Dossier Technique - Revue Finale

Projet Tournesol

Étudiant 2 : COUTAN Damien (EC)
Étudiant 3 : LUGHIRI Ayoub (IR)
Étudiant 4 : LELIEVRE Alexis (IR)

Table des matières

Projet Tournesol

Présentation du projet

On a besoin d'une alimentation électrique fiable pour alimenter les équipements nécessaires mais la présence d'un réseau électrique fonctionnel n'est pas toujours assurée dans les situations suivantes :

- installations isolées hors-réseau (phares, hôpitaux de campagne, ...)
- événements organisés en extérieur

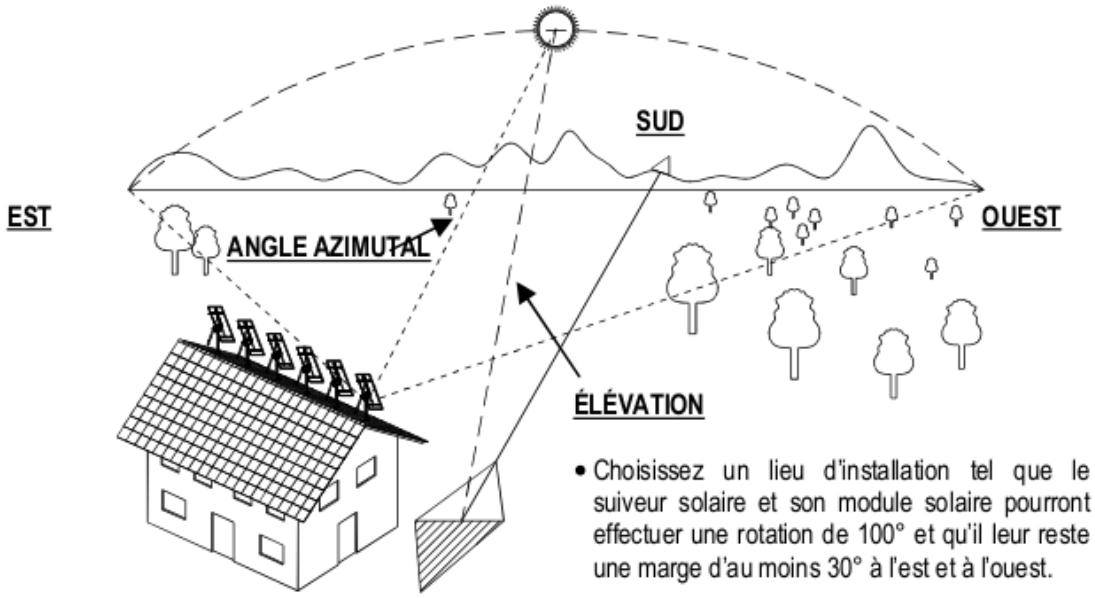
Si aucun réseau n'est disponible, un générateur serait le choix logique pour générer de l'électricité. Cependant, les générateurs sont bruyants et très polluants. De plus, les directives européennes et nationales encouragent fortement la création d'unités locales de production d'énergies renouvelables car la dépendance des pays industrialisés à l'égard des combustibles fossiles, et des pays qui les produisent, induit aujourd'hui à de véritables enjeux économiques et politiques.

Le solaire fait partie de ces énergies remises récemment au goût du jour. En effet, la théorie liée au photovoltaïque est ancienne puisqu'elle a été décrite par Hertz et Einstein au début du XX ème siècle. L'énergie solaire est considérée comme inépuisable (compte tenu de la durée de vie du Soleil) et n'émet potentiellement aucune particule nocive (fabrication des panneaux pour la capter et des batteries pour la stocker mises à part). Mais le solaire est un secteur qui reste onéreux : environ 680 euros le mètre carré de panneau (posé avec onduleur et accessoires) pour une qualité moyenne (silicium amorphe ou monocristallin), ce qui le défavorise face à la concurrence d'autres énergies.

Il ressort l'importance d'améliorer le rendement des installations solaires, pour exploiter au mieux les ressources potentielles. Celui-ci pourrait être augmenté de deux manières : la première consisterait à améliorer techniquement la cellule photovoltaïque, la seconde à optimiser l'angle d'éclairement du panneau en fonction de la position du soleil.

En retenant la seconde solution, il s'agira désormais d'appliquer le principe de l'héliotropisme 1 pour les panneaux solaires afin d'augmenter la production journalière d'électricité.

En effet, pour suivre le soleil de façon héliotropique nous utiliserons l'angle d'azimuts et d'élévation ; ainsi par le biais de ces deux paramètres on pourra orienter le panneau. Par ailleurs, les panneaux solaires seront contrôlés par un module composée de photorésistances LDR(Light Depend Resistor) qui permettront de diriger le panneau de façon automatique et donc d'optimiser la récolte.



Objectifs

Il faut réaliser un système qui assure le fonctionnement autonome d'une installation photovoltaïque motorisée permettant d'optimiser la récolte d'énergie.

Le système devra donc :

- S'orienter efficacement en toute sécurité pour optimiser la récolte d'énergie
- Assurer la régulation de l'énergie et le contrôle sécurisé de la charge des batteries
- Récupérer les informations de l'ensemble de l'installation et les stocker
- Signaler et journaliser les alarmes
- Communiquer avec l'utilisateur par le biais d'une IHM

Architecture matérielle du système

Le système sera donc composé de panneaux photovoltaïques motorisés et des équipements nécessaires au stockage de l'énergie et à la régulation de l'énergie.

On distinguera les composants suivants :

Composants	***Type et caractéristiques***
2 x Panneaux solaires	Mono Cristallin 80W-12V Nom: SPM80-12 Taille du module: 1110 x 540 x 35 mm

	Poids: 8 kg
2 x Batteries	Nom :BAT 212120080 soit AGM (Absorbent glass matt) 12V-14aAh Victron Energy taille : 151 x 98 x 101 mm poids : 4,1kg
1 x Régulateur de charge	Nom :Phocos-CX20 12/24V taille :92 x 93 x 38 mm poids : 168 g Communication : USB
1 x Système de commande de positionnement 2 axes	Nom :MICRO POZSOL36A1DR (24V) Taille : 55 x 110 x 17 mm Poids : 100g Communication : USB
1 x Anémomètre	LEXCA003
1 x Capteur solaire	Construction par l'étudiant 2 avec des photorésistances (LDR)

Missions du système

Le système devra donc assurer :

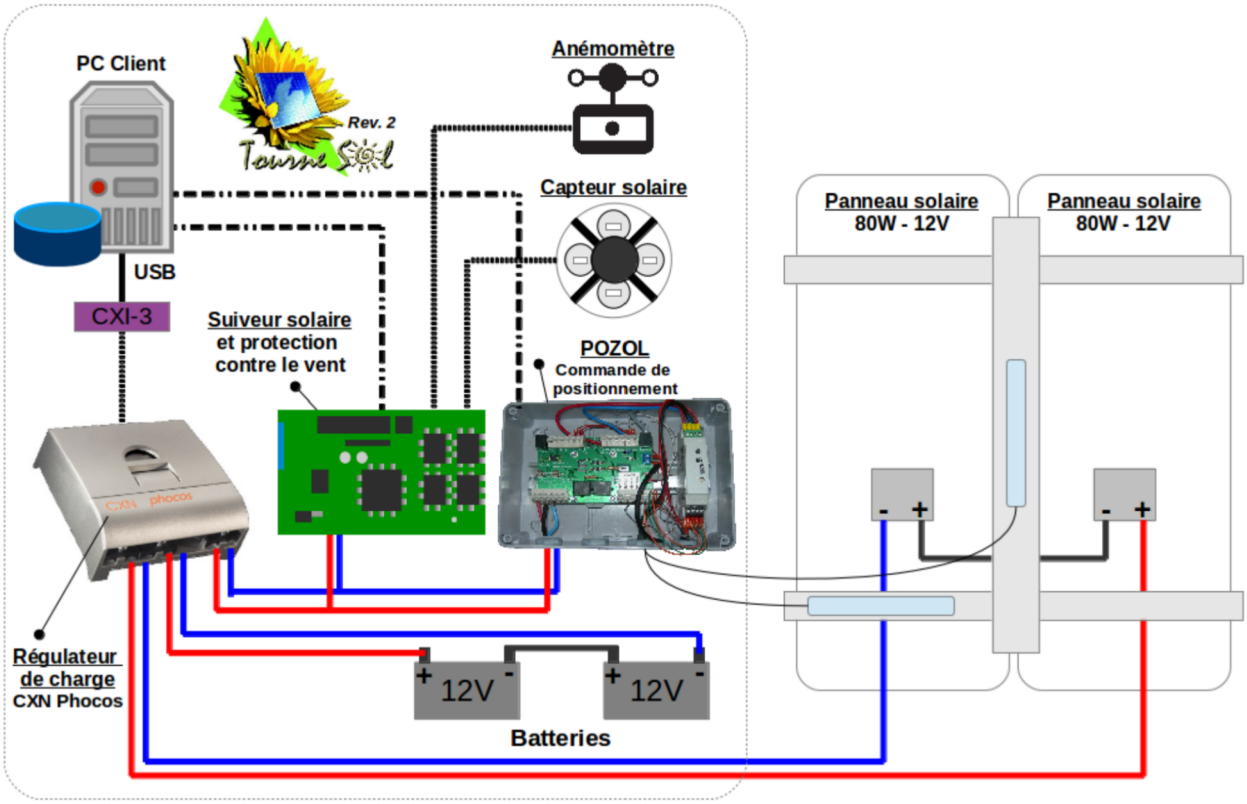
- **Le fonctionnement en mode manuel ou automatique** ;
- **La commande du positionnement des panneaux solaires en toute sécurité** ;
- Le pilotage du régulateur de charge;
- **Le paramétrage des consignes de protection contre le vent** ;
- **L'acquisition des données** des modules (**suiveur solaire**, régulateur de charge et station météo) ;
- **La visualisation des états, des données et des alarmes sur un IHM** ;
- L'archivage des états, des données et des alarmes dans une base de données.

On distinguera les modules suivants :

- **Suiveur solaire (panneaux photovoltaïques, système de commande de positionnement, capteurs de positionnement solaire)** ;

- Régulateur de charge (liaison avec les panneaux photovoltaïque, les batteries et les sorties ;
- Station météo (anémomètre, capteur de luminosité).

Configuration d'exploitation



Le système Tournesol

On retiendra donc les différents besoins de ce système soit :

- Transformer l'énergie solaire en énergie électrique
- Réguler l'énergie
- Stocker l'énergie
- Surveiller les alertes

Exigences

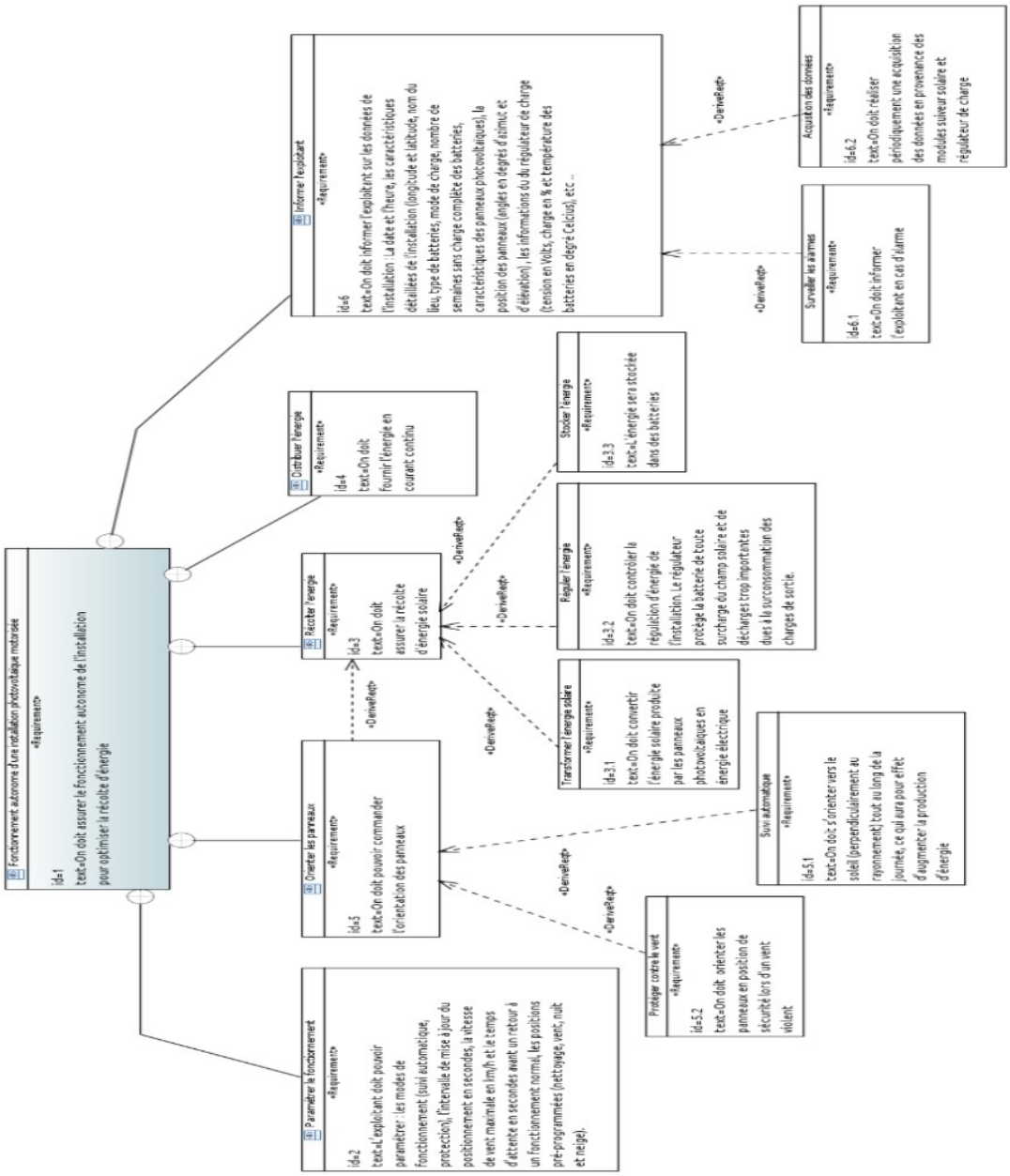


Diagramme des exigences

Diagramme des cas d'utilisation

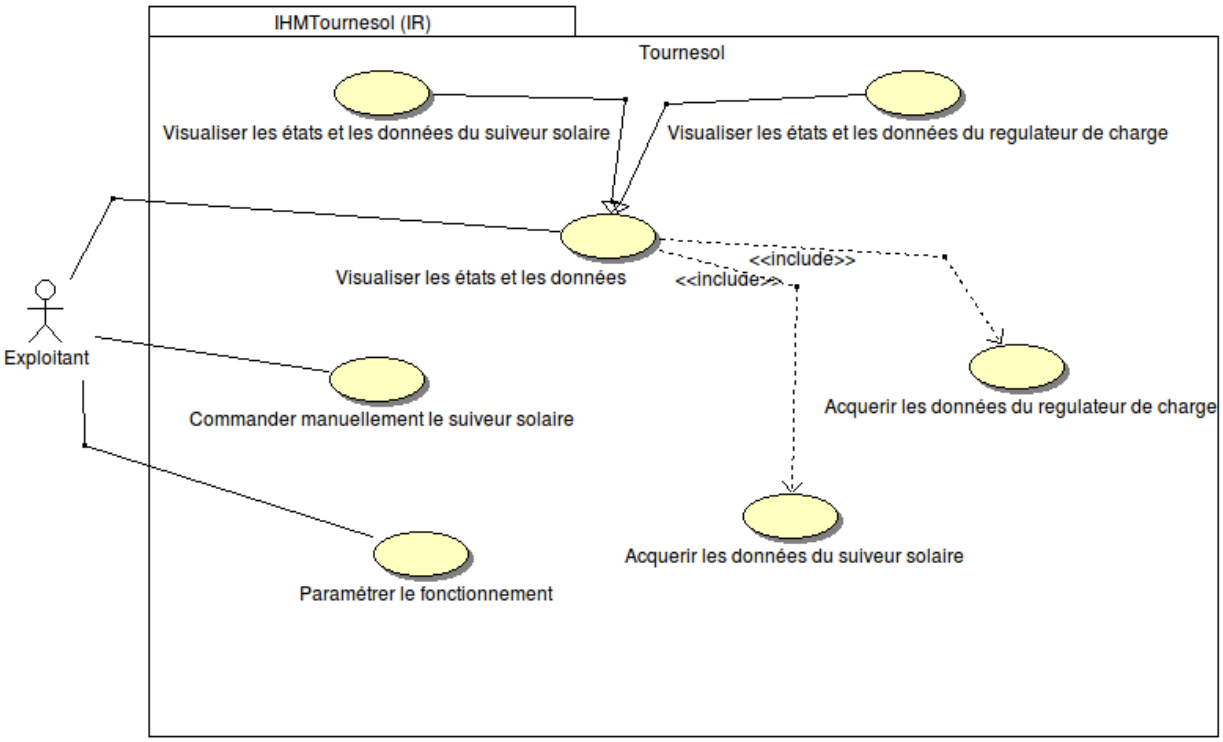
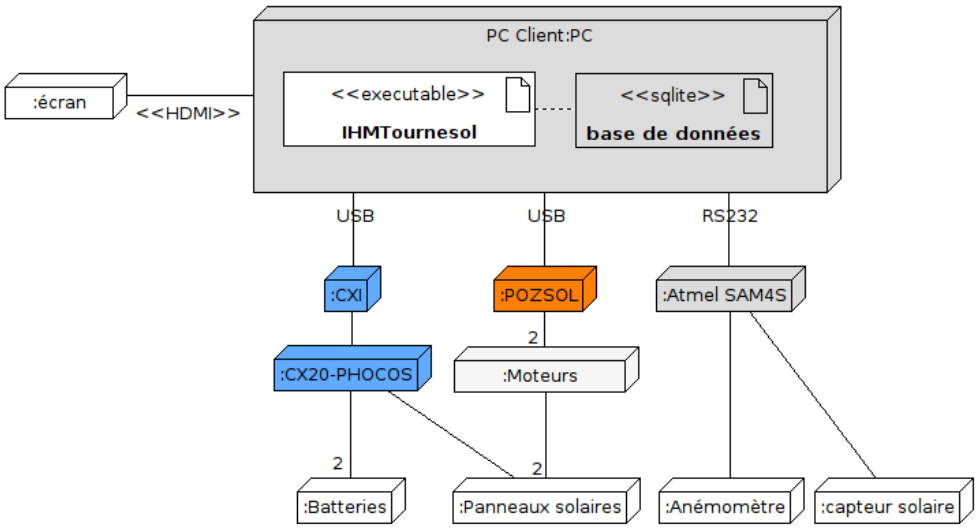


Diagramme de déploiement



Ressources de développement

Environnement de développement	© `ATMEL Studio`
Système d'exploitation de développement	© `Microsoft Windows` (7 minimum)
Système de gestion de bases de données relationnelles	`SQLite3` (`sqliteman` ou `SQLiteManager`)
Environnement de développement du PCC et PCT	`Qt Creator` 3.6.1 et `Qt Designer`
Compilateur du PCC et PCT	`MinGW` (*Minimalist GNU for Windows*)
API GUI	`Qt` 5.6 et `Qwt` version 5 ou 6
Atelier de génie logiciel	`bouml` 4.23
Plate-forme de tests unitaires	`CppUnit`
Logiciel de gestion de versions	`subversion` (client `svn`)
Générateurs de documentation	`Doxygen` et `pandoc`
Gestionnaire de projet	`Planner` ou `gantter`

Module de gestion du suiveur solaire

Objectifs

Ce projet a pour but de réaliser un système capable de d'optimiser la récolte de l'énergie solaire et d'être autonome dans sa consommation en énergie.

