Usage du nano-ordinateur Raspberry Pi et Matlab pour la régulation d'une alimentation à découpage

Manuel Avila, Christophe Charrière, Pascal Vrignat, Florent Duculty, Stéphane Begot, Jean-Christophe Bardet

▶ **To cite this version:**

Manuel Avila, Christophe Charrière, Pascal Vrignat, Florent Duculty, Stéphane Begot, et al.. Usage du nano-ordinateur Raspberry Pi et Matlab pour la régulation d'une alimentation à découpage. Journal sur l'enseignement des sciences et technologies de l'information et des systèmes, 2023, 22, pp.1008. 10.1051/j3ea/20231008 . hal-04273680

HAL Id: hal-04273680
https://hal.science/hal-04273680

Submitted on 6 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Usage du nano-ordinateur Raspberry Pi et Matlab pour la régulation d'une alimentation à découpage

Manuel Avila, Christophe Charrière, Pascal Vrignat, Florent Duculty, Stéphane Begot, Jean-Christophe Bardet
Manuel.Avila@univ-orleans.fr
Adresse : *Université d'Orléans, Laboratoire PRISME, EA 4229,*
IUT de l'Indre, Département GEII, 2 Av. François Mitterrand, 36000 Châteauroux-France

RESUME : Les nano-ordinateurs de type Raspberry Pi ont pris une place importante dans le domaine de l'EEA, que ce soit pour des Interfaces Homme-Machine (IHM), pour le contrôle-commande de systèmes (automatisation simple, fonction pour des services de communication, superviseur domotique…). Les outils disponibles sur ce type de plateforme ainsi que les systèmes d'exploitation sont nombreux. Les langages de programmation disponibles pour l'implémentation des programmes sont également nombreux. Dans ce contexte, Matlab permet notamment ces implémentations. Ce dernier est particulièrement adapté pour la mise en œuvre de systèmes asservis. Cependant, l'une des difficultés avec une carte Raspberry Pi est l'absence d'entrées et/ou sorties analogiques. Dans ce travail, nous proposons une solution de gestion d'entrées/sorties analogiques pour une fréquence d'échantillonnage de l'ordre de quelques kHz.

Mots clés : Matlab, Raspberry Pi (RPi), Asservissement.

1 INTRODUCTION

Dans le cadre de séances de travaux pratiques en automatique, nous utilisons une maquette consacrée à la régulation d'une alimentation à découpage de type Forward. Une carte DSpace intégrée dans un ordinateur de bureau (format PCI), est utilisée pour piloter la commande de l'alimentation. Celle-ci est programmée à l'aide de Matlab et de la chaine de développement associée. Cependant, ce système est malheureusement devenu obsolète sur différents points : pilote et chaine de développement uniquement compatible avec Windows XP, le format PCI n'est plus (ou rarement) disponible sur les ordinateurs récents.

Afin de mettre à jour cette maquette en conservant la stratégie de développement Matlab pour programmer une cible de type calculateur, nous aurions pu décider de commander de nouveaux produits du constructeur DSpace. En dehors du coût important de ce type de solution (quelques dizaines de k€), nous avons souhaité développer une solution sur une cible de type Raspberry Pi (RPi). Les toolbox Rpi (Matlab et Simulink) sont effectivement disponibles depuis plusieurs années dans l'environnement Matlab. Dans [2], les auteurs proposent une prise en main progressive de la carte RPi avec Matab, sur plusieurs séances de TP, depuis l'allumage de LEDs au traitement d'images. Des solutions orientées Open Source existent également pour la programmation de la RPi [3]. Elles sont relativement stables, mais ne proposent pas de fonctions simples de gestion d'entrées ou sorties analogiques puisque celles-ci ne sont pas natives sur la carte. Il est possible d'utiliser les entrées/sorties audio (via le pilote ALSA), comme les auteurs dans [1], pour mettre en œuvre du traitement du signal mais la nature des signaux audio limite l'usage de ces entrées. Dans ces conditions, il faut donc ajouter des circuits annexes à l'aide des bus SPI ou I^2C. Nous avons fait le choix d'utiliser des composants compatibles SPI car après quelques essais, ils se sont avérés plus rapides que les versions I^2C.

Dans la partie 2, nous rappelons le principe de mise en œuvre d'une carte RPi avec Matlab. La partie 3, présente la maquette de régulation d'alimentation à découpage utilisée ainsi que l'interface créée pour la RPi. Dans la partie 4, nous montrons la mise en œuvre du régulateur, ainsi que les limites de cette solution. Après la conclusion, nous proposons quelques pistes d'amélioration.

