

**HAYES Jean-Philippe**

**DII4**

**HERGAULT Jeremy**

# Rapport projet d'électronique : **HEXAPOD**

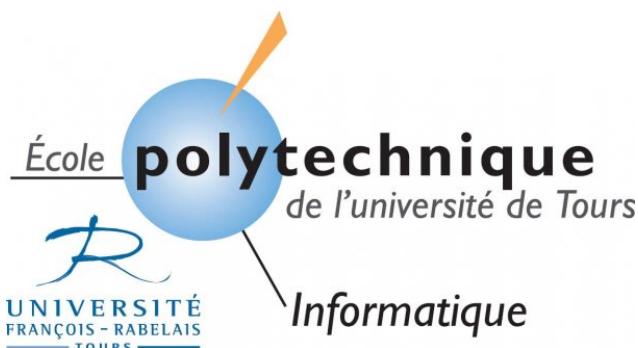
## Remerciement

Nous tenons à remercier Mr MAKRIS Pascal d'avoir accepté d'être notre encadrant pour ce projet fléché, pour sa disponibilité et son investissement personnel.

Il nous a donné les moyens de concevoir la partie électronique de ce robot, de mener une étude complète sur ce produit et nous permettra, dans un second temps, de développer un programme de contrôle de mouvement de robot.

Nous voudrions aussi remercier Mr. Patrick Martineau et l'école Polytech'Tours pour avoir investi dans ce matériel typé robotique.

Enfin, Je voudrais remercier la société « Génération Robot » et plus particulièrement Mr. Yassine Serhouchni qui a joué le rôle d'intermédiaire entre nous et le distributeur TROSSEN ROBOTICS pour acquérir la base mécanique de l'hexapod. Il a aussi été présent pour nous aider en qualité de conseillé technique pour les servomoteurs.



## Sommaire

### Table des matières

Remerciement.....	2
1) Organisation.....	4
a) Introduction .....	4
b) Cahier des charges .....	5
2) Système Existant .....	6
a) Servomoteur AX12 .....	6
b) Base mécanique .....	11
c) Carte électronique ArbotiX .....	12
3) Modification apporté .....	13
a) Communication I2C .....	13
b) Accéléromètre / Gyroscope .....	14
c) Composant logique (transceiver) .....	16
d) Etude 1 : Solution avec microcontrôleur ATxMega.....	17
e) Etude 2 : Solution avec uniquement un Raspberry Pi .....	18
f) Raspberry Pi .....	19
g) Bilan des solutions.....	21
1) Point de vue énergétique.....	21
2) Point de vue financier .....	22
h) Comparaison XBee / Bluetooth / Wifi.....	23
4) Réalisation.....	24
a) Commande et montage de la base mécanique.....	24
b) Conception du schéma électrique .....	24
c) Conception du typon .....	25
d) Confection de le carte électronique.....	26
5) Test de fonctionnement.....	27
6) Organisation projet .....	28
7) Conclusion.....	30
8) Table des illustrations .....	31

## 1) Organisation

### a) Introduction

Le projet « HEXAPOD » est un projet fléché qui a été proposé dans le but de travailler sur différentes notions acquises lors de notre première partie de formation d'ingénieur.

L'HEXAPOD est un robot autonome qui a la particularité de posséder six pattes lui permettant de se déplacer dans toutes les directions. Le kit avec lequel nous avons choisi de travailler peut marcher de manières différentes (comme un crabe ou encore comme une araignée), la télécommande fournit autorise des vitesses différentes, en fonction des choix de l'utilisateur.

L'objectif principal de ce projet est de réaliser une étude complète afin de déterminer la solution la plus adapté pour améliorer le système existant.

L'étude et la réalisation s'étendront sur les deux projet DII4 pour répondre aux exigences des projets électroniques et développement définit par l'école polytech' de Tours.

L'étude et la conception seront réalisées sur une base mécanique existante conçu et distribué par la société TROSSEN ROBOTICS sous le nom de « PhantomX AX Hexapod Mark II ». Seul les actionneurs, le châssis (corps + pattes) et la batterie seront conservés pour ce projet tandis que nous travaillerons sur la partie conception électronique et développement du robot.

Dans un premier temps, nous allons analyser le système existant et plus particulièrement les servomoteurs afin de définir les réels besoins de ce système embarqué. Il va falloir définir de quel manière les actionneurs communiquent avec le « contrôleur », et déterminer le matériel le plus adapté pour utiliser les « muscle » de notre robot.

Dans un second temps, nous présenterons les modifications que nous souhaitons apporter au système embarqué en justifiant nos choix technologiques et proposerons une solution modulable et évolutive pour atteindre notre objectif.

Dans un troisième temps, nous vous décrirons l'étape de la commande du système existant et présenterons la carte réalisé. Enfin nous comparerons notre système avec le robot vendu par TROSSEN ROBOTICS et expliquerons ce que ce projet nous a apporté.

### b) Cahier des charges

Etant donné que le sujet est fléché c'est nous qui avons défini le premier cahier des charges. En accord avec notre encadrant nous avons fait un avenant à celui-ci. Il a donc évolué au fur et à mesure de nos recherches pour répondre à une conception d'un produit cohérent et correspondant à un public large.

- Analyser la gestion de l'ensemble des servomoteurs
- Déterminer l'architecture en fonction des besoins de l'application
- Réaliser une carte d'alimentation / commande

### Les objectifs :

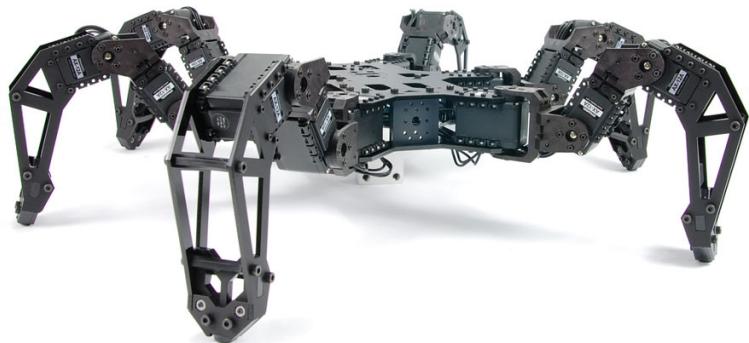
- ▶ Amélioration d'un système existant
- ▶ Proposer une analyse cohérente, modulable et évolutive
- ▶ Réaliser une architecture fonctionnelle

### Les besoins :

Le robot est une base de travail et doit facilement être exploitable et réutilisable pour d'autres études ou améliorations.

On doit par exemple pouvoir rajouter une bibliothèque de mouvements sans être obligé de modifier l'ensemble de la structure du programme.

Des périphériques différents pourraient être ajoutés pour améliorer les performances comme une caméra « kinect » par exemple (qui possède un capteur de distance), donc le choix du matériel doit être flexible open source mais cohérent.



## 2) Système Existant

### a) Servomoteur AX12

#### Caractéristique AX12

Les servomoteurs de la marque Dynamixel sont des actionneurs modulaires intelligents qui intègrent un moteur à courant continu de précision, un réducteur de vitesse et un circuit de commande possédant des fonctionnalités de réseau, le tout dans un élément de taille compact.

En effet, ces actionneurs numériques ne se commandent plus par le classique système de modulation en largeur d'impulsions périodiques(PWM), mais par un jeu de commandes (trames) qui circulent sur un bus numérique. Celui-ci permet la connexion de plusieurs actionneurs sur un bus unique et des possibilités de retour d'information des servomoteurs vers un contrôleur.

Il existe une différence entre l'appellation "servomoteur à commande numérique" de "servomoteur numérique", ces derniers conservant une commande analogique par impulsions, mais ayant remplacé l'asservissement par ampli-op par un asservissement numérique, plus précis et éventuellement configurable.

