



Banc de test éolienne

Dossier Personnel

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Arnaud JULLIEN |  |  |  | 16/01/2017 |

Table des matières

1. [II. Situation du projet 3](#_Toc513990327)

[1. Contexte 3](#_Toc513990328)

[2. Synoptique de la réalisation 3](#_Toc513990329)

[3. Rappel des tâches de l’étudiant 5](#_Toc513990330)

[4. Contraintes liées au développement 6](#_Toc513990331)

1. [III. Conception et mise en œuvre 7](#_Toc513990332)

[1. Diagramme de classe 7](#_Toc513990333)

[a. La classe CAcqForce 8](#_Toc513990334)

[b. La classe CAcqPuissance 10](#_Toc513990335)

# Situation du projet

## Contexte

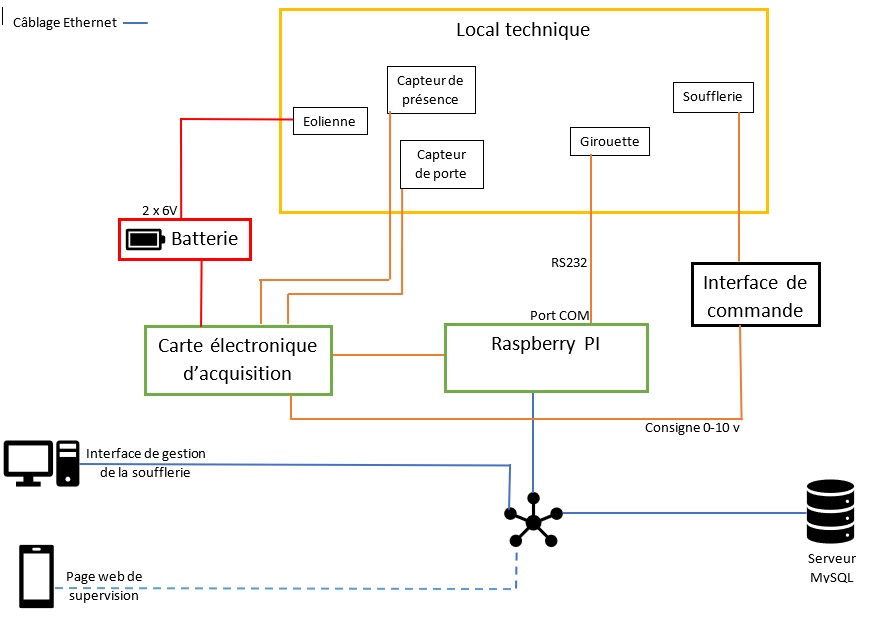
En France, d’après l’Association des Professionnels du Petit Eolien (AFPPE), 2 500 petites éoliennes ont été installées entre 2010 et 2012. Le marché compte aujourd’hui plusieurs fabricants français, mais ceux-ci éprouvent des difficultés à se développer : en effet, la rentabilité économique pour l’utilisateur (le producteur ou l’auto consommateur) n’est pas toujours garantie, et la concurrence de fabricants étrangers produisant de petites éoliennes en grande série reste forte.

Elles permettent de s’alimenter en électricité en total autonomie. Si elle n’est pas consommée immédiatement, elle est stockée dans des batteries de stockage. Malheureusement, leur prix élevé des batteries marginalise cette solution. De plus leur durée de vie excède rarement 7ans en moyenne c’est pourquoi le choix de l’éolienne est important.

Mais en milieu urbain une éolienne avec axe horizontal n’est pas adaptée, c’est pourquoi Monsieur Jacky ROBIN a donc conçu et fabriqué une première éolienne à axe vertical, type Savonius. Celle-ci fut testée avec la soufflerie disponible dans l’établissement.

