OLIVA Maxime

Théo Martin

G4

Rapport bibliographique « PolyBras »

Sommaire

1. Application Bluetooth

- Introduction

Dans cette partie nous allons étudier et comparer les différentes manières de réaliser une application permettant de contrôler notre bras robotisé à distance à l'aide de module Bluetooth ou Wifi afin de déterminer la meilleure et de la mettre en place par la suite.

Au cours de nos recherches nous avons pu déterminer trois principales catégories proposant chacune des moyens différents afin de parvenir à notre objectif. Parmi ces trois dernières, la principale question est de choisir entre un module Bluetooth ou un Shield Wifi. C'est ce que nous allons déterminer maintenant.

- Le Shield PHPoC

Le Shield que nous allons voir permet d'établir une liaison avec la carte Arduino à travers l'interface SPI en se connectant à internet en wifi.

Tout d'abord, Il existe deux sortes de Shield PHPoC, tous deux compatibles Arduino, que l'on branche directement sur la carte :

- Le Shield PHPoC (P4S-348) : Prend en compte à la fois Ethernet et Wifi.
- Le Shield PHPoC Wifi Shield (P4S-347) : Supporte uniquement le Wifi.

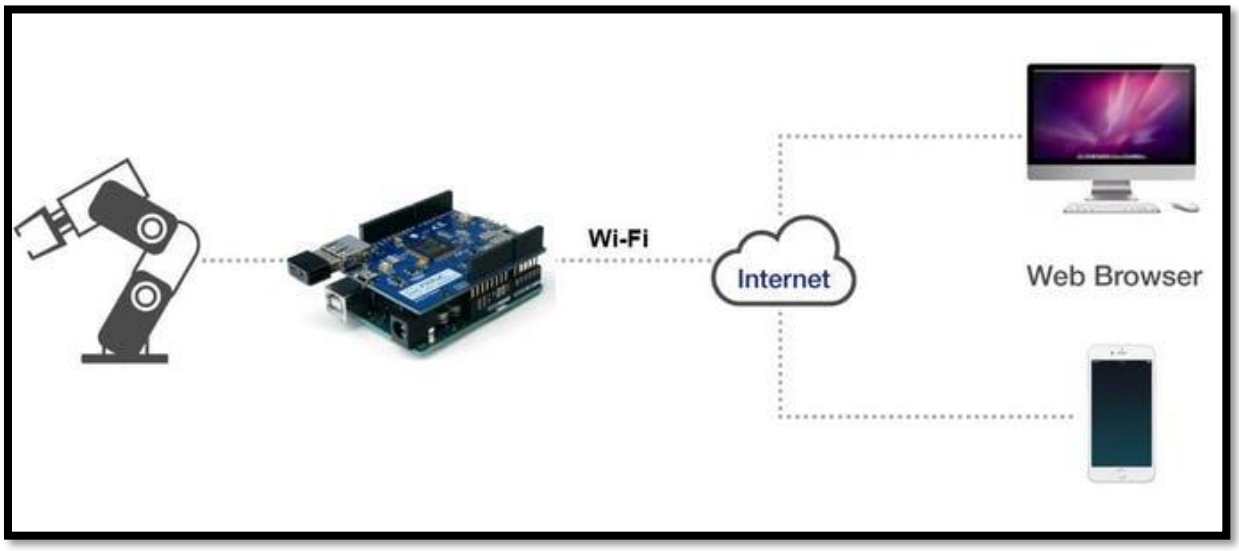
P4S-347 (Wifi)

P4S-348 (Ethernet + Wifi)

Dans notre cas, le Shield P4S-347 s'avère être suffisant. En effet, l'objectif de notre application, est qu'elle fonctionne à distance et donc sans quelconque branchement. De plus, le code pour chaque Shield est le même. Il y a également les deuxièmes versions de ces Shield qui proposent des options un peu plus avancées, lesquelles ne nous sont pas utiles dans le cadre de notre projet. Ces derniers sont très simples d'utilisation et semblent être une bonne solution pour la réalisation de notre projet mais nous allons voir dans le paragraphe suivant l'inconvénient de cette solution.

1. L'application

Une fois notre Shield choisi, il a fallu trouver un moyen efficace permettant de faire la liaison entre l'humain et la machine, en effet il faut créer une application qui, une fois reliée au Shield, enverra les informations nécessaires à notre carte Arduino afin de contrôler le bras articulé. Ci-dessous l'architecture du système.



C'est ici que la tâche se complique, les Shield PHPoC possèdent chacun un serveur web dédié, il va falloir installer l'IDE (Integrated Development Environment) du Shield afin de développer l'application web en php. Ce qui par conséquent demande une certaine maîtrise du langage de programmation php. Néanmoins cette solution est très complète car de nombreuses bibliothèques sont disponibles afin d'explorer de nombreux domaines et la programmation de l'application permettrait de ne pas être limité à uniquement des options que proposent les applications que nous allons voir par la suite.

Notre objectif va donc être de trouver une solution nous permettant d'arriver au même résultat sans passer par la découverte d'un nouveau langage de programmation qui nous ferait perdre énormément de temps dans la réalisation du projet et donc d'essayer d'optimiser notre temps de travail.