2 MATLAB POUR PROGRAMMER UNE RASPBERRY PI

2.1 Mise en route de la Raspberry Pi

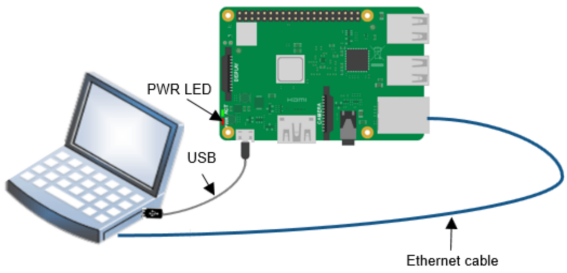


Fig 1 : Développement Matlab vers Raspberry Pi

L'utilisation d'une carte RPi en tant que « système embarqué » suppose, la plupart du temps, que la carte sera utilisée sans clavier et souris et éventuellement sans écran. La prise en main du système doit donc se faire à distance à travers une liaison sécurisée (SSH[1]) (Fig 1).

[1] Secure Shell

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Article publié par EDP Sciences et disponible sur le site https://www.j3ea.org ou https://doi.org/10.1051/j3ea/20231008

Pour démarrer le système, on pourra suivre l'un des nombreux tutoriels disponibles sur Internet (par exemple [4]). Il faut au moins disposer de l'adresse IP de la carte et activer la connexion via SSH :

- Ajouter IP=XX.XX.XX.XX, l'adresse IP choisie dans le fichier /boot/cmdline.txt pour un adressage statique,
- Ajouter le fichier « ssh », fichier vide dans le répertoire /boot/ pour autoriser les connexions SSH.

Le répertoire « boot » est la partition de la carte mémoire SD lisible depuis un ordinateur.
Après le démarrage de la RPi, on peut donc se connecter à partir d'un terminal (Mac ou Linux) (Fig 2) ou avec Putty (Windows), pour ouvrir une session « bash » qui va permettre de configurer la carte.

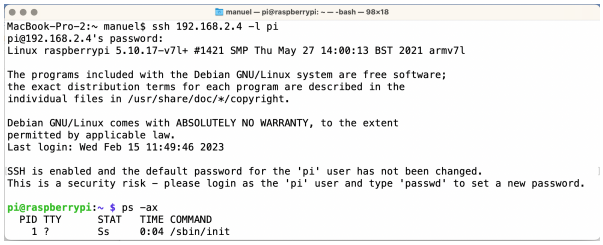


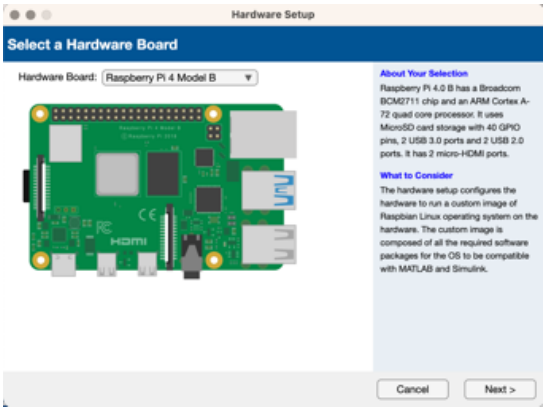
Fig 2 : SSH via Terminal sur Mac OS.

2.2 Configuration des Toolbox Matlab

Pour pouvoir programmer la carte RPi depuis Matlab, il faudra installer les Toolbox spécifiques (Fig 3) : les deux premières permettent d'utiliser Simulink ou Matlab pour programmer la carte RPi.

Fig 3 : Les Toolbox Matlab et Simulink

Après l'installation de ces Toolbox, Matlab propose de configurer la carte RPi pour lui permettre de recevoir les programmes Matlab à partir du réseau.



