BoucharD luka, renaud émile

Élaborer l'installation et l'intégration de systèmes ordinés

247-616-LI , gr.1

Bras robotisé :

**Six axes de liberté**

Travail présenté à

M. Yves Roy

M. Kevin Cotton

M. Stéphane Deschenes

M. François Bouchard

Département de technologie du génie électrique

Cégep Limoilou

Le 31 mars 2020

Table des matières

[Liste des tableaux 2](#_Toc36551764)

[Liste des figures 2](#_Toc36551765)

[Description : 3](#_Toc36551766)

[Démonstration : 3](#_Toc36551767)

[Planification : 3](#_Toc36551768)

[Semaine 1 et 2: 3](#_Toc36551769)

[Semaine 3 : 3](#_Toc36551770)

[Semaine 4: 3](#_Toc36551771)

[Semaine 5 : 3](#_Toc36551772)

[Semaine 6, 7 et 8: 3](#_Toc36551773)

[Semaine 9: 4](#_Toc36551774)

[Semaine 10: 4](#_Toc36551775)

[Livrables : 4](#_Toc36551776)

[Analyse des risques : 4](#_Toc36551777)

[Requis client: 5](#_Toc36551778)

[Requis matériel : 6](#_Toc36551779)

[Requis logiciel : 7](#_Toc36551780)

[Architecture système : 8](#_Toc36551781)

[Architecture Matérielle: 10](#_Toc36551782)

[Carte d’extension du Beaglebone Black 10](#_Toc36551783)

[Contrôleurs de moteurs 11](#_Toc36551784)

[Architecture logicielle : 12](#_Toc36551785)

[Interface usager 12](#_Toc36551786)

[Unité de contrôle 13](#_Toc36551787)

[Contrôleur de moteur 14](#_Toc36551788)

# Liste des tableaux

[Tableau 1 : Requis client 6](#_Toc24041071)

[Tableau 2 : Requis matériel 7](#_Toc24041072)

[Tableau 3 : Requis logiciel 8](#_Toc24041073)

# Liste des figures

[Figure 1: Schéma de l’architecture système global 9](#_Toc24041074)

[Figure 2 :Schéma de l'architecture système du bras robotisé 10](#_Toc24041075)

[Figure 3 : Système de coordonné du robot 10](#_Toc24041076)

[Figure 4: Architecture matériel de la carte d'extension du BeagleBone Black 11](#_Toc24041077)

[Figure 5 : Architecture des contrôleurs de moteurs 12](#_Toc24041078)

[Figure 6 : Architecture logicielle de l'interface usagé 13](#_Toc24041079)

[Figure 7: Architecture logicielle de l'unité de control 14](#_Toc24041080)

[Figure 8 : Architecture logicielle du contrôleur de moteur 15](#_Toc24041081)

# Description :

Le projet consiste à construire un bras robotisé ayant 6 axes de rotation. Il ressemblera à ceux que l’on retrouve dans les usines, mais en modèle réduit. Il pourra être programmé simplement pour faire des tâches répétitives. Le bras pourra être contrôlé grâce à une interface graphique ~~ou une interface console~~. Il sera équipé d’une pince pour prendre des objets.

# Démonstration :

Lors de la démonstration, le bras va bouger tous ses axes un à un pour montrer l’ampleur des mouvements qu’il peut faire. Il va ensuite bouger autour d’un point imaginaire fixe. Il va avoir un mini-programme préprogrammé qui va faire des mouvements à répétition comme dans une usine. Il sera aussi contrôlé à l’aide de l’interface graphique par un être humain.