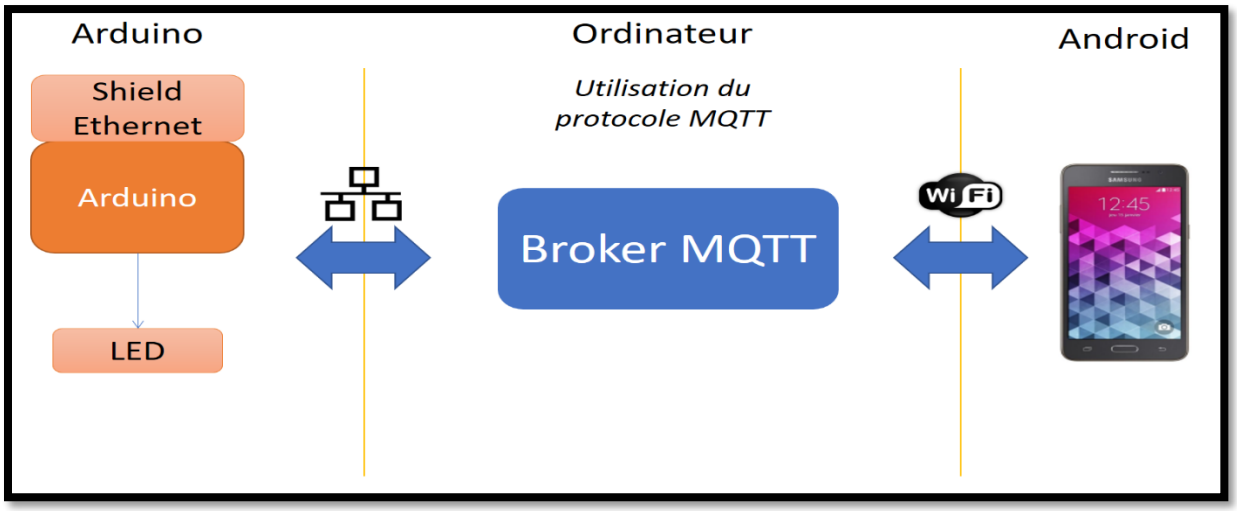
- Protocole MQTT

Une autre solution permettant de faire communiquer un appareil Android avec l'Arduino est d'utiliser un langage commun : le protocole MQTT. L'avantage de ce protocole est qu'il existe une librairie pour Android et une librairie pour Arduino. C'est un protocole de messagerie de type « Publish/Subscribe » basé sur le protocole TCP/IP. Il se décompose en trois parties :

- Les « Publishers » : ils envoient des messages sur un « Topic ».
- Le « Broker » MQTT : il fait le lien entre les « Publishers » et les « Subscribers ».

- Les « Subscribers » : ils s'abonnent à un ou plusieurs « Topic ». Lorsqu'un message est publié sur un topic, tous les subscribers du topic reçoivent le message.

L'architecture de ce système se présente de cette manière :



De nombreux Brokers existent déjà et sont disponibles en libre accès sur internet. Une fois le broker téléchargé il suffit de rentrer une ligne de commande afin de le lancer (exemple pour lancer le broker « Mosquitto » : C:\Program Files (x86)\mosquitto\mosquitto.exe" -v -p 1883).

Une fois le broker prêt à recevoir et envoyer des messages il manque plus qu'à connecter notre appareil Android au broker MQTT. Il faut dans un premier temps initialiser notre projet afin qu'il puisse communiquer avec le protocole MQTT :

- Créer un fichier Gradle.
- Ajouter les dépendances dans le fichier Gradle.
- Ajouter un service dans le Manifest.
- Ajouter les permissions dans le Manifest.
- Créer la méthode en java permettant de se connecter au broker.

2. L'application

Une fois toutes les étapes précédentes réalisées, nous allons créer notre application qui va permettre la communication avec notre Arduino.

On va donc envoyer un message sur un topic en utilisant la méthode « publish » de notre client afin que les subscribers de ce topic reçoivent le message. Voici un exemple d'un morceau de code en java :

```
Try {
        Client.publish(topic, message) ;
} catch (MqttException e) {
        e.printStackTrace() ;
```

}

Maintenant il faut créer l'application en ajoutant des boutons et autres fonctionnalités permettant de contrôler notre bras. Pour les réaliser, nous allons coder en XML et relier cette partie à notre code java qui enverra un message à l'aide de la méthode « sendMsg » précédemment écrite.

- Exemple de code XML d'un des boutons :

```
<Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="AllumerLed"
        android:text="ALLUMER" />
```

- Ainsi que du code java permettant d'envoyer le message :

```
public void AllumerLed(View v) {
     sendMsg("ON");
  }
```

Lorsque l'application est terminée, elle devrait pouvoir se connecter au broker MQTT et nous pouvons suivre dans la console que le broker MQTT reçoit un message etc.

Cette solution est également très complète car de nombreuses bibliothèques sont disponibles afin d'explorer de nombreux domaines et la programmation de l'application permettrait de ne pas être limité à uniquement des options que proposent les applications que nous allons voir par la suite. Néanmoins elle est également assez complexe et demande de nombreuses connaissances en langage java et XML ce qui représente une perte de temps potentielle si nous arrivons à trouver une solution plus adaptée et plus simple.

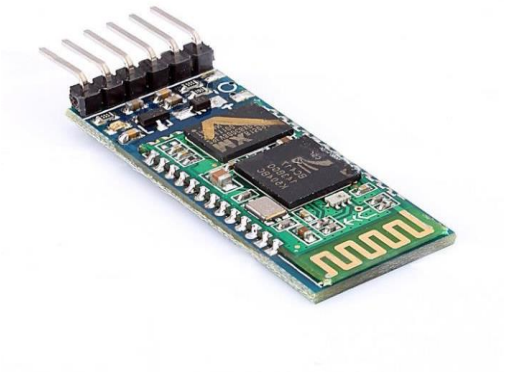
- Les modules Bluetooth (HC-05, HC-06)

Les modules Bluetooth que nous allons voir permettent d'établir une liaison Bluetooth (liaison série) entre une carte Arduino et un autre équipement possédant une connexion Bluetooth (smartphone, tablette, autre carte Arduino…).

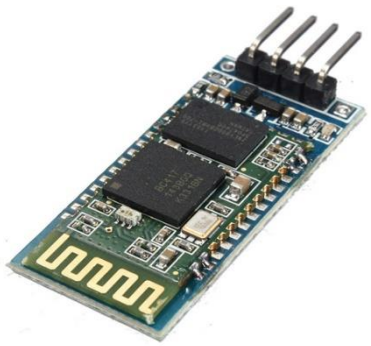
Tout d'abord, Il existe deux sortes de module Bluetooth, tous deux compatibles Arduino et utilisables sur un breadboard. On les distingue par le nombre de pattes d'entrées / sorties :

- HC-05 : 6 sorties. Ce module peut être "maître" (il peut proposer à un autre élément bluetooth de s'appairer avec lui) ou "esclave" (il ne peut que recevoir des demandes d'appairage).

- HC-06 : 4 sorties. Ce module ne peut être qu'esclave.



