

Projet Technologique : Robot humanoïde



20 mars 2015

Créé par : Sylvain Ravard

Tuteur : Vincent Barreau

Projet Technologique : Robot humanoïde

Sommaire

GLOSSAIRE	2
INTRODUCTION	3
OBJECTIFS	3
ETAT DE L'ART	3
ROBOT POPPY.....	3
ROBOT NIMBRO-OP.....	4
ROBOT NAO.....	4
ROBOT INMOOV	4
PARTIE MECANIQUE	5
IMPRESSION	6
MONTAGE	7
PARTIE ELECTRONIQUE	9
BEAGLEBONE BLACK REV C	9
GPIO	10
PWM	10
UART.....	11
CONNEXION A INTERNET.....	12
AX12	12
CONVERSION 3.3 V A 5V	13
GESTION DU FULL DUPLEX TO HALF DUPLEX	13
MONTAGE ELECTRONIQUE	15
FONCTIONNEMENT DES REGISTRES	15
DEBUGUE.....	16
CONCLUSION	17
BIBLIOGRAPHIE	18

Glossaire

Open source : Libre de droit

Open hardware : Matériel libre de droit

Beaglebone : carte électronique permettant de contrôler le robot

Servomoteurs : moteurs contrôlable numériquement

Boot : démarrage d'un système d'exploitation sur une carte électronique avec système embarqué ou un ordinateur

Arduino Mega : carte électronique permettant de contrôler le robot

Bus Half duplex : canal de transmission numérique capable de communiquer dans plusieurs sens à la fois avec un/des émetteur(s) et un/des récepteur(s).

Bus Full duplex : canal de transmission numérique capable de communiquer dans un seul sens à la fois avec un émetteur et un ou plusieurs récepteur(s).

Introduction

Le FabLab de Lannion est une association dont l'objectif est d'aider les personnes qui le souhaite à réaliser leurs projets technologiques en leur donnant des conseils et accès à des outils spécifiques.

Ce projet technologique organisé à l'ENSSAT en partenariat avec le FabLab de Lannion a pour objectif de créer un robot humanoïde. La création de ce robot humanoïde est un moyen de montrer les capacités du FabLab de Lannion et donne les moyens à l'ENSSAT d'expérimenter une solution technologique dans le monde de la robotique.

Dans l'objectif de réaliser un maximum de travail dans le temps imparti, l'ENSSAT c'est associé avec le FabLab pour réaliser ce projet.

Objectifs

L'objectif est de créer une plateforme robotique open source et open hardware. Ce support matériel permettra à l'ENSSAT de réaliser différents projets technologiques en rapport avec le monde humanoïde.

La réalisation complète d'un robot humanoïde est un projet trop important pour être réalisé complètement en 6 mois, nous avons choisi de ne pas partir de zéro. De nombreux robots humanoïdes ont déjà été développés et sont disponibles librement. La solution la plus évidente est d'utiliser une base existante pour gagner du temps.

Etat de l'art

Afin d'effectuer le meilleur choix nous avons étudié plusieurs plateformes robotiques qui pouvaient répondre à notre besoin.

Robot Poppy

Poppy est un robot humanoïde totalement open source (hardware y compris) et peut entièrement être construit sur une imprimante 3D. Son buste mobile le rend assez souple pour la marche. Ce robot n'est pas autonome, un ordinateur sert de microcontrôleur (via USB) et l'alimentation est externe sur le modèle de base. Les deux caméras PS eye permettent à Poppy de reconnaître son environnement et interagir avec lui. De nombreuses variantes de ce robot ont déjà été créées sur le site <https://forum.poppy-project.org>.

Robot NimbRo-OP

NimbRo-OP est un robot humanoïde de quasiment 1 mètre open source (y compris hardware). Ce robot conçu initialement pour jouer au football dispose de jambes puissantes (6 servomoteurs MX-106 par jambes). Ce robot dispose comme la plupart des humanoïdes d'un accéléromètre, gyroscope et caméra pour se repérer et d'un processeur dual core avec une carte wifi et une batterie pour son autonomie. Le prix de ce robot en aluminium coûte environ 20 000 € à cause de ses servomoteurs puissants.

<http://www.nimbro.net/OP/>

Robot Nao

Le Nao est un robot humanoïde autonome open source (seulement pour la partie software) connu de tous. Son design et son équipement très évolué (zones tactiles sur les mains et la tête et bumpers aux pieds) le rend très intéressant mais aussi très peu modulable car sa partie hardware est sous licence. Ce robot n'est donc pas utilisable pour notre projet.

<https://www.aldebaran.com/fr>

Robot Inmoov

Ce robot semi-humoïde (homme-tronc) open source (y compris hardware) est constructible à l'aide d'une imprimante 3D. Malgré son manque de mobilité ce robot a l'avantage d'être très habile grâce à ses mains composées de 5 doigts. L'ensemble des servomoteurs dans le cou et les bras permettent à Inmoov de regarder et de suivre un objet du doigt. Pour rendre le système moins fragile et plus puissant, les doigts sont articulés par des moteurs situés dans les avant-bras du robot. L'ensemble du robot est piloté par deux arduino Mega et possède un système de caméra et micro pour pouvoir communiquer. L'équilibre n'étant pas un problème pour ce robot (car homme-tronc) le coût a donc pu être réduit (de 1000 à 1300€) en mettant des servomoteurs moins performants.