# Planification :

## Semaine 1 et 2:

* Révision des objectifs
* Planification
* Révision des PCB
* Setup IDE
* Setup gestion fichiers
* Code des pilotes

## Semaine 3 :

* Dessouder les Blue Pills
* Assemblage 2 PCB
* Debug des PCB
* Test des pilotes

## Semaine 4:

* Debug des PCB
* Test des pilotes

## Semaine 5 :

* Assemblage de tous les PCB (Semaine 3 & 4)
* Montage des PCB sur le bras
* Mécanique du bras final

## Semaine 6, 7 et 8:

* Programmation objets

## Semaine 9:

* Interface GUI python
* Intelligence du bras? (python/c++)
* Intégration

## Semaine 10:

* Intégration
* Rapport

# Livrables :

* 6 PCB de pilotes des moteurs.
* 1 PCB d’interfaçage des périphériques du BeagleBone Black.
* 6 axes du bras robotisé.
* 1 interface graphique pour l’écran LCD.
* 1 programme de contrôle des moteurs.
* 1 programme de gestion des capteurs.
* 1 programme de gestions des erreurs.
* 1 programme de communication ~~LIN~~ RS485.
* 1 programme de contrôle intégral du bras robotisé.
* 1 schéma des pilotes des moteurs
* 1 schéma de l’interfaçage des périphériques du BeagleBone Black.
* 1 circuit d’alimentation fonctionnel dans le système.
* ~~1 interface graphique pour le PC (bonus)~~
* 1 rapport complet
* Acheter le matériel nécessaire à la mise en marche du bras.

# Analyse des risques :

Au cours du projet, il y a divers imprévus qui peuvent survenir. Il est possible que la livraison des pièces prend plus de temps qu’il avait été prévu. Si nous n’avons pas accès à certaines ressources matérielles, il est fort probable que nous ne puissions pas progresser dans l’horaire que nous nous étions fixé. Dans la même ordre d’idée, il est à considérer que certaines des pièces se brisent lors du transport, retardant par le fait même l’avancement du projet. Au plan matériel, il est possible que les PCB, une fois soudés, ne soient pas fonctionnels. Aussi, certaines pièces pourraient se briser lors du projet. Il pourrait s’agir de pièces compromettant la structure de notre bras ou encore rendant son fonctionnement impossible. Certaines pièces sont plus à risque, comme les moteurs pas à pas et les pilotes de moteurs. Il s’agit de pièces de « puissance » qui sont plus propice à chauffer et briser. Il faut donc prévoir du matériel supplémentaire, permettant de remplacer les pièces défectueuses. Certains conflits matériels peuvent nous empêcher de mener le projet à terme. Entre autres, il peut y avoir de l’interférence magnétique attribuées aux divers capteurs magnétiques présents sur le circuit. ~~La communication LIN pourrait s’avérer beaucoup plus complexe que prévu à implémenter dans le projet. Puisqu’elle sert de lien entre tous les modules du système, cela pourrait être problématique.~~ Il y a des risques qui sont davantage liés aux facteurs humains et aux facteurs environnementaux. Par exemple, il faut prendre en compte le retard causé par les tempêtes de neige. Bien qu’on débute le projet en mars, il est tout de même envisageable qu’il y ait une tempête. Il pourrait aussi y avoir une panne de courant. Cette dernière pourrait provoquer la perte de fichiers importants (non sauvegardés). En 2019, il est aussi possible que des grévistes extrémistes nous imposent leur façon de voir le monde et que le Cegep soit fermé pour des raisons « d’insécurités émotionnelles » : « On veut être payé pour étudier! On veut des stages payés à regarder quelqu’un d’autre travailler! »

# Requis client:

Le bras doit avoir 6 axes de rotation. Il sera muni d’une pince qui pourra soulever un poids d’au moins 250 grammes. S’il doit répéter un même mouvement plusieurs fois, il devra exécuter ces mouvements avec une imprécision d’au maximum 1 cm. Grâce à une interface, le bras sera programmable en GCode\*\* (À voir). ~~Il y aura des capteurs de positions permettant de savoir si les axes du bras sont à la bonne position.~~ Mise à l’origine du bras par des Hall Sensor. Il y aura un système de code d’erreur permettant de décoder les diverses erreurs que les capteurs du bras nous envoient. Il doit avoir une interface usagée ~~sur ordinateur ou~~ directement sur le bras. Le bras devra être le plus silencieux possible. Chacun des axes du bras doivent être muni de leur propre contrôleur, pour faciliter la communication. Il est primordial d’utiliser la communication ~~LIN~~ RS485 pour parler avec les moteurs pas à pas (voir le Tableau 1).

