Samuel Cloutier – Contributions au projet de Design III, 28 janvier 2016

**Contrôle manuel d’une roue unique:**

* *Moteur et pont H :*

Le pont H est alimenté par une tension principale de 12V ainsi qu’une tension secondaire de 5V pour les circuits électroniques. La tension de sortie de la roue de test est contrôlée par un PWM envoyé au pont H. Celui-ci est, pour l’instant, généré grâce au modulateur de fonction disponible au laboratoire.

(Plus de details à ce sujet écris par Amine)

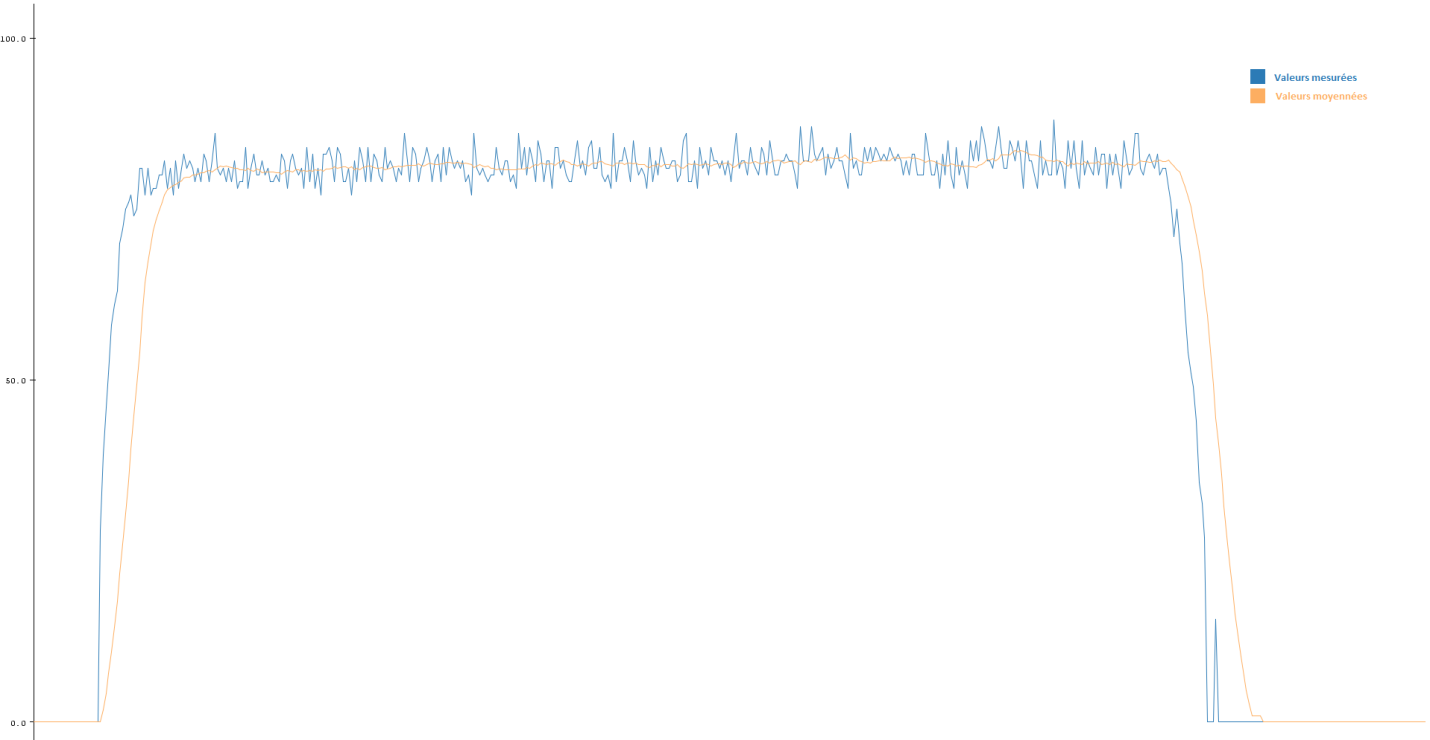
* *Capteur de vitesse :*

Chaque roue possède un capteur de vitesse rotationnelle devant être alimenté en DC. Un signal carré est générée par le capteur à une fréquence variant en fonction de la vitesse rotationnelle du moyeu central. Deux senseurs sont placés sur le moyeu du moteur afin de générer chacun 16 pulsations par rotation du moyeu. Étant donné que ce dernier entraine la roue avec un rapport de transmission de 100:1 (100 tours de moteur pour un tour de roue), chaque rotation complète de la roue est caractérisée par 1600 front montant du signal. Les deux senseurs produisent des signaux à la même fréquence, mais légèrement asynchrones. La différence entre les signaux permet de déduire le sens de rotation de la roue, alors que leur fréquence permet de calculer sa vitesse. Étant donné que dans ce projet, le microcontrôleur devant faire la mesure de vitesse est aussi en contrôle de l’alimentation des roues et que l’on connait naturellement leur direction, on peut ignorer l’un des senseurs et ne mesurer que la fréquence de l’autre.

