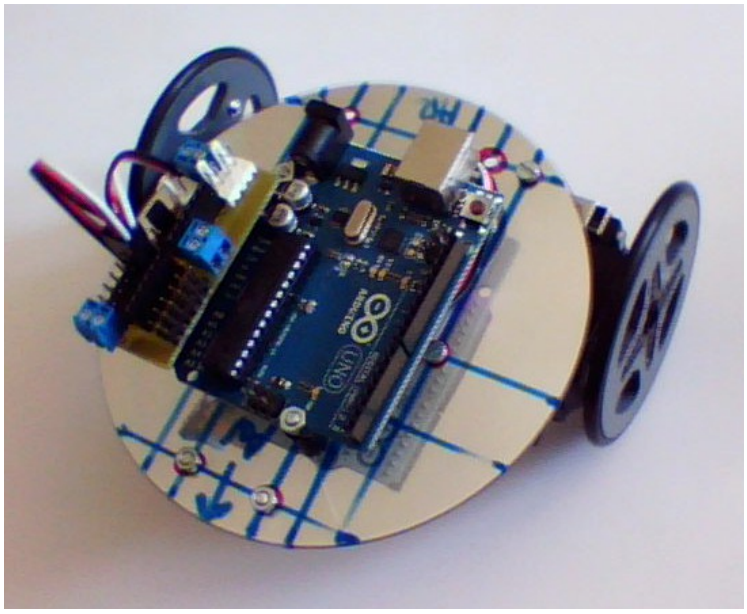
Le Simple CD-Bot

Par X. HINAULT – Juin 2012 – Avril 2014 – Licence GPL – open source & hardware – www.mon-club-elec.fr



Un robot didactique « low-cost » facile à fabriquer et permettant de s'initier aux mouvements de base d'un robot mobile contrôlé par programmation ainsi qu'à l'utilisation de capteurs.

Cahier des charges

L'objectif ici est de construire un petit robot d'initiation facilement reproductible en nombre pour être utilisés dans le cadre d'ateliers Arduino d'initiation à la robotique.

Les contraintes sont :

- **reproductibilité :** toutes les pièces utilisées doivent être faciles à trouver, à mettre en oeuvre ou à fabriquer soi-même
- **low-cost :** le coût doit être acceptable pour une utilisation en nombre, de l'ordre de 30 à 40€
- **réutilisabilité :** les pièces utilisées ne doivent pas être dédiées spécifiquement pour ce robot mais doivent pouvoir être réutilisées pour d'autres réalisations ultérieures sans difficulté.
- **électronique libre et ouverte :** électronique open-source et open-hardware
- **programmation libre et open-source :** code open-source et GPL
- **simplicité :** la réalisation de ce robot ne doit pas demander de matériel particulier en dehors d'un matériel de base. Il doit être facilement réalisable sans avoir de compétences particulière en mécanique ou bricolage.

Outillage utile

Pour la fabrication (simple) des plateaux du châssis :

- une paire de gros ciseaux
- une règle 20cm ou +
- un stylo et un feutre indélébile
- une mini-perçeuse type Dremel
- 1 mèche de perçage métal Ø3mm

Pour le montage mécanique :

- un tournevis plat
- un petit tournevis cruciforme
- +/- une clé à pipe 5,5 (pour écrous M3)

Pour le montage électronique du mini-shield de connexion des servomoteurs

- d'un fer à souder à pointe fine, 25-30W et de son support
- de soudure d'étain 60% dite « électronique » - Ø 1mm
- d'un rouleau de scotch (pratique pour faire tenir les composants)
- d'une petite pince coupante
- d'une pompe à dessouder (pour rattraper le coup au besoin)

Pour la programmation

- Ordinateur avec logiciel Arduino installé
- Câble USB

Les pièces mécaniques du robot

Châssis

- 2 CD usagés (12cm de diamètre)
- cage Easy pour fixation de servomoteur x 2

Moteurs et roues

- servomoteurs standards à rotation continue x 2
- petite roue libre dite « 20kg » avant x 1
- 2 roues plastiques 65mm pour servomoteur avec palonnier intégré + vis fixation sur axe servo x2

Visserie utile

- vis tête plate Ø3mm (dite M3) – long. 15mm x 2 (4 pour fixer cages Easy et 2 pour fixer carte Arduino)
- vis tête plate Ø3mm (dite M3) – long. 10mm x 8 (4 pour fixer châssis inf. et 4 pour fixer servo dans cage Easy)
- vis tête plate Ø3mm (dite M3) – long. 20mm x 4 (fixation roue libre avant)
- écrous Ø3mm (dits M3) x 10 env.
- entretoises plastique 5mm x 4 (fixation châssis inférieur)
- entretoises 10mm x 6 (4 pour fixer cages Easy et 2 pour fixer carte Arduino)
- entretoises 15mm x 4 (roue libre avant) (voire 10mm selon modèle roue avant)

Alimentation du robot

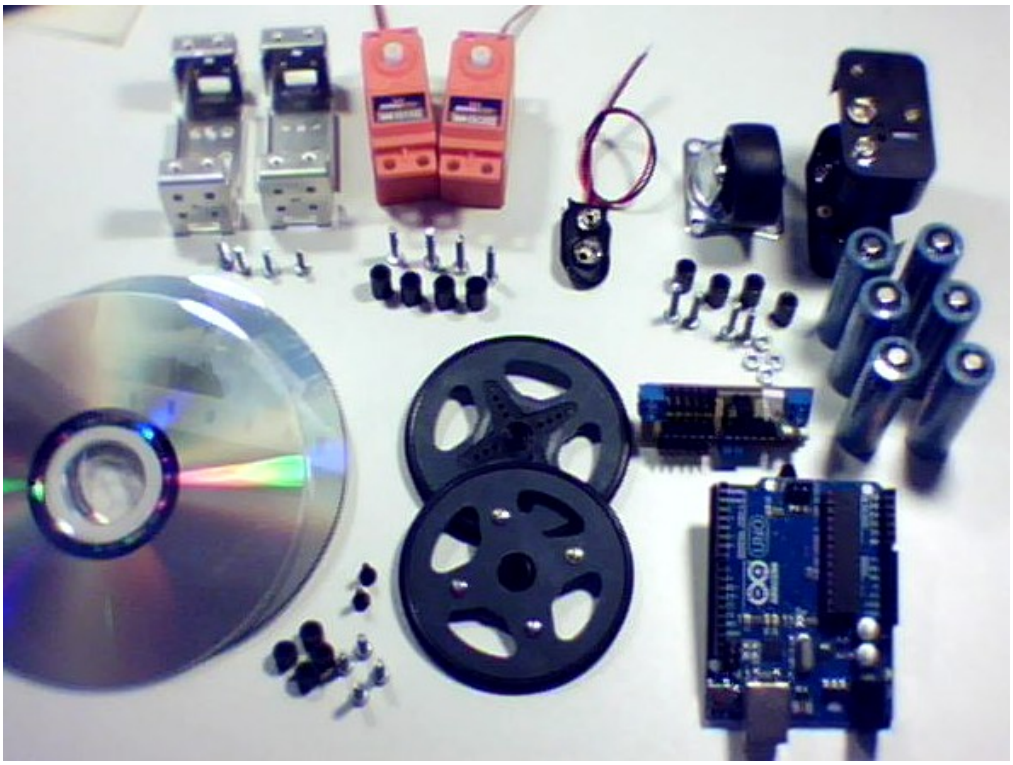
- soit par le port USB en 5V/500mA
- soit par un bloc de 6 piles réalisé avec :
  - un coupleur de 6 piles LR6
  - un câble de connexion pile 9V à pression

L'électronique du robot

- carte de commande : une carte Arduino UNO v3 (non fournie)
- un mini-shield de dédoublement des 6 broches dites « analogiques » sur connecteurs droits 3 broches.

