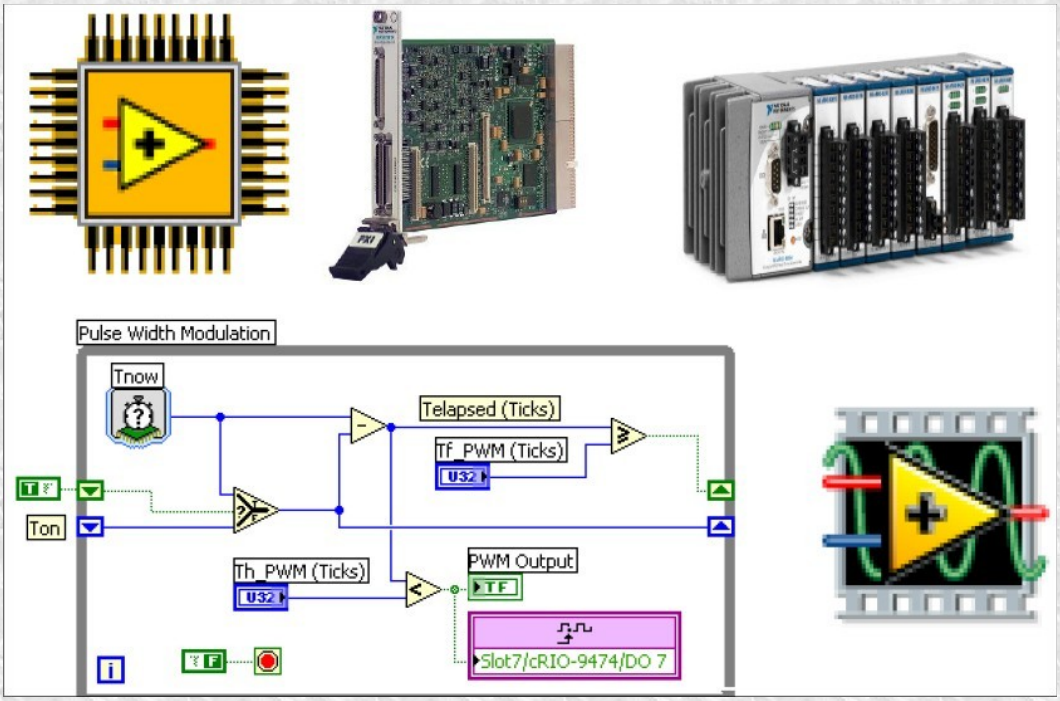
MICHAUT Théo MOINE Jérémy

PROJET INDUSTRIEL 2016

RÉGULATEUR DE VITESSE



Licence professionnelle VEGA IUT BM

REMERCIEMENTS

En premier lieu on remercie notre enseignant Mr GAVIGNET, pour nous avoir offert la possibilité de travailler sur ce projet. Également pour le suivi régulier et l'aide apportée à la réussite de celui ci.

Nous tenons à remercier Mr HUBERT, qui nous à permis de créer et monter différentes cartes, ainsi que pour le temps accordé aux besoins techniques de la réalisation du projet tels que la maquette ou la roue dentée.

Enfin nous voulons remercier de manière générale l'ensemble de l'équipe enseignante. Les connaissances apportées au cours de la formation, ainsi que les compétences développées nous auront permis de passer cette année d'étude dans de bonnes condition, et de faciliter notre future insertion dans la vie professionnelle.

Régulateur de vitesse

SOMMAIRE

Régulateur de vitesse

INTRODUCTION

Intitulé : Le but du projet est d'implanter dans le FPGA toutes les fonctions permettant d'assurer une régulation en vitesse similaire à celle en automobile. Pour cela nous allons utiliser une platine FPGA.

Le système embarqué proposé est une platine FPGA de chez National Instruments (my RIO) programmable graphiquement à partir de LAbVIEW FPGA. A partir de ce système, on désire implanter dans le FPGA toutes les fonctions permettant d'assurer une régulation en vitesse similaire à celle utilisée dans l'automobile :
- Commande par PWM d'un moteur électrique (simulant un moteur thermique)
- Mesure de la vitesse de rotation du moteur (capteurs, mesures,….)
- Implantation de l'asservissement en vitesse
- Réalisation de l'Interface Homme Machine permettant de contrôler l'ensemble.

Le principe est simple, mais nécessite trois modes de fonctionnements différents :

Le mode normal, l'utilisateur envoi une commande d'accélération au moteur, qui se mettra à tourner à une certaine vitesse.

Le mode limiteur de vitesse. Lorsqu'il est activé l'utilisateur peut continuer à envoyer sa commande d'accélération, mais la vitesse de moteur ne pourra pas dépasser une valeur de consigne.

Le mode régulateur. En cas d'utilisation, le conducteur n'a même plus besoin d'envoyer de commande d'accélération. Le moteur lit la vitesse de consigne, et ira de lui même se caler à la vitesse prévue.

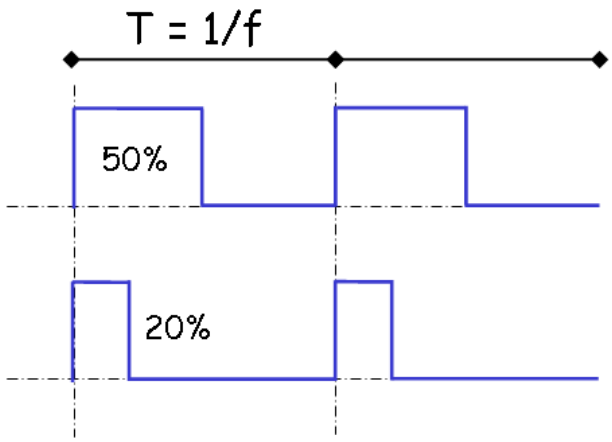
Ces trois modes de fonctionnement devront être gérés via un My Rio, on s'arrangera pour créer une petite maquette à l'aide d'un moteur 12V. La régulation sera assurée via un pupitre IHM. Il sera donc nécessaire de créer une face avant à l'aide de LabVIEW, afin de pouvoir paramétrer et contrôler les différents modes.