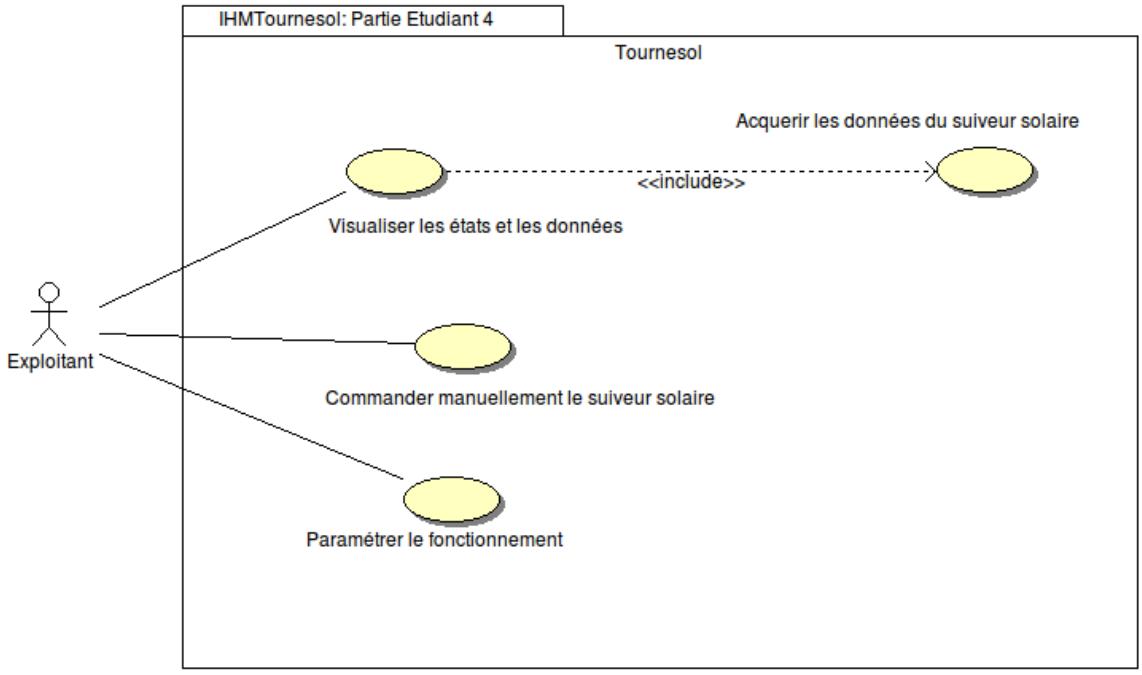
Le système utilisé possède un **module de gestion du suiveur solaire**. Ce module contrôle deux moteurs pour pouvoir orienter le panneau. En effet, le panneau peut être orienté de façon manuelle ou de façon automatique grâce à un programme d'orientation déjà inclus dans ce module. Il pourra orienter le panneau selon un référentiel scientifique qui lui permet de suivre le soleil.

Je dois réaliser ce module qui permet :

- **Commander** et **paramétrer** le fonctionnement du suiveur solaire avec une IHM sous Qt .

Mon rôle est de développer une application avec une interface graphique permettant de gérer le panneau solaire.

Cas d'utilisation



En résumé, j'ai la responsabilité du « Module : Suiveur Solaire » qui à pour but de commander manuellement les panneaux ainsi que de les commander en fixant les angles horaire et d'élévation, d'acquérir les données et les états du suiveur solaire. Il devra aussi créer des positions pré-programmées qui peuvent être modifiées, elles permettront en effet de déplacer le panneau en

fonction des conditions météorologiques telles que le vent, la nuit ou la neige et également pour le nettoyage.

Ceci nous donne les tâches suivantes :

- Commander manuellement les panneaux
- Paramétrer le fonctionnement du suiveur solaire
- Surveiller les alarmes du suiveur solaire
- Visualiser les états et données du suiveur solaire
- Archiver dans une base de données

Production attendue

➔ Une application informatique fonctionnelle

➔ Un modèle UML complet de la partie à développer ;

➔ Le code source commenté de l'application ;

➔ Les documentations associées au module à produire.

Planification

Pour réaliser l'ensemble des ces tâches, j'ai du mettre en place une planification. En effet, j'ai réalisé un diagramme de Gantt que vous trouverez en annexe 1.1.

J'ai aussi réalisé un fichier texte « TODO » que vous pouvez trouver dans la documentation de mon code via internet, lequel liste les choses que je devais faire. Vous trouverez ci-dessous un tableau qui récapitule les tâches définies pour le projet au moment de sa création.

		6			9		10		11		12		13		14		17		18		19		20		21		22		23	
Tâches		ET3	ET4	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4	ET3	ET4
S'approprier le cahier des charges	9%	10	10		4	4	2	2																						
Installer et configurer le système d'exploitation du PC	1%						2	2																						
Installer et configurer la base de données	1%						2	2																						
Installer et configurer son environnement de développement	1%				2	2																								
Installer et raccorder les appareils	1%						1	1																						
Mettre en œuvre les programmes de test fournis	7%				2	2	4	4	6	6																				
Finaliser la modélisation UML du module	12%						2	2	2	2	2	2	4	4	2	2	2	2	2	2	2	2	2	2						
Produire la maquette de l'IHM du module	5%										2	2			1	1	2	2							2	2			1	1
Coder les classes du module	23%										4	4	4	4	4	4	4	4	5	5	3	3	6	6	4	4	4	4	1	1
Réaliser les tests unitaires	6%										1	1	1	1	1	1	1	1	1	1	1	1			2	2	2	2	1	1
Faire la recette du module	5%																				4	4							4	4
Intégrer en équipe l'application complète	2%																												4	4
Rédiger le dossier technique et les documents relatifs au projet	18%	1	1		2	2			2	2	2	2	2	2	3	3	2	2	2	2	2	2	2	2	4	4	4	4	2	2
Produire un guide de mise en route et d'utilisation du module.	2%																								2	2	2	2		
Gérer la planification	5%	1	1		1	1	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
Revue n°0	1%				1	1																								
Revue n°1	1%										1	1																		
Revue n°2	1%																				1	1								

Matériels mis en œuvre

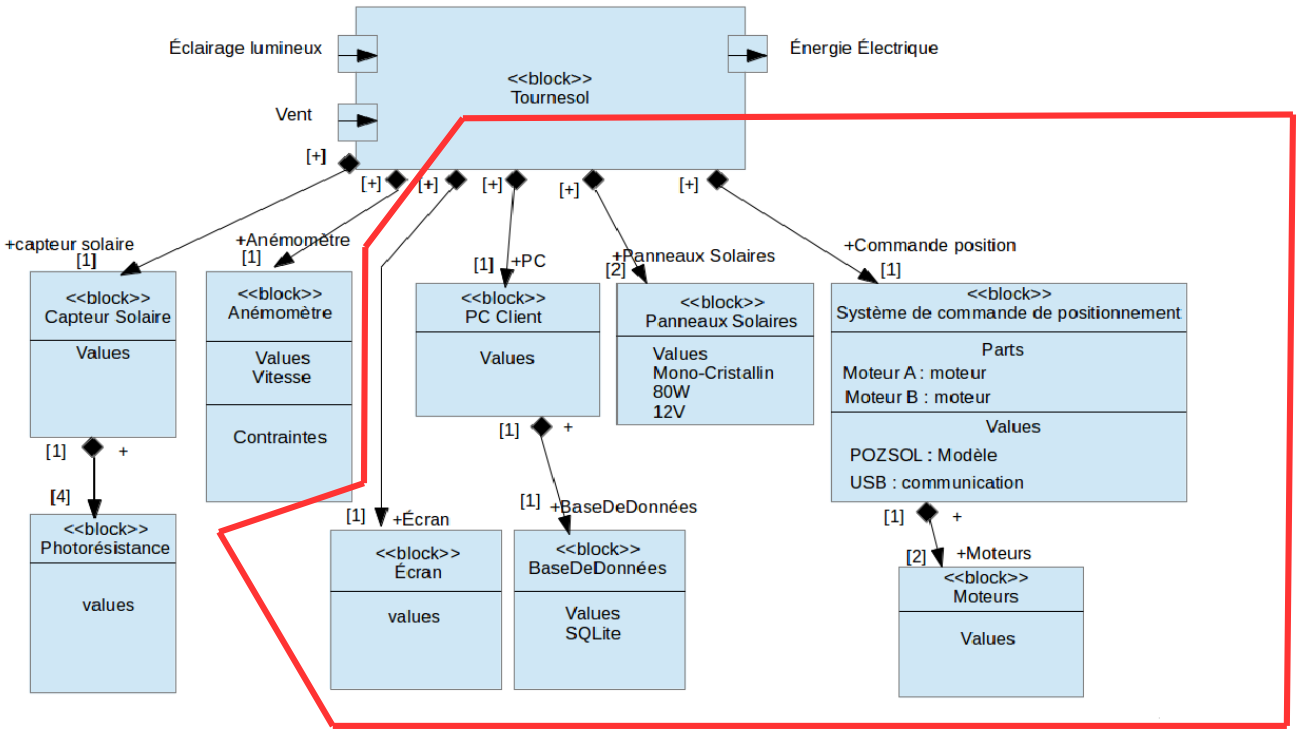
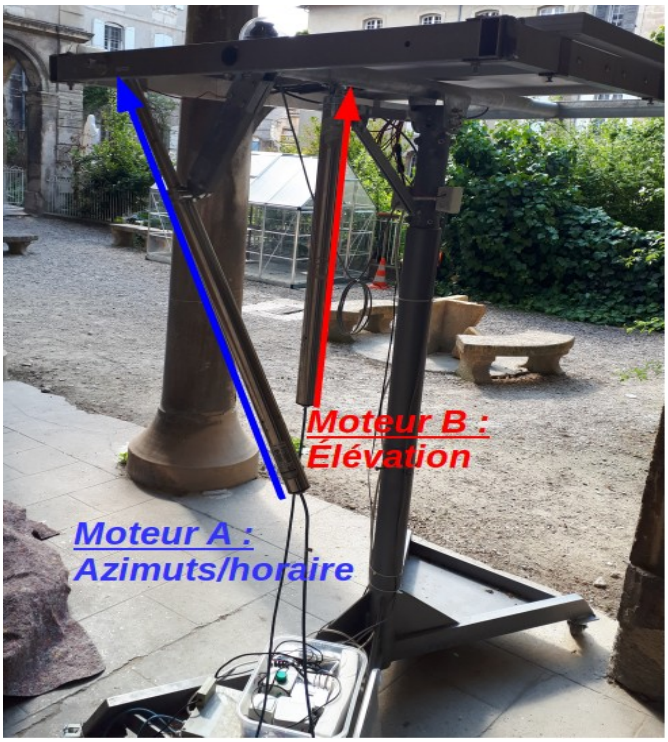


Diagramme de blocs

En rouge, vous pouvez voir le matériel sur lequel j'ai pu travailler.

Système de déplacement du panneau solaire POZSOL

POZSOL est le module de commande des vérins sur le suiveur solaire. Il permet de contrôler la direction que va prendre le suiveur solaire. En effet il peut contrôler les deux moteurs, l'un pour suivre un angle horaire et l'autre l'élévation.

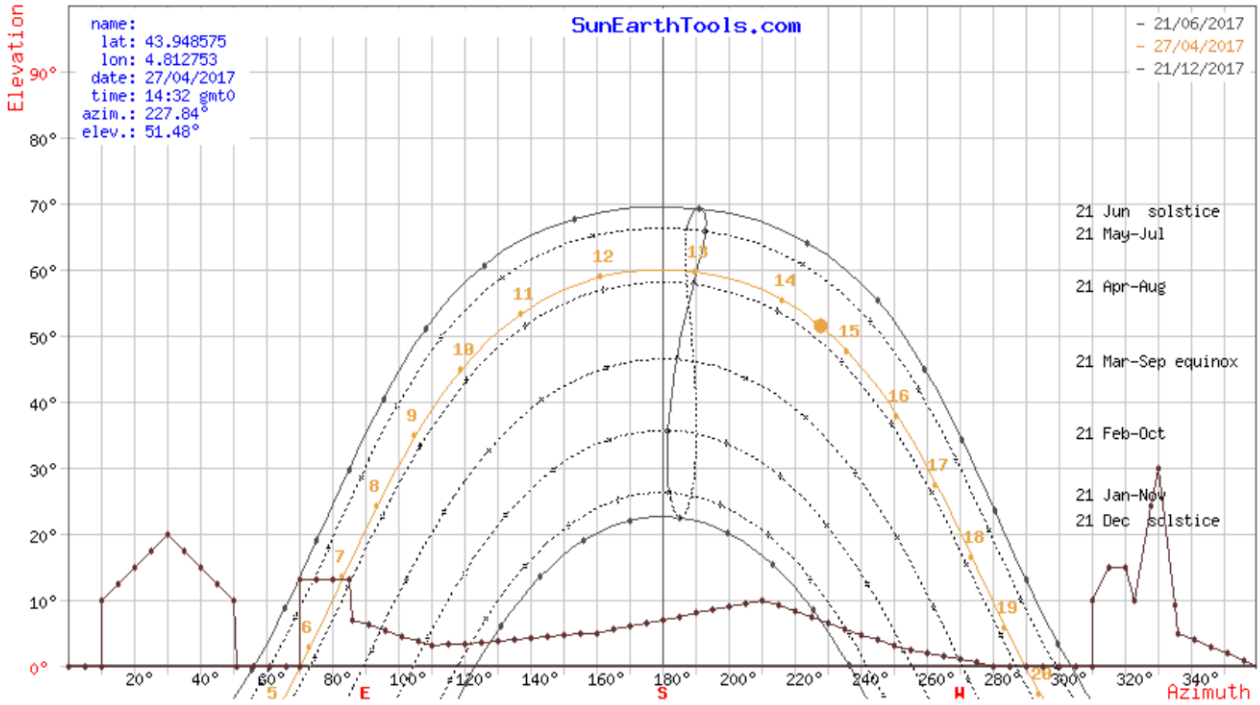


Ce module était celui qui était prévu dans le projet. Mais lors de la mise en œuvre de celui-ci en début de projet, nous avons rencontré un problème. En effet le module n'était plus fonctionnel.

Nous avons dû en commander un nouveau chez le fournisseur (annexe 1.2: bon de commande). Suite à des problèmes administratifs au sein de l'établissement, la commande prit du retard. Celui-ci fut livré le **vendredi 5 mai**.

J'ai donc travaillé sur mon projet pendant 4 mois avec un simulateur qu'on m'a fourni. Celui m'a permis de valider le développement de ma partie. J'ai pu ensuite faire des tests de validation avec le module reçu.

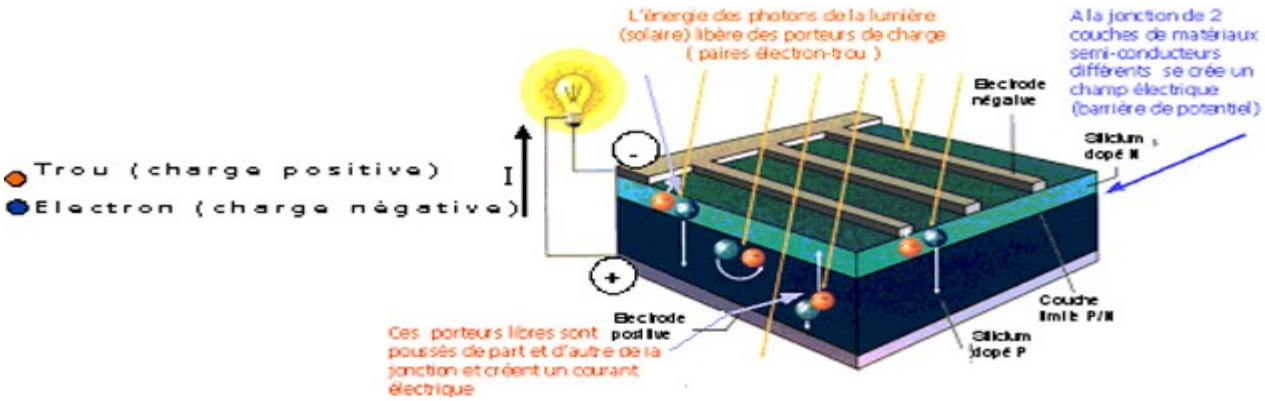
Ce module fournit aussi un programme de déplacement automatique qui suit un référentiel mathématique qui permet au panneau de s'orienter efficacement. On observe sur le schéma ci-dessous l'orientation du soleil selon notre position :



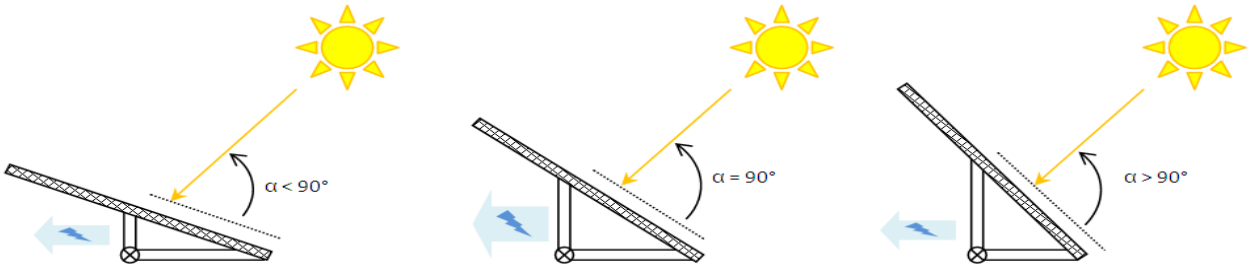
Suiveur Solaire (SM44M1V3P)

Le suiveur solaire est composé de 2 panneaux photovoltaïques monocristallin. Le mono-cristallin est un type de panneau qui provient du fait que les cellules de celui-ci sont issues d'un seul cristal de silicium. Ce type de panneau solaire possède une production moins importante qu'un polycristallin mais son coût lui est moindre.

Explication du fonctionnement d'un panneau solaire :

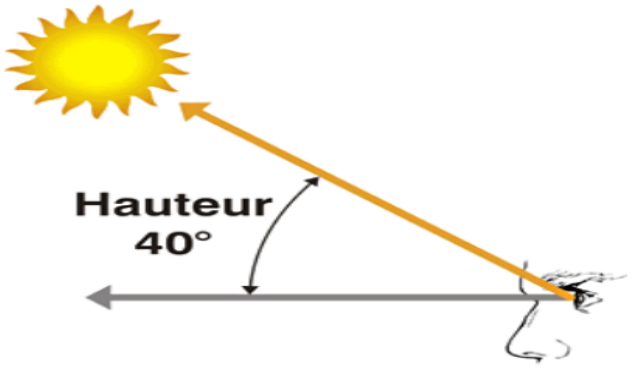


Pour revenir sur la manière dont le panneau s'oriente efficacement, il faut savoir que celui-ci s'oriente avec deux vérins. L'un permet de se déplacer sur l'angle horaire jusqu'à 100° max. Il faut savoir que l'angle horaire peut être représenté par l'angle formé par le plan vertical du soleil et le plan méridien (voir schéma ci-dessous).