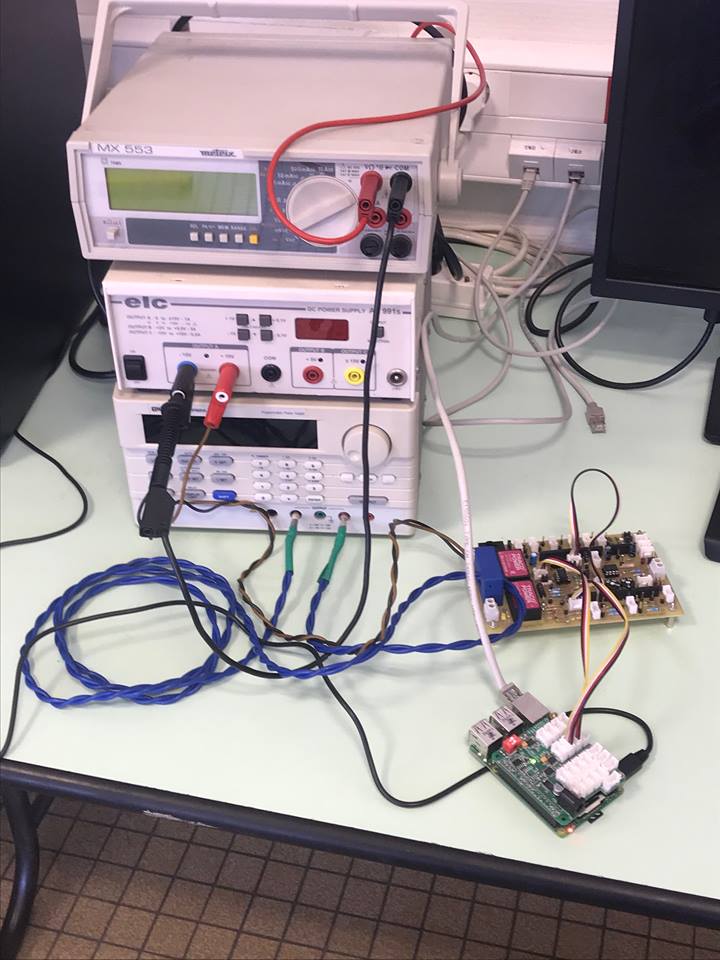
L’objectif principal de ce projet est donc de vérifier la rentabilité de l’éolienne et en évaluer son efficacité.