Régulateur de vitesse

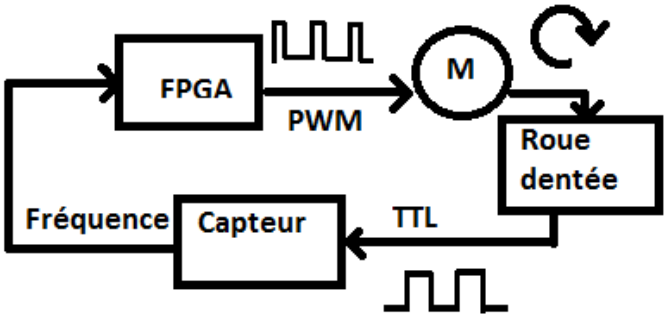
ANALYSE FONCTIONNELLE : IDENTIFICATION DES FONCTIONS

Commande de vitesse moteur : Afin de faire varier la vitesse du moteur il suffit de faire varier sa tension d'alimentation. Ainsi en faisant varier la valeur moyenne de sa tension, il sera possible d'accélérer ou de freiner le moteur. Une façon simple de faire varier la valeur moyenne d'une tension est d'utilisée une commande en PWM (Pulse With Modulation). Un signal en PWM voit sa largeur d'impulsion modulée, et peut ainsi faire varier la valeur moyenne d'une tension entre 0 et la valeur maximale de cette tension.

On se placera dans la cas typique en automobile, une PWM à la fréquence de 1kHz. Comme le moteur ne simulera pas la marche arrière, il ne sera pas nécessaire d'inverser son sens de rotation.



Lecture de la vitesse moteur : Afin d'assurer une régulation, il est nécessaire de connaître en permanence la vitesse de rotation du moteur. Toujours en se basant sur l'automobile, on peut utiliser un capteur de position (ABS), qui permettra de compter un nombre d'impulsion. En connaissant le nombre d'impulsions par tour du moteur, il sera possible de connaître aussi sa vitesse.



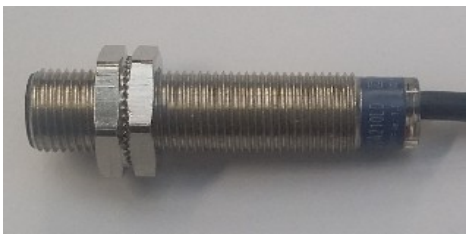
Consigne de vitesse : On utilisera LabVIEW afin de créer une face avant, servant d'interface Homme-Machine. Les différents paramètres de la régulation, ainsi que des boutons pour les modes et la consigne devront apparaître.

Régulateur de vitesse

ANALYSE FONCTIONNELLE : CHOIX DES COMPOSANTS

Commande de vitesse moteur : On utilisera un pont en H afin de faire varier la valeur moyenne de la tension moteur, avec une commande PWM en entrée. Ce pont en H ne sera câbler que dans un sens, et l'autre entrée sera inutilisée. Nous avons choisi un L298 (Annexe 1). Ce double pont en H permettra de contrôler la vitesse moteur, avec un seul sens de rotation, et supportant un courant maximal de 2A.

Le signal PWM sera isolé galvaniquement grâce à un optocoupleur.



Lecture de la vitesse moteur : Les capteurs utilisés dans l'automobile tels le capteur ABS, sont des capteurs inductifs (ou capteur de proximité). Il n'y a donc aucune liaison mécanique (pas de frottements) et les informations sont transmises par un champs magnétique. On à choisi le XS1N12, ce capteur est alimenté en 12V, et sa sortie commute de l'état haut à l'état bas en cas de détection.

Pour mettre en forme le signal du capteur on utilisera un comparateur LM311, qui ramènera le signal de détection sous la forme d'un signal TTL classique.

La fréquence de ce signal TTL est une image de la vitesse moteur. On à choisi un convertisseur fréquence-tension 2907N (Annexe 2) afin de pouvoir traiter une tension analogique, qui sera image de la vitesse moteur.

Afin d'économiser une alimentation, il sera nécessaire de créer un 5V pour les signaux logiques, à partir du 12v fourni pour les signaux de puissance. On utilisera un régulateur de tension 7805.

Les schémas électriques des fonctions sont fournies en annexe.

ANALYSE FONCTIONNELLE : TESTS, INTÉGRATION, VALIDATION

Une fois les fonctions identifiées et les composants choisis, on réalise sur plaque labdec des essais de ces fonctions. Le schéma électrique est fourni en annexe 3. L'alimentation à été découplée pour garantir 5V en sortie du régulateur. On à du filtrer la composante continue du signal TTL du capteur afin de pouvoir détecter les passages par 0V dans le 2907N. Les valeurs des résistances ont été choisies de cette manière :