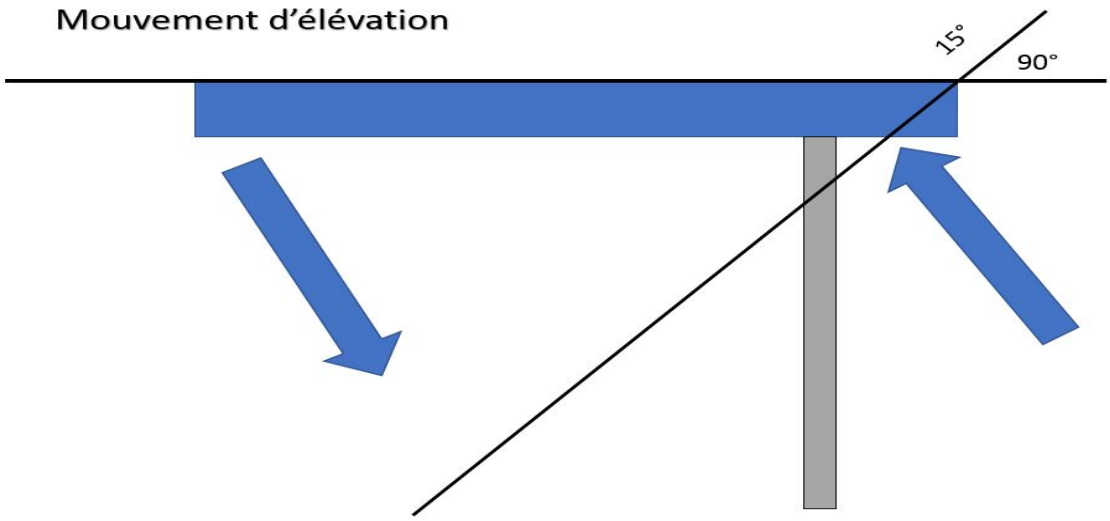


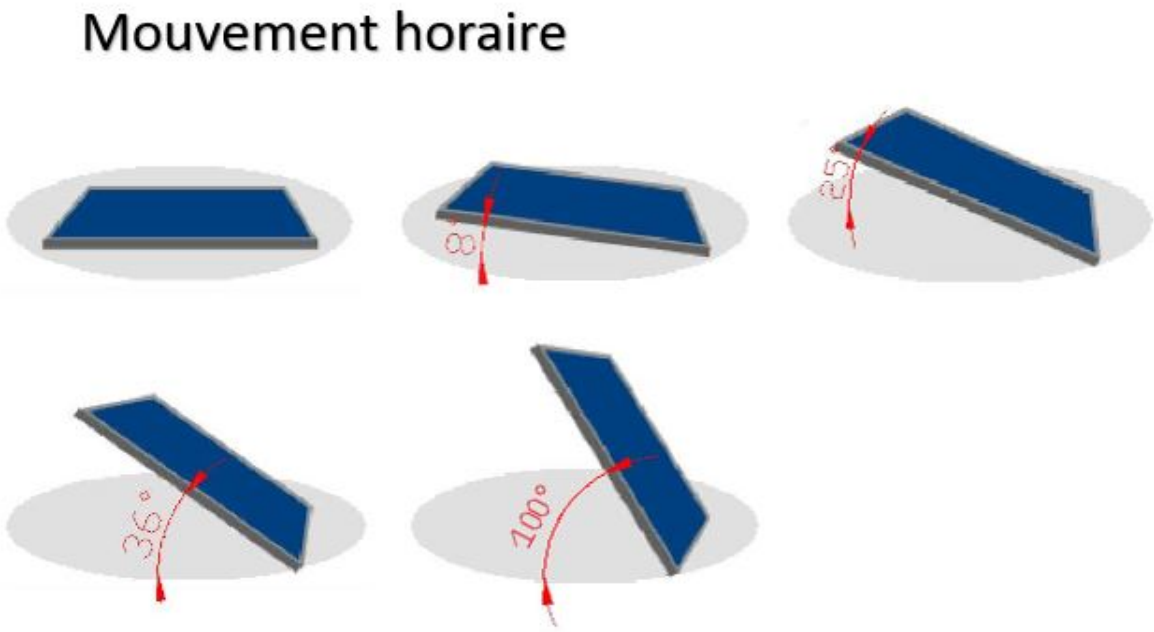
Sur le schéma α représente l'angle horaire.

S'agissant de l'autre vérin, il est utilisé pour se déplacer sur un angle d'élévation 15 – 90°max. L'angle d'élévation peut être représenté par le schéma ci-dessus lui aussi, mais l'angle d'élévation a pour particularité qu'il emploie un point d'observation sur terre comme l'homme (voir schéma ci-dessous).



Vous pouvez voir ci dessous les schémas qui représentent le déplacement du panneau :

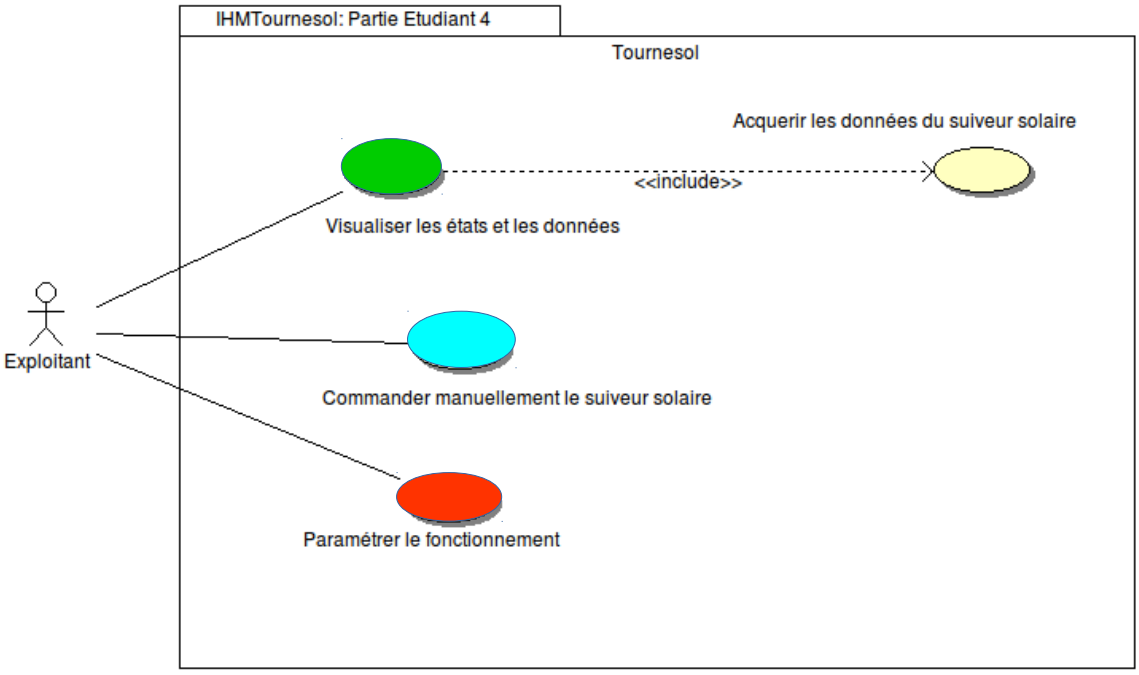

Mouvement d'élévation
15°
90°

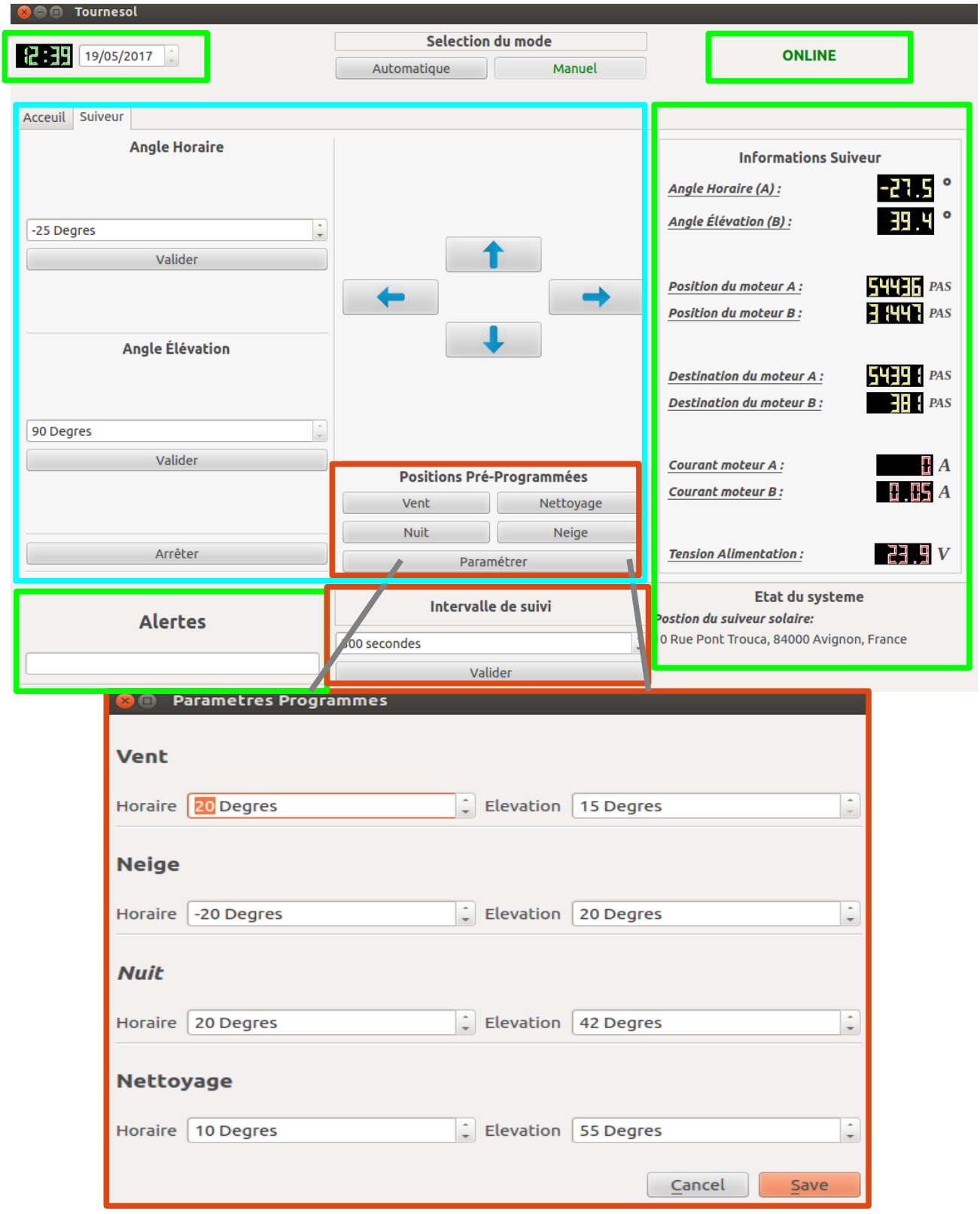

Mouvement horaire
25°
36°
100°

Conception et réalisation

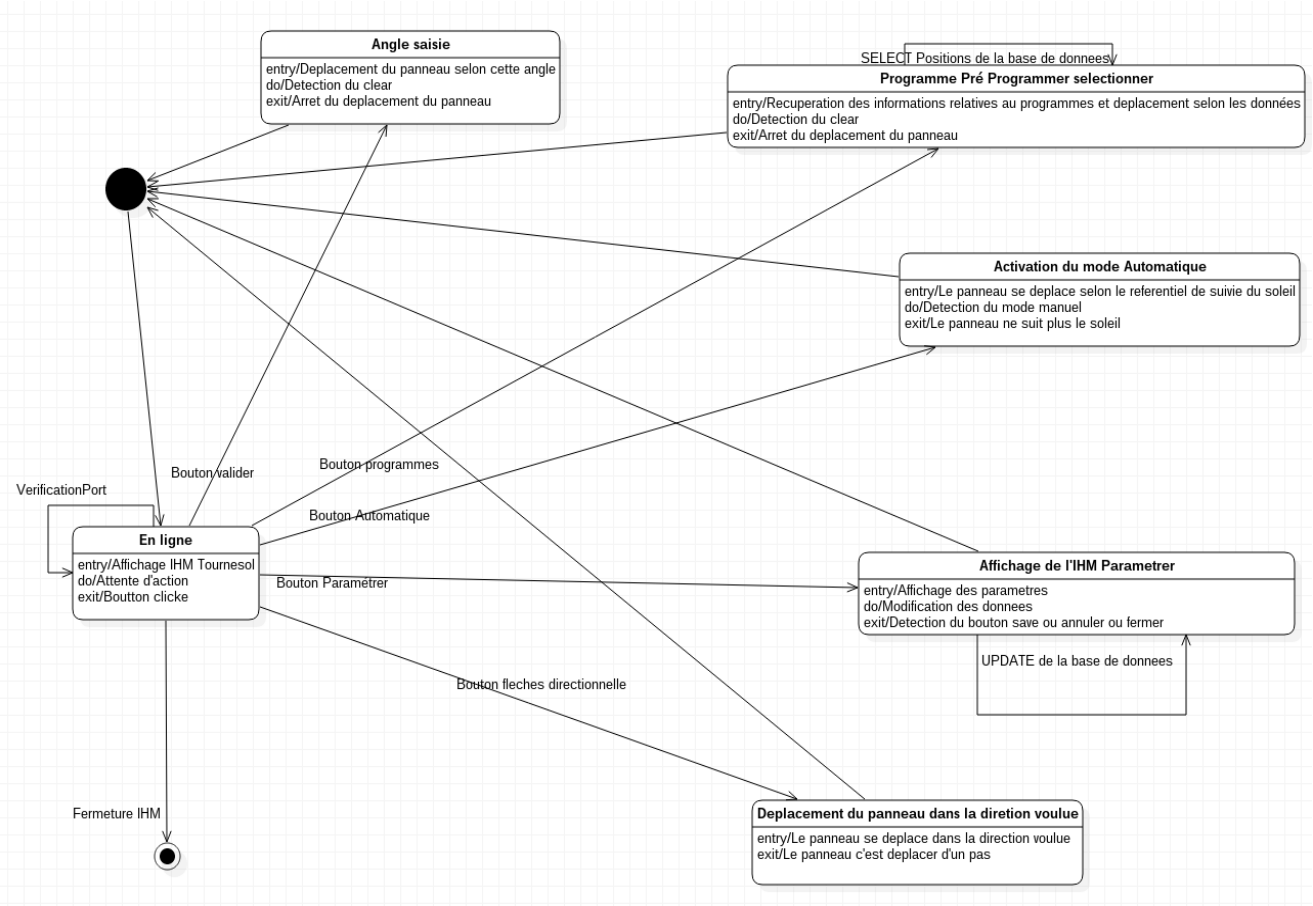
IHM

Si on reprend le cas d'utilisation qui a été défini plus haut, on retrouve chacune des parties qui ont été développées dans l'IHM.




Tournesol
12:39
19/05/2017
Selection du mode
Automatique
Manuel
ONLINE
Acceuil
Suiveur
Angle Horaire
-25 Degres
Valider
Angle Élévation
90 Degres
Valider
Arrêter
Positions Pré-Programmées
Vent
Nettoyage
Nuit
Neige
Paramétrer
Informations Suiveur
Angle Horaire (A) :
-27.5
°
Angle Élévation (B) :
39.4
°
Position du moteur A :
54436
PAS
Position du moteur B :
31447
PAS
Destination du moteur A :
54391
PAS
Destination du moteur B :
381
PAS
Courant moteur A :
0
A
Courant moteur B :
0.05
A
Tension Alimentation :
23.9
V
Alertes
Intervalle de suivi
300 secondes
Valider
Etat du systeme
Postion du suiveur solaire:
0 Rue Pont Trouca, 84000 Avignon, France
Parametres Programmes
Vent
Horaire
20 Degres
Elevation
15 Degres
Neige
Horaire
-20 Degres
Elevation
20 Degres
Nuit
Horaire
20 Degres
Elevation
42 Degres
Nettoyage
Horaire
10 Degres
Elevation
55 Degres
Cancel
Save

Pour détailler chacune des actions de l'interface graphique, j'ai modélisé un diagramme d'états transitions qui résume l'IHM.



Base de données

Cette base de données m'a été fournie avec le projet. Parmi les tables qui la composent, je n'en utilise que trois :

- *Positions* pour pouvoir enregistrer les données des programmes
- *Suiveur* pour enregistrer l'intervalle du suiveur solaire
- *Installation* pour récupérer les données initiales du suiveur solaire soit
  - tous les champs de *longitude* à *nbBatterie*

