Sion, le 17.05.2017

RASPBERRY HUMANOIDE – R0B1

FRANCELET SAMY – GASPOZ FRÉDÉRIC

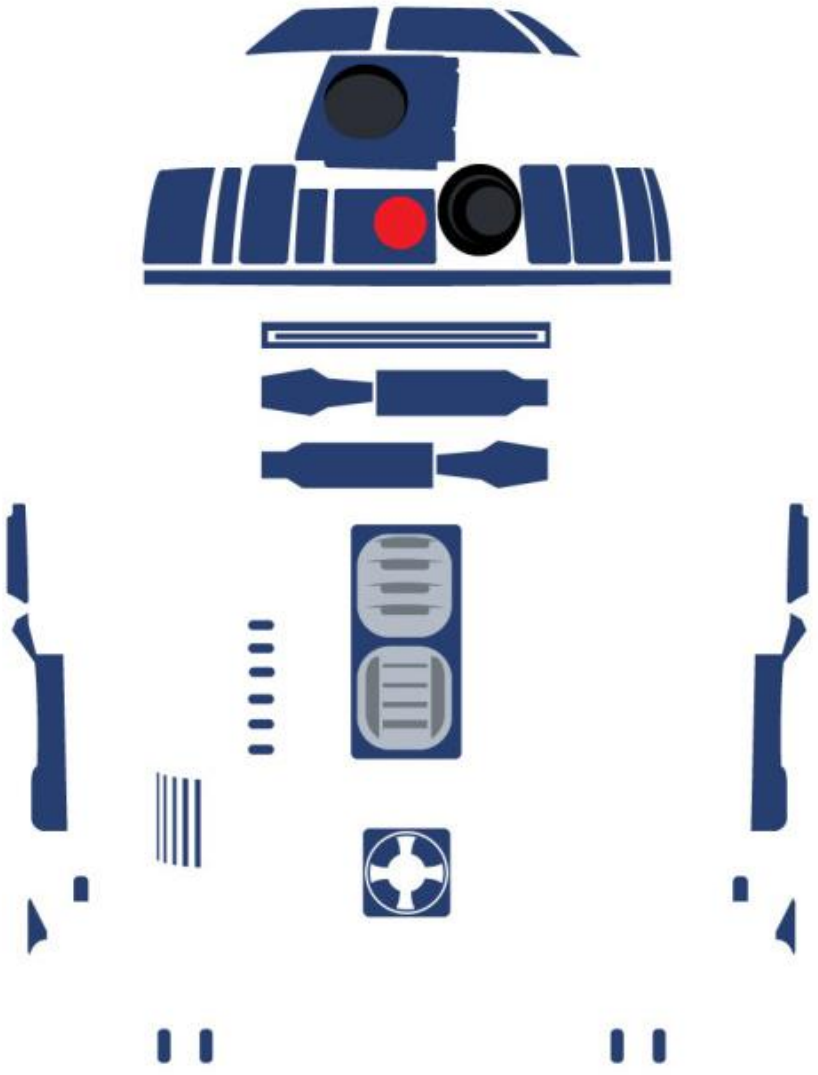


Table des matières

1 Introduction 3
2 Description 3
2.1 Cahier des charges 3
2.2 Schéma bloc 4
2.3 Planning 5
2.3.1 Planning prévu 5
2.3.2 Planning réalisé et problèmes rencontrés 5
3 Etude et développements 5
3.1 Description des variantes possibles 5
3.1.1 Alimentation du circuit 5
3.1.2 Communication avec les moteurs 6
3.1.3 Interface utilisateur 6
3.2 Explication de la solution choisie 7
3.2.1 Alimentation 7
3.2.2 Communication avec les moteurs 7
3.2.3 Interface utilisateur 7
4 Résultats obtenus 7
4.1 Mesures obtenues 7
4.2 Analyse des résultats 9
4.3 Tableau récapitulatif des résultats 9
5 Orientations futures et Conclusion 9
5.1 Orientations futures 9
5.2 Conclusion 9
6 Annexes 9

Résumé du projet

Ce projet est un torse avec deux bras robotiques à servomoteurs, contrôlés avec un Raspberry Pi. Les mouvements des bras seront programmables simplement en les bougeant, le Raspberry se chargera d'enregistrer les valeurs, puis il renverra les valeurs lors de la lecture des mouvements programmés.