$$Vs = F_1 \times Vcc \times R_1 \times C_1$$

$$F_1 = \frac{Vs}{Vcc \times R_1 \times C_1}$$

$$\frac{1}{12 \times 10^6 \times 10^{-8}} = 8{,}3\ Hz/V$$

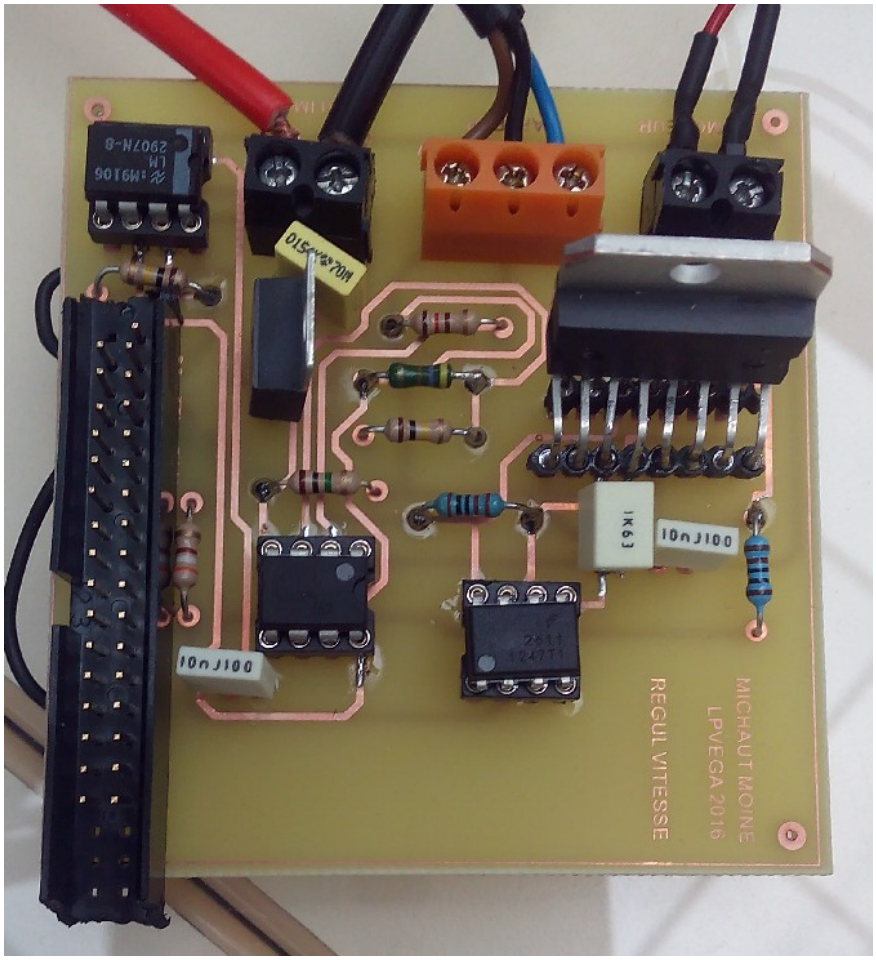
$$F_1 = Fréquence\ entrée$$

On à donc choisi R1=1MΩ et C1=10nF pour avoir une sensibilité d'environ 8Hz/V, comme la roue dentée possède 12 dents pour une vitesse de rotation de 180 tr/min, cela nous donne une résolution maximale de 36Hz soit 4,4V au maximum.

En simulant la PWM grâce à un GBF, on parvient à faire varier la vitesse du moteur, et à relever une tension analogique dépendant de cette vitesse. Nous commençons donc par créer une première carte d'essai pour valider les fonctions, sans connecteur pour se brancher sur le FPGA.

Régulateur de vitesse

Une fois les fonctions validées, on reprend la carte en y intégrant le connecteur 34 broches.

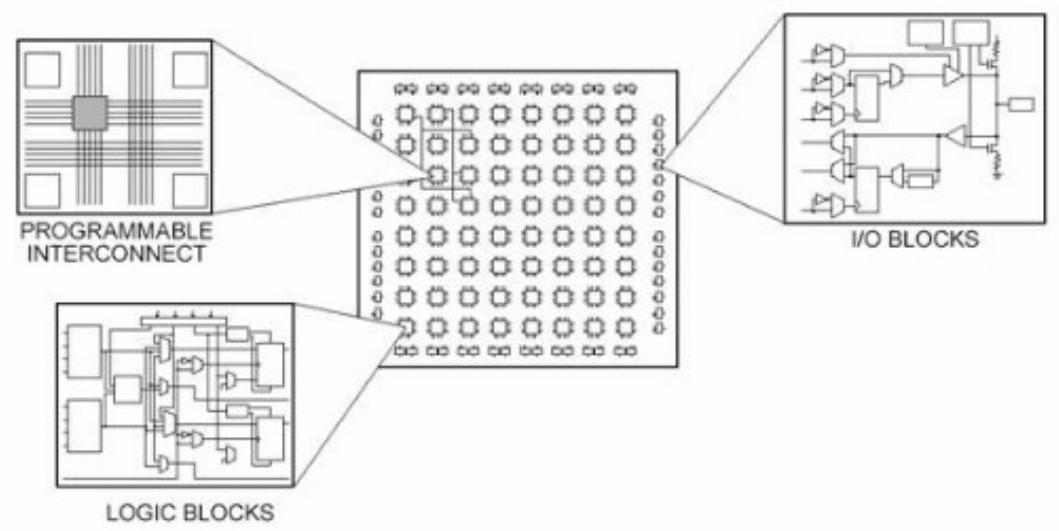


Cette carte étant fonctionnelle, on peut commencer à envisager la création d'une maquette et la programmation des différents modes.

Pour mettre en marche cette carte, il suffit de brancher le connecteur 34 broches, l'alimentation 12V ainsi que le moteur et le capteur (marron 12V, noir data et bleu GND).

Les plans et schémas des typons sont fournis en Annexe.

SUPPORT DE PROGRAMMATION : LES FPGA



Un circuit logique programmable, ou réseau logique programmable, est un circuit intégré logique qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires et bascules logiques librement connectables (c'est justement la reconfiguration, ou programmation, du composant qui définit les connexions faites entre portes logiques).

La plupart des FPGA modernes sont fondés sur des cellules SRAM aussi bien pour le routage du circuit que pour les blocs logiques à interconnecter.Un bloc logique est de manière générale constitué d'une table de correspondance (LUT ou Look-Up-Table) et d'une bascule (Flip-Flop en anglais). La LUT sert à implémenter des équations logiques ayant généralement 4 à 6 entrées et une sortie. Elle peut toutefois être considérée comme une petite mémoire, un multiplexeur ou un registre à décalage.

Comme la configuration (routage et LUT) est faite par des points mémoire volatils, il est nécessaire de sauvegarder le design du FPGA dans une mémoire non volatile externe, généralement une mémoire Flash série, compatible « JTAG ». Certains fabricants se distinguent toutefois par l'utilisation de cellules EEPROM pour la configuration, éliminant le recours à une mémoire externe, ou par une configuration par anti-fusibles (la programmation par une tension élevée fait « claquer » un diélectrique, créant un contact). Cette dernière technologie n'est toutefois pas reconfigurable.

Les FPGA modernes sont assez vastes et contiennent suffisamment de mémoire pour être configurés pour héberger un cœur de processeur ou un DSP, afin d'exécuter un logiciel. On parle dans ce cas de processeur softcore, par opposition aux microprocesseurs hard-core enfouis dans le silicium.

Les FPGA sont utilisés dans diverses applications nécessitant de l'électronique numérique (télécommunications, aéronautique, transports…). Ils sont également utilisés pour le prototypage d'ASIC.

Les procédés technologiques de base pour les composants programmables sont les suivants :

SRAM - (Static Random Access Memory). Programmables à volonté et in situ. Habituellement en technologie CMOS.

