

Partie personnelle

de

Killian LABATTUT

**Serre Automatique**

Système d’acquisition

# Dossier Technique - Partie Personnelle

Table des matières

[Dossier Technique - Partie Personnelle 2](#_Toc9429232)

[Situation dans le projet 3](#_Toc9429233)

[La tâche de l’étudiant dans le contexte du projet 4](#_Toc9429234)

[Diagrammes SysML 5](#_Toc9429235)

[Diagramme de déploiement 7](#_Toc9429236)

[Présentation des logiciels utilisées 8](#_Toc9429237)

[Le programme de la carte de gestion (Raspberry) 8](#_Toc9429238)

[Classe Serial 8](#_Toc9429239)

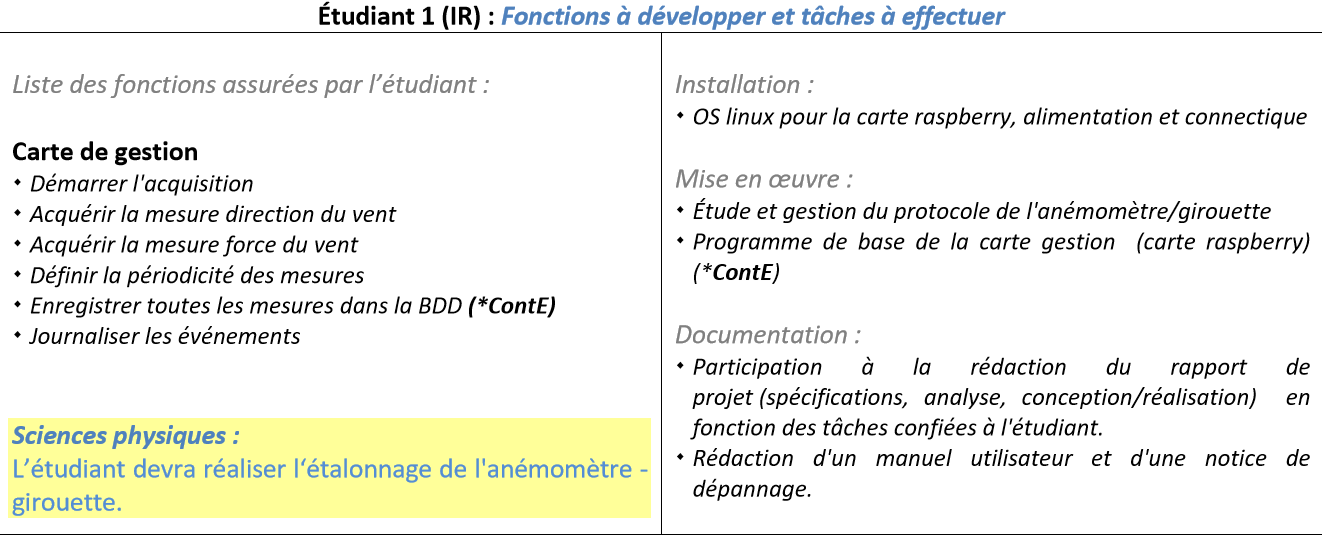
[Classe BDD 9](#_Toc9429240)

[Le programme de la carte d’acquisition (Arduino) 10](#_Toc9429241)

[Classe Periode 10](#_Toc9429242)

[Boucle principale 11](#_Toc9429243)

## Situation dans le projet



* Démarrer l’acquisition,

Au sein du projet, j’ai eu pour mission de faire échanger les cartes de gestion et d’acquisition afin de récupérer les valeurs des sept capteurs présents dans la serre. Pour se faire un programme est codé en langage python et interprété par la Debian 8.7 installée.

* Acquérir les mesures direction et force du vent,

Le station météo La Crosse « Weather Station La Crosse WS 3650 » m’a été confié pour réaliser les captures de direction et de force du vent. Cette station est un module qui combine les deux capteurs, ainsi les données sont transmises ensemble par un BUS RS232 et un câble RJ-11.

* Définir la périodicité des mesures,

La périodicité est récupérée par le programme d’acquisition depuis la base de données et est directement modifiable via l’application.

