Rapport final du projet HopeAndBike

Gaylord LEBRUN - Mehdi BRINIS

Tuteurs : Tomas MENARD

Clients : Relais d'sciences

Maison du vélo

Remerciements

Tout d'abord, nous tenons à remercier Monsieur Sébastien SAEZ responsable de la formation Mécatronique et Systèmes Nomades ainsi que Monsieur Basile DUFAY responsable de l'organisation des Projets pour nous avoir permis de réaliser ce projet.

Ensuite, nous souhaitons remercier Monsieur Tomas MENARD, enseignant chercheur au GREYC, responsable de la troisième année systèmes embarqués et tuteur de notre projet pour nous avoir consacré du temps et de l'aide pour la réalisation de celui-ci. En nous donnant des conseils, des méthodes de travailles, des directives ainsi qu'en nous fournissant le matériel nécessaire.

Nous tenons aussi à remercier particulièrement Monsieur Jean-Marc ROUTOURE, enseignant chercheur à l'Université de Caen, chargée de projet HnB[1] au sein du relais de science pour avoir mis à notre disposition l'ensemble du kit HnB. Mais aussi pour avoir répondu aux questions que nous lui avons posées.

Nous remercions aussi monsieur Julien GASNIER, technicien au GREYC pour nous avoir fourni son expérience et son aide dans le domaine de l'électronique.

[1] Hope And Bike

Sommaire

1) Introduction

Le projet « HopeAndBike » est un projet culturel et scientifique ayant pour but de concevoir un système électronique open source[2] capable de relier un moteur électrique avec une batterie. Ce système permet de transformer un simple vélo en un VAE[3]

Figure 1 : Vélo possédant le système HopeAndBike

Etudiant en troisième année d'école d'ingénieur en Mécatronique et systèmes nomades à l'ESIX Normandie nous devons réaliser un projet ingénierie. Nous avons fait le choix de contribuer à l'avancer de ce projet open source. La réalisation de ce projet à plusieurs objectifs pédagogiques comme :

- Utiliser les compétences techniques acquises tout le long de la formation afin de les retranscrire dans la réalisation d'un projet technologique concret. Notamment les compétences acquises en gestion de projets
- Favoriser l'initiative, la créativité et le sens de l'organisation.

Dans la version actuelle, une machine d'état associé à une régulation de vitesse à l'aide d'un correcteur proportionnel est mise en place.

[2] « Code source mis à disposition du grand public, ce code est généralement le résultat d'une collaboration entre programmeurs. » cf. Wikipédia.

[3] Vélo à assistance électrique

2) Les attentes des clients

Pour commencer, après avoir testé le VAE nous avons observé que celui-ci donnes des accélérations brusques et importantes. C'est-à-dire que la puissance fournie par le moteur n'est pas proportionnelle à la puissance fournie par l'utilisateur (pédalage).

Les clients attendent que nous améliorons les défauts causés par le système d'asservissement actuel. Pour cela nous allons :

- Améliorer/Optimiser la régulation du système (puissance moteur fournisse proportionnelle à la puissance émise par l'utilisateur).
- Réaliser des simulations à l'aide de Simulink[4].
- Déterminer un ou plusieurs modèles prenant en compte la puissance (PWM) et la vitesse.
- Déterminer un modèle théorique du moteur
- Faire l'Identification d'un modèle d'asservissement du système.

[4] « Logiciel de modélisation système multi-physique édité par l'entreprise américaine The MathWorks ». cf. Wikipédia.

3) Matériels utilisés

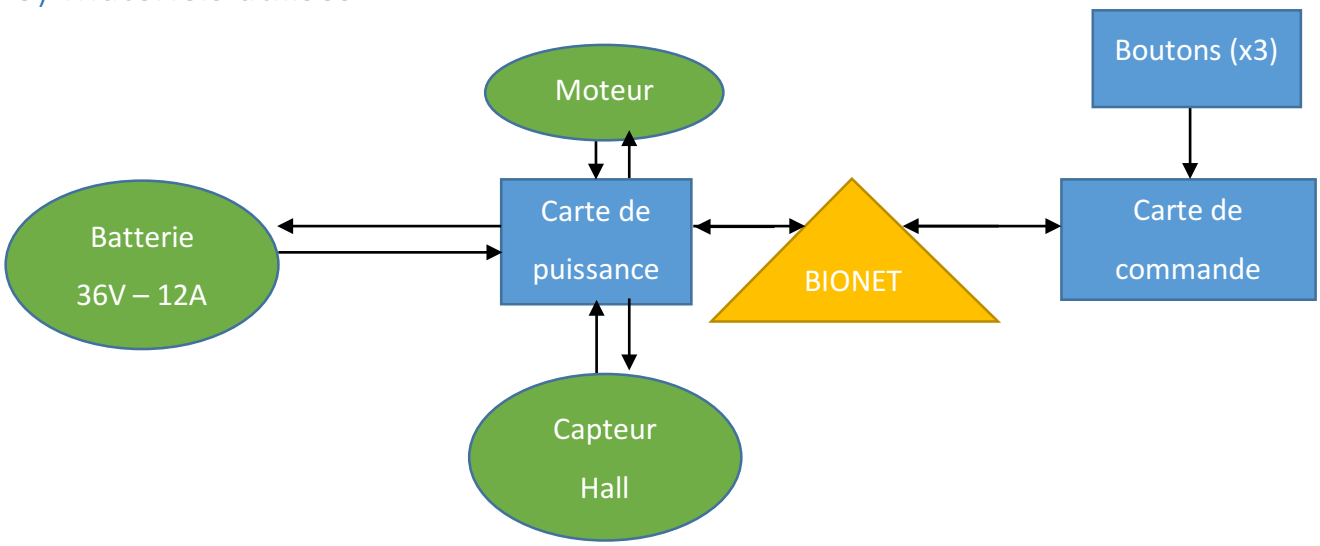


Schéma représentant le système HopeAndBike.

Composants achetés

Composants créés

a) Moteur

Moteur Brushless (synchrone) installé dans la roue arrière du vélo. Il permet de fournir un couple et ainsi augmenter la vitesse.

Figure 2 : Photo du moteur dé-rayonné

b) Batterie

Batterie fournissant une tension de 36V et un courant de 12A (puissance = 432W).

c) Carte commande

Carte Arduino pro mini permettant de commander les signaux avec un PCB permettant de visualiser le niveau l'assistance choisie (2 LED par niveau), s'il y a un problème d'échange de données entre les deux cartes (clignotement d'une LED), ainsi que le niveau de la batterie.

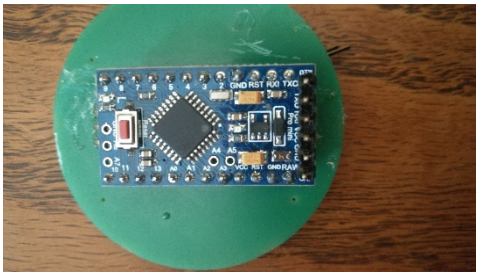


Figure 3 : Photo de la carte commande

Nous nous intéresserons au données échangées entre la carte de puissance et la carte de commande via le bus BIONET. Avec ces mesures nous pourrons modéliser différents modèles (puissance, vitesse). En sachant que le système est commandé par une machine d'état réalisé sur EXCEL.

d) Bouton – Interface Homme/Machine

Trois bouton poussoirs permettant de choisir le niveau d'assistance. Un bouton permettant d'augmenter le niveau d'assistance, un autre pour diminuer et le troisième pour couper l'assistance. Plus l'assistance est grande est plus le moteur fournie un couple fort, ce qui permet à l'utilisateur de fournir moins d'efforts.