Nous avons choisi le robot Inmoov car il présente plusieurs avantages parmi ses concurrents. Tout d'abord son prix aux alentours de 1000€ a été un critère très important car nous voulions réaliser un maximum du robot avec l'argent accordé par l'ENSSAT (environ 300€). De plus sa conception simple (autant sur le plan mécanique que sur le plan électronique) nous a permis de la personnaliser (dans notre cas l'améliorer) pour qu'il réponde au mieux à nos attentes. Enfin la communauté déjà

importante autour de ce robot et la rencontre de l'inventeur de ce robot (Gaël Langevin) qui a pu nous donner de nombreux conseils a été déterminant dans le choix de cette base robotique.

Une fois que la base robotique (dans notre cas le Robot Inmoov) et les différentes solutions technologiques ont été choisies (choix de la carte électronique, moteurs, ...) nous pouvons passer à l'étape de conception et de fabrication.

Dans cette étape nous avons dû développer la partie électronique et la partie mécanique. La première difficulté que nous avons rencontré est que les différents choix que nous avons fait pour améliorer la partie électronique nous a obligé à revoir la partie mécanique et cela essentiellement pour des contraintes de volume (les nouveaux moteurs ne tiennent pas dans les bras du robot).

Pour pouvoir réaliser la partie mécanique qui est majoritairement l'impression en 3D, nous avons dû nous associer au FabLab qui à l'équipement et les connaissances nécessaires pour cette partie. D'ailleurs l'orientation technologique de départ s'est faite en fonction des équipements du FabLab car dès le début du projet les membres de cette communauté avaient déjà manifesté leur intérêt pour ce projet. Le projet se déroule donc en tandem entre le FabLab pour la partie mécanique et le travaille à l'ENSSAT pour la partie électronique.

Partie Mécanique

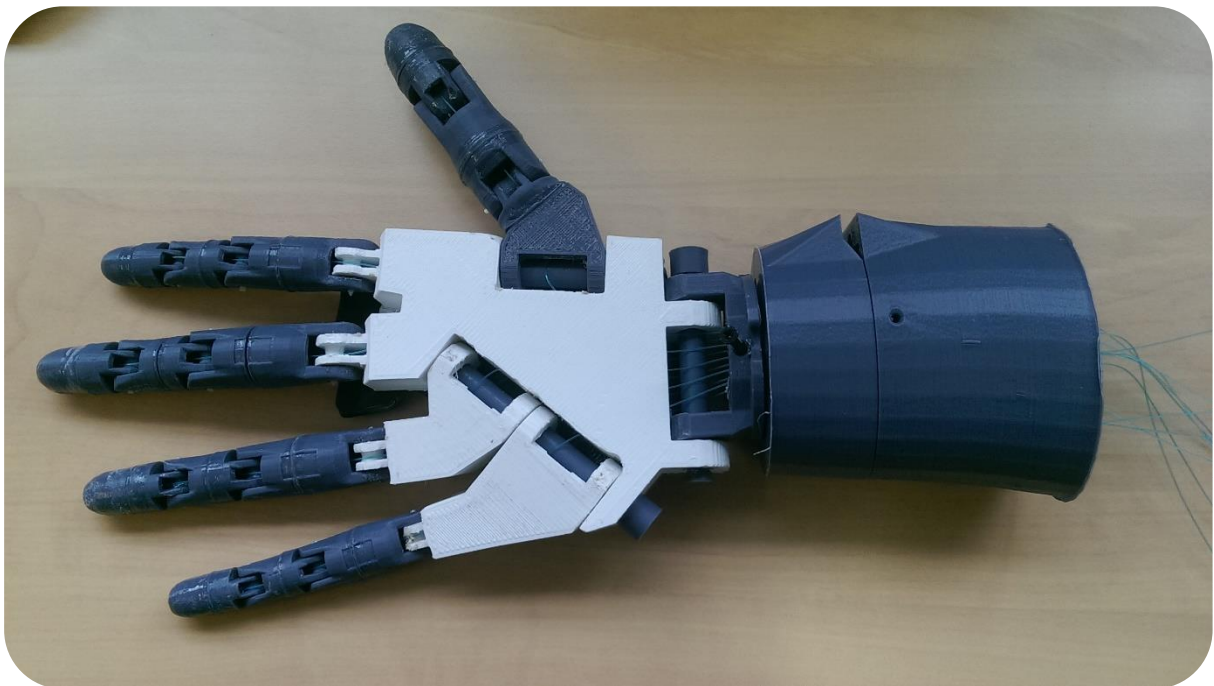


IMAGE 1

Chaque doigt de la main dispose de 3 articulations et l'annulaire et l'auriculaire dispose d'une articulation supplémentaire. Pour pouvoir motoriser l'ensemble des doigts nous avons passé 2 fil (un pour « ouvrir » chaque doigt de la main et l'autre pour « fermer » chaque doigts de la main). Les deux fils visibles sur l'image 2 sont commandés par des moteurs situés dans l'avant-bras du robot.