Tableau : Requis client

|  |  |
| --- | --- |
| Référence | Requis |
| 1 | Les 6 axes du robot doivent bouger |
| 2 | Le bras doit être capable de soulever une charge de 250 grammes. |
| 3 | Le bras doit pouvoir répéter un mouvement avec une imprécision d’au plus 1 cm |
| 4 | Le bras doit pouvoir être programmer facilement pour créer une séquence de mouvement |
| 5 | Le bras doit être muni d’une pince |
| 6 | Le bras doit corriger les erreurs qu’il fait dans les mouvements à l’aide de capteur de positions. |
| 7 | Le bras doit se brancher facilement à un ordinateur |
| 8 | Le bras doit être le plus silencieux possible |
| 9 | Les moteurs doivent être équipé de leur propre contrôleur. |
| 10 | Les moteurs doivent se parler en RS485 ~~LIN bus~~ |
| 11 | Interface graphique pour contrôler manuellement les axes. |
| 12 | Le bras robotisé sera muni d’un système de codes d’erreur. |

# Requis matériel :

Le projet se base sur le bras à 6 axes WE-R2.4. Il aura 3 moteurs NEMA 17 pour les 3 plus grosses articulations. Les trois autres articulations seront munies de moteur pas à pas aussi, mais de plus petits moteurs 35 mm pour des questions d’optimisation de l’espace disponible. Pour contrôler la pince, ~~un moteur DC~~ un servo-moteur sera utilisé. ~~Afin de savoir la position des moteurs, ils seront chacun munis d’encodeur de position magnétique~~. Chaque axe du bras à son propre module. Ils possèdent donc tous un microcontrôleur ~~STM8~~ STM32. Pour permettre au bras d’avoir une fonction « homing », il sera muni d’un capteur à effet Hall. Pour refroidir les pilotes des moteurs et le moteur pas à pas, des ventilateurs seront installé près des PCB et des moteurs. Il y aura aussi plusieurs capteurs de température, pour s’assurer que la température est bien régulée (changer la vitesse des ventilateurs). Des dissipateurs de chaleurs seront aussi utilisé à des fins de refroidissement. Un BeagleBone Black sera utilisé pour contrôler les modules de chaque axe. Il sera connecté à un écran qui permettra d’afficher en temps réel la position du bras ainsi que les diverses codes d’erreur. (voir le Tableau 2)

Tableau : Requis matériel

|  |  |  |
| --- | --- | --- |
| Identification | Référence | Requis matériel |
| M1 | 1 | Moteur NEMA 17 pour les moteurs de puissance |
| M2 | 2 | Moteur pas à pas (35mm) pour les petites articulations |
| M3 | 3 | Contrôleur de moteur intégré sur chaque moteur |
| M4 | 4 | 1 PCB par moteur (contrôleur + encodeur) |
| M5 | 5 | ~~Encodeur de position magnétique pour moteur pas à pas~~ |
| M6 | 6 | ~~STM8~~ STM32 comme contrôleur sur chaque pilote de moteur (QFP or SSOP) |
| M7 | 7 | Capteur à effet Hall pour le retour à l’origine des moteurs |
| M8 | 8 | Capteur de température (TO-92-3) |
| M9 | 9 | Mini ventilateur pour chaque pilote de moteur (refroidissement actif ) |
| M10 | 10 | Dissipateur de chaleur pour les pilotes de moteur (refroidissement passif) |
| M11 | 11 | ~~Moteur DC~~ Servo-moteur pour la pince |
| M12 | 12 | Écran TFT LCD 7’’ |
| M13 | 13 | Bras WE-R2.4 |
| M14 | 14 | 1 PCB pour l’interface du BeagleBone Black et de ses périphériques. |
| M15 | 15 | 1 BeagleBone Black |

# Requis logiciel :

Le projet sera programmé en C++. Les diverses parties du programme devront être bien structurées dans des fichiers dont le nom leur est significatif. Il y aura des pilotes pour chacun des périphériques du système. Une interface graphique sera nécessaire pour le contrôle de la diffusion de l’information sur l’écran LCD. Une ~~autre~~ interface sera utile à la gestion des codes d’erreur du système. Idéalement, la gestion du projet sera faite à l’aide du système de gestion GIT. Les contrôleurs de moteur, puisqu’ils possèdent chacun leur microcontrôleur, seront programmable via ST-Link. Finalement, si nous avons assez de temps, nous aimerions implémenter ~~le logiciel ROS à notre projet~~.(voir la Tableau 3).