1 Introduction

Tout le monde aimerait bien avoir un robot qui effectue les actions du quotidien à notre place, le problème étant que les robots commercialisés ne savent faire que quelques mouvements enregistrés d'usine, ou il faut le programmer et donc avoir certaines compétences en programmation.
Mon projet consiste à faire un robot, dont les mouvements sont programmables simplement, juste en bougeant les bras. L'interface sera simple pour éviter de perdre l'utilisateur dans des milliers de paramètres, et sera sur un écran tactile. L'écran servira aussi de visage au robot lorsqu'il répétera les mouvements enregistrés, un ou plusieurs visages seront sélectionnables.
Ce rapport contiendra le cahier des charges du projet, des explications simple via des schémas blocs, ainsi que le développement avec toutes les solutions et explications des choix utilisés, des mesures prouvant le bon fonctionnement du système, les orientations futures ainsi que les améliorations imaginables dans le futur, et finalement les annexes.

2 Description

2.1 Cahier des charges

- Le changement du visage vers l'interface se fera via une pression sur l'écran, et le changement de l'interface vers le visage se fera en sélectionnant un programme
- Les moteurs utilisés sont les AX-12A pour leur mode de communication utile à la récupération des valeurs de position pour la programmation.
- Plusieurs programmes pourront être enregistrés dans la mémoire du Raspberry, ils seront tous enregistrer dans des fichiers, ce qui permettra de ne pas perdre les données à l'extinction.
- Les bras seront facilement programmables, simplement en les bougeant.
- Si possible, au démarrage du Raspberry, le programme sera lancé automatiquement en plein écran. Le programme pourra être fermé avec un bouton tactile.
- Un mode debug sera disponible pour le développement, qui indiquera à l'utilisateur les positions enregistrées et les positions envoyées. Les données seront ensuite envoyées dans des fichiers xlsx ou csv.
- Le code sera en Python.
- L'interface graphique sera faite sous Kivy.
- Le robot sera alimenté par le secteur, alimentation 12V.
- Il y aura un régulateur de tension pour créer du 5V pour le Raspberry et l'écran.
- Un ou plusieurs visages animés seront enregistrer dans le Raspberry, et on pourra choisir le visage pour chaque programme (optionnel)
- La conception mécanique du robot sera réalisée en 3D sur ordinateur.

2.2 Schéma bloc

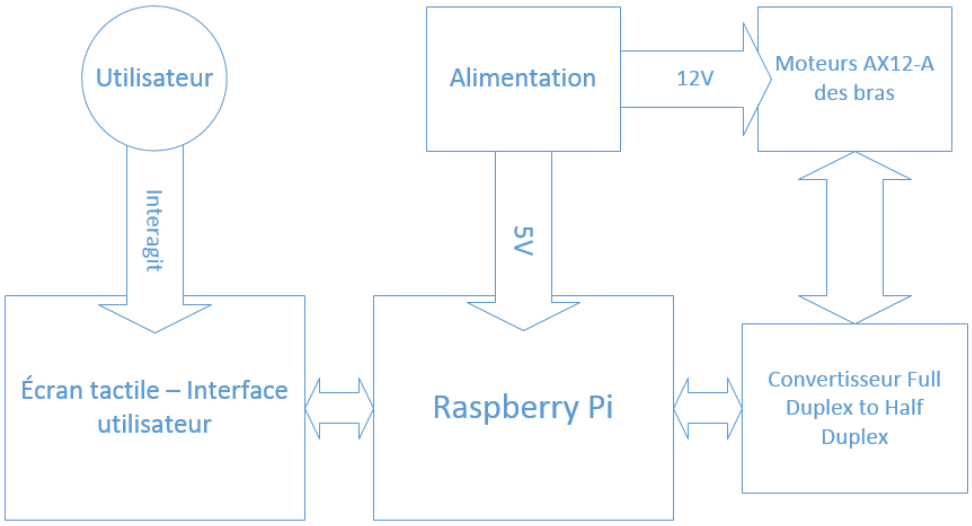


Figure 1 - Schéma bloc hardware

L'utilisateur contrôlera le robot via l'écran tactile, qui fait office d'interface utilisateur et de visage. Le Raspberry, en fonction des actions de l'utilisateur, enverra ou recevra des informations des moteurs via le port série.

