

**Serre Automatique**

Système d’acquisition

Partie personnelle

de

Killian LABATTUT

# Dossier Technique - Partie Personnelle

Table des matières

[Dossier Technique - Partie Personnelle 2](#_Toc9504059)

[Situation dans le projet 3](#_Toc9504060)

[La tâche de l’étudiant dans le contexte du projet 4](#_Toc9504061)

[Diagrammes SysML 5](#_Toc9504062)

[Diagramme de déploiement 7](#_Toc9504063)

[Présentation des logiciels utilisées 8](#_Toc9504064)

[Le programme de la carte de gestion (Raspberry) 8](#_Toc9504065)

[Classe Serial 8](#_Toc9504066)

[Classe BDD 9](#_Toc9504067)

[Le programme de la carte d’acquisition (Arduino) 10](#_Toc9504068)

[Classe Période 10](#_Toc9504069)

[Boucle principale 11](#_Toc9504070)

[Programme Acquisition.py 12](#_Toc9504071)

[Test Unitaire 13](#_Toc9504072)

[Fiche recette 14](#_Toc9504073)

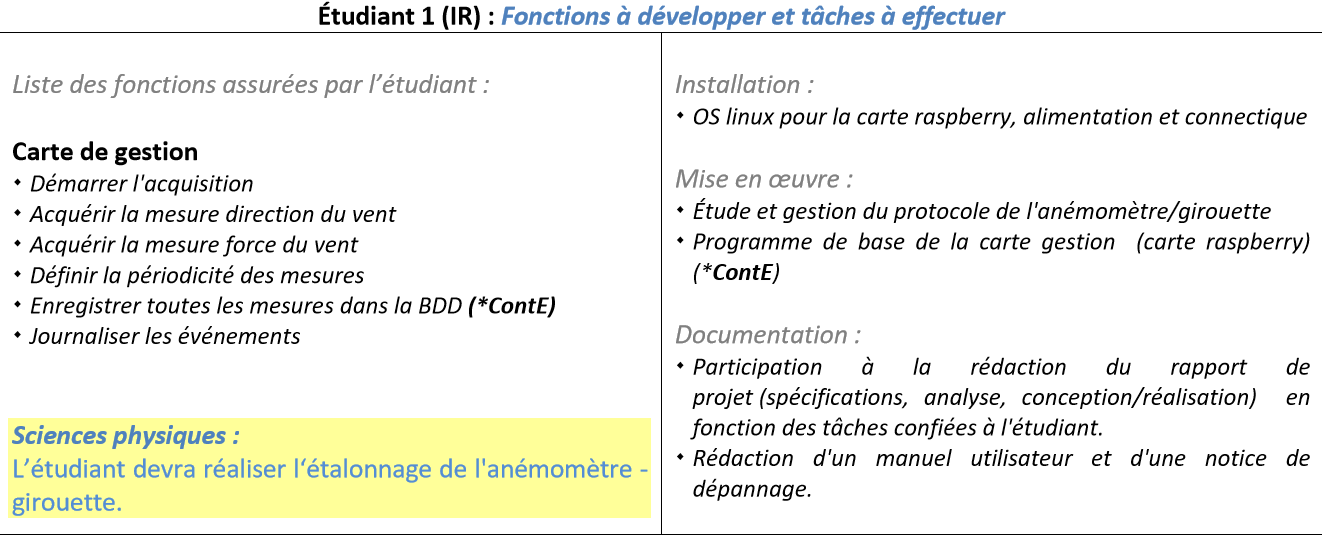
[La station météo : La Crosse WS 3650 14](#_Toc9504074)

[Les caractéristiques de la station La Crosse WS 3650 15](#_Toc9504075)

[Journalisation des événements 15](#_Toc9504076)

[Conclusion 15](#_Toc9504077)

## 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 l’O.S Debian 8.7 installé.

* Acquérir les mesures de la direction et de la force du vent,

Ces mesures ne sont pas encore acquises à ce stade du projet.

La station météo « Weather Station La Crosse WS 3650 » m’a été confiée 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 protocole 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 Android.

* 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.