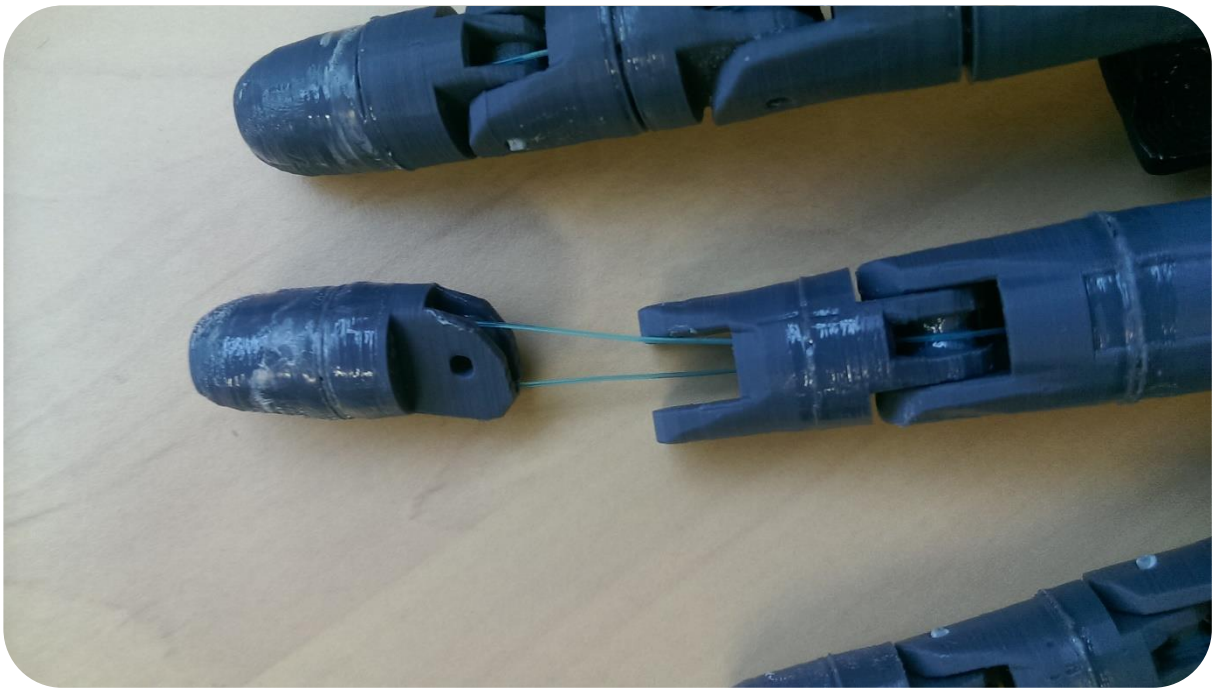


IMAGE 2

Impression

Nous avons réalisé la main droite et le poignet de du robot humanoïde à l'aide des fichiers .stl fournis sur le site inmoov.fr (<http://www.inmoov.fr/download/>). La réalisation de la main gauche est identique à celle de la main droite (certains fichiers stl sont un peu différents).

La construction commence par les phalanges. Tous les doigts sont composés de 6 demi-phalanges que l'on peut imprimer en même temps. Pour éviter les problèmes d'impression il est conseillé d'imprimer les phalanges une par une (ou deux par deux pour gagner du temps) afin d'éviter que les impressions soit trop longues car plus l'impression prend du temps plus la pièce risque de se décoller.

Pour modifier les fichiers .stl récupéré sur le site Inmoov, il est conseillé d'utiliser google sketshup (logiciel gratuit). Avant d'utiliser ce logiciel il ne faut pas oublier de télécharger l'extension STL Import & Export (extension gratuite) et de l'ajouter au projet sketchup dans « Fenêtre->Préférence->Extension->Installer l'extension » et ajouter le fichier. L'interface de Sketchup est simple il suffit d'importer le fichier stl de le modifier en supprimant les phalanges que l'on ne veut pas et exporter le fichier stl.

Avant l'impression il faut utiliser le logiciel disponible au FabLab pour convertir les fichiers stl en fichier imprimable (pour les premières impressions les membres du FabLab expliqueront le fonctionnement de l'imprimante 3D de manière plus précise).

Toutes les phalanges ont été imprimant à un taux de 100% (impression très longue par rapport au volume) et les pièces plus massives (intérieur de la main et poignet) à 70% à peu près.

Montage

Une fois que toutes les pièces sont imprimées nous pouvons commencer l'assemblage.

La première étape consiste à coller l'ensemble des demi phalanges ensemble avec de l'acétone sauf les ongles qui seront collé en tous dernier. Pour le collage à l'acétone il suffit de tremper au moins une surface avant de maintenir les deux pièces ensemble pendant quelques secondes. Certaines surfaces de collages peuvent être déformé et arrondi dans ce cas il faut limer la surface.

Maintenant que les pièces sont collées on peut mettre toutes les pièces en position et limer toutes les surfaces qui empêchent les articulations de bouger sans frottement (surface en rouge sur l'image 3).