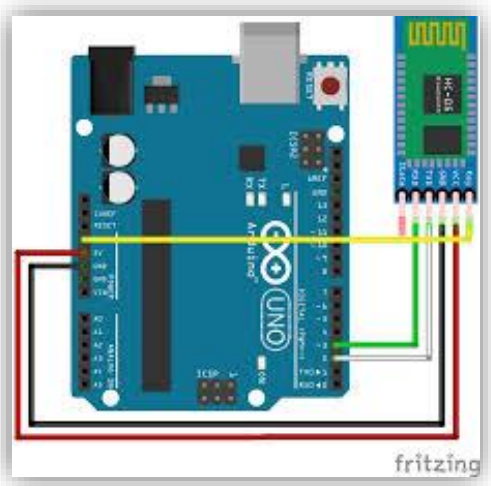
Module HC-05



Module HC-06

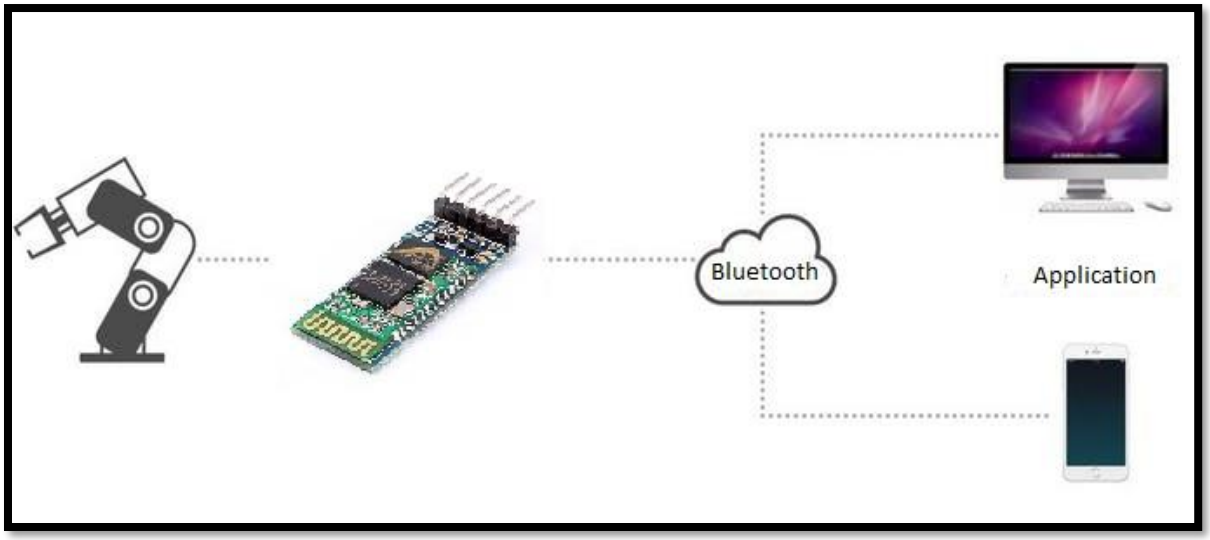
Dans notre cas, le module HC-06 s'avère être suffisant mais le module HC-05 pourrait nous permettre par la suite d'ajouter d'autres fonctionnalités au bras. De plus, après plusieurs recherches, le module HC-06 présenterait dans certains cas des soucis de compatibilité. Il y a également plus de documentation et d'aides d'utilisation sur le module HC-05. Tous ces points font que nous avons décidé de choisir le module HC-05.

En ce qui concerne le branchement du module, rien de bien compliqué, il suffit de suivre le shéma ci-dessous.



3. L'application

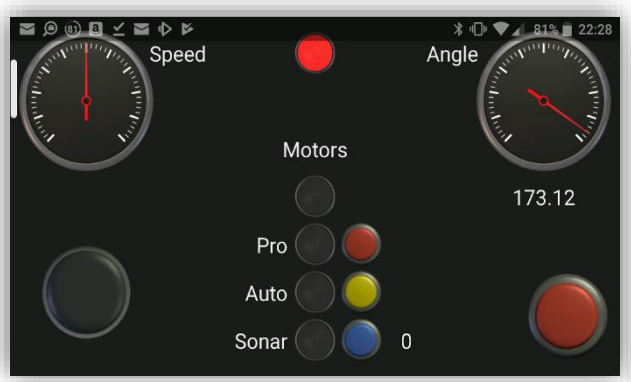
Une fois notre module choisi, il a fallu trouver un moyen efficace permettant de faire la liaison entre l'humain et la machine, en effet il faut créer une application qui, une fois connectée au module Bluetooth (en tant qu'esclave dans notre cas), enverra les informations nécessaires à notre carte Arduino afin de contrôler le bras articulé.



Dans le cadre de notre projet, le rôle de l'application est d'envoyer une instruction à la carte Arduino afin que celle-ci puisse donner les ordres aux moteurs (et autres) de se mettre en marche. Plusieurs applications existent afin de simplifier le développement d'applications sous Android et le rendre accessible à tout le monde, ne demandant pas de compétences spécifiques en programmation (php par exemple). Nous avons relevé deux de ces principales applications :

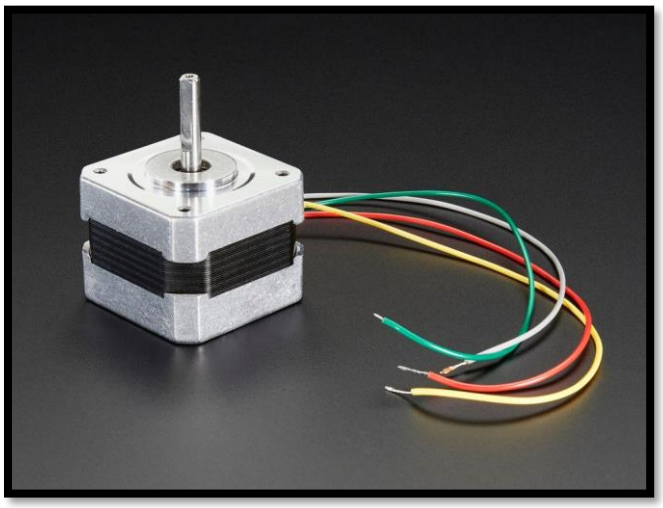
- MIT App Inventor
- Bluetooth Electronics

Ces dernières proposent des fonctionnalités similaires permettant de contrôler des moteurs à l'aide de bouton, de barres de jauge, de joystick…



De plus, elles ne nous limitent pas, en effet nous pouvons réaliser tout ce dont nous avons besoin afin de contrôler le bras articulé. Cette solution semble donc être la plus efficace car plus simple d'utilisation et plus rapide pour un résultat équivalent aux solutions vues précédemment.