* Enregistrer toutes les mesures dans la BDD,

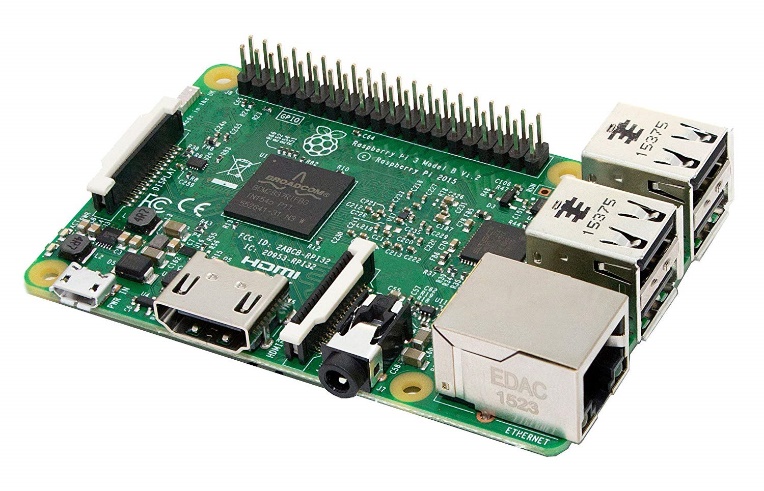
Les mesures sont « effectuées » ou non par la carte d’acquisition Arduino Uno, d’après le calcul de périodicité.

* Journaliser les évènements,

Ils sont écrits dans un fichier dit « fichier log » ainsi toutes traces d’envoi de données dans la base d’erreurs d’envoi est justifié.

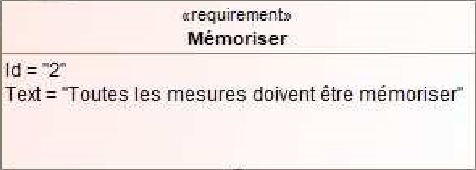
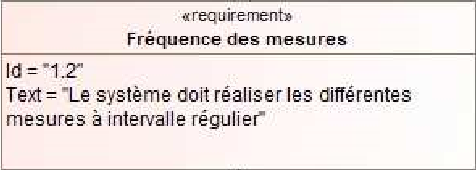
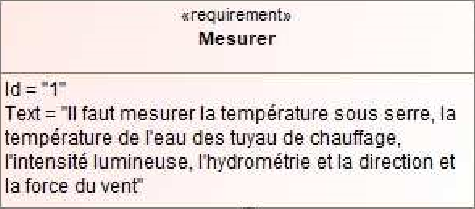
## La tâche de l’étudiant dans le contexte du projet

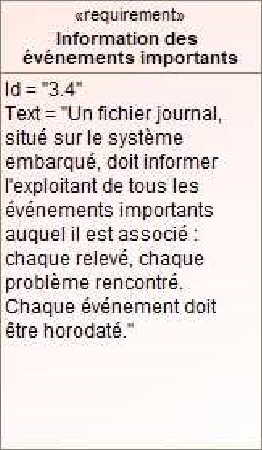
Carte de Gestion Carte d’acquisition



**Capteurs**

**Capteurs**

À l’aide du diagramme d’exigences formulées par le Groupe Olivier, j’ai extrait les différentes exigences qui me concernent affichés ci-dessous.

J’ai réfléchi aux à la construction des programmes, et je me suis décidé de programmer la Raspberry à demander constamment les valeurs des sept capteurs. Le message contient le numéro du capteur ainsi que le temps en secondes écoulé depuis 1970 et la période d’acquisition du capteur.

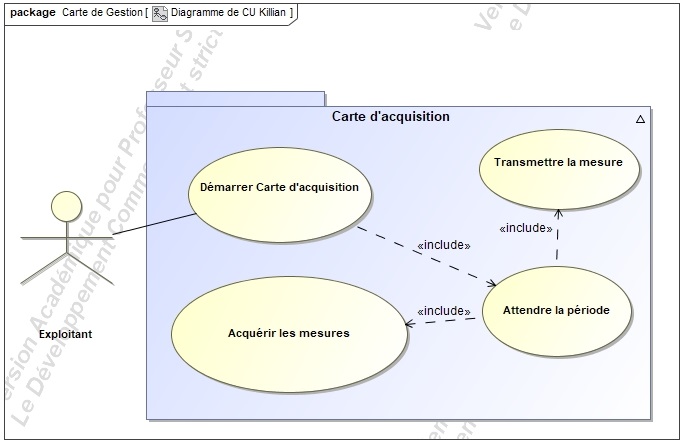
Pendant ce temps la carte d’acquisition reçoit le message et traite les données reçus elle fait la différence entre le temps du message et le dernier temps reçu. S’il est égal à la période d’acquisition transmise, la valeur du capteur et son id sont envoyé à la Raspberry qui enregistre dans le fichier « log » envoi la valeur dans la base de données.