Observons les caractéristiques du composant AX12-A :

Caractéristique	description
Amplitude	300° ou rotation continue
Couple	16 kg.cm sous 10 V
Vitesse	0.2 s/deg sous 10V
Consignes	position et vitesse
Résolution	les consignes sont quantifiées sur 1024 pas (soit 0.35° en position)
Réglages	vitesse, couple maximum, courbes d'évolution du couple à l'approche de la position cible, dead zone,...
Feedback	position, vitesse, charge mécanique, température, tension
Alarmes et sécurités	tension, température, couple
Câble de Communication	bus 3 fils TTL , débit jusqu'à 1Mbps
Divers	possibilité de synchronisation des mouvements de plusieurs servomoteurs par bufferisation des commandes respectives et déclenchement synchronisé des mouvements

La gamme de tension d'alimentation possible pour alimenter un servomoteur est compris entre 7 et 10 V (tension recommandé : 9.6V) avec une consommation en courant, en pleine charge, de 900mA par moteur et 10mA au repos.

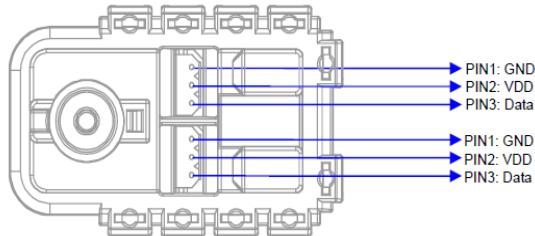
Les servomoteurs sont chainables jusqu'à 254 sur le même bus et sont identifiés par un numéro d'identification.

La consommation n'est pas un élément négligeable de notre projet car les servomoteurs sont au nombre de 18. On comprend facilement qu'il va falloir prendre en compte cette consommation importante dans ce système embarqué, même si l'ensemble des actionneurs ne fonctionnera pas en pleine charge au même moment.

Les servomoteurs sont contrôlés par le biais d'un bus de communication de type « Half duplex Asynchronous Serial » sur un seul fil (pas de bit de parité). Ce fonctionnement se justifie par le fait que les moteurs reçoivent des ordres de commandes et ensuite renvoient un certain nombre d'informations sur leurs états. Nous pensons qu'une communication full duplex aurait été plus adaptée.

La communication série asynchrone décrit un protocole de communication série, dans lequel un signal de démarrage est envoyé avant chaque octet (1bit de Start) et un signal d'arrêt (bit de stop) est envoyé après que chaque mot de code, de manière à rendre la communication asynchrone.

Le signal de démarrage sert à préparer le mécanisme de réception et d'enregistrement d'un symbole de la part du moteur et le signal d'arrêt sert à amener le mécanisme de réception de repos en préparation pour la réception du symbole suivant. Il n'y a pas de bit de parité qui permet de détecter les éventuelles erreurs car il y a dans le message envoyer un checksum pour s'assurer de la cohérence du message.



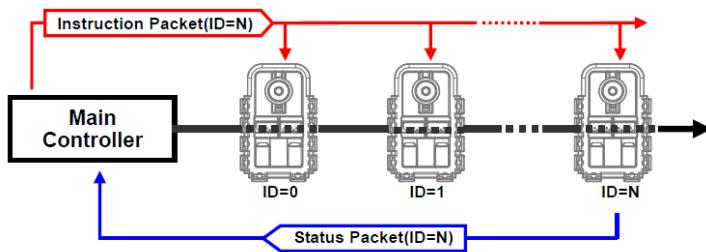
TTL (Transistor Transistor Logic) n'est pas un protocole. C'est une technologie plus ancienne de la logique numérique, mais le nom est souvent utilisé pour se référer à la tension d'alimentation de 5 V, souvent à tort référant à ce qui devrait être appelé UART.

### Protocole de communication

Le système de contrôle communique avec les unités dynamixels par envoie et réception de paquet de données. Il y a deux types de paquet :

- Instruction Packet : envoyer depuis le système de contrôle pour commander les servomoteurs
- Status Packet : envoyer depuis les servos pour rendre compte de son état

Cette notion de retour d'information permet de développer des systèmes asservis pour obtenir un contrôle total entre la demande et le résultat de positionnement de l'actionneur.



La structure des actionneurs étant chainé sur un même bus, les « Instructions Packet » sont identifiées par un numéro d'identification compris entre 0 et 253. Il est impératif que chaque servomoteurs aient chacun son propre ID pour éviter les risques de collisions et les problèmes de communications. Ce système permet de cibler le servomoteur à qui on veut envoyer une trame de commande.

Tous les messages qui transitent sur le bus sont constitués d'une succession de nombres codés en binaire. Pour plus de facilité, nous les écrivons en hexadécimal (0x00 à 0xFF). On distingue deux types de message : ceux reçus par les AX12-A (ordre ce commande : « Instruction ») et ceux envoyés par les AX12-A (« statut »). Les messages envoyés au récepteur dynamixel sont structurés selon le modèle suivant :

**Instruction Packet**      **0xFF 0xFF ID LENGTH INSTRUCTION PARAMETER1 ... PARAMETER N CHECK SUM**

Nous allons maintenant présenter la structure des paquets d'instructions :

**0xFF 0xFF** : Permet de prévenir les moteurs que l'on va leur parler.

**ID** : Donne l'identifiant du servomoteur auquel on souhaite envoyer une instruction

**LENGTH** : La longueur du message qui sera défini comme le nombre de paramètres N + 2 (instruction + N paramètres + checksum).

**INSTRUCTION** : Instruction de pilotage du servomoteur. La liste des actions possibles sont regroupées dans le tableau suivant :

Instruction	Fonction	Valeur	Nombre du paramètre
PING	Pas d'action. Utiliser pour obtenir le statut du paquet	0x01	0
READ DATA	Lire des valeurs dans la « control table »	0x02	2
WRITE DATA	Écrire des valeurs dans la control table	0x03	2~
REG WRITE	Semblable à WRITE_DATA, mais reste en veille mode jusqu'à ce que l'instruction Action soit	0x04	2~

	donnée		
ACTION	Déclenche l'action enregistrée par l'instruction REG_WRITE	0x05	0
RESET	Change la valeur d'un paramètre de la control table pour la remettre à défaut.	0x06	0
SYNC WRITE	Utiliser pour contrôler plusieurs actionneurs en même temps	0x83	4~

En fait les AX possèdent une table de valeurs modifiables (la position à atteindre, la vitesse à laquelle ils doivent se déplacer,...) et de valeurs non modifiables qui sont mises à jour en permanence (la position actuelle, la température...).

On pourra donc demander la valeur inscrite dans la case température par exemple avec un READ DATA pour qu'il nous envoie sa température interne, ou modifier la position à atteindre avec un WRITE DATA pour qu'il aille à cette position automatiquement.

**PARAMETRE 1..N** : Utilisé si il y a des informations complémentaires à envoyé autre que l'instruction elle-même.

**CHECK SUM** : pour vérifier que le message n'a pas été modifié ou corrompu.

De la même manière que « Instruction Packet », nous allons analyser la structure des trames de d'état qui sont renvoyé par les servos moteur AX12-A :

OXFF OXFF ID LENGTH ERROR PARAMETER1 PARAMETER2 ... PARAMETER N CHECK SUM

Nous allons maintenant présenter la structure des « Status Packet » :

**OXFF OXFF** : Permet de d'indiquer le début du paquet.

**ID** : Correspond à l'unique numéro d'identification du servomoteur qui retourne le paquet

**LENGTH** : la longueur du message qui sera défini comme le nombre de paramètres N + 2 (instruction + N paramètres + checksum).

**ERROR** : Ce bit représente l'éventuelle l'erreur provoqué par le message de commande :

ERREUR	Nom	détails
Bit 7	0	-
Bit 6	Instruction Error	Mis à 1 si une instruction non définie est envoyé ou une action instruction est envoyée sans une instruction de REG_WRITE.
Bit 5	Overload Error	Mis à 1 si le couple maximal spécifié ne peut pas contrôler la charge appliquée.
Bit 4	Checksum Error	Mis à 1 si la somme de contrôle du paquet

		d'instruction est incorrecte.
Bit 3	Range Error	Mis à 1 si l'instruction est envoyée hors de la plage définie.
Bit 2	Overheating Error	Mis à 1 si la température interne de l'appareil est Dynamixel dessus de la plage de température de fonctionnement tels que définis dans latable de contrôle.
Bit 1	Angle Limit Error	Définit comme 1 si la position de l'objectif est fixé en dehors de la plage entre CW Angle limite et CCW Angle Limite.
Bit 0	Input Voltage Error	Mis à 1 si la tension est en dehors de la plage de tension de fonctionnement en tant que défini dans le tableau de commande.