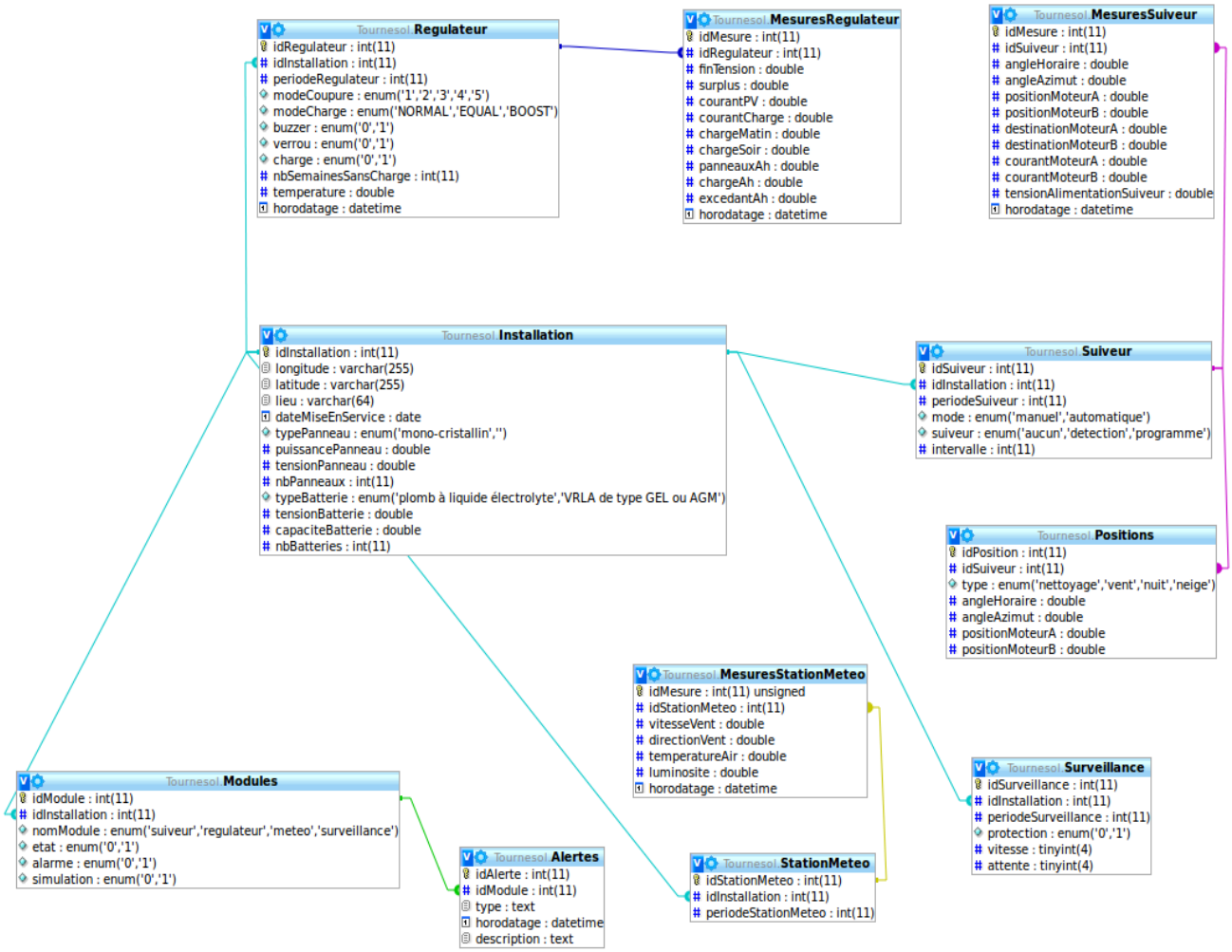


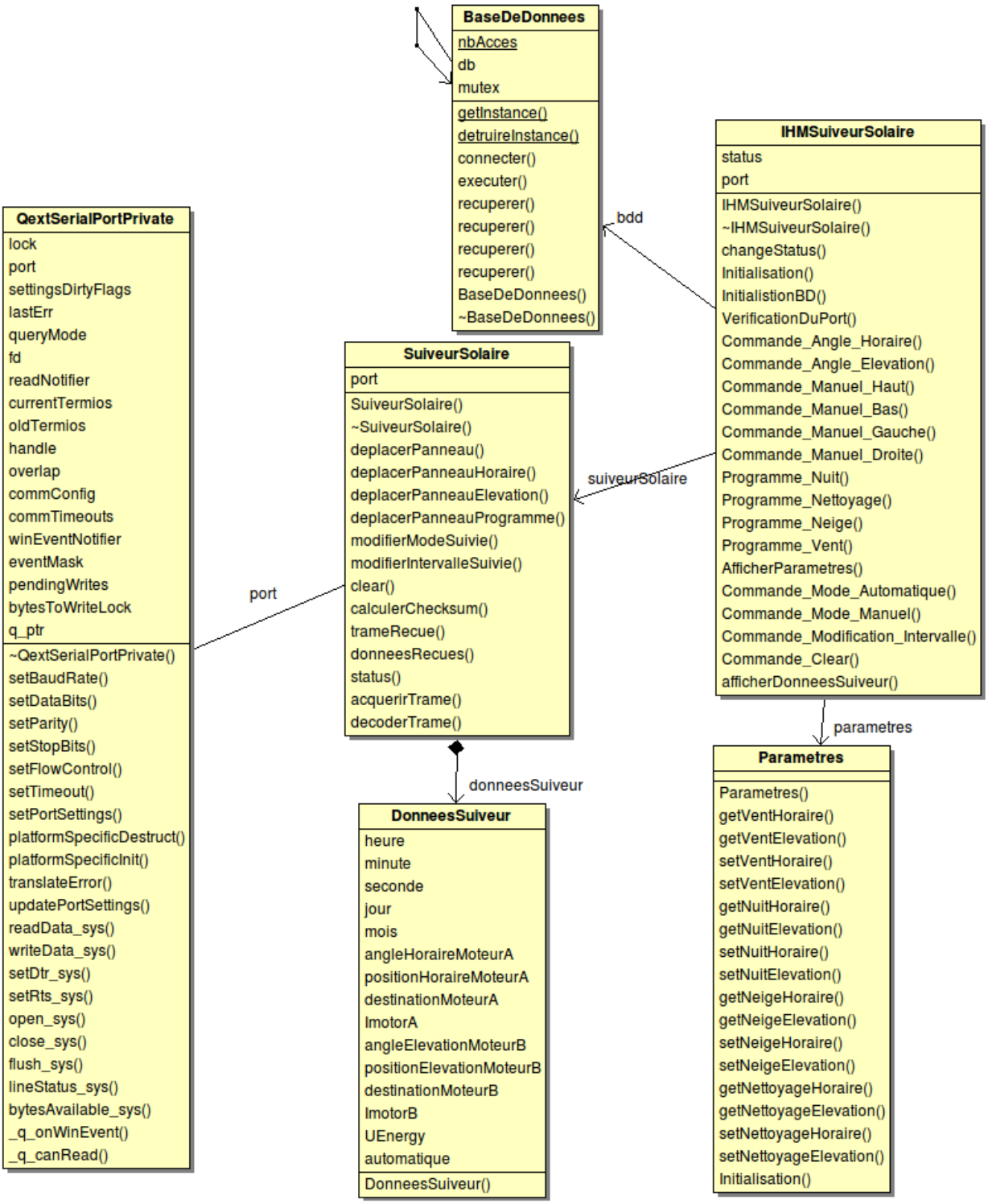
Diagramme de classes

Ce diagramme est composée de 6 classes dont chacune à son rôle propre.

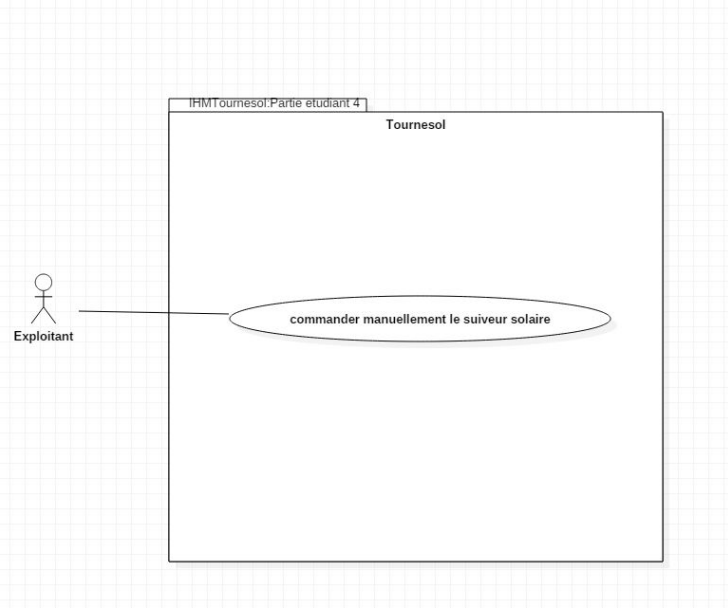
La classe principale est la classe ***IHMSuiveurSolaire*** qui recense toutes les fonctionnalités de l'interface graphique. Elle est en liaison directe avec la classe mère du système ***SuiveurSolaire***.

Elle permet en effet de générer les trames vues auparavant et d'acquérir les trames émises par le suiveur solaire pour les stocker dans la classe ***DonneesSuiveur*** qui sert de stockage de données, pour être ensuite utilisées dans l'interface graphique, cela grâce à son exploitation du port qui est réalisé par l'utilisation de la classe ***QextSerialPort*** permettant l'exploitation des périphériques.

La classe ***IHMSuiveurSolaire*** possède aussi deux autres relations avec deux autres classes, la classe ***Parametres*** qui a pour but de gérer la récupération des valeurs saisies dans la boite de dialogue qui est en liaison avec les programmes pré-programmés. Ces mêmes valeurs sont alors envoyées dans la base de données grâce à l'utilisation de la classe ***BaseDeDonnees***.


BaseDeDonnees
nbAcces
db
mutex
getInstance()
detruireInstance()
connecter()
executer()
recuperer()
recuperer()
recuperer()
recuperer()
BaseDeDonnees()
~BaseDeDonnees()
bdd
IHMSuiveurSolaire
status
port
IHMSuiveurSolaire()
~IHMSuiveurSolaire()
changeStatus()
Initialisation()
InitialistionBD()
VerificationDuPort()
Commande_Angle_Horaire()
Commande_Angle_Elevation()
Commande_Manuel_Haut()
Commande_Manuel_Bas()
Commande_Manuel_Gauche()
Commande_Manuel_Droite()
Programme_Nuit()
Programme_Nettoyage()
Programme_Neige()
Programme_Vent()
AfficherParametres()
Commande_Mode_Automatique()
Commande_Mode_Manuel()
Commande_Modification_Intervalle()
Commande_Clear()
afficherDonneesSuiveur()
QextSerialPortPrivate
lock
port
settingsDirtyFlags
lastErr
queryMode
fd
readNotifier
currentTermios
oldTermios
handle
overlap
commConfig
commTimeouts
winEventNotifier
eventMask
pendingWrites
bytesToWriteLock
q_ptr
~QextSerialPortPrivate()
setBaudRate()
setDataBits()
setParity()
setStopBits()
setFlowControl()
setTimeout()
setPortSettings()
platformSpecificDestruct()
platformSpecificInit()
translateError()
updatePortSettings()
readData_sys()
writeData_sys()
setDtr_sys()
setRts_sys()
open_sys()
close_sys()
flush_sys()
lineStatus_sys()
bytesAvailable_sys()
_q_onWinEvent()
_q_canRead()
port
SuiveurSolaire
port
SuiveurSolaire()
~SuiveurSolaire()
deplacerPanneau()
deplacerPanneauHoraire()
deplacerPanneauElevation()
deplacerPanneauProgramme()
modifierModeSuivie()
modifierIntervalleSuivie()
clear()
calculerChecksum()
trameRecue()
donneesRecues()
status()
acquerirTrame()
decoderTrame()
suiveurSolaire
donneesSuiveur
DonneesSuiveur
heure
minute
seconde
jour
mois
angleHoraireMoteurA
positionHoraireMoteurA
destinationMoteurA
lmotorA
angleElevationMoteurB
positionElevationMoteurB
destinationMoteurB
lmotorB
UEnergy
automatique
DonneesSuiveur()
parametres
Parametres
Parametres()
getVentHoraire()
getVentElevation()
setVentHoraire()
setVentElevation()
getNuitHoraire()
getNuitElevation()
setNuitHoraire()
setNuitElevation()
getNeigeHoraire()
getNeigeElevation()
setNeigeHoraire()
setNeigeElevation()
getNettoyageHoraire()
getNettoyageElevation()
setNettoyageHoraire()
setNettoyageElevation()
Initialisation()

Cas d'utilisation « Commander manuellement le suiveur solaire »



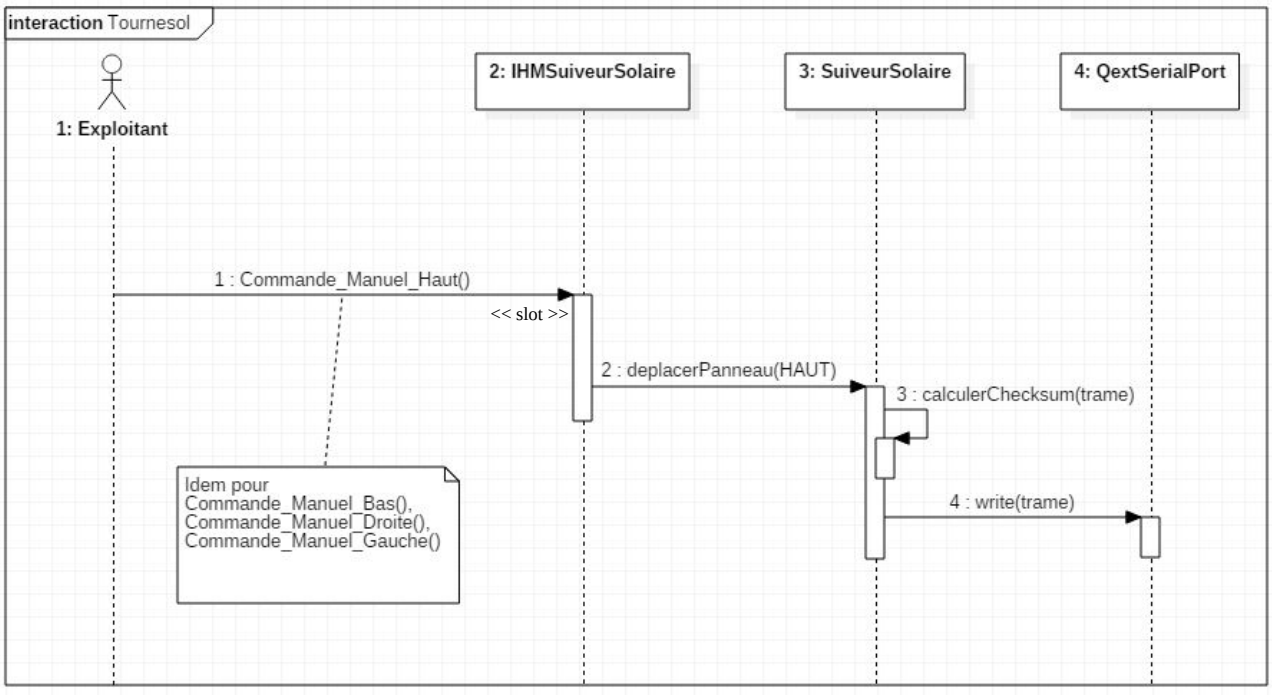


Diagramme Séquence : Modélisation de la commande manuel Haut

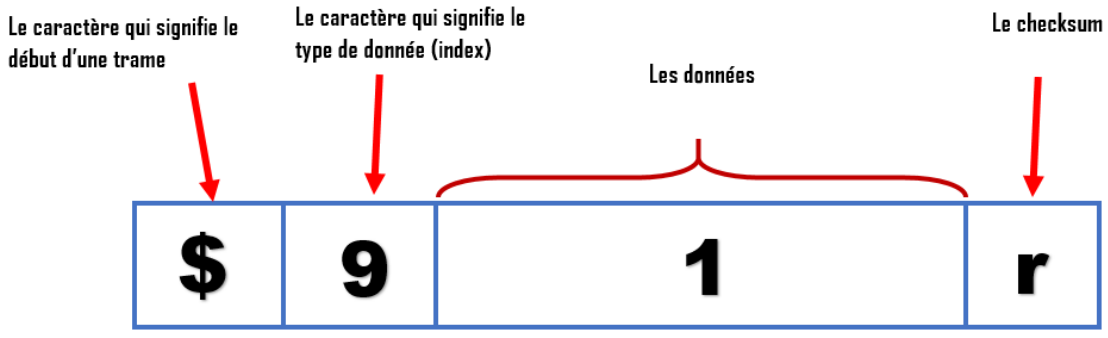
Lorsque l'exploitant clique sur le bouton « flèche haute », cela déclenche le *slot* Commande_Manuel_Haut() qui s'occupe d'appeler la méthode *deplacerPanneau()* de la classe ***SuiveurSolaire***.

La méthode *deplacerPanneau()* permet de déplacer le panneau dans l'une des quatre directions :

```
void SuiveurSolaire::deplacerPanneau(char direction)
{
    QString trame;
    unsigned char checksum;

    trame = "$" + QString(INDEX_Manual_move) + QString(direction);

    checksum = calculerChecksum(trame);

    //+= "Concatenation"
    trame += QString(checksum);

    qDebug() << Q_FUNC_INFO << "trame" << trame;

    port->write(trame.toLatin1());
}
```

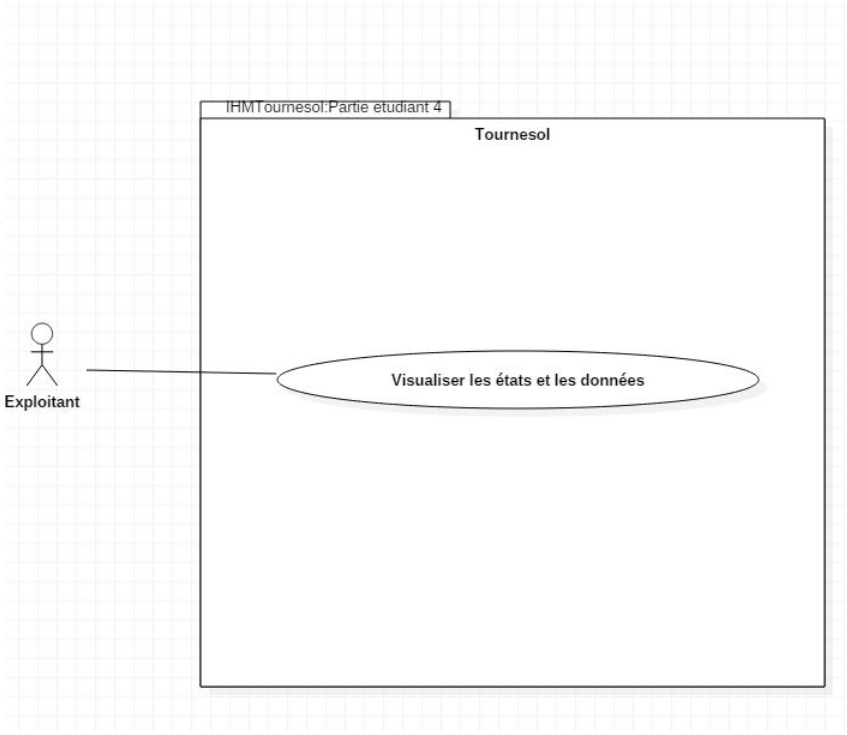
La trame générée pour un déplacement vers le haut sera `$91r` où :

- $ → 0x24 en Hexadécimal
- 9 → 0x39 Il permet de définir une commande manuelle
- 1 → 0x31 Il permet de définir un mouvement vers le haut
- r → 0x72 Le checksum qui permet de valider la trame



Pour finir la trame est envoyée au suiveur solaire en appelant la méthode *write()* de l'objet port.

Cas d'utilisation « Visualiser les états et les données »



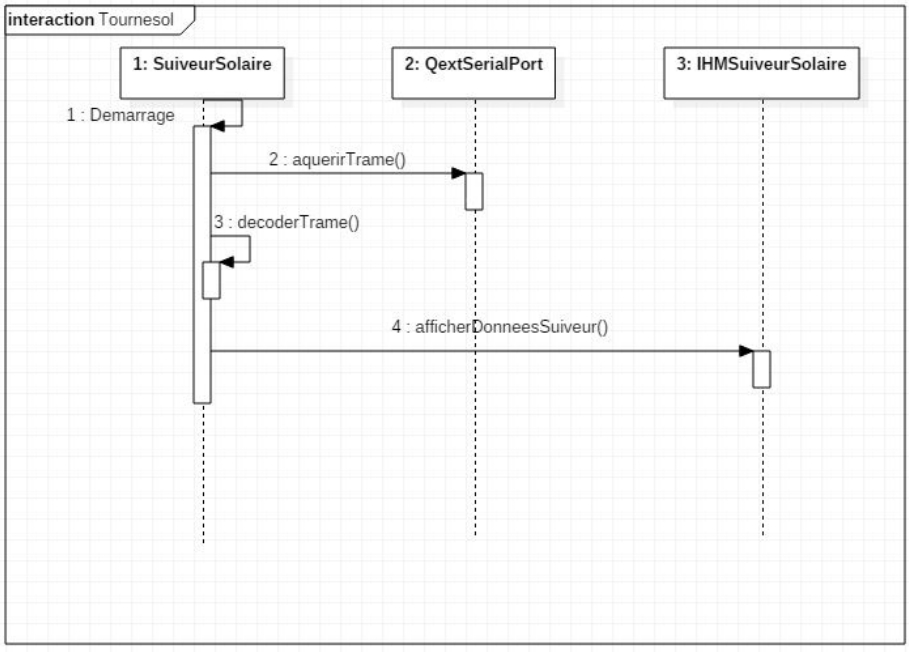


Diagramme Séquence : Modélisation de la récupération des données du suiveur solaire

Lorsque l'IHM se lance, *la récupération* de la trame est réalisée par la méthode acquerirTrame() membre de la classe ***SuiveurSolaire*** qui s'occupe de récupérer les données qui arrivent sur le port et puis elle les met dans l'attribut trame.