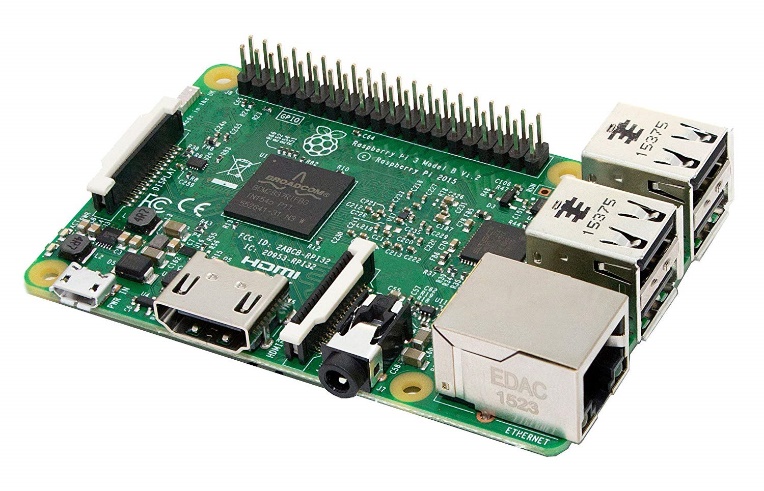
* Installation,

J’ai déployé une image Debian nommée « Jessie » sur la carte Raspberry car cette version est dotée de Python 2.7.9 et 3.4.2.

* Documentation j’ai participé à la rédaction du rapport de projet

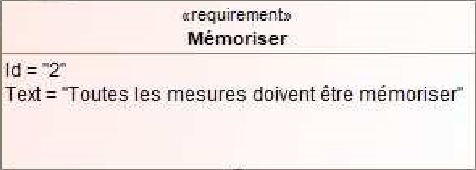
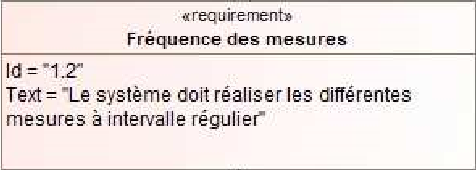
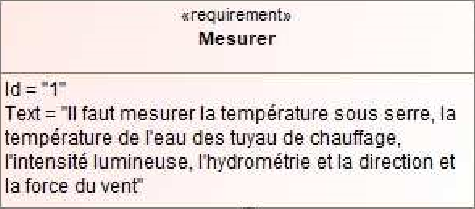
## La tâche de l’étudiant dans le contexte du projet

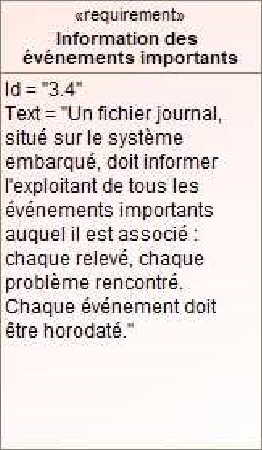
Carte de Gestion Carte d’Acquisition



**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 à la construction des programmes, et je me suis décidé de programmer la carte 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.

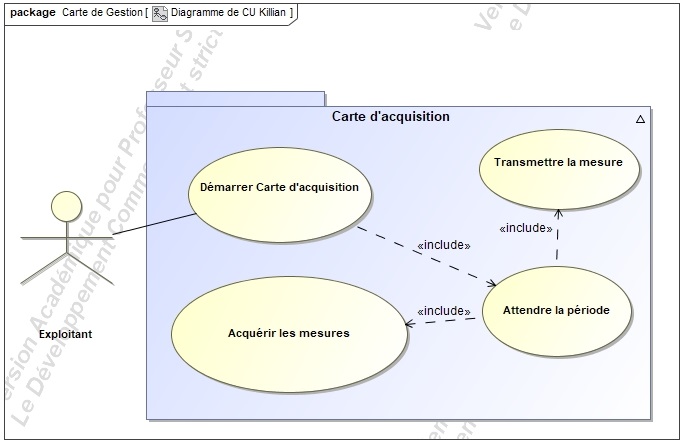
Pendant ce temps la carte d’acquisition reçoit le message et traite les données reçues, elle fait la différence entre le temps du message et le dernier temps reçu. S’il est égal à la période d’acquisition transmise, la valeur du capteur et son identifiant sont envoyés à la carte Raspberry qui enregistre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