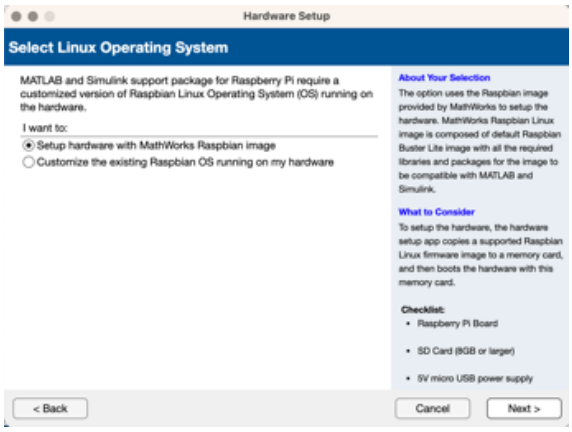


Fig 4 : Choix du mode d'installation

Après avoir choisi le type de carte à utiliser, Matlab propose de télécharger une image « Clé en main » du système ou de configurer un système existant (Fig 4).
Pour débuter, la première solution est la plus rapide. Les experts qui souhaitent utiliser la carte RPi avec des fonctionnalités spécifiques ou qui souhaitent conserver la maitrise des packages installés opteront pour le second choix. Cependant, la procédure d'installation des packages peut échouer en cours de route. Il faut alors résoudre les dysfonctionnements au cas par cas.

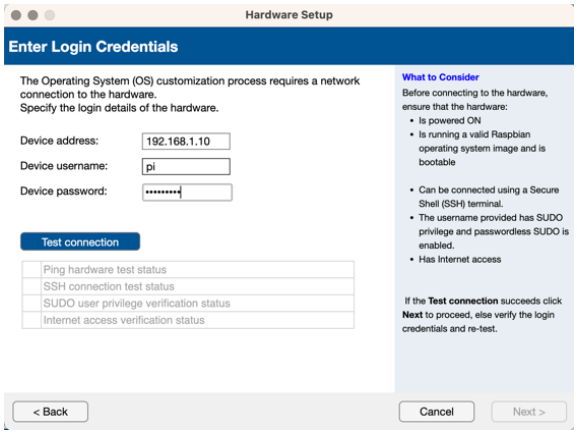


Fig 5 : Personnalisation d'un OS existant

Pour cette deuxième solution (Fig 5), Matlab ouvre une session SSH sur la carte RPi et lance un script qui enchaîne l'installation des différents packages nécessaires. Il est indispensable que la carte RPi soit connectée à Internet. Ces conditions peuvent poser des problèmes lorsqu'il faut franchir un proxy qui demande une authentification sur une page Internet. C'est souvent le cas dans nos établissements de formations.

2.3 Premier programme : Blink

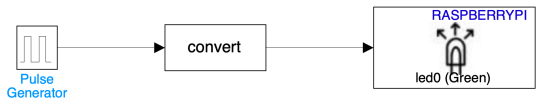


Fig 6 : Programme Blink

A l'aide de Simulink, la réalisation d'un programme permettant de faire clignoter la Led verte de la RPi se résume à un générateur connecté à la Led par le biais d'un bloc de mise en forme (Fig 6).

Fig 7 : Déployer sur la cible

Pour déployer le programme sur la cible, il suffit d'appuyer sur le bouton « Build Deploy & Start » (Fig 7). Ce qui déclenche les actions suivantes :

1. Le schéma Simulink est alors traduit en langage C.
2. L'ensemble des fichiers du projet est transféré sur la carte RPi par la liaison SSH.
3. La compilation du projet se fait à l'aide de GCC sur la carte RPi.
4. Une compilation avec succès entraine l'activation du programme.

3 MISE EN ŒUVRE D'UN ASSERVISSEMENT D'ALIMENTATION A DECOUPAGE

3.1 La maquette « Forward »

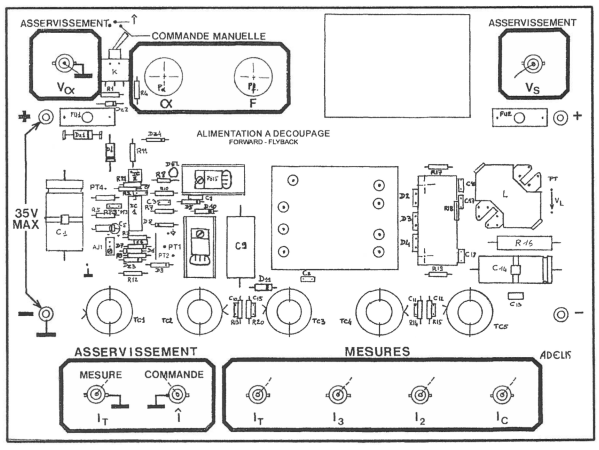


Fig 8 : Maquette Forward

La maquette « Forward » que nous utilisons (Fig 8), dispose de plusieurs modes de fonctionnement : automatique (asservi), manuel (réglage F et α) et pilotée par un système externe. C'est ce dernier mode que nous utilisons pour identifier le système en mode fréquentiel pour ensuite mettre en œuvre l'asservissement à l'aide d'un calculateur programmé à l'aide de Matlab.

