créé une table user ou j'ai rentré un nom, prénom, login et mot de passe (crypter automatiquement en SHA1 dans php my admin) pour trois utilisateurs : bdd, enzo et matthieu.

					id	nom	prenom	login	mdp
☐	Éditer	Copier	Supprimer		1	bdd	bdd	bdd	fe55c29e928524cbd10fad9c4f65ba82f67d73f9
☐	Éditer	Copier	Supprimer		2	brochard	matthieu	matthieu	00d70c561892a94980befd12a400e26aeb4b8599
☐	Éditer	Copier	Supprimer		4	garagnoli	enzo	enzo	2736fab291f04e69b62d490c3c09361f5b82461a

J'ai ensuite fait une IHM simple qui se connecte à la BDD automatiquement et qui permet d'ajouter, modifier ou supprimer un utilisateur. Pour l'utiliser il suffit de cliquer sur extraire pour extraire tous les utilisateurs de la BDD, choisir celui que l'on veut modifier et ensuite changer son nom ou login (on ne peut pas changer le mot de passe). Pour ajouter un utilisateur on doit juste rentrer ces informations dans le cadre à droite.

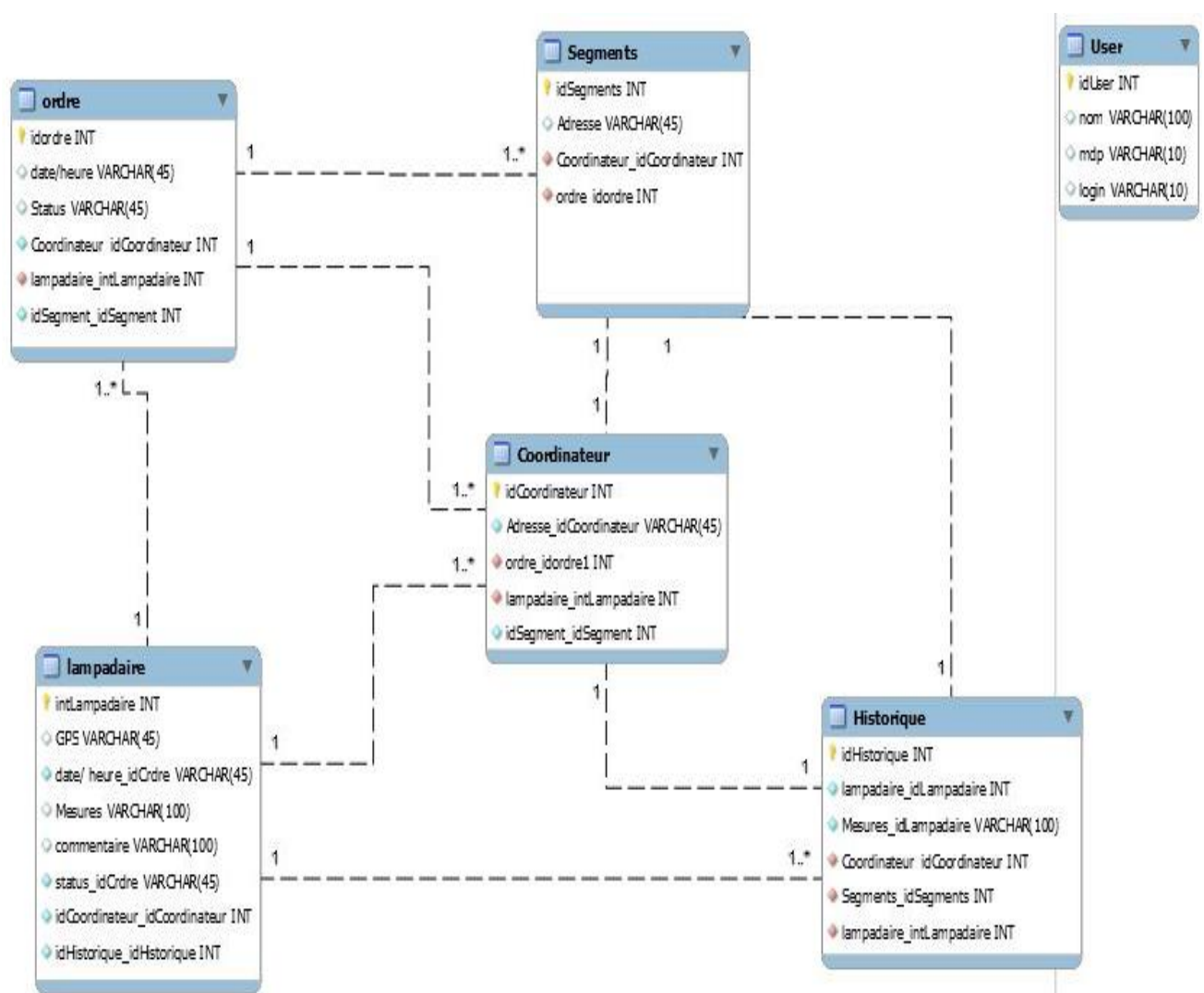
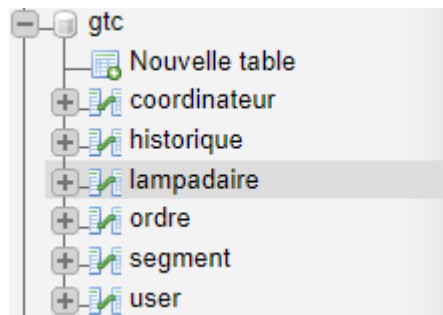


The screenshot shows a web application window titled 'appSgbd'. It contains three main sections for user management:

- Extraire les utilisateurs:** A list box containing 'bdd', 'matthieu', and 'enzo'. The 'matthieu' entry is selected. Below the list is an 'Extraire' button.
- Ajouter un utilisateur:** A form with four input fields labeled 'Nom', 'Prénom', 'Login', and 'Mot de passe'. Below these fields is an 'Ajouter' button.
- Modifier un utilisateur:** A form with two input fields labeled 'Nom' and 'Login'. The 'Nom' field contains 'brochard' and the 'Login' field contains 'matthieu'. Below these fields are 'Effacer' and 'Modifier' buttons.

Itération 2

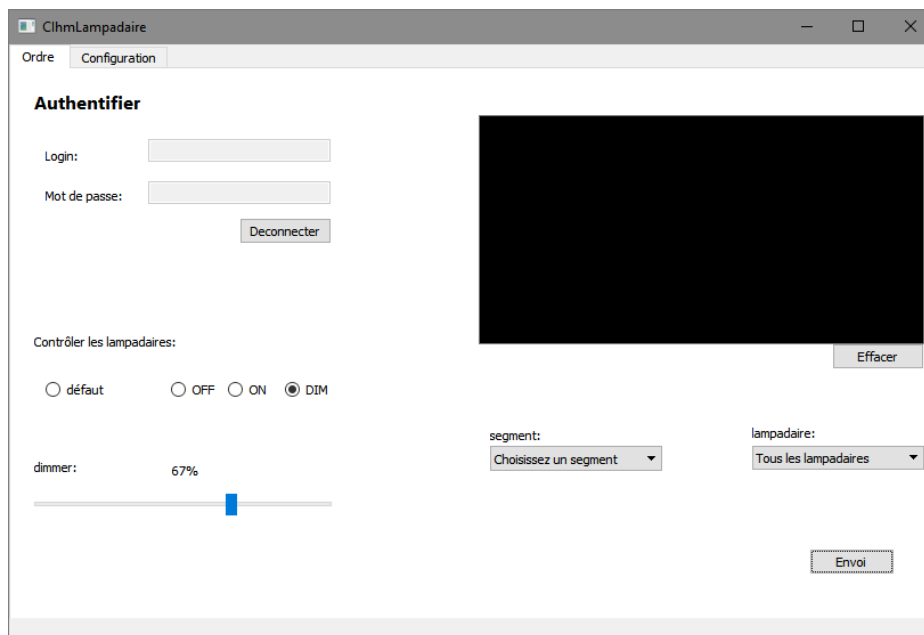
J'ai créé une base de donnée appeler gtc dans MySQL grâce au site phpMyAdmin où j'ai créé les tables coordinateur, historique, lampadaire, ordre, segment et user.



J'ai ensuite ajouté un onglet au logiciel qui permet de s'authentifier dans la BDD. Si l'utilisateur du logiciel ne s'authentifie pas il n'a pas accès au onglets de configuration et ne peut pas envoyer d'ordre. Il y a trois utilisateur différents :

- Le Super admin qui a le droit de tous modifier.
- Le technicien qui a le droit de configurer les modules xbee mais ne peut pas envoyer d'ordres.
- L'admin qui a le droit d'envoyer des ordres mais ne peut pas reconfigurer les modules xbee.

Onglet ordre :



L'IHM est simple à utiliser : il faut d'abord entrer son login et son mot de passe et cela débloquent alors les onglets. On peut maintenant utiliser l'onglet modifier pour changer les paramètres des modules xbee ou rester sur l'onglet ordre (selon les privilèges de l'utilisateur connecté).

Si on reste sur cet onglet on peut contrôler les lampadaires en leur envoyant des ordres de cette façon : on choisit le segment et le ou les lampadaires que l'on veut. Une fois sélectionner on peut demander au lampadaire de soit s'éteindre grâce au bouton OFF soit de s'allumer avec le bouton ON ou changer la luminosité grâce au bouton DIM qui activera le slider en dessous des boutons. Le bouton défaut remet le lampadaire en mode automatique pour qu'il suive la plage horaire qui lui est attribuée.

L'extrait du code est celui du bouton connecter de l'onglet ordre. On peut voir qu'avant de s'authentifier les boutons et les autres onglets sont désactivés mais ils sont tous activés une que fois la connexion à la bdd est réussie.

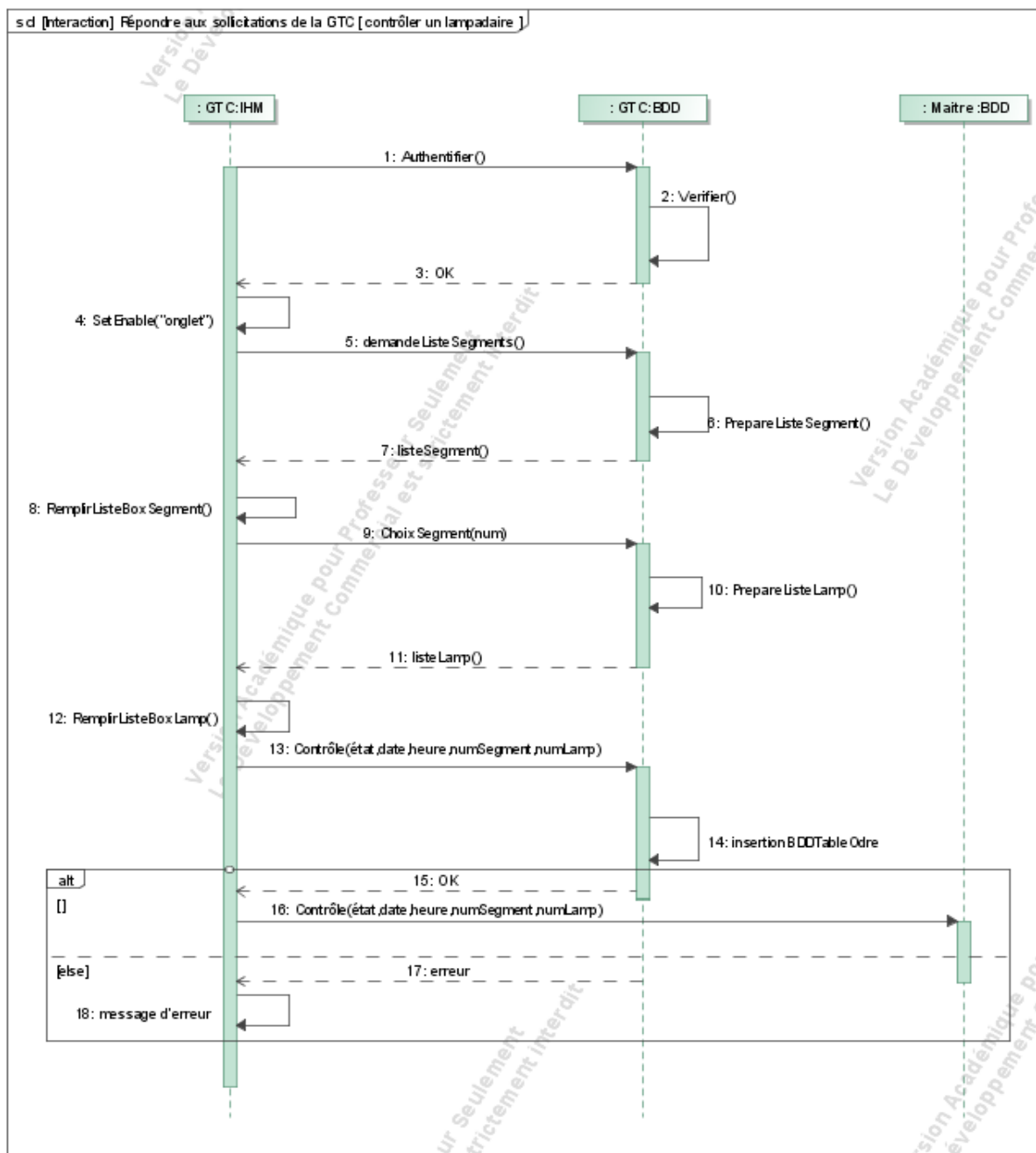
```
//authentification
void CIhmLampadaire::on_pbConnecterOrdre_clicked()
{
    QSqlError err;

    bdd.setUserName(ui->lineLogin->text());
    bdd.setPassword(ui->lineMdp->text());
    bool ok = bdd.open();
    if (!ok) {
        err = bdd.lastError();
        ui->pbConnecterOrdre->setText("Connecter");
        ui->tab->setDisabled(true);
        ui->lineLogin->setEnabled(true);
        ui->lineMdp->setEnabled(true);
        ui->sliderDim->setEnabled(false);
        ui->radDIM->setEnabled(false);
        ui->radOFF->setEnabled(false);
        ui->radON->setEnabled(false);
        ui->radDef->setEnabled(false);
        bdd.close();
    }
    else{
        qDebug() << "Connexion à la BDD " << ok;
        ui->tab->setEnabled(true);
        ui->pbConnecterOrdre->setText("Deconnecter");
        ui->lineLogin->setDisabled(true);
        ui->lineMdp->setDisabled(true);
        ui->lineLogin->clear();
        ui->lineMdp->clear();
        ui->radDIM->setEnabled(true);
        ui->radOFF->setEnabled(true);
        ui->radON->setEnabled(true);
        ui->radDef->setEnabled(true);
        ui->pbEffacerOrdre->setEnabled(true);
        ui->pbEnvoi->setEnabled(true);
        ui->boxSegments->setEnabled(true);
    }
}
```

setEnabled(false) ;
les widgets graphique
reste désactivé.

setEnabled(true) ;
on active les widget
graphique.

Diagramme séquence contrôler un lampadaire :



Cet extrait de code montre comment les segments sont sélectionnés dans la BDD et ensuite ajoutés dans la liste de l'IHM. On crée une liste qstring appelée segments et on exécute la commande sql :

« SELECT * FROM segment ORDER BY Adresse » cette commande permet de sélectionner toute les adresses qui se trouve dans la table segment. Après les avoir sélectionnés elles sont enregistrées dans la liste qstring et ensuite ajouter dans l'IHM.

```
// liste des segments
QStringList segments;
QSqlQuery q(bdd);
q.exec("SELECT * FROM segment ORDER BY Adresse");
while (q.next()) {
    segments << q.value("Adresse").toString();
}
ui->boxSegments->addItem("Choisissez un segment");
for(int i=0 ; i<segments.size() ; i++)
{
    ui->boxSegments->addItem(segments.at(i));
}
```



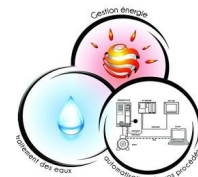
Problèmes et solutions

Problème : Les tables et colonnes de la BDD de la GTC et dans la BDD des coordinateurs n'étaient pas les mêmes et ne pouvait pas communiquer.

Solution : Je me suis concerté avec Enzo qui s'occupe de la BDD des coordinateurs et nous avons choisi les mêmes.

Problème : Le driver QMYSQL n'était pas installé et m'empêché de me connecter à la BDD.

Solution : La version de Qt que j'avais installé n'était pas la dernière version qui intègre le driver j'ai donc réinstallé la dernière version.



Journal de bord personnel

18/01/2018: création du drive du projet, lecture du dossier de présentation début du Projet

19/01/2018 : Finalisation de Project et configuration des module xbee via XCTU+problème d'installation Qt sur Windows pendant 4h.pas de kit installé

19/01/2018 : Création de l'IHM partie configuration (avec boutons qui marchent pour faire une présentation basique sans récupérer les paramètres de la carte).

26/01/2018 : Début du code des port série pour connecter le module xbee (qserialport utiliser mais problème de détection du module).

14/02/2018 : Envoie des données sur le module xbee mais impossible d'enregistrer la configuration.

15/03/2018 : code des port série : ajout de slots dans le code pour résoudre le problème d'enregistrement de configuration

04/04/2018: exercices sur les BDD pour apprendre à coder un logiciel pour configurer la BDD

05/04/2018 : création de la BDD gtc après avoir consulter la BDD des coordinateurs

06/04/2018 : Début du code de l'IHM sur Qt Creator

11/04/2018 : Intégration de la BDD dans le logiciel

18/04/18 : Visite de l'entreprise API et présentation de l'état d'avancement du projet

19/04/2018 : Début du code pour envoyer des ordres sur la BDD

Partie : Garagnoli Enzo

Ma partie de ce projet concerne les lampadaires Maitres :

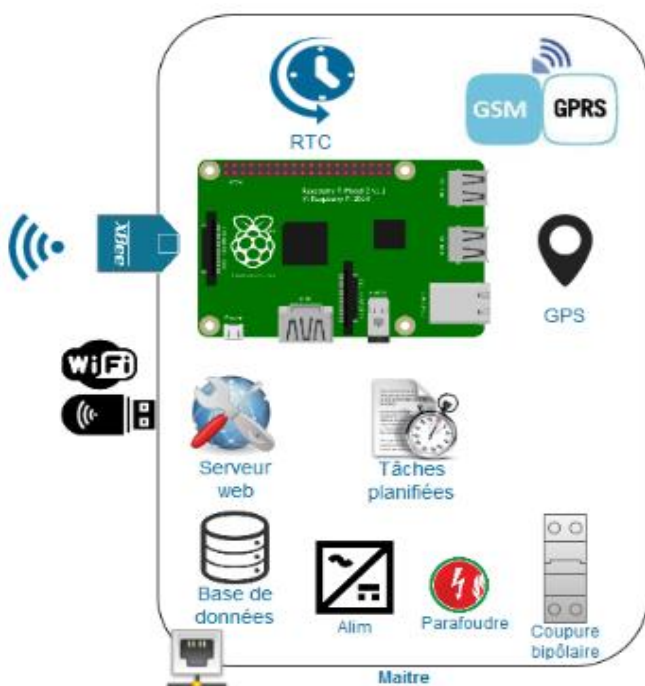
Je dois mettre en place le réseau Ethernet/IP afin d'assurer la communication entre le maître (Raspberry Pi) et la GTC.

Je dois aussi répondre à plusieurs cas d'utilisation comme : "Communiquer avec la GTC", "Sauvegarder l'état des esclaves », Communiquer avec les esclaves", et "Répondre aux sollicitation de la GTC" avec tous les cas d'utilisation inclus lorsque la sollicitation s'adresse au lampadaire maître (fonctionnalité esclave).

Pour répondre à ces cas, ma mission est de crée et faire fonctionner une Base De Données (BDD) avec la création de tables qui va permettre de ranger les différents paramètres par catégorie. Pour le XBee, une autre mission est de communiquer au travers du XBee Maître et expédier la trame qui contient les ordres vers les esclaves

Ensuite, je dois créent 2 différentes Bibliothèques :

- une pour accéder à la base de données sur une Raspberry Pi
- une autre pour contrôle d'un lampadaire (via XBee)



Ma partie du projet de façon imagé

Diagramme de Gantt

4 Tâches Enzo	34,71 jours?	Ven 19/01/1	Ven 15/06/18	
BDD	30 hr	Ven 19/01/1	Jeu 01/02/18	7
création des tables de la BDD	30 hr	Ven 02/02/18	Jeu 15/03/18	18
configuration xbee maître	20 hr	Jeu 15/03/18	Ven 23/03/18	19
logiciel maître	70 hr	Ven 23/03/18	Jeu 10/05/18	20
creation de la bibliotheque c++ acces BDD	30 hr	Jeu 10/05/18	Mer 23/05/18	21
creation de la bibliotheque c++ contrôle lampadaire	30 hr	Mer 23/05/18	Mer 06/06/18	22
IR2 IT1	60 hr	Ven 19/01/1	Lun 12/03/18	
IR2 IT2	118 hr?	Mar 13/03/18	Jeu 10/05/18	24
IR2 IT3	89 hr	Ven 11/05/18	Ven 15/06/18	25

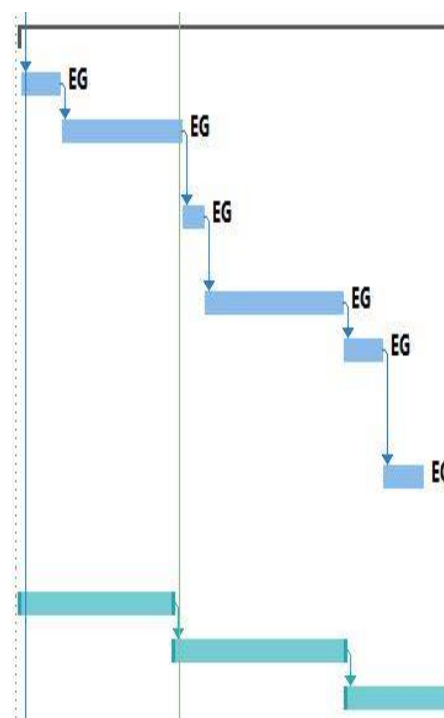


Diagramme de Gantt final

4 Tâches Enzo	34,71 jours?	Ven 19/01/1	Ven 15/06/18	
création des tables de la BDD	60 hr	Ven 19/01/18	Jeu 15/03/18	7
configuration de la raspberry (coordinateur)	20 hr	Jeu 15/03/18	Ven 23/03/18	18
logiciel maître	45 hr	Ven 23/03/18	Jeu 12/04/18	19
creation de la bibliotheque c++ acces BDD	45 hr	Jeu 12/04/18	Jeu 17/05/18	20
creation de la bibliotheque c++ contrôle lampadaire	45 hr	Jeu 17/05/18	Jeu 07/06/18	21
IR2 IT1	60 hr	Ven 19/01/1	Lun 12/03/18	
IR2 IT2	118 hr	Mar 13/03/18	Jeu 10/05/18	23
IR2 IT3	89 hr	Ven 11/05/18	Ven 15/06/18	24