Ces servomoteurs permettent de s'assurer que l'instruction de commande a bien été pris en compte et qu'il est en mesure de réaliser ce qu'on lui a demandé de faire.

**PARAMETRE 1..N** : Utilisé si il y a besoin d'indiquer des informations

**CHECK SUM** : pour vérifier que le message n'a pas été modifié ou corrompu.

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)

Si la valeur calculée est supérieure à 255, l'octet de poids faible est définie comme étant la somme de contrôle valeur. ~ Représente l'opération logique NON.

#### La table de contrôle

La table de contrôle contient les informations de statuts et d'opérations des actionneurs « dynamixel ». On vient écrire par le biais d'une « Instruction Packet » dans les paramètres des actionneurs et les statuts sont vérifiés par lecture de valeur dans celle-ci.

Cette table de contrôle ne sera pas décrite dans ce rapport. Cependant, il existe deux zones de données. La première est la zone mémoire EEPROM, qui correspond à de la mémoire non volatile, ou les informations modifiés par des instructions de commande resteront, même après la mise hors tension. Tandis que les valeurs des données de la zone RAM seront réinitialisées à leur valeur par défaut à chaque mise sous tension.

On remarque également que chaque « caractéristique » est accessible en lecture et/ou écriture et qu'il possède chacun un paramètre par défaut. La description de chaque item est expliquée dans la datasheet du servomoteur utilisé ici.



### 3-4. Control Table

Address	Item	Access	Initial Value
0(0X00)	Model Number(L)	RD	12(0x0C)
1(0X01)	Model Number(H)	RD	0(0x00)
2(0X02)	Version of Firmware	RD	?
3(0X03)	ID	RD,WR	1(0x01)
4(0X04)	Baud Rate	RD,WR	1(0x01)
5(0X05)	Return Delay Time	RD,WR	250(0xFA)
6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
10(0x0A)	(Reserved)	-	0(0x00)
11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
14(0X0E)	Max Torque(L)	RD,WR	255(0xFF)
15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
16(0X10)	Status Return Level	RD,WR	2(0x02)
17(0X11)	Alarm LED	RD,WR	4(0x04)
18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
19(0X13)	(Reserved)	RD,WR	0(0x00)
20(0X14)	Down Calibration(L)	RD	?
21(0X15)	Down Calibration(H)	RD	?
22(0X16)	Up Calibration(L)	RD	?
23(0X17)	Up Calibration(H)	RD	?
24(0X18)	Torque Enable	RD,WR	0(0x00)
25(0X19)	LED	RD,WR	0(0x00)
26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
32(0X20)	Moving Speed(L)	RD,WR	0
33(0X21)	Moving Speed(H)	RD,WR	0
34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
36(0X24)	Present Position(L)	RD	?
37(0X25)	Present Position(H)	RD	?
38(0X26)	Present Speed(L)	RD	?
39(0X27)	Present Speed(H)	RD	?
40(0X28)	Present Load(L)	RD	?
41(0X29)	Present Load(H)	RD	?
42(0X2A)	Present Voltage	RD	?
43(0X2B)	Present Temperature	RD	?
44(0X2C)	Registered Instruction	RD,WR	0(0x00)
45(0X2D)	(Reserved)	-	0(0x00)
46(0xE)	Moving	RD	0(0x00)
47(0xF)	Lock	RD,WR	0(0x00)
48(0x30)	Punch(L)	RD,WR	32(0x20)
49(0x31)	Punch(H)	RD,WR	0(0x00)

#### b) Base mécanique

Vous l'aurez compris, c'est actionneur sont plus que des servomoteurs même si ils en ont la fonction. Nous voulions travailler absolument avec ce matériel qui offre énormément de possibilités et ce type de matériel qui deviendra sûrement un standard dans quelque années pour ce genre de produit.

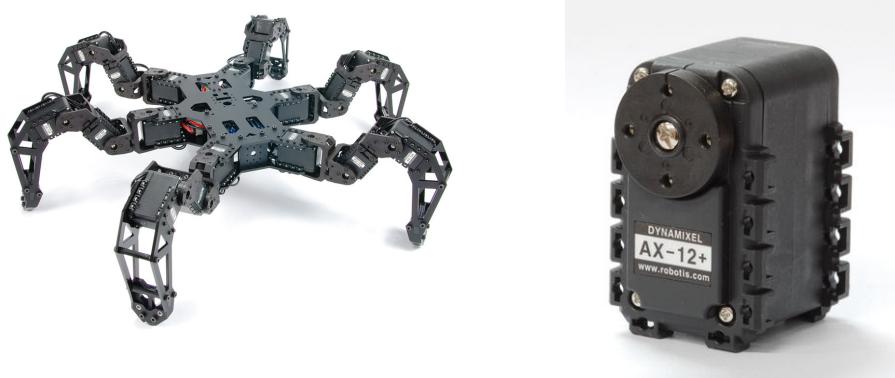
Nous nous sommes orienté vers ce robot pour 2 raisons, la première est qu'il coordonnait ses mouvements grâce aux actionneurs AX12-A, le second est son aspect esthétique. En effet, La

structure de son corps et de ses pattes en font un robot imposant et moderne. De plus, il offre beaucoup de possibilité de mouvement.

Le kit comprend l'ensemble des pièces pour faire fonctionner le robot :

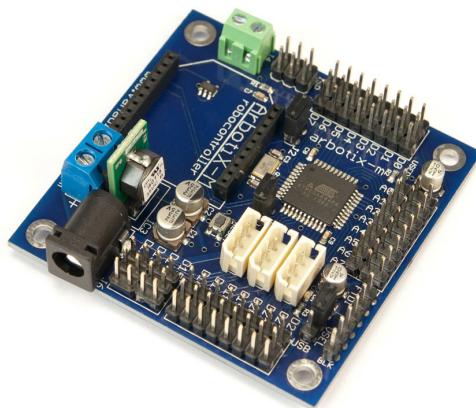
- Le robot en pièce détaché à monter
- 18 servomoteurs AX12-A
- La télécommande permettant de piloter le robot
- La batterie et son chargeur associé

Après avoir reçu la base mécanique, nous nous sommes aperçus que celle-ci était de mauvaise qualité. Les pièces usinées au laser dans le kit (corps + pattes) n'étaient pas très bien ajustées. Les seules pièces de bonne qualité étaient celles fournies entre les servomoteurs.



### c) Carte électronique ArbotiX

La carte ArbotiX, présent dans le kit fourni, repose sur un ATMega644p cadencé à 16MHz. Il possède 1 UART série, 3 ports pour les servomoteurs AX12, 28 entrées/sorties, et 1 emplacement pour un module XBee.



Nous avons voulu remplacer cette carte, car nous voulons ajouter des capteurs de mouvement de type accéléromètre et gyroscope, et aucun module de ce type ne prend en charge des communications par simple entrées / sorties.

De plus les liaisons vers l'extérieur sont limitées uniquement aux modules XBee.

### 3) Modification apporté

#### a) Communication I2C

Le protocole de communication I2C ( Integrated-circuit Communication) a été développé par Philips au début des années 80. Il est destiné à interconnecter des composants qui sont situés les uns à proximité des autres.

Le bus I2C est un bus de données série synchrone half-duplex. Les données sont transmises les une à la suite des autres et il y a donc transmission de l'horloge.

Plusieurs équipements, soit maîtres, soit esclaves, peuvent être connectés au bus et les échanges ont toujours lieu entre un seul maître et un (ou tous les) esclave(s), toujours à l'initiative du maître

La connexion est réalisée par l'intermédiaire de 2 lignes :

- SDA (Serial Data Line) : ligne de données bidirectionnelle,
- SCL (Serial Clock Line) : ligne d'horloge de synchronisation bidirectionnelle.

Les 2 lignes sont tirées au niveau de tension  $V_{DD}$  à travers des résistances de pull-up ( $R_p$ ) placées au plus près du maître pour limiter la CEM. Il ne faut également pas oublier la masse qui doit être commune aux équipements.