EPROM (UVPROM) - (Erasable Programmable Read-Only Memory). Peuvent être effacés (et reprogrammés) par exposition aux rayons ultra-violets. Technologie CMOS, en cours de disparition au profit de l'EEPROM.

EEPROM - (Electrically Erasable Programmable Read-Only Memory). Peuvent être effacés et reprogrammés à volonté. Quelques-uns peuvent être programmés in situ (souvent par une connexion JTAG). Technologie CMOS.

Flash - (Flash-erase EPROM). Mêmes propriétés qu'EEPROM mais avec une densité supérieure (donc avec un coût inférieur pour une complexité donnée). Technologie CMOS.

Fusible - Programmables une seule fois. Technologie bipolaire.

Anti-fusible - Ne sont programmables qu'une seule fois. Technologie CMOS.

Afin de pouvoir finaliser un FPGA, il est nécessaire d'utiliser un langage de description matériel ou bien un outil de saisie graphique. Après compilation de cette description, on obtient un fichier de configuration pour le FPGA choisi. VHDL et Verilog sont les deux langages de description les plus répandus.

SUPPORT DE PROGRAMMATION : LE MYRIO 1900

Basé sur l'architecture bien établie d'E/S reconfigurables (RIO) de National Instruments, NI myRIO place les E/S personnalisables et performantes temps réel du processeur ARM Cortex-A9 double cœur dans les mains des étudiants. La plate-forme myRIO est disponible en deux versions : la version sous boîtier (NI myRIO-1900) offre le WiFi prêt à l'emploi, trois ports d'E/S et un boîtier conçu pour les étudiants.

Voici une partie des fonctions proposées par le myrio 1900 :

-10 entrées analogiques, 6 sorties analogiques, 40 lignes d'E/S numériques

-WiFi, DEL, bouton poussoir et accéléromètre intégrés

-Xilinx FPGA et processeur ARM Cortex-A9 double cœur

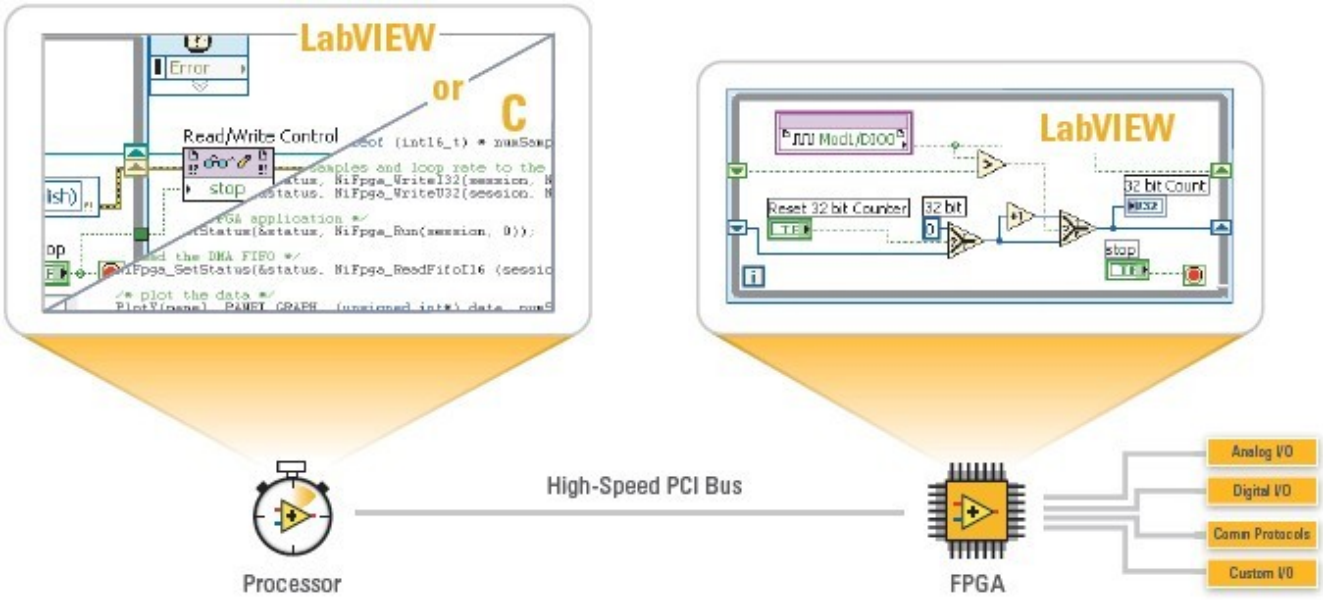
-Programmable en LabVIEW ou en C

Le schéma d'un des connecteurs est fourni en Annexe 6.

On utilisera LabVIEW pour programmer ce FPGA.

Régulateur de vitesse

LabVIEW FPGA



Le Module NI LabVIEW FPGA étend le développement graphique LabVIEW aux cibles FPGA sur le matériel d'E/S reconfigurables (RIO) de NI. LabVIEW FPGA donne aux développeurs la possibilité de concevoir plus efficacement des systèmes complexes en leur fournissant un environnement de développement intégré, un écosystème important de bibliothèques d'IP, un simulateur haute fidélité et des fonctionnalités de mise au point.

L'apprentissage et l'utilisation du HDL est un processus fastidieux et chronophage. Le Module LabVIEW FPGA fournit une approche par programmation graphique qui simplifie l'interfaçage aux E/S et la communication des données.

Le Module LabVIEW FPGA intègre des fonctionnalités de simulation et des outils de mise au point, pour vous permettre de repérer autant d'erreurs d'implémentation que possible avant de commencer la compilation. Dans le cadre de la simulation, vous pouvez mettre au point votre code à l'aide des fonctionnalités de débogage de LabVIEW comme l'animation de l'exécution, les points d'arrêt et les sondes.