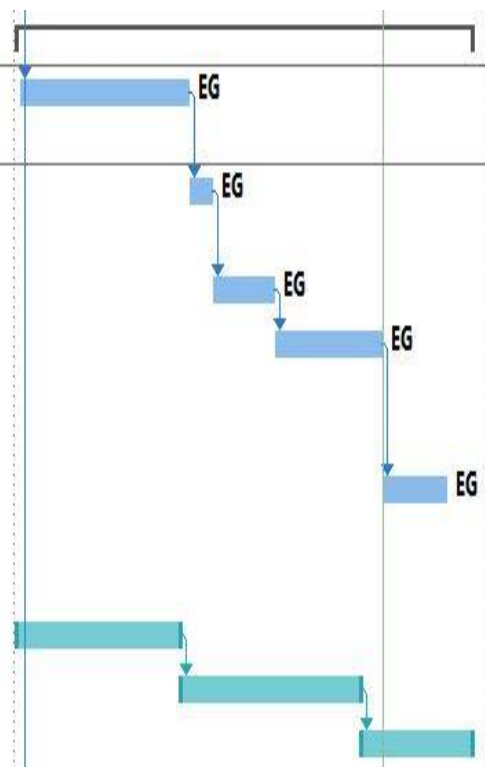


Diagramme des cas d'utilisation

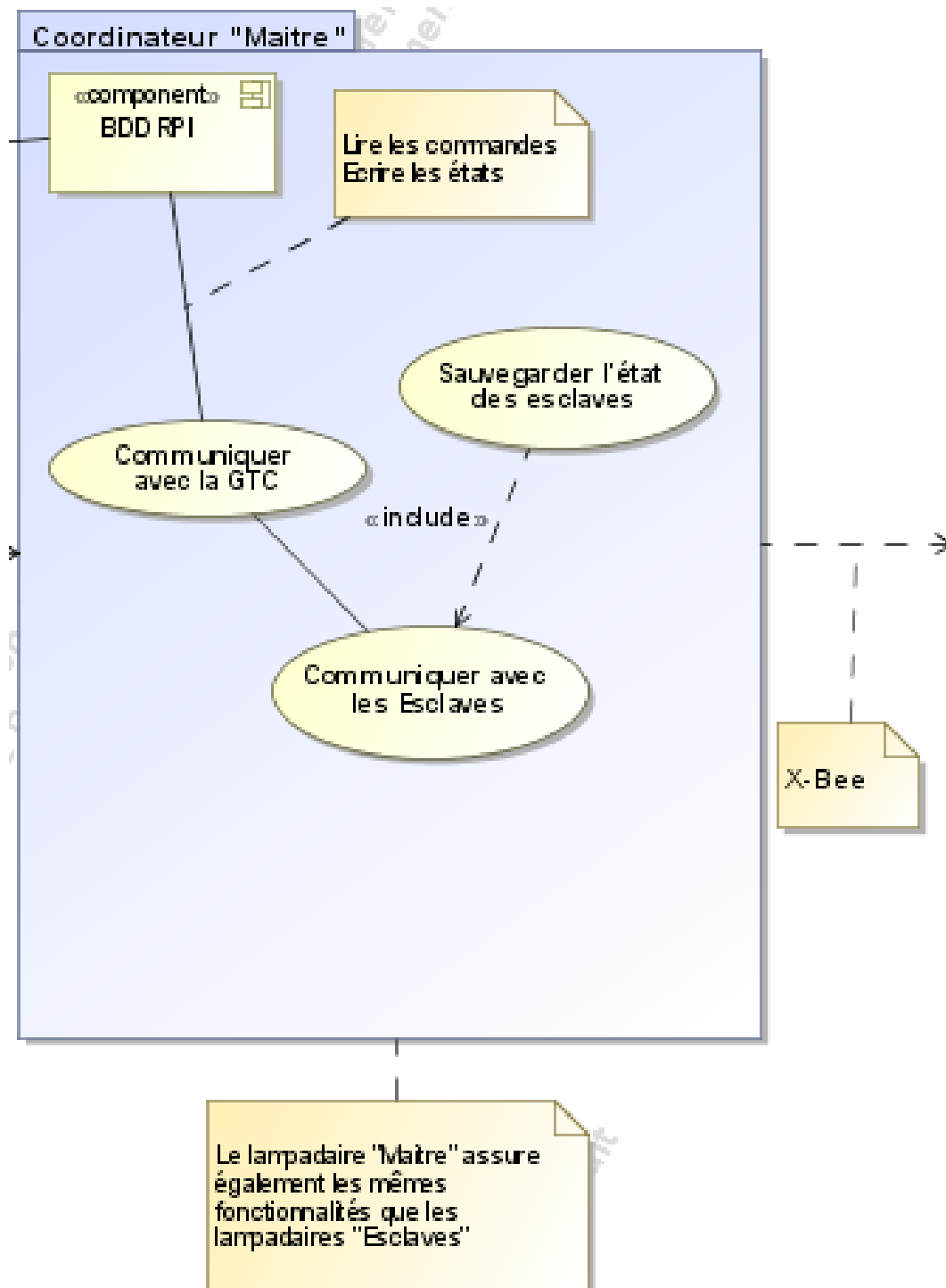


Diagramme de bloc structure

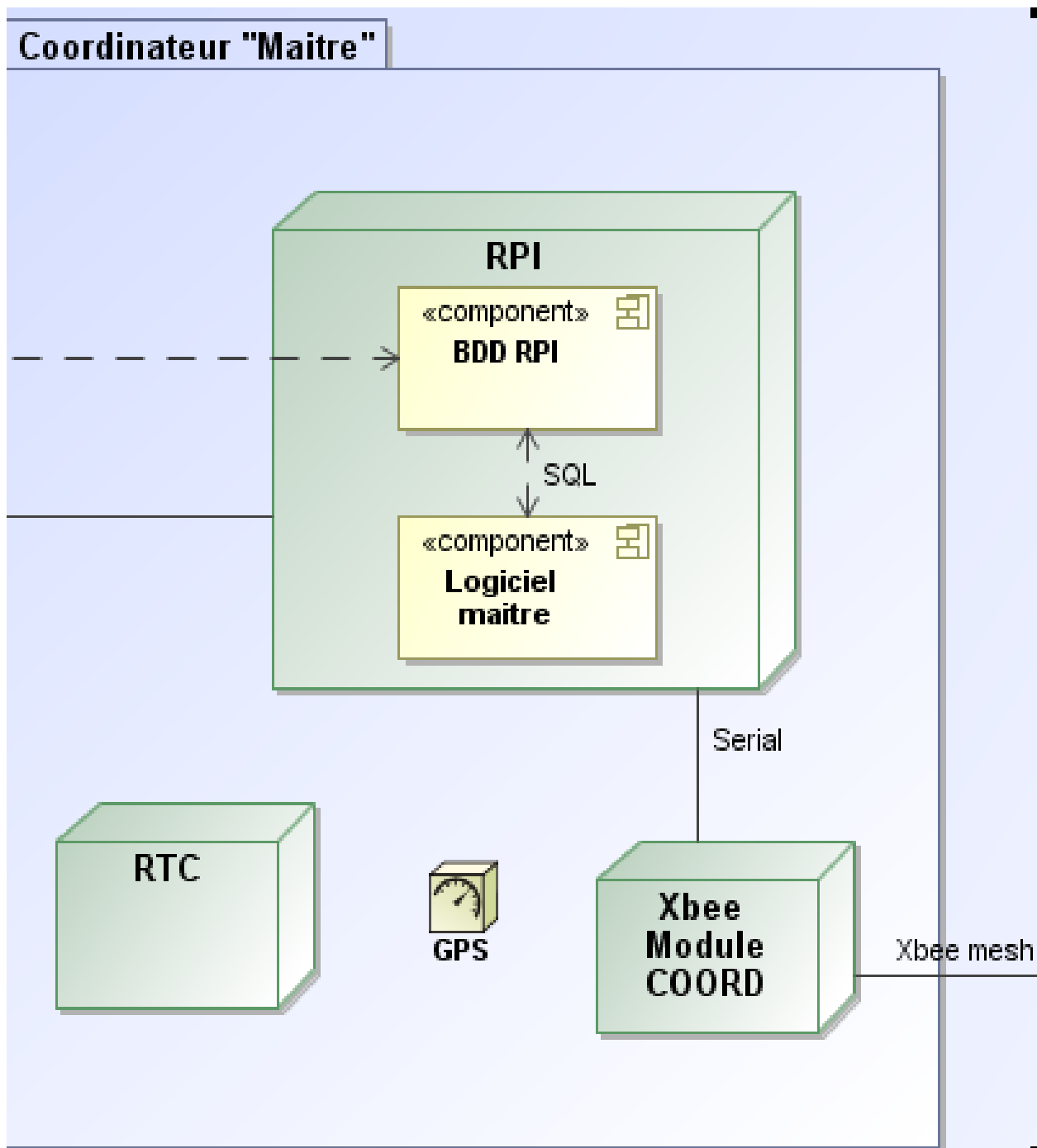
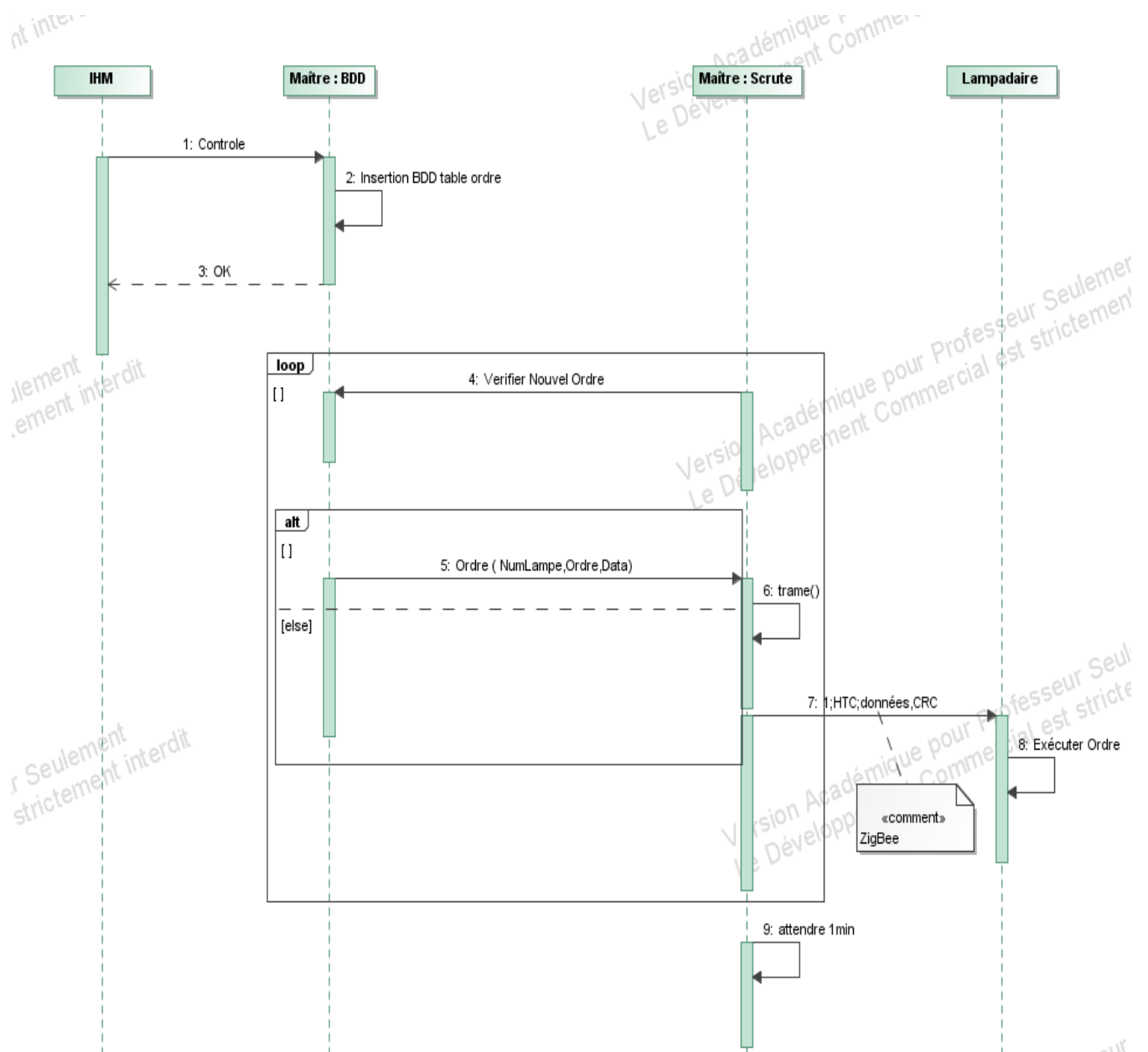
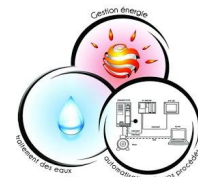


Diagramme de séquence pour l'envoi d'ordre:



Plan des itérations

Taches	Itération	Pourquoi?
Création des tables et de la base de données	1	Etape facile mais longue qui va permettre aux autres membres du groupe de pouvoir l'utiliser.
Communiquer au travers du XBee Maître	3	Expédier la trame qui contient les ordres vers les esclaves.
Logiciel maître	2	Logiciel similaire à celui des esclaves
Création de la Bibliothèque C++ d'accès à la BDD sur RPI	2	Doit avoir les paramètres et table de la BDD pour la création de la bibliothèque
Création de la Bibliothèque C++ de contrôle d'un lampadaire (XBee)	3	Doit avoir les paramètres du module XBee maître pour la création de la bibliothèque



Cas d'utilisation

Dans ma partie de ce projet, on a plusieurs cas d'utilisation qui sont accompagnés de scénarii et même de scénarii alternatifs. Les voici :

Cas d'utilisation : Répondre aux sollicitation de la GTC

Scénario Nominal : L'esclave reçoit une commande de ...

Scénario alternatif : Esclave déconnecté. Exécute la dernière commande en mémoire non volatile

Dans ce cas d'utilisation, notre objectif est de faire envoyer un ordre a un esclave pour qu'il l'exécute. Si cette commande n'arrive pas à être exécutée, alors l'esclave va exécuter la dernière commande qu'il aura reçu et qui est stocké dans sa mémoire non volatile.

Cas d'utilisation : Communiquer avec les esclaves

Cas d'utilisation : Sauvegarder l'état des esclaves

Ce cas est explicite, sauvegarder l'état d'un esclave dans une période donnée avec les paramètres comme la plage horaire ou l'intensité lumineuse, ainsi que sa consommation instantanée.

Cas d'utilisation : Communiquer avec la GTC

Scénarii : Lire les commandes et écrire les états

Ce cas est exclusif au lampadaire maitre, qui va envoyer des informations aux esclaves, lire et écrire les états des réverbères, les enregistrer dans sa BDD, puis la GTC viendra les copier dans la sienne.

Activités de mise en œuvre

Concernant la partie de base de données, j'ai essayé de faire un programme qui permette à Qt de se connecter directement à la base de données.

Malheureusement je n'ai pas réussi à faire un programme qui le permette faute de temps néanmoins, j'ai réussi à trouver un exemple de code :

```

1  #include <QApplication>
2  #include <QtGui>
3  #include <QtSql>
4
5  int main(int argc, char *argv[])
6  {
7      QApplication app(argc, argv);
8
9
10     QWidget *fenetre = new QWidget;
11     QLabel *texte = new QLabel("<center>Testeur de connexion à une BDD distante.</center>");
12     texte->setText(texte->text());
13
14     QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");
15     db.setHostName("127.0.0.1");
16     db.setUserName("root");
17     db.setPassword("");
18     db.setDatabaseName("test");
19     db.open();
20
21     if (!db.open())
22     {
23         QMessageBox::critical(fenetre, "Erreur", "La connexion à la BDD a échoué !!!");
24         qApp->quit();
25     }
26     else
27     {
28         QMessageBox::information(fenetre, "OK", "La connexion à la BDD a été établie");
29         qApp->quit();
30     }
31
32     fenetre->setWindowTitle("Test de connexion à la BDD");
33     fenetre->setFixedSize(291, 19);
34     fenetre->show();
35
36     return app.exec();
37 }

```

Ici, le problème est qu'il utilise la librairie « QtSql » qui utilise SQLite.

Pour notre utilisation, nous allons besoin de MySQL et non SQLite.

Ce que nous avons à faire est juste supprimer la ligne 'include « QtSql »'

Dans la forme, les deux codes sont similaire :

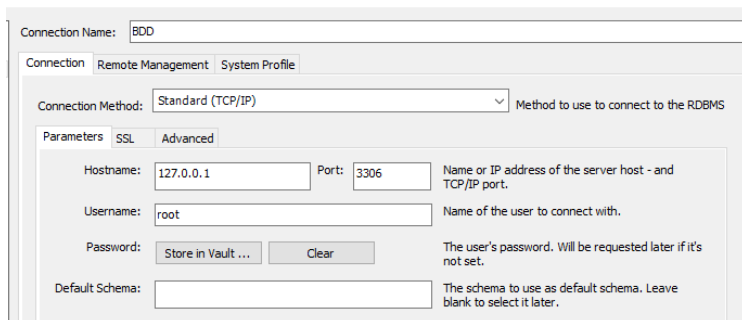
- L'HostName (adresse)
- L'identifiant
- Le mot de passe
- Le nom de la base de données

Itération 1

Ma première itération consiste à créer les tables et la base de données.

Pour cela on va utiliser le logiciel MySQL WorkBench.

Dans un premier temps, nous allons devoir créer une connexion à MySQL, cela va se faire grâce à l'interface proposée :



Ici on va pouvoir configurer la méthode de connexion, le nom, le port utilisée (ici 3306 pour correspondre au port du logiciel WAMPServer), un identifiant, mot de passe

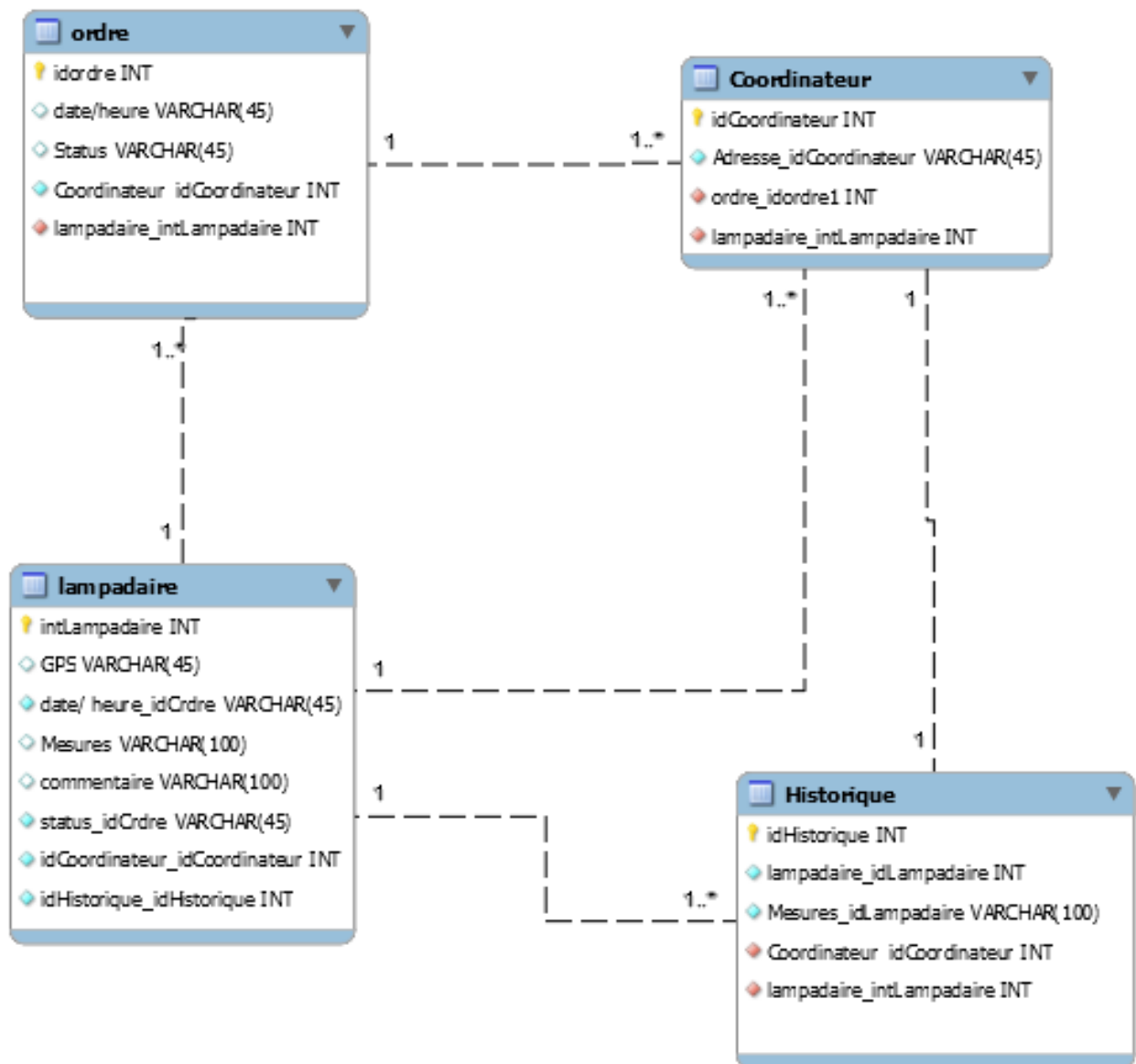
Une fois connecté, on va pouvoir commencer à créer les tables.

Dans la barre des tâches, on vient cliquer sur « file », « new model », puis arriver sur un nouveau menu est on sélectionne « add diagram ».

Ensuite lorsque nous avons créé toutes les tables nécessaires, les relations vont pouvoir être mises grâce à l'option dans le rectangle vert, avec les relations 1-1, 1-*, etc.

Ensuite, nous devons transformer cela en base de données pour cela on va dans le menu « database », « forward Engineer », puis « next » jusqu'à « Review SQL Script ». On copie le contenu du code SQL sur Notepad. On enregistre en .sql.zip (cela permet l'importation dans PhpMyAdmin).

Voici le résultat final :

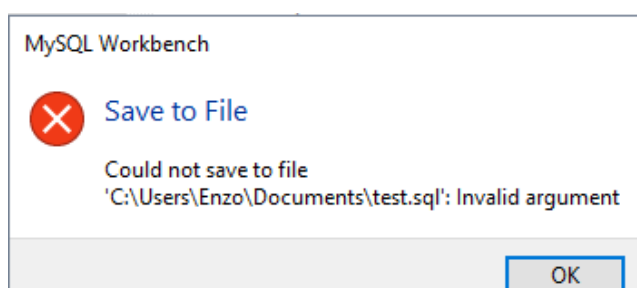


Problèmes et solution

Durant cette partie, plusieurs problèmes ont été rencontrés notamment au niveau du transfert du format MDL à la plateforme d'administration PhpMyAdmin. En effet, en fouillant dans plusieurs forums, j'ai trouvé que plusieurs personnes avaient ce problème.

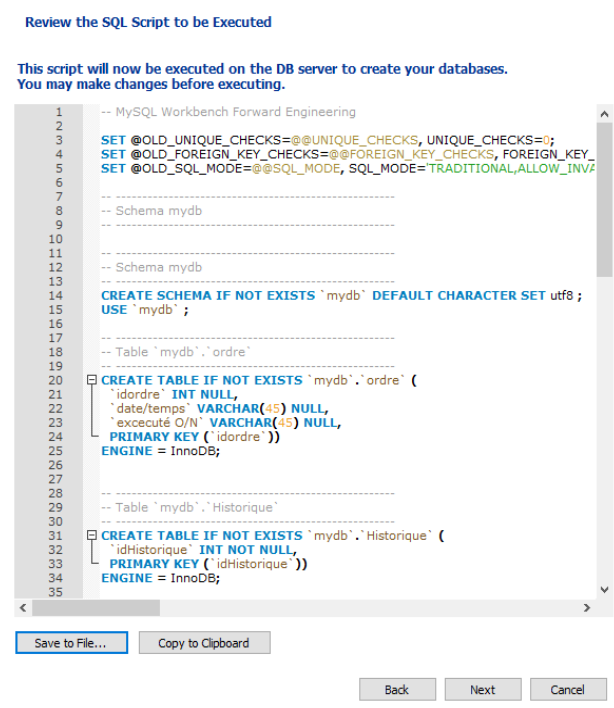
Explication du problème :

Lorsque nous voulions sauvegarder le fichier en .sql, cette erreur s'affiche :



Donc à cause de cette erreur, aucun moyen de transférer le MDL sur PhpMyAdmin.

Solution :



Dans ce menu, l'option « copy to Clipboard » va permettre de copier le contenu du code SQL. Puis nous le transférons sur un éditeur de texte telle que Notepad.

Puis nous faisons un « enregistrer sous... » et l'enregistrons en.sql.zip pour être reconnu par l'administrateur de données.

Un autre problème est venu pour l'installation d'un serveur WAMP (acronymes de « Windows », « apache », « MySQL », « PHP »). En effet j'ai dû passer par plusieurs logiciels pour arriver à trouver le bon, qui sois facile, accessible, et préconfigurer,

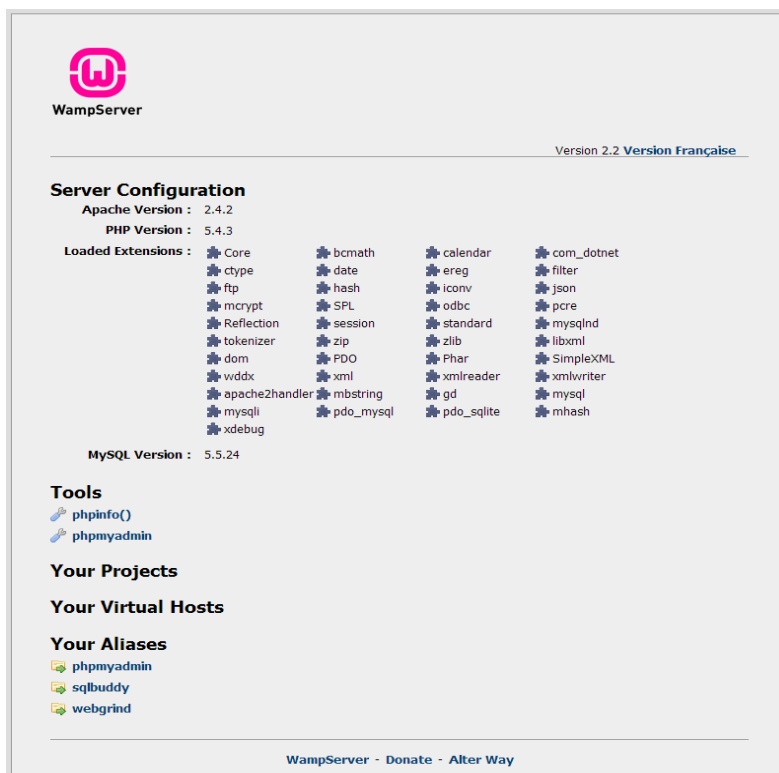
Solution :

Le logiciel qui rassemble les conditions que je voulais est le logiciel WampServer dans sa version 3.0.6

Les avantages :



Cette interface est simple d'utilisation avec les paramètres des différents services, mais aussi des boutons pour stopper, démarrer ou redémarrer tous les services en cas de bug ou autre problèmes.



La page localhost est facile d'utilisation elle aussi avec notamment des liens vers les différents services

Itération 2

Situation partie personnelle

Dans cette 2^{ème} partie, je n'ai pas fait exactement les itérations dans l'ordre établi c'est-à-dire le logiciel maître ainsi que le XBee Maître, ma raison est que je me suis lancé dans la création d'une base de données donc pour prouver que cela fonctionne, je me suis lancée dans la création de l'application.