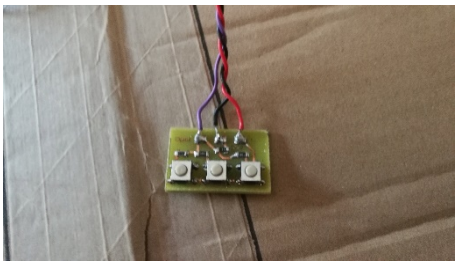


Figure 4 : Photo des trois boutons

e) Carte de puissance

La carte de puissance est composé d'un PCB comportant des microcontrôleurs PIC12 (mesures et calculs) et d'un PIC16 (contrôle moteur). Elle est reliée d'un coté à la batterie et de l'autre à la carte de commande.

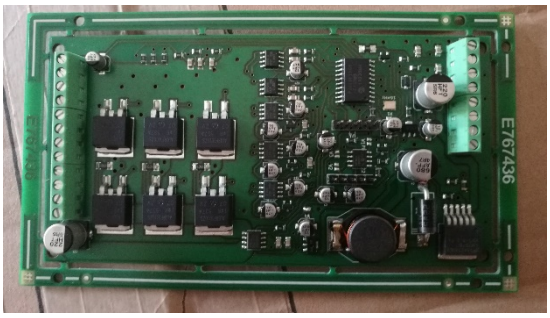


Figure 5 : Photo de la carte de puissance

f) Capteur Hall – pédaliers

Capteur à effet Hall permettant de capter un changement d'état sur le pédalier (à chaque front montant).

g) BIONET (Bic Internal Open NETwork)

Bus interne permettant de communiquer entre les différents éléments du système. Ce bus a été développé par les utilisateurs (principe open source).

Nous avons étudié le code afin de comprendre son fonctionnement. Différentes trames sont envoyées entre les deux cartes (Niveau de batterie, courant du moteur, vitesse de pédalage et la rotation de la roue). Quand on l'a décrypté nous avons retranscris toutes les informations sur le STM32 en espionnant directement le BIONET de la carte de puissance. Ensuite nous avons travaillé directement sur MATLAB/SIMULINK en récoltant les informations via l'UART2 (liaison série USB).

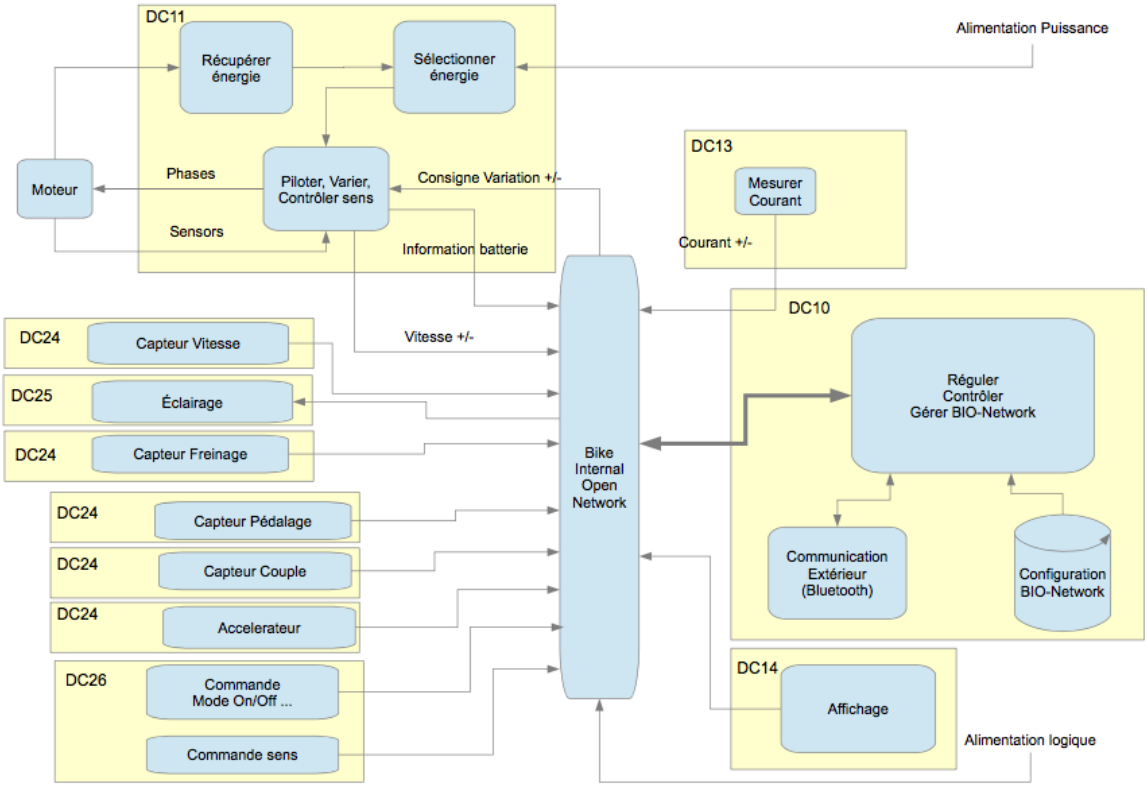


Figure 6 : Schéma du bus interne BIONET

f) Logiciels utilisés

- Matlab/Simulink :
  - Asservissement du système.
  - Réalisation de la commande.
  - Retranscriptions langage C (STM32) vers le BIONET (et vice-versa).
  - Récolter puis traiter les données échangées à travers le BIONET.

- Keil :
  - Implémentation du STM32.

- CubeMX :
  - Configurer le STM32.

- Arduino :
  - Implémentation de la carte Arduino.

4) Réalisations techniques

4.1) Tests n°1

Pour commencer nous avons conçu un banc de tests simple nous permettant de faire l'acquisition des données afin de vérifier les données échanger à travers le BIONET. Ce qui nous as permis de vérifier le fonctionnement du système.

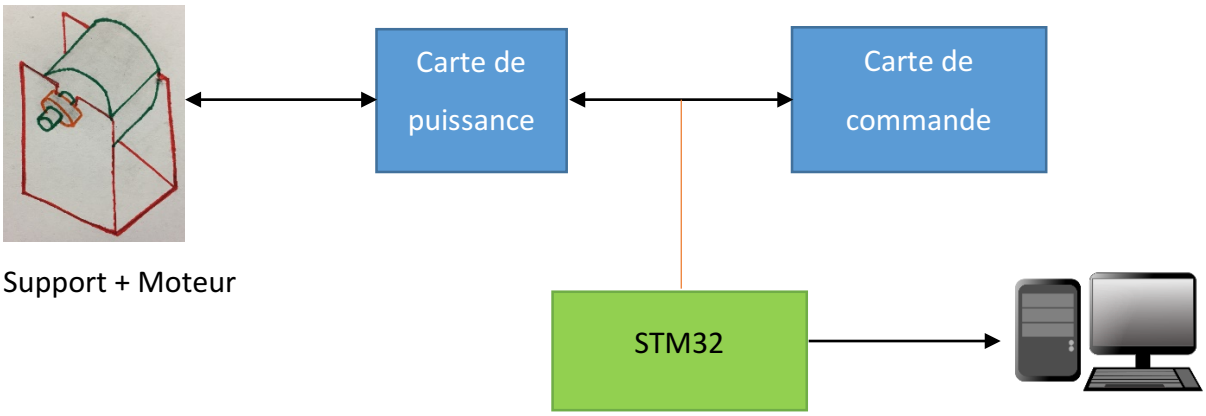


Schéma du banc de tests n°1

Ci-dessus, on peut voir le schéma de principe de notre banc de tests n°1. Le moteur est relié à la carte de puissance afin de l'alimenter.