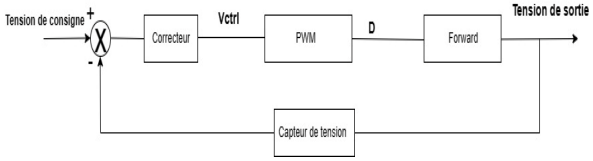


Fig 9 : Schéma bloc de l'alimentation Forward

Le correcteur était jusqu'à présent implémenté sur une carte DSpace intégrée dans un ordinateur de type PC (Fig 8). Cette solution a été mise en service il y a plusieurs années. Cet ordinateur ainsi que le système d'exploitation (Windows XP) sont devenus obsolètes. Notre service informatique ne souhaite plus que nous hébergions des systèmes d'exploitation trop anciens présentant trop de failles de sécurité. Le standard PCI n'est que très rarement disponible dans les ordinateurs et les pilotes de la carte DSpace ne sont plus disponibles depuis plusieurs mises à jour de Windows.
Toutes ces considérations nous ont conduits à envisager une mise à jour du système permettant de réaliser le régulateur de l'alimentation à découpage, tout en conservant l'architecture : Matlab permettant de programmer une cible de type calculateur.

3.2 La gestion des entrées sorties analogiques.

La RPi est un nano-ordinateur avec des capacités de calcul suffisantes pour la mise en œuvre d'un « simple » correcteur PI avec des contraintes temps réel peu exigeantes. Cependant, ce calculateur ne dispose pas de façon native d'entrées ou de sorties analogiques.
Il existe plusieurs kits de convertisseurs ADC ou DAC, mais très peu directement adaptés à la carte RPi.
Pour la phase de prototypage de notre système, nous avons utilisé la carte Gertboard.

3.2.1 La carte Gertboard

La carte Gertboard (Fig 10) est une carte d'extension pour RPi. Elle permet de réaliser des prototypes nécessitant des entrées et des sorties.

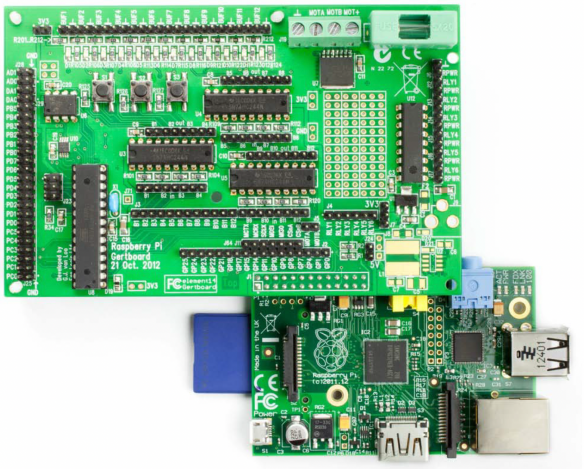


Fig 10 : Vue de la carte Gertboard

Elle dispose de :

- 12 entrées/sorties bufferisées,
- 3 boutons poussoirs,
- 6 drivers collecteur ouvert (50V, 0.5A),
- 1 Contrôleur moteur 18V, 2A,
- 1 Microcontrôleur ATmega,
- 2 voies DAC 8 bits (MCP3002),
- 2 voies ADC 10 bits (MCP4802).

Cette carte de prototypage nous a permis de vérifier la faisabilité de notre système.

3.2.2 Gestion ADC et DAC sur Matlab

Un driver pour ADC/DAC est disponible au téléchargement (Fig 11). Cependant, il n'est prévu que pour ces 2 composants : ADS1115 et MCP4725. Mais depuis plusieurs versions de Matlab, ces projets ne peuvent être mis en œuvre facilement. En effet, ce driver utilise la librairie « WiringPi » qui nécessite de configurer l'option '-lwiringpi' à la compilation. Ce qui nécessite la commande « xmakefilesetup » qui n'est plus disponible sous Matlab.