Activités de mise en œuvre

Ici nous allons voir toute la construction du code pour la création de l'application de base de données

Explication application pour la BDD :

(Attention le code présenter ici est un test, ceci ne reflète en rien le résultats final)

Le .h :

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QSqlDatabase>
#include <QSqlError>
#include <QSqlQuery>
#include <QStringListModel>
#include <QDebug>
#include <QWidget>
#include <QComboBox>

namespace Ui {
class MainWindow;
}
class QSqlTableModel;
class QStringListModel;

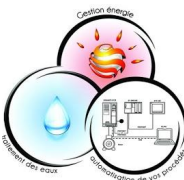
class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_pbExtraire_clicked();
    void on_pbExtraire_2_clicked();
    void on_comboBox_clicked();
    void on_comboBox_3_clicked();

private:
    Ui::MainWindow *ui;
    QSqlDatabase db;
    QSqlTableModel *model;
    QStringListModel *typeModel;
};

#endif
```



Ici nous avons toutes les bibliothèques nécessaires pour la connexion a une base de données avec « QSql ».

Ensuite nous avons les différents attributs qui sont majoritairement en privée pour qu'elle ne soit pas accessible au public ; seul l'interface est en public.

Nous avons 2 méthodes :

- **On_pbExtraire_clicked** (extraire les information d'une table)
- **On_comboBox_clicked** (va permettre l'affichage d'un menu déroulant)

Le fichier .cpp :

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QSqlTableModel>
#include <QtWidgets>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    QSqlError err;

    // init
    ui->setupUi(this);
    ui->tableView_3->setVisible(false);
    model=NULL;

    // connexion à la bdd
    db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName("localhost");
    db.setDatabaseName("test");
    db.setUserName("GTC");
    db.setPassword("admin");
    bool ok = db.open();
    if (!ok) {
        err = db.lastError();
        qDebug() << err.text();
    } else
        qDebug() << "Connexion à la BDD " << ok;
    // click dans zone de liste
    connect(ui->tableView, SIGNAL(clicked(QModelIndex)), this, SLOT(onSelectLineList(QModelIndex)));
}
```

Ici nous avons le début du .cpp avec la connexion à la BDD avec ces options:

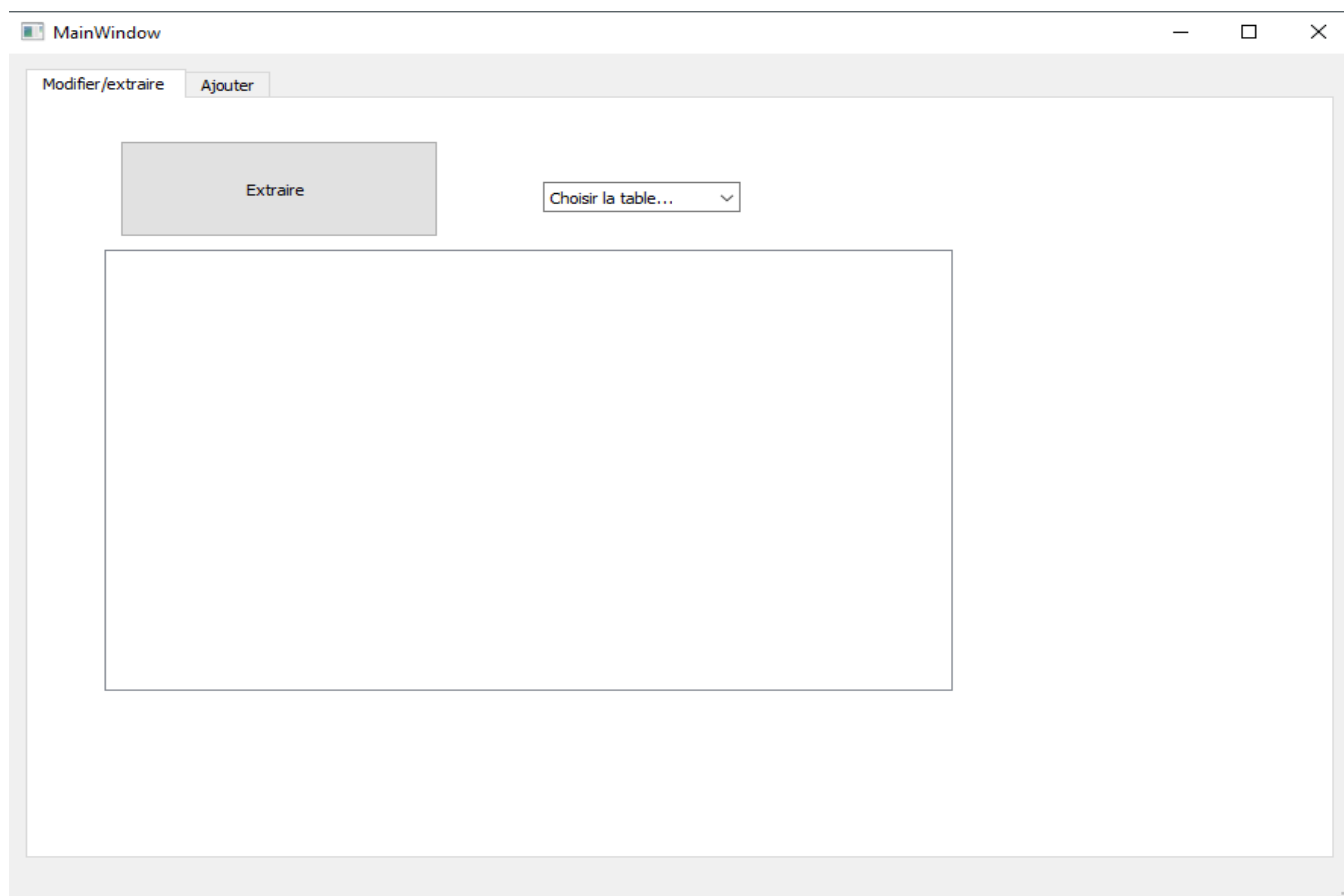
- Host Name (en local avec localhost ou bien une adresse IP de type 192.x.x.x)
- DatabaseName (le nom de la base données)
- UserName (nom d'utilisateur pour la connexion)
- Password (mot de passe pour la connexion)

Ensuite nous avons les paramètres pour les attributs privés avec l'attribut **on_pbExtraire_clicked** avec des commandes SQL pour aller chercher les informations et par la suite les mettre à jour dans la base de données.

```
void MainWindow::on_pbExtraire_clicked()
{
    QSqlQuery q(db);
    model = new QSqlTableModel (this);
    model->setTable("nom");
    model->select();
    ui->tableView->setModel(model);
    q.prepare("INSERT INTO nom VALUES (NULL, :nom2, :prenom, :age)");
    q.prepare("UPDATE nom SET :nom2, :prenom WHERE :age");
}
```

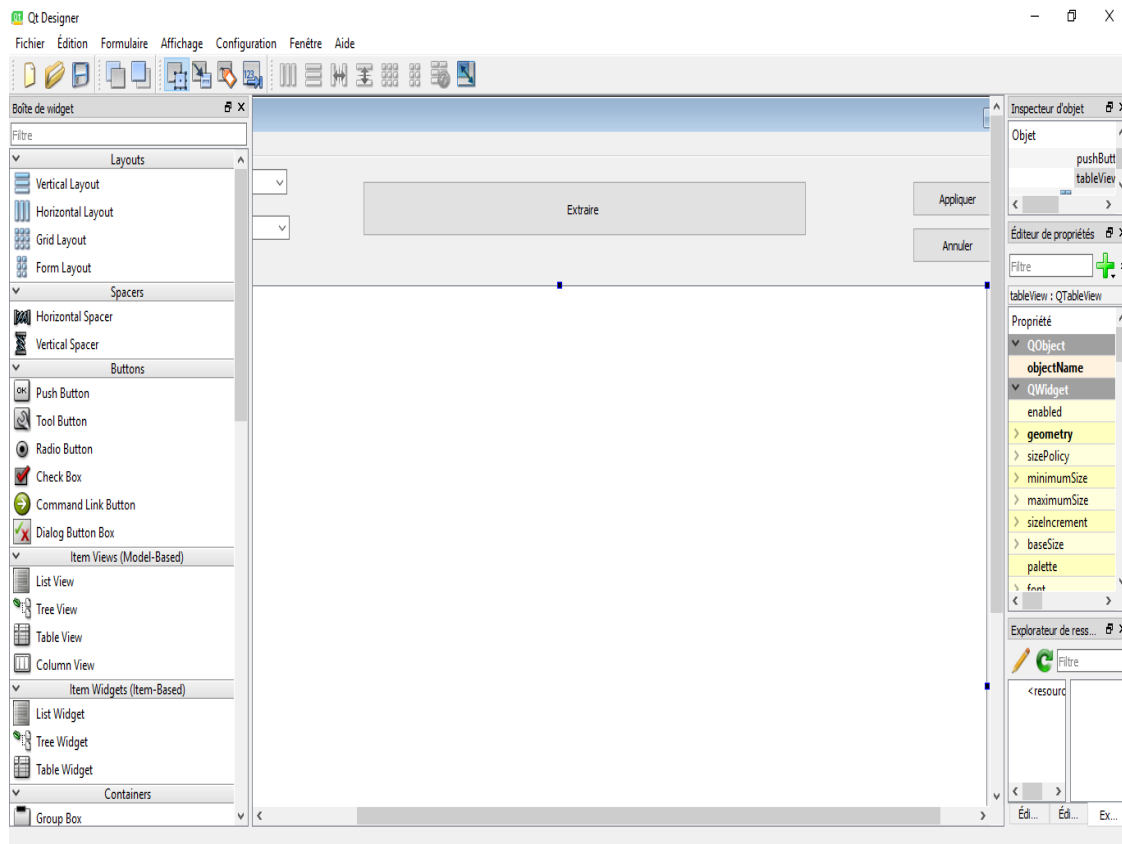
Ceci va permettre de faire une opération supplémentaire sans rajouter de code : en effet, avec la commande « set table » suivi de la table que nous voulons, nous allons pouvoir modifier directement les informations et les mettre à jour directement dans la base de données.

Voici l'IHM utilisée :



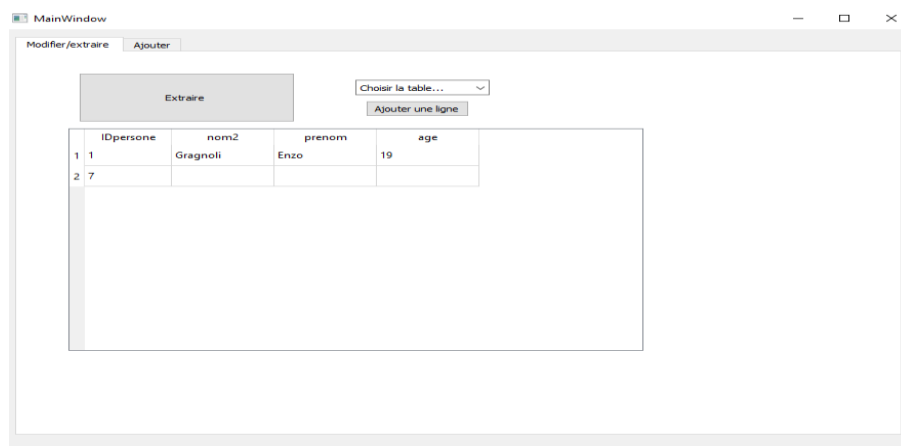
Ici, une interface très simple et minimaliste qui va permettre d'exécuter les commandes au-dessus avec le choix de table que nous voulons.

Pour réaliser cette IHM, je me suis aidée du logiciel QtDesigner qui permet de créer une interface de façon simple



Interface de QtDesigner

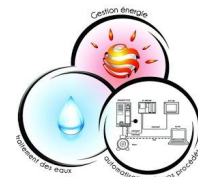
Pour expliquer comment marche l'interface voici comment elle fonctionne :



Ici, lorsque que l'on appuie sur extraire la table s'affiche dans la fenêtre et peut être modifier à souhait et ces modifications vont se répercuter dans la base de données

nom2	1	prenom	age
Gagnoli	Enzo	19	
NULL	NULL	NULL	

Modification apportée dans la base de données !



Problèmes rencontrés

Dans cette partie des problèmes rencontrés, nous allons voir quelques problèmes liés à la programmation elle-même ou au problème de compatibilité.

Problème n°1 : Lors de la compilation sur Qt, nous pouvons avoir ce message d'erreur qui apparaît :

```
"Can't connect to MySQL server on 'localhost' (10061) QMYSQL: Unable to connect"
```

Ce message signifie que nous ne pouvons pas nous connecter au serveur MySQL nommé « localhost »

Solution : le problème vient du fait que le logiciel WampServer n'est pas exécuter.

Problème n°2 :

Pour la compilation sur Raspberry Pi, un message d'alerte apparaît :

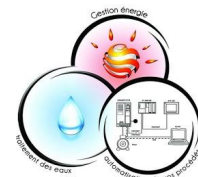
```
Error while building/deploying project test (kit: Desktop)  
When executing step "qmake"
```

Solution :

En effet, les paramètres de compilation n'étant pas configurés par défaut, nous devons le faire manuellement en allant dans les paramètres de Qt.

Pour cela :

- Aller dans le menu déroulant outils -> option
- Compiler & Exécuter -> compilateur
- Dans auto détecte ou manuel cliquer sur Ajouter
- GCC-> C++
- Appliquer
- Relancer la compilation



3eme problème :

```
! undefined reference to `MainWindow::on_comboBox_3_clicked()' moc_mainwindow.cpp  
moc_mainwindow.cpp
```

Solution :

Ici le problème qu'il ne trouve pas la ligne « on :comboBox_3_clicked() » dans le .cpp

Donc la solution la plus rapide serait de :

- Aller dans le .h
- Sélectionner la ligne `void on_comboBox_3_clicked()` ;
- Faire un clic droit
- Sélectionner Refractor puis ajouter la définition dans mainwindow.cpp

Nous avons le résultat présent :

```
void MainWindow::on_comboBox_3_clicked()  
{  
  
}
```

Pour des raisons de comptabilité, il serait préférable de faire :

- Faire clic droit le projet
- Cliquer sur Nettoyer
- Faire la modification expliquée au-dessus
- Refaire un clic droit sur le projet
- Cliquer sur recompiler

Pour la Raspberry Pi, nous avons eu 2 problèmes majeurs :

Problème de compatibilité au niveau des logiciels pour la base de données tel que : MySQL, PhpMyAdmin et apache2

Solution : désinstaller complètement tous les traces des logiciels présent sur la Raspberry puis recommencer l'installation

Les packages de MySQL, PhpMyAdmin et apache2 disponibles sur les dépôts par défaut de Raspbian ne sont pas compatibles avec la version stretch actuellement distribuée.

Solution : configurer l'installation en modifiant la liste des dépôts.

Partie Physique : réseaux Wi-Fi

Le dongle Wifi va permettre de se connecter directement à la Raspberry.
Celle-ci va permettre de se connecter en Wifi a des moments précis pour ne pas avoir de la Wifi constamment pour éviter tout risque de piratage et ainsi pour que le technicien en charge puisse s'occuper de la maintenance en toute sécurité.

Pour caractériser un dongle, nous avons plusieurs critères de classification :

- Fréquence
- Modulation
- Protocole de communication et de sécurité
- Architecture du réseau
- Débit
- Portée



*Exemple de Dongle
Wifi dans le bas-
milieu de gamme*

Pour montrer des caractéristiques nous allons les monter avec le dongle ci-dessus qui est D-Link DWA-121 :

La modulation pour le wifi peut être différente en fonction des lettres suivant de la portée de communication, de l'utilisation et du débit.

	802.11n	802.11n IEEE Specification	802.11ac Wave 1 Today	802.11ac Wave2 WFA Certification Process Continues	802.11ac IEEE Specification
Band	2.4 GHz & 5 GHz	2.4 GHz & 5 GHz	5 GHz	5 GHz	5 GHz
MIMO	Single User (SU)	Single User (SU)	Single User (SU)	Multi User (MU)	Multi User (MU)
PHY Rate	450 Mbps	600 Mbps	1.3 Gbps	2.34 Gbps - 3.47 Gbps	6.9 Gbps
Channel Width	20 or 40 MHz	20 or 40 MHz	20, 40, 80 MHz	20, 40, 80, 80-80, 160 MHz	20, 40, 80, 80-80, 160 MHz
Modulation	64 QAM	64 QAM	256 QAM	256 QAM	256 QAM
Spatial Streams	3	4	3	3-4	8
MAC Throughout*	293 Mbps	390 Mbps	845 Mbps	1.52 Gbps- 2.26 Gbps	4.49 Gbps

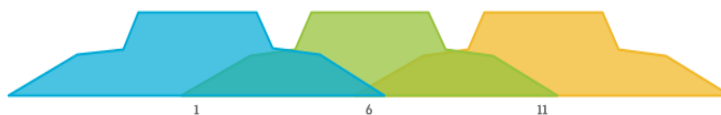
* Assuming a 65% MAC efficiency with highest MCS

Source : <https://le-routeur-wifi.com/normes-wi-fi-802-11abgnac-comprendre-le-wifi-et-ses-normes/>

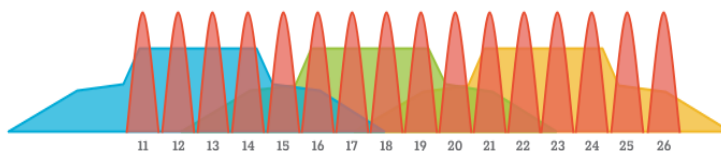
Pour Comprendre l'utilisation du 2.4GHz a la place de l'actuel 5GHz, il y a plusieurs raisons pourquoi il est toujours autant utilisé à nos jours :

- Dans un premier temps, le fait que de nombreux périphérique ne sont pas compatible avec le 5Ghz
- La portée du réseaux 5GHz est bien moindre mais propose plus de bande passante

Le problème avec le protocole ZigBee et le wifi, c'est les canaux qui se superpose entre eux et peuvent crée des interférences



2.4 GHz WiFi Channels

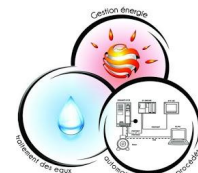


Ici, on peut voir que lorsque les deux protocoles sont dans le même canaux, le wifi à interférer avec le Zigbee.

Pour une utilisation optimale des deux protocoles, nous devons les séparer et faire en sorte qu'ils ne soient pas dans les mêmes canaux.



Source : <https://www.metageek.com/training/resources/zigbee-wifi-coexistence.html>



Journal de bord Personnel

11/01/18 : création du Google drive pour le projet, lecture du dossier du système et début du Project

12/01/18 : Finalisation du Project et configuration des module XBee a l'aide de X-CTU

17/01/18 : début de l'installation et configuration des logiciels *MySQL Workbench* et WampServer

19/01/18 : début de la création des tables et de la base de données

7/02/18 : fin de la création de la base de données + modification dans PhpMyAdmin

15/02/17 : début de l'utilisation de la Raspberry Pi avec l'installation de LAMP + tentative d'installation de Qt

21/02/18 : début de la rédaction du rapport ainsi que du PowerPoint pour la première revue de projet

2/03/18 : succès de l'installation de la Raspberry Pi et des modules PhpMyAdmin, MySQL, Apache 2 ainsi que du logiciel QtCreator

7/03/18 : début de la réflexion sur l'application pour la base de données + modification interne de la base de données

9/03/18 : début de la création de l'application pour les base de données

18/04/18 : Visite de l'entreprise API et présentation de l'état d'avancement du projet

2/05/18 : installation des logiciel Heidi SQL pour une phase de test de l'application en hors connexion

16/05/18 : fin de la création de l'application de base de données

Partie Maxime EECKEMAN

La partie du projet qui me concerne consiste à gérer la partie « maître », pour cela j'utilise une raspberry pi ainsi que plusieurs capteurs qui seront connectés à la carte via un shield qui sera implantable directement sur la raspberry.

Le matériel que j'utilise est donc le suivant :

- Une carte RaspBerry Pi 3.
- Un GPS grove.
- Une horloge temps réel (DS3231)
- Un module Xbee
- Un module GSM/GPRS (TEL0113)

D'autres éléments imposés par l'entreprise devront être présents sur le shield que je dois concevoir comme notamment le parafoudre.

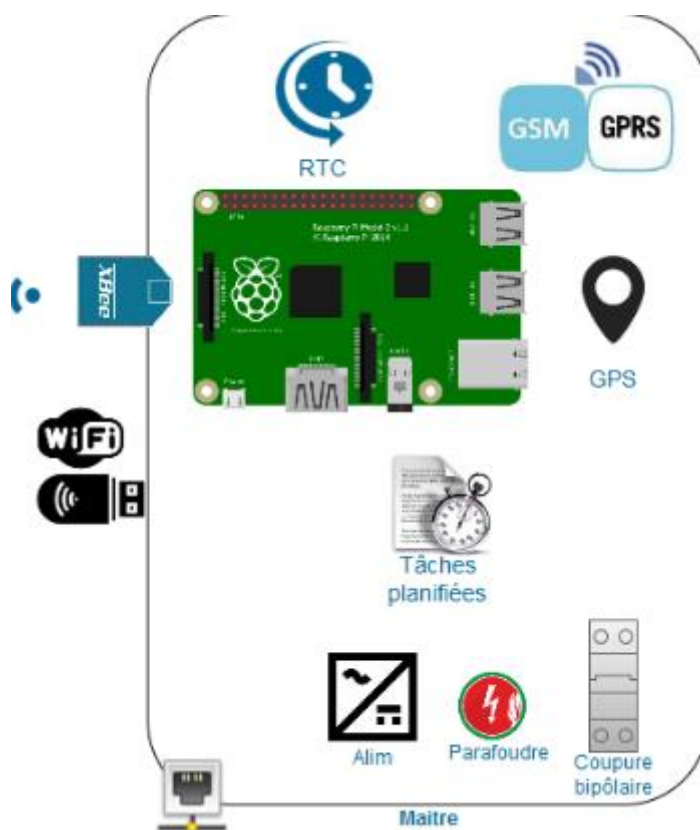


Diagramme de Gantt

Diagramme de Gantt provisoire :

Tâches Maxime	34,43 jours?	Ven 19/01/1	Ven 15/06/18	
Xbee	20 hr	Ven 19/01/1	Mer 31/01/18	7
horloge RTC	10 hr	Mer 31/01/1	Jeu 01/02/18	45
mise en page IHM	30 hr	Ven 02/02/1	Jeu 15/03/18	46
Alim	10 hr	Jeu 15/03/18	Jeu 22/03/18	47;59
GPS	35 hr	Jeu 22/03/18	Jeu 05/04/18	48
WIFI	10 hr	Jeu 05/04/18	Mer 11/04/18	49
GSM/GPRS	35 hr	Mer 11/04/1	Jeu 10/05/18	50
creation du schema structurel	20 hr	Jeu 10/05/18	Jeu 17/05/18	51
routage	30 hr	Jeu 17/05/18	Jeu 31/05/18	52
commande	20 hr	Jeu 31/05/18	Jeu 07/06/18	53
Parafoudre	5 hr	Jeu 07/06/18	Ven 08/06/18	54
EC2 Phase 1	53 hr	Ven 19/01/1	Lun 12/03/18	
EC2 Phase 2	118 hr	Mar 13/03/1	Jeu 10/05/18	56
EC2 Phase 3	89 hr	Ven 11/05/1	Ven 15/06/18	57

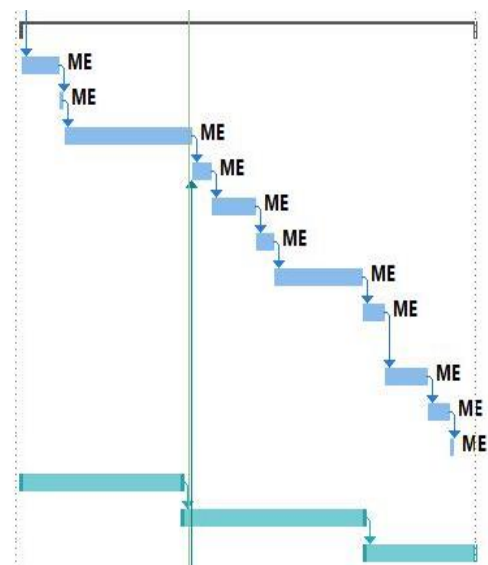
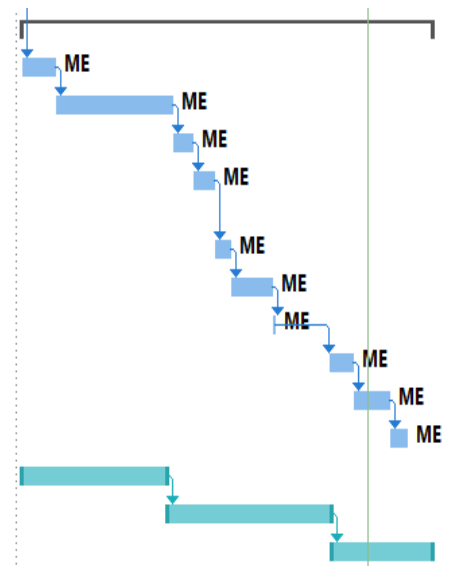
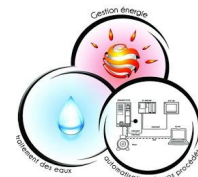


Diagramme de Gantt réel :

Tâches Maxime	34,43 jours?	Ven 19/01/1	Ven 15/06/18		ME
Xbee	20 hr	Ven 19/01/1	Mer 31/01/18	7	ME
mise en page IHM	30 hr	Mer 31/01/1	Mer 14/03/18	42	ME
horloge RTC	20 hr	Mer 14/03/1	Jeu 22/03/18	43	ME
creation du schema structurel	20 hr	Jeu 22/03/18	Jeu 29/03/18	44	ME
WIFI	10 hr	Jeu 29/03/18	Mer 04/04/18	45	ME
GPS	35 hr	Mer 04/04/1	Jeu 19/04/18	46	ME
commande	10 hr	Jeu 19/04/18	Ven 20/04/18	47	ME
GSM/GPRS	30 hr	Mer 09/05/1	Ven 18/05/18	48	ME
routage	30 hr	Ven 18/05/1	Jeu 31/05/18	49	ME
Parafoudre	5 hr	Jeu 31/05/18	Mer 06/06/18	50	ME
EC2 Phase 1	53 hr	Ven 19/01/1	Lun 12/03/18		
EC2 Phase 2	118 hr	Mar 13/03/1	Jeu 10/05/18	52	
EC2 Phase 3	89 hr	Ven 11/05/1	Ven 15/06/18	53	





Plan des Parties

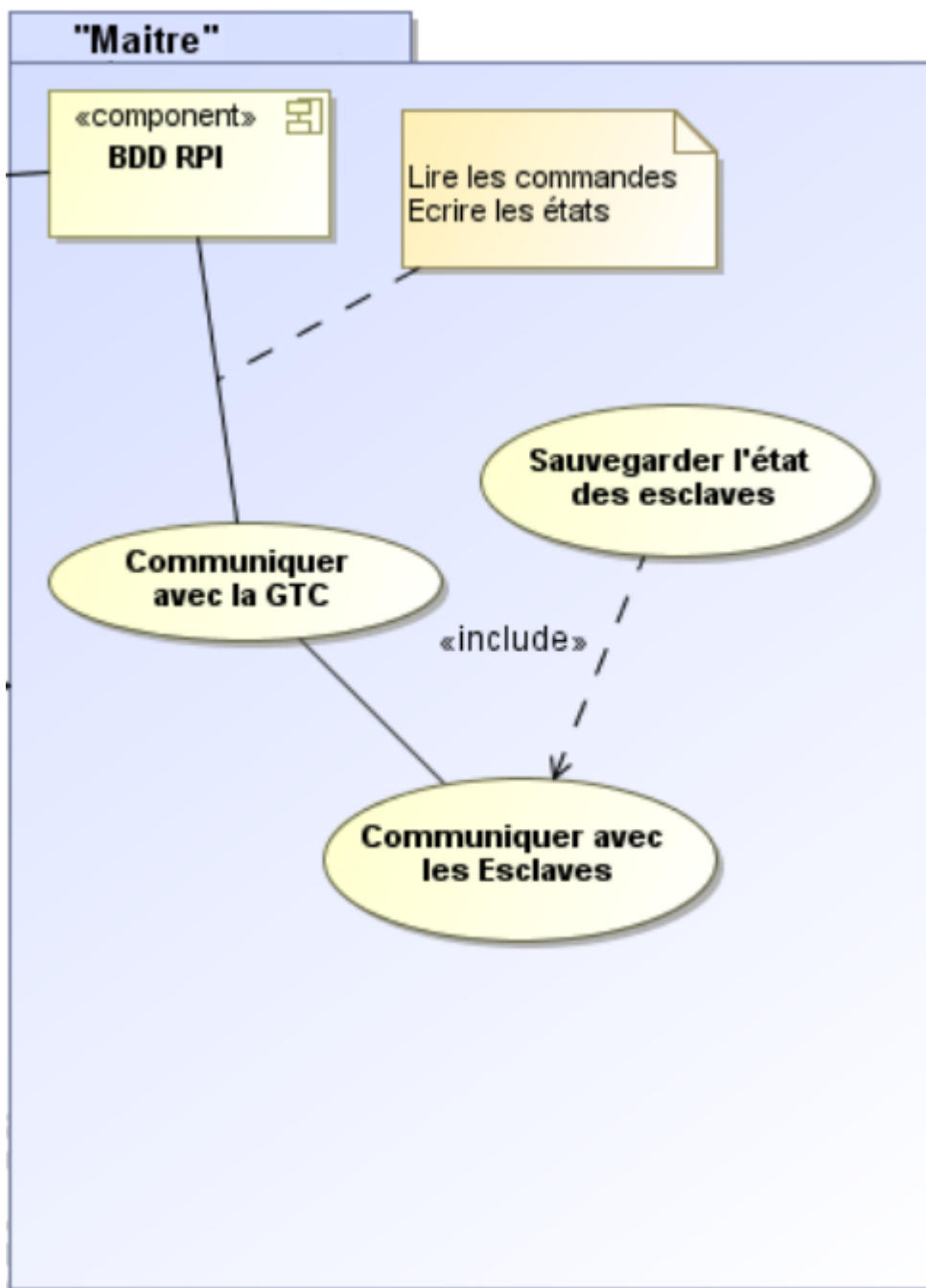
Ma partie du projet est composée de 3 parties qui concerne chacune plusieurs éléments que je dois mettre en fonction. Chaque Partie correspond à des objectifs prévisionnels à effectuer pour chaque revue de projet.

Partie 1 : Cette partie concerne la programmation et la mise en place de la communication avec un module Xbee, la création d'un programme en C++ et d'une IHM permettant de gérer les éléments que le maître doit pouvoir commander.

Partie 2 : La seconde partie est dédiée à l'utilisation et la mise en place de l'horloge temps réel, la programmation du GPS, le choix et la mise en place d'une alimentation correctement dimensionnée pour la raspberry pi et l'utilisation d'une clé Wifi pour permettre une communication locale via Wifi.