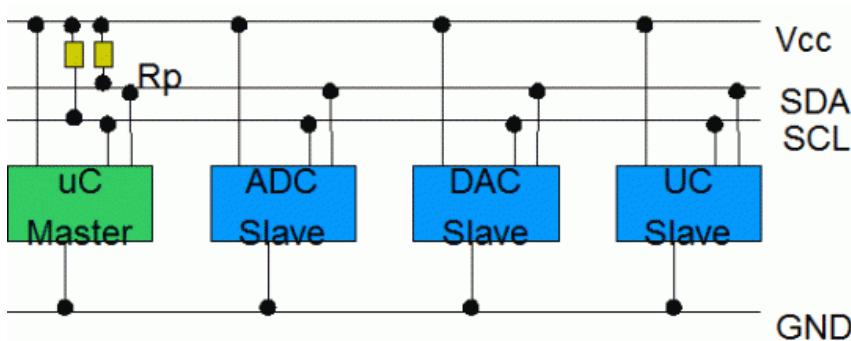


Figure 1 : Ligne I2C

Comme chaque composant apporte une capacité en parallèle, le nombre maximal d'équipements est limité afin que la charge capacitive maximale du bus soit inférieure à 400pF.

## b) Accéléromètre / Gyroscope

Ces deux composants ont été choisis chez le même constructeur (« STMicroelectronics ») afin de favoriser leur comptabilité sur le même bus de communication. En effet, les trames d'information et de commande sont structurées de la manière. Les données de mesure qui sortent de ces composants sont sur 16bit.

### Accéléromètre



- Boitier LGA (85°C)
- Tension d'alimentation : 2,16V ->3,6V
- 3 axes de mesure
- Courant de consommation : 250, 10, 1 µA
- plages d'accélération linéaire de ± 2 , ± 4 et ± 8 g
- Mode basse consommation

Figure 2 : Accéléromètre

Le LIS331DLH est un accéléromètre linéaire basse consommation performant réalisant des mesures sur trois axes.

L'appareil dispose de modes qui permettent des économies d'énergie et de sommeil intelligent pour réactiver les fonctions.

Le LIS331DLH permet de sélectionner dynamiquement des plages de réglage et de sensibilité et il est capable de mesurer des accélérations avec des débits de données de sortie de 0,5 Hz à 1 kHz. La capacité d'autotest permet à l'utilisateur de vérifier le fonctionnement du capteur.

Le la fonction « Sleep to wake-up », en liaison avec le mode de faible puissance, permet de réduire la consommation d'énergie du système et de développer de une application intelligentes. LIS331DLH peut être réglé dans un mode de fonctionnement à faible puissance, caractérisé par des débits de données inférieurs rafraîchissements. De cette façon, l'appareil, même si le sommeil, continuez à détecter une accélération et générer des demandes d'interruption.

Lorsque la fonction « Sleep to wake-up » est activée, LIS331DLH peut automatiquement se réveiller dès que l'événement d'interruption a été détecté, et augmenter le débit de données de sortie et la bande passante.

Avec cette fonction, le système peut être efficacement mis en mode de faible puissance à « full » performance en fonction du positionnement et de l'accélération des événements sélectionnables par l'utilisateur, et ainsi assurer économie et la flexibilité du système.

### Gyroscope



- Boîtier LGA (85°C)
- Tension d'alimentation : 2,4V -> 3,6V
- 3 axes de mesure
- plages d'accélération linéaire de  $\pm 250/500 \pm / \pm 2000$  dps
- Mode basse consommation
- gamme de températures de -40 ° C à +85 ° C.

Figure 3 : Gyroscope

Le L3GD20 est un gyroscope calculant la vitesse angulaire sur les trois axes. Ce capteur de vitesse de faible puissance comprend un élément de détection capable de fournir la vitesse angulaire mesurée à un composant « maître » par l'intermédiaire d'une interface numérique (I2C/SPI).

Le L3GD20 possède une gamme de mesure de  $\pm 250 / 500 \pm / \pm 2000$  dps et une sensibilité sélectionnable par l'utilisateur.

Ce composant a été sélectionné pour sa modularité avec une grande gamme de paramétrage ajustable mais aussi pour sa faible consommation dépendant de son mode de fonctionnement : 6.1mA en fonctionnement normal, 2mA en mode veille permettant un redémarrage plus rapide par rapport à la mise hors tension.

L'interface I2C possède 2 modes de contrôle : le mode standard (100kHz) et le mode rapide (400kHz) pour la fréquence de l'horloge. Et les temps de communication sont évidemment beaucoup plus court (standard = 3.45 $\mu$ s et 0.9 $\mu$ s).

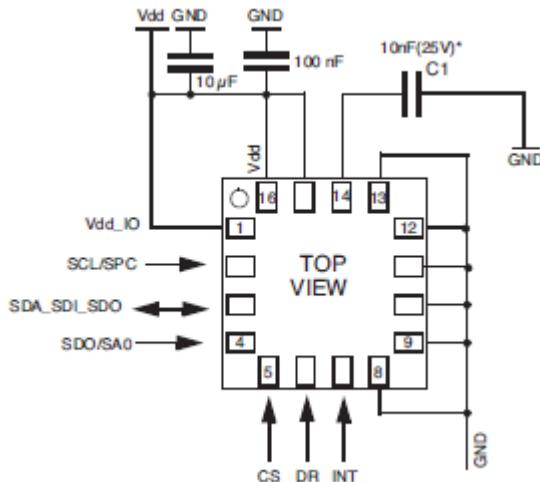


Figure 4 : Schéma gyroscope

L'adresse de l'esclave (SAD) associé au L3GD20 est 110101xb. La broche SDO peut être utilisée pour modifier le bit de poids faible de l'adresse du dispositif. Si la broche SDO est connectée à alimentation en tension, LSb est à '1' (adresse 1101011b). Dans le cas contraire, si la broche de SDO est reliée à masse, la valeur LSB est '0' (adresse 1101010b).

Cette solution permet de se connecter et de répondre à deux gyroscopes différents sur le même bus I2C. Dans notre cas nous avons vérifié que les deux dispositifs possèdent des adresses esclaves différentes afin qu'il n'y ait pas de confusion sur le bus de communication.

### c) Composant logique (transceiver)

En ayant choisi des niveaux de tension de 3.3 V, notre problème était que les servomoteurs communiquent par une liaison série TTL.

La technologie TTL est normalisée pour une tension d'alimentation de 5 V. Un signal TTL est défini comme niveau logique bas entre 0 et 2.4 V, et comme niveau logique haut entre 2,8 V et 5 V (ces niveaux peuvent varier légèrement).

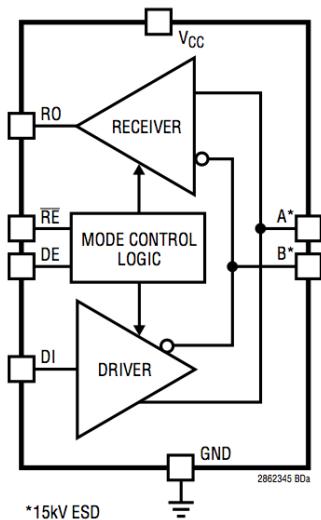
Il a donc fallu trouver un composant qui puisse convertir la tension de 3,3V, en 5V. Mais surtout qu'il fasse office de multiplexeur pour l'information RX / TX.

Nous avons donc trouvé ce composant, qui prend en charge les bons niveaux de tension. En effet, lorsqu'il est alimenté en 5V, l'état haut de l'étage d'entrée est au minimum de 2.25, ce qui est suffisant pour les 3,3V de l'état haut du Raspberry pi.

La communication avec les moteurs se fait avec 1 seul fil sur lequel transitent les données de commandes, et le retour d'information renvoyé par les moteurs.

Pour cette partie un choix s'impose : Soit on gère le problème de manière software, pour que celui qui émet les ordres ne reçoive pas ses propres données sur le RX. Cette solution est envisageable avec le moteur AX12-A puisque l'on peut avec un trame de commande lui demander de ne pas nous renvoyer d'information d'état.

Sinon, on peut gérer cette partie de manière Hard, avec la bascule que nous avons choisie. C'est avec les Pins RE et DE, que l'on choisit le mode que l'on souhaite selon une table de vérité :



DE	RE	Mode	A	RO
0	0	Réception de données	$R_{IN}$	Active
0	1	Aucune transmission	$R_{IN}$	High-Z
1	0	Emission et réception de données	Active	Active
1	1	Emission de données	Active	High-Z

Figure 5 : Schémas Transceiver

Dans notre cas, nous utiliserons uniquement le mode « émission de données » dans un fonctionnement normal. La gestion de la liaison série sera de type asynchrone, car on enverra tous les ordres de positionnements à la suite.