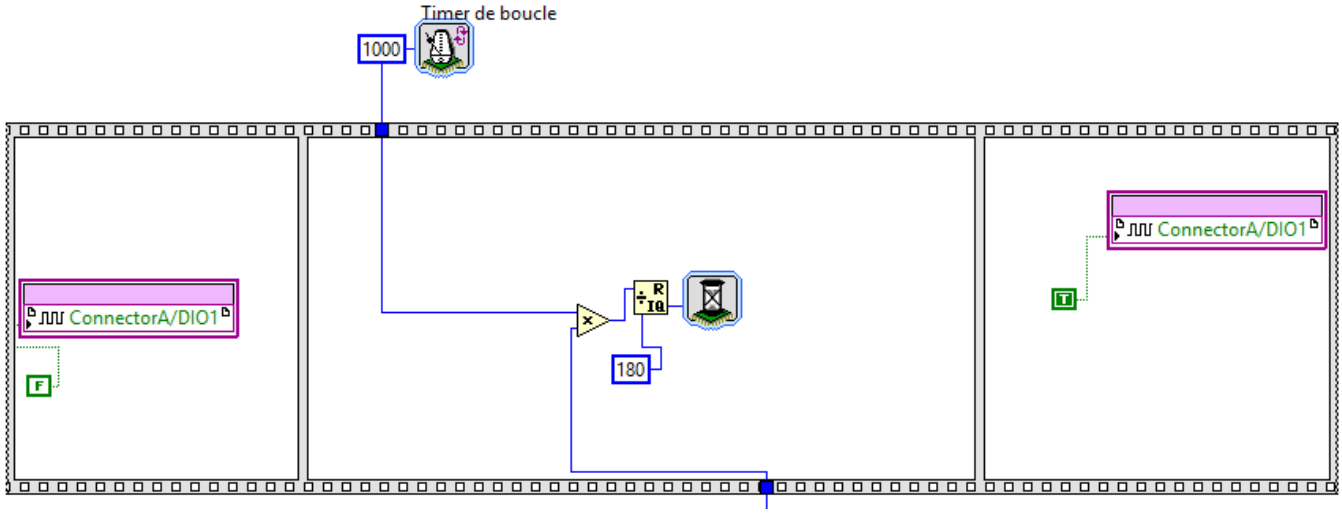
En plus d'être très intuitif, il à l'avantage d'embarquer de nombreuses fonctions utiles telles que des convertisseurs analogique-numérique, de nombreux outils de contrôles et des séquences temporelles et concurrentes faciles à utiliser.

Régulateur de vitesse

ASSERVISSEMENT EN BOUCLE FERMÉE : COMMANDE PWM

Il faut maintenant réussir à piloter le moteur grâce à LabVIEW en envoyant un signal PWM depuis le myrio. On choisira pour ça une de ses sorties numérique (DIO_1). Cette PWM sera fixe à la valeur de 1kHz. Afin de réaliser cette PWM, on utilisera une structure séquentielle. Il faut déjà fixer un rapport cyclique (entre 0 et 1) qui correspondra à la valeur moyenne désirée (entre 0 et 100 %).

Ce rapport cyclique sera multiplié par la période du signal, afin de définir combien de temps le signal restera dans l'état faux. Le rapport cyclique devra dans un premier temps apparaître sous la forme d'une commande sur la face avant, mais en mode régulation il sera remplacé par une valeur de consigne.



Ici le moteur est supposé tourner à la vitesse maximale de 180km/h, on à donc ramener le rapport cyclique entre 0 et 180 afin de faciliter la suite du traitement. Le timer de boucle est réglé sur 1 000µs soit 1ms (1kHz). La sortie DIO_1 restera à l'état faux tant que le délai d'attente ne sera pas atteint.

Cette PWM fonctionne et est donc prête à être envoyé sur la carte pour piloter la vitesse du moteur.

Régulateur de vitesse

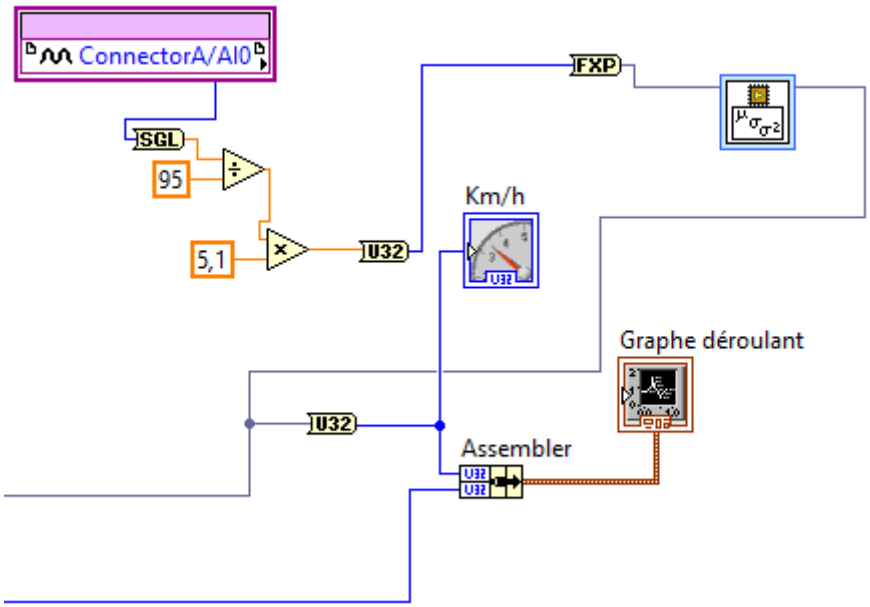
ASSERVISSEMENT EN BOUCLE FERMÉE : LECTURE DE VITESSE

Il existe deux façons de relever la vitesse du moteur à partir du capteur inductif :

-Compter le nombre de tops/tour en une seconde. Le moteur est équipé d'une roue à 12 dents, et peut tourner à une vitesse maximale de 180 tr/min. Donc il suffit de compter le nombre de tops/s et de diviser par quatre pour retrouver la fréquence de rotation. Un rapide calcul permet de la mettre sous forme de km/h (entre 0 et 180). Il suffit donc de récupérer le signal TTL directement en sortie du comparateur.

-Le signal TTL à été convertit en tension analogique à l'aide d'un 2907N, il suffit de traiter cette tension analogique comme une image de la vitesse.

Nous avons préféré la seconde solution pour des difficultés de programmation. Les entrées analogiques sont codées sur 12 bits, il est donc nécessaire de remettre en forme la valeur acquise pour récupérer une vitesse.