Partie 3 : Cette dernière partie a pour but l'utilisation d'un GSM/GPRS qui permettra d'avoir un accès à internet si aucun réseau Wifi n'est disponible, la création du schéma structurel, le routage et la création des fichiers permettant de commander les cartes qui se placeront sur la raspberry pi et qui contiendront l'ensemble des éléments nécessaires au bon fonctionnement du projet.

Cas d'utilisation

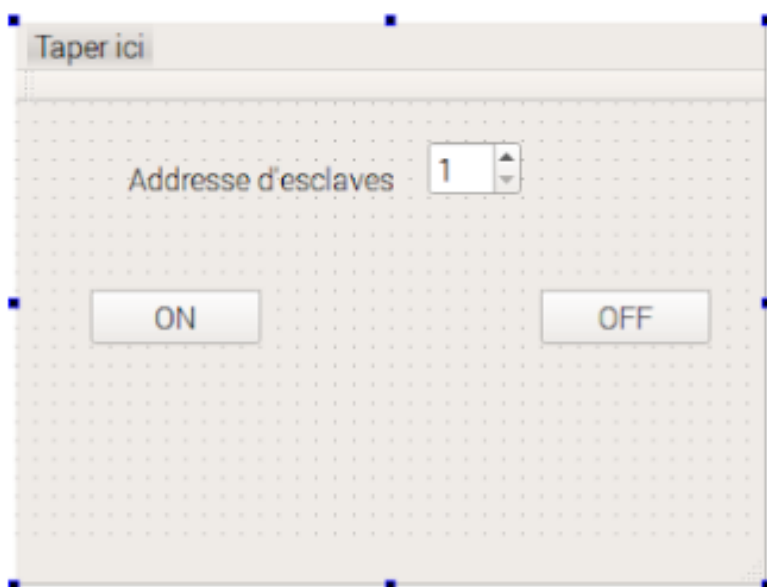


Le but principal de la partie qui m'est dédiée est de rendre possible la communication entre le maître et les esclaves pour recevoir les données que retournent les lampadaires ainsi que de récupérer des données qui sont demandées dans le cahier des charges.

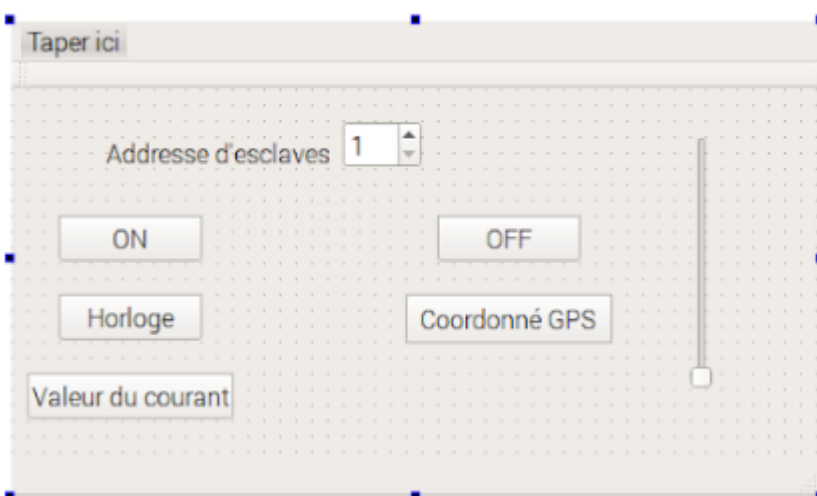
Activités de mise en œuvre

Durant l'avancée du projet j'ai créé et fait évoluer des programmes de test et de mise en place des éléments de ma première itération

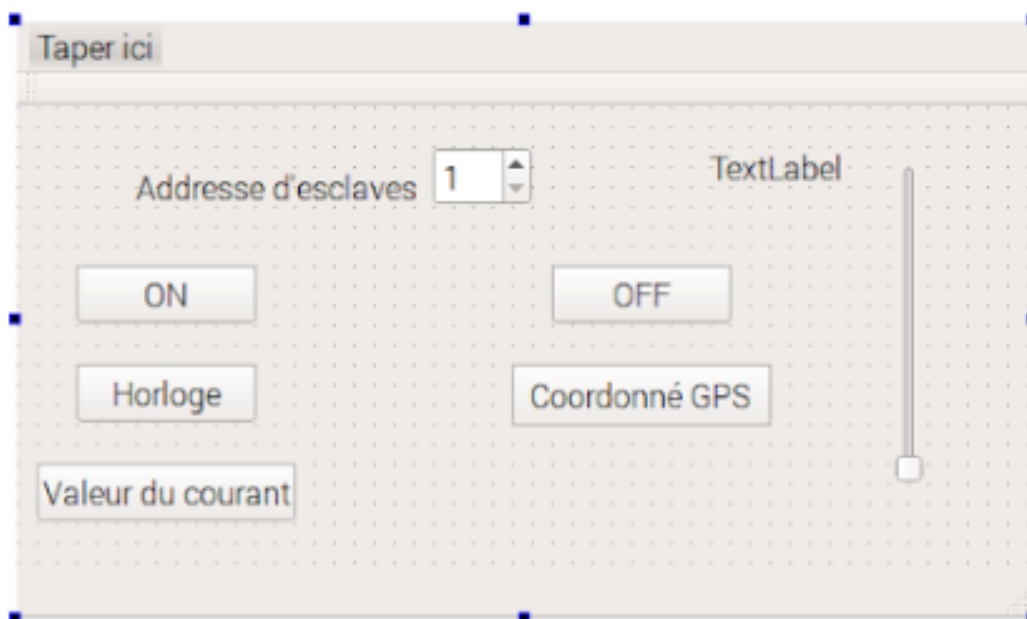
- Différentes versions de l'IHM permettant de mettre en avant les possibilités du maître :
Version 1 ne disposant que de deux boutons permettant d'activer et de désactiver le port série et une scroll bar pour sélectionner une adresse d'esclave.



Version 2 où j'ai ajouté deux boutons permettant d'envoyer une trame pour interroger l'esclave et lui demander ses coordonnées GPS et la valeur de l'intensité du courant.



Troisième version ou une scroll bar de gestion de l'intensité du dimmer a été ajoutée et programmée. Les boutons permettent maintenant d'allumer une LED et de l'éteindre



Détail du câblage de l'horloge temps réel :

Disposition des broches du DS3231

TOP VIEW

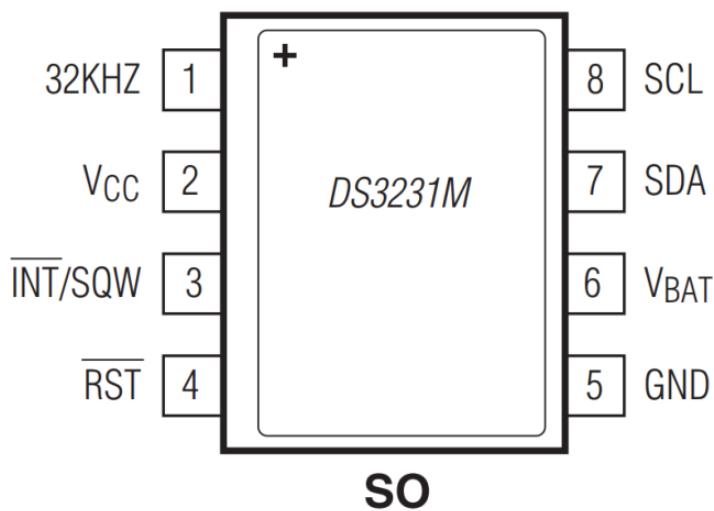
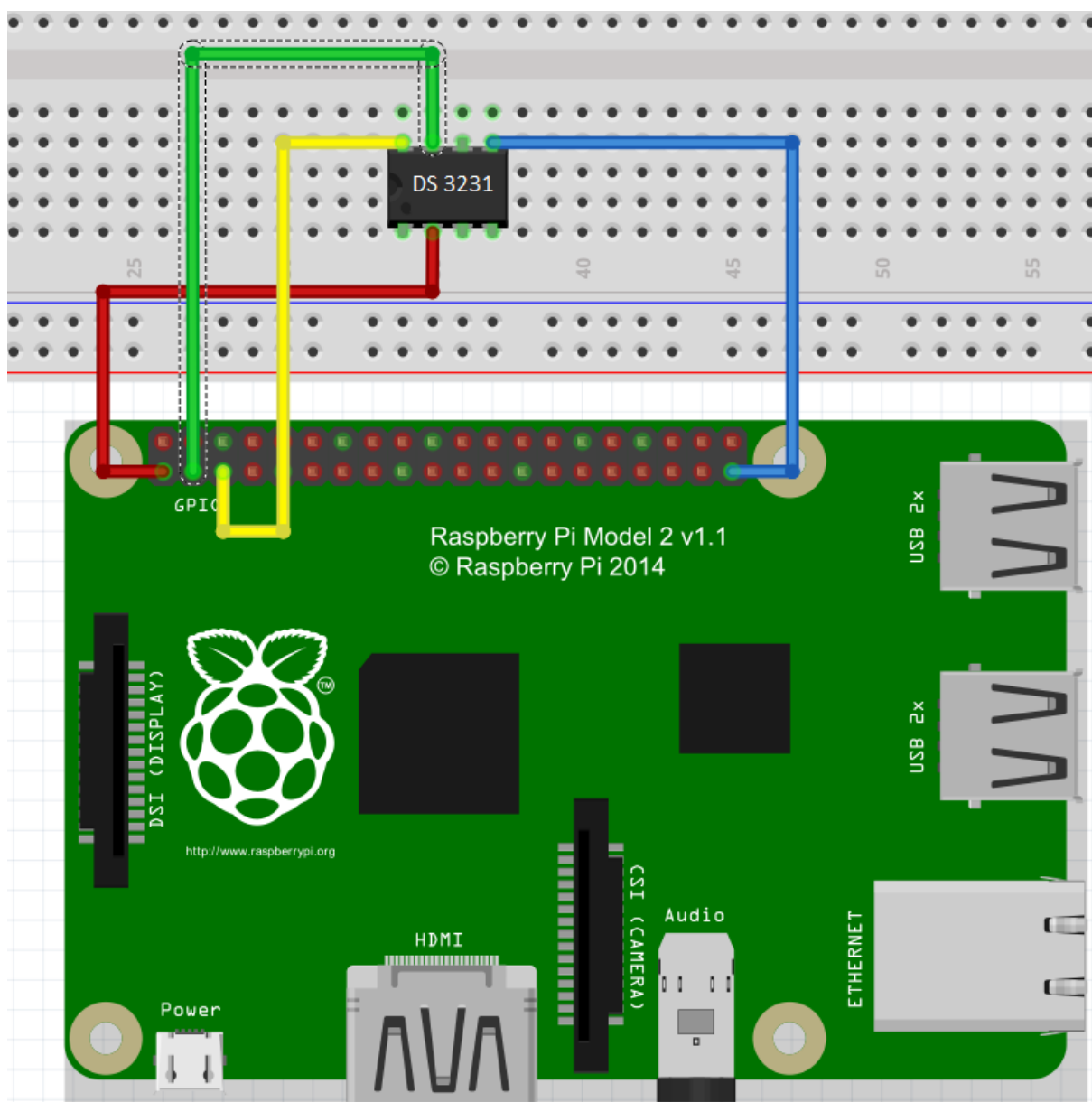


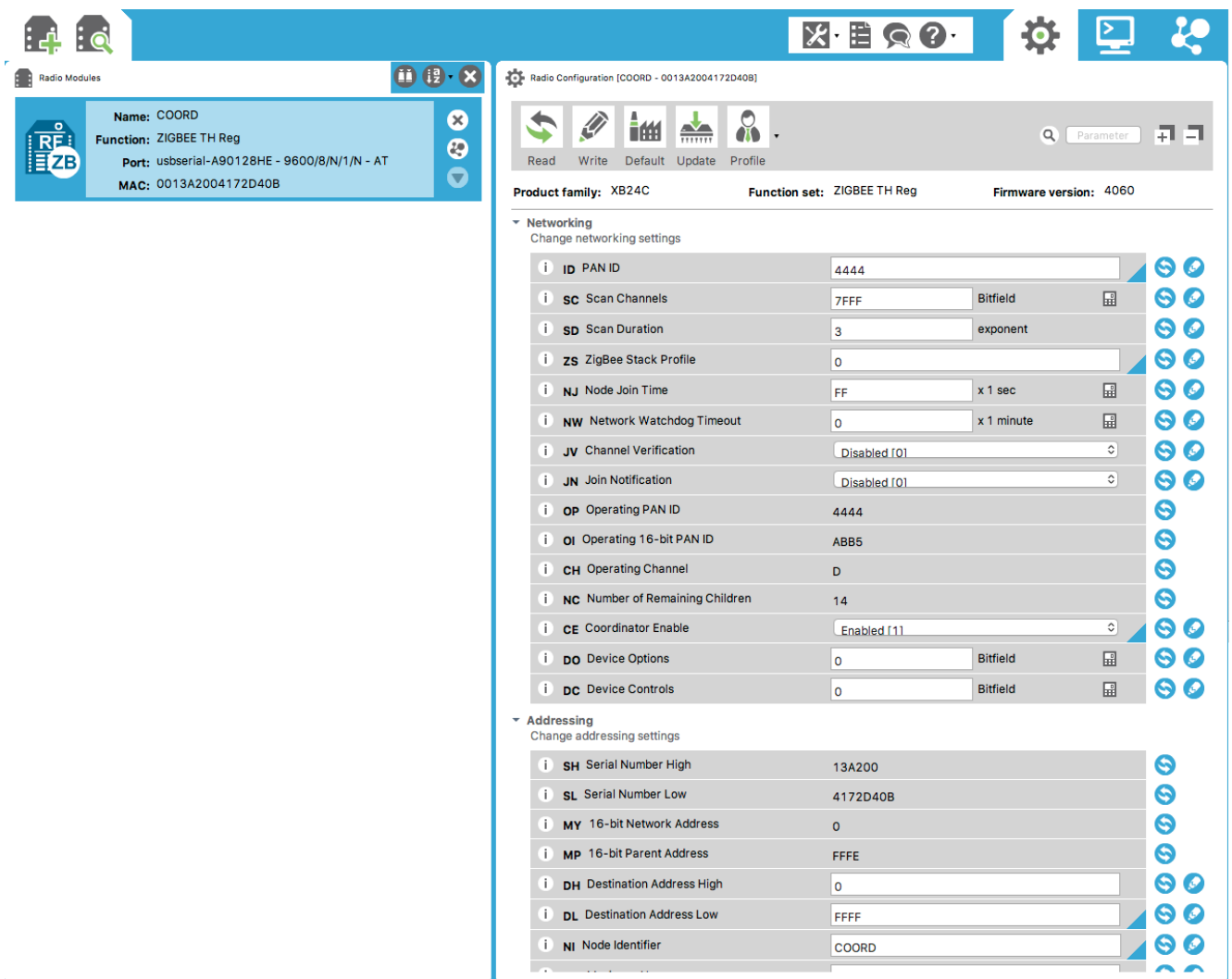
Schéma de câblage rapide sur fritzing :



Partie 1

- **Partie 1** : Cette partie concerne la programmation et la mise en place de la communication avec un module Xbee, la création d'un programme en C++ et d'une IHM permettant de gérer les éléments que le maître doit pouvoir commander.

Configuration du module xbee : J'ai utilisé le logiciel XCTU pour programmer le module de communication xbee



The screenshot shows the XCTU software interface for configuring a ZigBee module. The left sidebar displays the 'Radio Modules' list with details for a module named 'COORD'.

Radio Modules List:

Name	Function	Port	MAC
COORD	ZIGBEE TH Reg	usbserial-A90128HE - 9600/8/N/1/N - AT	0013A2004172D40B

The main window shows the 'Radio Configuration' for the 'COORD' module. The 'Networking' section is expanded, showing various settings for the ZigBee network.

Networking Settings:

Parameter	Value	Unit/Type
ID PAN ID	4444	
SC Scan Channels	7FFF	Bitfield
SD Scan Duration	3	exponent
ZS ZigBee Stack Profile	0	
NJ Node Join Time	FF	x 1 sec
NW Network Watchdog Timeout	0	x 1 minute
JV Channel Verification	Disabled [0]	
JN Join Notification	Disabled [0]	
OP Operating PAN ID	4444	
OI Operating 16-bit PAN ID	ABB5	
CH Operating Channel	D	
NC Number of Remaining Children	14	
CE Coordinator Enable	Enabled [1]	
DO Device Options	0	Bitfield
DC Device Controls	0	Bitfield

The 'Addressing' section is also expanded, showing settings for the device's address and identifier.

Addressing Settings:

Parameter	Value
SH Serial Number High	13A200
SL Serial Number Low	4172D40B
MY 16-bit Network Address	0
MP 16-bit Parent Address	FFFE
DH Destination Address High	0
DL Destination Address Low	FFFF
NI Node Identifier	COORD

Chaque module doit être configuré avec des paramètres qui lui permettront de communiquer avec les autres modules et définiront s'il sera utilisé en tant qu'esclave ou en tant que maître.

Param	XBeeA	XBeeB	XBeeC
ID	5555	5555	5555
JV	[0]	Enable [1]	Enable [1]
CE	Enable [1]	[0]	[0]
DH	0	0	0
DL	FFFF	0	0
NI	COORD	ROUTER1	ROUTER2

ID : L'ID doit être le même pour tous les modules Xbee que l'on souhaite faire communiquer entre eux.

JV et CE : Ces champs permettent de définir si le module Xbee sera un esclave ou un maître.

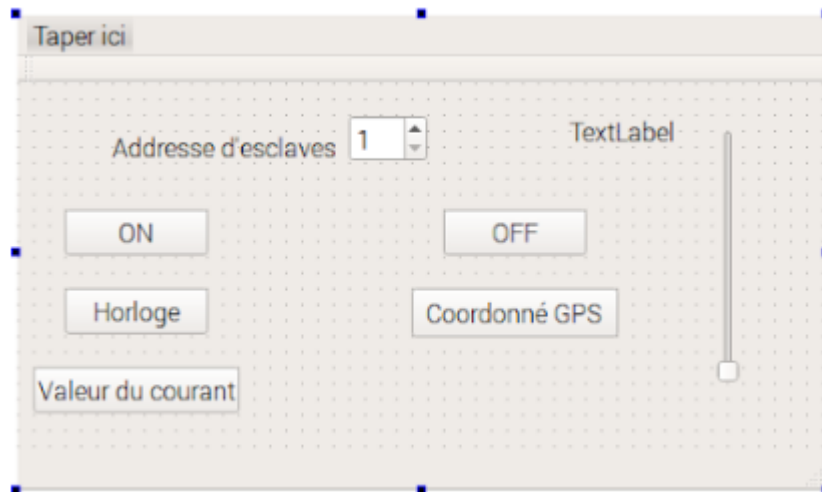
DL : DL concerne uniquement le maître, c'est l'adresse sur laquelle le module Xbee va émettre. Il faut fixer cette valeur à « FFFF », ce qui correspond à l'adresse de diffusion.

NI : C'est un identifiant fixé par l'utilisateur qui lui permet de se repérer.

Cette partie concerne la configuration du module maître qui sera donc connecté sur la raspberry.

Pour le configurer il suffit de rentrer les informations de la colonne correspondant à la configuration du module que l'on souhaite configurer, la colonne XBeeA correspond à un « maître » et les XBeeB et XBeeC correspondent à des modules esclaves.

Création d'une IHM de contrôle d'utilisation du maître : L'IHM devant être réalisée avec un code en C++ j'ai fait le choix d'utiliser le logiciel Qt



Chaque bouton permet d'ouvrir le port série d'envoyer une commande comprise dans une trame qui se compose de la manière suivante : « ID esclave ; Commande ; Donnée ; CRC » puis par la suite de refermer le port série pour le laisser libre.

Pour l'ouverture j'utilise la ligne de code suivante :

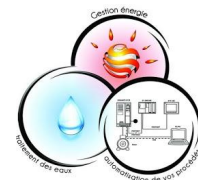
```
fser.open("/dev/ttyUSB0");
```

Pour l'envoi de la trame j'utilise :

```
fser<< ID esclave << "Commande;Donnée;CRC";
```

Enfin pour fermer le port série :

```
fser.close();
```



Partie 2

- **Partie 2** : La seconde partie est dédiée à l'utilisation et la mise en place de l'horloge temps réel, la programmation du GPS, le choix et la mise en place d'une alimentation correctement dimensionnée pour la raspberry pi et l'utilisation d'une clé Wifi pour permettre une communication sur internet via Wifi.

Mise en place de la RTC DS3231 :

La RTC se connecte en i2c sur la raspberry pi, il va donc falloir activer la communication i2c sur la carte pour faire fonctionner le DS3231.

Activation de l'i2c sur la carte raspberry pi :

Ouvrir un Terminal et se mettre en mode super user :

```
sudo su
```

Editer le fichier **/etc/module** avec l'éditeur nano :

```
nano /etc/module
```

Ajouter les lignes :

```
i2c-bcm2708  
i2c-dev
```

Sauvegarder puis lancer l'interface de configuration :

```
raspi-config
```

Entrer dans le menu « Advanced Options », sélectionner l'i2c et valider.

Quitter puis redémarrer la carte :

```
reboot
```

Pour vérifier le bon fonctionnement de l'i2c sur la carte

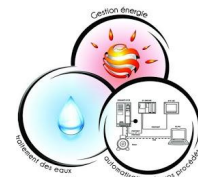
```
dmesg|grep i2c
```

Réponse attendu : **i2c /dev entries driver**

Initialisation du DS3231 :

Il faut écrire la valeur 0x04 dans le registre « control » qui est le 0x0E ainsi que la valeur 0x08 dans le registre « status » qui est le 0x0F

```
i2cset -y 1 0x68 0x0E 0x04 b
```



```
i2cset -y 1 0x68 0x0F b
```

Réponse attendu : **0x88**

```
i2cset -y 1 0x68 0x0F 0x08 b
```

```
i2cset -y 1 0x68 0x0F b
```

Réponse attendu : **0x08**

Il est à présent possible de lire les secondes, minutes, heures, jours, mois et années en questionnant le bon registre, registres allant de 0x01 à 0x06 et correspondant à chacune des valeurs ci-dessus dans l'ordre.

Par exemple la commande « **i2cget -y 1 0x68 0x06 b** » va permettre d'obtenir la valeur hexadécimale correspondant à l'année.

Il faut maintenant installer la RTC. Pour cela il faut que la raspberry soit connecté à internet, pour avoir accès à l'heure du réseau. Puis taper la commande suivante :

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

(Dans cette commande on peut voir la référence d'un composant DS1307, c'est une autre horloge temps réel mais qui est compatible avec le DS3231.)

Il faut maintenant affecter la RTC au démarrage de la raspberry pi, pour que l'heure qu'elle puisse avoir accès à une heure même déconnecté du réseau.

Editer le fichier **/etc/modules** avec l'éditeur nano :

```
nano /etc/modules
```

Rajouter la ligne :

```
rtc-ds1307
```

Sauvegarder et quitter.

Editer le fichier **/etc/rc.local** avec l'éditeur nano :

```
nano /etc/rc.local
```

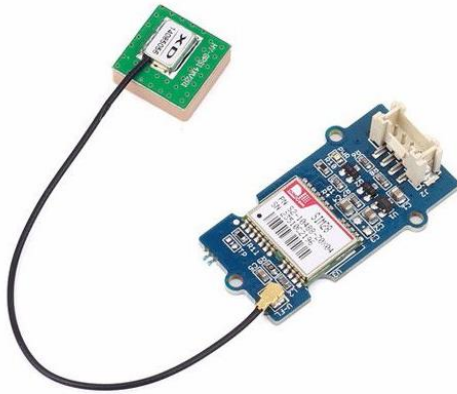
avant la ligne « exit 0 » rajouter les lignes

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

```
sudo hwclock -s
```

Sauvegarder et quitter.

Utilisation du GPS Grove :



Pour obtenir des informations que renvoie le GPS j'ai modifié et fait évoluer un programme en C++ issu d'un TP crée pendant l'année, le programme final est le suivant :

```
#include <iostream>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>

using namespace std;

// g++ -o GPS1 -lwiringPi GPS1.cpp

int main ()
{
    int fd;

    if (wiringPiSetup () == -1)
    {
        cout<<"Impossible de démarrer wirinPi. Erreur : "<<strerror (errno);
        return 1;
    }

    if ((fd = serialOpen ("/dev/ttyS0",9600)) < 0)
    {
        cout<<"Impossible d'ouvrir le port /dev/ttyAMA0. Erreur : "<<strerror (errno);
        return 1;
    }

    char data;
    char trame[176];
    string sTrame;

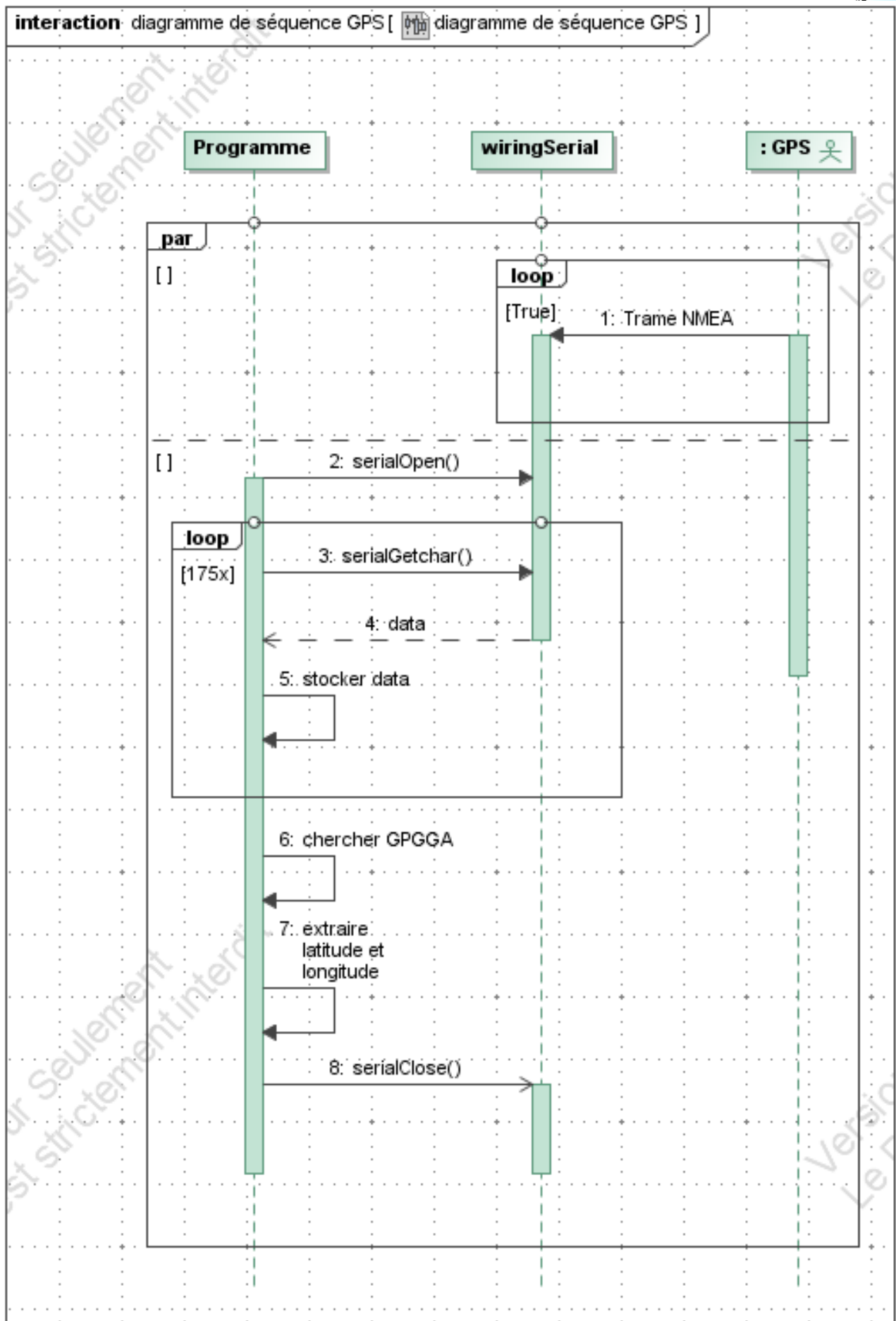
    for (int i=0;i<175;i++)
    {
        data = serialGetchar(fd); // Lecture et stockage dans data
        //data&=128;
        trame[i] = data;
    }

    sTrame = trame;
    cout << sTrame;
    cout<<endl;

    serialFlush(fd); // Effacement des buffers série avant lecture Rx

    serialClose(fd); // Fermeture de l'interface série

    return 0;
}
```



Dimensionnement de l'alimentation.