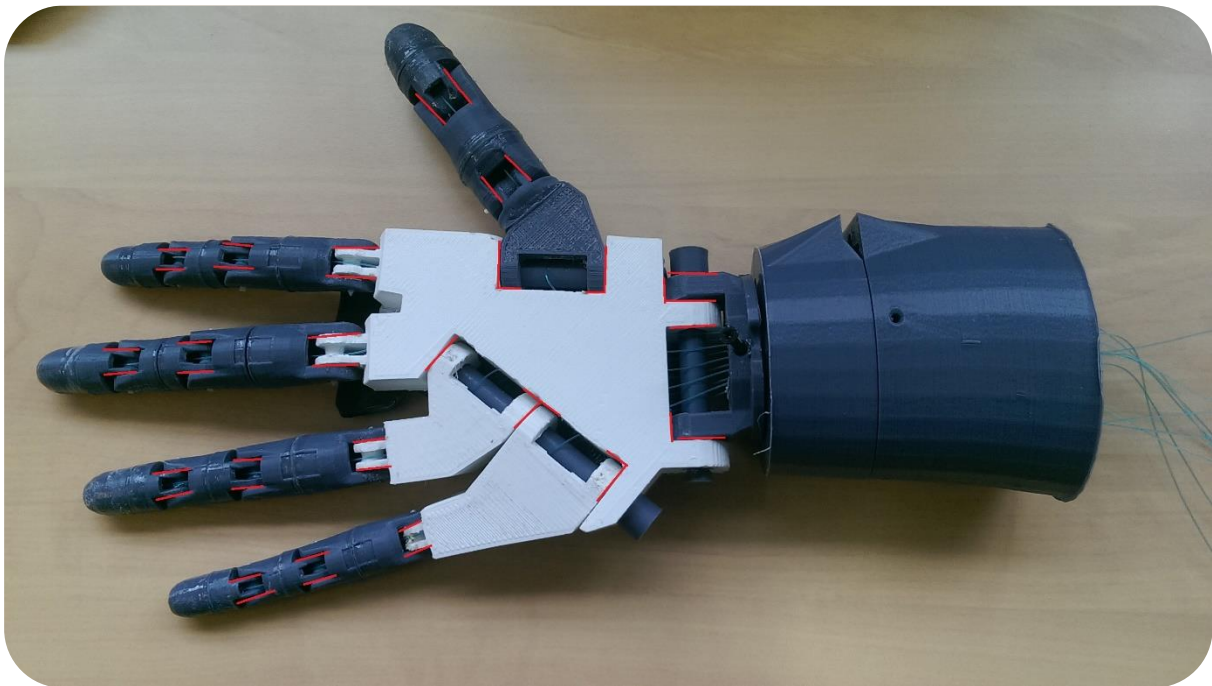


IMAGE 3

Nous pouvons passer à l'étape suivante qui est la mise des fils d'une longueur de 75cm dans l'ensemble des doigts (2 fils par doigts), de la main et du poignet. On remarquera que les trous nécessaire au passage des fils dans la main et poignet est déjà faite. Il faut ensuite attacher les deux fils de chaque doigt au niveau des ongles et refermer les ongles à l'acétone.

En cas de problèmes (si un fil casse ou s'il y a une erreur de montage) il est toujours possible de démonter les ongles à l'aide d'un cutter en tamponnant la pièce que l'on souhaite couper à l'acétone.

Enfin nous pouvons ajouter les « tiges » des pivots de tous les doigts afin de solidariser la main avec du fil d'ABS (le même plastique que pour la création de la main) que l'on peut bloquer en l'humidifiant avec de l'acétone. En cas de problèmes (si l'articulation est bloquée ou non fluide) on peut toujours évacuer la tige en l'humidifiant avec de l'acétone et en tirant et poussant l'articulation en même temps comme sur l'image 4.



IMAGE 4

La tige au milieu de l'articulation est beaucoup plus faible que le doigt manipulé il n'y a donc pas de risque de casser le doigt.

L'utilisation des servomoteurs AX12 nous empêche de réutiliser les avants bras fournis tel-quel. Il a donc fallu repenser le support disposé dans l'avant pour tenir les moteurs.

Nous avons donc pensé à deux types de support représenté sur les figures suivantes :