Fig 11 : Driver ADC DAC PWM

Pour remédier à ce problème de librairie, nous proposons des modules qui utilisent les blocs SPI disponibles dans la Toolbox Simulink. Ils sont précédés ou suivis par une fonction de composition de trame et/ou d'une fonction de décodage de la réponse. Ces fonctions seront donc aisément adaptables en fonction du choix des convertisseurs.

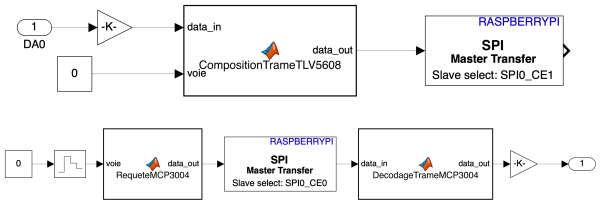


Fig 12 : Bloc de gestion des entrées et sorties analogiques

Afin de régler la fréquence d'échantillonnage, un bloqueur d'ordre zéro (Fig 12) est inséré dans la chaine de transmission SPI.

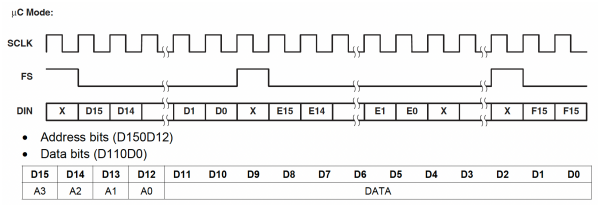


Fig 13 : Trame SPI à envoyer

Le bloc « CompositionTrame » s'inspire du chronogramme (Fig 13) pour définir la trame à envoyer sur le convertisseur DAC. La fonction est détaillée ci-après.

- CompositionTrameTLV5608

```
function data_out = CompositionTrameTLV5608( data_in, voie )
% CompositionTrameTLV5608 convertisseur 10 bits
% constitution de la trame à envoyer au TLV 5608
% data_in : La valeur à envoyer
% voie : le numéro de la sortie
% D15 D14 D13 D12 D11 D10 D9 D8    D7 D6 D5 D4 D3 D2 D1 D0
% A3  A2  A1  A0 ----     DATA SUR 10 BITS    ---    X  X
data_out = [uint8(0) uint8(0)]; % sortie de la fonction
data_out(1) = bitand ( bitsrl(uint16(data_in) , 6), 15) +
bitsll( bitand (uint16(voie) , 15) , 4 );
data_out(2) = bitsll( bitand ( uint16(data_in), 63), 2);
end
```

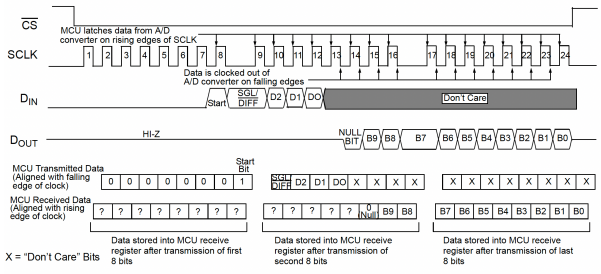


Fig 14 : Trame SPI lecture données

La lecture du convertisseur analogique numérique MCP3004 s'effectue de la façon suivante. Une trame de demande de lecture est envoyée. Elle déclenche le début de conversion et la valeur numérique est renvoyée sur la fin de la trame (Fig 14). Pour effectuer cette opération, deux fonctions sont utilisées.

- RequeteMCP3004

```
function data_out  = RequeteMCP3004( voie )
% mode standard
% 3 octets data_out = [uint8(0) uint8(0) uint8(0) ];
% Octet1 0        0  0  0  0 0 0 start
% Octet2 SGL/DIFF D2 D1 D0 x x x   x
% Octet3 x x x x x x x x
data_out = [uint8(0) uint8(0) uint8(0)];
if voie == 0
    data_out(1) = bin2dec ( '00000001');   % Start
    data_out(2) = bin2dec ( '00000000');   % Single et voie 0
elseif voie == 1
    data_out(1) = bin2dec ( '00000001');   % Start
    data_out(2) = bin2dec ( '00010000');   % Single et voie 1
elseif voie == 2
    data_out(1) = bin2dec ( '00000001');   % Start
    data_out(2) = bin2dec ( '00100000');   % Single et voie 2
else
    data_out(1) = bin2dec ( '00000001');   % Start
    data_out(2) = bin2dec ( '00110000');   % Single et voie 3
end
data_out(3) = 0;
end
```