2. Choix du moteur



L'objectif est de trouver des informations concernant les composants à utiliser pour notre bras robotique et plus particulièrement quels sont les moteurs les plus adaptés à nos objectifs. En effet les moteurs sont des éléments très important pour notre projet, ce sont eux qui vont permettre à notre bras de réaliser les mouvements souhaités. Il faut donc choisir des moteurs assez petits pour être intégrer à la structure du bras mais néanmoins assez puissant pour donner la force nécessaire pour soulever des objets.

Notre moteur doit remplir différents critères :

-Il ne doit pas être trop volumineux.

-il doit être assez maniable pour réaliser des mouvements précis.

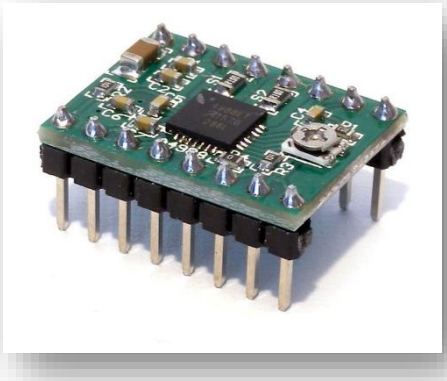
-il doit être utilisable avec la carte Arduino Uno.

-il doit être assez puissant afin de pouvoir manier des objets.

Pour cela nous avons comparé différents projets de bras robotique afin de voir quels étaient les moteurs utilisés pour ensuite sélectionner celui qui nous conviendra le plus :

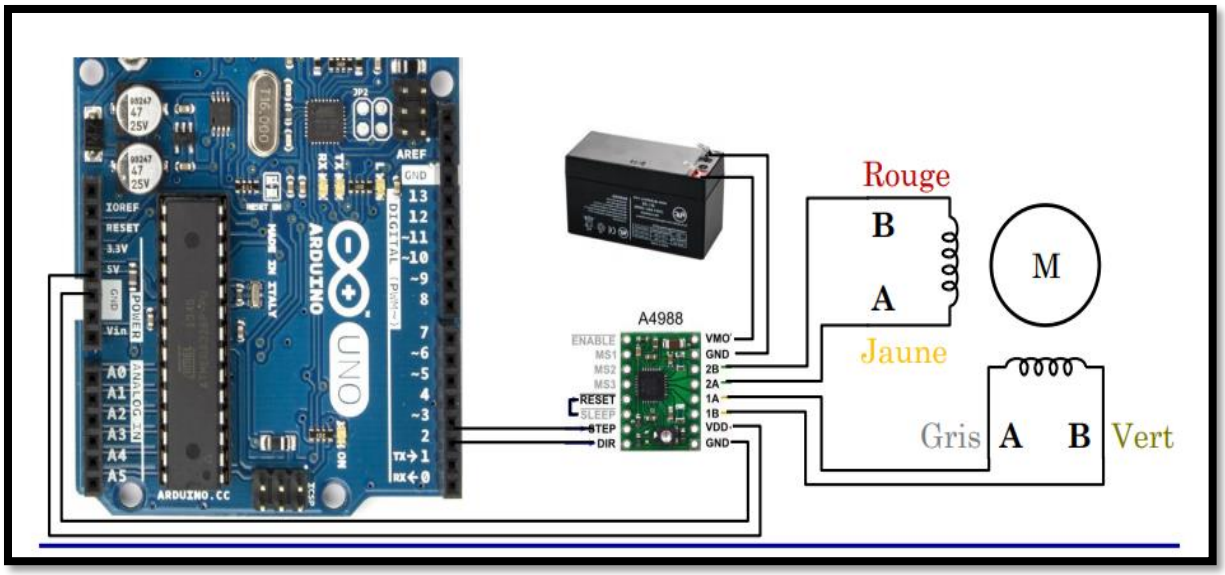
- Stepper motor NEMA-17

Ce moteur semble correspondre à nos critères de sélections. Il est assez petit, de plus il apparaît que ce moteur est un moteur pas à pas c'est-à-dire un moteur capable de de tourner à des vitesses variables et que l'on peut contrôler précisément (200 pas pour un tour complet soit un pas de 1.8 degrés). Concernant le fonctionnement du moteur, le driver A4988 pourrait nous faciliter la tâche :



Ce driver permet un contrôle très simple du moteur de plus il possède un potentiomètre capable d'ajuster le courant maximum dans les bobines et ainsi ne pas dépasser le courant limite du moteur (ici ce sera 350 mA). Mais ce n'est pas le seul avantage de ce driver, en effet il permet aussi d'augmenter la précision des mouvements du moteur et ainsi faire 1/2, 1/4, 1/8, 1/16 de pas (par exemple pour le moteur NEMA 17 un pas représente un mouvement de 1.8 degré on pourra donc faire des mouvements bien plus précis grâce à ce driver).

En ce qui concerne l'alimentation ce driver nécessite l'alimentation de la carte Arduino mais aussi celle du moteur nous obtiendrions un montage comme celui-ci :



Le problème le plus évident avec ce moteur est qu'il ne possède pas de capteurs permettant de connaître la position angulaire du moteur. Il nous faudra donc trouver une alternative afin de connaître la position angulaire de notre moteur.

Deux solutions s'offrent à nous :

- Compter le nombre de pas :

  L'avantage de cette méthode est que nous n'avons pas de matériel à ajouter il faudrait en revanche créer un programme capable de compter le nombre de pas du moteur afin de savoir à quel moment nous revenons au point initial. Ce programme devra aussi remettre les moteurs dans la position initiale à chaque fois que l'on éteint le bras robotique. Pour qu'à la prochaine utilisation les moteurs soient directement dans leurs positions angulaires initiales.