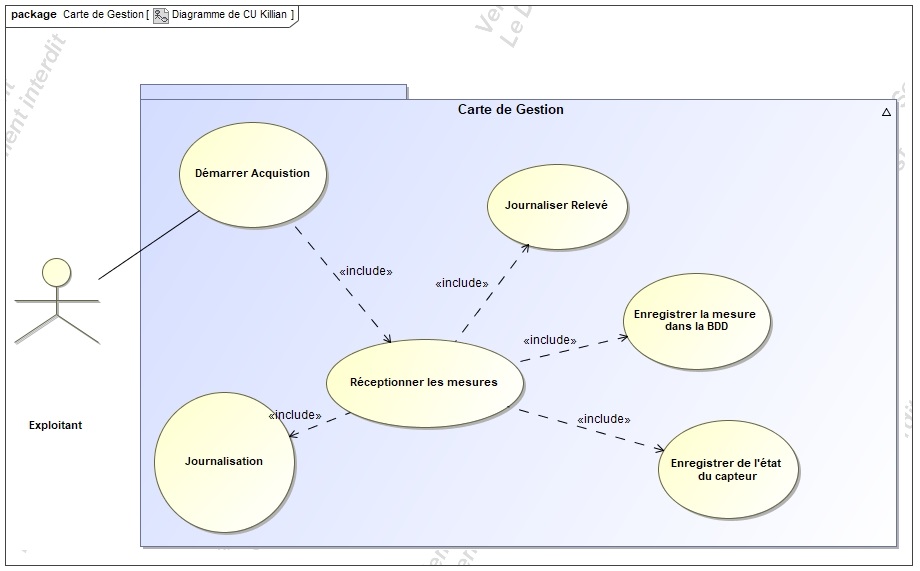
Pour pouvoir répondre aux exigences, j’ai réalisé différents diagrammes UML afin de détailler ma partie.