Les capteurs du robot

- 2 photo-résistances + résistance en série
- connecteurs servomoteurs JR x 2
- adhésif d'électricien



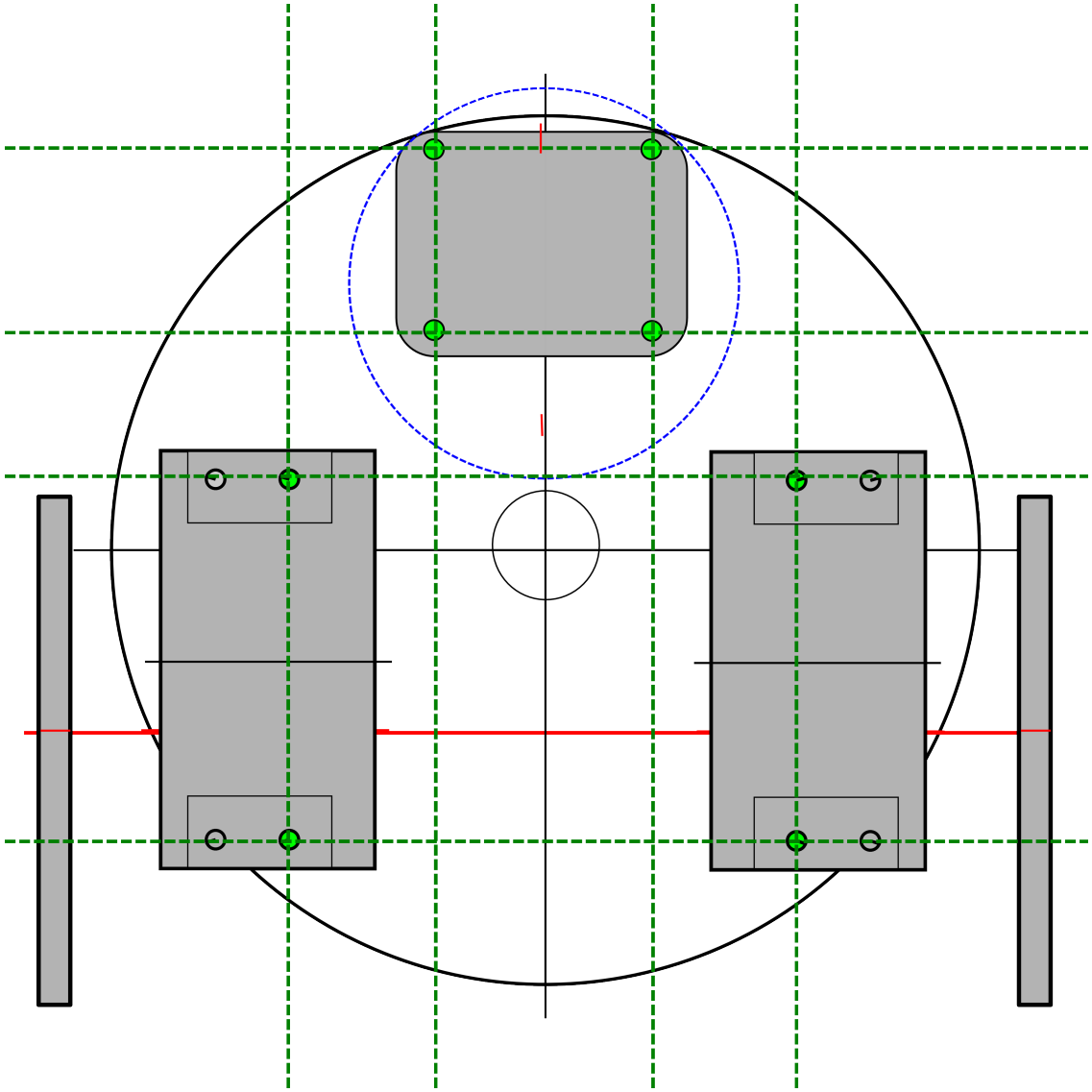
Les pièces utiles pour construire le Simple CD-Bot (hors capteurs)
Coût du « kit » à la boutique de l'atelier = 40€ (hors Arduino et vis)

Fabrication du plateau supérieur du châssis

Le but de cette étape est de préparer le plateau supérieur du châssis du robot qui est fabriqué au final à l'aide de 2 CD usagés (un pour le plateau supérieur et un pour le plateau inférieur – vu ultérieurement).

Noter que l'on propose ici une solution « very-low-cost » (= « très peu pas cher du tout » !) pour le châssis. Mais, si l'on veut faire les choses un peu plus sérieusement, on pourra se rabattre à tout moment sur un châssis aluminium pré-découpé (diamètre 13cm contre 12cm pour le CD - le reste du robot est identique... (disponible sur commande à la boutique des ateliers Arduino) :

- Imprimer le patron de traçage du CD fourni (fichier *.png de l'archive) Le CD fait 12cm :

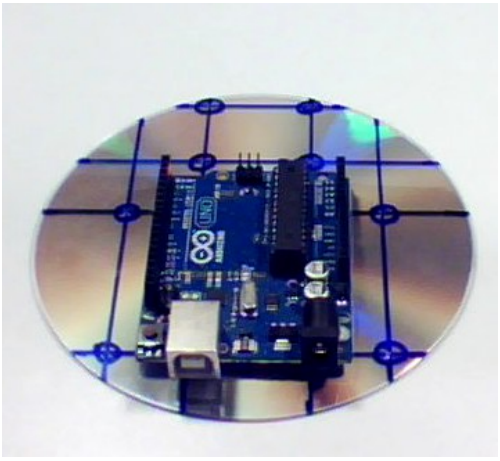


- Poser et centrer le premier CD sur le patron de traçage

- Tracer au feutre indélébile (effaçable ensuite avec un chiffon imbibé d'alcool à brûler) les lignes de repérage (lignes vertes du patron) :
  - 4 lignes horizontales (soit de haut en bas à 5mm, 30mm, 50mm, 100mm)
  - 4 lignes verticales (soit de gauche à droite à 25mm, 45mm, 75mm, 95mm)

- repérer ensuite les points d'intersection à percer comme sur la photo :
  - vérifier le bon placement des trous de la roue avant en la posant sur le tracé – corriger au besoin
  - faire une encoche de perçage à chaque point en enfonçant légèrement le coin d'un tournevis plat :

- ensuite, poser la carte Arduino sur le CD en tournant le connecteur USB vers l'arrière (la roue libre est à l'avant) et de façon à ce que la carte soit centrée entre les 2 lignes verticales externes et que le bord du connecteur USB soit à 1,5cm environ du bord inférieur du CD (comme sur la photo). Marquer les points de fixation.




- une fois fait, percer simplement les trous à la mini-perceuse avec une mèche métal de 3mm. Installez-vous dans un endroit où vous pouvez faire un peu de poussière... Mettre une planchette de bois sous le CD pour éviter de percer la table...