- Créer un repère qui nous permettrait de savoir quand est-ce que le moteur est revenu à son point initial :

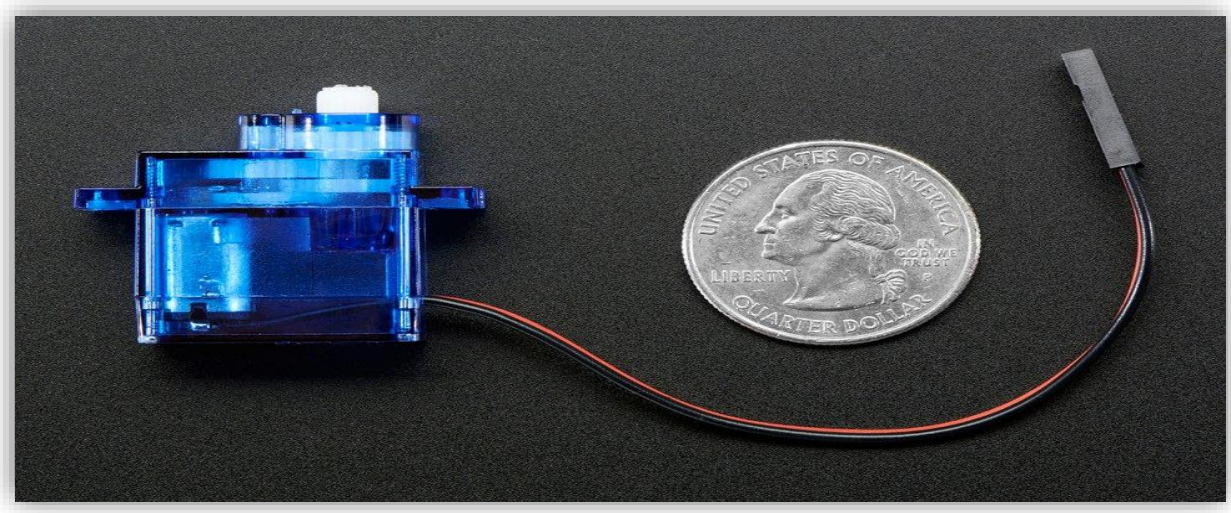
  On pourrait faire cela à l'aide d'un faisceau laser qui serait coupé à chaque fois que le moteur reviendrait à sa position initiale. Ainsi nous pourrions détecter le moment ou le faisceau est coupé et l'associer à la position initiale du moteur.

- Une autre option serait un interrupteur qui changerait de position lorsque le moteur revient à sa position initiale. Cependant il faudra voir si la présence de l'interrupteur dans notre montage ne gêne pas le bon fonctionnement du moteur.

  Les points négatifs de cette méthode sont qu'il faut rajouter un élément a notre montage et en plus de cela la position du moteur après avoir quitté la position initiale nous sera inconnue.

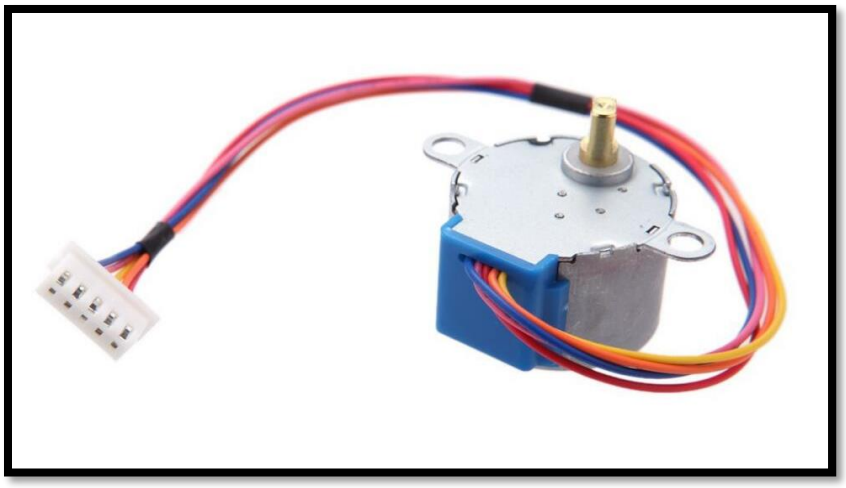
Un autre moteur qui pourrait nous intéresser serait :

- DC Motor in Micro Servo Body



Ce moteur est différent du précédent puisque c'est un servo moteur c'est-à-dire un moteur capable de réaliser une rotation d'un certain angle compris entre 0° et 180°. Il est plutôt petit donc assez facile à intégrer à la structure cependant il semble offrir moins de puissance et comparé au moteur pas à pas il offre un angle de rotation réduit de moitié. Cependant un moteur comme celui-ci peut parfaitement servir. En effet ce type de moteur nous permettrai d'éviter les problèmes d'angle que nous rencontrons avec le moteur pas à pas. Il reste donc une option envisageable pour notre projet.

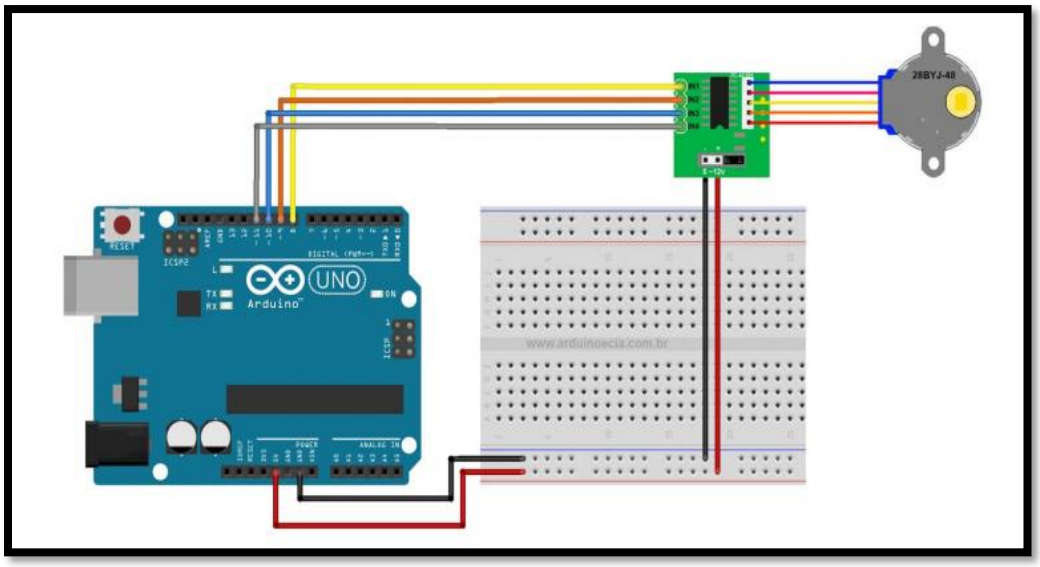
- Moteur 28BYJ-48-08 5 Vcc



Ce moteur aussi semble correspondre à nos critères, c'est lui aussi un moteur pas à pas tout comme le NEMA-17, cependant ce moteur possède 32 pas pour un tour complet. Mais ce qui