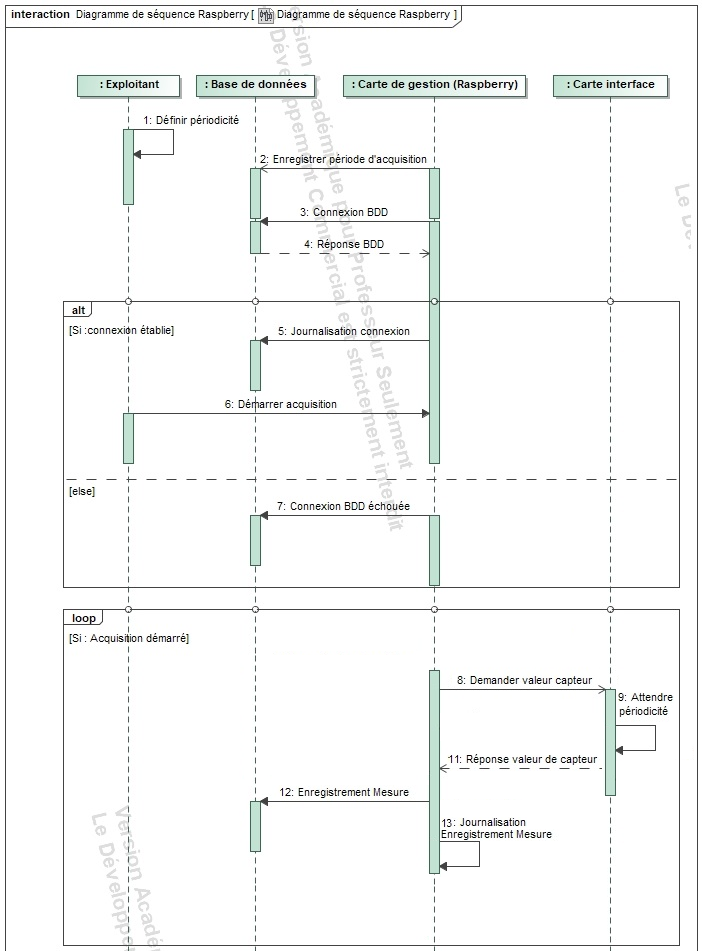
## Diagrammes SysML

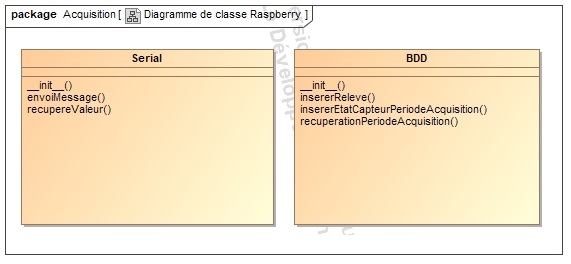
Voici les diagrammes de cas d’utilisation de la carte de Gestion et de la carte d’Acquisition.

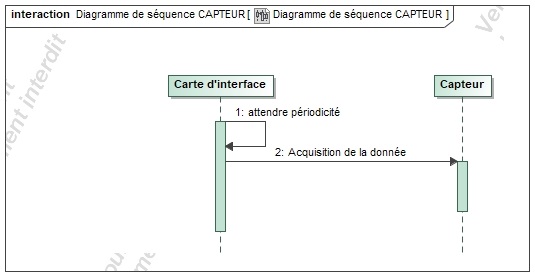
Les diagrammes représentent les deux systèmes et les actions qu’ils effectuent.





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 dont je suis en charge dans le projet.

## Présentation des logiciels utilisées

Les logiciels que j’ai utilisés sont :

Arduino 1.8.8 Word

MagicDraw UML 17.0.3

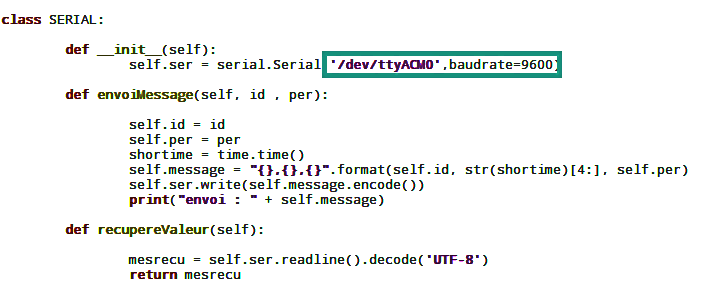
Python 3 (IDLE)