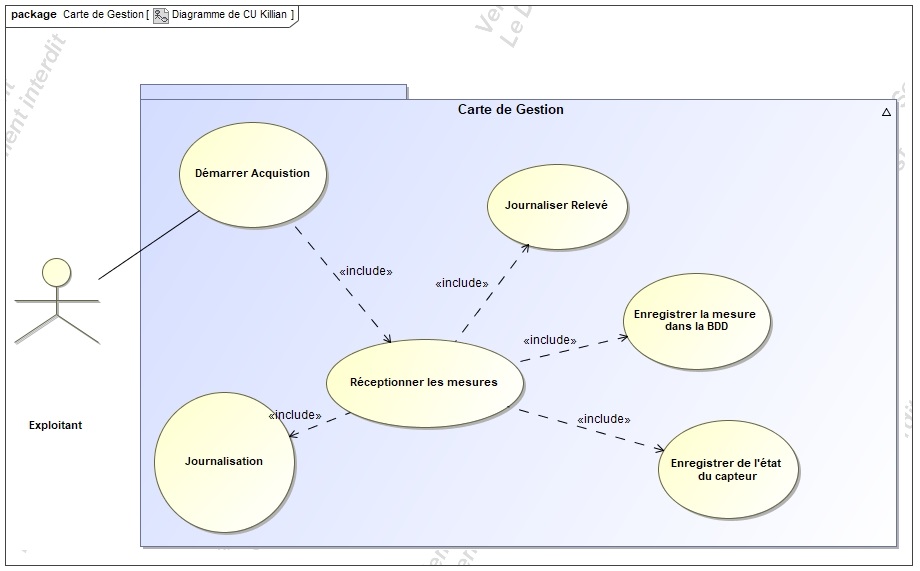
Pour pouvoir répondre aux exigences j’ai réalisé différents diagrammes UML afin de détailler ma partie.