Tableau : Requis logiciel

|  |  |  |
| --- | --- | --- |
| Identification | Référence | Requis logiciel |
| S1 | 1 | ~~Logiciel ROS (Bonus)~~ |
| S2 | 2 | Contrôleur de moteur programmable via ST-Link |
| S3 | 3 | Utilisation d’un système de gestion de fichiers GIT |
| S4 | 4 | 1 pilote de direction des moteurs pas à pas |
| S5 | 5 | 1 pilote de capteur à effet Hall |
| S6 | 6 | 1 pilote de capteur de température |
| S7 | 7 | ~~1 pilote d’encodeur magnétique~~ |
| S8 | 8 | Interface graphique en ~~C#~~ Python pour le LCD |
| S9 | 9 | 1 service de communication ~~LIN~~ RS485 |
| S10 | 10 | 1 pilote pour l’écran LCD |
| S11 | 11 | 1 pilote de contrôle des ventilateurs |
| S12 | 12 | 1 interface de gestion des codes d’erreurs |
| S13 | 13 | 1 service du TMC2130 (pilote de moteur) |
| S14 | 14 | 1 interface pour l’écran LCD |
| S15 | 15 | 1 pilote de synchronisation d’horloge |
| S16 | 16 | Le projet est programmé en C++ |
| S17 | 17 | Structurer le programme en fichiers |

# Architecture système :

Le système est composé principalement du bras robotisé. Il va parler à une unité de contrôle via un lien ~~LIN~~ RS485. Cette unité de contrôle va avoir la capacité d’envoyer des commandes à chaque moteur pour contrôler le bras. Il va jouer le rôle de décodeur entre les demandes de l’utilisateur et les commandes envoyées aux moteurs. L’utilisateur va pouvoir utiliser le bras robotisé à partir de l’interface usagé sur ~~l’ordinateur~~ l’écran. L’alimentation va, en premier lieu, prendre le 120V d’une prise secteur et le transformer en ~~24V~~ 12V pour alimenter les moteurs. Une partie de ~~24v~~ 12V va être ensuite transformée en 5v pour alimenter les contrôleurs de moteur.

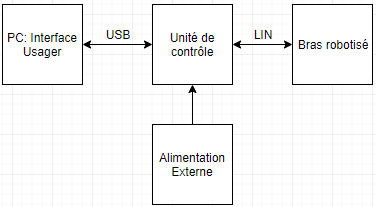


Figure : Schéma de l’architecture système global

Le bras robotisé comporte à son tour plusieurs sous-section (Voir Figure 2). Chaque axe est une sous-section. Ils sont nommés et comportent chacun un moteur et un contrôler de moteur. Les 3 premiers axes ont un moteur de type Nema 17 tandis que les 3 derniers axes ont des moteurs plus petit de type L6HAYYYK0023, communément appelé *tin-can stepper.*

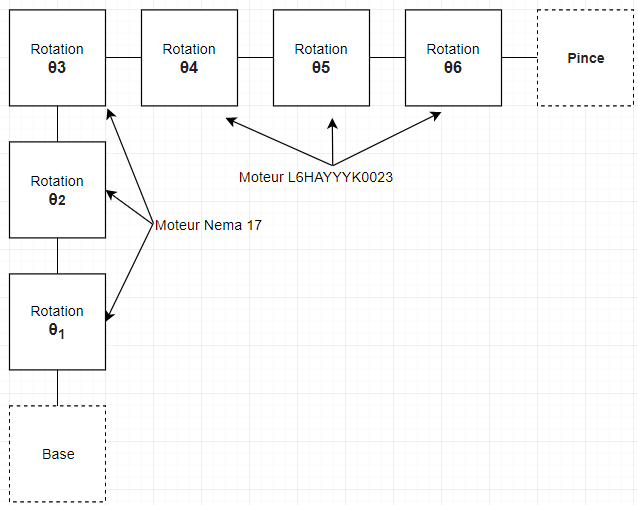


Figure :Schéma de l'architecture système du bras robotisé

Le bras robotisé va utiliser un système de coordonné Y, X, Z pour faciliter son utilisation par l’usager.