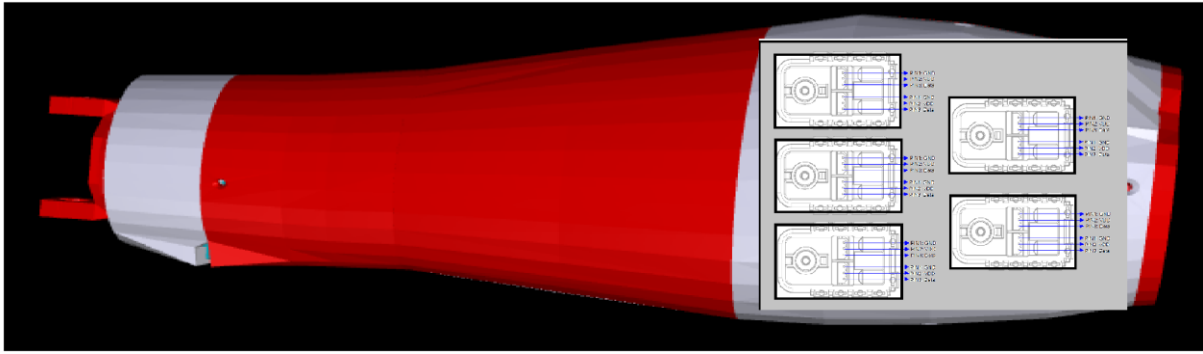


FIGURE 1 SUPPORT DES 5 MOTEURS DE LA MAIN

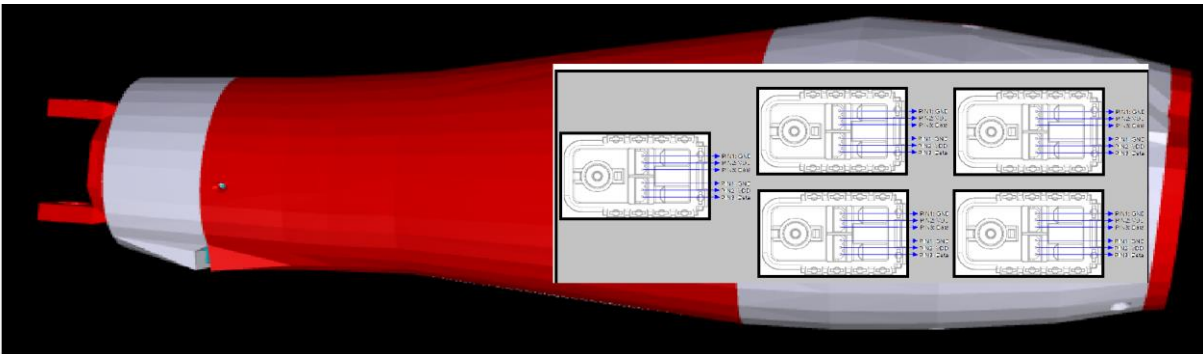


FIGURE 2 SECOND SUPPORT DES MOTEURS DE LA MAIN

Les 5 moteurs de la main seront fixés (vissé de préférence) sur ce support et ce même support sera fixé sur la surface de l'avant-bras. Ce support peut être fait en différents matériaux (bois, aluminium, plastic) mais l'impression d'une pièce aussi simple par une imprimante 3D est longue, coûteuse et pas très solide, le mieux serait de demander au FabLab si nous pouvons utiliser leur découpeuse laser.

Nous ne sommes pas obligés non plus d'être obligés de suivre la construction donnée par le site Inmoov pour créer la surface de l'avant-bras, les membres du FabLab ont proposé le moulage plastique de l'avant-bras.

Partie électronique

Beaglebone Black rev C

Pour utiliser la carte il faut vérifier que les pilotes nécessaires sont installés sur l'ordinateur [1].

La version de Debian utilisée sur la Beaglebone est la Debian GNU/Linux 7.8 (wheezy). Pour avoir la version la plus récente disponible pour une beaglebone aller sur : <http://beagleboard.org/latest-images>

La connexion SSH à la carte se fait à l'aide de putty à l'adresse 192.168.7.2. Login : root
L'utilisation de winscp est très pratique pour les transferts de dossier (seulement sur Windows).

L'ensemble de la programmation des moteurs se fait à l'aide de deux scripts python (python version 2.7) `moteurax12.py` pour le test des moteurs ax12 et `moteurpwm.py` pour le test des moteurs PWM.

La Beaglebone dispose d'une centaine de ports pour communiquer avec l'extérieur et la majorité de ces ports sont reconfigurables en fonction de notre utilisation de la carte. Dans ce projet nous utiliserons les ports GPIO, le PWM et l'UART.

Des tutoriels sont disponibles sur le site d'adafruit [2] [3] [4] pour mieux comprendre l'utilisation des broches GPIO, PWM et UART. D'autres liens très utiles sont disponibles sur le site du FabLab [5] pour mieux comprendre le fonctionnement d'une Beaglebone ainsi que d'autres sites [6] ou encore à l'annexe 1.

GPIO

L'utilisation des ports GPIO est très simple, il suffit d'initialiser la broche que l'on veut utiliser avec la commande « `GPIO.setup("PX_YY", GPIO.OUT)` » qui permet d'initialiser la broche en sortie (ou entrée avec `GPIO.IN`) avec X et Y les numéros des broches appartenant au tableau suivant :

65 possible digital I/Os

P9				P8			
GPIO	1	2	GPIO	GPIO	1	2	GPIO
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUTTON	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
GPIO_6	19	20	GPIO_60A	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
GPIO	43	44	GPIO	GPIO_72	43	44	GPIO_73
GPIO	45	46	GPIO	GPIO_70	45	46	GPIO_71

FIGURE 3 ROCHES GPIO UTILISABLE DE LA BEAGLEBONE

Dans ce projet, les pins GPIO permettent de gérer la conversion full duplex to half duplex (plus de précision dans le chapitre Gestion du full duplex to half duplex). Ces broches sont très utiles pour la gestion de périphériques simples (interrupteurs, LEDs, ...) mais il faut faire attention à protéger les broches à l'aide d'un transistor lorsque le périphérique réclame de la puissance et alimenter le circuit avec les broches prévu à cette effet sur la carte ou alors utiliser une alimentation extérieur.

PWM

Pour éviter de refaire la partie mécanique du poignet nous avons décidé d'utiliser des moteurs analogiques qui sont plus petit que les moteurs numériques. Ces moteurs sont facilement commandable à l'aide de la bibliothèque `Adafruit_BBIO.PWM` qui permet de

contrôler le en mettant une consigne à la broche de commande entre 0 et 180 pour un angle entre 0° et 180°. L'ensemble des broches pouvant commander un moteur PWM sont indiqué sur la figure suivante (12 disponible).

8 PWMs and 4 timers

P9				P8			
GGND	1	2	GGND	GGND	1	2	GGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	TIMER4	7	8	TIMER7
PWR_BUTTON	9	10	SYS_RESETN	TIMER5	9	10	TIMER6
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	EHRPWM1A	EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
GPIO_19	19	20	GPIO_60A	EHRPWM2A	19	20	GPIO_63
EHRPWM0B	21	22	EHRPWM0A	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	ECAPPWM2	GPIO_86	27	28	GPIO_88
EHRPWM0B	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
EHRPWM0A	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	EHRPWM1B
AIN6	35	36	AIN5	GPIO_8	35	36	EHRPWM1A
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	ECAPPWM0	GPIO_74	41	42	GPIO_75
GGND	43	44	GGND	GPIO_72	43	44	GPIO_73
GGND	45	46	GGND	EHRPWM2A	45	46	EHRPWM2B

FIGURE 4 BROCHES PWM UTILISABLE DE LA BEAGLEBONE

UART

Pour communiquer avec la majorité des moteurs du robot nous avons choisi d'utiliser le bus de communication UART. Ce bus à l'avantage de contrôler tous les moteurs du robot avec un seul fil car le nombre de moteurs contrôlable par un seul bus est de 255. Il faut cependant éviter de surcharger ce bus car il ne permet pas la communication dans les deux sens (moteurs vers Beaglebone et Beaglebone vers moteurs) à cause d'une particularité des moteurs utilisé (explication fournit dans le chapitre AX12). Il faut donc laisser le bus le plus libre possible pour que les moteurs puissent transmettre leurs états de fonctionnement. La Beaglebone peut gérer 5 bus UART on peut donc répartir les moteur sur les différents bus (un bus par main par exemple).