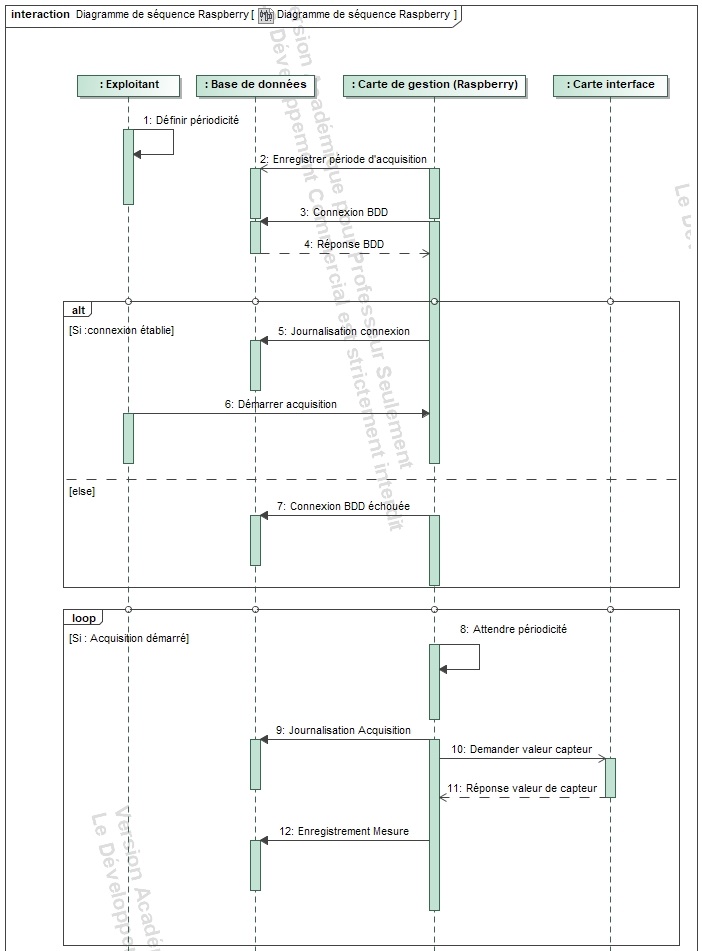
## Diagrammes SysML

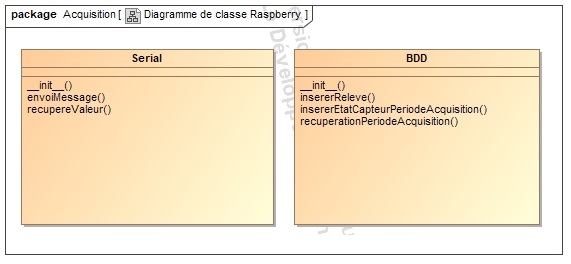
Voici les diagrammes de cas d’utilisation de la carte de Gestion et de la carte d’Acquisition.

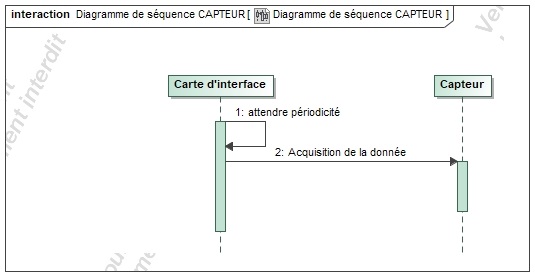
Les diagrammes représentent les deux systèmes et les actions qu’ils effectuent.





Le diagramme de séquence, il affiche l’ordre chronologique les interactions réalisés entre les systèmes.





## Diagramme de déploiement

Voici le diagramme de déploiement, on peut voir en jaune la partie donc je suis en charge dans le projet.

## Présentation des logiciels utilisées

Les logiciels que j’ai utilisés sont :

Arduino 1.8.8

MagicDraw UML 17.0.3

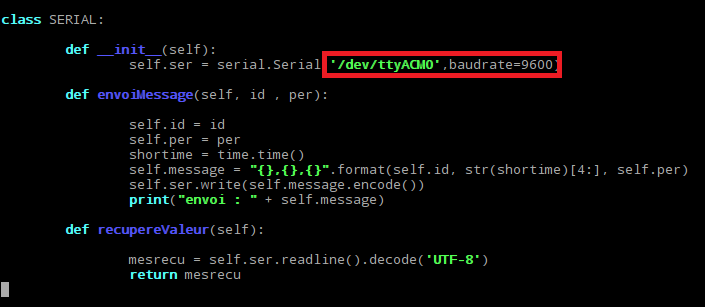
Python 3 (IDLE)

## Le programme de la carte de gestion (Raspberry)

Voici les deux classes présentes dans le programme python de la Raspberry,

### Classe Serial

L’instance de Serial sert à demander constamment les sept messages via le port Serial son adresse est encadrée en rouge :