Et si nous voulons asservir le moteur, nous utiliserons les modes « réception de données », et « émission de données ». Dans ce cas la gestion de la liaison série sera de type synchrone, car il faudra attendre d'avoir reçu la réponse du servomoteur interroger avant d'émettre une autre trame de commande.

Dans aucun cas, nous utiliserons le mode « Emission et réception de données », car il ne faut pas que l'émetteur reçoive ses propres données envoyées.

L'autre intérêt de ce composant, est qu'il permet d'adapter le niveau de tension du Raspberry pi (3,3V), à celle du servomoteur (5V TTL).

#### d) Etude 1 : Solution avec microcontrôleur ATxMega

Après une première étude, nous voulions gérer les servomoteurs à l'aide d'un ATxMega, qui est la gamme des plus puissants microcontrôleurs ATMEL.

Cette solution devait nous permettre de gérer les servomoteurs, de les asservir avec leur retour d'information, et les capteurs de mouvements.

L'intérêt du microcontrôleur ATxMega128a1u était qu'il possède 7 UART série, qui devait chacun contrôler une patte du robot.

Ce microcontrôleur devait se connecter en USB, sur un PC, ou autre, et fournir une API de plus haut niveau, pour contrôler les mouvements de base du robot.



Finalement, cette solution n'a pas été retenue, pour plusieurs raisons :

La conception de notre carte sur deux étages impliquer deux types de programmation, une pour le microcontrôleur, et une autre pour le Raspberry Pi. Si on se place du point de vue de l'utilisateur qui voudrait rajouter (implémenter) un mouvement, cette structure lui compliquerait l'accessibilité et la lisibilité. Sans compter que l'utilisateur qui vient d'investir (1200€) dans ce produit ne dispose pas forcément de carte de développement ATxMega vendue 70 €.

Le faible apport qu'apporterai cet élément ne justifie pas son utilisation sachant que le Raspberry Pi est largement capable de coordonner le robot.

Cette étude a été analysée car elle correspondait à notre première idée de développement.

### e) Etude 2 : Solution avec uniquement un Raspberry Pi

La deuxième étude se focalise sur l'utilisation du Raspberry Pi, pour commander les servomoteurs.

Cette solution se veut plus simple que la première. En effet, il suffit de créer une carte d'extension avec les composants d'adaptation, que nous aurions utilisée avec l'ATxMega. Nous avons donc réutilisé toute l'étude sur les composants périphériques de l'étude 1.

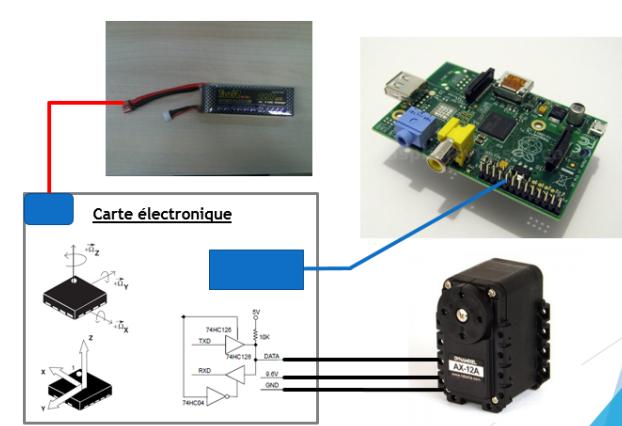


Figure 6 : Représentation interconnexions

Voici le schéma fonctionnel :

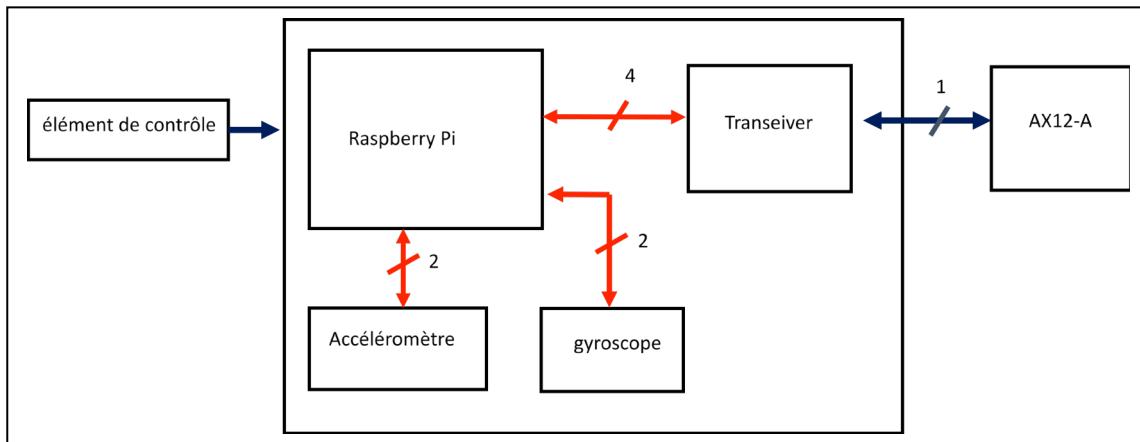


Figure 7 : Schéma fonctionnel

### f) Raspberry Pi

Pour remplacer l'ArbotiX, nous avons choisi d'utiliser un Raspberry Pi. Dans un premier temps nous allons décrire la structure de la carte et dans un second temps nous justifieront son utilisation adaptée à notre projet.

#### Présentation Raspberry Pi :

Cette carte électronique nommée « Raspberry Pi » (Rpi) est un ordinateur de petite taille (85,60 mm × 53,98 mm × 17 mm) sur lequel on peut venir brancher un certain nombre de périphérique. Le Rpi est équipé d'un processeur ARM cadencé à 700MHz possédant une mémoire vive de 256MB (pour notre modèle) et permet l'exécution de plusieurs variantes du système d'exploitation libre GNU/Linux et des logiciels compatibles.



Figure 8 : Raspberry Pi model A

La puce ARMv6, est une technologie très répandue dans le domaine de l'application dite embarquée sûrement pour une question d'opérabilité. Elle Simple à mettre en œuvre et a l'avantage de consommer peu grâce à sa technologie IEM (Intelligent Energy Management).

Il s'agit d'une puce SoC (System on Chip), qui n'embarque pas seulement une unité de calcul mais aussi d'autres coprocesseurs et mémoire qui améliore son fonctionnement. On note donc sur le Raspberry Pi l'absence de mémoire RAM, de mémoire Flash, l'absence d'un Processeur Graphique et de tout autre processeur d'Entrée/Sortie. Effectivement, tout est intégré dans la puce ARM.

Le premier coprocesseur le VFP (Vector Floating Point) permet à la puce d'effectuer des calculs de nombre à virgule flottante de manière bien plus rapidement que si il avait fallu traiter la compression des nombres cotés logiciels pour les calculer avec le processeur de base. Ce coprocesseur améliore donc la rapidité des applications qui traitent d'énorme quantité de nombre à virgule.

Le VideoCore est le deuxième coprocesseur qui gère l'affichage vidéo en Full HD 1080p il servira lors du traitement de la vidéo de la caméra.

Le deuxième élément le plus important dans une architecture informatique, la Mémoire RAM, est présent sur le système pour stocker toutes les données volatiles que notre programme au cours de leur exécution. Un programme que l'on exécute se loge immédiatement en mémoire RAM.

La mémoire flash accessible par le biais du lecteur de Carte SD va nous permettre de stocker le Système d'Exploitation. C'est sur cette carte que le boot loader doit être.

L'alimentation du Raspberry Pi peut être assurée par le port Micro-USB. Dans notre cas, nous laisserons le choix de l'alimenter par le biais du port GPIO ou par le port USB.