est intéressant est le fait qu'il possède aussi un réducteur permettant d'augmenter le nombre de pas à 2048 pour faire un tour complet nous offrant une précision plus que suffisante. Le problème de ce moteur est qu'il semble moins puissant que le moteur Nema-17.

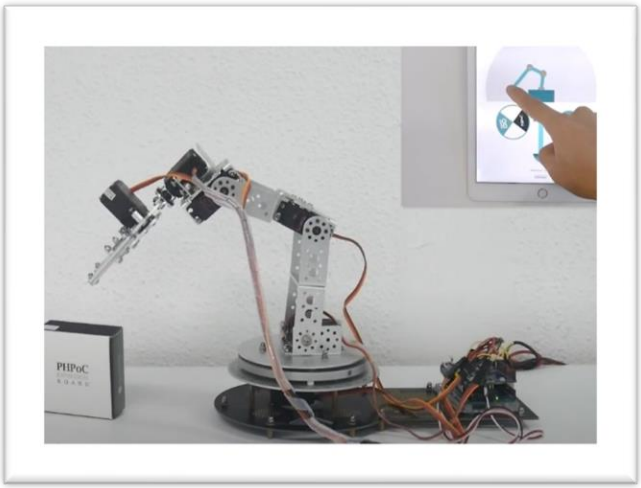
Ce moteur fonctionne avec le circuit ULN2003 et on obtient finalement ce montage :



En conclusion on a vu qu'il existait beaucoup de moteurs différents avec des avantages et des inconvénients. Il nous faudra donc choisir nos moteurs en fonction de nos besoins afin de minimiser le plus possible les inconvénients de chacun et de profité au maximum de leurs avantages.

3. La structure

Pour la structure du bras robotique on va s'inspirer de projets déjà existant.



Ces 2 projets proposent une structure assez similaire. Ce qui est intéressant ici ce sont les articulations du bras qui ne se gênent pas les unes les autres avec un design épuré.

Il nous faudra un socle pour tenir le bras capable de faire une rotation de 360° pour permettre au bras de saisir un objet peu importe où il se trouve. Il faudra ensuite faire les articulations puis gérer la pince afin que celle-ci puisse se fermer et s'ouvrir mais aussi qu'elle puisse se baisser ou monter. On pourrait rajouter un grip au niveau de la pince afin que celle-ci saisisse mieux les objets.

Pour les matériaux de la structure il nous faut quelque chose de résistant mais néanmoins assez léger pour que les moteurs ne peinent pas à faire bouger la structure.

- On pourrait imprimer les pièces grâce à une imprimante 3D et aux fichiers laissés par d'autres projets pour imprimer les pièces. Le plastique étant assez léger et résistant cela pourrait être une bonne option.

- Le carton est aussi une option envisageable car très léger cependant le carton risque de plier trop facilement si la charge que le bras essaie de lever est trop lourde. Il reste malgré tout utile s'il faut modifier des parties trop lourdes de notre projet.

- Le métal lui permettrait à la structure d'être très solide cependant le métal reste un matériel lourd. Il pourrait malgré tout permettre de solidifier une partie faible de notre structure.

- Le bois semble être un bon compromis entre poids et solidité bien que lourd lui aussi.

4. Comparaison avec un projet similaire

Dans cette partie nous allons comparer nos solutions à celles d'un projet déjà abouti afin de voir si nos solutions sont cohérentes ou s'il faut les changer.

Nous n'avons pas trouvé de projet similaire au notre sur le site de Pascal Masson, nous avons donc pris l'exemple d'un projet sur internet. Ci-dessous le lien du projet :

https://forum.phpoc.com/blogs/khanh-s-blog/1129-arduino-control-arm-robot-via-web

En ce qui concerne les composants utilisés :

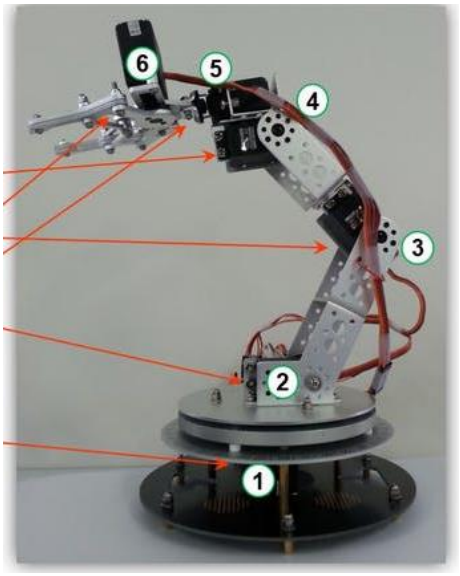
- Projet trouvé sur internet :
  Carte Arduino UNO
  PHPoC Wifi Shield
  6DOF Arm Robot

- Notre projet :
  Carte Arduino UNO
  Module HC-05

Structure du bras en métal très certainement

En ce qui concerne le module permettant de faire la liaison entre l'application et la carte Arduino nous avons opté pour une solution différente à celle du projet d'internet en remplaçant le Shield Wifi par un module Bluetooth. La structure du bras devrait être relativement similaire, en métal également.