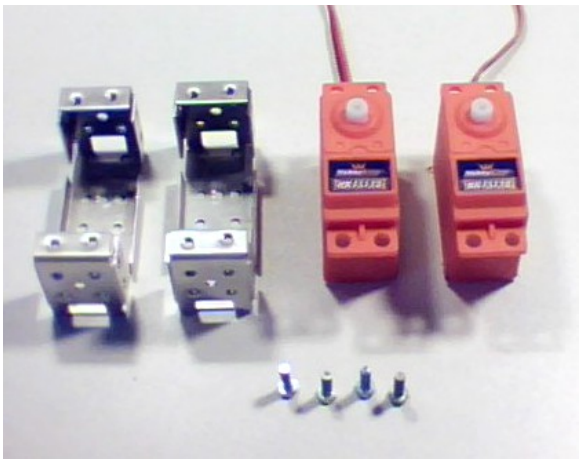
- Ebavurer simplement avec les doigts, passer un chiffon imbibé d'alcool à brûler pour effacer les marques de traçage. Voilà, le plateau supérieur du châssis est prêt :

Montage des moteurs dans les cages de fixation

Pièces nécessaires pour cette étape

Cette étape nécessite :

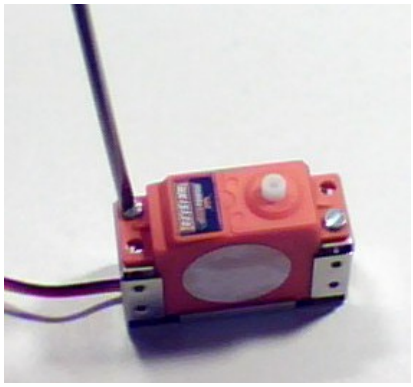
- les 2 cages Easy
- les 2 servomoteurs
- 4 vis tête plate Ø3mm (dite M3) – long. 10mm



Réalisation de cette étape

On va monter le premier servomoteur dans sa cage Easy. On commence par passer le câble dans la cage de façon à ce qu'il sorte à l'opposé de l'axe du servomoteur :

Puis on insère le servomoteur dans la cage sans forcer et on le fixe à l'aide de 2 vis tête plate M3 – utiliser de chaque côté l'un des trous, celui qui est le plus naturellement en vis à vis d'un trou du servo lorsque la cage enserre le servo – Veiller également à ce que le câble passe sous le servomoteur bien à plat :



On procède de la même façon pour le deuxième servomoteur :

Montage des cages avec servos sur le plateau supérieur du châssis.

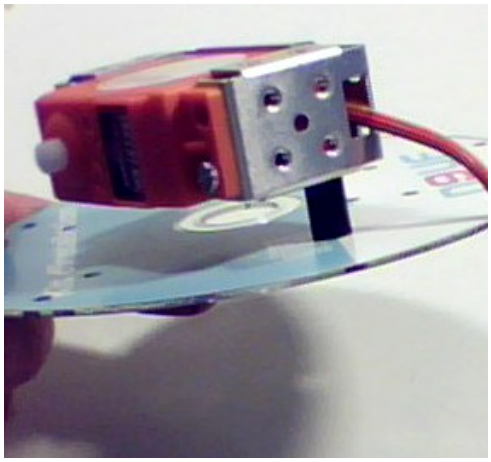
Matériel nécessaire pour cette étape

- 4 entretoises 10mm
- vis tête plate Ø3mm (dite M3) – long. 15mm
- le CD de plateau supérieur du châssis préparé précédemment

Réalisation de cette étape

Passer une vis M3 de 15mm dans l'un des trous les plus latéraux (pour se repérer : l'avant correspond au 4 trous en « carré ») et une entretoise :

Puis visser l'un des cages avec le servomoteur en veillant à ce que l'axe du servomoteur soit à l'arrière – utiliser le trou du côté de la cage le plus proche du fond. Ce trou est fileté au pas M3 :

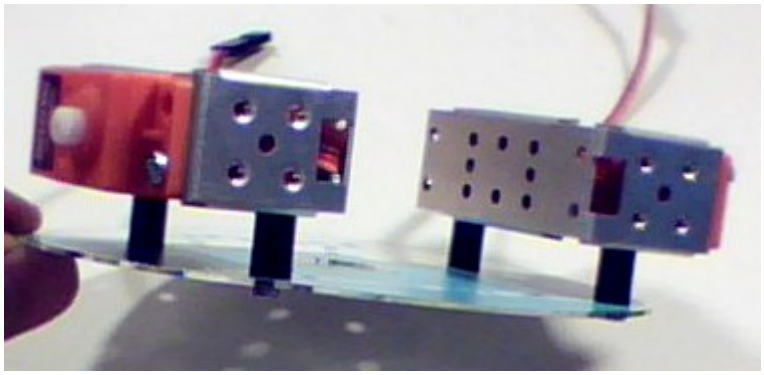


Ensuite passer une 2ème vis M3/15mm et son entretoise dans le trou associé et fixer la cage de la même façon :




Fixer la seconde cage avec servomoteur de la même façon de l'autre côté, l'axe vers l'arrière et en utilisant les 2 trous les plus proches du fond sur le côté de la cage :



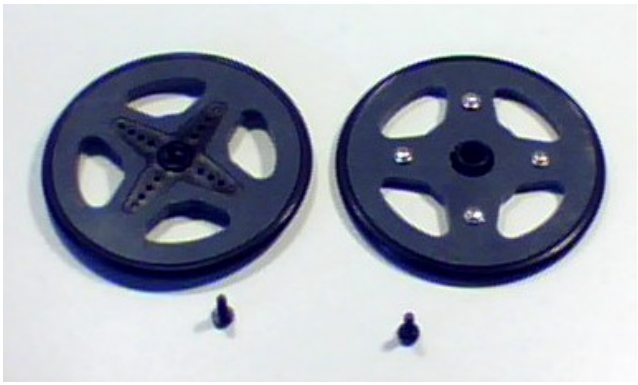


Faire passer les câbles via le trou central :

Montage des roues des servomoteurs

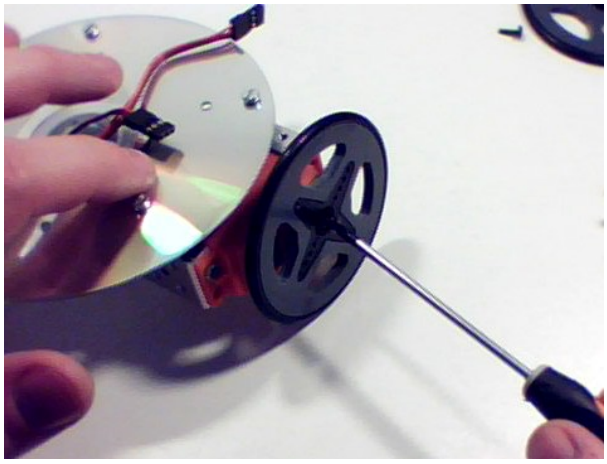
Pièces nécessaires pour cette étape

- les roues plastiques 65mm pour servomoteur avec palonnier servomoteur
- 2 vis de fixation de palonnier de servomoteur (1 vis par roue)

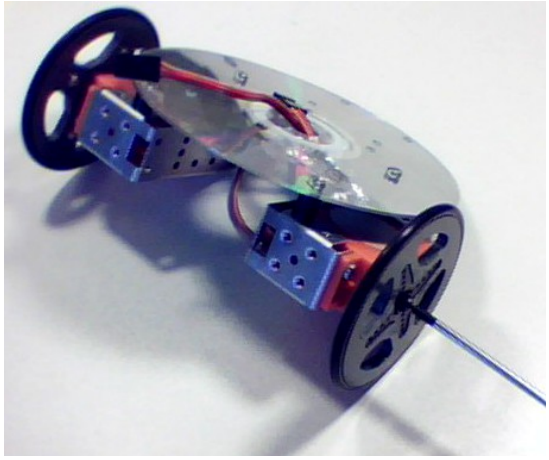


Réalisation de cette étape

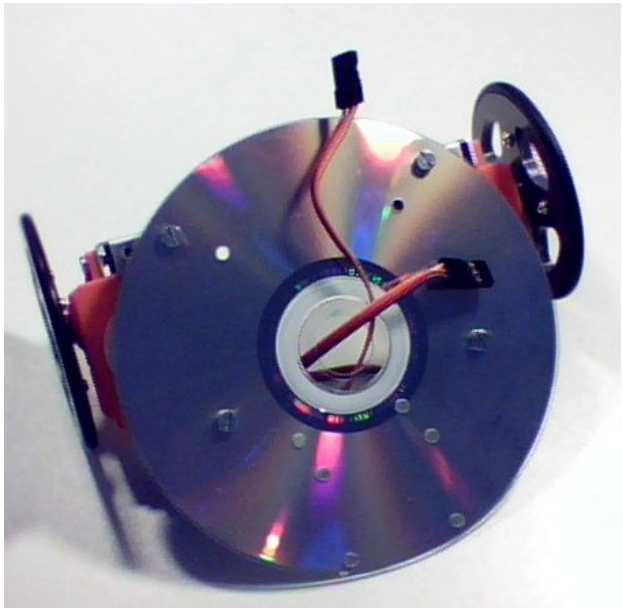
- Enficher une roue sur l'un des axes de servomoteur et visser la vis à l'aide d'un tournevis cruciforme en empêchant la roue de tourner :