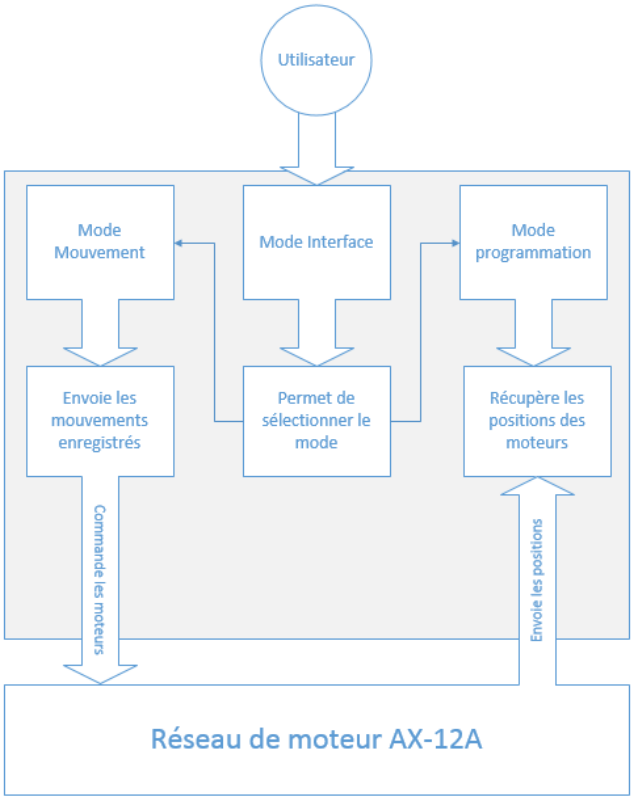


Figure 2 - Architecture software basique

Avec l'interface l'utilisateur aura accès à 3 modes, le mode interface étant un hub pour passer à un autre mode. Le mode programmation sert à lancer la programmation des bras, et stockera tous les mouvements effectués sur les moteurs. Le mode mouvement fera reproduire les mouvements enregistrés

2.3 Planning

2.3.1 Planning prévu

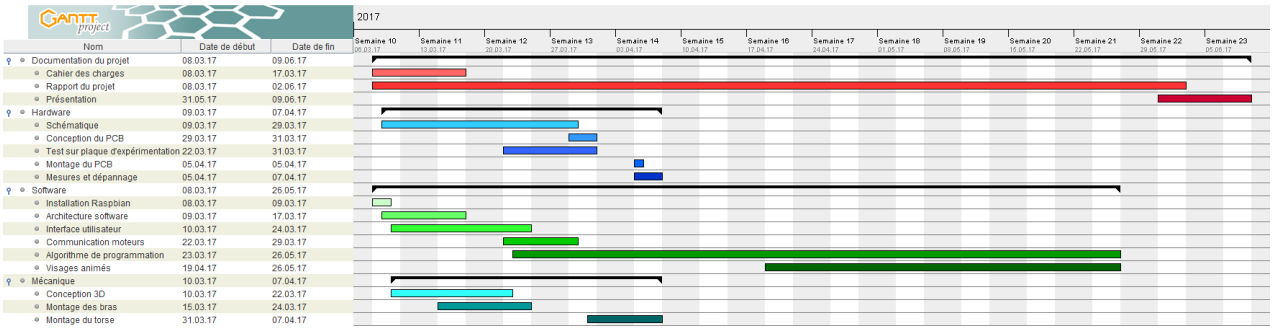


Figure 3 - Planning gantt prévu

2.3.2 Planning réalisé et problèmes rencontrés

3 Etude et développements

3.1 Description des variantes possibles

3.1.1 Alimentation du circuit

Pour l'alimentation du circuit, il faut pouvoir fournir assez de courant pour les moteurs en 12V, le Raspberry Pi et l'écran tactile en 5V.