La carte de commande envoie des trames à la carte de puissance afin de récolter certaines informations (niveau de batterie, le pédalage, la vitesse du moteur et le courant débité par la batterie).

La carte de commande reçoit, traite puis émet des valeurs afin de « commander » le système.

Nous avons réalisé ce banc de tests dans le but « d'espionner » les échanges de données entre les deux cartes. Par conséquent nous avons connecté une carte STM32 directement sur le BIONET (carte alimenté par l'ordinateur). Et ainsi on récupère toutes les données sur l'ordinateur via l'UART2 (liaison série USB).

4.2) Tests n°2

Par la suite on a réalisé un banc de tests permettant de récolter les données directement sur le vélo

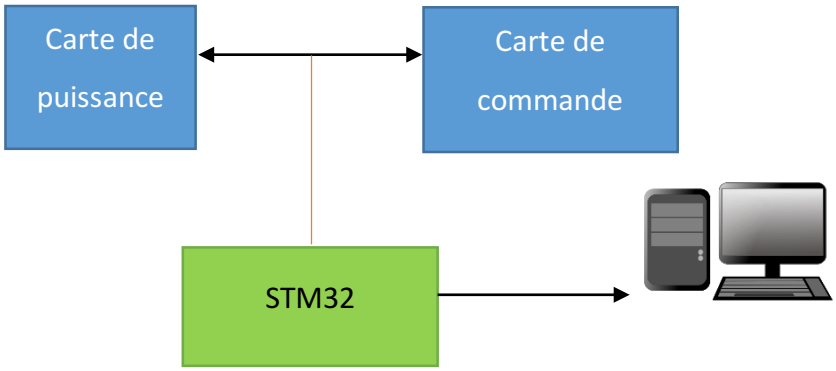


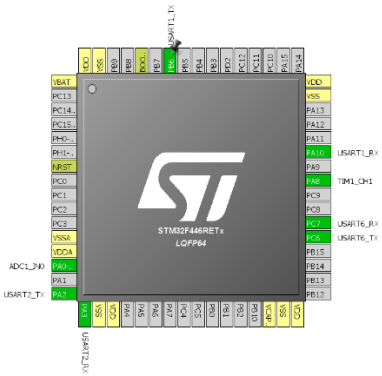
Schéma du banc de tests n°2

Ci-dessus, on peut voir le schéma de principe de notre banc de tests n°2. Cette fois-ci nous alimentons le STM32 avec le VCC de la carte de puissance (+5V) et nous récoltons soit la transmission carte de puissance vers la carte de commande, soit la transmission entre la carte de commande et la carte de puissance.

Comme auparavant on a réalisé ce banc de tests dans le but « d'espionner » les échanges de données entre les deux cartes en branchant le STM32 directement sur le BIONET et en récupérant toutes les données sur l'ordinateur via l'UART2 (liaison série USB).

4.2.1) Configuration du STM32 avec CubMX

Pour commencer nous avons besoin de configurer le STM32 (Timer, PWM et quelques périphériques).



Symbolisation des pins configurer sur le STM32

4.2.1) Création d'un schéma Simulink

Nous avons besoin de récolter les données afin de les traiter, pour cela on a créé un schéma SIMULINK nous permettant de les récupérer via l'UART2.

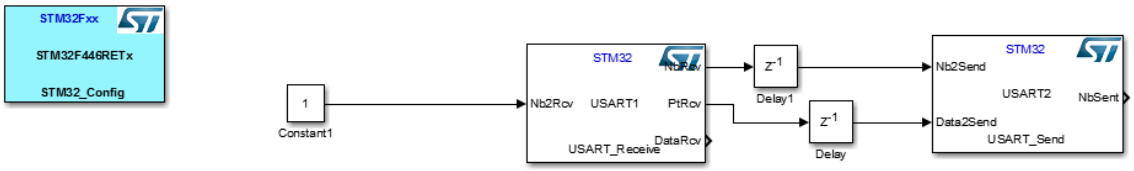
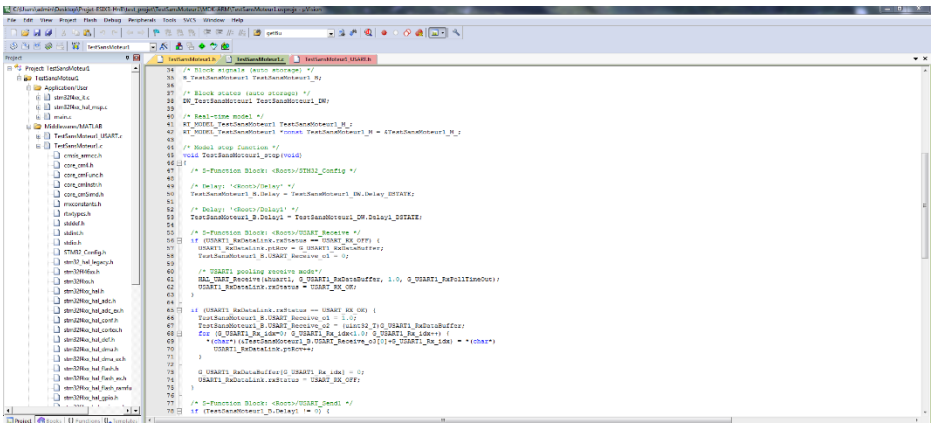


Schéma Simulink

On peut observer 3 blocs. Le premier en bleu nous permet de configurer le STM32 avec les paramètres souhaités (vitesse de transmission, type de MCU …). Ensuite un bloc « USART_Receive » et un bloc « USART_Send ». La réception nous permet d'observer les données échanger à travers le BIONET via l'USART1 et la transmission permet de retranscrire les données sur l'ordinateur via la liaison série USB (USART2).

4.2.2) Retranscription vers le STM32

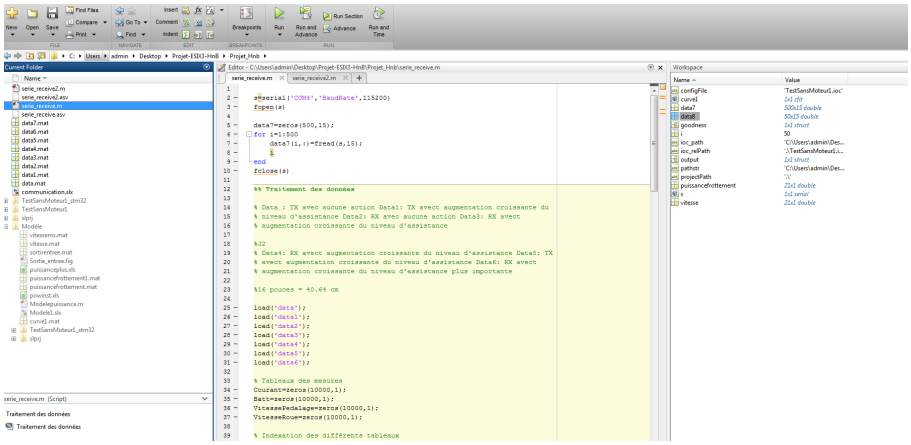
Cette étape nous permet de retranscrire le schéma Simulink en code C afin de l'implémenter sur le STM32.