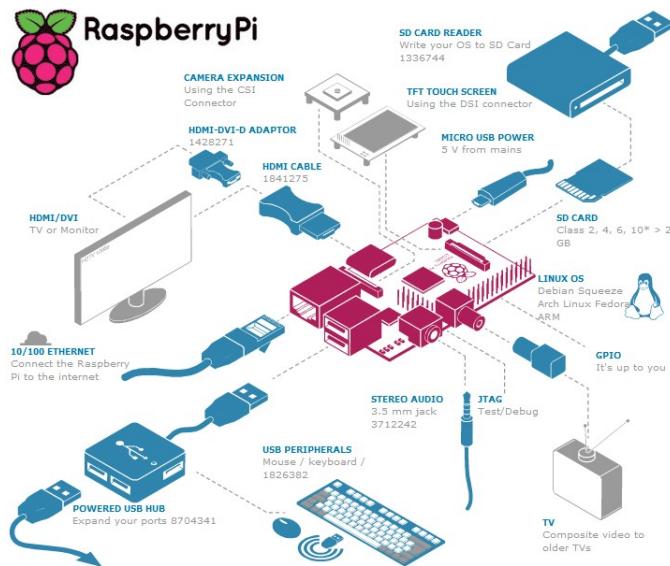


Figure 9 : Accessoires Raspberry Pi

Le Raspberry pi possède, en plus des connectiques classiques USB, HDMI, etc... Un connecteur GPIO. (General Purpose Input Output) qui sont des entrées/sorties.

Ce connecteur GPIO dispose de différents types de connexion :

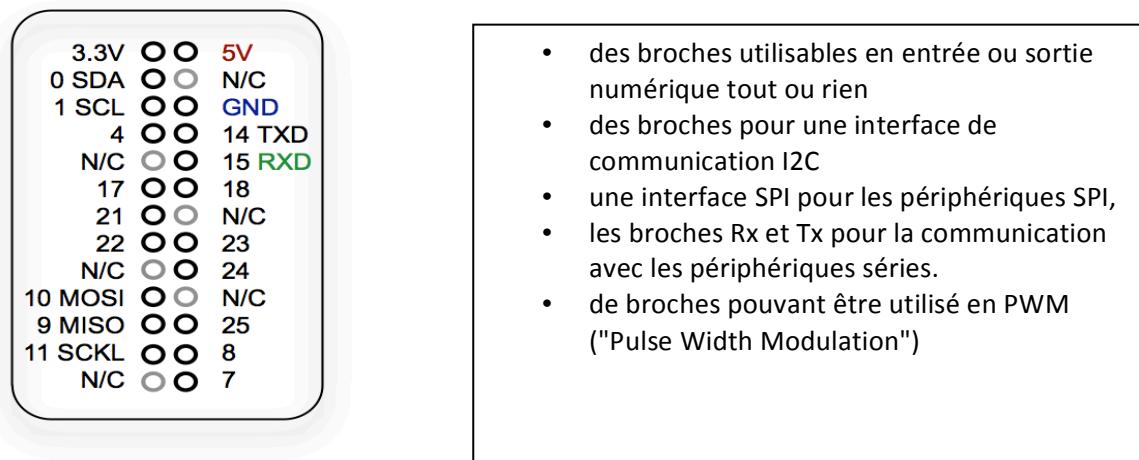


Figure 10 : Schéma GPIO Raspberry Pi

### g) Bilan des solutions

#### 1) Point de vue énergétique

Composants	Solution du Kit	Solution étude 1	Solution étude 2
ArbotiX	2.85 W		
ATxMega		0.2 W	
Raspberry Pi model B		2.89 W	2.89 W
Motion Sensors		0.04 W	0.04 W
Wifi / Bluetooth		1.02 W / 0.33 W	1.02 W / 0.33 W
XBee	0.6 W		
Total	3.45 W	4.15 W / 3.46 W	3.95 W / 3.26 W

Les tests ont été réalisés avec un testeur de puissance à partir de la prise EDF. Nous avons utilisé la carte ArbotiX du kit, et un Raspberry Pi model B avec un camera Pi.

Nous avons aussi utilisé un ampèremètre pour valider la puissance de la carte ArbotiX, pour ne pas prendre en compte le transformateur en amont.

On peut voir que le Raspberry Pi consomme moins que la carte ArbotiX, avec la solution retenue.

Cela peut s'expliquer avec le régulateur de l'ArbotiX, qui doit être de très mauvaise qualité pour diminuer les coûts.

Suite à ce tableau, on peut se demander néanmoins, si la contrainte de consommation est forte sur la commande, en prenant en compte les moteurs.

Les moteur AX-12 consomment 50mA au repos, et 900mA en mouvement chacun.

Ce qui nous donne une puissance de 10W au repos et 180W en mouvement.

On peut donc dire que la consommation en moyenne de la carte de contrôle est négligeable devant la puissance devant être fourni aux moteurs.

## 2) Point de vue financier

Le robot « PhantomX AX Hexapod Mark » est conçu par la société TROSSEN ROBOTICS. De ce fait, cette entreprise est le seul distributeur de ce produit et son coût est important : 1232.09€ (frais de port compris)

Ce prix est en partie dû au tarif des servomoteurs qui coutent 39€ pièce. Ce tarif peut paraître trop important pour un tel actionneur, mais si on observe le marché, on se rend compte que le prix n'est pas si prohibitif.

Par exemple, un servomoteur Hitec d'entrée de gamme comme le HS-422 coûte dans les 16 Euros, mais un servomoteur Hitec de ce prix ne vous fournira qu'un couple de 4 kg.cm environ, un débattement de 180°, aucun retour d'information, l'utilisation d'un fil de commande par servomoteur, et aucun paramétrage possible.

Si on monte en gamme chez Hitec pour avoisiner le prix de l'AX-12A, on tombe sur le HS-5496MH, qui dispose d'un couple de 7.2 kg.cm (*c'est toujours inférieur à l'AX12, même en alimentant celui-ci seulement en 7V*) et nous sommes de toute manière toujours dans une commande de type analogique et sans feedback.

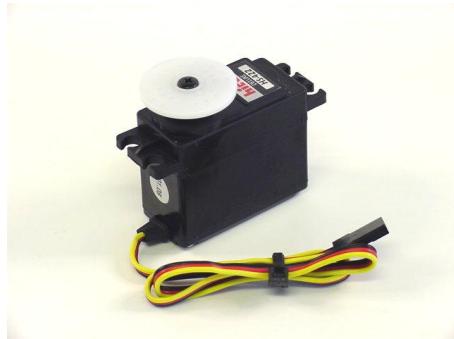


Figure 11: HS-422



Figure 12: HS-5496MH

Un autre avantage important de l'AX-12, si l'on compare le servomoteur analogique qui se situe au niveau mécanique : les multiples possibilités de fixation lui donnent une mobilité incomparable sur la structure dans laquelle il s'insère, ainsi que la précision des assemblages résultants.

Enfin, Concernant Le palonnier de sortie, son accouplement à l'axe est très rigide et limite les risques de désolidarisations avec la structure qui l'entoure.

Pour conclure sur le choix de la base mécanique et ses actionneurs, même à prix égal avec d'autre marque présent sur le marché, nous pensons que le prix est justifié et nous permet de travailler avec du matériel modulaire, flexible et totalement paramétrable.

Dans un second temps nous allons analyser les coûts matériels de nos modifications

Voici un tableau récapitulatif des coûts matériels électroniques

Composants	Solution du Kit	Solution étude 1	Solution étude 2
ArbotiX	39.99 €		
Raspberry Pi		25.99 €	25.99 €
ATxMega		5.47 €	
Régulateurs		1 €	0.41 €
Connecteurs		4.32 €	2.67 €
Fourniture ext.		29.33 €	16.42 €
Bascule		51.78 €	8.63 €
Motion Sensors		15.17 €	15.17 €
Hub connecteur	3.56 €		
XBee	43.9 €		
Wifi		7.50 €	7.50 €
Total	87.45 €	140.56 €	76.79 €

Si l'on regarde le prix, la solution retenue se situe dans la même gamme de prix que l'ArbotiX. Notre solution semble donc tout à fait réaliste avec un ajout de fonctionnalité non négligeable.

#### h) Comparaison XBee / Bluetooth / Wifi

Le module XBee présent dans le kit, servira de base. Nous prendrons des modules qui se rapprocheront tant que possible de ce module.