Au niveau des moteurs, le projet que nous avons trouvé propose six moteurs :

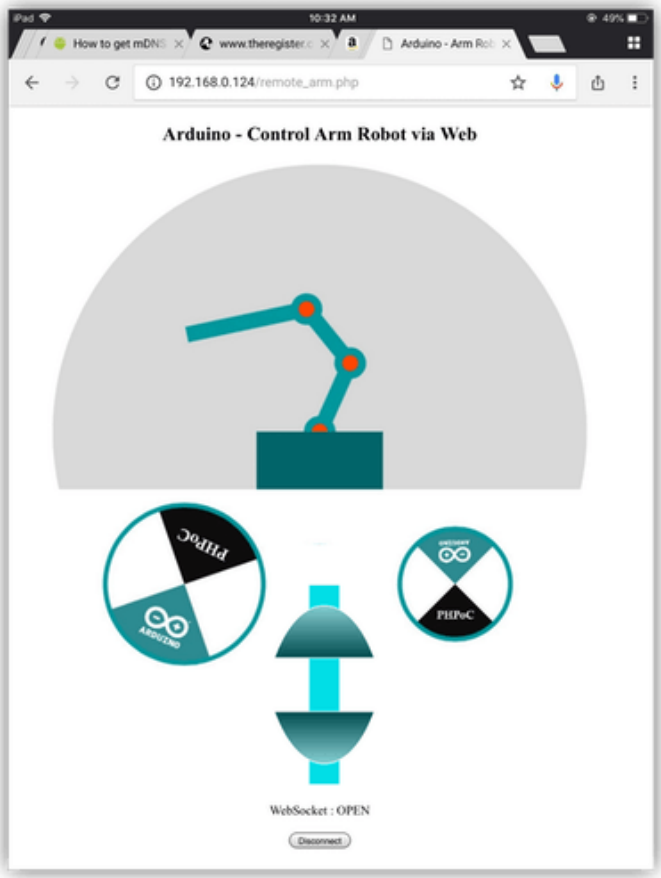


Nous avons décidé d'en supprimer un et de garder :

- Un moteur permettant de contrôler la base du bras.
- Un moteur permettant de contrôler la pince.
- Un moteur permettant de contrôler la rotation de la pince.
- Deux moteurs permettant de contrôler le bras.

En ce qui concerne L'application et son interface :

- Projet trouvé sur internet :



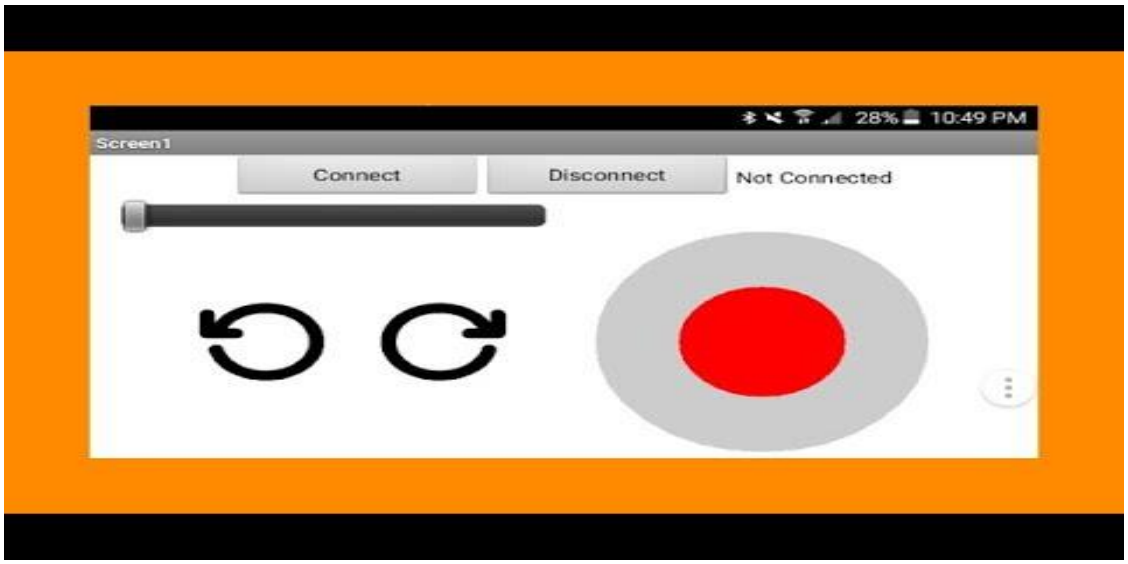
L'application est hébergée sur un serveur web.

Elle est entièrement codée en PHP et possède une partie avec un shéma du bras permettant de contrôler trois des six moteurs du projet.

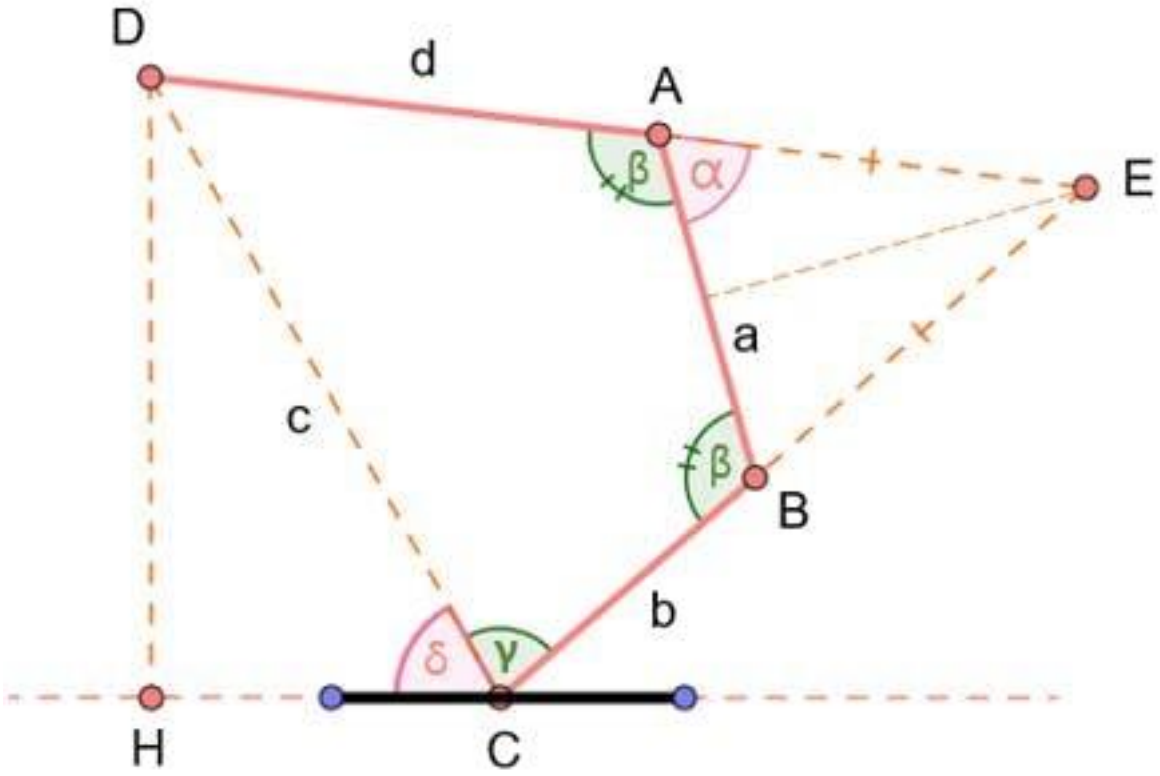
- Notre projet :