Code en C sur le logiciel Keil

4.2.3) Fichier .m sur Matlab

Nous avons édité un programme Matlab nous permettant de récolter les données. Mais aussi de les trier et de le mettre dans des tableaux afin de les exploiter plus facilement.

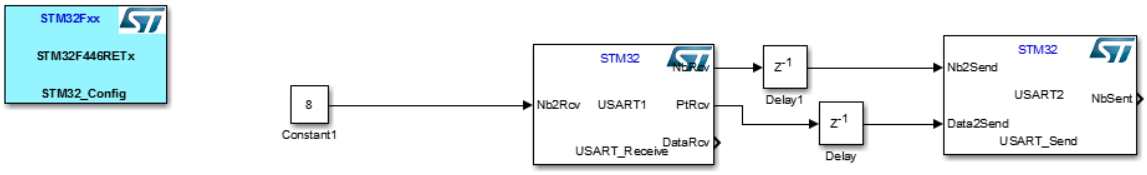


Fichier Matlab

Conclusions : Nous pouvons remarquer que l'envoi des données n'est pas du tout synchronisé et ainsi nous ne pouvons pas lier nos données avec le temps. Pour cela on va devoir reprogrammer la carte ARDUINO en lui demandant directement les valeurs que l'on souhaite obtenir.

4.3) Tests n°3

Les problèmes de synchronisations de l'envoi des données à travers le BIONET rencontrées dans le test n°2 nous ont amenés à reprogrammer la carte ARDUINO.
Nous avons ajouté une fonction nous permettant d'envoyer une trame de données contenant les informations utiles au projet, qui sont : le courant du moteur en ampère, la vitesse de pédalage en tr/min et la vitesse de rotation de la roue en km/h.
On reprogramme le STM32 en lui précisant qu'on envoie un paquet de trame contenant les informations souhaitées et ainsi on « synchronise » l'échange de données.



A l'aide de Matlab, l'ensemble des trames sont récoltées, triées puis traités. Les données sont stockées dans des tableaux. Les résultats obtenus subissent un post traitement afin de supprimer les valeurs dites « aberrantes ».

5) Etude mécanique

Afin de mieux comprendre le système et la liaison entre la vitesse et la puissance fournie par le cycliste nous avons réalisé une étude mécanique.
Pour cela nous allons vous présenter plusieurs conditions :

1.1) Déplacement sur un terrain plat

Frottement de l'air : On suppose qu'il n'y a pas de vent. Le frottement de l'air se traduit par une force F_{air} opposée au déplacement du cycliste. L'intensité de la force de frottement de l'air F_{air} est proportionnelle au carré de la vitesse.

$$F_{air} = KV^2 \quad (1)$$

F_{air} : en newton (N)
V : en mètre par seconde (m/s)
$K = 0{,}5*\rho*C_x$ avec C_x coefficient de frottement de l'air . K en $N \cdot s^2/m^2$

K dépend de la position du cycliste, du type de vélo, de la morphologie du cycliste, de l'équipement du cycliste (casque, combinaison...).

A vitesse constante appliquons le PFD[5] :

$$m\frac{dV}{dt} = \sum_{k=0}^{n} \vec{F}_{ext} = 0 = \overrightarrow{F_{air}} + \vec{F}$$

$$F = -F_{air} \quad (2)$$

Nous savons que la puissance mécanique fournie par le cycliste est liée à la force de déplacement et à la vitesse (en m/s) par la relation :

$$P = FV \quad (3)$$

Les équations (1), (2) et (3) permettent d'écrire :

$$P = KV^3 \quad (4)$$

[5] Principe Fondamental de la Dynamique

1.2) Influence du vent (sur un terrain plat)

- Les frottements de l'air deviennent :

$$F_{air} = \mathrm{K}(V + V_{air})^2 \ (5)$$

Remarques : On met un + pour un vent de face (défavorable, donc augmentation des frottements) et un moins pour un vent de dos (favorable, donc diminution des frottements)

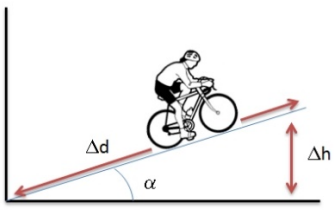
D'après (3) et (5) on trouve :

$$P = K(V + V_{air})^3 \ \ (4)$$

1.3) Déplacement sur une côte

Quand on dit qu'une pente p est de 10% cela veut dire que pour une distance Δd de 100m parcouru, le cycliste grimpe de 10m en altitude Δh. Donc on pose que l'angle de la pente vaut :

$$\alpha = \tan^{-1}(p/100)$$



- Poids : comme nous prenons en considération une cote on pourra observer que le poids aura une composante selon l'axe de l'avancement du cycliste.

$$P = mg \sin \alpha \ \ (5)$$

- Frottements : L'avancement du vélo sur la route crée des frottements causés par la route. Cette fois-ci nous les prenons en compte. On a l'expressions suivante :

$$F_{frottements} = \mathrm{mg}K_f \ (6)$$

Avec : m la masse (vélo + cycliste) en kg, g la constante gravitationnelle, K_f le coefficient de frottement.
A vitesse constante appliquons le PFD :

$$m\frac{dV}{dt} = \sum_{k=0}^{n} \vec{F}_{ext} = 0 = \overrightarrow{F_{air}} + \vec{F} + \vec{P} + \overrightarrow{F_{frottements}}$$

$$0 = -0{,}5 * \rho * Cx * V^2 + \frac{P}{V} - mgsin(\alpha) - mgK_f \quad (7)$$

Faisons un bilan des puissances :

- Puissance ascensionnelle : $P(poids) = mgVsin(\alpha)$
- Puissance de frottements : $P_{frottements} = \mathrm{mg}VK_f$
- Puissance des frottements de l'air : $P_{air} = \mathrm{K}V^3$

On sait que la puissance émise (instantanée) par l'utilisateur doit au supérieur ou égale à la somme de ces trois puissances pour réussir à avancer. Donc on peut écrire :

$$\sum P_{ext} = P_{instantanée}$$

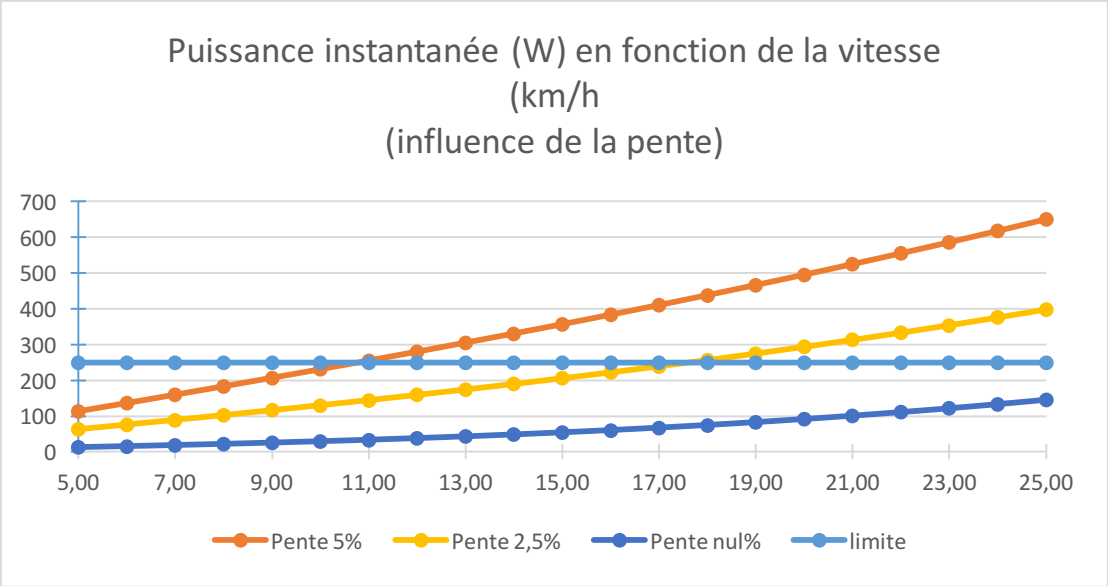
$$P = mgVsin(\alpha) + \mathrm{mg}VK_f + \mathrm{K}V^3 \quad (8)$$