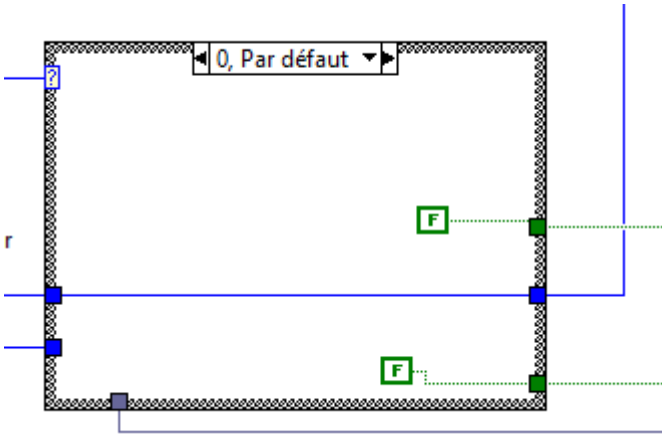


Régulateur de vitesse

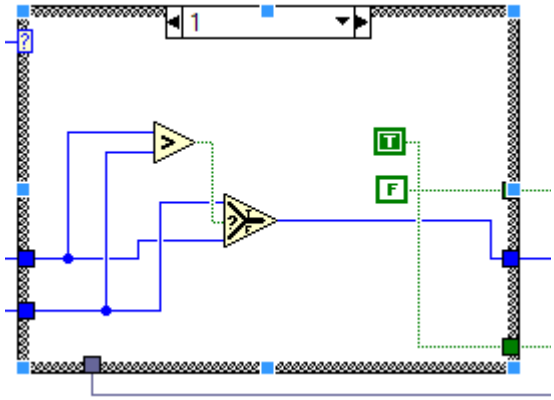
ASSERVISSEMENT EN BOUCLE FERMÉE : CORRECTEUR PID

Pour choisir le mode dans lequel on veut être, on utilise une structure conditionnelle. Les deux premiers modes ne nécessitent pas de codes particuliers :

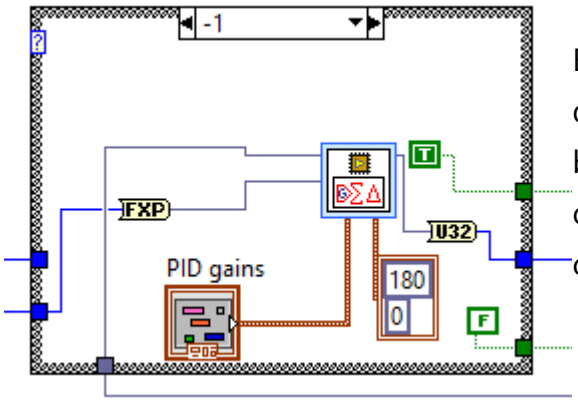
Mode normal :



Mode limiteur :



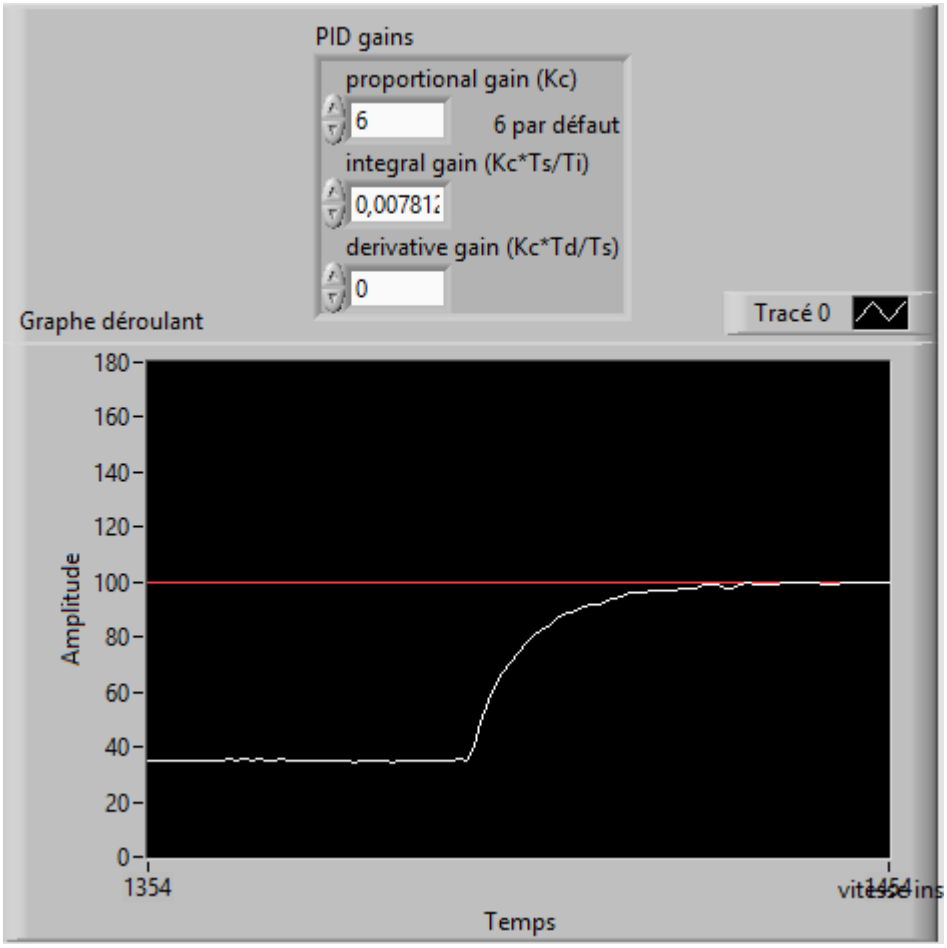
Le mode régulateur utilisera un correcteur PID, permettant d'asservir la vitesse du moteur par rapport à la consigne. On utilisera un graphe déroulant pour paramétrer le PID et ainsi obtenir un asservissement le plus fluide possible.



En consigne du PID on envois la même consigne que pour le régulateur et en valeur courante on boucle la valeur de la vitesse du moteur. Le correcteur aidera la valeur de la vitesse à atteindre la consigne en fonction des paramètres.

Régulateur de vitesse

Pour régler un correcteur PID il est nécessaire d'utiliser au moins l'action proportionnelle P et, en fonction des utilisations, un petit peu des actions dérivée D et intégrale I. On à donc fais des essais jusqu'à obtenir des photos de P, I et D.



Grâce au graphe déroulant on peut régler les paramètres avec précision.