Dans notre projet nous partons sur une application android développée à l'aide de MIT AppInventor ou Bluetooth Electronics.

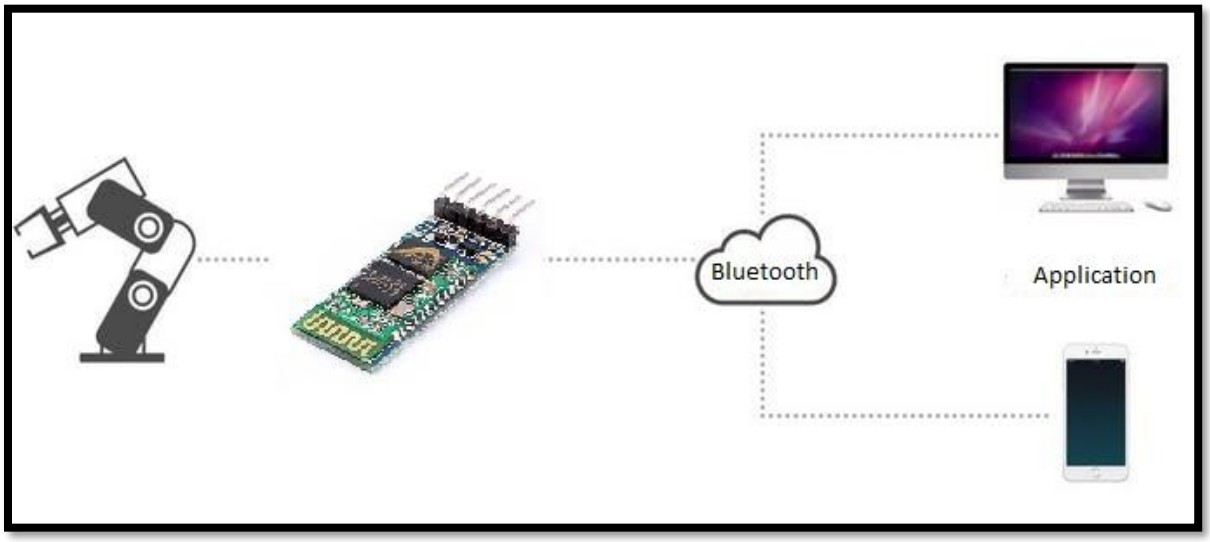
Le shéma du bras ne sera pas présent, à la place de celui-ci chaque moteur devrait être contrôlé séparément à l'aide de boutons/joysticks :



Nous évitant d'avoir à gérer tous les angles et les orientations des moteurs :



L'architecture de notre projet est très similaire à celle du projet que nous avons trouvé sur internet et semble être relatively efficace.



5. Sources

https://forum.phpoc.com/blogs/khanh-s-blog/1129-arduino-control-arm-robot-via-web

https://create.arduino.cc/projecthub/MisterBotBreak/how-to-make-a-robotic-arm-783525

https://www.adafruit.com/product/2941

https://www.adafruit.com/product/324#description

http://users.polytech.unice.fr/~pmasson/Enseignement/Elements%20de%20robotique%20avec%20arduino%20-%20Moteurs%20-%20Projection%20-%20MASSON.pdf

https://www.gotronic.fr/art-moteur-28byj-48-08-5-vcc-21213.htm#complte_desc

https://iknowvations.in/fr/arduino/a4988-stepper-motor-driver-arduino-tutorial/

https://create.arduino.cc/projecthub/slantconcepts/control-arduino-robot-arm-with-android-app-1c0d96?ref=search&ref_id=arme&offset=32

https://create.arduino.cc/projecthub/igorF2/nunchuk-controlled-robotic-arm-with-arduino-b1c0fa?ref=search&ref_id=robotic%20arm&offset=22

https://www.phpoc.com/phpoc_wifi_shield.php#phpoc_wifi_shield_overview

https://forum.phpoc.com/blogs/khanh-s-blog/1129-arduino-control-arm-robot-via-web

https://store.arduino.cc/arduino-uno-rev3

https://yadi.sk/d/DWNFC1O4dWeVvw

https://www.youtube.com/watch?v=iC564i0wf0k

https://retroetgeek.com/arduino/creer-une-application-android-avec-appinventor-et-controler-un-arduino-en-bluetooth/

https://zestedesavoir.com/tutoriels/2069/faites-communiquer-votre-arduino-avec-votre-appareil-android/

https://eskimon.fr/tuto-arduino-907-utiliser-un-module-bluetooth-hc-05-avec-arduino

https://www.youtube.com/watch?v=o-YVvxYiSuk

https://pedagogie.ac-reunion.fr/fileadmin/ANNEXES-ACADEMIQUES/03-PEDAGOGIE/03-LYCEE/Voie-generale-technologique/sciences-ingenieur/technologie/Comment_PILOTER_UNE_LED_SUR_arduino_PAR_UN_SMARTPHONE.pdf

https://howtomechatronics.com/tutorials/arduino/how-to-build-custom-android-app-for-your-arduino-project-using-mit-app-inventor/

https://www.gotronic.fr/pj2-35192-1551.pdf

https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/

http://www.techbitar.com/modify-the-hc-05-bluetooth-module-defaults-using-at-commands.html#