## Synoptique de la réalisation



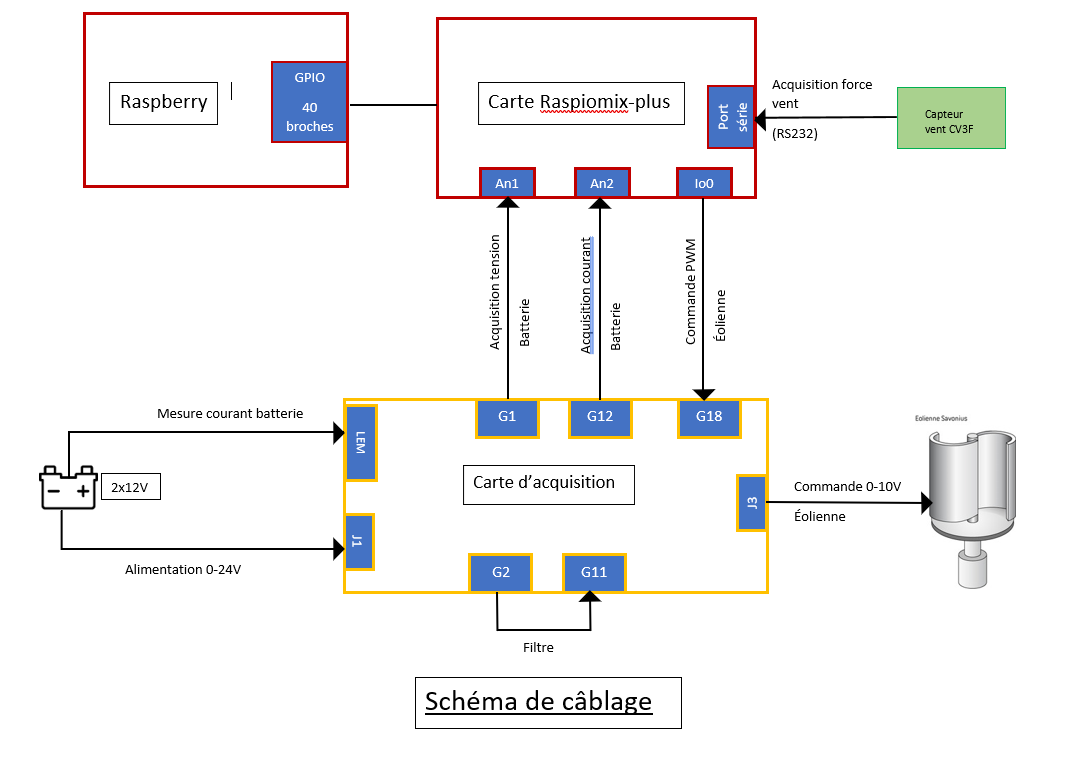
Parties sur lesquelles je travaille

Au sein du projet je devais directement être connecté à la soufflerie (local technique) je ne pouvais pas toujours travailler sur place c’est pour quoi je me suis servi d’une alimentation afin de simuler l’éolienne.



Raspberry

Carte d’acquisition



## Rappel des tâches de l’étudiant

Dans ce projet de gestion de banc de test éolienne, j’avais pour objectif de relier l’interface de gestion de la soufflerie au local technique en passant par une Raspberry et une carte d’acquisition. J’ai donc dû comprendre comment fonctionnait l’ancienne commande qui permettait de faire fonctionner la soufflerie puis trouver par quelle façon je pouvais récupérer toutes les données que je souhaitais

Dans un premier temps je devais donc envoyer un courant de 0 à 10V à la soufflerie pour la faire fonctionner. La puissance étant envoyé par l’application à ma Raspberry par socket.

Je devais ensuite récupérer la force du vent et la puissance produite par l’éolienne en instantanée et l’implanter dans une base de données afin qu’elles soient accessibles pour le site web.

Je travaille principalement sur la carte Raspberry PI, Mon but est de programmer dans un premier temps la commande consigne en python afin de rajouter les deux options de contrôle qui sont le mode instantané et le mode scénario.

**Le mode instantané** permet de modifier la puissance de la soufflerie entre 0 et 100%. Il doit aussi capturer la force du vent et la puissance produite par l’éolienne afin de l’afficher plus tard sur l’application “banc de test éolienne”.

**Le mode scenario** permet lui de créer des scenarios afin de pouvoir tester différentes situations pour l’éolienne. Avec ce mode l’utilisateur aura le choix entre plusieurs scenarios et le programme devra prendre en compte tous les changements liés à la soufflerie

Le système embarqué échangera constamment avec l’application c++ il devra dans un premier temps suivre les indications de celle-ci puis dans un second retransmettre les données qu’il aura reçu grâce aux capteurs de la soufflerie.

Le programme devra être en lien avec la base de données MySQL et l’application c++ via socket.

Le rôle de ma Raspberry sera de récupérer le mode choisit par l’application. Si le mode est en instantané, l’application changera manuellement la puissance de la soufflerie. Je récupèrerais ensuite par liaison TCP socket chaque modification de puissance effectuée sur l’application et enverrais directement l’ordre à la soufflerie.