4 UARTs and 1 TX only

P9				P8			
GGND	1	2	GGND	GGND	1	2	GGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUTTON	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
UART4_RXD	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
UART4_TXD	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
UART1_RTSN	19	20	UART1_CTSN	GPIO_22	19	20	GPIO_63
UART2_TXD	21	22	UART2_RXD	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	UART1_TXD	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	UART1_RXD	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	UART5_CTSN+	31	32	UART5_RTSN
AIN4	33	34	GNDA_ADC	UART4_RTSN	33	34	UART3_RTSN
AIN6	35	36	AIN5	UART4_CTSN	35	36	UART3_CTSN
AIN2	37	38	AIN3	UARR5_TXD+	37	38	UART5_RXD+
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	UART3_TXD	GPIO_74	41	42	GPIO_75
GGND	43	44	GGND	GPIO_72	43	44	GPIO_73
GGND	45	46	GGND	GPIO_70	45	46	GPIO_71

FIGURE 5 BROCHES UART UTILISABLE DE LA BEAGLEBONE

Connexion à internet

dhclient eth0 : Cette commande permet de configurer automatiquement sur la prise Ethernet

L'exécution de cette commande ne fonctionne pas à chaque démarrage de la carte quand le câble Ethernet est relié au pc. Dans ce cas il faut réinitialiser la Gate Way à l'aide de la commande `route add default gw 192.168.7.1`

Pour éviter les problèmes de connexion internet il est conseillé de connecter la beaglebone directement sur le réseau filaire de l'ENSSAT et de contrôler la carte via le câble USB. Les problèmes de partage de connexion internet sont principalement dus au réseau wifi de l'ENSSAT qui ne permet pas « toujours » de partager la connexion internet.

AX12

La motorisation disponible sur le site inmoov.fr est entièrement composée de moteur analogique commandable en PWM. Nous avons choisie d'utiliser des moteurs numériques AX12 pour plusieurs raisons.

Tout d'abord les moteurs AX12 ont l'avantage d'être plus précis (0.029° près) et dispose d'un angle mort (angle dans lequel le servomoteur ne peut pas aller) de seulement 60° au lieu de 180° pour les moteurs MG995m.

De plus l'AX12 est commandé par un bus UART qui permet de limiter le nombre de fils dans le robot. Nous pouvons brancher tous les moteurs à la suite avec un seul montage de la manière décrite sur la figure suivante.

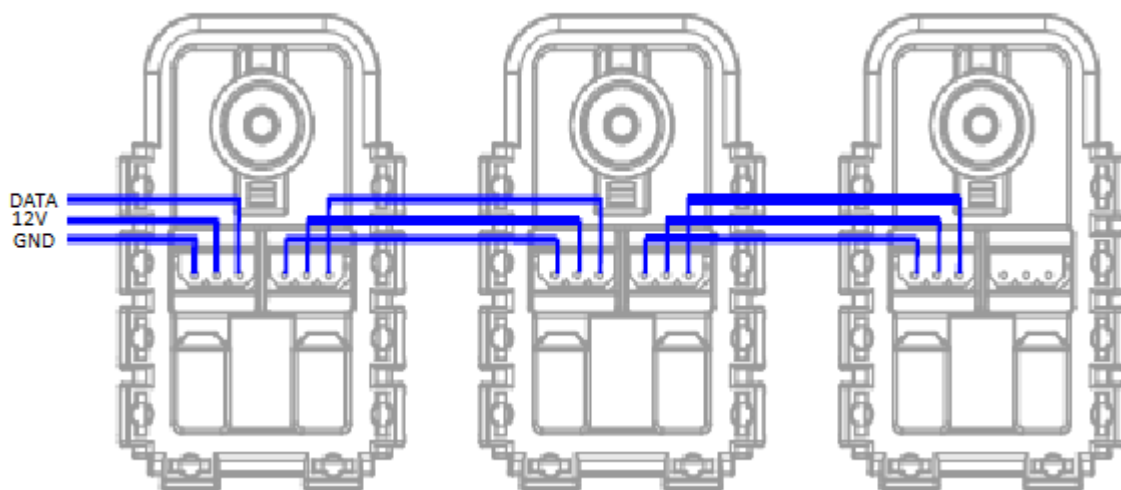


FIGURE 6 MISE EN SERIE DES AX12

L'avantage majeur du servomoteur est qu'il peut fournir des informations sur son état (position réelle, vitesse réelle, couple) qui permet de détecter les collisions et d'arrêter le moteur à temps. Pour une description plus complète du fonctionnement de l'AX12 voir l'annexe 3

Robot humanoïde | 20/03/2015

Projet Technologique

13

Pour gérer cette conversion nous utilisons le circuit fournit par le constructeur disponible sur la figure ci-dessous.



Ce circuit permet de transférer des données au moteur lorsque le DIRECTION_PORT est à 1 et de recevoir des données du moteur lorsque le DIRECTION_PORT est à 0. Le signal DIRECTION_PORT est commandé par une broche GPIO de la carte. Il existe deux possibilités pour générer ce signal :

- soit on utilise un inverseur de signal (le 74HC04 sur la figure ci-dessus qui correspond au SN74LS04N à la figure8) pour n'utiliser qu'une seule broche P8_10 sur la Beaglebone.