Fixer de la même façon la roue de l'autre côté :



La « bête » commence à prendre forme :



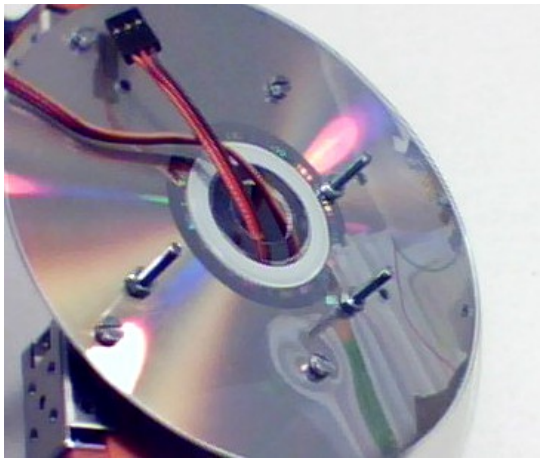
Mise en place des fixations de la carte Arduino de commande

Pièces nécessaires pour cette étape

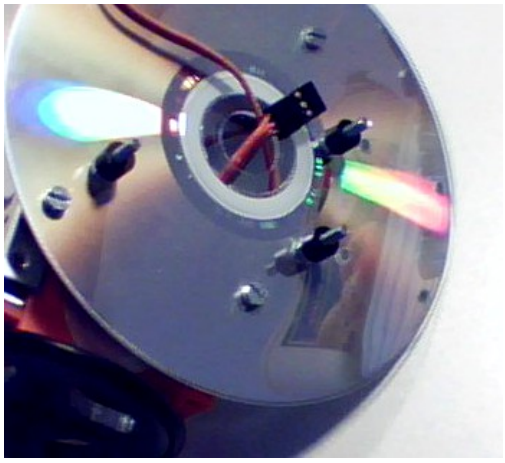
- 2 à 3 vis M3/15mm
- 6 écrous M3
- 3 entretoises 5mm
- une carte Arduino (pas indispensable ici...)

Réalisation de cette étape

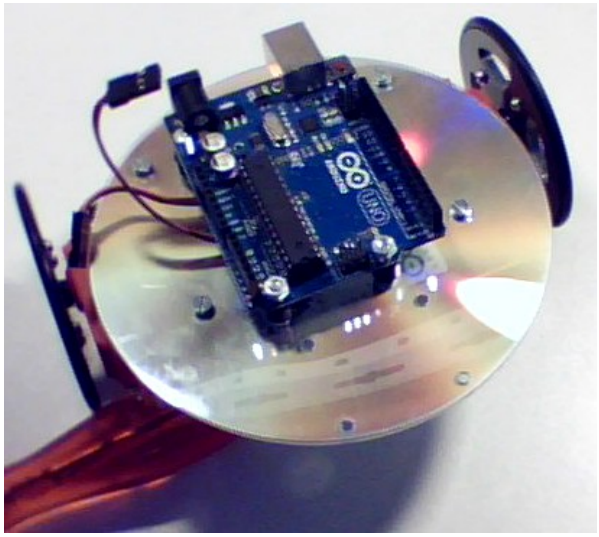
Fixer sur le plateau supérieur du châssis 2 à 3 vis M3/15mm avec un écrou, tournée vers le haut :



Ensuite passer 2 ou 3 entretoises sur les vis :



Tester ensuite le positionnement de la carte Arduino (connecteur USB vers l'arrière) en laissant sortir les câbles des servomoteurs du côté des broches analogiques et visser avec 3 écrous :



Pour la suite, on peut enlever la carte Arduino, tout en laissant les entretoises et l'écrou. Cette façon de faire permet d'utiliser la carte Arduino avec le robot uniquement lorsque cela est nécessaire. La carte Arduino reste disponible pour d'autres utilisations au besoin : pratique !

Fixation de la roue libre avant

Pièces nécessaires pour cette étape

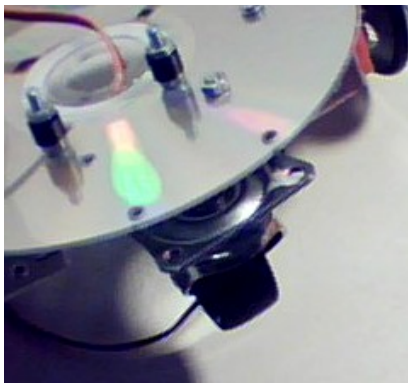
- une roue libre avant
- 4 vis M3/20mm
- 4 entretoises 10mm

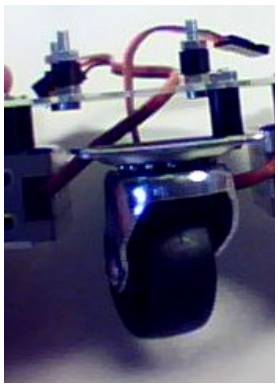
Selon le modèle de roue avant utilisée, il faut parfois utiliser plutôt 4 vis M3/25mm et 4 entretoises 15mm pour obtenir l'horizontalité du robot. L'idéal est une légère inclinaison du robot vers l'avant.

Réalisation de cette étape

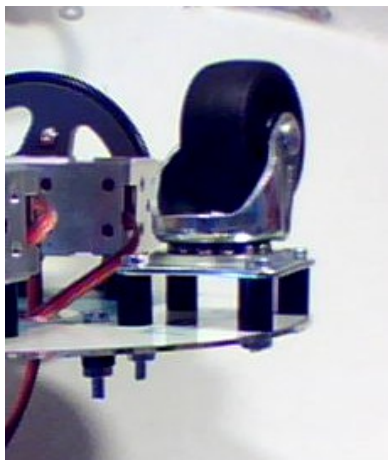
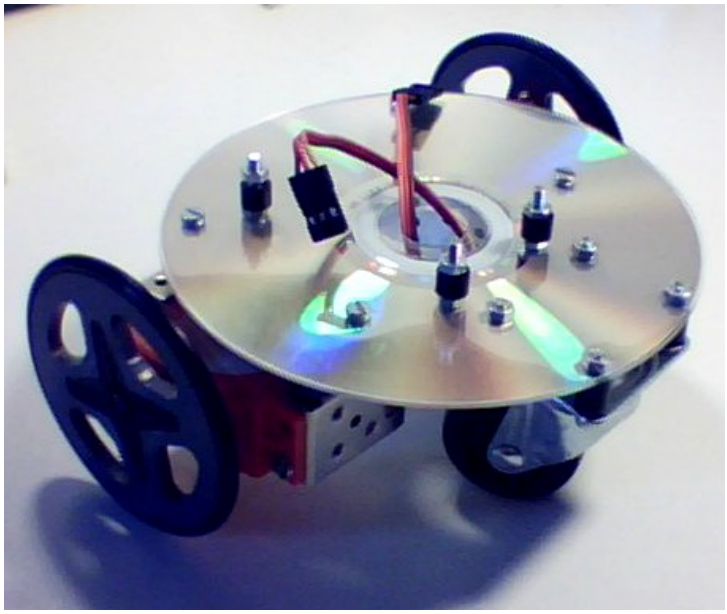
On enfile une vis M3 dans le support de la roue, tête en bas et filetage vers le haut et on enfile l'entretoise :