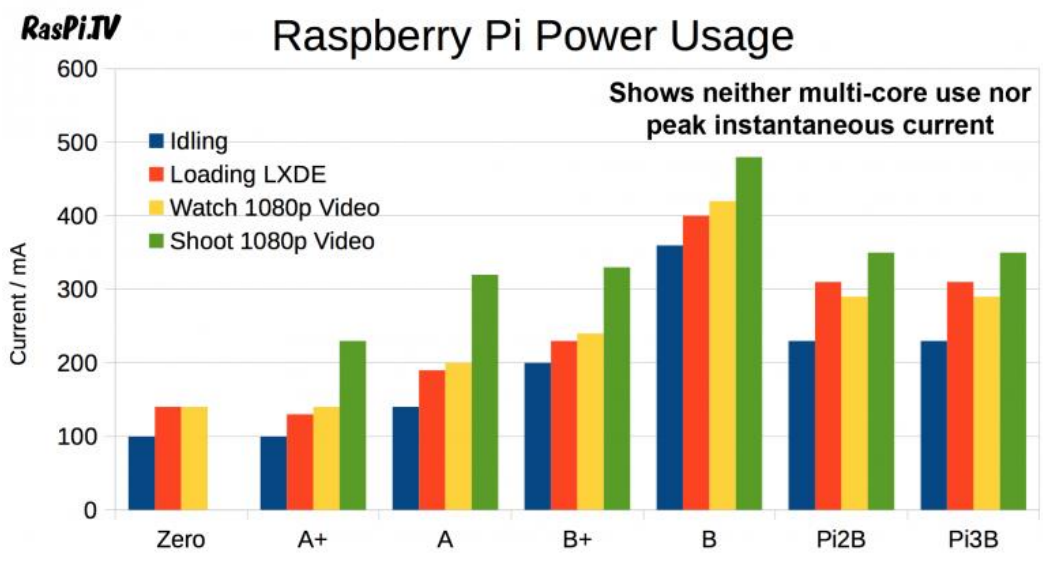


Figure 4 - Consommation des Raspberry Pi

Pour prendre de la marge, et parce que le graphe ne prend pas en compte le multi-core, on peut considérer que le Raspberry Pi 3B consomme 600mA. L'écran consomme plus de 500mA selon le site officiel, qu'on arrondit à 700mA pour avoir de la marge. Ce qui nous fait un total de 1,3A

Il nous faut donc une alimentation capable de nous fournir 5V et au moins 1,3A.

3.1.1.1 LM7805

Le LM7805 est simple d'utilisation et peu cher, mais son grand défaut est son rendement faible. Il va beaucoup trop chauffer et ne pourra jamais driver 1,3A.

3.1.1.2 Alimentation à découpage

Une alimentation à découpage serait très efficace, mais cela couterait trop cher et prendrait trop de temps à mettre en place