## 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 :



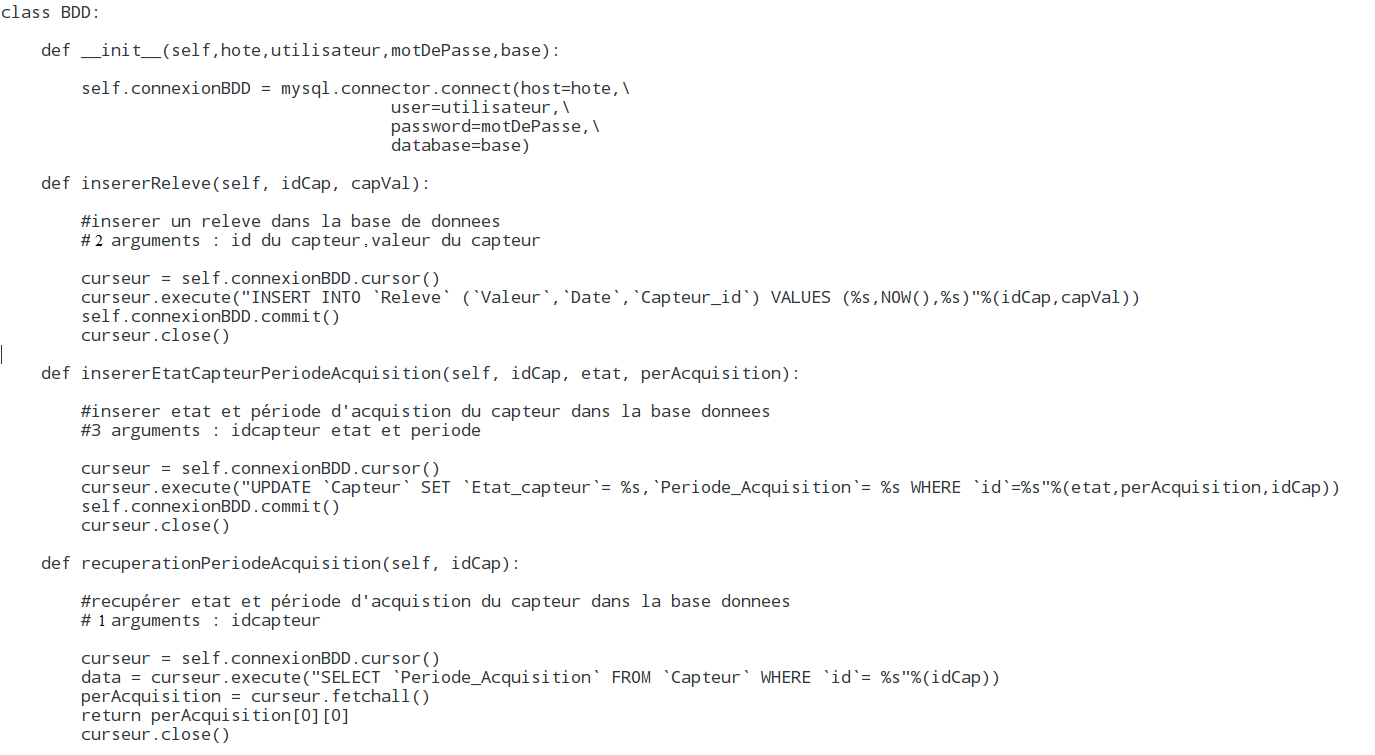
* envoiMessage

Cette méthode écrit un message composé de trois champs, le 1er est l’identifiant du capteur le 2ème est la position du chronomètre qui résulte de la fonction time.time(). time.time() renvoie un double du nombre de secondes qu’il s’est écoulé depuis 1970. Le 3ème est la période d’acquisition qui est récupérée par commande SQL dans la base de données.

* recupereValeur

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

### Classe BDD



* \_\_init\_\_()

C’est le constructeur par défaut, 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’état du capteur ainsi que sa période dans les champs de la base de données.

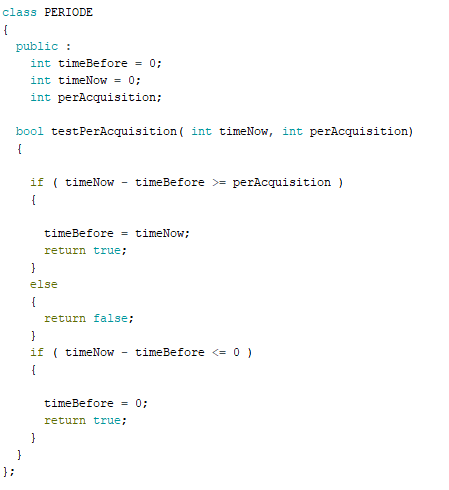
* recuperationPeriodeAcquisition()

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

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

Le programme est constitué principalement : d’une classe Période et d’une boucle principale.

### Classe Période



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 calcul réussi, soit quand la valeur a 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



## Programme Acquisition.py



Ce programme réalise les actions dans l’ordre prévu du diagramme de séquence. Pour cela il utilise les Classes expliquées précédemment.

Valeur = Cap.recupereValeur()