On vient ensuite passer la vis dans le plateau supérieur du châssis en veillant à ce que les 4 trous soient en correspondance (le support de la roue n'est pas carré...) et on visse avec un écrou :



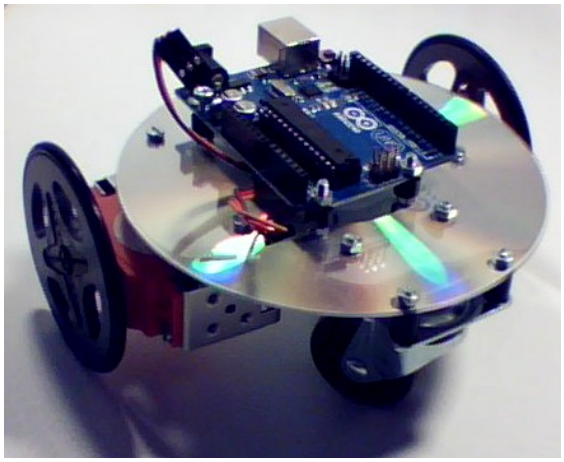


Faire la même chose pour les 3 autres vis M3 et les entretoises – l'opération est plus ou moins aisée : le plus simple consiste garder le plateau à plat à l'envers, à placer l'entretoise (entre le support de roue et le plateau de châssis) en correspondance du trou voulu , puis la vis et visser l'écrou :

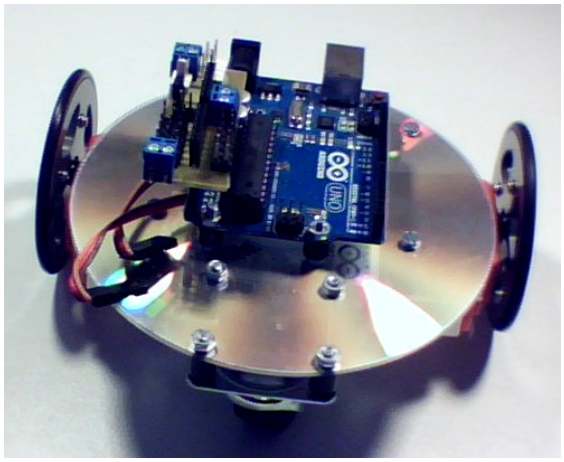




A ce stade, le robot est d'ors et déjà utilisable en restant branché sur le port USB. Pour cela :

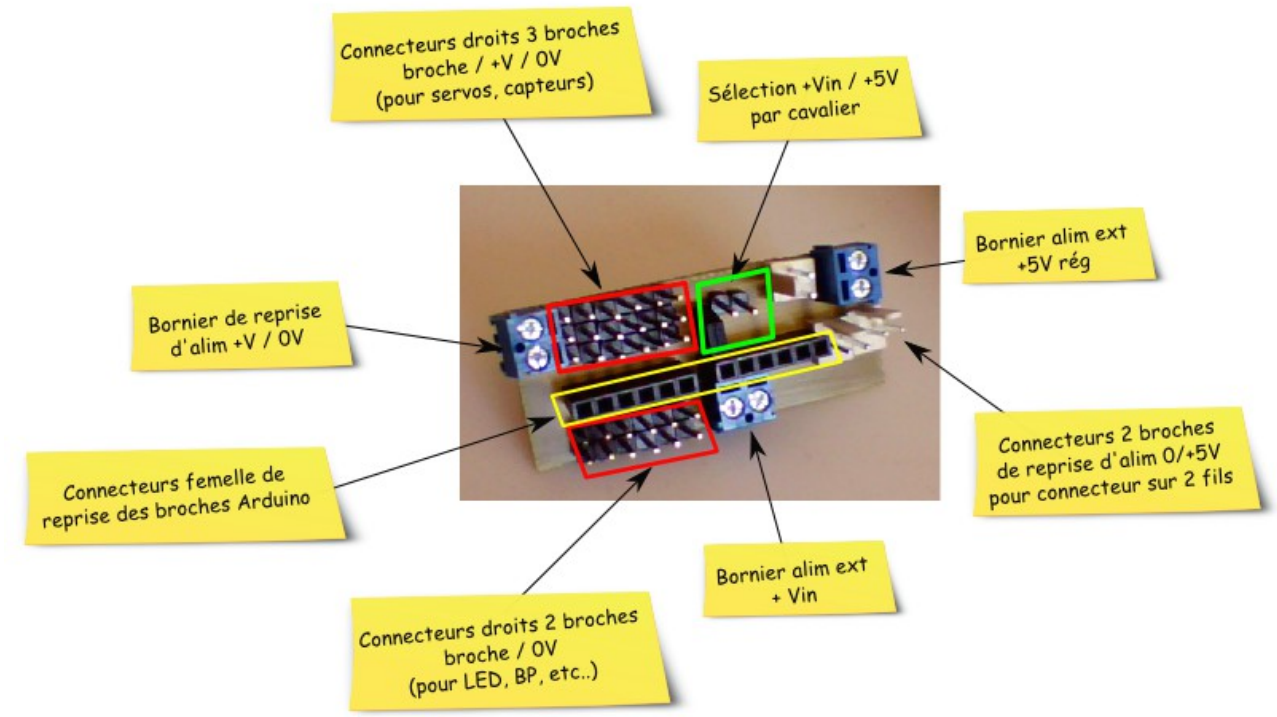
- mettre en place la carte Arduino



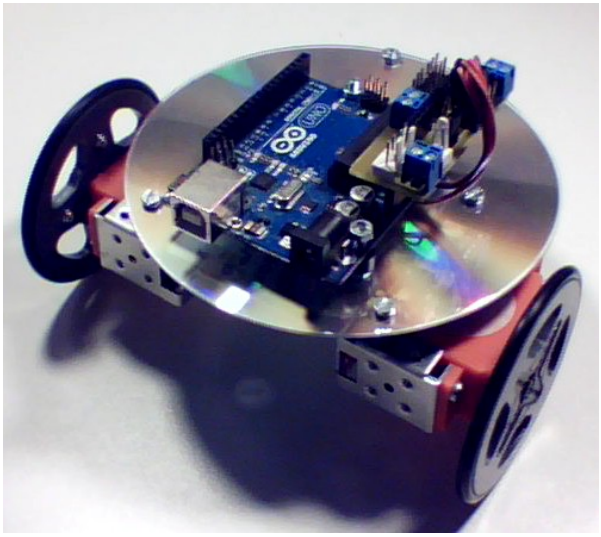
- mettre en place le mini-shield de connexion pour servomoteur en place – ce mini-shield est à monter vous-même – voir la fiche dédiée :

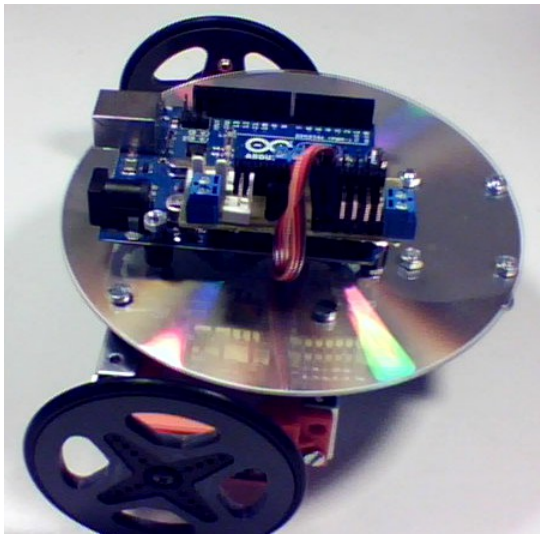


- pour mémoire, voici la description du mini-shield de connexion des servomoteurs



- connecter les 2 servomoteurs sur 2 des connecteurs droits en veillant à mettre le fil de commande sur le connecteur le plus près de la broche Arduino (fil orange ou blanc selon les modèles des servomoteurs). Noter que si vous ne disposez pas du mini-shield de connexion, vous pouvez faire un montage transitoire équivalent avec des jumpers.