Possédant la relation liant toutes les puissances nous avons réalisé des calculs afin d'observer les différentes influences de chacune des caractéristique (pente, vitesse du vent). Nous avons pris K_f =0,001 et K = 0,26.

6) Evolution de la puissance en fonction des caractéristiques

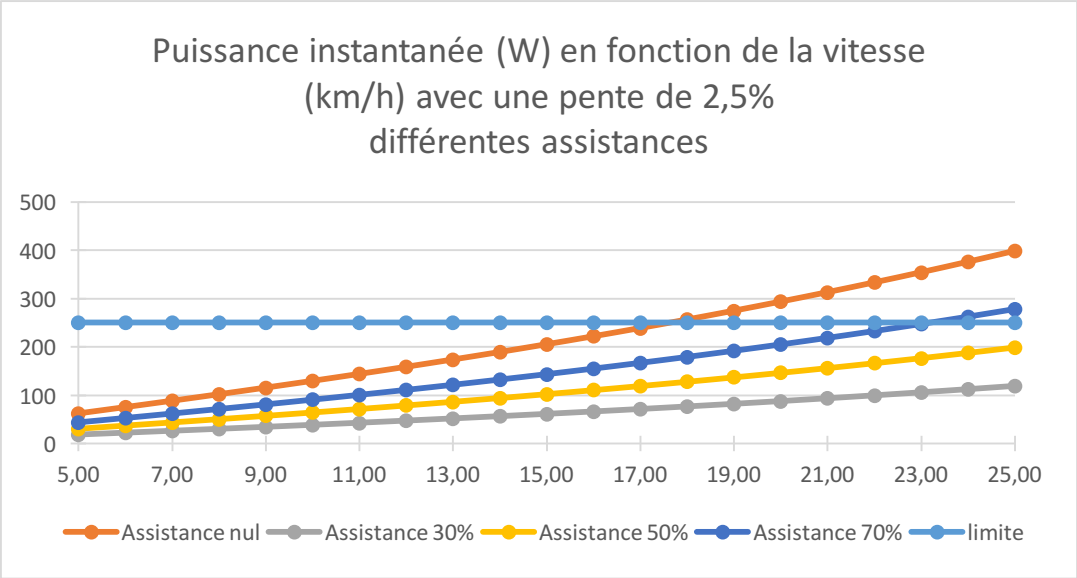
6.1) Influence de la pente

Nous avons calculé la puissance instantanée (en W) en fonction de la vitesse (en km/h) en faisant varié la valeur de la vitesse avec différentes valeurs de pente (en monté). On observe que plus la pente est grande est plus la puissance à fournir par le cycliste doit être forte pour une vitesse donnée.



De plus on peut voir avec la courbe « limite » symbolisant la puissance maximale que le moteur peut fournir (250W) que plus la pente est grande et plus cela va être compliquée pour le moteur de nous fournir une assistance.

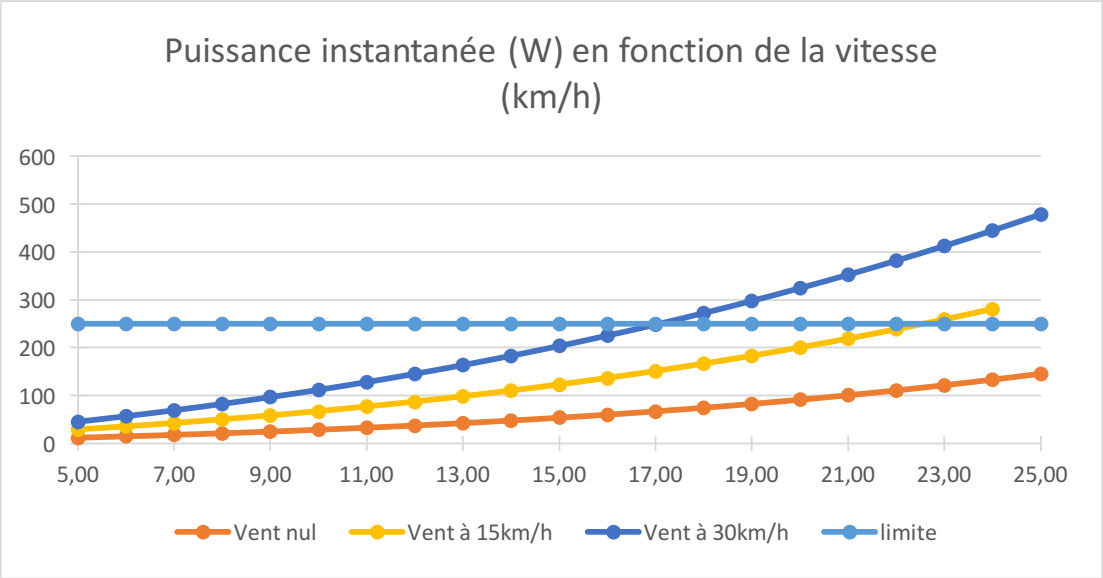
Ensuite nous avons calculé la puissance pour différentes assistances. Comme on souhaite réguler notre système en fonction de la puissance fournie par l'utilisateur. Il est donc logique que l'on calcul un pourcentage de la puissance instantanée (équivalent à la puissance que le moteur devra fournir à chaque instant t).



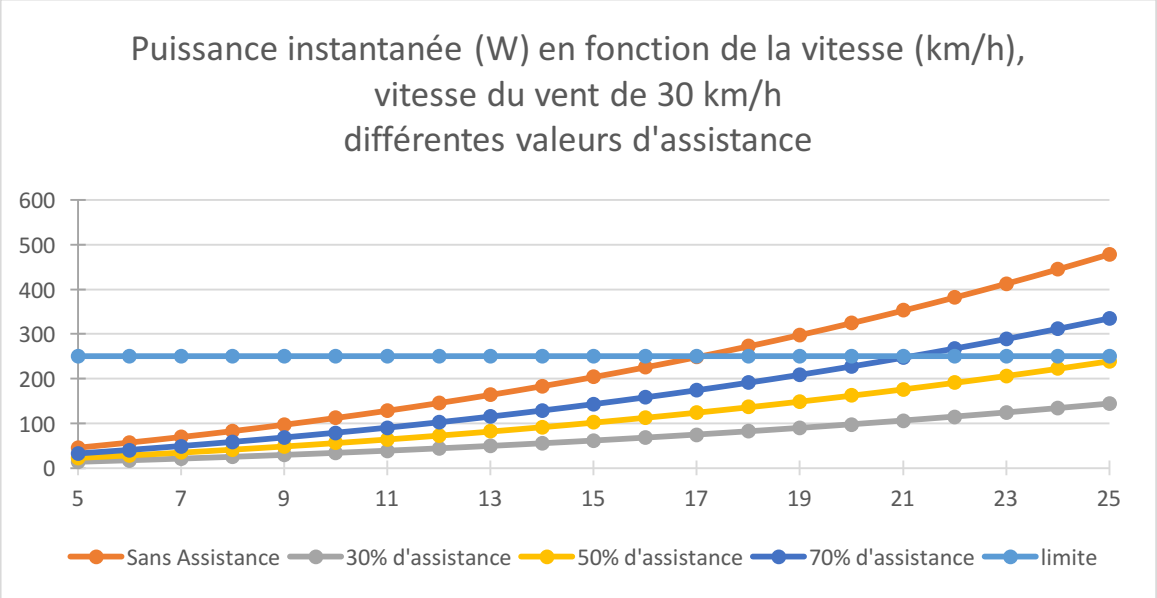
On observe que la puissance à fournir dépasse la valeur limite (250W) pour une assistance de 70% à partir de 23 km/h (valeurs très proche de celle attendues). On peut donc en conclure que le moteur fonctionnera très bien pour une pente maximale de 2,5%.