Après une modification à la demande de l'entreprise, l'alimentation de la carte se fera par le transformateur classique utilisé pour alimenter la raspberry pi 3, Il s'agit d'une alimentation 5V 2,5A micro USB se connectant sur le réseau 230V domestique.



Partie 3

- **Partie 3** : Cette dernière partie a pour but l'utilisation d'un GSM/GPRS qui permettra d'avoir un accès à internet si aucun réseau Wifi n'est disponible, la création du schéma structurel, le routage et la création des fichiers permettant de commander les cartes qui se placeront sur la raspberry pi et qui contiendront l'ensemble des éléments nécessaires au bon fonctionnement du projet.

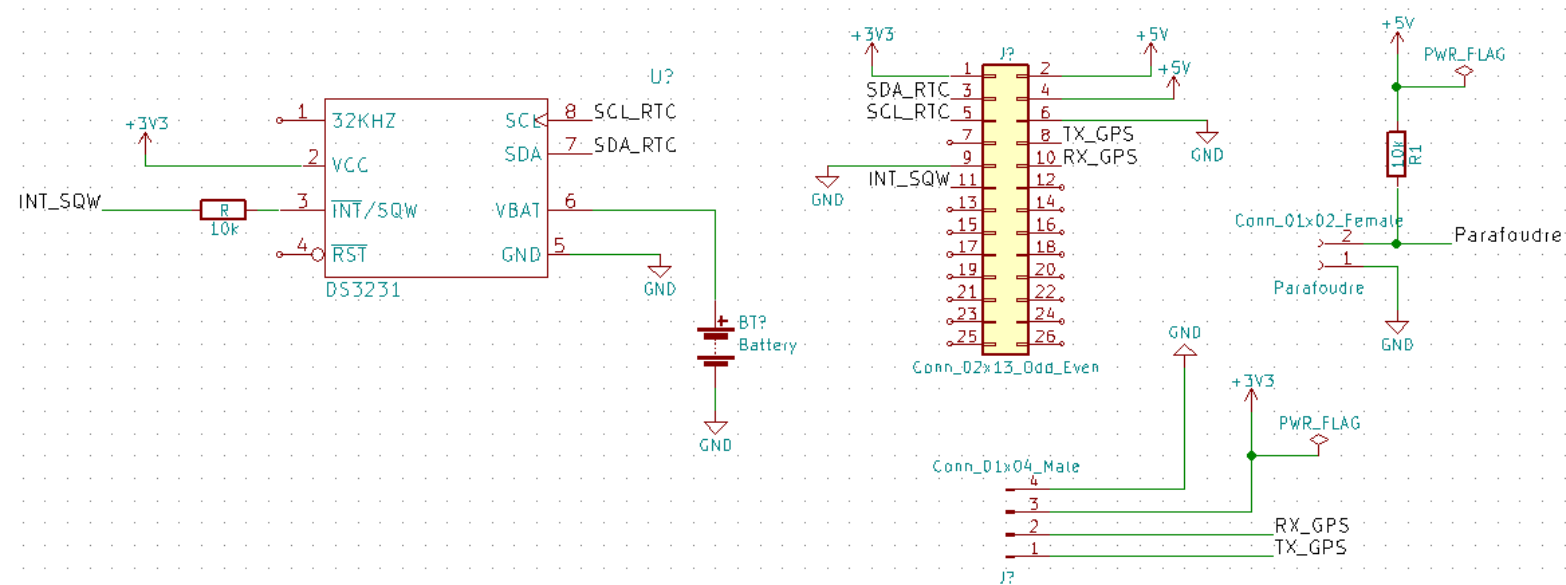
Module GSM/GPRS :



Le module GSM/GPRS permet d'avoir accès à internet via le réseau 3G.
Il se connecte sur le port série de la raspberry mais pour des raisons pratiques j'utiliserais un convertisseur série/USB

Création du schéma structurel :

Pour la création du schéma structurel nous avons utilisé le logiciel kicad. L'entreprise n'ayant pas d'exigence particulière concernant le choix du logiciel nous avons fait le choix de kicad car ce logiciel est gratuit ce qui peut permettre à l'entreprise de faire évoluer le schéma et le routage en interne sans coup supplémentaire.



Partie Physique

Pour la communication entre les modules, les antennes sont indispensables car la distance entre les lampadaires doit être de 3 fois leurs hauteurs.

Exemple : pour un lampadaire de 10m son espacement d'un autre lampadaire doit être de 30m.

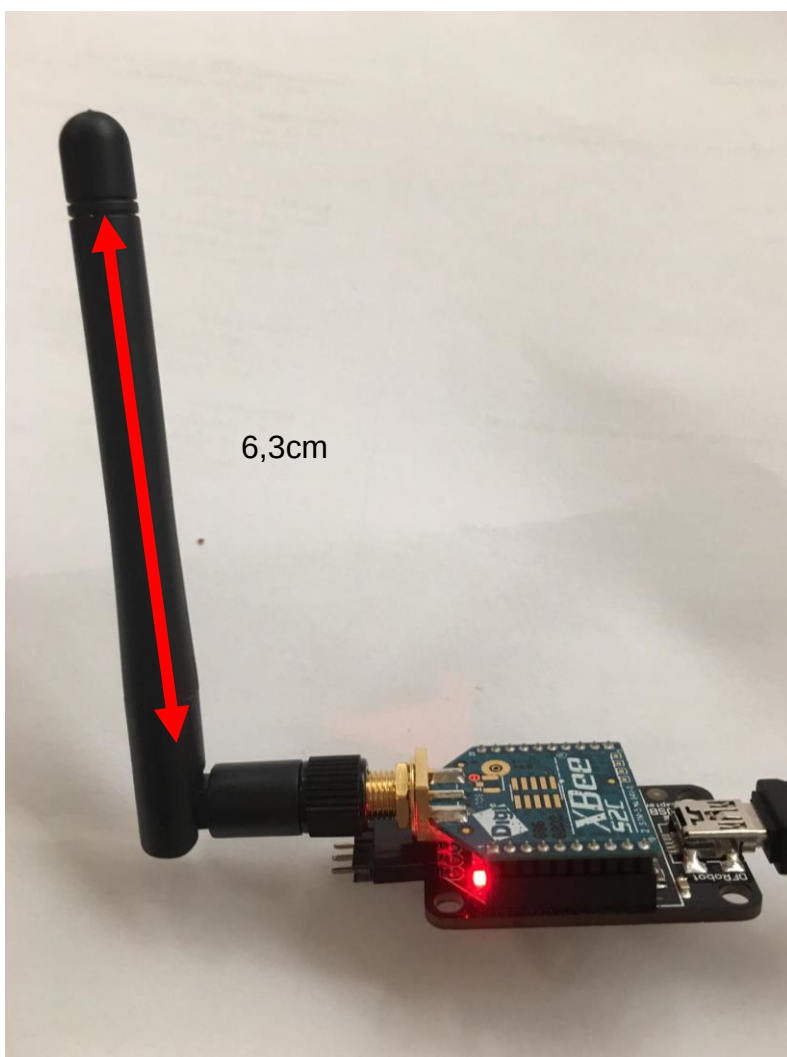
Sachant que le module xbee a une portée de 60m sans l'antenne, à la moindre atténuation du signal ou le dysfonctionnement de l'un des modules xbee la perte de signal empêcherait la communication.

Calcul de la taille de l'antenne nécessaire :

$$\frac{3 \cdot 10^8}{2,4 \cdot 10^9} = 0,125m \rightarrow 12,5 \text{ cm}$$

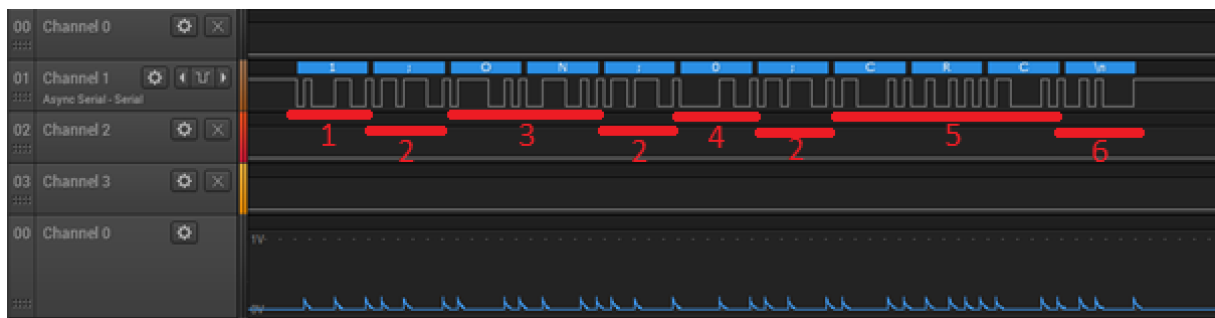
Nous utilisons donc une antenne demi onde

$$\frac{12,5}{2} = 6,25 \text{ cm ce qui correspond à la longueur de notre antenne.}$$



SPECIFICATIONS		XBee® S2C ZigBee	
		Standard	Programmable
PERFORMANCE			
TRANSCIEVER CHIPSET		Silicon Labs EM357 SoC	
DATA RATE		RF 250 Kbps, Serial up to 1 Mbps	
INDOOR/URBAN RANGE		200 ft (60 m)	
OUTDOOR/RF LINE-OF-SIGHT		4000 ft (1200 m)	
FREQUENCY BAND		ISM 2.4 GHz	

Analyse de la trame émise par le module xbee :



- 1** : Identifiant de l'esclave.
- 2** : Séparateur de champ de données.
- 3** : Commande.
- 4** : Valeur de la commande (zéro si non nécessaire).
- 5** : CRC (vérificateur de conformité de la trame).
- 6** : Saut de ligne et retour au début de la ligne.

Etude et comparaison de la consommation des lampadaires :

Eclairer avec la juste quantité nécessaire

Les anciennes lampes étaient des lampes à vapeur de mercure.

Notre tête de lampadaire consomme une puissance de 75W, une tête équivalente équipée de lampes à vapeur de mercure à une consommation de 250W. Ce qui représente une réduction de puissance de 70%

Problèmes et solutions

Le principal problème auquel je dois faire face dans ce projet est la présence de trois éléments qui utilisent un port série pour leur fonctionnement alors que la carte raspberry pi ne dispose que d'un seul port série.

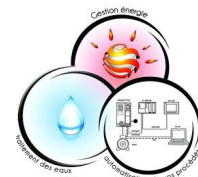
J'ai donc fait le choix d'utiliser des convertisseurs Série/USB pour le module Xbee et pour le GSM/GPRS, le port série de la carte raspberry pi est donc utilisé pour le GPS

Convertisseur série/USB
Du module Xbee



Convertisseur série/USB
du GSM/GPRS





Journal de bord personnel

Jeudi 19 janvier : Prise de connaissance du projet et début de la conception du gantt.

Mercredi 31 janvier : Première utilisation des modules xbee et découverte du logiciel XCTU qui permet de les programmer.

Mercredi 7 février : Début de la conception de l'interface permettant de gérer les demande qu'un maître peut faire à l'esclave.

Jeudi 22 février : mise en place de l'HTC sur la raspberry pi.

Vendredi 30 février : Reprise de la programmation de l'IHM.

Mercredi 21 mars : Première revue de projet.

Jeudi 22 mars : Début de la création du programme pour le GPS.

Vendredi 13 avril : Fin de la création du programme et mise en fonction du GPS.

Mercredi 18 avril : Rencontre avec l'entreprise API.

Mercredi 9 mai : Etude des modifications demandées par l'entreprise.

Jeudi 10 mai : Constitution du dossier et élaboration du diaporama pour la 2ème revue de projet.

Mercredi 23 mai : deuxième revue de projet.

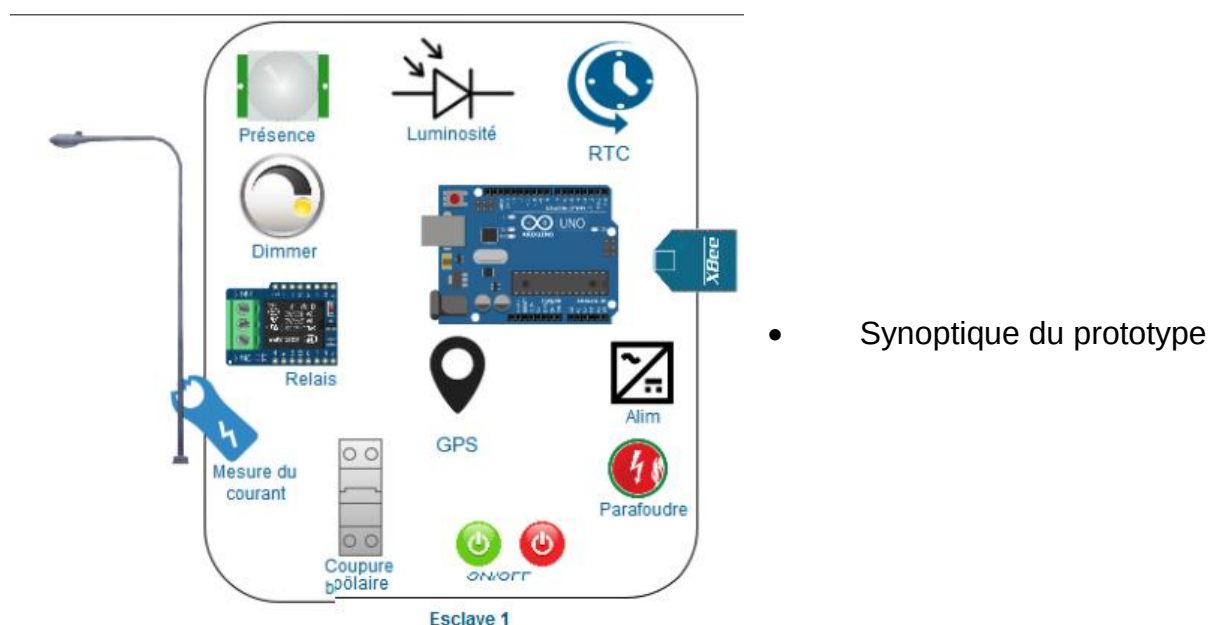
Jeudi 24 mai : Constitution du dossier pour la revue de projet final

Partie Achor Mehdi

Ma partie du projet consiste à mettre en place les « Lampadaires esclaves » qui seront implanté dans la ville, chaque lampadaire esclave se verra attribuer une Arduino et une adresse pour pouvoir être différencié.

Les fonctions principales des esclaves sont : Recevoir les commandes de « Marche/Arrêt »,
Auto-Contrôle de fonctionnement : mesure du courant consommé pour remonter les informations au maître, coupure bipolaire pour intervention ou isolement du lampadaire.

Mon but est de pouvoir faire communiquer grâce à une liaison XBEE le module maître qui sera représenté par une Raspberry et le module esclave qui sera représenté par une arduino .



Une fois que tous les capteurs seront configurés et que la communication maître /esclave sera opérationnelle, la création d'un shield sera alors en vue. Le shield aura pour but de rassembler tous les composants utiliser sur une seule et même carte. La création du shield aura lieux dans l'atelier du lycée par une machine dédiée cela nous permettra d'obtenir le shield plus rapidement car si je la fais sous-traiter sur internet le temps d'attente serra de 2 semaine pour un prix raisonnable (environ 4,99\$ pour 10 cartes) dans des usines délocalisé, pour des usines française le prix grimpe en flèche (100€ pour 10 cartes).

Les composants

Composent	Photo	Prix (ttc)
Arduino UNO Fournisseur : GO tronic Code commande : 25950		19€50.
XBEE Fournisseur : Farnell Code commande : 2766676		22€80.
DS3231MZ+ (RTC) Fournisseur : RS Code commande : 165-9743		5€31.
GPS Fournisseur : GO tronic Code commande : 31275		32€90.
Alimentation a découpage Fournisseur : RS Code commande : 891-9211		29€52.
Capteur de courant Fournisseur : GO tronic Code commande : 33814		5,90€

Tous les composants sont disponibles sur les sites GO TRONIC et RS lien :

<https://www.gotronic.fr>

<https://fr.rs-online.com/web/>

Pour le choix de l'alim a découpage deux choix se sont proposés, celle présentée plus haut qui ne se soude pas sur l'Arduino , elle sera en dehors du shield et l'alim a découpage de marque «MEAN WELL » mais qui elle est soudée sur le shield de la carte Arduino .



Le prix de cette alim a découpage est de 10,68€ contre 29,52€ pour celle sélectionnée. Malgré la différence de prix le choix de cette alim c'est porté sur la plus chère car le 230V du lampadaire ne sera pas apporté sur le shield de l'Aduino ce qui y est très bien pour la sécurité de l'Arduino et du technicien.

Description des composants

TACHES	Parties	DESCRIPTION
XBEE	1	Le module Xbee permet une communication entre les deux modules , on doit donc le mettre en place en premier pour pouvoir continuer.
Horloge RTC	1	L'horloge temps réel nous servira à cadencer tous les composants, elle a donc un rôle important dans le système.
GPS	1	Le module GPS qui nous servira pour la localisation des lampadaires, il a une importance

		car ce sera la première donnée envoyée depuis l'Arduino a l'installation.
Mesure de courant	2	Le composant mesure de courant sera traiter en deuxième partie la car la BDD sera nécessaire pour le traitement des données.
Dimmer	2	Le dimmer serra intègre dans la tête de lampadaire envoyer par l'entreprise.
Capteur de présence	3	Le capteur de présence pour détecter le passage des piétons ne pourra être fonctionnel a 100% si la partie dimmer n'est pas traitée.

Pour la mise en place des composants, j'ai choisi de diviser l'installation en 3 parties qui seront toutes dans un ordre bien précis : dans la 1^{ère} partie on retrouve tous les composants nécessaires pour le bon fonctionnement de l'installation, dans la 2eme et 3eme parties on traite tous les composants qui tournent autour de l'installation.

Diagramme de gantt

Diagramme de gantt prévisionnel :

« Tâches Mehdi	34,71 jours?	Ven 19/01/1	Ven 15/06/18	
Xbee	15 hr	Ven 19/01/1	Ven 26/01/18	7
horloge RTC	10 hr	Ven 26/01/1	Jeu 01/02/18	28
GPS	20 hr	Jeu 01/02/18	Jeu 08/02/18	29
Alim	5 hr	Jeu 08/02/18	Mer 14/03/18	30
Mesure du courant	25 hr	Mer 14/03/1	Jeu 22/03/18	31
Bouton ON/OFF	15 hr	Jeu 22/03/18	Jeu 29/03/18	32;41
Relais	15 hr	Jeu 29/03/18	Jeu 05/04/18	33
Dimmer	20 hr	Jeu 05/04/18	Jeu 12/04/18	34
capteur de presence/Luminosité	40 hr	Jeu 12/04/18	Mer 16/05/18	35
creation du schema structurel	15 hr	Jeu 17/05/18	Mer 23/05/18	36;42
routage	25 hr	Mer 23/05/1	Jeu 31/05/18	37
commande	20 hr	Jeu 31/05/18	Ven 08/06/18	38
Parafoudre	5 hr	Mer 13/06/1	Jeu 14/06/18	39
EC1 Phase 1	60 hr	Ven 19/01/1	Lun 12/03/18	
EC1 Phase 2	118 hr	Mar 13/03/1	Jeu 10/05/18	41
EC1 Phase 3	89 hr	Ven 11/05/1	Ven 15/06/18	42

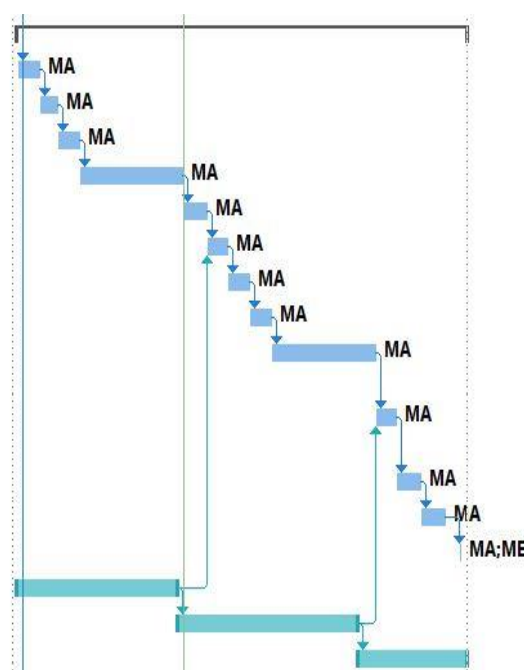
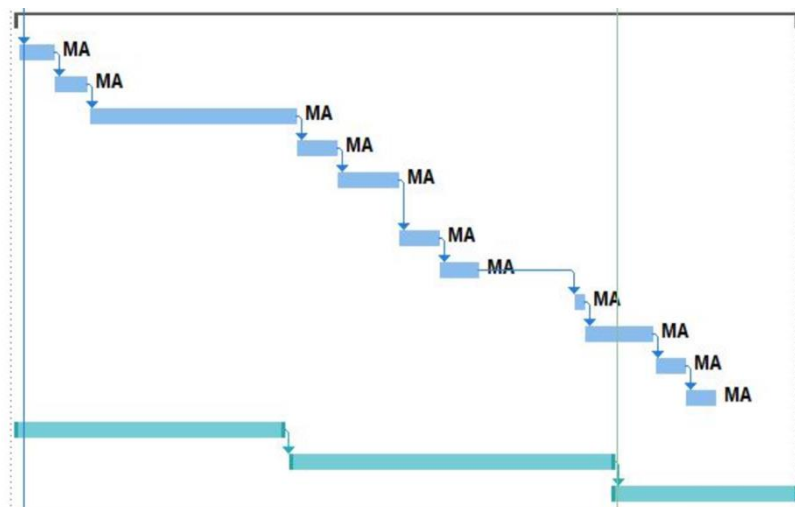
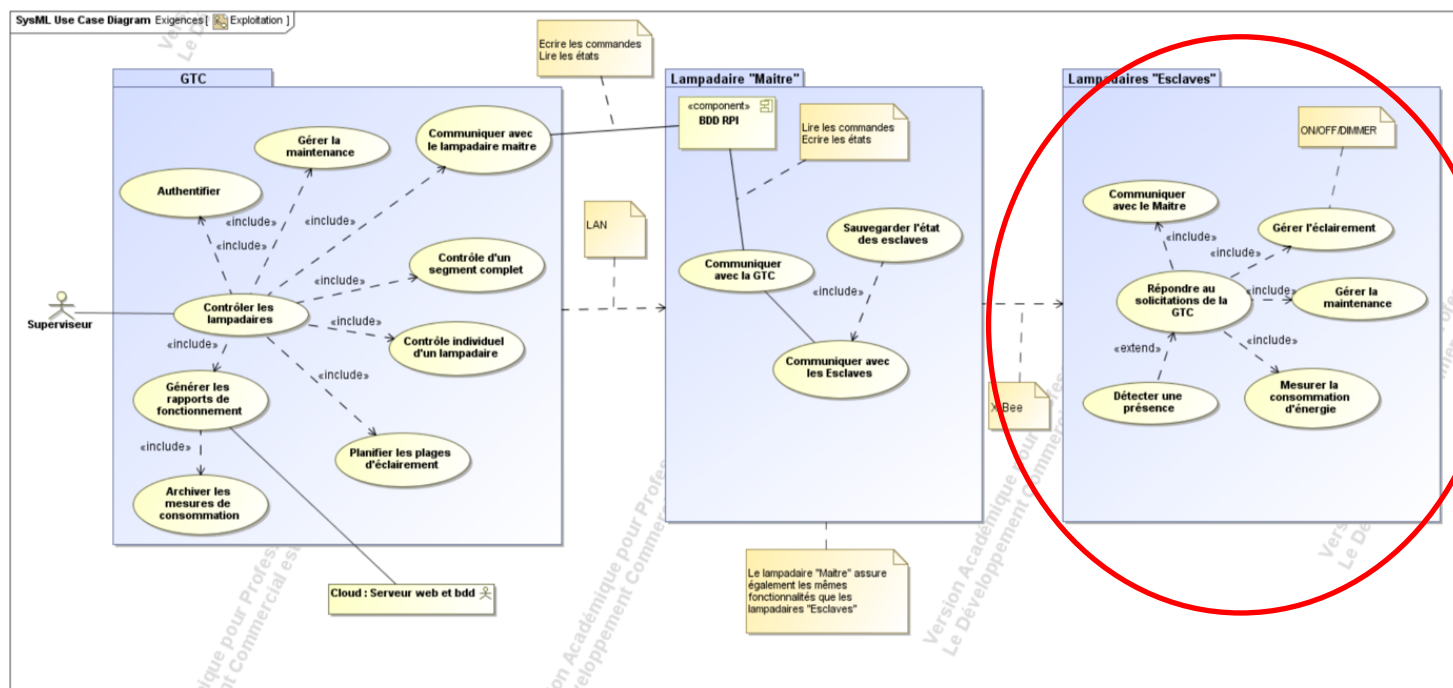


Diagramme de gantt final :

Tâches Mehdi	36,86 jours?	Ven 19/01/1	Ven 22/06/18	MA
Xbee	15 hr	Ven 19/01/1	Ven 26/01/18	7 MA
horloge RTC	15 hr	Ven 26/01/1	Jeu 01/02/18	27 MA
GPS	30 hr	Ven 02/02/1	Jeu 15/03/18	28 MA
Dimmer	20 hr	Jeu 15/03/18	Ven 23/03/18	29 MA
creation du schema structurel	20 hr	Ven 23/03/18	Mer 04/04/18	30 MA
Mesure du courant	25 hr	Mer 04/04/1	Jeu 12/04/18	31 MA
Bouton ON/OFF	20 hr	Jeu 12/04/18	Ven 20/04/18	32 MA
Relais	15 hr	Mer 09/05/1	Ven 11/05/18	33 MA
routage	30 hr	Ven 11/05/1	Jeu 24/05/18	34 MA
Parafoudre	10 hr	Ven 25/05/1	Jeu 31/05/18	35 MA
Alim	10 hr	Jeu 31/05/18	Mer 06/06/18	36 MA
EC1 Phase 1	60 hr	Ven 19/01/1	Lun 12/03/18	
EC1 Phase 2	118 hr	Mer 14/03/1	Jeu 17/05/18	38
EC1 Phase 3	89 hr	Jeu 17/05/18	Ven 22/06/18	39



Cas d'utilisation



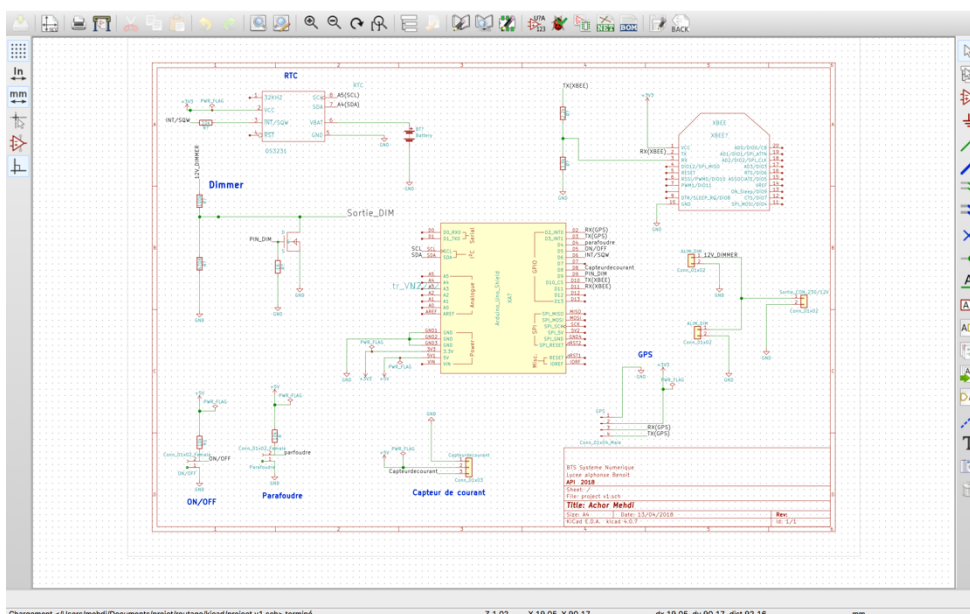
- La partie communication avec le maître sera réalisée avec un module Xbee, directement relié sur le port série de l'Arduino, l'esclave attendra un message du maître sur ce port série pour pouvoir lui répondre sans cela il ne peut pas interagir avec les autres esclaves ou le module maître.
- Le côté détecteur de présence sera interne au module esclave, le capteur devra en permanence s'actualiser pour augmenter la luminosité du lampadaire le plus rapidement possible en cas de nécessité ou au contraire la réduire quand le passage sera nul.
- Pour l'éclairage des lampadaires cela se fera directement avec un bouton ON/OFF installé sur les lampadaires ou à distance avec un relais piloter par l'administrateur depuis son poste de contrôle. Les lampadaires seront aussi équipés d'un dimmer qui sera en relation avec le détecteur de présence pour pouvoir gère un éclairage plus ou moins important selon le besoin.
- La mesure de la consommation d'énergie sera gérée par un capteur qui enverra les mesures effectuées au maître pour qu'elles soient envoyées et traitée dans la BDD.