- … et c'est tout !

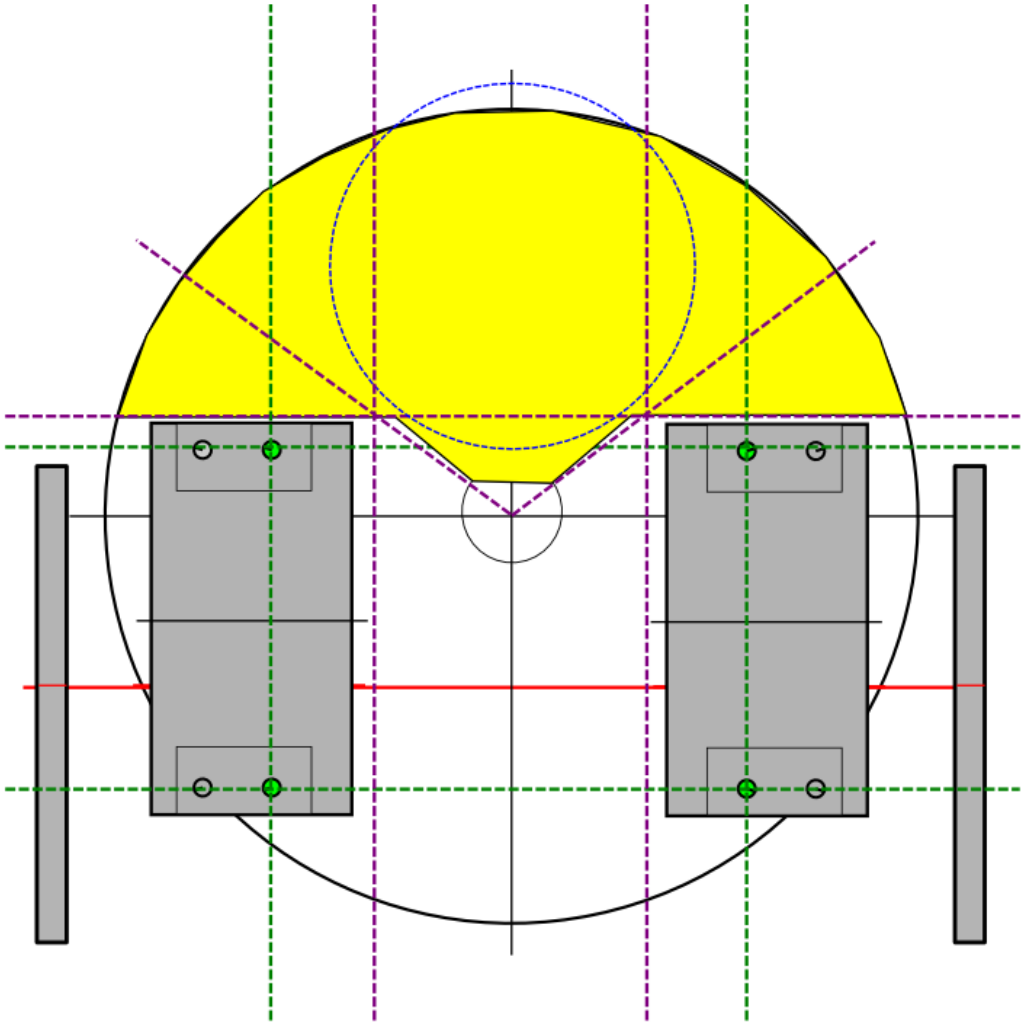
Vous avez envie de tester la motorisation du robot ?
çà y est, vous pouvez passer à l'action !

Les étapes suivantes consisteront à ajouter le plateau inférieur pour pouvoir utiliser une alimentation embarquée (6 piles ou 6 accus type LR6) et à installer les capteurs.

Fabrication du plateau inférieur du châssis

La découpe du plateau n'est pas obligatoire tant que l'on utilise le robot connecté au port USB. Par contre, il est nécessaire pour pouvoir utiliser des piles ou des accumulateurs afin d'utiliser le robot en autonomie.

- Imprimer le patron de traçage du CD fourni (fichier *.png de l'archive) Le CD fait 12cm :



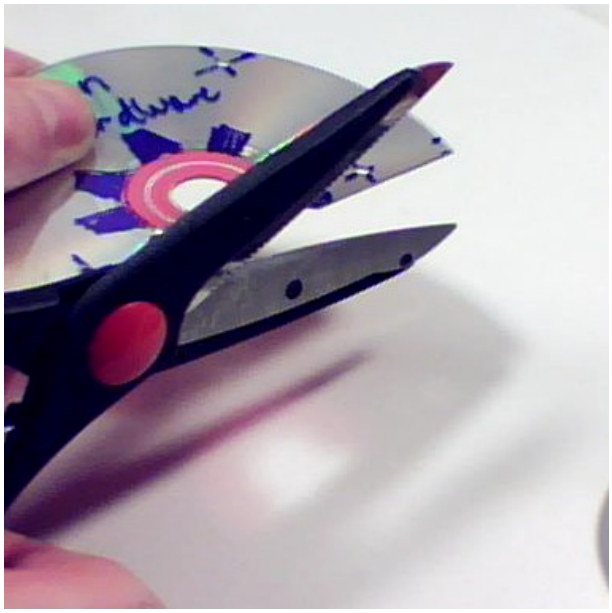
- Poser et centrer le second CD sur le patron de traçage
- Tracer au feutre indélébile (effaçable ensuite avec un chiffon imbibé d'alcool à brûler) :
  - 3 lignes horizontales (soit de haut en bas à 45mm (découpe), 50mm, 100mm)
  - 4 lignes verticales (soit de gauche à droite à 25mm, 40mm, 80mm et 95mm)
  - 2 lignes obliques (découpe) passant par le centre du CD et le point d'intersection des lignes violettes

La zone en jaune sur le patron correspond à la partie du CD à enlever.

  - tracer les lignes de « découpe » en pontillé...
  - repérer les trous de perçage

- Ensuite couper avec le gros ciseau le CD le long de la ligne de découpe horizontale, puis toujours avec le gros

ciseau, découper le CD le long des 2 lignes obliques :



Noter que la découpe du CD au ciseau se fait avec plus ou moins de bonheur : çà se fait bien avec certains CD bien souples, moins bien avec d'autres qui donnent des éclats. Dans ce cas, couper à la scie à métaux ou au cutter (attention les doigts ++) .