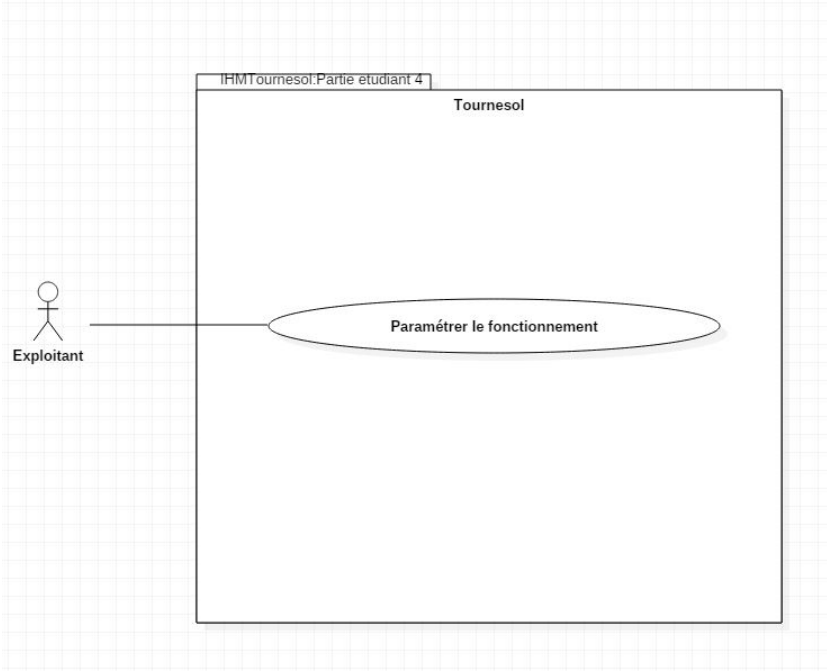
```
void SuiveurSolaire::acquerirTrame()
{
    QByteArray trame;

    trame = port->readAll();

    do
    {
        ::usleep(200*1000);
        trame += port->readAll();
    }
    while(port->bytesAvailable());

    //qDebug() << Q_FUNC_INFO << "trame" << trame;

    if(trame.length() > 0)
    {
        emit trameRecue(trame);
    }
}
```

Cette trame est ensuite analysée dans la méthode *decoderTrame()* membre de la classe ***SuiveurSolaire***. Pour ce faire, on va découper la trame en plusieurs, et cela à chaque fois que l'on détecte un « $ ». Puis on va commencer à analyser chacun de ces trames en recherchant l'index que l'on désire. Cela est réalisé par la fonction *switch/case* qui va déterminer la trame qui correspond à chacune des informations (celle-ci ne sont pas toutes lister ci-dessous).

```
void SuiveurSolaire::decoderTrame(QString trame)
{
    //qDebug() << Q_FUNC_INFO << "trame" << trame;

    QStringList donneesTrame = trame.split("$");
    QString donneeTrame;

    //qDebug() << Q_FUNC_INFO << "trame" << donneesTrame;
    for(int i = 1; i < donneesTrame.size();i++)
    {
        donneeTrame = donneesTrame.at(i);
        // index ?
        switch(donneeTrame.at(0).toAscii())
        {
        case INDEX_Elevation:
```

```
            donneeTrame.remove(0, 1); // on retire l'index
            donneesSuiveur.angleElevationMoteurB = donneeTrame.toDouble();
            break;

        case INDEX_Angle_horaire:
            donneeTrame.remove(0, 1);
            donneesSuiveur.angleHoraireMoteurA = donneeTrame.toDouble();
            break;
```

Cas d'utilisation « Paramétrer le fonctionnement »



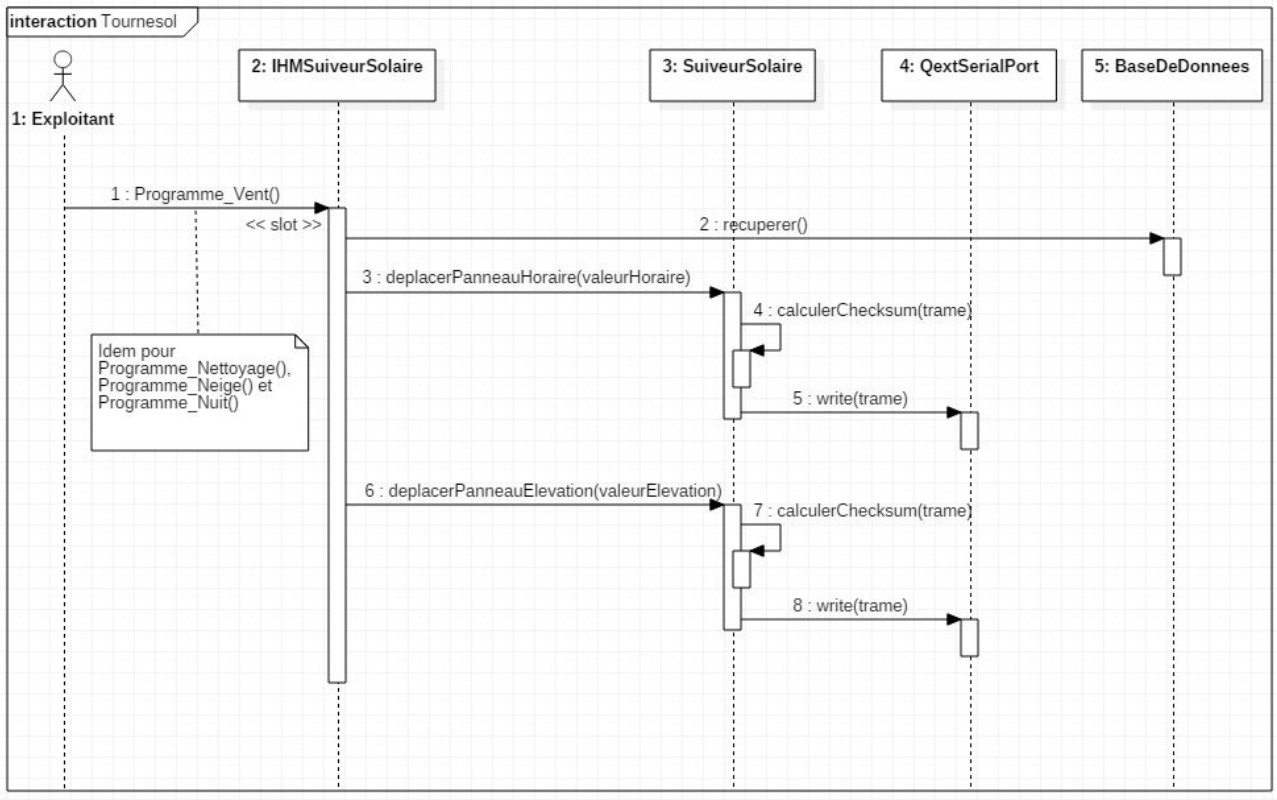


Diagramme Séquence : Modélisation du programme vent

Lorsque l'exploitant clique sur le bouton « Programme Vent», cela déclenche le *slot Programme_Vent()* qui s'occupe de récupérer les données correspondant au programme vent dans la base de données. Puis elle envoie une par une les commandes de déplacement, une pour l'angle horaire avec la méthode *deplacerPanneauHoraire* de la classe ***SuiveurSolaire***, et ensuite pour l'angle d'élévation la méthode *deplacerPanneauElevation* elle aussi de la classe ***SuiveurSolaire***.

```
   void IHMSuiveurSolaire::Programme_Vent()
  {
    QStringList commande;
    QString requeteProgramme = "SELECT  idSuiveur, type, angleHoraire,
angleAzimut, positionMoteurA, positionMoteurB FROM Positions WHERE type =
'vent'";
    qDebug() << Q_FUNC_INFO << commande;
    bdd->recuperer(requeteProgramme, commande);
    //qDebug() << commande ;
    int valeurHoraire;
        valeurHoraire = commande.at(2).toDouble();
        qDebug() << valeurHoraire ;
    suiveurSolaire->deplacerPanneauHoraire(valeurHoraire);
    sleep(1);
```

```
 int valeurElevation;
            valeurElevation = commande.at(3).toDouble();
            qDebug() << valeurElevation;
      suiveurSolaire->deplacerPanneauElevation(valeurElevation);
  }
```

Pour les fonctions de déplacement en horaire et élévation, celles-ci fonctionnent comme la méthode *deplacerPanneau()* mais au lieu de se déplacer dans les directions (haut, bas, gauche et droite), celle-ci va faire en sorte que le panneau se déplace jusqu'à être orientée selon l'angle donné.

```
void SuiveurSolaire::deplacerPanneauHoraire(double angleHoraire)
{
    QString trame;
    unsigned char checksum;

    trame = "$" + QString(INDEX_Angle_horaire) + QString::number(angleHoraire,
'f', 1);

    checksum = calculerChecksum(trame);

    //+= "Concatenation"
    trame += QString(checksum);

    qDebug() << Q_FUNC_INFO << "trame Azimuts" << trame;

    port->write(trame.toLatin1());
}

void SuiveurSolaire::deplacerPanneauElevation(double angleELevation)
{
    QString trame;
    unsigned char checksum;

    trame =  "$" + QString(INDEX_Elevation) + QString::number(angleELevation,
'f', 1);

    checksum = calculerChecksum(trame);

    //+= "Concatenation"
    trame += QString(checksum);

    qDebug() << Q_FUNC_INFO << "trame Elevation" << trame;

    port->write(trame.toLatin1());
}
```

Tests de validation

Ce que la méthode doit faire	Ce que la méthode fait	Validation
Utiliser la classe *QextSerialPort* pour recevoir une trame du suiveur solaire	Réception de la trame du suiveur solaire	OUI
Réceptionner la trame et décoder la trame ainsi que distinguer chacune des commandes	La trame est reçue et décodée par la méthode *décoderTrame()*	OUI
Afficher les données relatives au suiveur solaire en temps réel	Les données sont affichées périodiquement	OUI
Les programmes sont enregistrés et réutilisables	Les programmes sont enregistrés dans la base de données et peuvent être exploités	OUI
Déplacer le panneau dans les directions haut, bas, gauche, droite	Le panneau se déplace dans la position voulue	OUI
Déplacer le panneau avec un angle	Le panneau se déplace selon l'angle sélectionné	OUI
L'intervalle de suivi du suiveur est modifiable	L'intervalle de suivi peut être modifié	OUI
L'heure et la date sont affichés périodiquement	L'heure respecte celle du suiveur solaire et la date correspond à celle du système	OUI

Conclusion

Le tableau ci-dessous récapitule les tâches qui ont été réalisées.

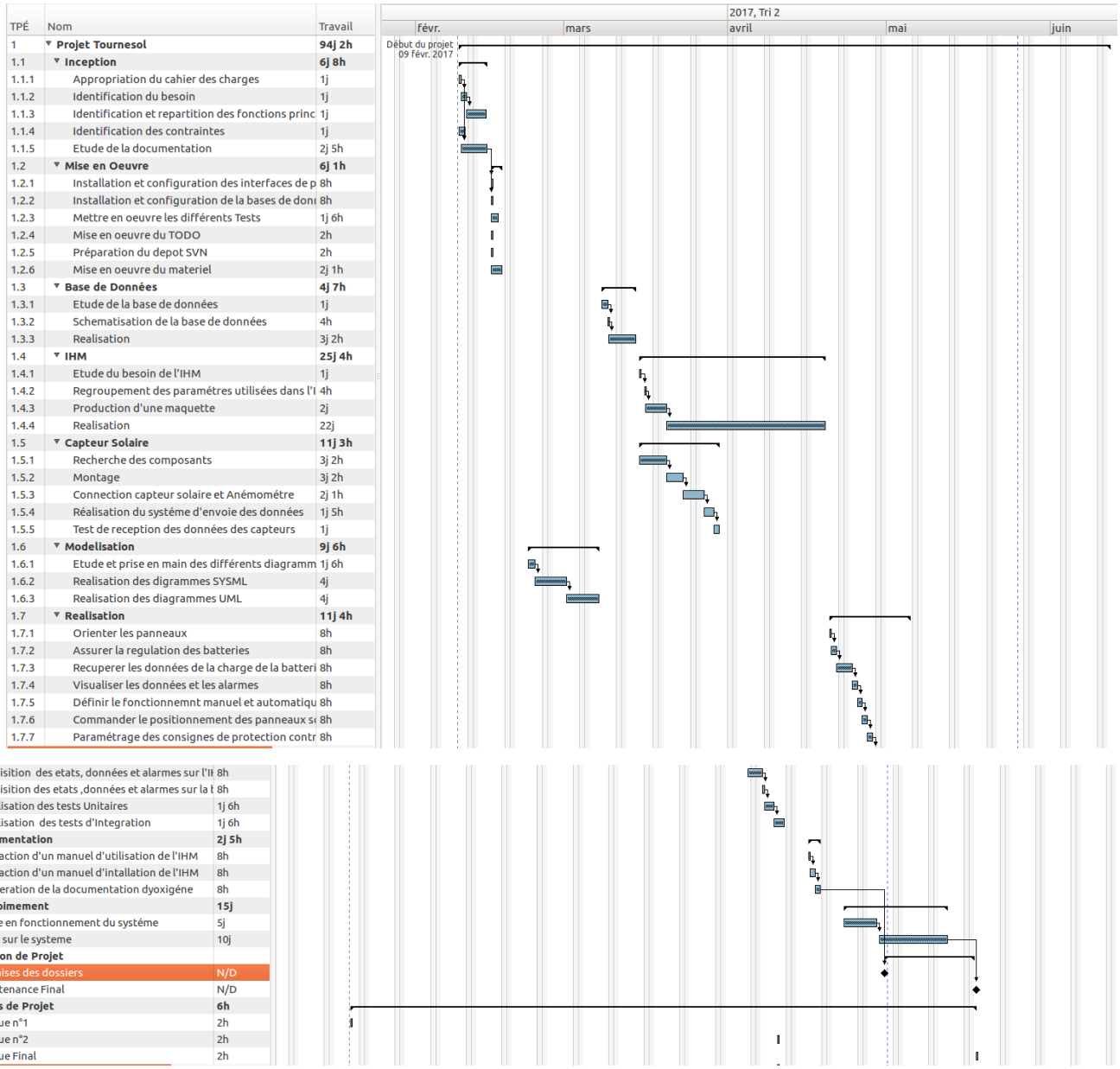
La plupart d’entre-elles on été réalisées sauf celle qui concerne l’utilisation du capteur solaire qui n’a pas pu être réalisée à temps pour l’ajout dans l’interface et le déplacement du panneau.

Nom de la tâche	OUI	NON
La date et l'heure sont affichées périodiquement	X	
L'état du suiveur solaire est visible sur l'IHM	X	
Le positionnement du suiveur solaire (angle d’azimut et d’élévation courants) sont affichés et mis à jour périodiquement	X	
Les données détaillées du suiveur solaire sont affichées périodiquement dans l'écran correspondant et archivées dans la base de données	X	
La commande manuelle d'orientation des panneaux est réalisable soit en fixant les angles d'azimut et d'élévation soit en dirigeant les quatre directions	X	
La commande manuelle d'orientation des panneaux est réalisable en sélectionnant une position pré-programmée (nettoyage, vent, nuit et neige)	X	
Les position pré-programmées (nettoyage, vent, nuit et neige) sont modifiables et enregistrées	X	

L'intervalle de suivi automatique est transmis au suiveur solaire et il est enregistré dans la base de données	X	
L'activation et la désactivation du suivi automatique est effective	X	
L'envoi des commandes du capteur solaire vers le suiveur solaire est réalisée		X
Le réglage des consignes de protection contre les vents violents est transmis au suiveur solaire		X

Annexes

1.1 GANTT



TPÉ	Nom	Travail
1	**Projet Tournesol**	**94j 2h**
1.1	**Inception**	**6j 8h**
1.1.1	Appropriation du cahier des charges	1j
1.1.2	Identification du besoin	1j
1.1.3	Identification et repartition des fonctions princ	1j
1.1.4	Identification des contraintes	1j
1.1.5	Etude de la documentation	2j 5h
1.2	**Mise en Oeuvre**	**6j 1h**
1.2.1	Installation et configuration des interfaces de p	8h
1.2.2	Installation et configuration de la bases de donı	8h
1.2.3	Mettre en oeuvre les différents Tests	1j 6h
1.2.4	Mise en oeuvre du TODO	2h
1.2.5	Préparation du depot SVN	2h
1.2.6	Mise en oeuvre du materiel	2j 1h
1.3	**Base de Données**	**4j 7h**
1.3.1	Etude de la base de données	1j
1.3.2	Schematisation de la base de données	4h
1.3.3	Realisation	3j 2h
1.4	**IHM**	**25j 4h**
1.4.1	Etude du besoin de l'IHM	1j
1.4.2	Regroupement des paramétres utilisées dans l'I	4h
1.4.3	Production d'une maquette	2j
1.4.4	Realisation	22j
1.5	**Capteur Solaire**	**11j 3h**
1.5.1	Recherche des composants	3j 2h
1.5.2	Montage	3j 2h
1.5.3	Connection capteur solaire et Anémométre	2j 1h
1.5.4	Réalisation du systéme d'envoie des données	1j 5h
1.5.5	Test de reception des données des capteurs	1j
1.6	**Modelisation**	**9j 6h**
1.6.1	Etude et prise en main des différents diagramm	1j 6h
1.6.2	Realisation des digrammes SYSML	4j
1.6.3	Realisation des diagrammes UML	4j
1.7	**Realisation**	**11j 4h**
1.7.1	Orienter les panneaux	8h
1.7.2	Assurer la regulation des batteries	8h
1.7.3	Recuperer les données de la charge de la batteri	8h
1.7.4	Visualiser les données et les alarmes	8h
1.7.5	Définir le fonctionnemnt manuel et automatiqu	8h
1.7.6	Commander le positionnement des panneaux s	8h
1.7.7	Paramétrage des consignes de protection contr	8h
1.7.8	Aquisition des etats, données et alarmes sur l'I	8h
1.7.9	Aquisition des etats ,données et alarmes sur la l	8h
1.7.10	Realisation des tests Unitaires	1j 6h
1.7.11	Realisation des tests d'Integration	1j 6h
1.8	**Documentation**	**2j 5h**
1.8.1	Redaction d'un manuel d'utilisation de l'IHM	8h
1.8.2	Redaction d'un manuel d'intallation de l'IHM	8h
1.8.3	Generation de la documentation dyoxigéne	8h
1.9	**Deploiement**	**15j**
1.9.1	Mise en fonctionnement du systéme	5j
1.9.2	Test sur le systeme	10j
1.10	**Gestion de Projet**	
1.10.1	Remises des dossiers	N/D
1.10.2	Soutenance Final	N/D
1.11	**Revus de Projet**	**6h**
1.11.1	Revue n°1	2h
1.11.2	Revue n°2	2h
1.11.3	Revue Final	2h

1.2 Bon de commande

SAT CONTROL®
Want to get more?