## Contraintes liées au développement

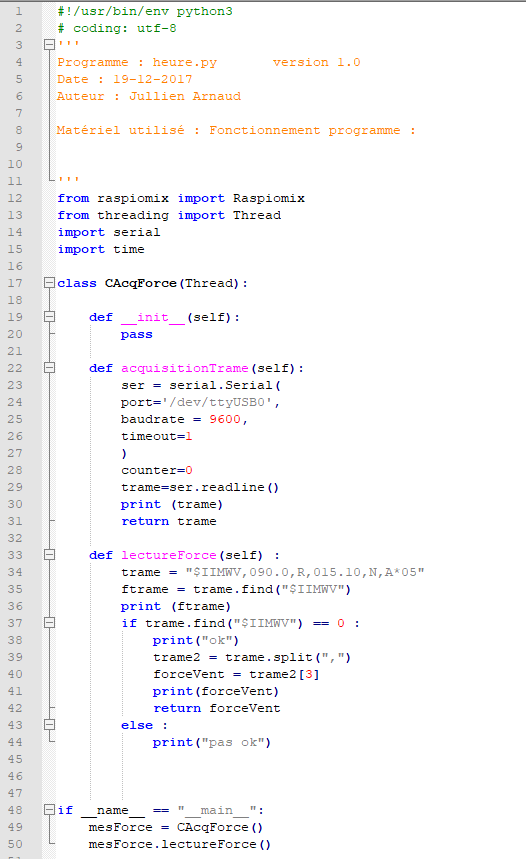
Dans ce projet la contrainte que j’avais était de travailler sur Raspberry et de devoir communiquer avec l’application, le site web et la soufflerie. Je devais donc coder en python sous linux.

# Conception et mise en œuvre

## Diagramme de classe

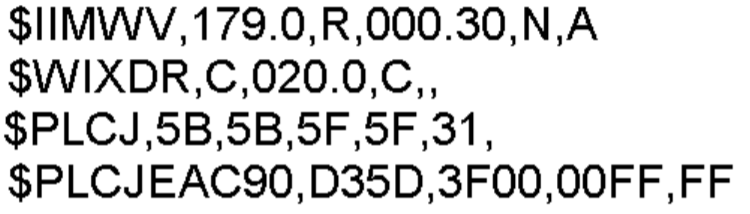
En fonction des données dont j’avais besoin j’ai pu réaliser le diagramme de classes précédent.

### La classe CAcqForce



Elle est utilisée pour récupérer la trame du capteur CV3F qui est le capteur de force du vent

La force du vent est obtenue par un port série RS232 directement connecté à la Raspberry. J’acquis donc une trame que je dois ensuite décoder afin de récupérer seulement la donnée voulue. Le capteur envoie la force du vent en nœud. Je dois donc la multiplier par 1,852 afin de l’obtenir en km/h

Exemple de trame :

A l’acquisition des trames j’en reçois plusieurs différentes, ici nous nous intéressons à la trame commençant par $IIMWV :