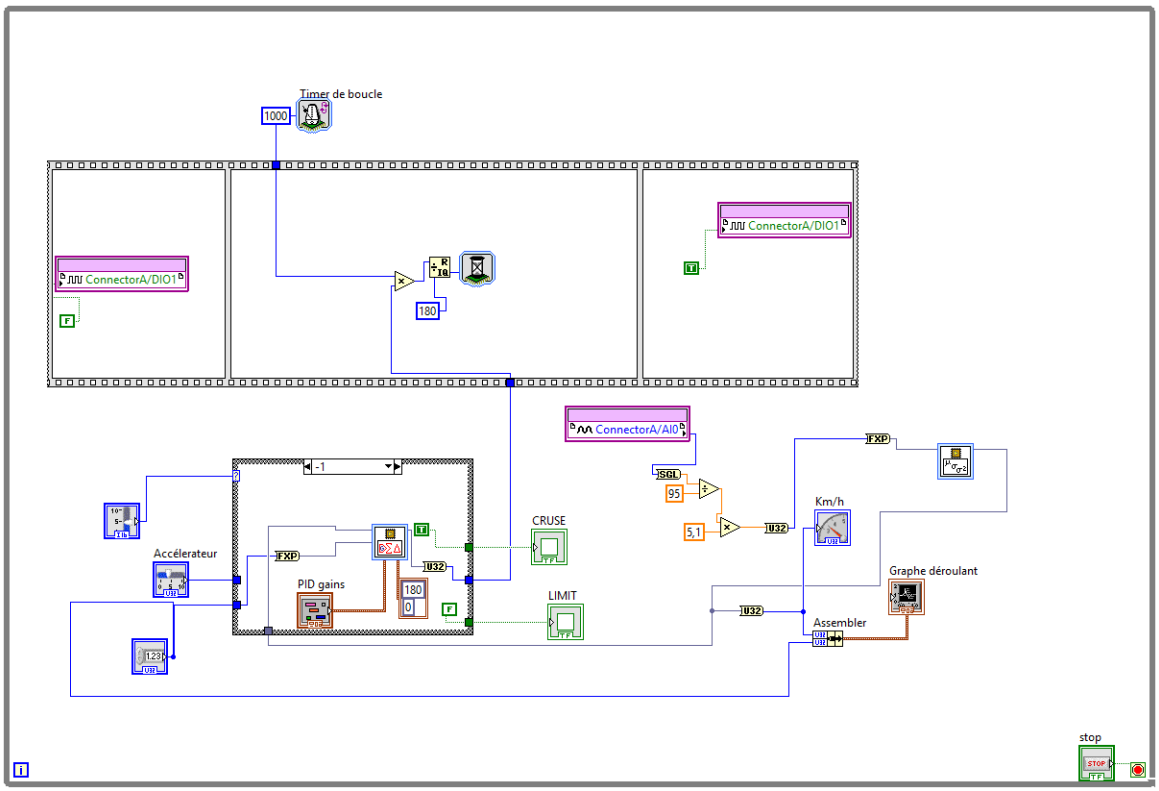
Afin d'obtenir une bon asservissement de vitesse de la part du correcteur, on a été obligé de mettre beaucoup de P afin de le faire accélérer vite, et un tout petit peu de I afin de limiter les rebonds et oscillations autour de la consigne.

Valeurs normalisées :

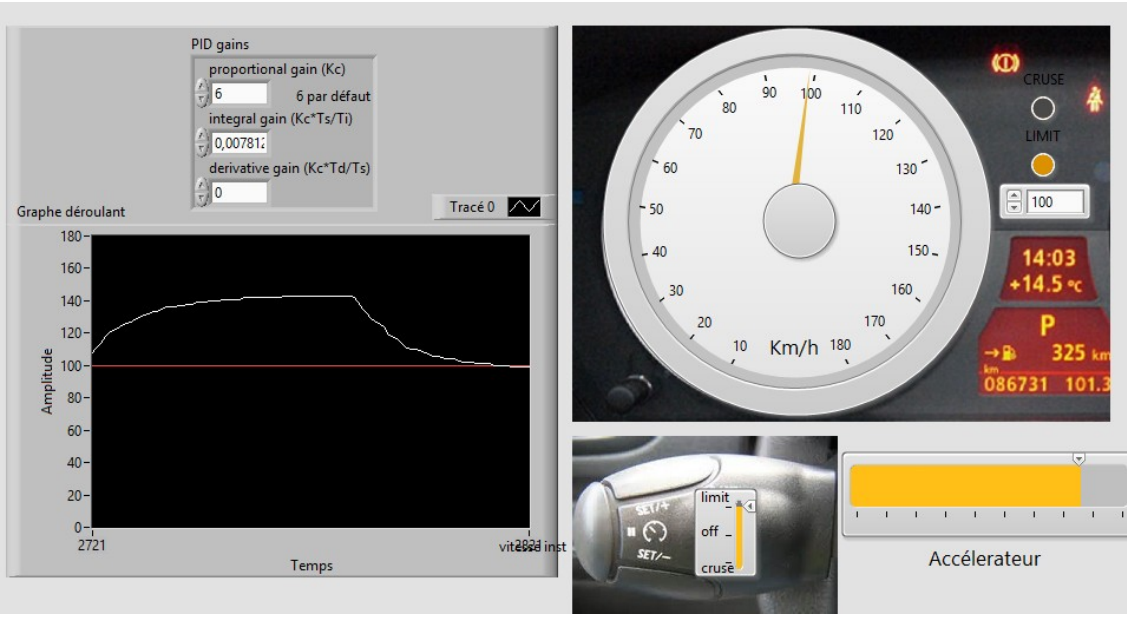
P = 6 I = 0,007 D = 0

Régulateur de vitesse

En intégrant le PID dans le code final on obtient ceci :