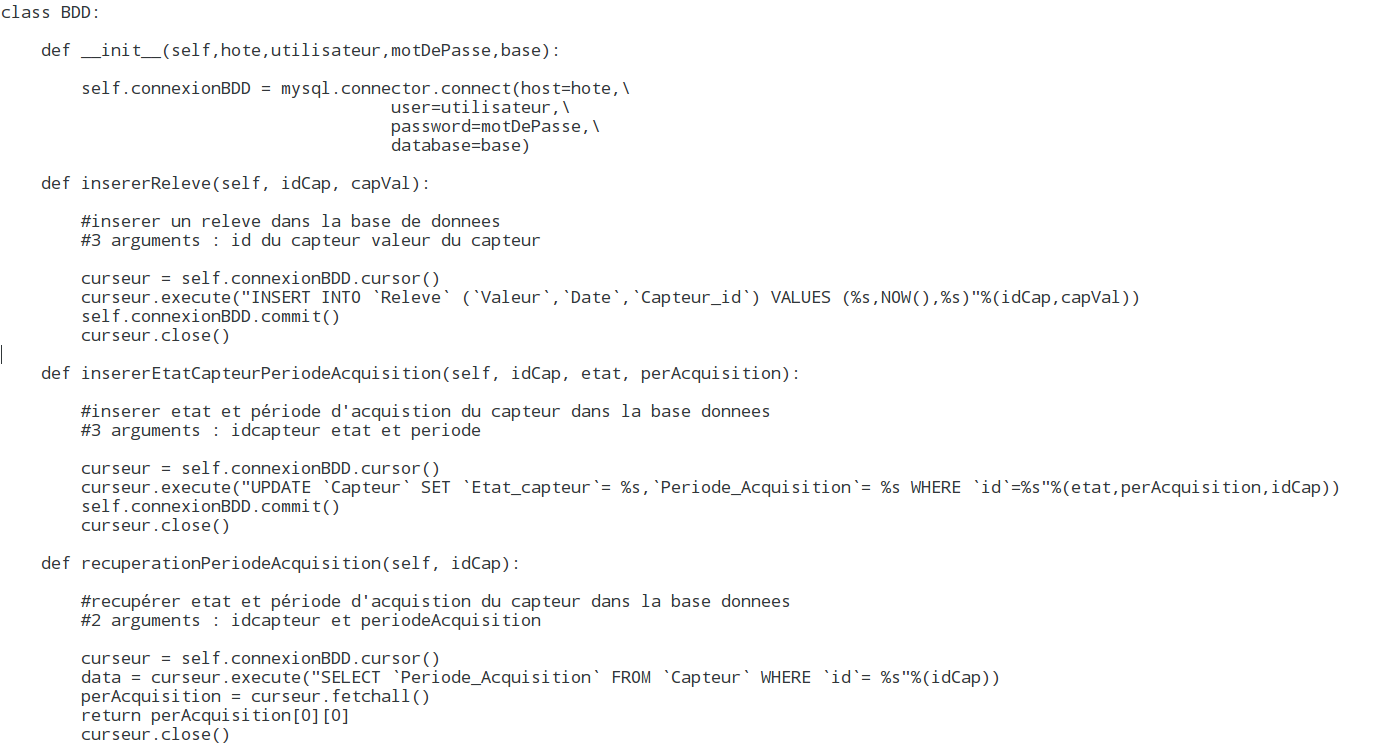
* envoiMessage

Cette méthode écrit un message composé de trois champs, le 1er est l’id du capteur le 2ème est la position du chronomètre qui résulte de la fonction time.time(). Le 3ème est la période d’acquisition récupérer par commande SQL dans la base de données.

* recupereValeur

Cette fonction retourne la ligne reçu dans le port serial, grâce à ma fonction readline()

### Classe BDD



* \_\_init\_\_

C’est le constructeur par défault il prend les paramètres de la base de données et des identifiants pour s’y connecter.

* insererReleve()

Cette méthode insère le relevé dans la base de données.

* insererEtatCapteurPeriodeAcquisition()

Cette méthode permet de modifier l’etat du capteur ainsi que sa période dans les champs de la base de données.

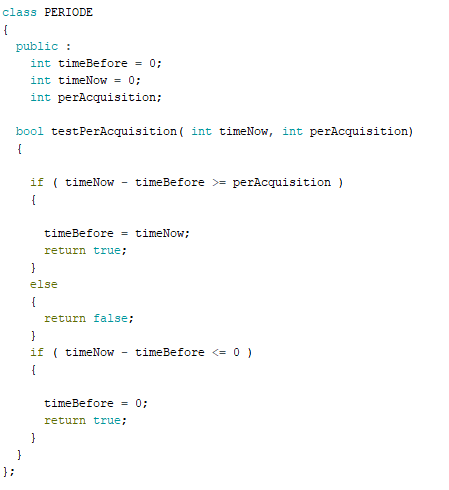
* recuperationPeriodeAcquisition()

Cette fonction retourne la période d’acquisition du capteur demandé dont l’id lui est passé en paramètre.

## Le programme de la carte d’acquisition (Arduino)

Le programme est constitué d’une classe PERIODE et d’une boucle while.

### Classe Periode



Les objets de la classe ont trois variables déclarées publiques. Elles sont nécessaires au calcul de la période.

* timeBefore contient la valeur du temps du dernier le calcul réussi, soit quand la valeur à été transmise.
* timeNow contient la valeur actuelle du temps.
* perAcquisition contient le temps de la période d’acquisition.

La fonction testPerAcquisition retourne « true » quand la période d’acquisition est validée, « false » sinon.

### Boucle principale

## Test Unitaire

## Fiche recette