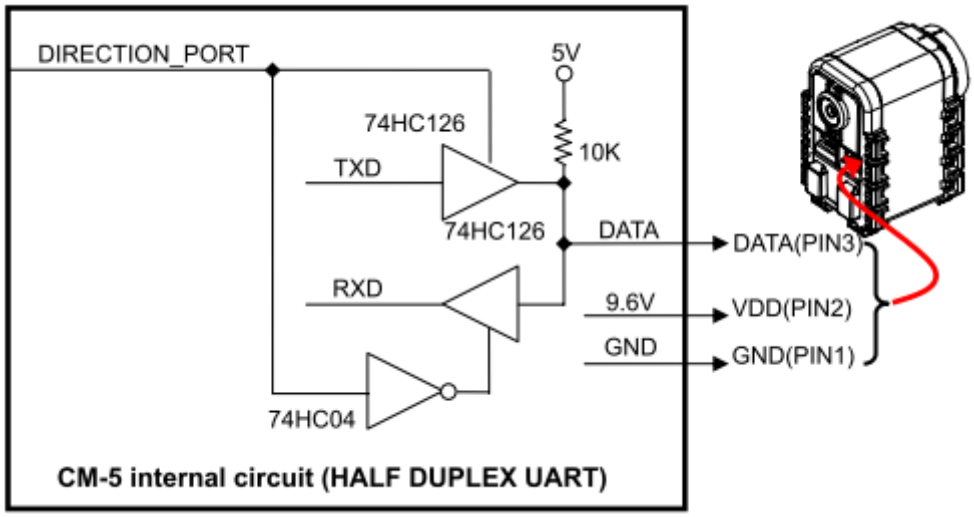
3.1.1.3 Module convertisseur DC/DC

Un module pouvant convertir du 12V en 5V (comme les Traco) serait parfait, car simple d'emploi, ils ont un bon rendement et ne sont pas excessivement chers.

3.1.2 Communication avec les moteurs

Les moteurs AX-12A utilisent un protocole UART pour communiquer, mais en half-duplex. Le Raspberry peut communique en UART, mais en full duplex, il faut donc convertir le signal.

Pour cela je vais utiliser le montage proposé dans le datasheet du servomoteur.



3.1.3 Interface utilisateur

Le code étant en Python, il faut une librairie Python permettant de faire des interfaces tactiles.

3.1.3.1 PyQt

PyQt est la version Python du framework Qt. Qt est très efficace, bien documenté, Open Source et très stable, mais il est plutôt fait pour des interfaces bureau classiques.

3.1.3.2 Kivy

Kivy est un framework simple pour Python, étant presque spécialement fait pour le tactile. Sous linux il ne nécessite pas d'avoir un serveur X de lancer, il peut donc se lancer depuis le terminal.

3.2 Explication de la solution choisie

3.2.1 Alimentation

Pour l'alimentation nous avons choisis de prendre un module convertisseur DC/DC Recom, car il a les même pates et branchement qu'un simple LM7805. Le module choisis est le Recom R78B5, 0-1,5.

3.2.2 Communication avec les moteurs

Pour la communication avec les moteurs, nous avons gardé le modèle du datasheet, mais remplacer le 74HC126 par un 74HC241 car il possède deux direction port, un actif bas, l'autre actif haut, nous permettant de ne pas prendre d'inverseur.

3.2.3 Interface utilisateur

Kivy sera utilisé pour l'interface utilisateur, pour sa simplicité, sa documentation bien fournie, et pour faire des interfaces parfaite pour le tactile.