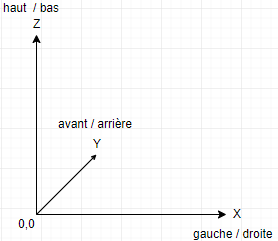


Figure : Système de coordonné du robot

# Architecture Matérielle:

## Carte d’extension du Beaglebone Black

L’unité de contrôle est basée sur un BeagleBone Black. Le circuit imprimé va pouvoir se piquer dessus. Il va avoir un convertisseur pour notre protocole de communication ~~LIN~~ RS485 ~~ainsi qu’un convertisseur de tension pour alimenter les composantes à 5v~~. Il y a une option de mettre un écran tactile sur le PCB aussi. (voir la Figure 5)

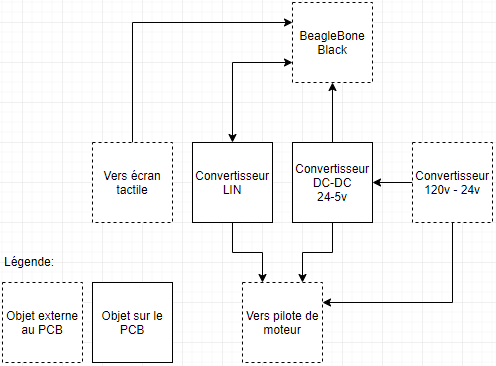


Figure : Architecture matériel de la carte d'extension du BeagleBone Black

## Contrôleurs de moteurs

Les contrôleurs de moteur sont basés sur le microcontrôleur STM8 STM32 (Voir Figure 5). Pour la communication LIN RS485, ils vont utiliser la même composante que sur la carte d’extension du BeagleBone Black. Le STM8 STM32 va avoir plusieurs entrées comme ~~un encodeur magnétique,~~ un capteur de température ainsi qu’un capteur à effet Hall pour s’assurer du bon fonctionnement du moteur. Ils vont pouvoir contrôler un ventilateur pour gérer la température du circuit. Ils vont contrôler les moteurs à l’aide d’une composante spécialement conçu pour cette application. Nous allons pouvoir communiquer avec ce pilote grâce au protocole de communication SPI. Des connecteurs sont prévus pour brancher le moteur, le ventilateur, l’alimentation ~~24v~~ 12V et 5v et le signal ~~LIN~~ RS485.

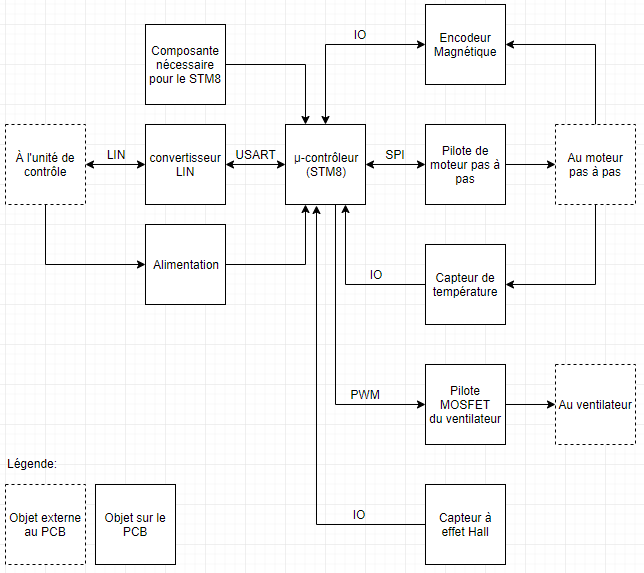


Figure : Architecture des contrôleurs de moteurs

# Architecture logicielle :

## Interface usager

L’interface usager va être disponible ~~sur un ordinateur ou~~ directement sur l’unité de contrôle. ~~Dans le cas où elle se retrouve sur un ordinateur. Nous allons faire une application en C# qui communique via le port série au BeagleBone Black.~~ Nous allons mettre plusieurs boutons pour contrôler les axes et enregistrer des séquences (voir la Figure 6).