- DecodageTrameMCP3004

```
function data_out  = DecodageTrameMCP3004( data_in )
% recomposition d'un mot de 16 bits à partir de 3 octets
% data-in(1) inutilisé
% data-in(2)  x  x  x  x  x  0 B9 B8
% data-in(3) B7 B6 B5 B4 B3 B2 B1 B0
data_out = double(0);
data_out = double(  bitsll( uint16 ( bitand ( data_in(2), 3,
"uint8")), 8 )) + double( data_in(3) );
end
```

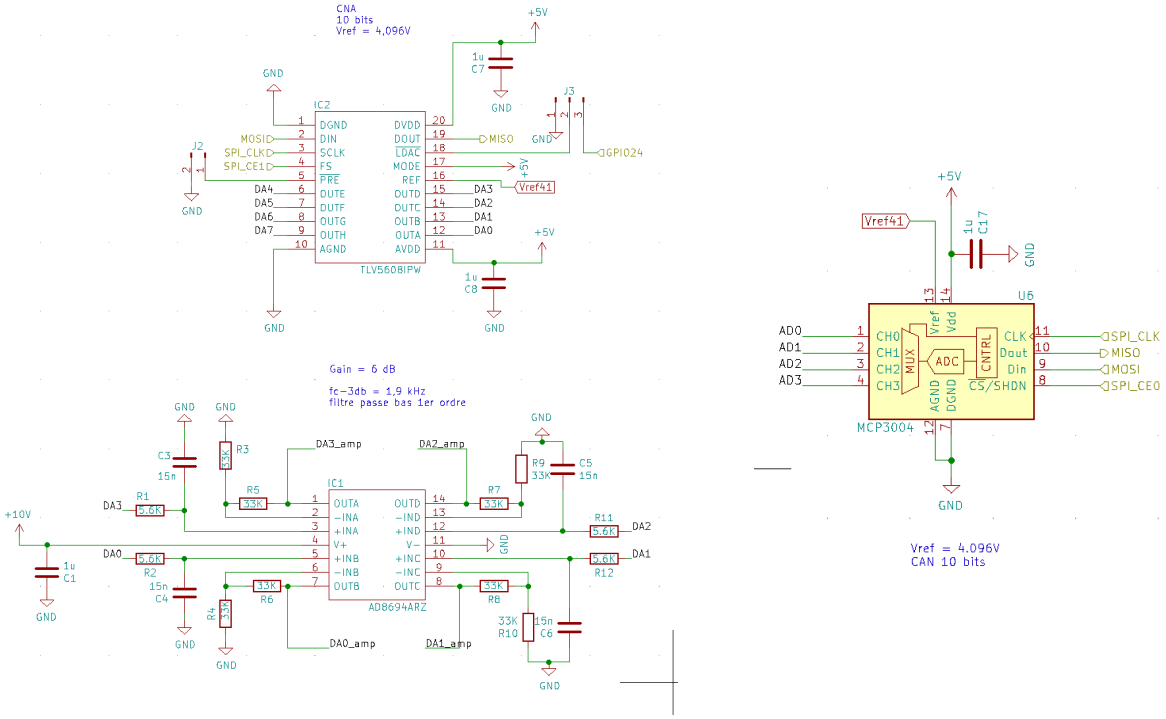


Fig 16 : Extrait du schéma de la carte d'interface

4 REGULATION DE L'ALIMENTATION A DECOUPAGE

4.1 La carte interface pour Raspberry Pi

Fig 15 : PCB de la carte d'interface

A partir des essais effectués sur la carte de prototypage Gertboard, nous avons conçu une carte « fille » qui vient se connecter sur le connecteur GPIO de la carte RPi (Fig 15). Sur cette carte « fille », nous avons prévu des connecteurs BNC et Embase 4 mm pour faciliter le câblage pendant la séance de travaux pratiques.
Nous avons opté pour des convertisseurs avec au moins 4 voies : MCP3004 pour le convertisseur A/D (10 bits) et TLV5608 pour le convertisseur D/A (10 bits).
Un amplificateur est utilisé pour chaque sortie ainsi qu'un pont diviseur pour chaque entrée pour disposer d'une plage de tension de 0-10V en entrée et 0-8V en sortie (Fig 16).

4.2 Régulateur PI sur Raspberry Pi

Le schéma (Fig 17) montre le régulateur PI qui était mis en œuvre sur la carte DSpace. Les seules parties qui ont changé sont les blocs de gestion d'entrées/sorties analogiques. Nous avons utilisé les blocs définis au paragraphe 3.2.2.