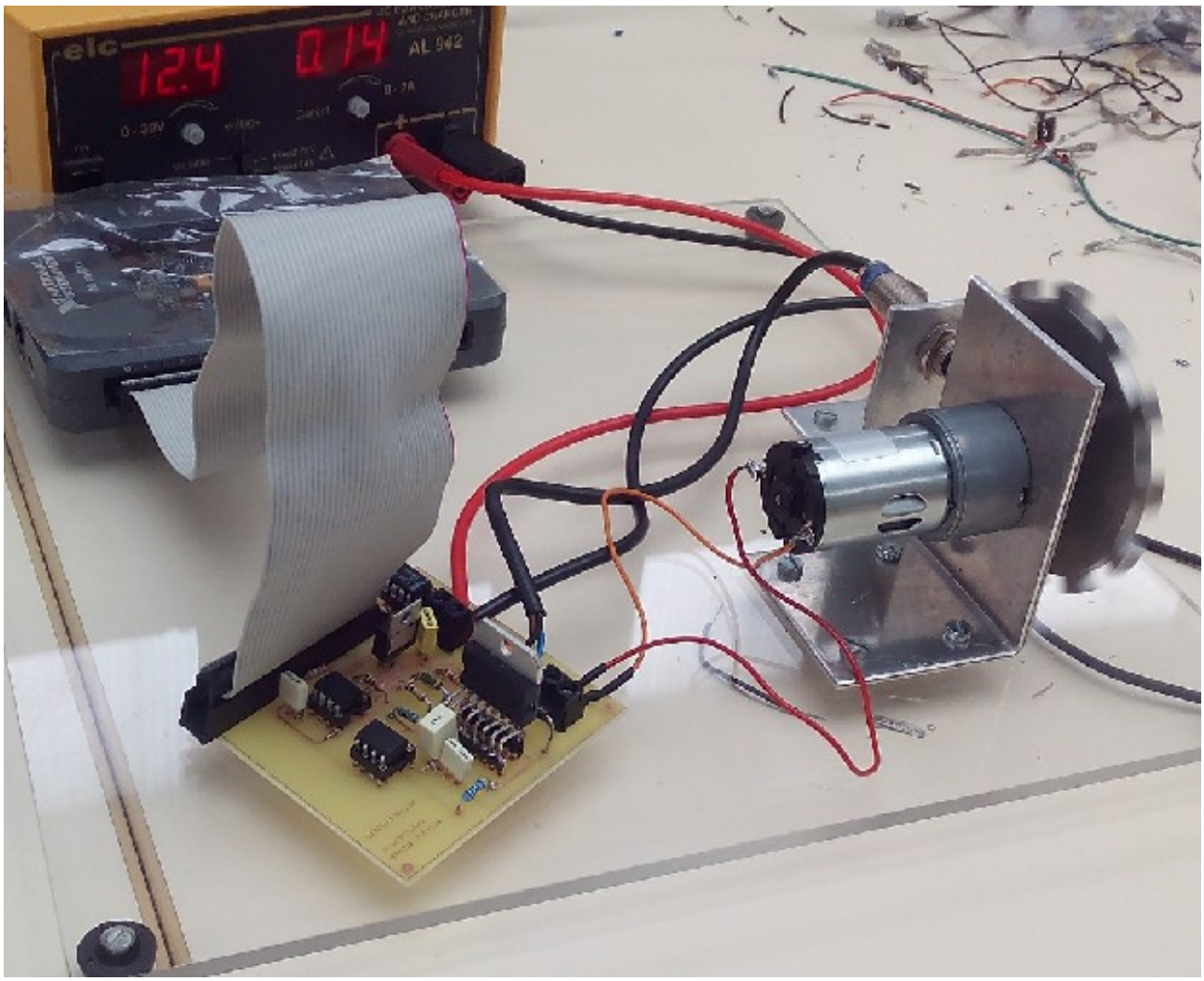


On retrouve la structure séquentielle qui permet de commander la PWM en fonction des modes. Les modes sont choisis dans la structure condition, ici c'est le mode régulateur. La lecture de vitesse s'effectue à droite.



Régulateur de vitesse

On à ensuite réaliser les plans d'une maquette de test, qui comporte le MYRIO, la carte électronique, le moteur avec sa roue dentée et le capteur. Les seuls liens à faire sont les branchements liés au FPGA (alimentation et USB) ainsi que l'alimentation de la carte électronique 12V.



Projet terminé, il suffit de compiler le VI dans le FPGA, de brancher le connecteur A sur la carte et l'alimentation 12V. Le reste se pilote à l'aide de LabVIEW, on retrouve la commande d'accélération, le choix du mode, sa consigne et une visualisation de la vitesse.

CONCLUSION

C'est après un semestre d'investissement personnel que le projet à finalement été mené à son terme. La maquette finale est réalisée, les fonctions sont validées et le soft est programmé. Il reste encore une petite chose à faire, lire la vitesse de rotation du moteur à partir du signal TTL comptant le nombre d'impulsion de la roue dentée.

Ce projet nous à permis d'aborder plusieurs côtés de l'électronique, en s'appuyant sur des solutions existant dans le monde automobile. Ainsi on à commencer par étudier les différentes fonctions électroniques, avant de les intégrer sur une carte. On à choisi des composants utilisés dans le monde de l'automobile tel le capteur inductif d'ABS. La programmation graphique est très intuitive et possède de nombreux avantages par rapport à la programmation en ligne.

Nous avons apprécier travailler sur ce projet qui fait appel à plusieurs compétences de la formation.

BIBLIOGRAPHIE

COMPOSANTS INTÉGRÉS :

L298 :	http://www.alldatasheet.com/datasheet-pdf/pdf/22437/STMICROELECTRONICS/L298.html
LM2907 :	http://www.alldatasheet.com/datasheet-pdf/pdf/8817/NSC/LM2907.html
7805 :	http://www.alldatasheet.com/datasheet-pdf/pdf/223217/ESTEK/7805A.html
LM311	http://www.alldatasheet.com/datasheet-pdf/pdf/8611/NSC/LM311.html
HCPL-2611	http://fr.rs-online.com/web/p/optocoupleurs/0590367/

COMPOSANTS AUTRES :

Myrio 1900	http://digital.ni.com/manuals.nsf/websearch/EE9557A29E63D0F986257BB8005700C5
Capteur	http://fr.rs-online.com/web/p/capteurs-de-proximite-inductifs/2811177/

LIENS UTILES :

Tutoriel LabVIEW FPGA	http://www.ni.com/tutorial/14532/fr/
Tutoriel LabVIEW FPGA en Anglais	http://en.youscribe.com/catalogue/tous/professional-resources/it-systems/labview-fpga-and-compactrio-getting-started-tutorial-537728