- Une fois la découpe terminée et le perçage des trous faits à la mini-perçeuse avec une mèche 3mm, on obtient :

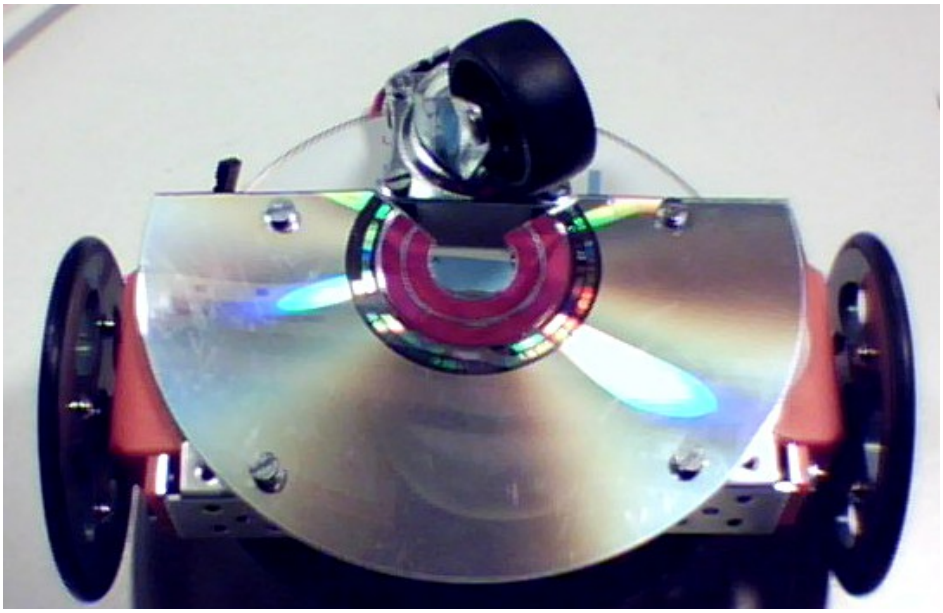
- Remarquer la « reprise » (involontaire d'ailleurs) du logo « Open Hardware » dans la forme obtenue !

Mais çà tombe bien finalement...

- Voilà, le plateau inférieur du châssis est terminé. Le plus dur est fait...

Montage du plateau inférieur

- Le plateau se fixe ensuite très simplement sur les cages Easy déjà en place, à l'aide de 4 vis M3x10mm, en retournant le robot :

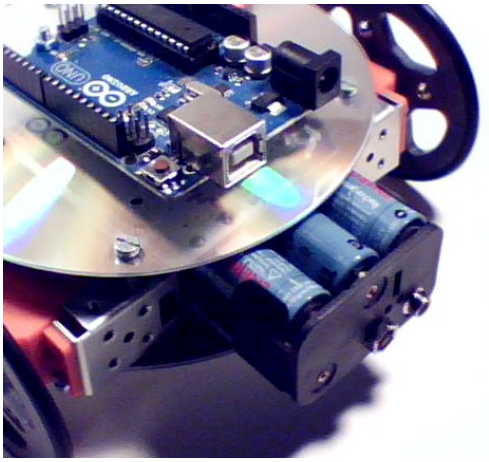
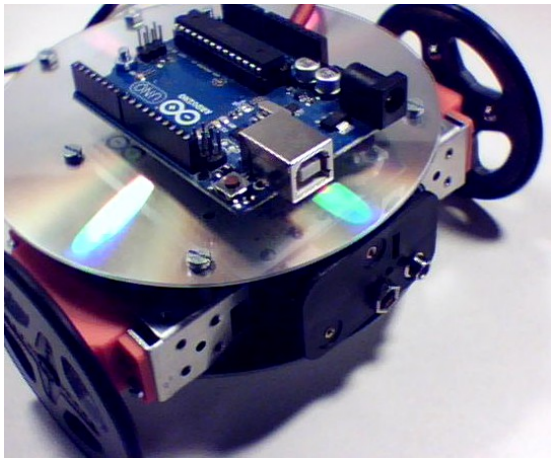


Montage de l'alimentation du robot

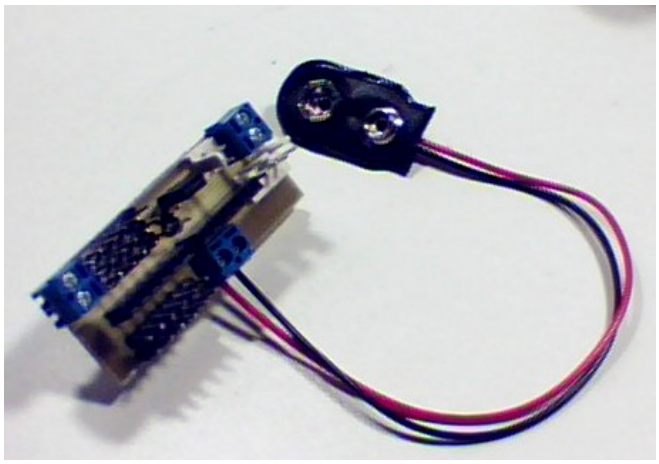
- Pour l'alimentation du robot, on utilise un bloc coupleur pour 6 piles ou accus LR6 et un connecteur à pression pour pile 9V :

- Mettre les 6 piles ou accu en place dans le bloc coupleur :

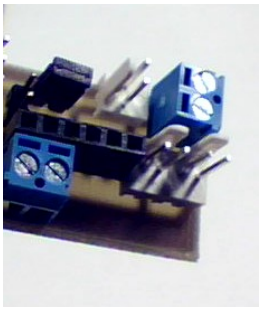
- puis, glisser le bloc entre les 2 cages Easy, entre le plateau supérieur et inférieur (au besoin insérer un petit bout de papier ou de carton entre le bloc et la cage pour éviter qu'il ne glisse trop facilement une fois en place) :

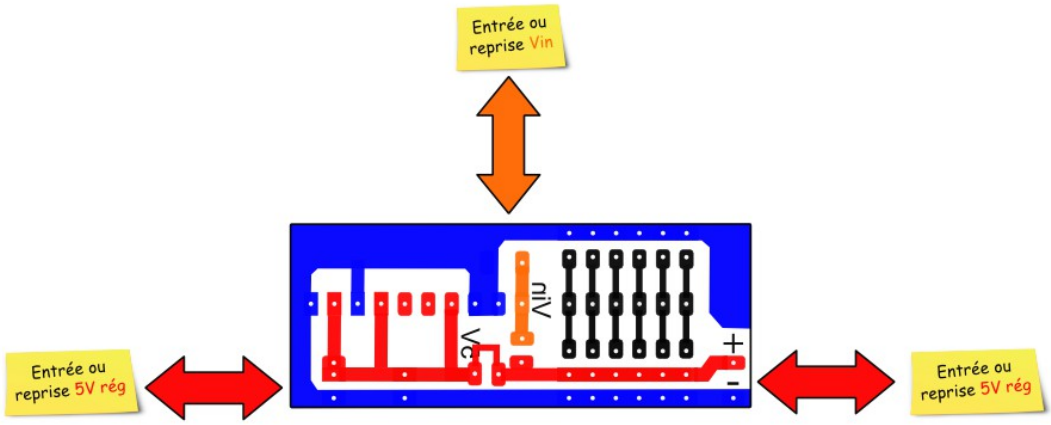
- une fois fait, connecter le connecteur 9V sur le bornier Vin du mini-shield « broches analogiques » :



- Noter que le cavalier de sélection de V+ doit être en place sur le mini shield pour une utilisation du 5V selon :



- on obtient ainsi la distribution suivante de l'alimentation sur le mini-shield :



- ensuite, il ne reste plus :
  - qu'à mettre place le mini-shield en place sur la carte Arduino,

- à connecter des servomoteurs sur les 2 premiers connecteurs 3 broches (fil blanc ou orange côté broche Arduino, fil noir ou marron côté extérieur et rouge en central)
- à connecter le connecteur 9V au bloc d'accumulateurs