4 Résultats obtenus

4.1 Mesures obtenues

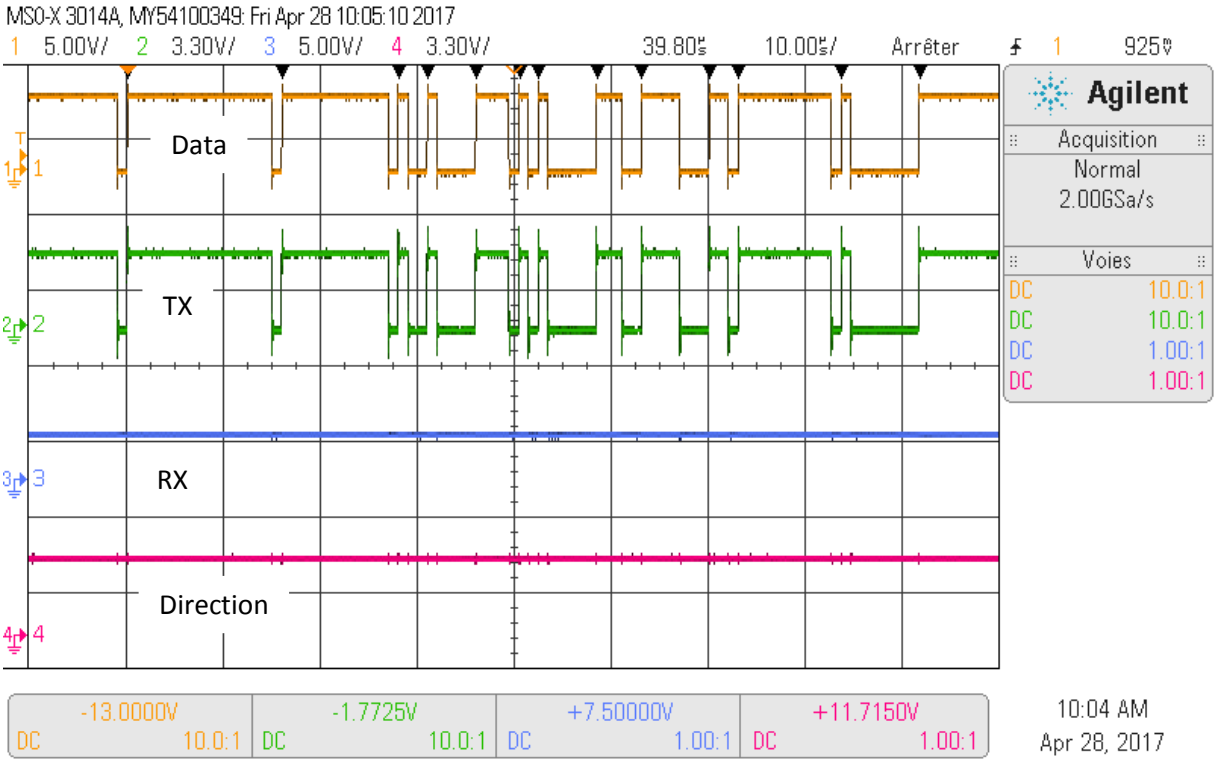


Figure 5 - Test d'envoi d'une trame dans le 74HC241

La pin de direction est à 1, le TX arrive bien à envoyer des données aux moteurs.