	Bluetooth (class II)	XBee	Wifi n
Consommation	330 mW	600 mW	1020 mW
Portée théorique	50 m	100 m	250 m
Débit	3.0 Mbit/s	250 kbit/s	150 Mbit/s

A la suite de ce tableau, notre choix c'est porté sur le wifi. En effet, la consommation étant négligeable devant les moteurs, on peut choisir le wifi qui permet d'avoir aussi un débit supérieur pour pouvoir transporter de la vidéo issu de la webcam.

Dans leur gamme, XBee fourni des modules plus puissant avec un meilleur porté. Mais ce choix est très limité. En effet, si l'on veut contrôler le robot avec un ordinateur, on peut utiliser le wifi ou Bluetooth directement, alors que le XBee nécessite forcement une carte d'adaptation.

## 4) Réalisation

### a) Commande et montage de la base mécanique

Le seul distributeur de ce modèle est TrossenRobotics qui est une société basé aux l'Etat Unis. Etant donné que cette entreprise n'est pas présente dans la base fournisseur de Polytech'Tours La commande de la base mécanique a posé quelques problèmes.

Après avoir étudié les alternatives de commande misent à notre disposition, nous avons fait le choix de travailler avec la société « Génération robot » qui nous servi d'intermédiaire pour commander notre produit. Malgré notre suivi rigoureux, et des contacts réguliers nous n'avons reçu le robot qu'en semaine 47 juste avant de partir en entreprise.

Ce kit comprenait les 18 servomoteurs ainsi que l'Hexapode à assembler.

Voici le kit après le montage :



Figure 14 : Kit avant assemblage



Figure 13 : Kit assemblé

Nous avons passé une journée de 8h pour monter le robot (assemblage mécanique et électrique) et une autre journée de 8h pour l'installation du programme arbotix dans le robot.

L'installation a duré beaucoup de temps car il a fallu faire l'identification des moteurs un par un, programmer le système Xbee entre la télécommande et le robot. De plus, il y avait des erreurs dans le code du robot que nous avons corrigé afin d'obtenir un fonctionnement correct.

### b) Conception du schéma électrique

Après avoir abandonnée la version avec l'ATxMega, nous avons voulu réaliser une carte d'extension pour le Raspberry pi.

Nous sommes donc partis du schéma du GPIO du Raspberry Pi (Figure 10), pour y placer les éléments que nous souhaitions.

Nous avons utilisé le port I2C du GPIO, pour ajouter l'accéléromètre, et le Gyroscope.

Et nous avons ajouté le transceiver après le RX/TX.

Une alimentation 5V a aussi été rajoutée pour alimenter le Raspberry Pi grâce au GPIO.

## Photo schémas électriques

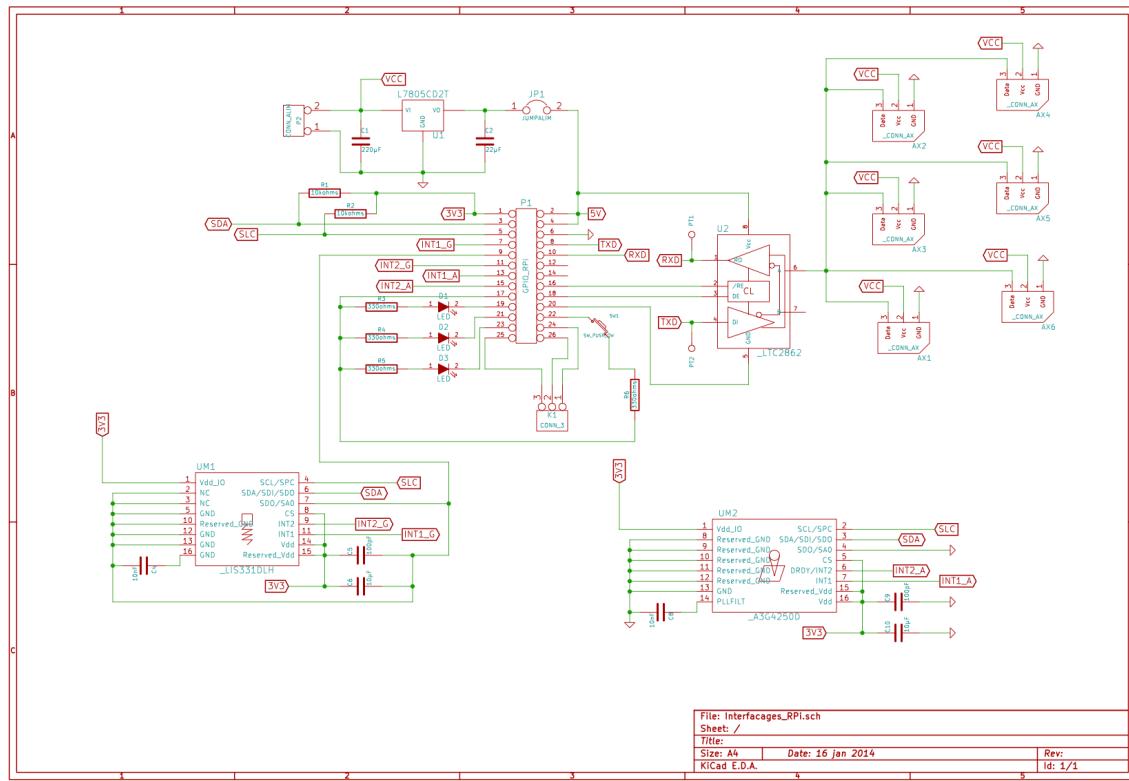


Figure 15 : Schéma électrique

## c) Conception du typon

Le typon a été réalisé afin de limiter les effets de CEM notamment avec la différence des pistes entre les composants discrets et les CMS.

Nous avons choisis d'utiliser des composants CMS, afin de permettre l'intégration de notre carte dans la base robotique.

On arrive à une petite carte de 64x52mm.

Voici le typon de cette carte :

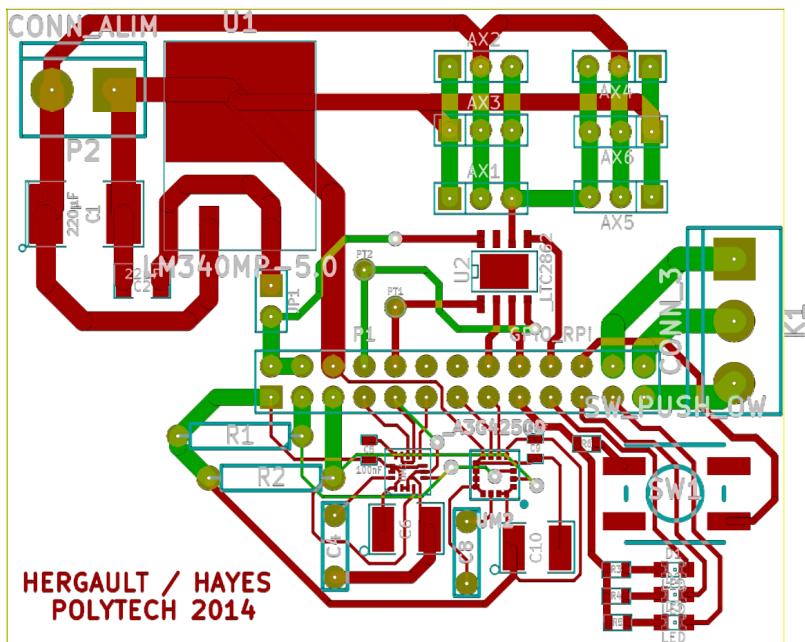


Figure 16 : Typon carte électronique

#### d) Confection de la carte électronique

La carte a été réalisée par la société EuroCircuit :

Voici la carte terminée :



Figure 17 : Carte finale

## 5) Test de fonctionnement

Afin de valider le fonctionnement de notre carte, nous avons testé tout d'abord l'alimentation 5V de celle-ci.

Une fois celle-ci validé, nous avons essayé de brancher un Raspberry, pour tester si la carte l'alimentait correctement.