6.2) Influence du vent

Maintenant que l'on a vu qu'une pente peut avoir de lourdes conséquences sur la puissance à fournir par le moteur nous allons nous intéresser à l'influence du vent.
Pour cela comme pour les calculs réaliser auparavant, nous faisons varier la vitesse avec cette fois-ci différentes valeurs pour la vitesse du vent (augmentation de la puissance de l'air).



On observe que pour une vitesse de vent de 30km/h on dépasse la valeur de puissance limite pour une vitesse de 17km/h. Comme pour les calculs précédents nous allons comparer ces valeurs à l'assistance que le moteur devra fournir.



On remarque que pour une assistance de 70% on dépasse la valeur limite de 250W pour une vitesse supérieure à 21 km/h (valeur assez proche de la valeur attendue).

Par conséquent on peut dire que le vélo pourra très bien être réguler avec une assistance de 70% de la puissance fournie par le cycliste pour un vent de 30km/h.

7) Détermination de la commande

Dans un premier temps nous avons analysé l'asservissement actuel implémenté sur la carte ARDUINO Uno Pro mini. Elle est pilotée par une machine d'état et la commande est alimentée par un correcteur P (proportionnel) simple. Lorsque nous avons testé le vélo nous avons pu réaliser que l'assistance fournie par le moteur est assez brusque. C'est à dire qu'on peut atteindre les 25km/h (vitesse limite) sans pédaler très vite. Alors que nos clients attentes que l'assistance fournie soit proportionnel à l'effort fourni par le cycliste.

Pour faire l'acquisition des données nous avons utilisé nos programmes réalisés en test n°3 (voir partie 4.3).

Dans un premier temps nous devons déterminer le modèle du vélo et le modèle du moteur. Une fois que cela est réalisé nous allons pouvoir déterminer le correcteur respectant les stabilités souhaitées.

7.1) Détermination du modèle du moteur

Afin de déterminer ce modèle, on a du générer un échelon directement sur la carte de commande en modifiant les valeurs de la PWM (2 valeurs différentes).

Les résultats obtenus nous ont permis de faire le lien entre la vitesse de la roue et le courant débité par le moteur.
Ensuite à l'aide de l'outil IDENT[6] nous avons pu déterminer le modèle de notre moteur.
L'autocorrélation du système nous indique qu'il possède une période d'échantillonnage de retard.

[6] Outil MATLAB permettant de faire l'identification de modèle

Après plusieurs essais de différents modèles, notre choix s'est porté sur un modèle ARMAX composé de deux pôles, deux zéros et un retard. Ce modèle est de la forme

$$A(z)y(t) = B(z)u(t) + C(z)e(t)$$

Avec :

$$A(z) = 1 - 1.589z^{-1} + 0.6061z^{-2}$$
$$B(z) = 0.1182z^{-1} - 0.1162z^{-2}$$
$$C(z) = 1 - 0.9544z^{-1} - 0.04556z^{-2}$$

Ce qui nous permet d'avoir notre fonction de transfert (TF = B/A).

Après la mise en continu du modèle, la fonction de transfert est :

$$G(t) = \frac{1.024 + 2.963p}{29.01 + 9.987p + p^2} * e^{-0.05p}$$

Avec Ts (temps d'échantillonnage) =0.05s.

Ce modèle repli tous les critères de validation d'un modèle.

7.3) Détermination d'un Correcteur

Une fois que le modèle a été déterminé on doit calculer un correcteur. Nous avons fait le choix d'étudier un PI[7], pour cela nous créons une fonction permettant d'identifier tous les termes de notre correcteur (Gain, Marge de phase, Dépassement) tout en respectant les règles de stabilités.

Nous avons trouvé un correcteur de la forme :

$$C(t) = \frac{14.06 + 0.5387p}{p}$$

Avec un gain statique Kp= 0.5387 et Ti= 0.0383.

[7] Proportionnelle Intégrale

7.4) Schéma SIMULINK de la commande

Une fois que l'on a identifié tous ce que l'on voulait il nous suffit de dresser un schéma SIMULINK avec un « anti wind-up » afin d'asservir le système en limitant les valeurs trop importante et donc protégé le moteur (voir annexe 1).

7.5) Implémentation de la régulation

Une fois que l'on a identifié la commande et la consigne que l'on souhaite, on implémente la régulation réalisé sur SIMULINK directement sur l'ARDUINO. La commande est de la forme :

$$u(k) = u(k-1) + kp\big(y(k)^* - y(k)\big) + kp\left(1 - \frac{Ts}{Ti}\right)\big(y(k-1)^* - y(k-1)\big) + \frac{Ts}{Ti}(u_s(k-1) - u(k-1))$$

Avec :

- u(k) : Commande ➔ valeur de PWM envoyé au moteur
- y(k) : sortie ➔ courant débiter par le moteur
- $y(k)^*$: sortie consigne ➔ courant souhaité que débiter par le moteur
- $u_s(k)$: Commande saturé

La commande saturé se défini de telle sorte :

Si commande > 100 alors commande saturé = 100

Si commande < 0 alors commande saturé = 0

Sinon commande saturé = commande

Cela applique l'effet « anti wind-up » souhaité.

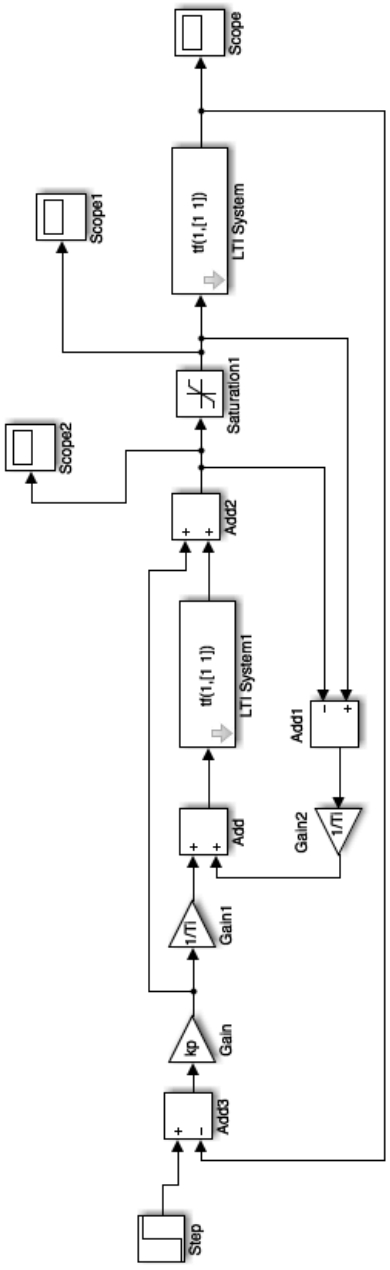
Le courant que l'on souhaite que le moteur débite est calculé en fonction de la vitesse de pédalage et du niveau d'assistance demandé (voir annexe 2).