Logiciel KICAD

Pour la création des schémas structuels et des routages du projet le logiciel kicad a été sélectionné car il offre une licence d'utilisation gratuite ce qui permettra a l'entreprise API de pouvoir reprendre les fichiers dans la continuité du projet sans devoir racheter une licence.

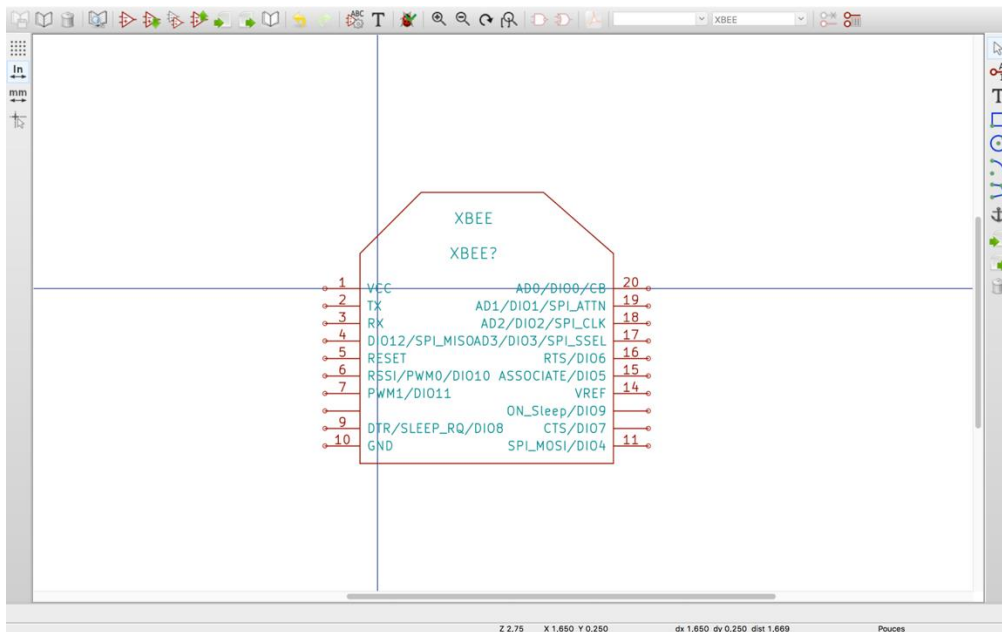
En plus d'être gratuit il est très simple d'utilisation avec des fonctionnalités claire et précises, puis il y a plusieurs tutos sur internet qui apprennent l'utilisation de kicad.

Création du schéma structurel pour le shield arduino : (voir annex 1)



Pour la mise en place sur le schéma du module XBEE j'ai dû crée le composant sur le logiciel,

Pour cela il m'a fallu aller dans le menu « éditeur de librairie – créer et éditer des composants »



Ensuite il faudra placer les premières broches, attribuer un numéro de broche puis lui attribuer le titre d'entrée ou de sortie. Après cela, on trace alors le contour du composant il suffira ensuite d'enregistrer le composant dans la librairie et de l'appeler dans notre projet.

Lien utiliser : Tutoriels (vidéos) sur Kicad de Eric Peronnin (professeur IUT Nantes) - <https://www.youtube.com/channel/UCe3v5cVACw-5BKQOcwUaM8w/videos>



Mise en place des composants

1. Le Module XBEE.

La mise en place du module XBEE est très importante car elle nous permette de communiquer entre le module maitre et le module esclave.

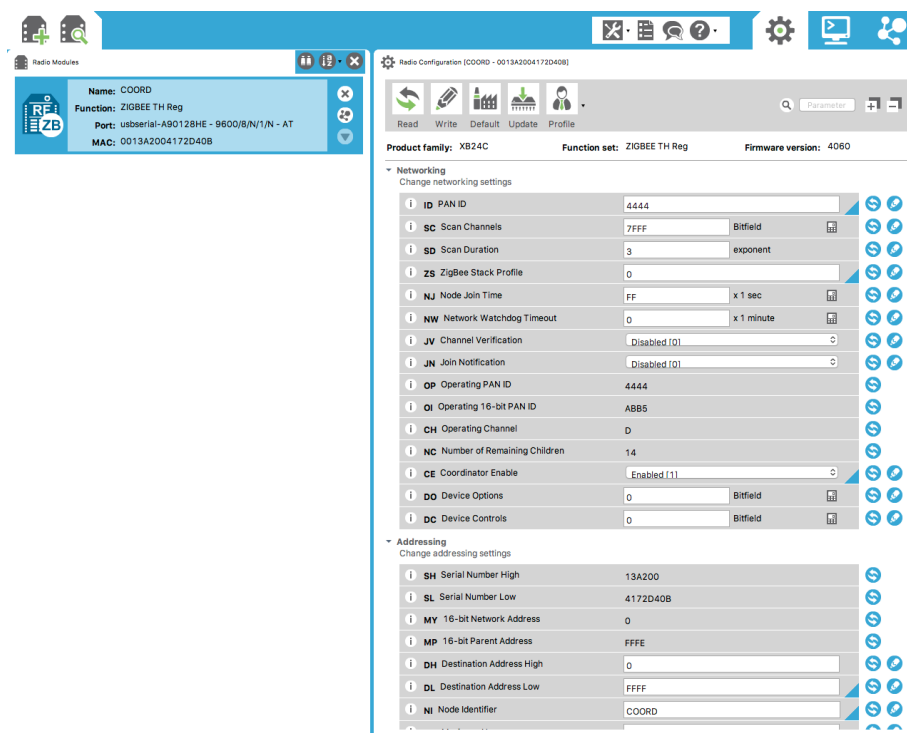
Norme : IEEE802.15.4

Faible consommation d'énergie : Cycles d'émissions/réceptions et connexions au réseau très rapides

Pour la mise en place il m'a fallu connecter le module XBEE qui utilise la communication série, sur un convertisseur Série/USB :



Ensuite j'ai dû utiliser le logiciel XCTU, un petit software qui me permettra de faire les premières configurations de mon module :



Après avoir lancée la recherche, le logiciel trouve le module, je le sélectionne puis je le configure manuellement en mode « ROUTER » :

Param	XBeeA	XBeeB	XBeeC
ID	5555	5555	5555
JV	[0]	Enable [1]	Enable [1]
CE	Enable [1]	[0]	[0]
DH	0	0	0
DL	FFFF	0	0
NI	COORD	ROUTER1	ROUTER2

ID : L'ID doit être le même pour tout les modules Xbee que l'on souhaite faire communiquer entre eux.

JV et CE : Ces champs permettent de définir si le module Xbee sera un esclave ou un maître.

DL : DL Concerne uniquement le maître, c'est l'adresse sur la quel le module Xbee va émettre. Il faut fixer cette valeur à « FFFF », ce qui correspond l'adresse de diffusion.

NI : C'est un identifiant fixé par l'utilisateur qui lui permet de se repérer.

Après avoir effectué la configuration appropriée, un premier test de communication est effectué avec le logiciel arduino :

```

/dev/cu.usbmodem14331 (Arduino/Genuino Uno)

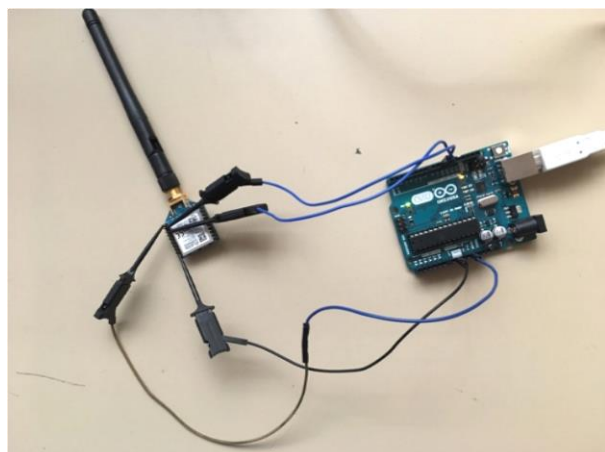
Goodnight moon!
1;ON;0;CRC
1;ON;0;CRC
1;ON;0;CRC
1;ON;0;CRC

```

☒ Défilement automatique
 Pas de fin de ligne
 9600 baud

On reçoit bien la trame envoyée par le module maitre.

Le test de communication est donc positif.



Le câblage utilise les broches suivantes :

Tx(Xbee) → Rx(Arduino)

3,3v (Arduino) → Vcc(Xbee)

GND(Xbee)→ GND(arduino)

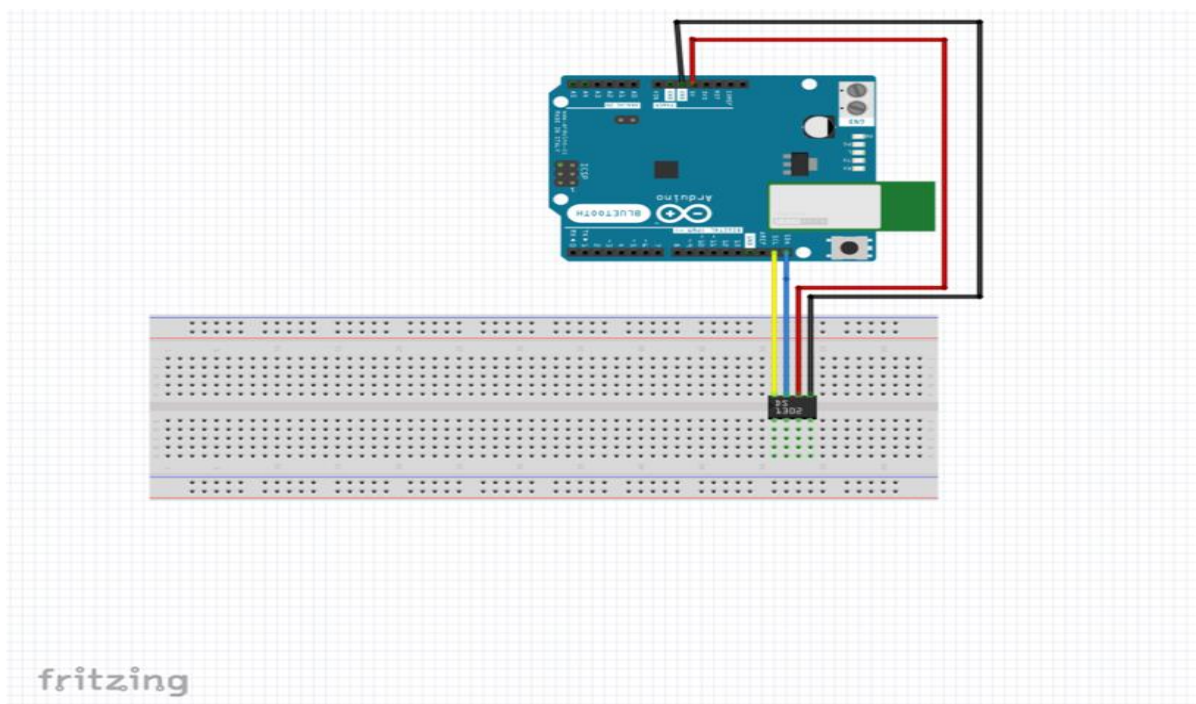
2. Le module RTC.

Le module RTC qui est le DS3231MZ + cadencera le lampadaire en cas de panne du maître car le module sera automatiquement mis à jour par la RTC du maître afin que tout le parc lampadaire est la même heure pour s'allumer ou s'éteindre. L'horloge utilise les ports SDA (Serial Data Line) et SCL (Serial Clock Line).

Température de fonctionnement : -40 ° C à + 85 ° C

Tension d'alimentation de + 2.3V à + 5.5V

Schéma de câblage rapide fritzing :



Lien fritzing : <http://fritzing.org/home/>

Pour tester le module RTC j'ai utilisé un Code arduino qui me permet de rentré en dure l'heure et la date pour cadencer l'horloge : ANNEXE 2

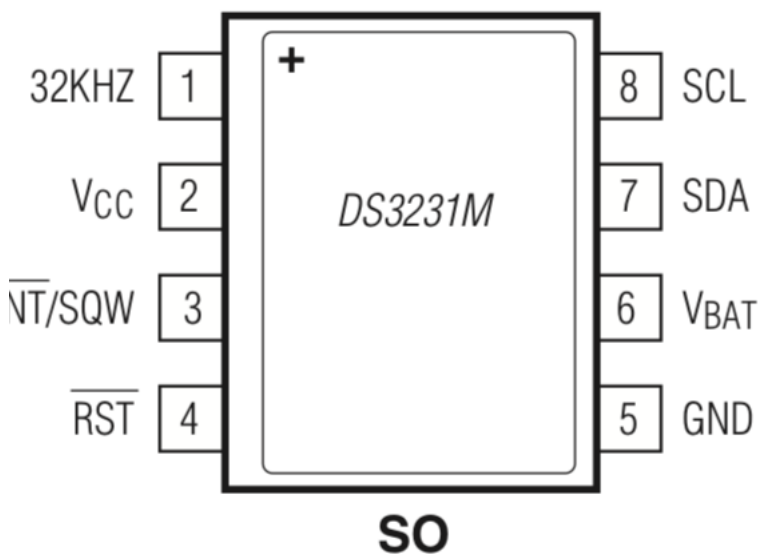


```

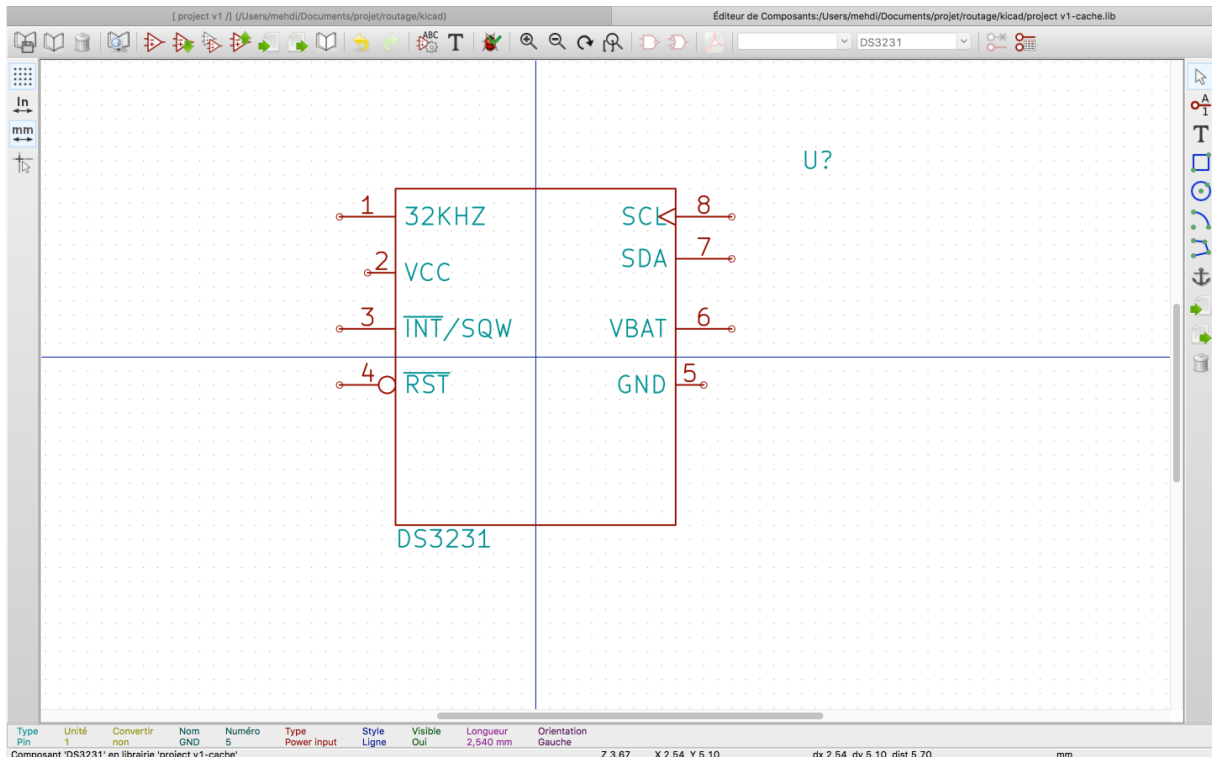
horloge_test
1 #include <Wire.h>
2 #include <ds3231.h>
3
4 struct ts t; //déclaration variable t
5
6 void setup() {
7   Serial.begin(9600);
8   Wire.begin();
9   DS3231_init(DS3231_INTCN);
10
11   t.hour=16; // données pour mettre à l'heure l'horloge
12   t.min=22;
13   t.sec=0;
14   t.mday=01;
15   t.mon=02;
16   t.year=2018;
17
18   DS3231_set(t); // mise à l'heure de l'horloge
19 }
20
21 void loop() {
22   // put your main code here, to run repeatedly:
23   DS3231_get(&t);
24   Serial.print("date : ");
25   Serial.print(t.mday);
26   Serial.print("/");
27   Serial.print(t.mon);
28   Serial.print("/");
29   Serial.print(t.year);
30   Serial.print("\t Heure : ");
31   Serial.print(t.hour);
32   Serial.print(":");
33   Serial.print(t.min);
34   Serial.print(".");
35   Serial.println(t.sec);
36
37   delay(500);
38 }
39
40
41 //A1 /A2 /A3 /1 R

```

Pour pouvoir identifier les différentes broches du module j'ai dû me référer à la documentation :



Je n'utiliserai pas les broches 1 et 4 car elles n'auront pas d'utilité, pour pouvoir intégrer la RTC dans mon schéma structurel j'ai dû créer mon composant car la DS3231 mhz+ n'était pas disponible dans les bibliothèques du logiciel kicad :



3. Le GPS

(Suite à la réunion avec l'entreprise API et modification du cahier des charge du 19/04/18 le module gps ne figurera plus sur les lampadaire esclave)

Module gps GROVE, Ce module GPS économique compatible Grove permet de connaître la position en temps réel. Il communique avec le microcontrôleur via un port série.



Sensibilité: -160 dBm
Consommation: 40 mA sous 3 Vcc
Vitesse de transmission: 4800 à 57600 bauds

Le Gps sera mis en place sur chaque module esclave, ils renverront leur position à l'administrateur. Le but est de créer une carte avec tous les lampadaires actifs avec un indicateur de disfonctionnement, en cas de panne le lampadaire affichera un voyant rouge qui pourra solliciter une équipe de maintenance.

Pour la mise en place et le test du module gps le code utilisé est le suivant :

ANNEXE 3

```
GPS $
1 #include <SoftwareSerial.h>
2 SoftwareSerial SoftSerial(2, 3); // rx , tx
3 unsigned char buffer[64];
4 int count=0;
5 void setup()
6 {
7     SoftSerial.begin(9600);
8     Serial.begin(9600);
9 }
10
11 void loop()
12 {
13     if (SoftSerial.available())
14     {
15         while(SoftSerial.available())
16         {
17             buffer[count++]=SoftSerial.read();
18             if(count == 64)break;
19         }
20         Serial.write(buffer,count);
21         clearBufferArray();
22         count = 0;
23     }
24     if (Serial.available())
25         SoftSerial.write(Serial.read());
26 }
27
28 void clearBufferArray()
29 {
30     for (int i=0; i<count;i++)
31     {
32         buffer[i]=NULL;
33     }
34 }
35 }
36
37
```

Le module récupère les données de position puis les stock dans un tableau de 64 caractères, une fois le tableau plein il l'envoie sur le port série de l'arduino et efface le tableau.

Communication

Pour la communication un type de trame a été mis en place par le groupe, la trame nous servira à échanger les données ou les commandes de façon universelle entre tous les esclaves et maitres.

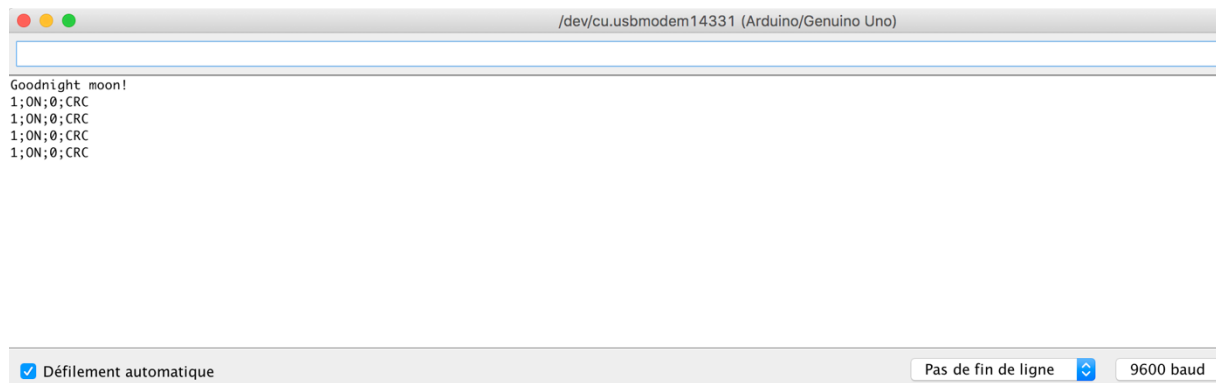
Model de la trame =

-Maitre vers esclave : num_esclave;horodatage;coord_gps;commande;data;crc\n

- Esclave vers maitre : 0;horodatage;coord_gps;etat;crc\n

Comme séparateur de champ de données le caractère « ; » sera utilisé.

Exemple de trame reçue par l'esclave :



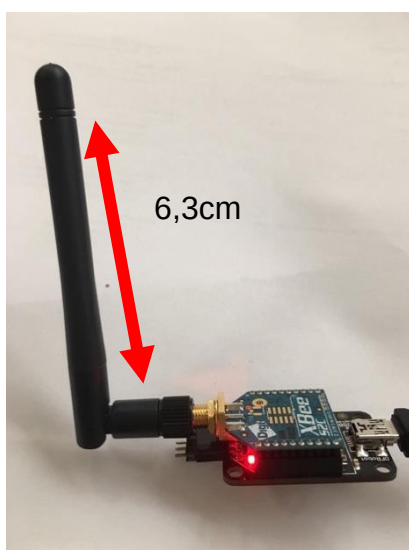
Pour la communication entre les modules les antennes sont indispensables car la distance entre les lampadaires doivent être de 3 fois leurs hauteur exemple pour un lampadaire de 10m son espacement d'un autre lampadaire doit être de 30m.

Sachant que le module xbee a une portée de 60m sans l'antenne, a la moindre atténuation du signal ou le disfonctionnement d'un module XBEE la perte de signal empêcherait la communication.

Nous utilisons donc une antenne demis onde : $\frac{3 \cdot 10^8}{2,4 \cdot 10^9} = 0,125\text{m} \rightarrow 12,5 \text{ cm}$

$\frac{12,5}{2} = 6,25 \text{ cm}$ cela correspond à la longueur de notre antenne.