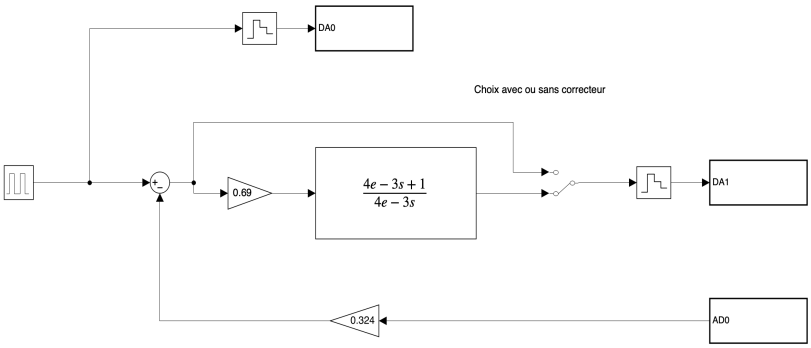


Fig 17 : Régulateur PI pour l'alimentation à découpage

La duplication de ces blocs (DA0, DA1) ne pose pas de difficulté. Les résultats obtenus sont similaires avec ceux obtenus avec l'ancienne maquette.
La Figure 18 montre la tension de sortie de l'alimentation en boucle fermée (BF), pour un échelon de consigne, sans correcteur (choix avec switch Fig 17).

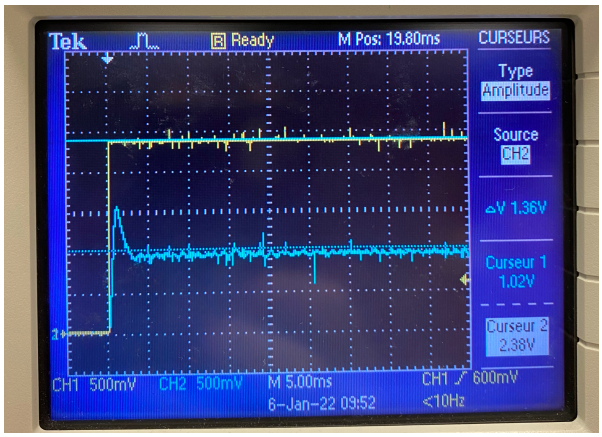


Fig 18 : Réponse du système en BF sans correcteur

La Figure 19 montre la réponse pour la tension de sortie de l'alimentation, pour le même échelon de consigne, en boucle fermée (BF) avec un correcteur PI. Cela montre l'action du correcteur à l'aide du système implémenté sur la carte RPi.

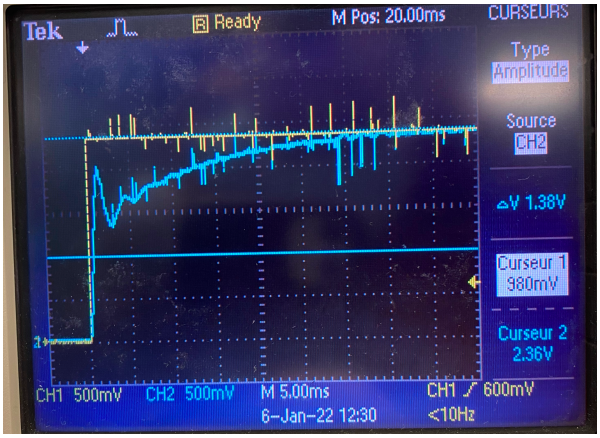


Fig 19 : Réponse du système en BF avec correcteur PI

4.3 Limites fréquentielles

La fréquence d'échantillonnage, qui cadence le traitement des données, peut être ajustée au niveau du bloqueur d'ordre zéro (Fig. 20).