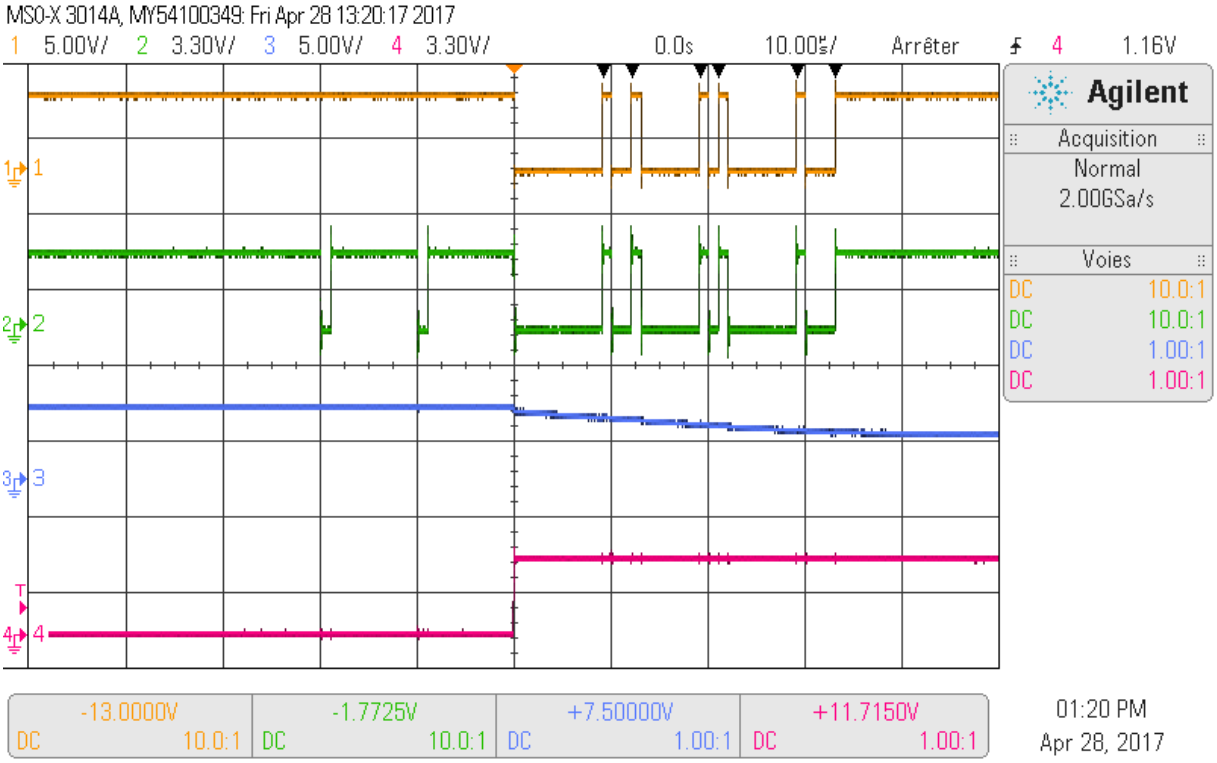


Figure 6 - Test du code driver de moteur

Pour une raison obscure, la pin de direction monte à 1 bien après qu'on lui ait demandé de monter dans le code.

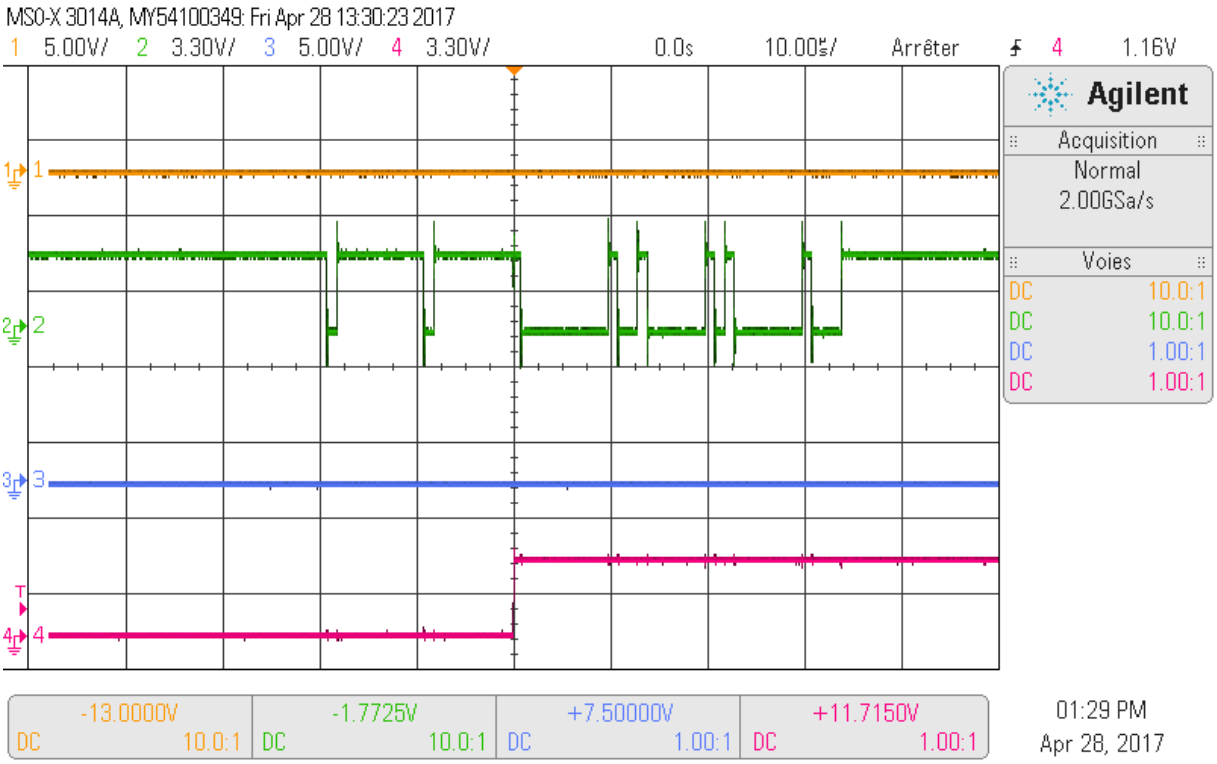


Figure 7 - Test du code driver sans la carte 74HC241

On remarque bien que le problème est software, car même sans le chip le problème subsiste.

4.2 Analyse des résultats

4.3 Tableau récapitulatif des résultats

5 Orientations futures et Conclusion

5.1 Orientations futures

5.2 Conclusion

6 Annexes