* *Microcontrôleur Arduino:*

Grâce à un microcontrôleur Arduino, l’intervalle entre chaque pulsation peut être lue aisément en utilisant la fonction *pulseIn().* Celle-ci fait une lecture d’une broche du microcontrôleur et retourne le temps entre deux variations d’un signal, en microsecondes. La vitesse en RPM de la roue est ensuite calculée grâce à l’équation suivante :

La valeur v ainsi obtenue est finalement passée dans une moyenne glissante de 16 points. La taille de la moyenne glissante en points d’analyse est importante car le PID de la roue ne considère que la valeur moyennée pour calculer l’erreur. Il faut donc que le tableau de valeurs ne soit pas trop large afin de changer le moins possible le procédé de la roue, mais doit avoir assez de point pour retourner une valeur stable au PID.

  
Fig 1 : Moyenne glissante sur 16 points du signal, en RPM, retourné par l’Arduino

La figure ci-haut a été générée grâce au *Serial Plotter* inclus avec le IDE Arduino. Bien que ne permettant pas d’enregistrer les données pour analyse subséquente, il permet néanmoins de les visualiser rapidement. Pour l’identification du procédé avec Matlab, un script d’acquisition est développé. Celui-ci est décrit dans la section suivante.

**Acquisition de données de vitesse dans Matlab :**

* *Port Série connecté à l’Arduino :*

Le logiciel Matlab permet de communiquer à travers un port série de l’ordinateur. Dans le cas de la communication avec un Arduino, les drivers prennent automatiquement en charge le protocole du câble USB et l’interfacent comme un port série. Matlab comprend un module de communication série permettant d’ouvrir, lire et écrire sur un port tel que « *COM4 »* par exemple. Un script à cet effet est donc développé afin de faciliter l’acquisition de données, les afficher en temps réel ainsi que les conserver dans un format déjà compatible avec la librairie d’identification de procédés de Simulink.

* *Acquisition de données d’une roue soumise à un échelon :*

Avec l’objectif d’identifier le procédé de chaque roue à l’aide de Simulink, on démarre la capture de données dans Matlab avec le moteur au repos. On soumet ensuite celui-ci à un échelon de 12V, tout en s’assurant que la vitesse s’affiche bien sur le graphe en temps réel. Lorsque la courbe semble avoir atteint un régime permanent, on peut terminer la capture en fermant le graphique. Les données ainsi collectées sont le résultat du procédé de la roue mesurée, prétraitées par la moyenne glissante employée dans le code du microcontrôleur. Un système PID peut ainsi être testé en simulation et implanté rapidement, tout en étant rapidement modifiable.

* *(À venir) Identification du système de chacune des roues (avec charge) :*

Le procédé capturé à la section précédente pourra être utilisé dans Simulink afin d’en identifier le comportement. Des paramètres d’un PID pourront alors être générés et implantés dans le microcontrôleur et modifiés lorsqu’une charge utile sera installée sur le robot.

* *(À venir) Asservissement de chaque système de façon indépendante :*

L’identification de chaque roue devra être fait de façon indépendante, afin d’assurer un asservissement optimal de celles-ci. Le microcontrôleur sera configuré pour recevoir quatre commandes indépendantes les unes des autres, soit une pour chaque roue. Cette approche permettra des possibilités de déplacement qui seraient impossible avec une approche plus naïve telle que le couplage de l’asservissement des roues parallèles.

* *(À venir) Écriture d’un module en Python afin de contrôler les mouvements :*

Une classe Python sera développée pour prendre en charge plusieurs tâches. En plus d’être le point de liaison entre le logiciel d’intelligence artificielle et le matériel (microcontrôleur), celle-ci fera la translation de méthodes simples telles que *rotate(degree, rotateSpeed)* et *foward(distance, speed)* en des commandes pour chaque roues du robot.