SAT CONTROL d.o.o.
POŽENIK 10,
SI-4207 CERKLJE
SLOVENIA
WWW.SOLAR-MOTORS.COM
INFO@SOLAR-MOTORS.COM
TEL: +3864-281-62-00
FAX: +3864-281-62-13

William Danthony
Lycee Technologique et Professionnel
9 Rue Notre Dame Des 7 Douleurs BP 50165

84008 Avignon Cedex
FRANCE

chefdetravaux@lasalle84.org

Tel.: +33 490 145 653

PRO-FORMA INVOICE:

No.:	**17 / 10868**
Page:	**1/1**
Date:	**17. marec 2017**
Validity:	**1. april 2017**
Terms of Delivery:	CPT
Payment Conditions:	T/T 100% deposit at order
Value:	EUR

Ser.	Product Code/Description	EAN	Quantity	UM	Price	Disc.	VAT	Amount
1	**POZSOLMICRO-D**	383106390441	1,00	PCS	160,00	15,00%	22,0%	160,00
	Solar Positioner 2-axis mod. MICRO-D, USB-RS485, DIN rail, w. lightning protecti							
2	**TRADHL1**	TRADHL1	1,00	SRV	18,20		22,0%	18,20
	DHL delivery costs							

	Amount:	**178,20**
	Discount:	**24,00**
Denote payment with code: **00-10868-502444146-99**	**VAT:**	**33,92**
	Total: EUR	**188,12**

Thanks for your inquiary. If there are some changes about the product, quality, delivery destination or delivery time, please ask for new quotation with changed conditions. The delivery time begins to run when we obtain payment of the deposit on our account. Deposit is not refundable if you cancel this order. Unless otherwise agreed in writing our business terms and conditions and warranty terms apply, published on our website: http://www.solar-motors.com/gb/general-business-terms-d14.shtml
Payment on:
Bank: ABANKA d.d. - Slovenska cesta 58, SI-1517 Ljubljana, Slovenia, IBAN:SI56 0510 0801 2559 067 SWIFT (BIC): ABANSI2X We do not accept any debeting comisions paid by SAT CONTROL d.o.o.

SAT CONTROL d.o.o.,Patricija Hudelja

Glossaire

• ***Élévation*** – angle vertical sous lequel le soleil est visible depuis un point d'observation sur la terre.

• ***Azimut*** – angle formé par le plan vertical du soleil et le plan méridien du point d'observation.

• ***Angle azimutal*** – angle formé par le plan vertical du soleil et le plan méridien.

• ***l'héliotropisme*** – Il s'agit d'un un mouvement diurne en réponse au changement de direction du soleil.

• ***Checksum*** – *ou la* **somme de contrôle** , est un nombre qu'on ajoute à un message à transmettre pour permettre au récepteur de vérifier que le message reçu est bien celui qui a été envoyé. L'ajout d'une somme de contrôle à un message est une forme de contrôle par redondance.

• ***Hexadécimale*** – Le **système hexadécimal** est un système de numération positionnel en base 16.

Tournesol

Revue finale

Étudiant 1 → non réalise, remplacé par le module POZSOL.
Étudiant 2→ Damien COUTANT(EC)
Étudiant 3→ Ayoub LOUDGHRI (IR)
Étudiant 4→ Alexis LELIEVRE (IR)

Version 1.0

Sommaire :

I-) Partie commune :

1) Cahiers de charges :

Présentation du projet :

On a besoin d'une alimentation électrique fiable pour alimenter les équipements nécessaires mais la présence d'un réseau électrique fonctionnel n'est pas toujours assurée. Si aucun réseau n'est disponible, un générateur serait le choix logique pour générer de l'électricité. Cependant, les générateurs sont bruyants et très polluants. De plus, les directives européennes et nationales encouragent fortement la création d'unités locales de production d'énergies renouvelables.

Le solaire fait partie de ces énergies considérée comme inépuisable, n'émettant potentiellement aucune particule nocive et de plus disponible en grande quantité sur Terre. Le solaire étant un secteur qui reste onéreux, il ressort l'importance d'améliorer le rendement des installations solaires, pour exploiter au mieux les ressources potentielles. Celui-ci pourrait être augmenté de deux manières : la première consisterait à améliorer techniquement la cellule photovoltaïque, la seconde à optimiser l'angle d'éclairement du panneau en fonction de la position du soleil.

Le solaire fait partie de ces énergies remises récemment au goût du jour. En effet, la théorie liée au photovoltaïque est ancienne puisqu'elle a été décrite par Hertz et Einstein au début du XXème siècle. L'énergie solaire est considérée comme inépuisable (compte tenu de la durée de vie du Soleil) et n'émet potentiellement aucune particule nocive (fabrication des panneaux pour la capter et des batteries pour la stocker mise à part). Elle est de plus disponible en grande quantité sur Terre. Il ressort l'importance d'améliorer le rendement des installations solaires, pour exploiter au mieux les ressources potentielles. Celui-ci pourrait être augmenté de deux manières : la première consisterait à améliorer techniquement la cellule photovoltaïque, la seconde à optimiser l'angle d'éclairement du panneau en fonction de la position du soleil.

En retenant la seconde solution, il s'agira désormais d'orienter efficacement les panneaux solaires photovoltaïques afin d'augmenter la production journalière d'électricité. Le système technique devra donc permettre de produire efficacement l'énergie électrique nécessaire pour alimenter des matériels portatifs ou satisfaire des besoins locaux en des lieux isolés.

Objectifs :

Il s'agit donc de réaliser un programme complet qui assure le fonctionnement autonome d'une installation photovoltaïque motorisée permettant d'optimiser la récolte d'énergie.

Le système devra :

- orienter efficacement les panneaux solaires photovoltaïques en toute sécurité pour optimiser la récolte d'énergie ;
- assurer la régulation de l'énergie et le contrôle sécurisé de la charge des batteries ;
- récupérer les informations de l'ensemble de l'installation et les stocker ;
- signaler et journalier les alarmes ;
- communiquer avec l'utilisateur.

A) Matériel :

La station de production d'énergie électrique sera composée de panneaux photovoltaïques motorisés et des équipements nécessaires au stockage et à la régulation de l'énergie :

- au moins deux panneaux photovoltaïques de type mono-cristallin
- au moins deux batteries VRLA (*Valve Regulated Lead Acid*)
- un régulateur de charge
- un système de commande de positionnement 2 axes
- une station météo (Anémomètre / Girouette / Thermomètre / Capteur de luminosité)
- un capteur de positionnement solaire

On distinguera les modules suivants :

- suiveur solaire (panneaux photovoltaïques, système de commande de positionnement, capteurs de positionnement solaire, anémomètre) ;
- régulateur de charge (liaisons avec les panneaux photovoltaïques, les batteries et les sorties à alimenter).

B) Contraires techniques et économiques :

Le système fonctionnera en régime continu sauf en cas de maintenance, lors d'un problème(surchauffe par exemple), le budget ne devra excéder 2000€ et sera à la charge de l'établissement.

2) Rôles du système :

Le système **Tournesol** devra remplir les missions suivantes pour exploiter l'installation :

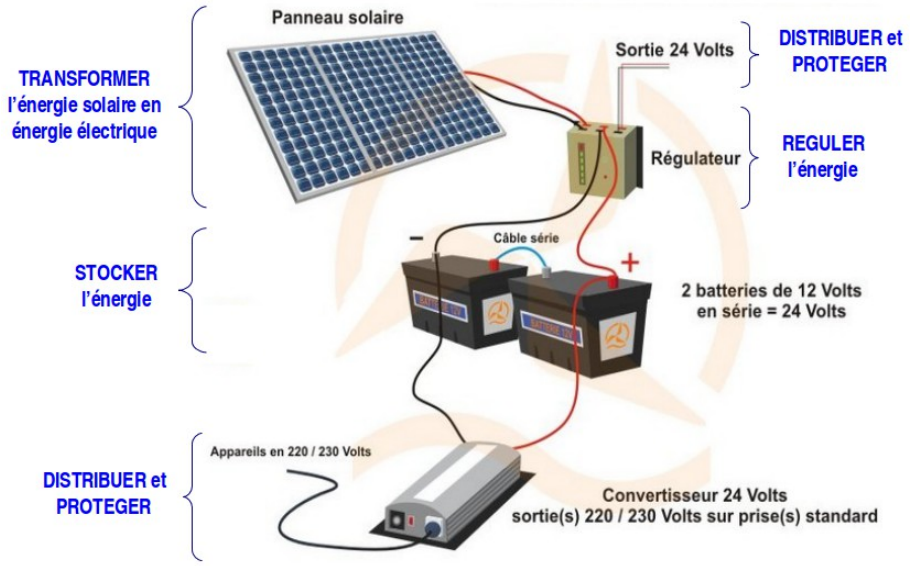
- Transformer l'énergie solaire produite par les panneaux photovoltaïques en énergie électrique ;
- Réguler l'énergie qui est réalisée par le régulateur de charge qui protège les batteries contre les risques de surcharges et les décharges au-dessous du seuil minimum (décharge profonde) ;
- Stocker l'énergie qui est réalisée par le parc de stockage constitué de batteries. Le parc de stockage permet de déphaser la production de la consommation de l'énergie et de palier aussi à des pointes de puissance ponctuelles ;
- Distribuer et protéger.

L'application embarquée devra donc assurer :

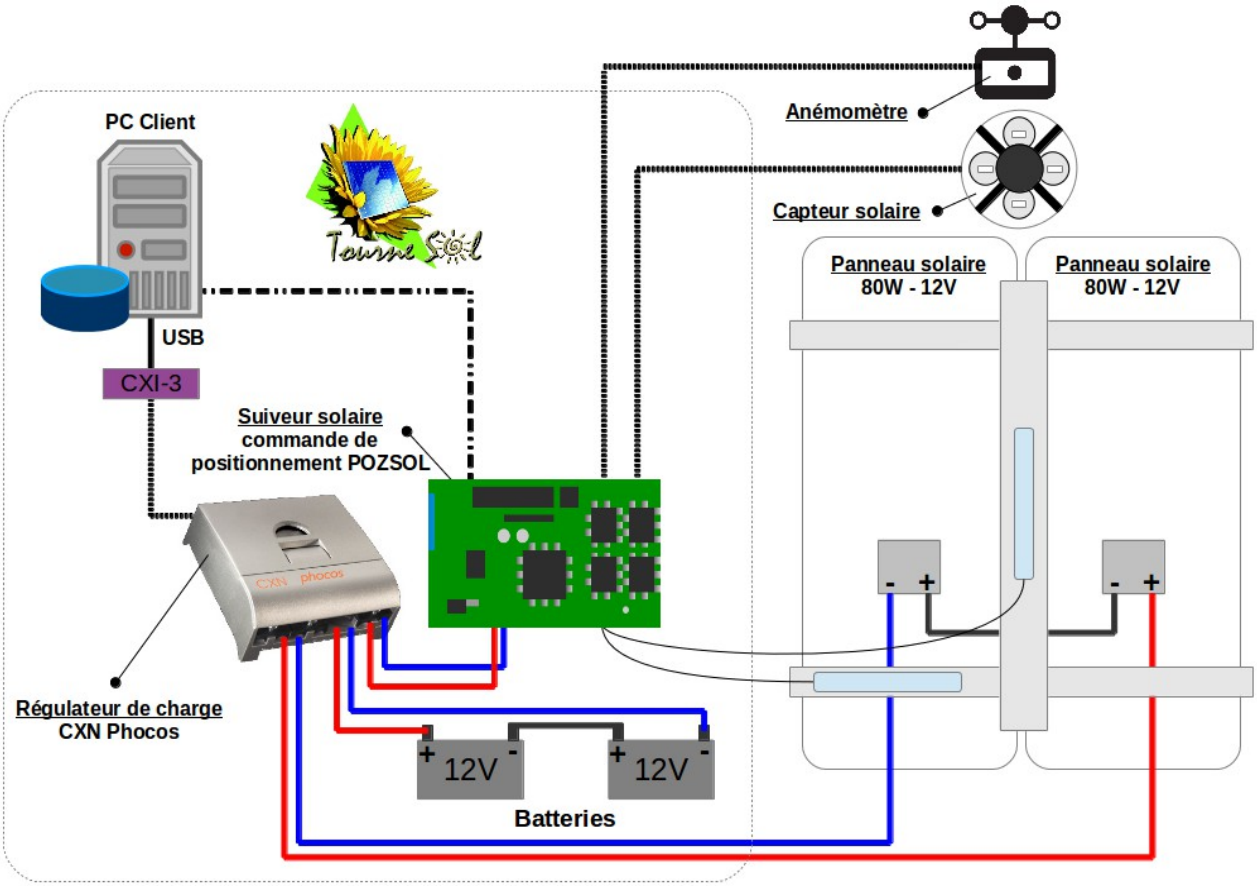
- Le fonctionnement en mode manuel ou automatique ;
- La commande du positionnement des panneaux solaires en toute sécurité ;
- Le pilotage du régulateur de charge;
- Le paramétrage des consignes de protection contre le vent ;
- L'acquisition des données des modules (suiveur solaire, régulateur de charge, et station météo) ;
- La surveillance des différents modules ;
- La visualisation des états, des données et des alarmes sur mini-écran tactile ;
- L'archivage des états, des données et des alarmes dans une base de données .

On distinguera les modules suivants :

- Suiveur solaire (panneaux photovoltaïque, système de commande de positionnement, des capteurs de positionnement solaire) ;
- Régulateur de charge (liaison avec les panneaux photovoltaïque, les batteries et les sorties ;
- Station météo (anémomètre, girouette, capteur de température, capteur de luminosité).



3) Configuration du système :



4) Diagrammes :

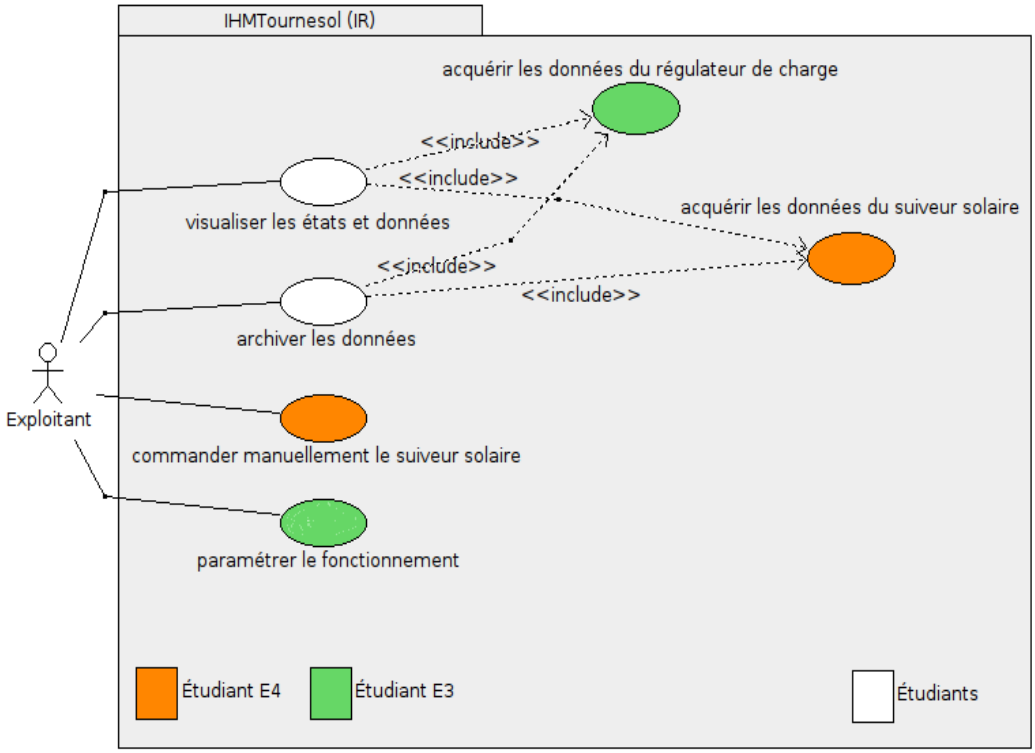


Diagramme du cas d'utilisations

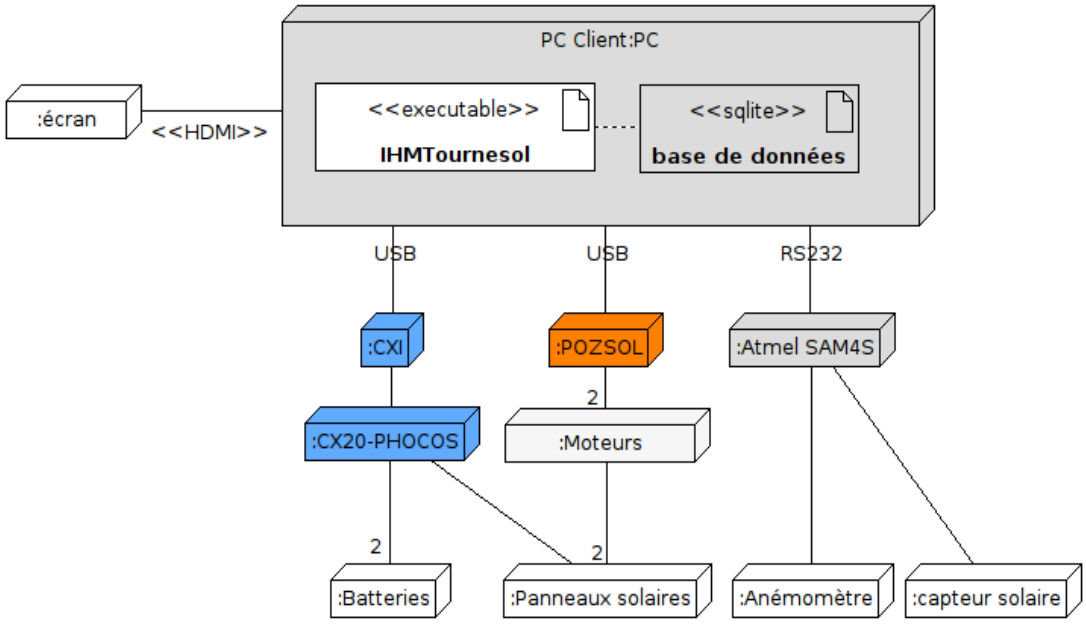


Diagramme de déploiement

TPÉ	Nom	Travail
1	**Projet Tournesol**	**94j 2h**
1.1	**Inception**	**6j 8h**
1.1.1	Appropriation du cahier des charges	1j
1.1.2	Identification du besoin	1j
1.1.3	Identification et repartition des fonctions princ	1j
1.1.4	Identification des contraintes	1j
1.1.5	Etude de la documentation	2j 5h
1.2	**Mise en Oeuvre**	**6j 1h**
1.2.1	Installation et configuration des interfaces de p	8h
1.2.2	Installation et configuration de la bases de donı	8h
1.2.3	Mettre en oeuvre les différents Tests	1j 6h
1.2.4	Mise en oeuvre du TODO	2h
1.2.5	Préparation du depot SVN	2h
1.2.6	Mise en oeuvre du materiel	2j 1h
1.3	**Base de Données**	**4j 7h**
1.3.1	Etude de la base de données	1j
1.3.2	Schematisation de la base de données	4h
1.3.3	Realisation	3j 2h
1.4	**IHM**	**25j 4h**
1.4.1	Etude du besoin de l'IHM	1j
1.4.2	Regroupement des paramétres utilisées dans l'I	4h
1.4.3	Production d'une maquette	2j
1.4.4	Realisation	22j
1.5	**Capteur Solaire**	**11j 3h**
1.5.1	Recherche des composants	3j 2h
1.5.2	Montage	3j 2h
1.5.3	Connection capteur solaire et Anémométre	2j 1h
1.5.4	Réalisation du systéme d'envoie des données	1j 5h
1.5.5	Test de reception des données des capteurs	1j
1.6	**Modelisation**	**9j 6h**
1.6.1	Etude et prise en main des différents diagramm	1j 6h
1.6.2	Realisation des digrammes SYSML	4j
1.6.3	Realisation des diagrammes UML	4j
1.7	**Realisation**	**11j 4h**
1.7.1	Orienter les panneaux	8h
1.7.2	Assurer la regulation des batteries	8h
1.7.3	Recuperer les données de la charge de la batteri	8h
1.7.4	Visualiser les données et les alarmes	8h
1.7.5	Définir le fonctionnemnt manuel et automatiqu	8h
1.7.6	Commander le positionnement des panneaux s	8h
1.7.7	Paramétrage des consignes de protection contr	8h
1.7.8	Aquisition des etats, données et alarmes sur l'I	8h
1.7.9	Aquisition des etats ,données et alarmes sur la l	8h
1.7.10	Realisation des tests Unitaires	1j 6h
1.7.11	Realisation des tests d'Integration	1j 6h
1.8	**Documentation**	**2j 5h**
1.8.1	Redaction d'un manuel d'utilisation de l'IHM	8h
1.8.2	Redaction d'un manuel d'intallation de l'IHM	8h
1.8.3	Generation de la documentation dyoxigéne	8h
1.9	**Deploimement**	**15j**
1.9.1	Mise en fonctionnement du systéme	5j
1.9.2	Test sur le systeme	10j
1.10	**Gestion de Projet**	
1.10.1	Remises des dossiers	N/D
1.10.2	Soutenance Final	N/D
1.11	**Revus de Projet**	**6h**
1.11.1	Revue n°1	2h
1.11.2	Revue n°2	2h
1.11.3	Revue Final	2h

Diagramme de gantt

5) Description des fonctions :

A) Fonction du régulateur de charge :

Il s'agira de contrôler la régulation d'énergie de l'installation. Le régulateur protège la batterie de toute surcharge du champ solaire et de décharges trop importantes dues à la surconsommation des charges de sortie.