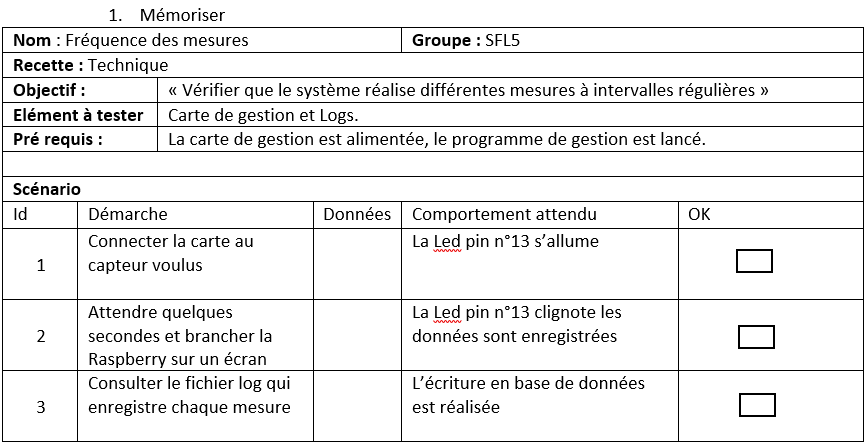
Cette ligne met le programme en attente de lecture du port serial. Le programme ne reste pas longtemps en attente puisque le flux de donnée est continu.

## Test Unitaire

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nom** : Ecriture en base de données | | | | **Groupe :** SFL5 | |
| **Test :** Technique | | | | | |
| **Objectif :** | | « Vérifier que le système enregistre en base de données les valeurs générés par la carte d’acquisition » | | | |
| **Elément à tester** | | Carte de gestion et base de données. | | | |
| **Pré requis :** | | La carte de gestion est alimentée. | | | |
|  | | | | | |
| **Scénario :** | | | | | |
| Id | Démarche | | Comportement attendu | | Comportement obtenu |
| 1 | Alimenter la carte de gestion et la connecter à la carte Arduino et au réseau. | |  | |  |
| 2 | Exécuter le fichier avec la commande « python /Documents/SFL5\_Projet/Acquisition.py ».  Celui-ci récupère dans valeurs aléatoires généré par la arduino (les capteurs n’étant pas encore intégrés). | | Aucune erreur dans l’exécution.  Une erreur est survenue, vérifier que le serveur hébergeant la BDD est en service. | |  |
| 3 | Consulter la base de données grâce à PHP MyAdmin sur le navigateur Web à « 10.16.37.161/phpmyadmin/ »  User : root  Password : Serreautomatique | | Constater la présence des valeurs dans les champs de la BDD : | |  |

1. Ecriture en base de données

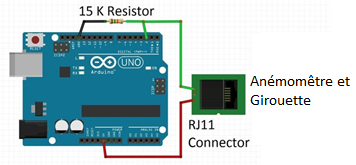
## Fiche recette



## La station météo : La Crosse WS 3650

Comme prévu dans le diagramme de Gantt j’ai commencé par effectuer la tâche acquisition de ma partie, cependant à ce stade du projet je n’ai pas encore configuré la station La Crosse. Au début de l’analyse j’ai tout de même effectué des recherches sur ce système qui me servent actuellement, les voici :

Ce schéma de montage me sert pour pouvoir utiliser les connecteurs RJ11 femelles et transmettre les données à la carte d’acquisition (Arduino), le montage a pu être réalisé sur un modèle MEGA. La platine d’essai est soudée aux connecteurs RJ11 femelles et les broches permettant de recevoir le RJ11 mâle du de la station La Crosse.



### Les caractéristiques de la station La Crosse WS 3650

Units: km/h, m/s, knots, mph or Beaufort

Range : from 0 to 180 km/h (or from 1 to 50 m/s)

Wind direction displayed on a 360° compass

Precision: 0.1 m/s

Recording of the minimum and maximum data measured with time and date

## Journalisation des événements

Afin de journaliser les événements, j’ai décidé d’écrire les événements d’insertion de relevés, d’insertion de valeur et de connexion à la base de données, dans un fichier log. Il sera enregistré sur la carte Raspberry.

Pour ce faire je compte donc utiliser la librairie : « logging » en python. Il existe plusieurs types de fichier logs : debug, information, alerte, erreur, et critique. Dans notre cas je vais créer un fichier log d’information.

À ce stade du projet, je n’ai pas encore mise au point l’écriture de la journalisation.

## Conclusion

J’ai privilégié la tâche Acquisition qui me paraissait la plus importante au début du projet. Je peux maintenant m’atteler à la journalisation et aux capteurs, à l’aide de mes précédentes recherches.

Le travail que j’effectué durant ce projet m’a beaucoup aidé à compléter mon enseignement en informatique.