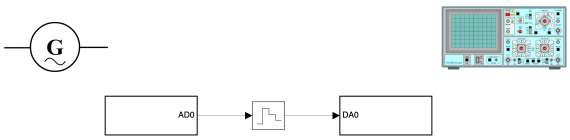
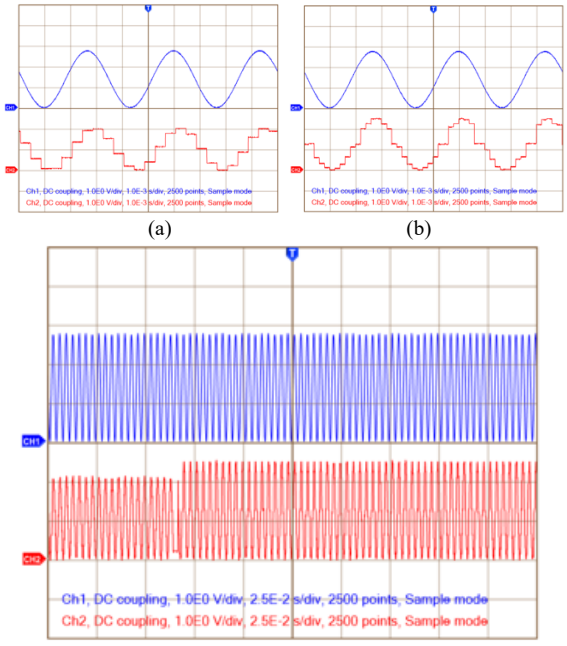


Fig 20 : Test de recopie entrée vers sortie

La valeur limite de période d'échantillonnage, pour laquelle le système fonctionne, est Te=0,4 ms (Fig 21 (a)). Si l'on réduit cette valeur, il y a une perte de points d'échantillonnage comme le montre la figure 21 (b) et (c). Le scheduler du système d'exploitation doit limiter cette période d'échantillonnage. La figure 21 (b) est un zoom sur la figure 21 (c) dans laquelle on visualise les périodes de bon et mauvais fonctionnement.



(a) (b)

(c)

Fig 21 : Oscillogramme pour différentes périodes d'échantillonnage

5 CONCLUSION ET PERSPECTIVES

Dans cet article, nous avons souhaité partager notre expérience sur la mise en œuvre d'un asservissement fondé sur un nano-calculateur de type Raspberry Pi. Ces calculateurs offrent des capacités de calcul très intéressantes, notamment avec la possibilité de les programmer depuis Matlab. Cependant, l'absence de solutions simples d'interface d'entrées ou sorties analogiques, nous a conduits à développer une carte fille pour Raspberry Pi. Nous avons montré comment intégrer, de façon « simple », ces entrées ou sorties analogiques sur un montage Simulink à partir des blocs SPI disponibles sur la Toolbox Raspberry Pi. Cette architecture pourra être adaptée au choix du convertisseur ADC ou DAC.
Nous avons montré les limites de la solution proposée, notamment au niveau de la période d'échantillonnage (limitée à 0,4 ms ou 2,5 KHz). Cependant, des améliorations sont envisageables. Une étude plus approfondie de la gestion des tâches sur Raspberry Pi OS pourrait permettre de réduire la période d'échantillonnage.
Une autre possibilité offerte pour la partie cadencement des tâches est la possibilité d'utiliser le composant audio ALSA (Fig 22) en guise de base de cadencement. Cette

possibilité suppose d'intégrer le bloc d'entrée ou de sortie audio sans utiliser les données associées.

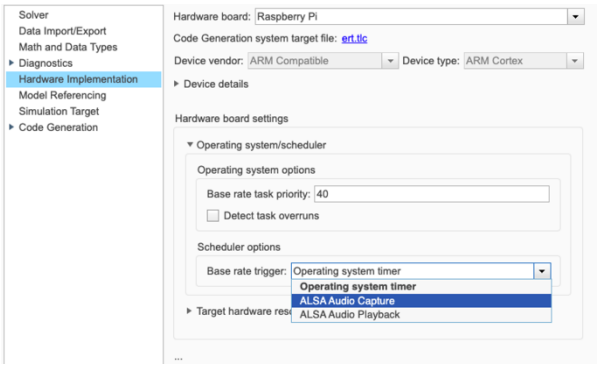


Fig 22 : Paramètres de l'OS/Scheduler

Bibliographie

[1] Pasolini, G., Bazzi, A., & Zabini, F. (2017). A raspberry pi-based platform for signal processing education [sp education]. *IEEE Signal Processing Magazine*, *34*(4), 151-158.

[2] Husseini, Naji S., and Ian Kaszubski. "Incorporating the Raspberry Pi into laboratory experiments in an introductory MATLAB course." *2017 ASEE Annual Conference & Exposition*. 2017.

[3] Hoyo, Ángeles, et al. "Teaching control engineering concepts using open source tools on a raspberry pi board." *IFAC-PapersOnLine* 48.29 (2015): 99-104.

[4] https://www.raspberrypi-france.fr/guide/