- soit on utilise 2 pins différents (P8_10 et P8_9) sur la carte avec une broche pour la transmission et l'autre pour la réception de données. Cette méthode permet d'avoir un circuit plus simple mais la commande de deux broches de GPIO sur la beaglebone n'est pas instantanée. Comme nous pouvons le voir le graphique où CH1 représente le signal de commande sur la broche de transmission (qui doit être à 0 pour que le signal passe) et CH2 le signal de commande sur la broche de réception. On remarque sur la partie gauche du graphique que CH1 et CH2 peuvent être tous les deux à 0 en même temps ce qui peut provoquer un problème de communication. On ne remarque pas cette erreur avec l'inverseur (figure 9). Nous utilisons donc le premier montage.

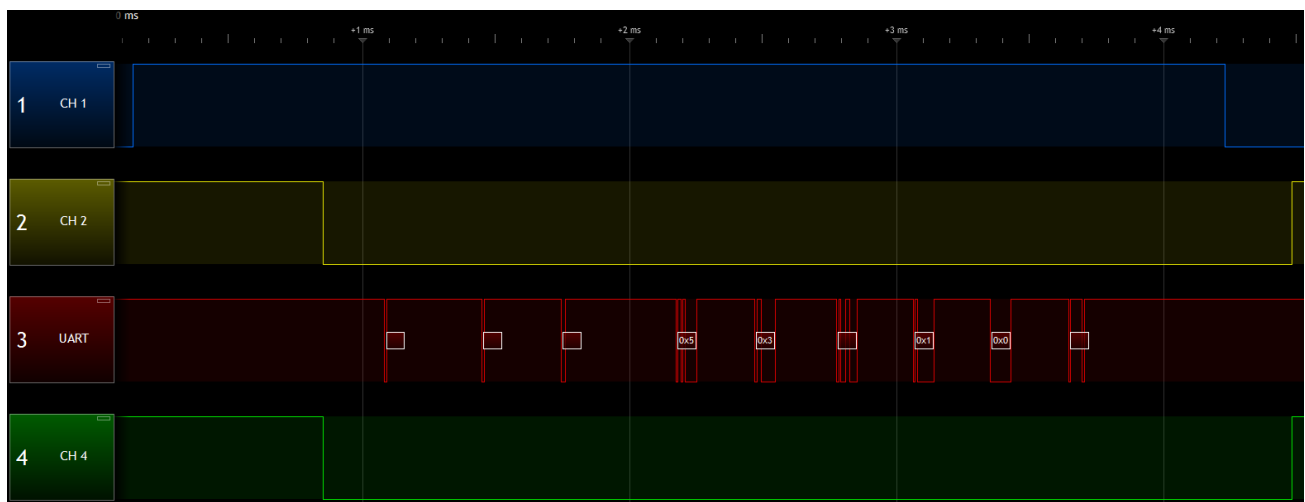


FIGURE 8

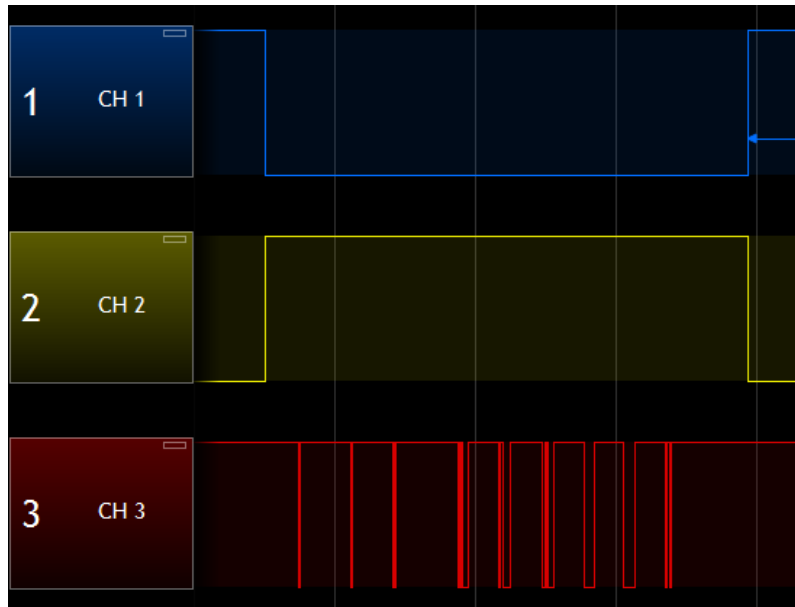


FIGURE 9

Montage électronique

Code couleur du montage : vert pour la data, rouge pour le 5V, orange pour le 3.3V et noir pour la masse. Les annexes 7 et 8 montrent la disposition du montage.

Les servomoteurs ax12 ont la particularités d'avoir la broche de DATA à 5V et la broche d'alimentation à 12V.

Fonctionnement des registres

Le scripte `moteursax12.py` permet de contrôler le(s) moteur(s) ax12 connecté à l'aide de quelques fonctions d'écriture/lecture.