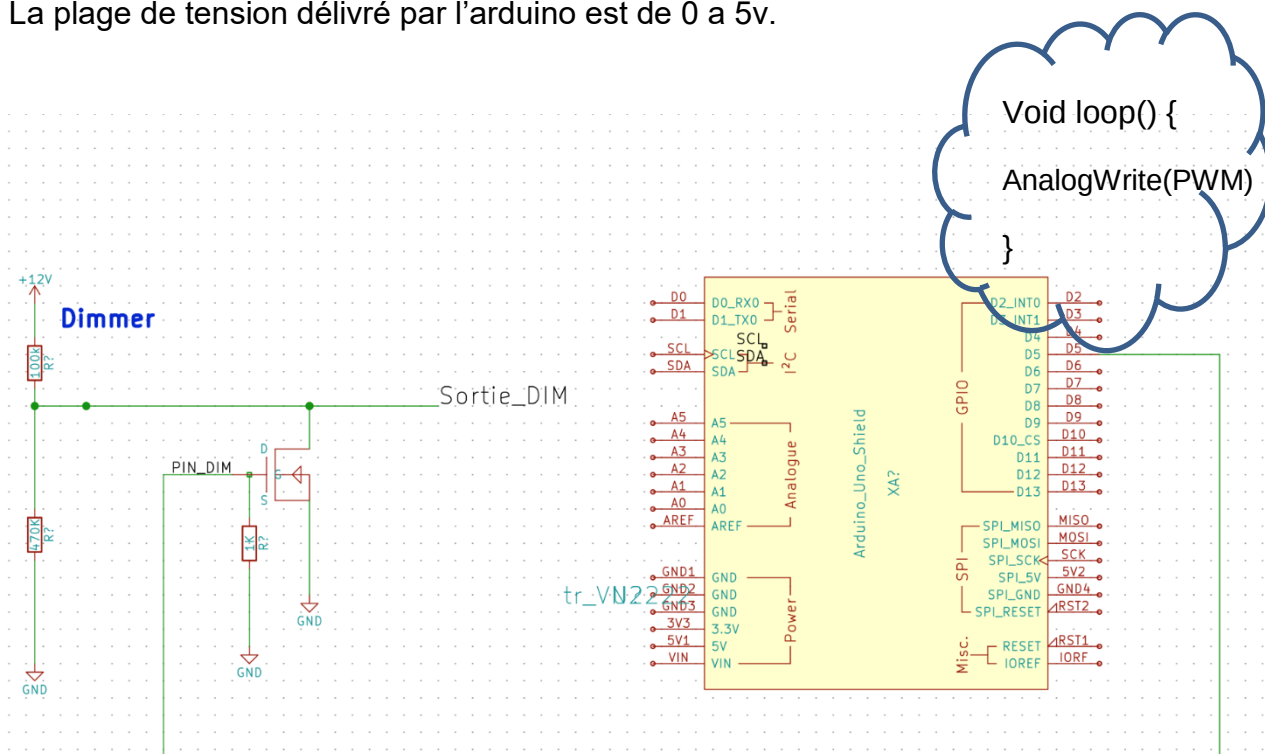
2,4 GB correspond à la fréquence de notre module XBEE.



Traitement de la partie dimmer

Pour varier la luminosité de la tête du lampadaire, un dimmer a été mis en place. Gérée par la carte arduino piloter par le maître, il augmentera ou diminuera la luminosité piloter par une sortie PWM de l'arduino (la broche 5) , la broche PWM ou modulation de largeur d'impulsion utilisera la commande « analogWrite » pour utiliser une valeur de 0 a 255.

La plage de tension délivré par l'arduino est de 0 a 5v.



Pour le fonctionnement le transistor VN2222 a été utiliser. Le transistor est de type : mos canal N. Pour pouvoir faire variée la tension au borne du dimmer, on joue sur le rapport cyclique de l'arduino. Sur ce montage avec les caractéristiques de notre transistor, plus le rapport cyclique serra élever plus la luminosité serra faible et plus le rapport serra faible plus la luminosité serra élever. Plus précisément au borde du transistor cela fonctionne :

Si $V_{GS} > V_{th}$ alors on peut dire que cela se comporte comme un circuit ferme.

Si $V_{GS} < \text{ou} = 0$ alors on peut dire que le circuit est ouvert.

Plus le circuit est « fermé » (se comporte comme une résistance faible) plus le rapport cyclique est faible.

Plus le circuit est « ouvert » (se comporte comme une résistance élevée) plus le rapport cyclique est élevé.

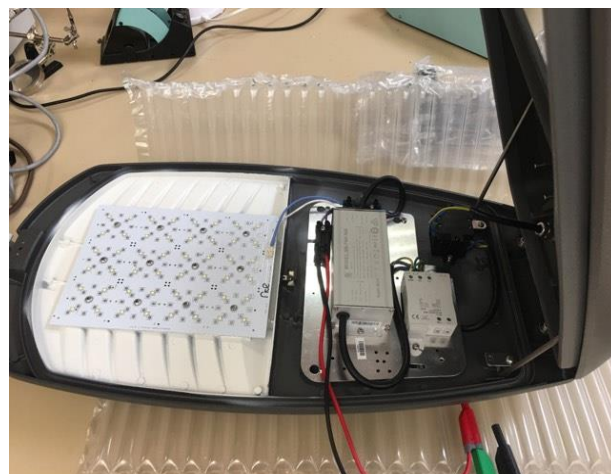
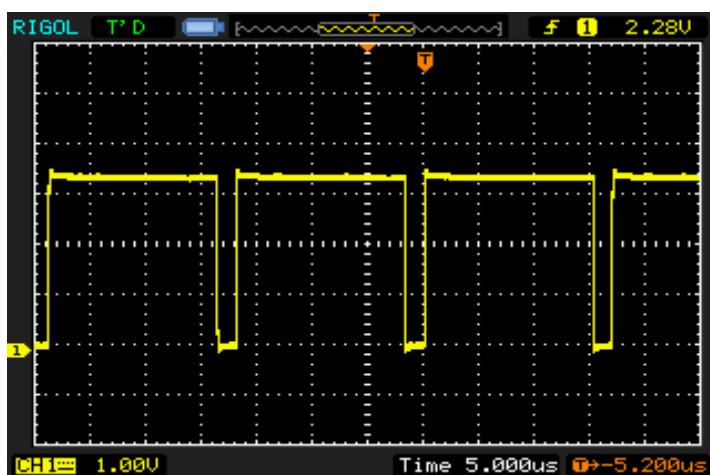
La tension attendu par la tête de lampadaire est de 0 à 10V alors j'utilise une alimentation externe en 12V qui passera par un pont diviseur de tension pour être atténuer en 10V.

Le pont diviseur de tension utiliser se dimensionne par une résistance de 100K et

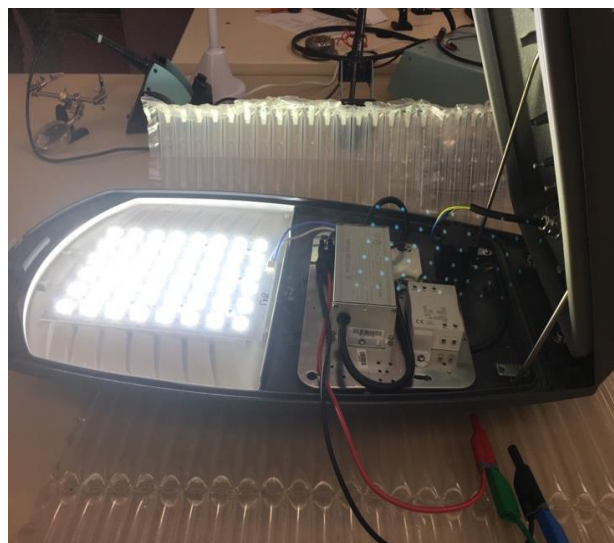
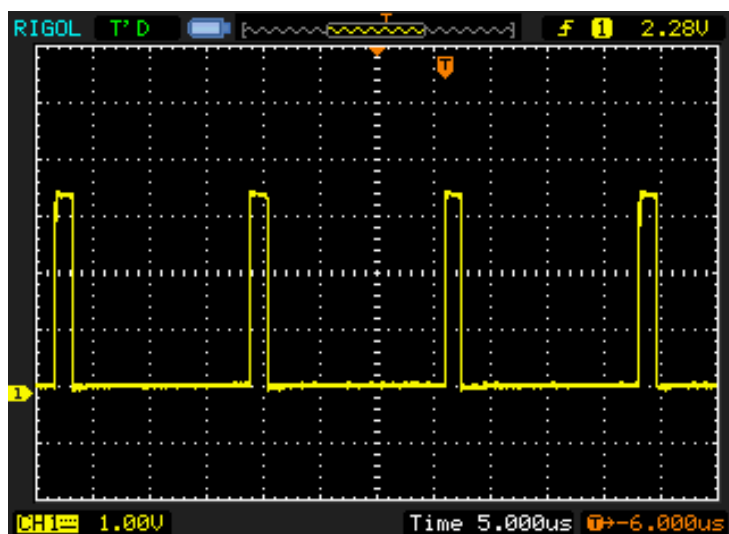
une autre de 470k :
$$\frac{12 \cdot (470 \cdot 10^3)}{470 \cdot 10^3 + 100 \cdot 10^3} = 9,895 \text{ V.}$$

Exemple du rapport cyclique pour les deux extrême 0 et 100% de luminosité :

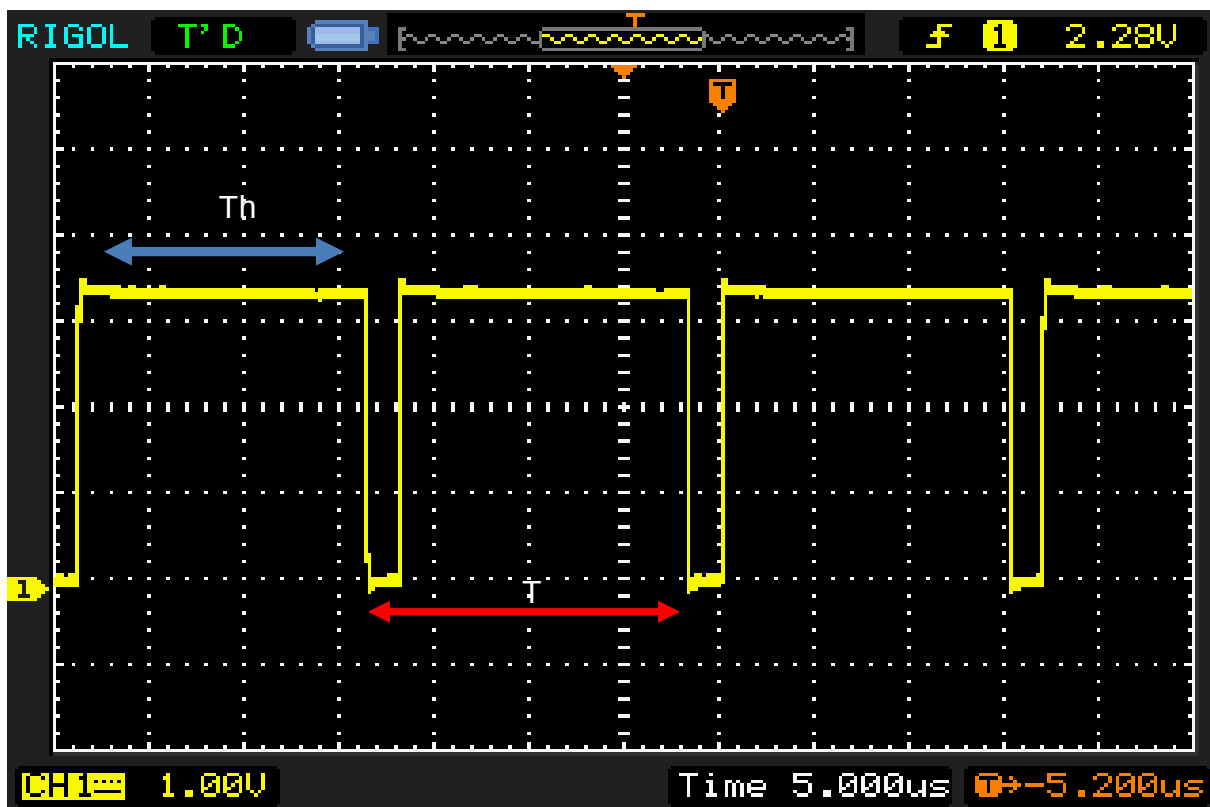
Pour un rapport cyclique élevé :



Pour un rapport cyclique faible :



Pour avoir la valeur de notre rapport cyclique on utilise la formule suivante : $\alpha = \frac{T_h}{T}$



Puis après avoir calculer le rapport cyclique on va pouvoir calculer la valeur moyenne de notre signal, La valeur s'exprime en pourcentage. Pour calculer la valeur moyenne de notre signal faut faire un calcul d'air :

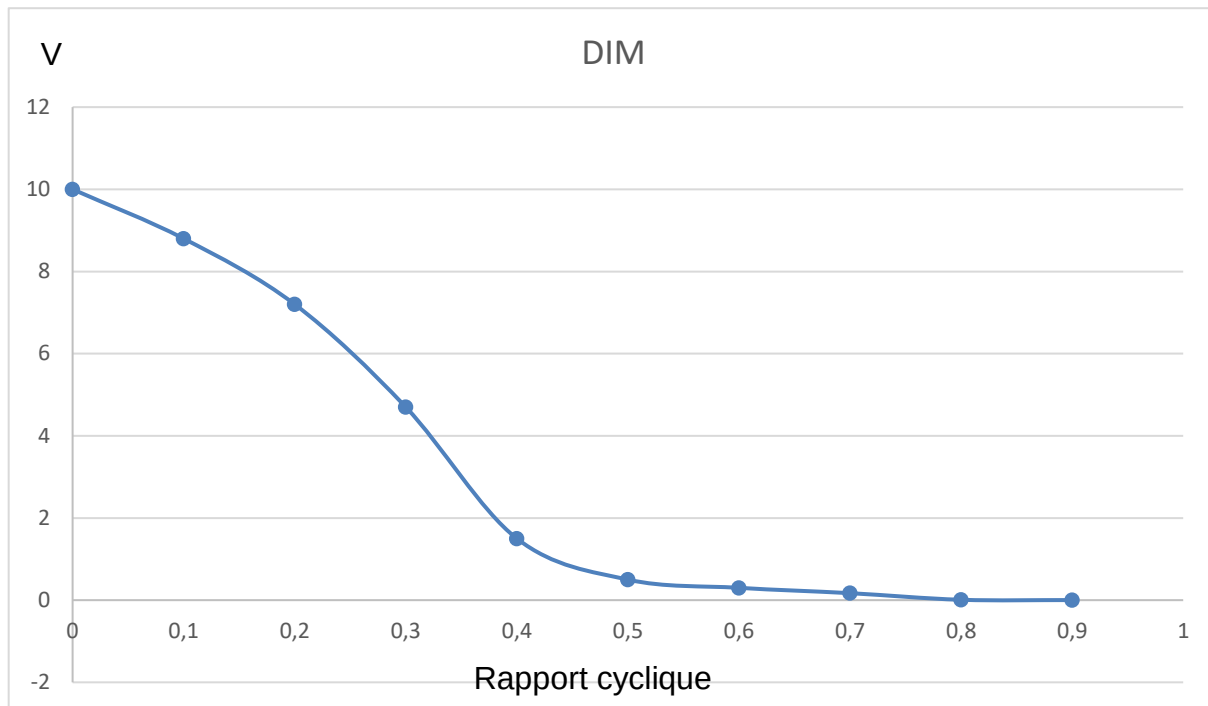
$$V_{moy} = \frac{1}{T} * 5 * T_h = 5 \frac{T_h}{T}$$

$$V_{moy} = \alpha * 5V.$$

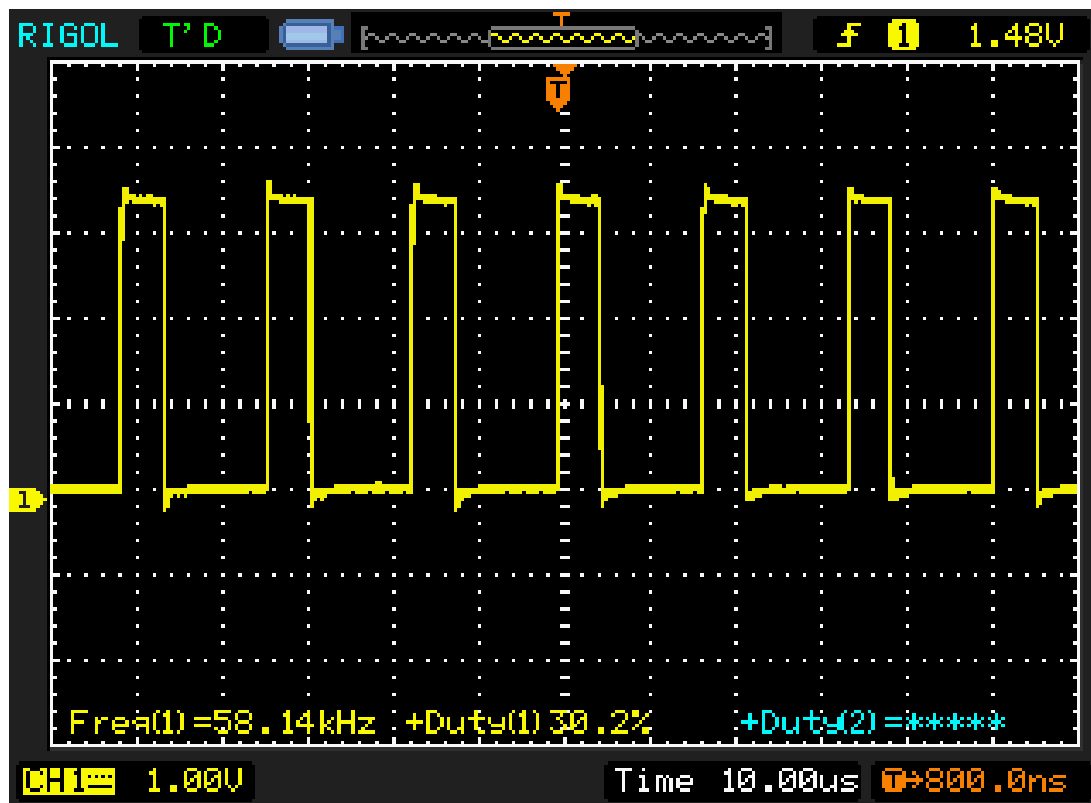
Avec la valeur moyenne on va pouvoir calculer la tension envoyer au dimmer. Pour cela j'ai dû mesurer à l'oscilloscope les signaux émis par l'arduino puis ceux sur le dimer après le passage dans le transistor. J'en ai déduit le tableau suivant :

DIMMER	Vmoy
0V	5V
10V	0

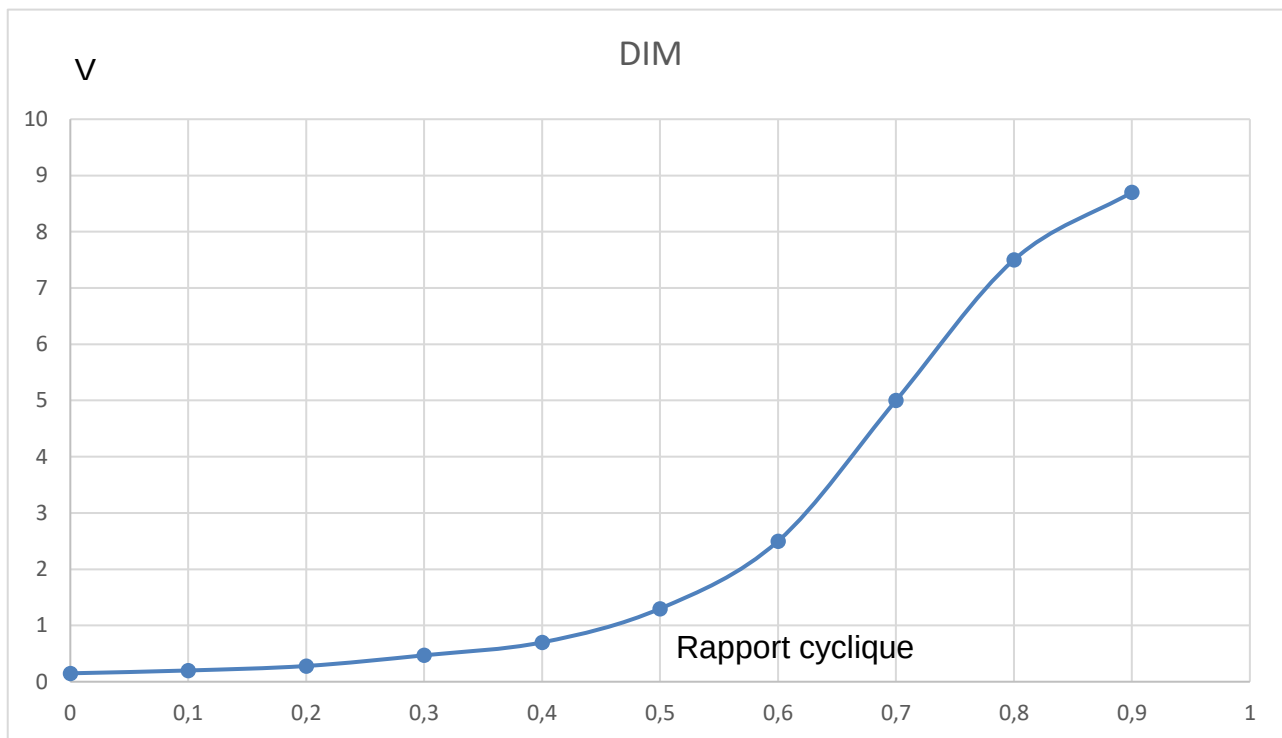
Voici la courbe du rapport cyclique face à l'intensité envoyer avant le passage dans le transistor :



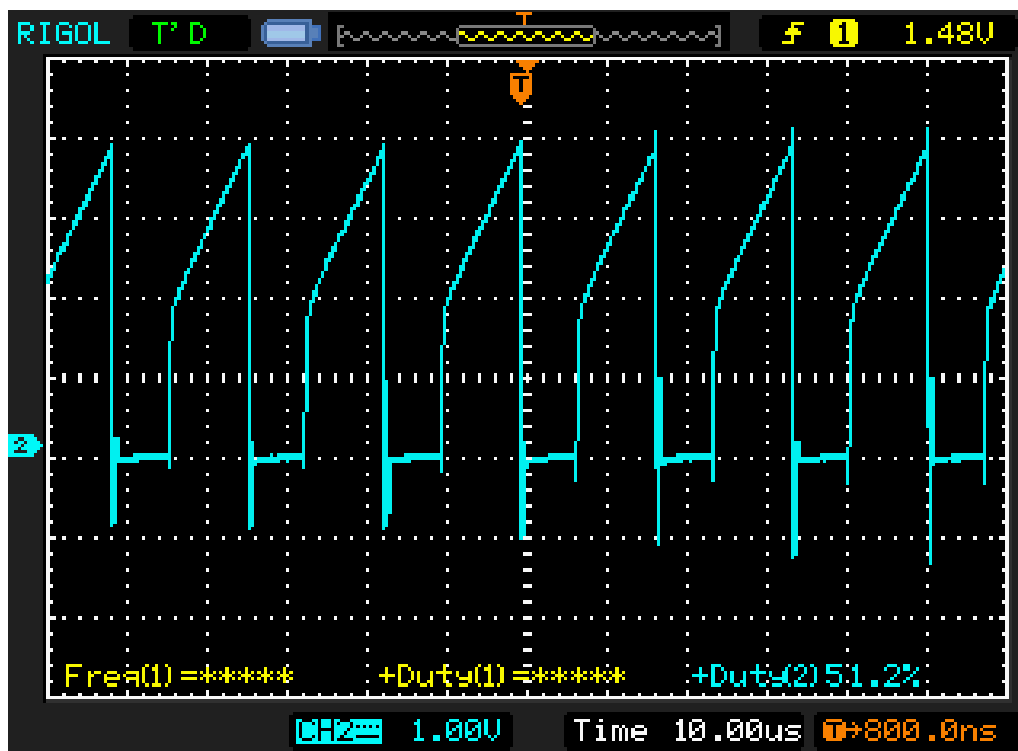
On peut voir cette courbe pour 6V, 30% du rapport cyclique :



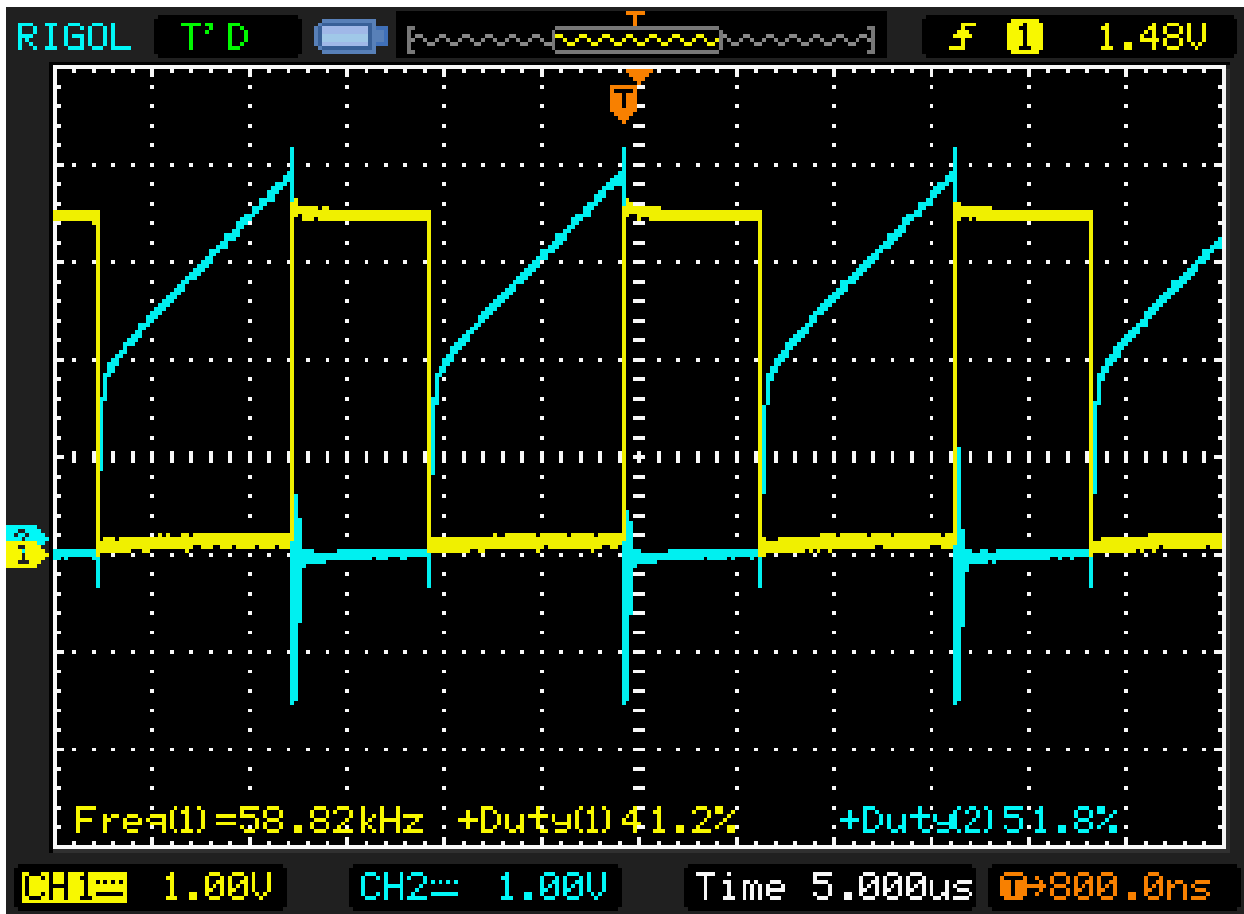
Voici la courbe du rapport cyclique face à l'intensité envoyer avant le passage dans le transistor :



On peut trouver pour 1,5V, 51% du rapport cyclique :



On retrouve bien l'inverse du signal sur la broche du dimmer , on peut le confirmer avec l'oscilloscope :



Le signal en jaune représente le rapport citrique avant le passage sur l'arduino puis le signal bleu représente le signal après le passage dans le transistor on remarque alors qu'un état haut pour l'un correspond à un état bas pour l'autre.