L'exploitant définira le type de batteries (au plomb à liquide électrolyte ou VRLA de type GEL ou AGM) à contrôler et la fonction "coupure charge faible" en sélectionnant un mode parmi ceux proposés. Le contrôleur dispose de cinq modes destinés à éviter que la batterie ne se décharge complètement (voir document constructeur).

Il pourra aussi activer ou désactiver :

- le verrouillage de la programmation
- le verrouillage du signal sonore (buzzer) de niveau de charge de la batterie
- la charge de sortie raccordée à son installation (mode jour/nuit).

B) Fonction commande manuellement d'orientation des peanneaux :

Par l'écran , l'utilisateur pourra commander manuellement l'orientation des panneaux soit :

- en fixant les valeurs des angles d'azimut et d'élévation ;
- en pilotant les axes suivant 4 directions ;
- en sélectionnant une position pré-programmée (nettoyage, nuit et neige).

C) Fonction du suivi automatique :

L'objectif est de s'orienter vers le soleil (perpendiculairement au rayonnement) tout au long de la journée, ce qui aura pour effet d'augmenter la production d'énergie de manière significative. On suivra le soleil en utilisant deux axes : en azimut (d'est en ouest, à mesure de l'avancée de la journée) et en hauteur (selon la saison et, de nouveau, l'avancée de la journée). L'exploitant aura la possibilité de choisir le mode de positionnement du suiveur solaire : mode suivi automatique ou mode manuel.

Si l'exploitant a activé le suivi automatique, il devra préciser à quel intervalle de temps le suiveur solaire corrigera sa position pour suivre le soleil. Les valeurs possibles sont de 60 à 900 secondes (1-15 minutes).

D) Fonction de surveillance des alarmes :

Le système doit informer l'exploitant (alarme) en cas de problème :

- du suiveur solaire (erreurs de communication, de positionnement, de commandes d'axes, ...) ;
- du régulateur de charge (erreurs de communication, de surcharge, de surchauffe, ...) ;
- de la station météo (erreurs de communication, de mesures, ...) ;

Seuls les alarmes du suiveur solaire, seront répertoriées dans la base de données.

E) Fonction de paramétrage :

L'exploitant pourra aussi sélectionner :

– le verrouillage de la programmation
– le verrouillage du signal sonore (buzzer) de niveau de charge de la batterie
– le mode de suivi automatique
– l'activation du mode de protection contre le vent ;
– l'activation du suivi automatique.

L'exploitant pourra donc régler :

– l'intervalle de mise à jour du positionnement en secondes ;
– la vitesse de vent maximale en km/h et le temps d'attente en secondes avant un retour à un fonctionnement normal ;
– les positions pré-programmées (nettoyage, vent, nuit et neige).

F) Fonction d'acquisition des données :

Périodiquement, il sera réalisé une acquisition des données en provenance des différents modules :

• « suiveur solaire » :

– les angles A et B en degrés
– les positions A et B en pas
– les destinations A et B en pas
– les courants moteurs A et B en Ampères
– la tension en Volts
– le mode de suivi (manuel ou automatique)

• « régulateur de charge » :

– la tension en Volts
– le niveau de charge en %
– la fin de tension de charge en Volts
– le surplus d'énergie en %
– les courants photovoltaïque et charge en Ampères
– le niveau de charge le matin et le soir en % les consommations (panneaux, charge) et excédant en Ampères/h
– le type de batteries, le mode de charge, le nombre de semaines sans charge complète des batteries et la température.

6) Exigences :

Facteurs liés à l'environnement d'exploitation et d'utilisation	
Facteur	Signification
Couplage	capacité de liaison avec un autre logiciel
Efficacité	optimisation de l'utilisation des ressources
Maniabilité	facilité d'emploi pour l'utilisateur
Robustesse	conservation d'un fonctionnement conforme aux besoins exprimés, en présence d'événements non prévus ou non souhaités (arrêt normal, intempestif ou d'urgence)
Sécurité	disponibilité assurant la continuité des traitements

Facteurs liés à l'environnement de maintenance et de suivi	
Facteur	Signification
Adaptabilité	facilité de suppression, d'évolution de fonctionnalités existantes ou d'ajout de nouvelles fonctionnalités
Maintenabilité	facilité de localisation et de correction des erreurs résiduelles
Portabilité	minimisation des répercussions d'un changement d'environnement logiciel et matériel

7) Développement :

Environnement de développement	© `ATMEL Studio`
Système d'exploitation de développement	© `Microsoft Windows` (7 minimum)
Système de gestion de bases de données relationnelles	`SQLite3` (`sqliteman` ou `SQLiteManager`)
Environnement de développement du PCC et PCT	`Qt Creator` 3.6.1 et `Qt Designer`
Compilateur du PCC et PCT	`MinGW` (*Minimalist GNU for Windows*)
API GUI	`Qt` 5.6 et `Qwt` version 5 ou 6
Atelier de génie logiciel	`bouml` 4.23
Plate-forme de tests unitaires	`CppUnit`
Logiciel de gestion de versions	`subversion` (client `svn`)
Générateurs de documentation	`Doxygen` et `pandoc`
Gestionnaire de projet	`Planner` ou `gantter`

La modélisation devra s'effectuer par UML2. L'architecture logicielle sera orientée objet et multitâche. Le système d'exploitation reconnu est Linux. Pour la réalisation du développement on utilisera Qt Creator et Qt Designer pour la maquette. Le logiciel possédera une suite de tests unitaires cppunit. La documentation du code sera générée à partir de doxygen. Le serveur de base de données est MySql. Les requêtes seront exprimées en langage SQL. Pour la réalisation du gantt, on utilisera Planner ou Ganterr.

Le dépôt subversion sera organisé de la façon suivante :

- un répertoire TRUNK pour le développement courant,
- un répertoire TAGS pour le stockage des versions identifiées.
- un répertoire BRANCHES pour les tests et la correction des bugs constatés.

8) Gestion des modifications :

Les modifications liées à des erreurs dans la demande du client (cahier des charges) ou des changements de stratégies liés à l'environnement ou les conditions financières ou des choix technologiques, devront faire l'objet d'une procédure de demande de modification clairement définie et approuvée par le responsable du projet et du client.

La procédure de gestion des modifications se déroule en trois temps :

1. établissement d'une demande de modification (remplir la "Fiche de demande de modification" fournie)
2. prise en compte de la modification (validée par un professeur responsable du projet)
3. réalisation de la modification

Un **avenant** sera alors joint au dossier technique.

II)- Contrat individuel étudiant n°3(LOUDGHIRI Ayoub) :

Module du régulateur de charge :

1) Objectif de l'étudiant 3 :

En tant que technicien SN-IR, je dois concevoir une partie du projet tournesol, celle du régulateur de charge. Je dois pouvoir réguler l'énergie, paramétrer également le fonctionnement du régulateur de charge, c'est-à-dire choisir le mode de batterie, le mode de charge etc.. Et, surveiller les alarmes en cas de surcharge, ou surchauffe par exemple. Et, enfin acquérir les données, visualiser les états et les données, et pouvoir les archiver dans une base de données.

Ref.	Fonction de service	Critère	Niveau	Flexibilité
FP-2	Réguler l'énergie	Sortie	M/A	F2
FP-5	Surveiller les alarmes	Module à surveiller	régulateur de charge	F0
		Type d'alarme	visuelle	F0
FP-6	Paramétrer le fonctionnement	Choix des batteries	2 types	F2
		Fonction "coupure charge faible"	cinq modes	F2
		Programmation	verrou par l'écran tactile	F2
		Signal sonore (*buzzer*)	verrou par l'écran tactile	F2
FP-7	Acquérir les données	Périodicité	configurée par fichier	F0
		Module	régulateur de charge	F0
FP-8	Visualiser les états et données	Données	avec les unités	F0
		États	sous forme d'icône	F0
FP-9	Archiver les états, données et alarmes	Base de données	SQL	F0
		Horodatage mesures	aaaa-mm-jj hh:mm:ss	F0

Protocole attendu :

- Une application informatique fonctionnelle
- Un modèle UML complet de la partie à développer
- Le code source commenté de l'application
- Les documentations associées au module à produire

2) Recette minimale :

Nom de la tâche	OUI	NON
L'état du régulateur de charge est visible dans l'IHM		X
Les valeurs de la tension, de la charge et de la température des batteries sont affichées périodiquement et graphiquement sur l'écran d'accueil	X	
Les données détaillées du régulateur de charge sont affichées périodiquement dans l'écran correspondant et archivées dans la base de données		X
Les données de l'historique sont affichées sous forme de graphes dans l'écran correspondant et archivées dans la base de données		X
Les alertes signalées par le régulateur de charge sont affichées périodiquement dans l'écran correspondant et archivées dans la base de données		X
Le verrouillage et déverrouillage de la programmation par clavier et du signal sonore sont pris en compte		X
Le choix du type de batteries et le mode de coupure sont paramétrables et transmis au régulateur de charge		X

3) Présentation du travail de l'étudiant n°3 :

Ma partie consiste à créer la partie logicielle prenant en charge le régulateur de charge **PHOCOS-CX20** (voir présentation du matériel).

Cette partie logicielle consiste à envoyer des commandes au régulateur de charge (trames 0x20 ou x021) afin de recevoir des informations relevées par le régulateur de charge puis les décoder et les traiter. Ces données-là seront visualisés dans une IHM. Mon travail consiste également à gérer les alarmes en cas de surcharge ou décharge.

4) Tâches à réaliser :

Tâches à réaliser	Production attendue	Estimation horaire
S'approprier le cahier des charges	Le cahier des charges est explicité. Les tâches à réaliser sont identifiées et les ressources sont définies.	10 h
Rechercher des solutions issues de l'innovation technologique pour le stockage d'énergie	Une étude comparative argumentée des différentes solutions est produite.	6 h (SCP)
Installer et configurer son environnement de développement	Le poste de développement est opérationnel.	1h
Installer et raccorder les appareils	Les appareils fonctionnent. La procédure d'installation a été respectée. Un compte rendu est rédigé.	4h
Étudier et documenter les caractéristiques des batteries	Les batteries sont caractérisées à partir de l'étude et des mesures réalisées.	10 h (SCP)
Relier et paramétrer l'interface de communication	L'interface de communication est correctement paramétrée et fonctionnelle. Un plan de câblage est réalisé. Le rapport de tests de mise en œuvre est rédigé.	4h
Mettre en œuvre les programmes de test fournis	Le rapport de tests est renseigné.	14 h + 4 h (SCP)
Finaliser la modélisation UML du module	Les diagrammes UML (diagrammes de séquence « paramétrer le régulateur de charge », diagrammes d'états correspondants) sont élaborés et finalisés.	20 h
Produire la maquette de l'IHM du module	La maquette de l'IHM correspond aux exigences du cahier des charges.	8h
Coder et tester les classes du module	Les classes Regulateur, acquisitionRegulateur et la structure DonneesRegulateur sont codées et valides. La classe IHMTournesol est complétée et valide.	40 h
Réaliser les tests unitaires	Les tests unitaires des classes Regulateur et AcquisitionRegulateur sont écrits et archivés.	16 h
Faire la recette du module	Le cahier de recette du module est	4 h + 4 h (SCP)

	validé.	
Intégrer en équipe l'application complète	L'application est intégrée et fonctionnelle.	4h
Rédiger le dossier technique et les documents relatifs au projet	Le dossier est rédigé en respectant les exigences.	30 h
Produire un guide de mise en route et d'utilisation du module.	Un manuel est fourni.	4h
Gérer la planification	Le planning prévisionnel est établi. Le planning est actualisé avec une mise en évidence des écarts par rapport au prévisionnel.	10 h
Total		200 heures

5) Présentation du matériel exploité par l'étudiant n°3 :

5.1) Système du régulateur de charge PHOCOS-CX20 (pour kits solaires de 80Wc à 720Wc 12V/24V)

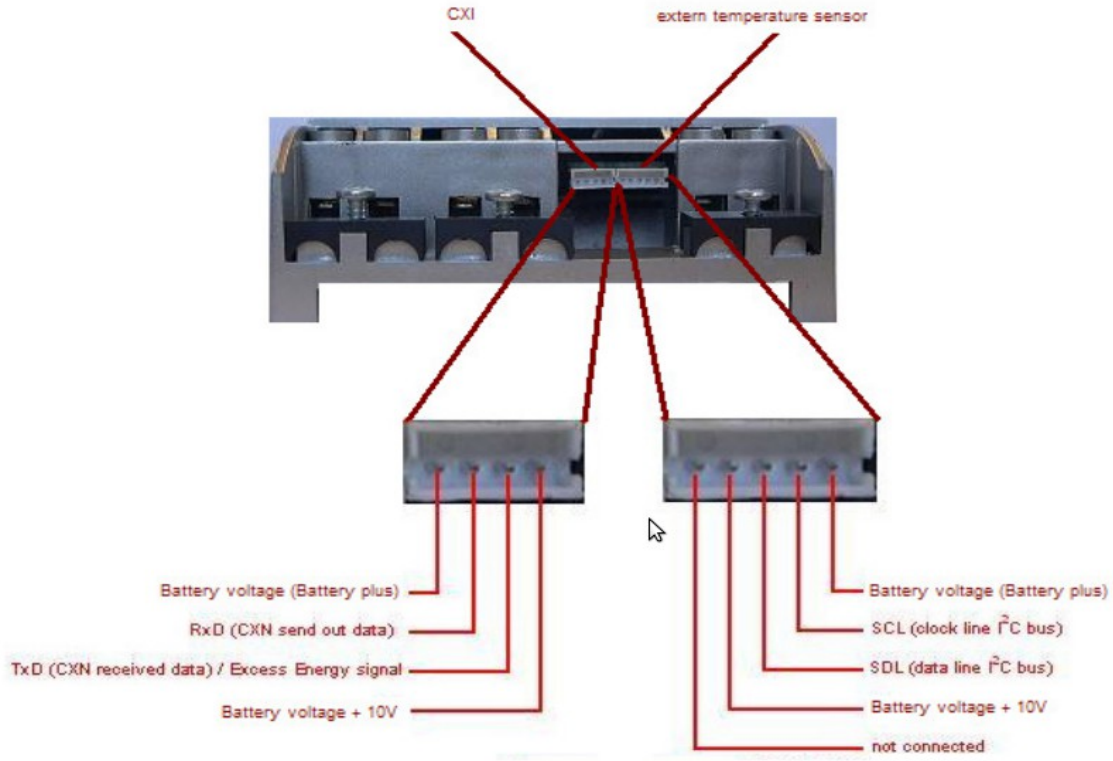
Le régulateur de charge protège les batteries contre les risques de surcharge et les décharges au-dessous du seuil minimum (décharge profonde). Il est en liaison avec les panneaux photovoltaïques, les batteries et les sorties.

Le régulateur protège donc la batterie de toute surcharge du champ solaire et de décharges trop importantes dues à la surconsommation des charges de sortie.

Le régulateur de charge peut définir le type de batteries (au plomb à liquide électrolyte ou VRLA de type GEL ou AGM) à contrôler et la fonction « coupure charge faible » en sélectionnant un des cinq modes. Ajouter à cela, on peut également activer ou désactiver le verrouillage de la programmation, le verrouillage du buzzer sonore de niveau de charge de la batterie, qui indique que la batterie est chargé ou déchargé. Le régulateur de charge fournit également :

- la tension en volts
- la fin de la tension de charge en volts
- le niveau de charge en % le matin, et le soir
- le type de batteries, le mode de charge (BOOST ou EQUAL)

5.2) Interface de communication CXI :



L'interface CXI est relié par une une liaison USB à un PC. Cette liaison USB sera pris en charge tel qu'un port série virtuel.