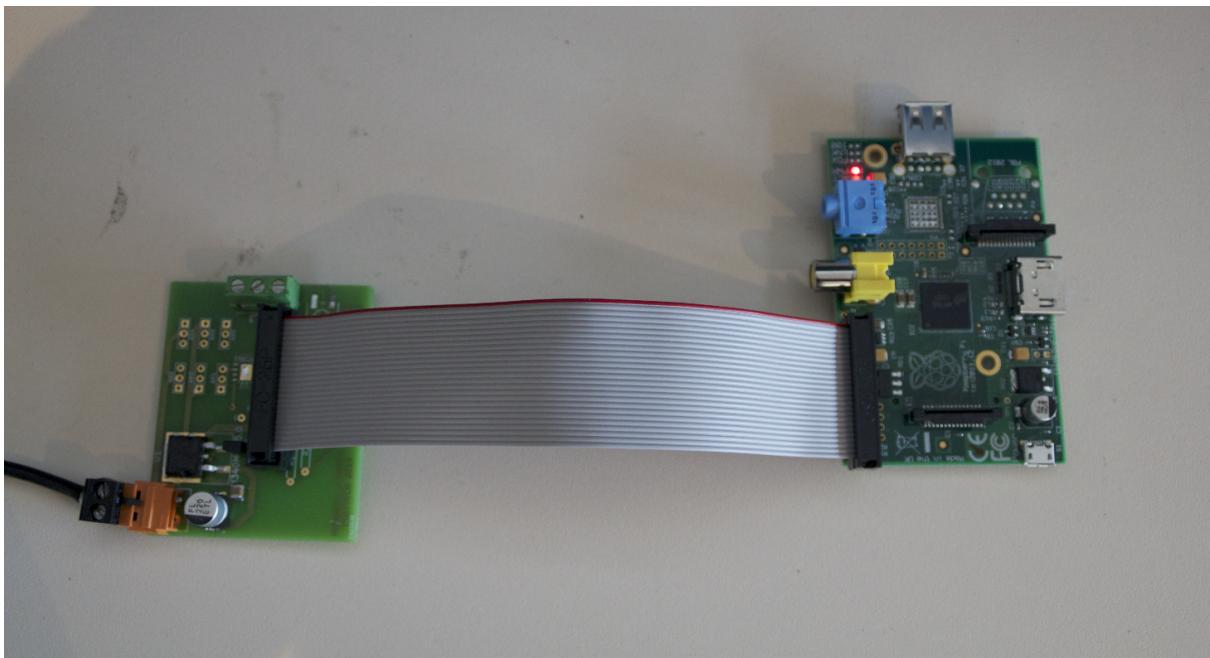


Figure 18 : Test de l'alimentation du Raspberry Pi par le GPIO

Par manque de temps, nous n'avons pas pu tester le gyroscope et l'accéléromètre. Ils seront donc testés dans un second temps pour le projet informatique.

Ayant commandé, un transceiver avec la mauvaise empreinte, nous avons essayé de commander les moteurs sans succès.

Pourtant la norme TTL des moteurs devrait prendre le 3,3V du Raspberry Pi, car le niveau de tension de l'état haut minimum du TTL est de 2,8V.

## 6) Organisation projet

Après avoir pris connaissance des délais de livraison concernant la base existante auprès de nos fournisseurs, nous avons réalisé le planning prévisionnel pour définir sous forme de macro étapes les grandes phases de notre projet. Celui-ci va nous permettre de définir la structuration de notre projet, de faire la planification des délais pour organiser la réalisation de la carte électronique.

Cette étape est indispensable pour suivre et piloter notre projet.

Tâches	Semaines N°																
	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3
Etude de la documentation robot	■	■	■														
définir les besoins		■	■	■													
réalisation schéma structurel					■	■	■										
Etude de la documentation matérielle						■	■	■									
Etude énergétique								■	■	■							
Etude financière								■	■	■							
réalisation schéma électronique										■	■	■					
commande du robot	■	■	■	■													
Commande matériel									■	■							
Réalisation typon											■						
Implantations composants												■					
Phase de test													■	■			

Voyons maintenant le planning réel :

Tâches	Semaines N°																
	39	40	41	42	43	44	45	46	47	48	49	50	51	52	1	2	3
Etude de la documentation robot	Green	Green	Green	Green	Green												
définir les besoins	White	Blue	Blue	Blue	Blue	Blue	White	Blue	Blue	Blue	Blue	Blue	Blue				
réalisation schéma structurel					Red	Red	Red	Red				Red	Red	Red			
Etude de la documentation matérielle						Green	Green	Green	Green								
Etude énergétique														Blue	Blue	Blue	
Etude financière														Blue	Blue	Blue	
réalisation schéma électronique									Green	Green				Green	Green		
commande du robot	Red	Red	Red	Red	Red	Red	Red	Red	Red								
Commande matériel						Green	Green							Green			
Réalisation typon															Blue		
Implantation composants																Red	
Phase de test																Green	

Le planning réel est assez différent de notre planning prévisionnel, et cela pour plusieurs :

- La commande de la base mécanique est arrivée beaucoup plus tard que prévu.
- L'étude de la base existante nous a pris plus de temps car cette phase est tombée pendant une période dense en enseignement et travail personnel.
- Au cours de l'étude, nous avons décidé de changer architecture fonctionnelle. En effet, ce projet est orienté open source et la première étude ne répondait pas à cette attente. Ce constat nous confirme que le système à mettre en place doit être accessible afin que l'utilisateur puisse développer de futur mouvement du robot.
- L'attente de la carte et ce changement d'architecture nous ont obligés à faire l'implantation des composants et la phase de test dans la dernière semaine.

## 7) Conclusion

Nous avons décidé de réaliser l'étude de ce projet en définissant la catégorie de personne qui serait en mesure d'acheter ce produit. En effet, quel genre de personne serait intéressé pour acquérir ce robot ? Et quelles sont leurs attentes ?

L'étude réalisée nous a permis de définir l'orientation à donner à ce projet et aussi à prendre en compte la partie développement qui sera présenté dans un autre rapport.

Nous avons proposé un système qui soit cohérent par rapport à un produit commercialisable en arrivant à une structure moins couteux avec architecture plus évoluer et plus puissante.

Notre alternative est plus modulable que la carte ArbotiX et permettrait d'accroître les capacités du robot en y rajoutant de nouveau périphérique, voire même de nouveau actionneur pour créer des pinces (par exemple).

Notre architecture Raspberry Pi est open source et donne à l'utilisateur une accessibilité accrue. En effet il pourra ajouter et modifier le programme mise en place pour définir une nouvelle classe de mouvement ou encore lancer un mode automatique ou le robot pourra enchaîner plusieurs figures.

La question concernant l'abandon du microcontrôleur aurait dû se faire au début du projet pour assurer la livraison du produit fini. Cette remise nous a fait perdre beaucoup de temps et nous avons réalisé les dernières phases du projet dans la dernière semaine avant la restitution. Ce qui nous a laissé trop peu de temps pour les phases de tests.

Mais je pense qu'il été indispensable de faire ce changement afin de proposer un projet réutilisable.

Ce projet nous a apporté de l'expérience dans la gestion de projet, gérer les délais de livraison et d'étude de la documentation du matériel afin de garantir un produit livrable.

Nous avons consolidé nos relations humaines en prenant contact avec des fournisseurs pour demander des conseils ou des devis.

Enfin, nous avons amélioré nos connaissances en électronique, en travaillant avec du matériel performant à l'image des moteurs AX12-A et consolidé notre démarche ingénieur en réalisant une étude complète et enrichissante sur le robot HEXAPOD.



## 8) Table des illustrations

Figure 1 : Ligne I2C .....	13
Figure 2 : Accéléromètre.....	14
Figure 3 : Gyroscope .....	15
Figure 4 : Schéma gyroscope.....	16
Figure 5 : Schémas Transceiver.....	17
Figure 6 : Représentation interconnexions .....	18
Figure 7 : Schéma fonctionnel.....	19
Figure 8 : Raspberry Pi model A .....	19
Figure 9 : Accessoires Raspberry Pi.....	20
Figure 10 : Schéma GPIO Raspberry Pi.....	21
Figure 11: HS-422                                  Figure 12: HS-5496MH .....	22
Figure 14 : Kit avant assemblage.....	24
Figure 15 : Schéma électrique.....	25
Figure 16 : Typon carte électrique .....	26
Figure 17 : Carte finale .....	26
Figure 18 : Test de l'alimentation du Raspberry Pi par le GPIO .....	27