Writeword

Fonction d'envoi des commandes à l'ax12 avec en premier paramètre l'identité du moteur (il faut utiliser l'adresse `0xFE` pour faire du broadcast). Le second paramètre est le nom de la commande que l'on souhaite effectuer. L'ensemble des commandes sont définies au début du programme et sont expliqué dans la datasheet (cf annexe 3). Le troisième paramètre permet de donner la valeur de la commande. Cette fonction communique avec les registres du servomoteur à l'aide de la fonction `txrx`.

readword

Cette fonction permet d'interroger le servomoteur sur son état. Le premier paramètre de cette fonction est l'identité du servomoteur et le second la commande que l'on souhaite envoyer. Cette fonction retourne la valeur du registre demandé et si la valeur reçu est valide ou pas. Contrairement à l'écriture (que l'on effectue avec la fonction `writeword`) on peut lire tous les registres qui sont répertorié dans le tableau ci-dessous.

readbyte et writebyte

Les fonctions `writebyte` et `readbyte` ont le même rôle que les fonctions `writeword` et `readword`. Elles permettent d'envoyer ou recevoir des commande taille plus petite (par

exemple AX_TORQUE_ENABLE qui ne peut valoir que 0 ou 1). Dans tous les cas on peut remplacer ces deux commandes qui ont l'avantage d'être très courtes par readword et writeword qui sont des commandes un peu plus longues sur le bus UART.

Txx

Cette fonction (qu'on utilise à travers les fonctions readword et writeword) permet d'empaqueter les commandes dans une trame UART (avec bit de Start, checksum, ...). Cette fonction retourne via les fonctions readword et writeword -2 si le retour du moteur n'est pas reçu et -3 si le checksum n'est pas bon.

Commande de position :

Les commandes très utiles pour configurer le servomoteur sont les commandes AX_GOAL_POSITION_LOW et AX_GOAL_POSITION_HIGH qui permettent de donner la consigne de position du moteur. La première commande peut avoir une valeur entre 0 et 255 et la seconde de 0 à 3. Ce qui nous permet d'avoir 1024 valeurs différentes pour coder un angle sur 300°.

Angle Limit :

Pour empêcher le moteur d'aller dans des positions qui risquent de casser le matériel, il est possible de limiter le moteur à une plage d'angle maximum. Dans notre exemple nous allons utiliser cette limite pour les doigts en « position haute » et « position basse ».

Compliance :

Les quatre registres CW Compliance Margin, CCW Compliance Margin, CW Compliance Slope et CCW Compliance Slope permettent de configurer la précision du moteur.

Les registres CW Compliance Margin, CCW Compliance Margin permettent de régler la plage angulaire qui définit si le moteur est arrivé à la position voulue avec CCW la marge « à gauche » du Goal position et CW la marge « à droite » du Goal position. Plus la valeur de ces registres est faible plus la précision est importante. Il faut cependant faire attention à ne pas mettre une valeur trop faible car la précision du moteur n'est pas infinie et si on recherche une précision trop importante le moteur risque de prendre beaucoup de temps à atteindre cette position et dans des cas extrêmes il ne pourra jamais l'atteindre.

Les registres CW Compliance Slope et CCW Compliance Slope permettent de régler le couple disponible lorsque le moteur approche de son goal position.

Pour plus d'information voir la datasheet de l'ax12 :

http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm#Actuator_Address_1E

Débugue

Afin de perdre le moins de temps possible il est très utile d'utiliser l'analyseur de signaux Scanalogic-2 (à demander au FabLab).

La configuration du logiciel est simple. Pour configurer le logiciel ScanStudio (fourni avec) il suffit d'aller dans configurer, modifier la voie de trigger sur le signal à observer, diminuer la fréquence d'échantillonnage à 500KHz, mettre le nombre d'échantillons au max et cliquer sur démarrer.

L'analyseur permet de décoder plusieurs types de bus dont l'uart. Il faut aller dans Décodeur->ajouter->UART->next et modifier la voie que l'on souhaite observer et changer le baud rate à 115200.

Conclusion

La communication de la Beaglebone vers le moteur ax12 et pwm fonctionne (il faut lancer les scripts `moteurax12.py` et `moteurpwm.py` pour les tester). Cependant la communication du moteur ax12 vers la beaglebone ne fonctionne pas. Le problème vient certainement du buffer 74hc125n qui n'est pas le buffer 74hc126 conseillé par Dynamixel (constructeur du servomoteur).

Sur la partie électronique il reste donc l'implémentation de la Leap Motion et de la kinect (les deux étant complémentaires). Il faudra cependant coordonner l'ensemble des servomoteurs lorsqu'ils seront en place.

La partie mécanique moins avancé car seulement la main et le poignet ont été construit. La fabrication du torse robot ne sera certainement pas faite en ABS car trop coûteux et de nombreuses autres solutions moins onéreuses et plus fiables seront sans doute adopté (aluminium ou bois par exemple).

ANNEXE

BBB_SRM_datasheet.pdf

Circuitversion1.png

datasheet AX-12_AX-12+_AX-12A.pdf

circuitversion1.png

BSS138_datasheet_level_converter.pdf

Logic_Level_Bidirectional_3.3to5v.pdf

Imagebeaglebone.png

Imageconverter.png

Bibliographie

- [1] <http://beagleboard.org/getting-started>
- [2] <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/gpio>
- [3] <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/pwm>
- [4] <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/uart>
- [5] <http://fablab-lannion.org/wiki/index.php?title=BeagleBone>
- [6] http://www.bwalle.de/docs/Diploma_Thesis.pdf
<https://learn.adafruit.com/introduction-to-the-beaglebone-black-device-tree/overview>
<http://hipstercircuits.com/enable-serialuarttty-on-beaglebone-black/>
- [7] http://wiki.airisep.fr/index.php?title=Dynamixel_AX-12

L'ensemble des documents ont été consultés le 20/03/15