Le paramétrage du port série sera le suivant :

- Débit : 9600 bits/s
- Données : 8 bits
- Parité : aucune
- Nombre de bits de Stop : 1

5.3) Batteries :

Nous avions deux types de batteries (VRLA ou AGM) :

VRLA est l'abréviation de « Valve Regulated Lead Acid »

AGM est l'abréviation de « Absorbent Glass Mat »

6) Tests de unitaires :

Méthode acquerir()

Ce que la méthode doit faire	Ce que la méthode fait	Validation (oui/non)
Utiliser la méthode envoyer() de la classe port régulateur transmettre la commande 0x20.	Utiliser la méthode envoyer() de la classe port régulateur transmettre la commande 0x20.	OUI
Utiliser la méthode recevoir() de la classe port pour recevoir la trame 0x20 du phocos-cx20.	Utiliser la méthode recevoir() de la classe port pour recevoir la trame 0x20 du phocos-cx20.	OUI
Utiliser la méthode decoderActualData() et décoder la trame reçu.	Utiliser la méthode decoderActualData() et décoder la trame reçu.	OUI
Transmettre les informations vers le BDD.	/	NON
nUtiliser la méthode envoyer() de la classe port régulateur transmettre la commande 0x21.	Utiliser la méthode envoyer() de la classe port régulateur transmettre la commande 0x21.	OUI
Utiliser la méthode recevoir() de la classe port pour recevoir la trame 0x21 du phocos-cx20.	Utiliser la méthode recevoir() de la classe port pour recevoir la trame 0x21 du phocos-cx20.	OUI
Utiliser la méthode decoderDataLoggerData() et décoder la trame reçu.	Utiliser la méthode decoderDataLoggerData() et décoder la trame reçu.	OUI
Transmettre les informations vers le BDD.	/	NON

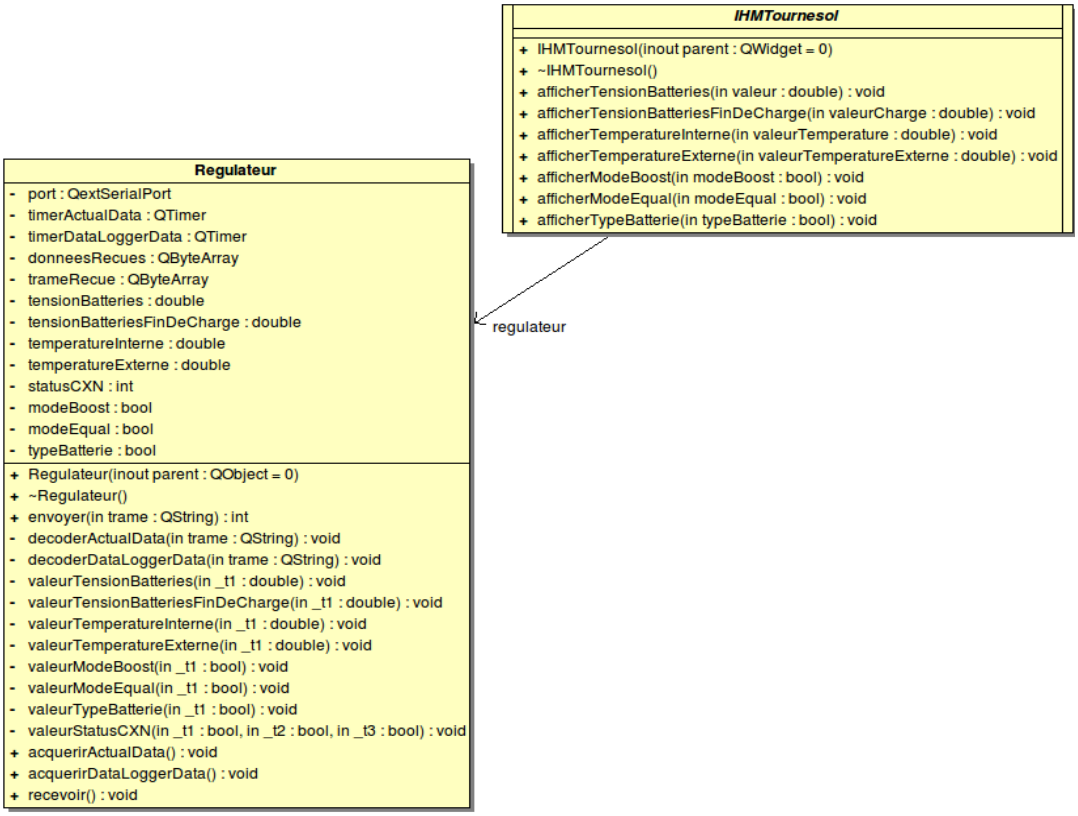
Méthode decoderActualData()

Ce que la méthode doit faire	Ce que la méthode fait	Validation (oui/non)
Identifier les différents champs de la trame d'information de base du PHOCOS-CX20	Identifier les différents champs de la trame d'information de base du PHOCOS-CX20	OUI
Effectuer le traitement (décodage) des données recueillis.	Effectuer le traitement (décodage) des données recueillis.	OUI
Transmettre les données décodé à l'IHM.	Transmettre les données décodé à l'IHM.	OUI

Méthode decoderDataLoggerData()

Ce que la méthode doit faire	Ce que la méthode fait	Validation (oui/non)
Identifier les différents champs de la trame de journalisation.	Identifier les différents champs de la trame de journalisation.	OUI
Effectuer le traitement (décodage) des données recueillis.	Effectuer le traitement (décodage) des données recueillis.	NON
Transmettre les données décodé à l'IHM.	Transmettre les données décodé à l'IHM.	NON

7) Diagramme de classe :



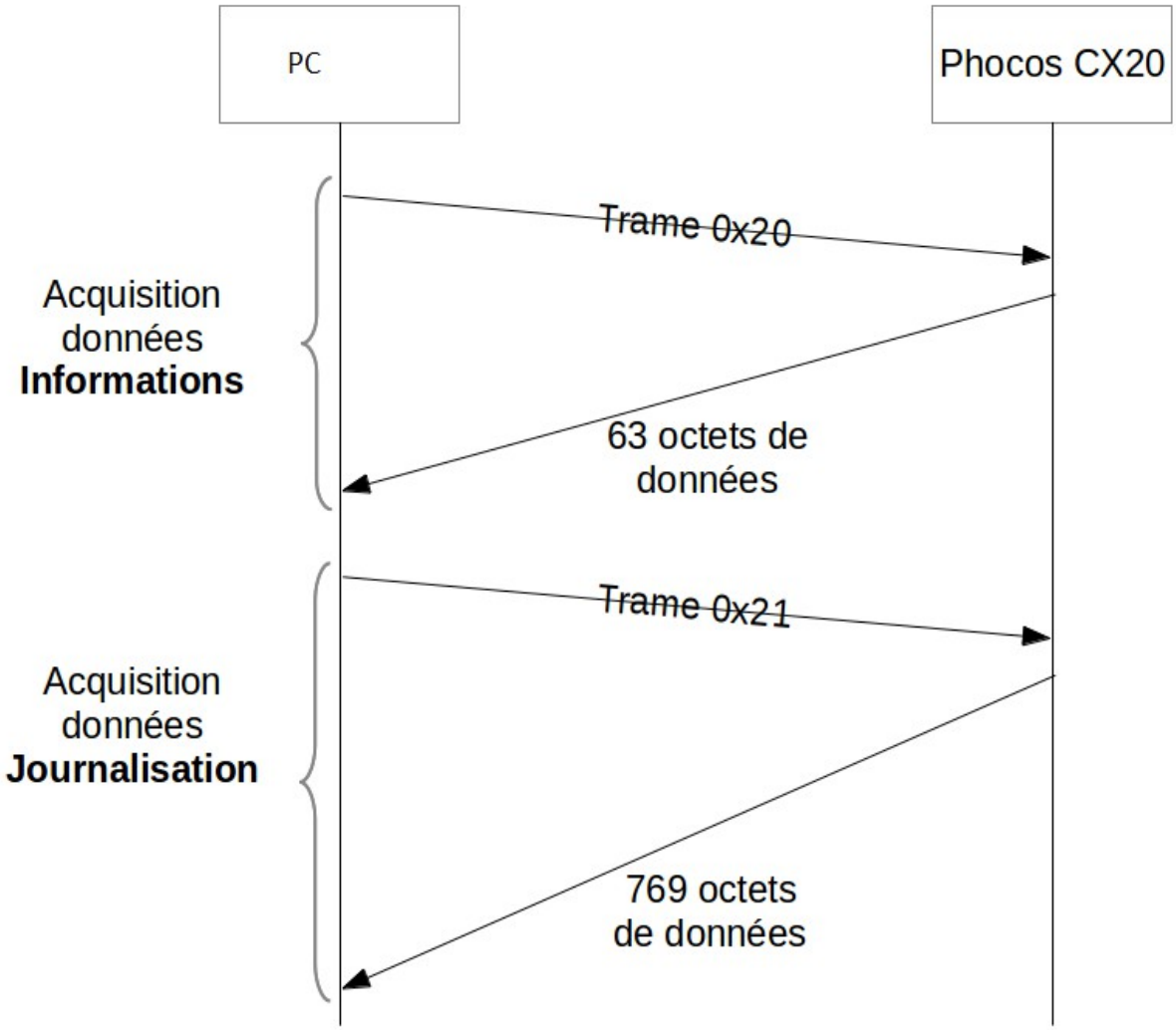
J'ai donc deux classes, régulateur qui permet d'acquérir une trame et décoder celle-ci, puis IHMTournesol qui affiche les données reçus par la classe régulateur, c'est pour cela qu'on a une associsation.

7.1) Acquérir les données du régulateur de charge :

La classe **regulateur** possèdes deux méthodes qui sont **acquerirActualData()** ; et **acquerirDataLoggerData()** ;
Elles appellent toutes les deux la méthode **envoyer « »()**; pour la *trame 0x20*, et **envoyer « ! »() ;** pour la *trame 0x21.* Puis elle appelle la méthode **recevoir() :** de la classe port afin de recevoir la trame en question. -Ensuite pour la trame0x21 elle appelle la méthode **decoderActualData() ;** afin de traiter cette trame et d'en extraire puis de calculer les données qu'elle contient. Tandis, que pour la trame 0x21 elle appelle la méthode **decoderDataLoggerData() ;**

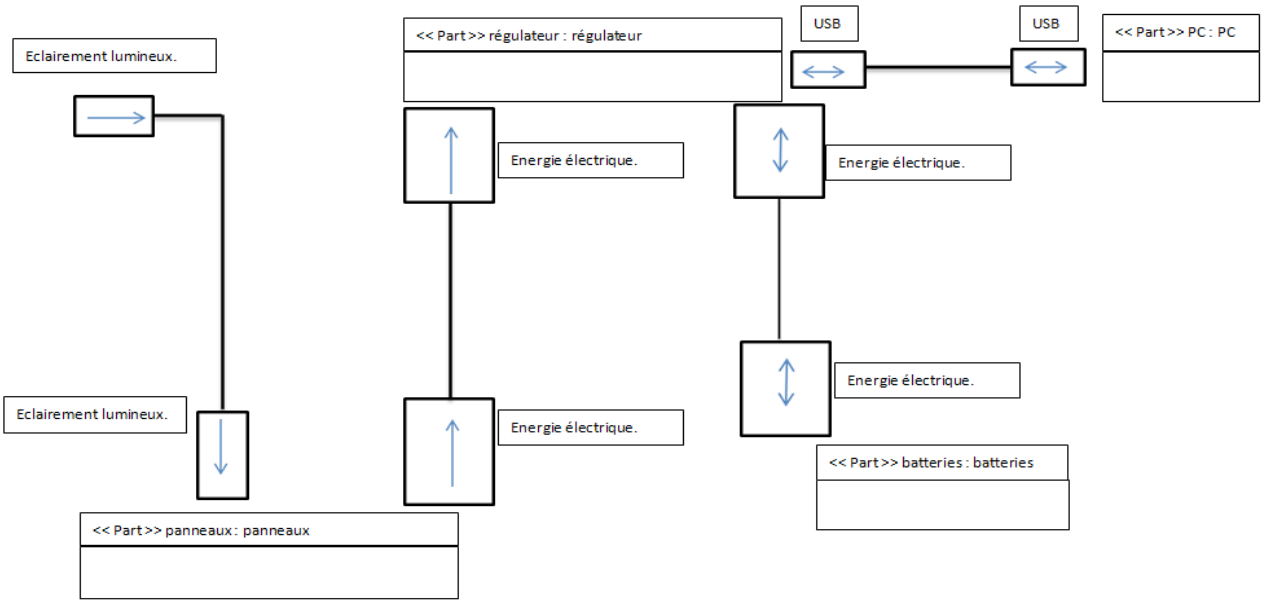
Les données recueillies par les méthodes de la classe Regulateur sont ensuite envoyé vers l'IHM par un système de signals/slots sous la forme de structures.

8) Communication avec le PHOCOS-CX20 :

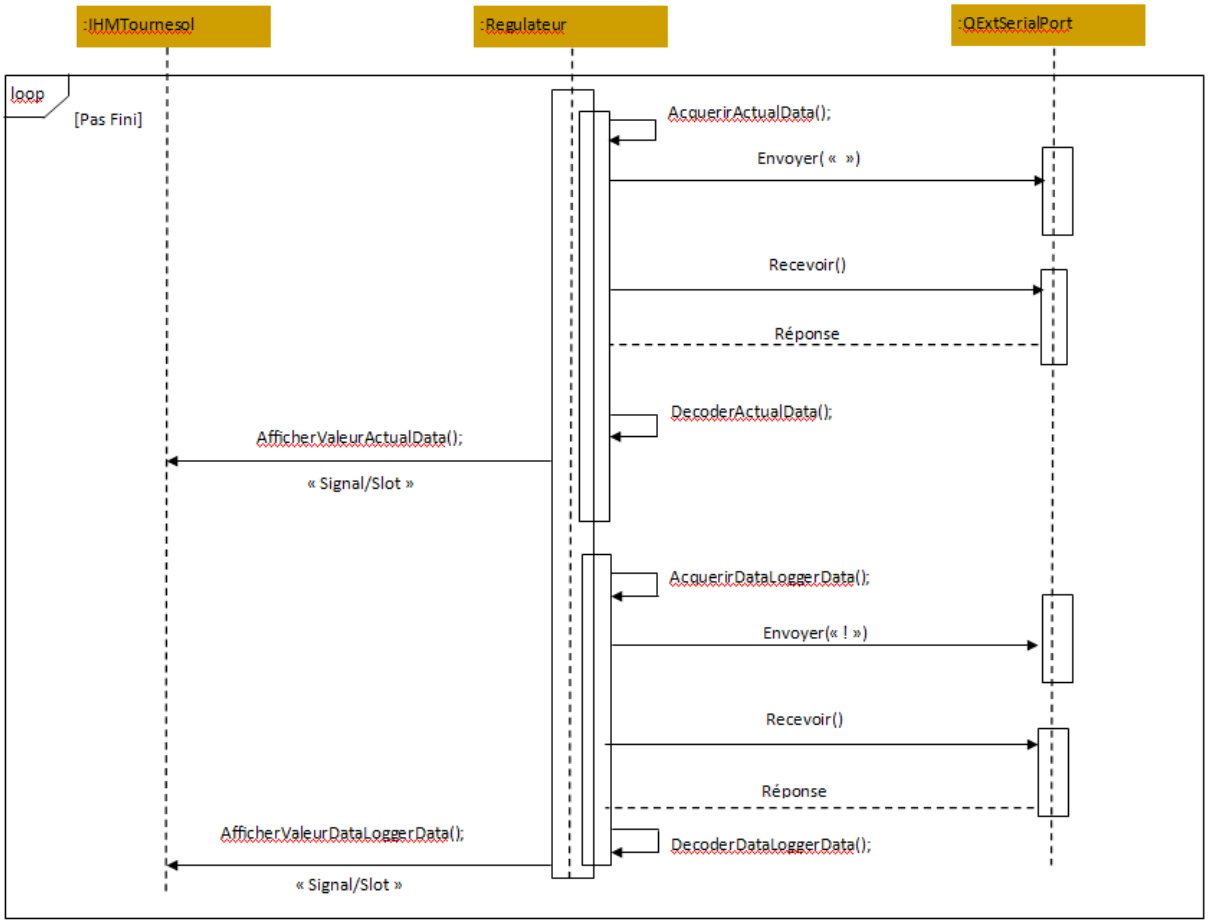


9) Diagramme de bloc interne :

Dans le diagramme ci-dessous, on peut voir l'interaction entre les différents matériels (PC, panneaux, régulateur de charge, batteries).



10) Diagramme de séquence :



11) Décodage de la trame du régulateur de charge :

Le protocole est basé sur un système de requête-réponse. Le logiciel interroge le PHOCOS en envoyant les requêtes de commande.

11.1) Trame ACTUAL DATA

La trame ACTUAL DATA se distingue par 0x20 qui correspond à l'espace en caractère ASCII. Pour recevoir la trame ACTUALDATA, on envoie un espace. La trame ACTUALDATA fait 65 octets et contient essentiellement des informations de bases contenus dans le régulateur de charge tels que que la tension des batteries, le niveau de charge en %, le niveau de charge le matin et le soir en %, le type de batteries, le mode de charge également. Pour ce type de trame, c'est le caractère espace en ASCII qui indique le début de la trame.

Exemple de trame ACTUAL DATA :

064 023 003 146 160 000 000 004 000 120 000 +000 +000 229

Chaque valeur correspond à une valeur du régulateur, par exemple le champ numéro 4 correspond au voltage des batteries.

Le champ 1 correspond à la version du CXN.
Le second champ 4 correspond au voltage de la batterie, tandis que le champ 5 à la tension de la batterie à la fin de charge.
Le champ 6 correspond à une valeur décimal qu'il faut convertir en binaire, donc ici 00000000, le bit 0 correspond au mode boost, s'il est sur 1cela veut dire que le mode est activé. Le bit 1 sur le mode equal, et le bit 2 sur le système de batterie (12 ou 24V).

Pour obtenir la tension de la batterie par exemple, il faut faire 2 * (146 * 0,032) + 9, à partir de là, on obtient la tension de la batterie en V.

11.2) Trame DATA LOGGER DATA

La trame DATA LOGGER DATA se distingue par 0x21 qui correspond à un point d'exclamation en caractère ASCII. Pour recevoir la trame DATA LOGGER DATA, on envoie un !. La trame DATA LOGGERDATA fait 769 octets, et contient toutes la journalisation des données du PHOCOS tels que la tension de la batterie le premier jour, de la dernière semaine jour par jour etc...

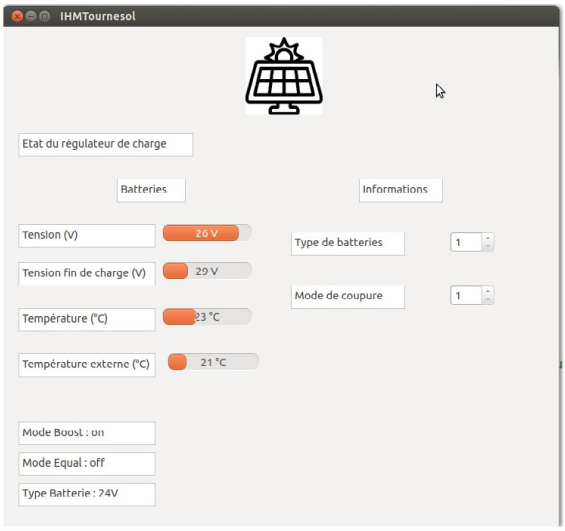
Exemple de trame DATA LOGGER DATA :

!F5 AF 6A 08 0B D7 00 01 1F 7D 0A 03 01 08 8E FF FF FF 28 12 00 07 00 00 04 0C 00 02 A0 00 01 F0 00 5C 60 5E 00 00 05 00 00 22 02 5E 5D 00 00 05 00 00 22 02 5E 5E 00 00 05 00 00 11 02 60 5E 00 00 05 00 00 22 02 62 61 00 00 05 00 00 44 02 5F 5E 00 00 05 00 00 22 02 70 70 00 00 05 00 00 BB 02 70 70 00 00 05 00 00 61 05 05

Ici pour le champ 14, ça correspond aux données du mode de coupure, le type de batterie, et le buzzer(on/off), le champ 14 est : 08
Il faut le convertir en décimal : 0000 1000
Les bits 0, 1 et 2 sont pour les modes de coupures, le bit 3 pour le type de batteries, s'il est sur 0 ça correspond à un type liquid electrolye, tandis que le 1 sur GEL (VRLA), pour le bit 4 c'est le buzzer, 0 indique qu'il est désactivé, 1 indique qu'il est activé.

Pour ici, on a 000 pour les modes de coupures qui indiquent que la tension est faible et s'arrête entre 11,4 et 11,9 V. Le type de batterie est sur 0 et donc indique le type GEL, et le bit 4 est sur 0 et indique que le buzzer est désactivé.

12) Maquette :



Dans l'IHM, nous retrouvons les valeurs décodés par nos méthodes. On peut voir la tension de la batterie, la température, ainsi que le mode activé pour le mode de charge des batteries. Le type de batteries, et le mode de coupure y est également.