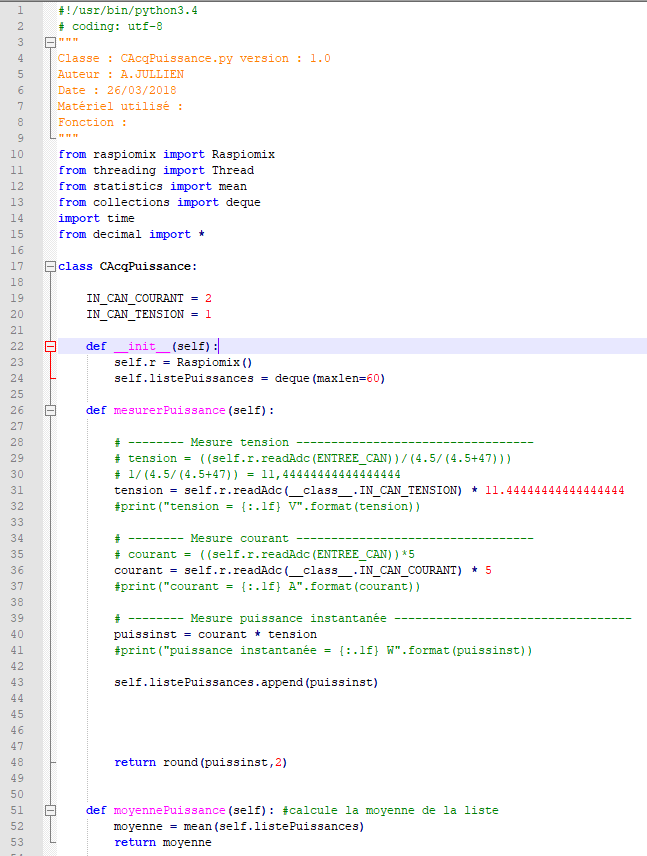
1 2 3 4 5

1. Cela correspond à l’angle du vent les valeurs peuvent osciller entre 0 et 360° (ici donc 179°)
2. Cela correspond à la référence on a donc deux valeurs : R= RELATIVE et T= TRUE (ici Relative)
3. Cela correspond à la vitesse du vent (ici 0.30)
4. Cela correspond à l’unité de la vitesse du vent il y a trois valeurs possibles K/M/N (ici N = nœud)
5. Le Statut, A = DATA VALIDE

Dans mon projet je m’intéresse donc seulement à la troisième et quatrième valeur qui correspond à la vitesse et l’unité du vent.

La fonction aquisitionTrame() sert faire comme son nom l’indique la lecture du capteur. C’est la fonction lectureForce() qui fait tout ce procédé.

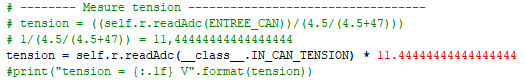
### La classe CAcqPuissance



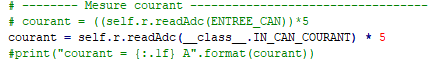
Elle est utilisée pour récupérer la puissance produite par l’éolienne

Pour obtenir des Watts et des Watt.h j’avais besoin dans un premier temps de récupérer la tension (volts) des batteries afin d’obtenir l’énergie produite par celle-ci. Je devais dans un second temps récupérer le courant (ampères). En effet : Watt = Ampère \* Volts

Pour récupérer la tension je devais prendre en compte les resistances de la carte d’acquisition afin de calculer par quel nombre je devrais multiplier la tension lue par la carte d’acquisition.



Pour le courant afin d’obtenir la valeur exacte je devais multiplier les ampères obtenus par 5



Puis multiplier les données obtenues entre elles afin d’obtenir la puissance instantanée. J’ai ensuite implanté le tout dans un tableau fixé à une limite de 60 grâce à la fonction **deque()** de la classe **Collections.** Le tout est donc codé dans la fonction **mesurerPuissance().**

La limite est fixée à 60 car je fais une aquisition des donées tooutes les secondes et il me suffit donc de faire une moyenne de cette liste grâce à la fonction **means()** de la classe **Statistics.** Je multipli par la suite le resultat par 60 afin d’obtenir des Watt.h.

En mode scénario l’application enverra l’id du scénario choisit et je récupérerais dans la base de données les changements de séquences liées au scénario (une séquence sera composée d’une puissance définit sur un intervalle de temps) je placerais ensuite chaque séquence dans un tableau a double entrés.

Mon but est ensuite de récupérer ensuite l’énergie produite par l’éolienne. A chaque changement de puissance j’enverrais les résultats à la base de données.

Pour l’affichage de l’énergie instantanée, celle-ci sera relevé toutes les secondes. Ainsi cela permettra à l’utilisateur de de voir en temps réel et d’une façon précise l’énergie produite.

Pour l’énergie produite en W.h je relève 10 valeurs dans un intervalle d’une seconde afin de faire une moyenne et me mettre cette énergie en W.h pour que le graphique affiché soit plus lisible.

# Test unitaire

## Test unitaire de la fonction mesurerPuissance()

La fonction mesurerPuissance() sert à calculer la puissance instantanée