Pour calculer la valeur envoyée sur le dimmer on effectue le calcul suivant :

$$V_{dim} = -2 * V_{moy} + 10$$

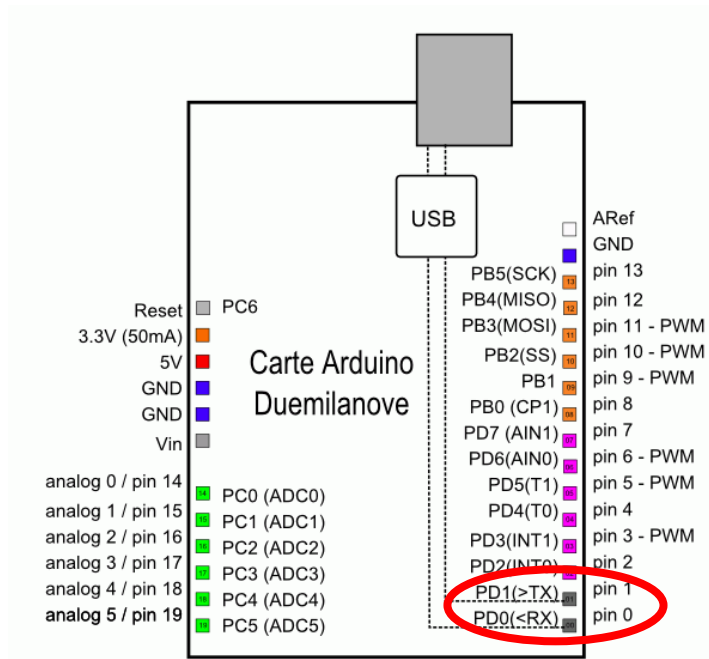
$$V_{dim} = -10 * \alpha + 10$$

La valeur d'alpha est comprise entre [0 ;1]

On peut constater que la courbe devient linéaire à partir de 0,4.

Problèmes et solutions

Le premier problème que j'ai pu rencontrer a été l'utilisation de plusieurs ports série par les nombreux capteurs alors que l'arduino utilisée en avait qu'un de disponible.



La solution retenue a été de créer des ports série virtuel dans le code arduino afin de pouvoir les faire communiquer dans une bonne entente :

Exemple pour le module GPS :

```
SoftwareSerial SoftSerial(2, 3); // rx , tx gps
```

Exemple pour le module XBEE :

```
SoftwareSerial xbee(10, 11); // 10-->TX ,11---> RX
```

Le second problème lié avec les ports série a été l'ouverture et la fermeture des ports car utilisent plusieurs port série qui sont « virtuel » sur l'arduino (pour le GPS, XBEE) il va falloir ouvrir le port dédié l'or de l'utilisation du capteur puis le fermer à la fin de la communication sous peine de ne pas pouvoir utiliser le prochain capteur.

Exemple pour le gps :

`SoftSerial.begin(9600);` → Ouverture du port série

```
//if (SoftSerial.available()) //
while(! SoftSerial.available() && SoftSerial.read()
  Serial.println("GPS waiting...");
}

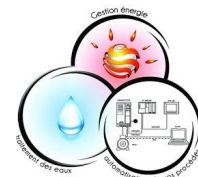
//{
while (SoftSerial.available()) // read
{
  buffer[count++] = SoftSerial.read(); // writin
  //xbee.print(SoftSerial.read());
  Serial.println(SoftSerial.read());
  if ((count == 64) || (buffer[count-1] == '*')) br
  delay(10);
}
```

`SoftSerial.end();` → Fermeture du port série

Amélioration

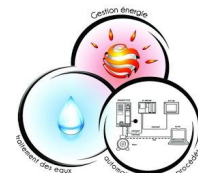
Pour l'avancement du projet, les futures améliorations pourront être apporter :

- Utilisation du panneau solaire pour alimenter l'arduino.
- Utilisation d'une arduino mini pour un gain de place.



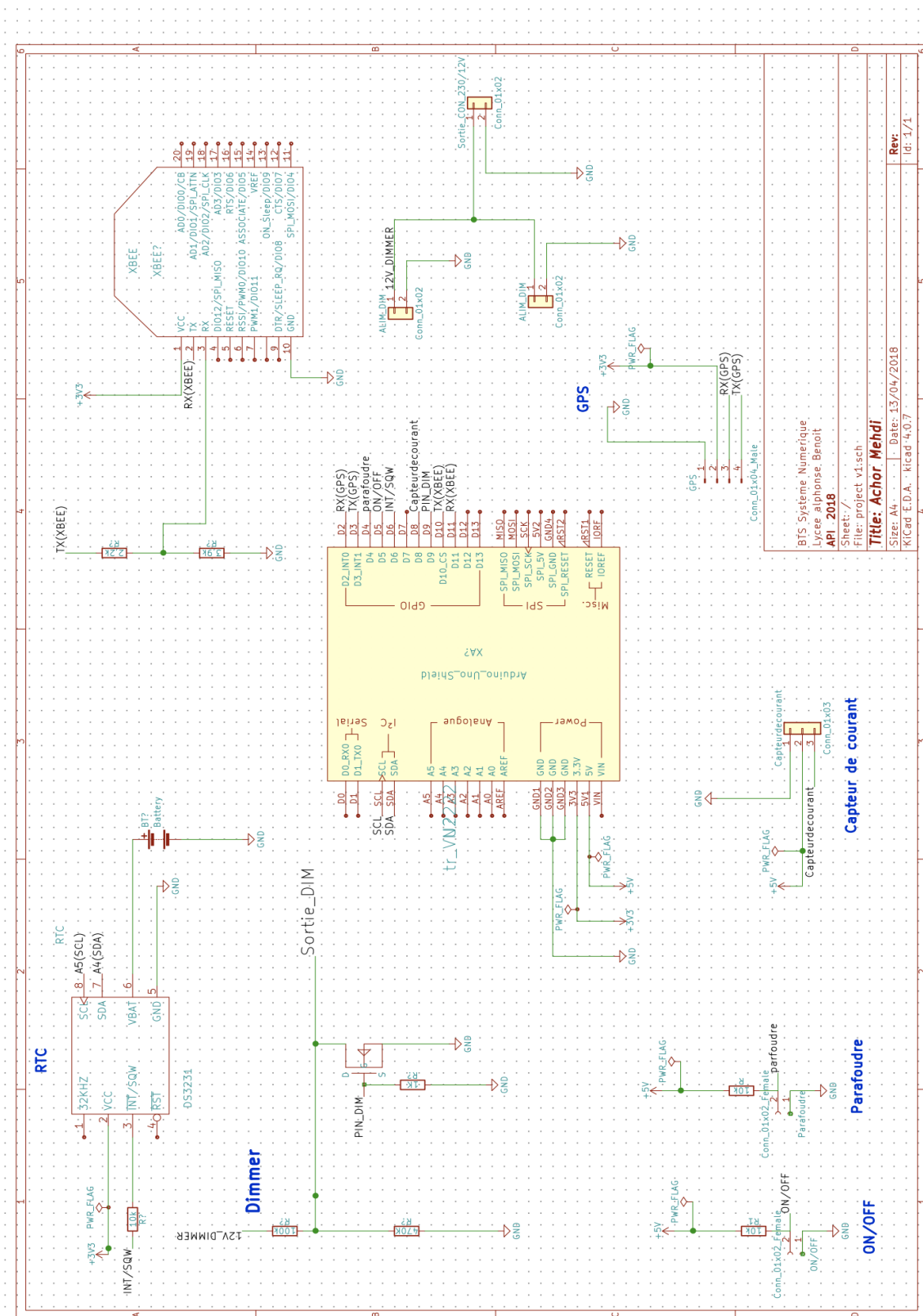
Journal de bord personnel

JOURS	ACTIVITEES
jeudi 19 janvier 2018	Mise en place des modules XBEE : → Depuis le logiciel XCTU → Depuis le terminal
vendredi 20 janvier 2018	Test de connexion depuis le terminal arduino avec création d'un code simple. Ne marche pas.
mercredi 24 janvier 2018	Bonne démarche trouver , se connecter avec des câbles sur l'arduino avec un port virtuel. (Programme dans le dossier arduino). → le test est positif Branchement d'un 3eme modules Xbee. Avec un analyseur logique, branchement sur les broches TX et RX pour percevoir les signaux envoyer.
Jeudi 25 janv 2018	Tentative de communication arduino/ raspberry avec les module Xbee , d'abord avec putty/arduino puis création du code raspberry. Analyse du schéma de la carte horloge. Je ressoude le câble VCC KC. Pour le programme, utilisation de la librairie : Wire.h Utilisation d'une horloge : Modèle : DS3231 MZ + Utilisation des broche SDA, SCL. Alimentation sous 5V (de 2,3V a 5,5V).
vendredi 26 janvier 2018	Analyse du code pour l'horloge.
Mercredi 31 janvier 2018	Mise en place du format de la trame à envoyer. Création du code d'allumage de la led depuis la raspberry. OK.
jeudi 1er février 2018	Mises-en place de l'horloge dans le programme. Création et rajout du code de l'interface graphique QT.



mercredi 21 février 2018	Le code a été rajouter au code de base de communication XBEE. Problème dans la partie void.
Jeudi 22 février 2018	Résolution du problème puis mise en place du code GPS. Problème de compatibilité des codes.
Vendredi 23 février 2018	Problème résolue , tous les composent présent fonctionne.
Mercredi 14 mars 2018	Préparation et mise en place du dossier.
Jeudi 15 mars 2018	Préparation et mise en place du dossier.
Vendredi 16 mars 2018	Préparation et mise en place du dossier.
Mercredi 21 mars 2018	Première revu de projet
Jeudi 22 mars 2018	Recherche fonctionnement du dimer
Vendredi 23 mars 2018	Première tentative code dimer « échec »
Mercredi 28 mars 2018	Mise en place d'un code dimer qui marche
Jeudi 29 mars 2018	Création du schéma structurel
Vendredi 30 mars 2018	Travaille sur le schéma structurel
Mercredi 4 avril 2018	Travaille sur le schéma structurel(xbee)
Jeudi 5 avril 2018	Travaille sur le schéma structurel(horloge)
Vendredi 6 avril 2018	Travaille sur le schéma structurel(GPS)
Mercredi 11 avril 2018	Préparation rencontre API
Jeudi 12 avril 2018	Préparation rencontre API
Vendredi 13 avril 2018	Préparation rencontre API
Mercredi 18 avril 2018	Rencontre avec l'entreprise API
Mercredi 9 mai 2018	Finition de l'assemblage des codes
Jeudi 10 mai 2018	Travaille sur le schéma structurel (dimer)
Vendredi 11 mai 2018	Travaille sur le schéma structurel(dimer)
Mercredi 16 mai 2018	Réception de la tête de lampadaire
Jeudi 17 mai 2018	Préparation revue de projet 2
Vendredi 18 mai 2018	Préparation revue de projet 2
Mardi 22 mai 2018	Préparation revue de projet 2
Mercredi 23 mai 2018	Presentation revu 2
Jeudi 24 mai 2018	Test dimer sur la tête de lampadaire
Vendredi 25 mai 2018	Preparation dossier final

ANNEXE 1 :



ANNEXE 2 :

```

horloge_test
1 #include <Wire.h>
2 #include <ds3231.h>
3
4 struct ts t; //déclaration variable t
5
6 void setup() {
7     Serial.begin(9600);
8     Wire.begin();
9     DS3231_init(DS3231_INTCN);
10
11     t.hour=16; // données pour mettre à l'heure l'horloge
12     t.min=22;
13     t.sec=0;
14     t.mday=01;
15     t.mon=02;
16     t.year=2018;
17
18     DS3231_set(t); // mise à l'heure de l'horloge
19 }
20
21 void loop() {
22     // put your main code here, to run repeatedly:
23     DS3231_get(&t);
24     Serial.print("date : ");
25     Serial.print(t.mday);
26     Serial.print("/");
27     Serial.print(t.mon);
28     Serial.print("/");
29     Serial.print(t.year);
30     Serial.print("\t Heure : ");
31     Serial.print(t.hour);
32     Serial.print(":");
33     Serial.print(t.min);
34     Serial.print(".");
35     Serial.println(t.sec);
36
37     delay(500);
38 }
39
40
41 //01/02/18

```

ANNEXE 3

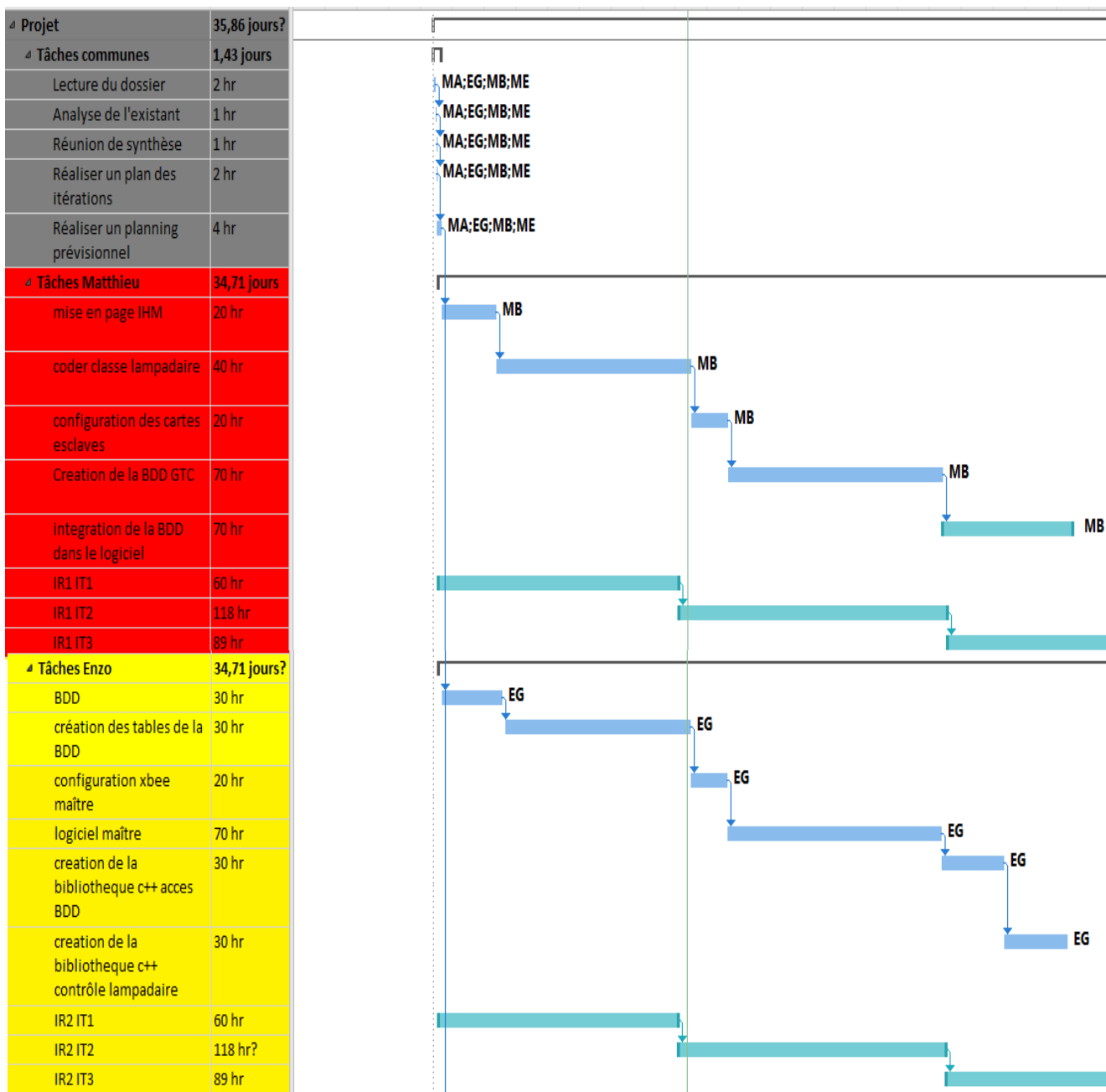
```

GPS §
1 #include <SoftwareSerial.h>
2 SoftwareSerial SoftSerial(2, 3); // rx , tx
3 unsigned char buffer[64];
4 int count=0;
5 void setup()
6 {
7     SoftSerial.begin(9600);
8     Serial.begin(9600);
9 }
10
11 void loop()
12 {
13     if (SoftSerial.available())
14     {
15         while(SoftSerial.available())
16         {
17             buffer[count++]=SoftSerial.read();
18             if(count == 64)break;
19         }
20         Serial.write(buffer, count);
21         clearBufferArray();
22         count = 0;
23     }
24     if (Serial.available())
25         SoftSerial.write(Serial.read());
26 }
27
28
29 void clearBufferArray()
30 {
31     for (int i=0; i<count;i++)
32     {
33         buffer[i]=NULL;
34     }
35 }
36
37

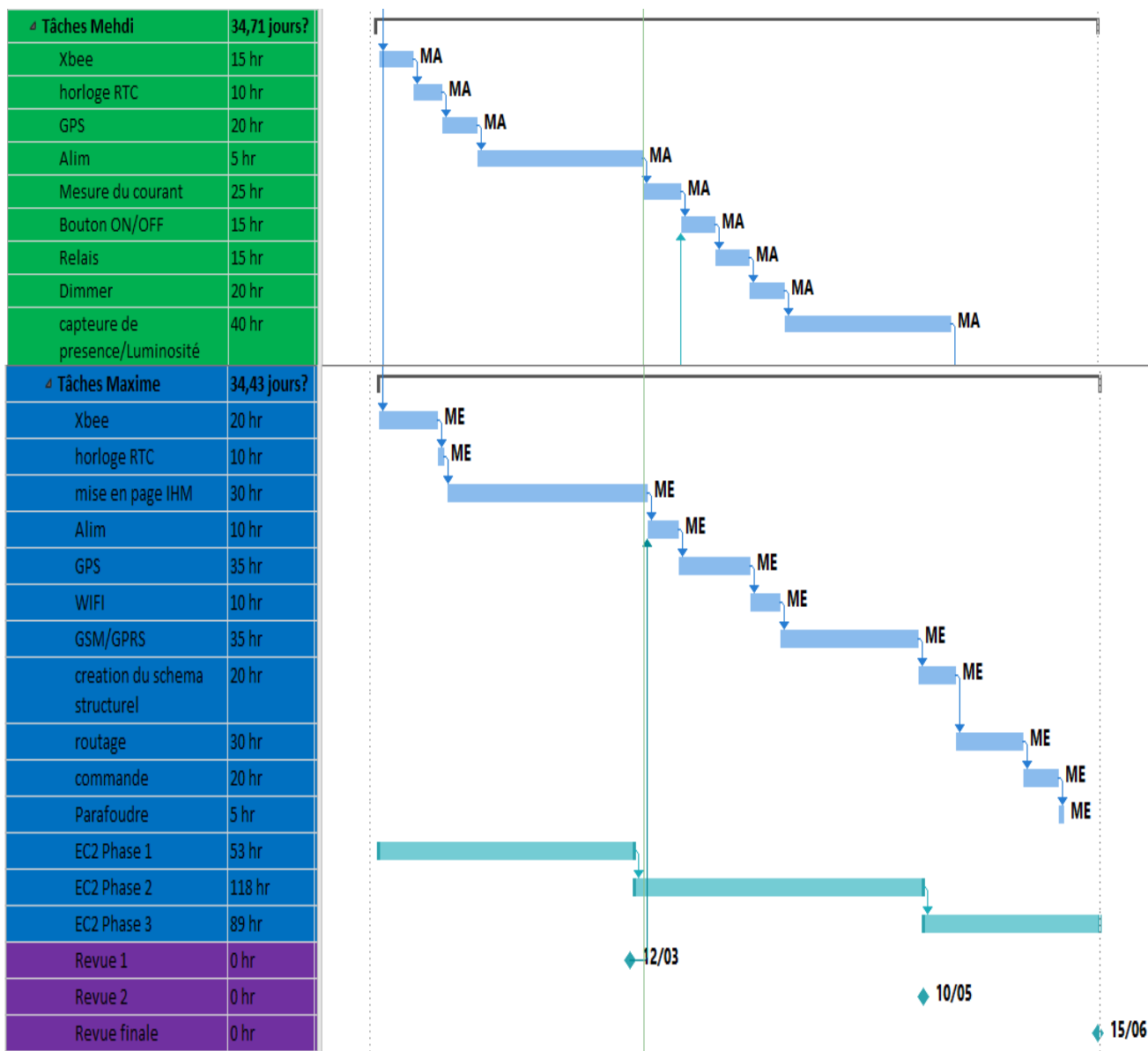
```

Annexe commun :

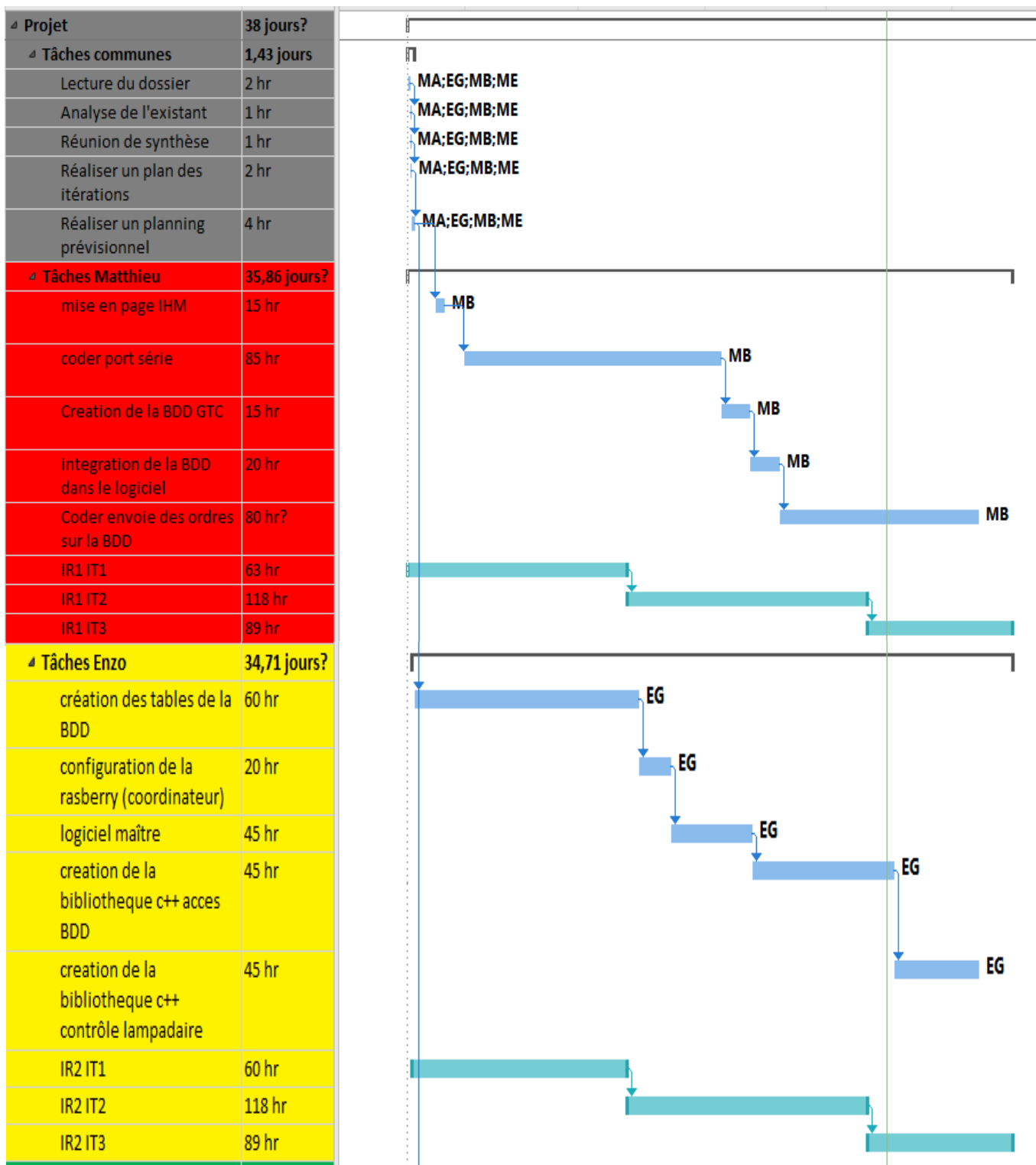
Gantt provisoire

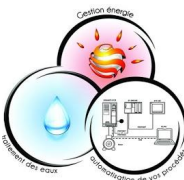


Gantt provisoire



Gantt final

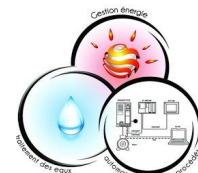




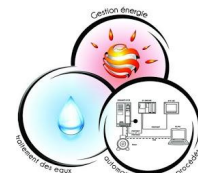
The Gantt chart displays the schedule for the 'MA' (Maintenance Activity) and 'ME' (Maintenance Event) tasks. The chart is divided into two sections by a vertical green line. The top section shows the 'MA' tasks, which are represented by blue bars and arrows. The bottom section shows the 'ME' tasks, represented by blue bars and arrows. A horizontal green line separates the two sections. The chart includes a timeline at the bottom with dates 12/03, 10/05, and 15/06 marked by diamond symbols.



Fiche de réunion		Réf.FMC-XX
Projet	Réalisateur (s)	Date
GTC	Achor, Brochard, Eeckeman, Garagnoli	19/01/2018
Réunion		
1		
Commentaire		
Première réunion : - Lecture du cahier des charges - Répartition des taches - Création du planning prévisionnel		



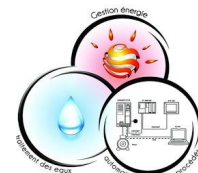
Fiche de réunion		Réf.FMC-XX
Projet	Réalisateur (s)	Date
GTC	Achor, Brochard, Eeckeman, Garagnoli	31/01/2018
Réunion		
2		
Commentaire		
Réunion module XBEE : - Visionnage du diaporama de présentation des modules XBEE - Présentation du logiciel XCTU		



Fiche de réunion		Réf.FMC-XX
<i>Projet</i> GTC	<i>Réalisateur (s)</i> Achor, Brochard, Eeckeman, Garagnoli	<i>Date</i> 15/02/2018
<i>Réunion</i> 3		
<i>Commentaire</i>		
Réunion de compte rendu du dossier : - Définition de la police d'écriture - Mise en place du thème du dossier - Mise en place du thème du diaporama		



Fiche de réunion		Réf.FMC-XX
<i>Projet</i>	<i>Réalisateur (s)</i>	<i>Date</i>
GTC	Achor, Brochard, Eeckeman, Garagnoli	04/04/2018
<i>Réunion</i>		
4		
<i>Commentaire</i>		
Compte rendu de la première revue de projet (note) : • La raspberry doit pouvoir mettre à jour l'heure de l'arduino. Synchronisation des horloges. • Mettre en place un système sur l'arduino qui indique s'il y a un dysfonctionnement. • Envoyer TOUTES les données de l'arduino à chaque communication. • Faire le CRC. • Renvoyer le nom du lampadaire à chaque communication.		



Fiche de réunion		Réf.FMC-XX
Projet	Réalisateur (s)	Date
GTC	Achor, Brochard, Eeckeman, Garagnoli	12/04/18
Réunion		
5		
Commentaire		
Réunion explication détailler de la trame utiliser		