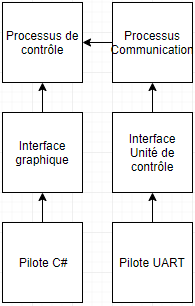


Figure : Architecture logicielle de l'interface usagé

## Unité de contrôle

Sinon nous allons intégrer l’interface dans l’unité de contrôle. Nous allons le programmer en ~~C++~~ Python. Nous allons prendre un protocole ~~LIN~~ RS485 déjà mis en place et l’utiliser comme service. Ce protocole va utiliser les pilotes USART. ~~De plus nous avons la possibilité d’utiliser ROS (~~*~~robot operating system~~*~~) pour faire les calculs mathématiques nécessaires pour une utilisation autonome du bras~~ (voir Figure 7).

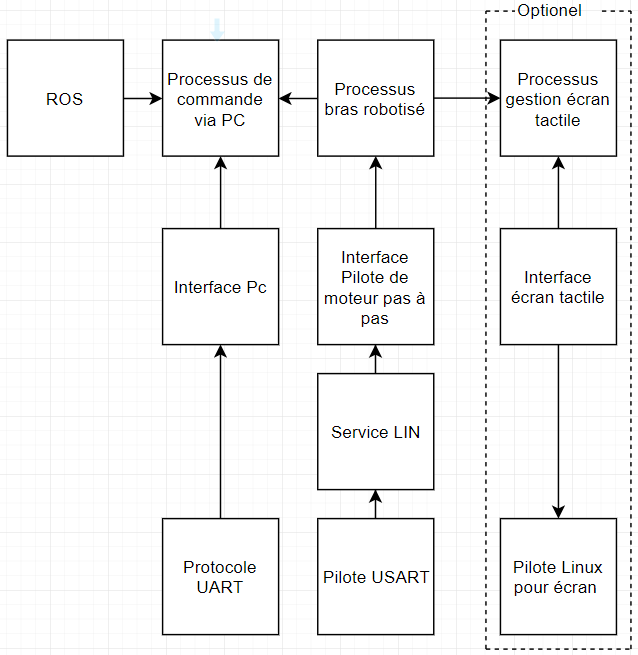


Figure : Architecture logicielle de l'unité de control

## Contrôleur de moteur

Le processus de température va utiliser les modules nécessaires pour contrôler le ventilateur et lire le capteur de température. Le processus moteur va utiliser les modules pour utiliser le TMC2130 et les modules pour aller lire son capteur de fin de course et capteur de position. Ces deux derniers processus en utilisent un autre pour la communication.

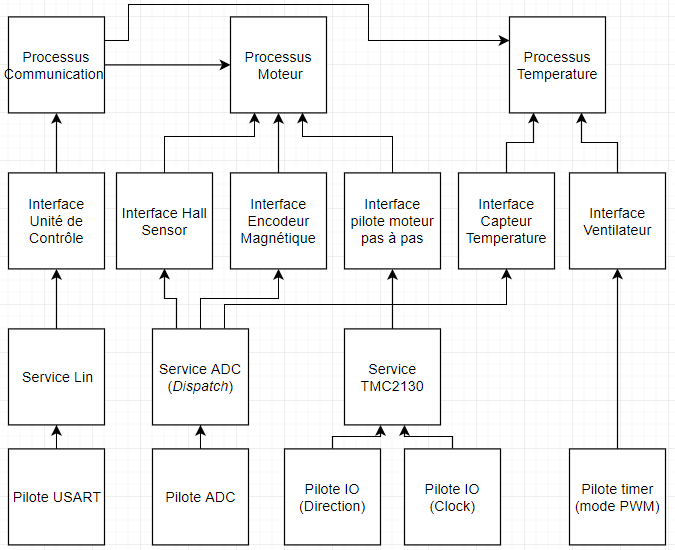


Figure : Architecture logicielle du contrôleur de moteur