Après implémentation, on peut observer que l'accélération et mais brusque et qu'elle dépend de la vitesse de pédalage de l'utilisateur (voir annexe 3).

8) Conclusion

Nous avons fait le choix de participer au projet « HopeAndBike » et ainsi de contribuer à l'avancée de ce dernier. Nous avons répondu aux attentes des clients car nous avons réussi à modifier l'asservissement du système. En remplaçant la machine d'état par notre régulation de la puissance (en jouant sur l'intensité du courant que le moteur envoi au système).
Pour cela nous avons optimisé le système de commande, déterminer plusieurs modèles liés au système ainsi que réaliser des simulations.
Ainsi nous avons mobilisé toutes les compétences que l'on a pu acquérir tout le long de notre formation pour venir à bout de ce projet. Notamment les cours d'identification et d'asservissement.
Ce projet nous as permis de travailler sur un système mécatronique réel. C'est-à-dire qu'il met en œuvre les domaines de la mécanique, de l'électronique, de l'automatique et de l'informatique. Les nombreux problèmes que nous avons rencontrés tout le long du projet ont permis de comprendre des choses que l'on n'avait pas assimilées pendant les cours magistraux.

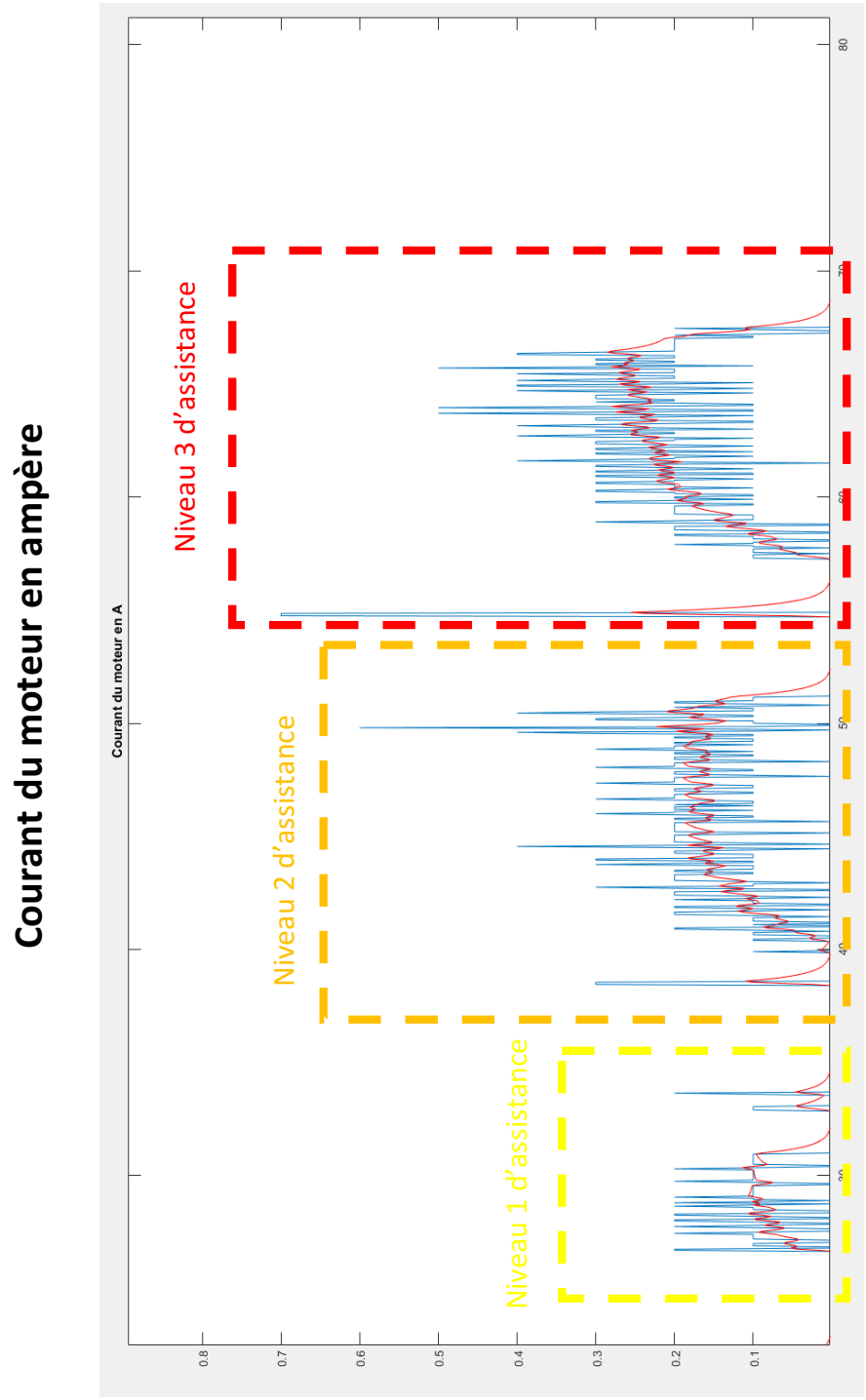
Annexe 1

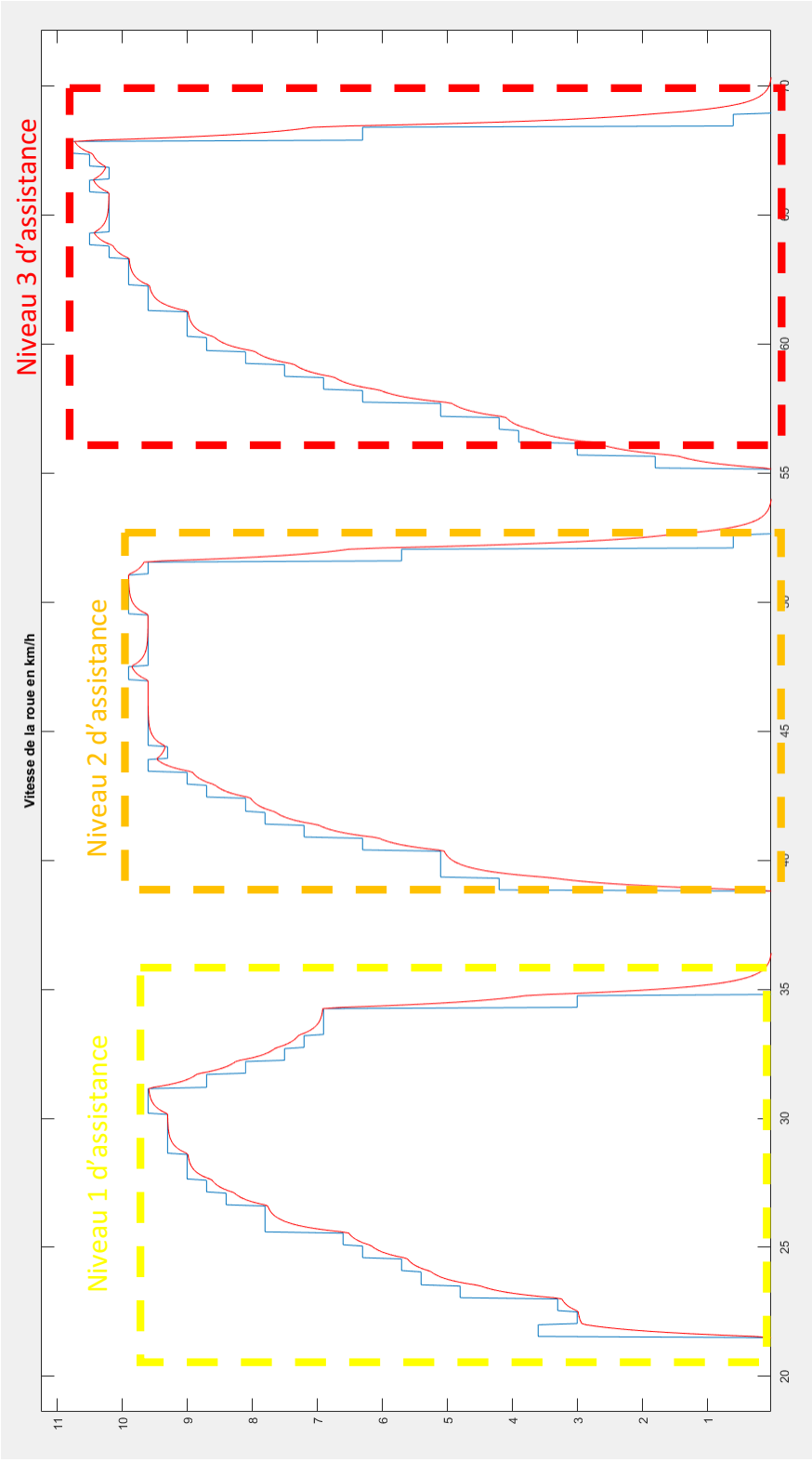

Scope
tf(1,[1 1])
LTI System
Scope1
Saturation1
Scope2
Add2
tf(1,[1 1])
LTI System1
Add1
Add
Gain2
1/Ti
Gain1
1/Ti
kp
Gain
Add3
Step

Annexe 2

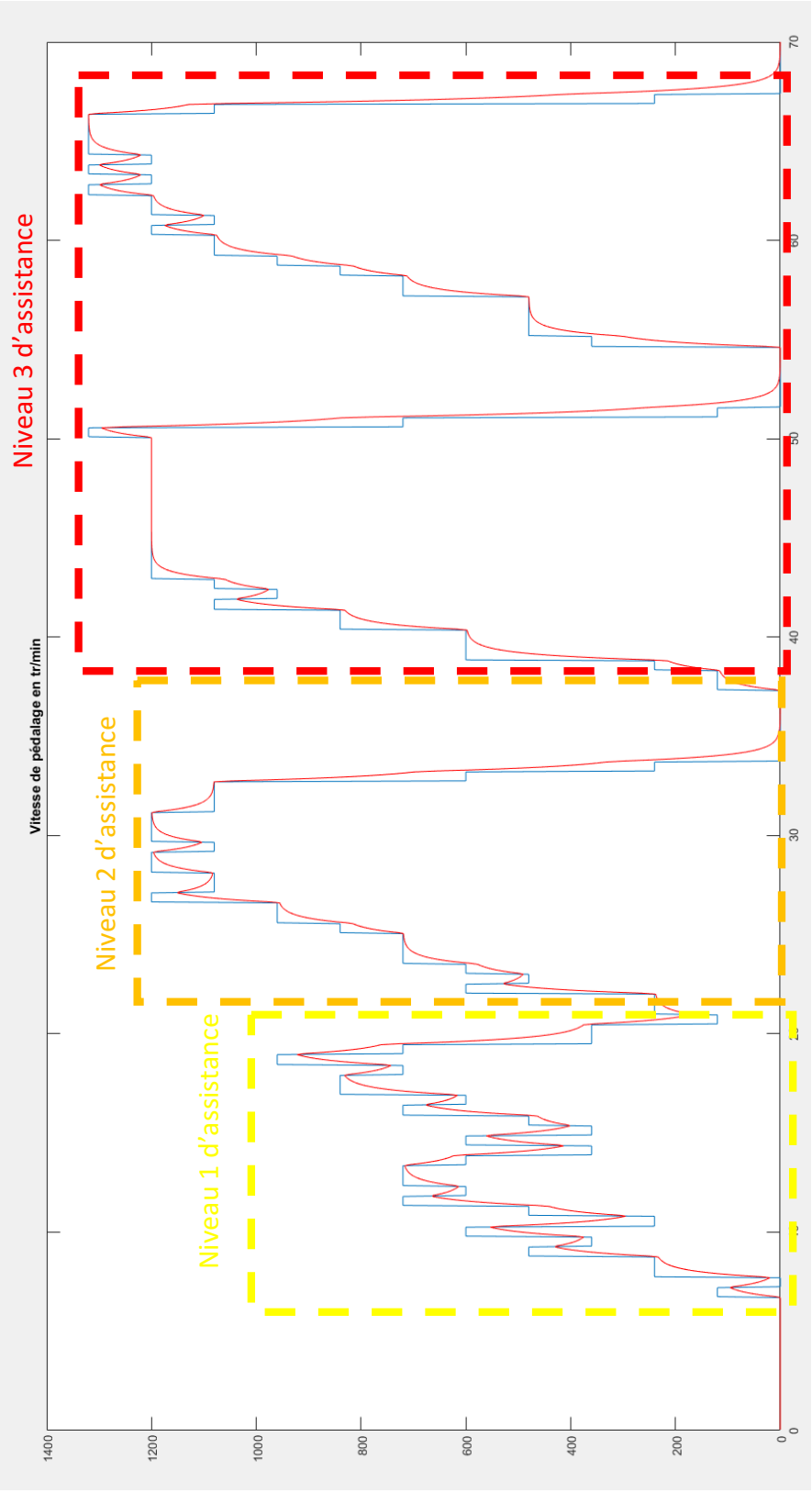
```
CurrentConsigned=(((float)ConsigneDesiree)/100)*0.17*parametre_réglable*u8VitessePedale;

 Current=(CurrentConsigned-s16CourantMotUnitAmp);
 commande=commande_old + (kp*Current) + (kp*(1-(Ts/Ti))*Current_old) + ((Ts/Ti)*(commande_sat_old-commande_old));
 commande_old=commande;
 Current_old=Current;

 if(commande > 100)
 {
  commande_sat=100;
 }
 else if(commande < 0)
 {
  commande_sat=0;
 }
 else
 {
  commande_sat=commande;
 }

 commande_sat_old=commande_sat;
 PWM_PID=commande_sat;

 i8PWM=PWM_PID; //PWM appliqué au moteur
```

Annexe 3


Courant du moteur en ampère
Courant du moteur en A
Niveau 3 d'assistance
Niveau 2 d'assistance
Niveau 1 d'assistance
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
80
60
40


Vitesse de la roue en km/h
Niveau 3 d'assistance
Niveau 2 d'assistance
Niveau 1 d'assistance
Vitesse de la roue en km/h

Vitesse de pédalage en tr/min



PWM fourni au moteur