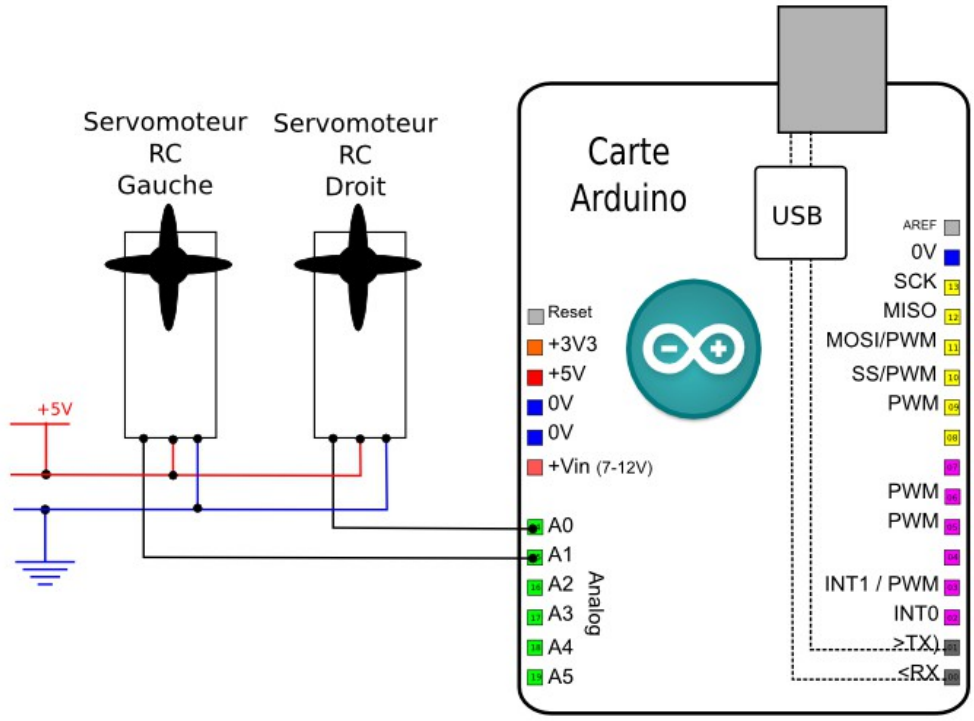
C'est terminé : le Simple Cd-Bot est opérationnel pour une utilisation en autonomie !
Le programmer par le port USB, le débrancher, puis le laissez se débrouiller tout seul !
Pour obtenir des comportements intéressants, il reste néanmois à ajouter des capteurs : c'est l'objet de l'étape suivante...

Etalonnage et test de la motorisation du SimpleCDbot

Le schéma électrique du câblage des moteurs du SimpleCDbot

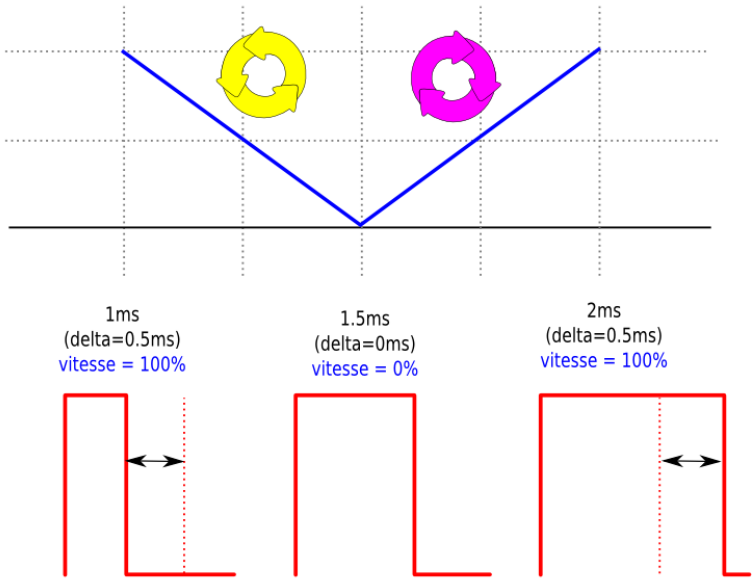
La connexion des moteurs du SimpleCDbot correspond au montage suivant :

- le servomoteur à rotation continue Droit est connecté sur la broche A0 utilisée en tant que broche numérique n°14
- le servomoteur à rotation continue Gauche est connecté sur la broche A1 utilisée en tant que broche numérique n°15



Rappel du principe de contrôle des servomoteurs à rotation continue

- Avec les servomoteurs à rotation continue, une seule broche permet le contrôle simultané de la vitesse et du sens de rotation :



- pour plus de détail, se reporter au support PDF de l'atelier consacré aux servomoteurs à rotation continue : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERSMoteursServosRotationContinue

Pré-requis : savoir installer une librairie Arduino : exemple de la librairie Utils

La librairie Utils

- La librairie Utils, dans laquelle j'ai rassemblé plusieurs fonctions utiles, permet de recevoir et d'extraire des paramètres numériques au sein de chaînes texte reçues notamment sur le port série.
- Le très gros avantage de cette librairie est de simplifier grandement le code qui serait beaucoup plus lourd pour

obtenir le même résultat uniquement avec des fonctions Arduino de base.

Télécharger la librairie

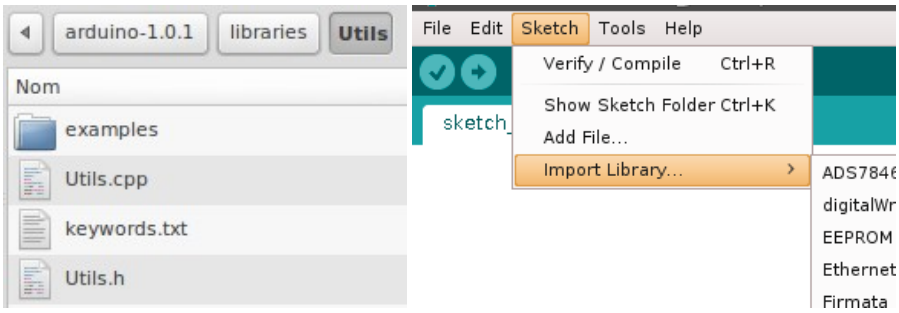
- Ma librairie Utils est disponible ici : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage

Documentation de la librairie

- http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.HomePage

Installation

- Télécharger l'archive. au format zip ou autre. L'extraire
- Vérifier que le nom du répertoire de la librairie est strictement le même que le nom du fichier *.h ou *.cpp principal. Corriger au besoin. Ici le nom est **Utils**
- Copier/coller le répertoire de la librairie dans le répertoire libraries de votre répertoire Arduino
- Relancer Arduino et vérifier que la librairie est présente dans le menu **Sketch > ImportLibrary.**



Le constructeur principal

- Le constructeur principal se nomme Utils et est de la forme :

```
Utils utils;
```

Fonctions de la librairie

Fonctions de réception de chaîne de caractères sur le port Série :

- String waitingString (boolean debugIn) : réception d'une chaine sur le port Série
- String waitingString () : réception d'une chaine sur le port Série
- void waitForString(String stringForWaitIn) : attente de la réception d'une chaine précise sur le port Série

Fonctions d'analyse de chaîne de caractères :

- String testInstructionString (String chaineTest, String chaineRefIn): extraction d'un paramètre texte
- boolean testInstructionLong (String chaineReception,String chaineTest, int nbParam, long paramsIn[]) : extraction d'un ou plusieurs paramètres entiers

Code d'exemple

Copier coller ce code dans le logiciel Arduino :

```
#include <Utils.h> // inclusion de la librairie
Utils utils; // déclare objet racine d'accès aux fonctions de la librairie Utils

String chaineReception=""; // déclare un String
long params[6]; // déclare un tableau de long - taille en fonction nombre max paramètres attendus

void setup() {

 Serial.begin(115200); // Initialisation vitesse port Série
  // Utiliser vitesse idem coté Terminal série
  Serial.println("Saisir une chaine de la forme FONCTION(valeur)"); // message initial

} // fin setup

void loop() {

    chaineReception=utils.waitingString();// sans debug

    if (chaineReception!="") { // si une chaine a été reçue

        if(utils.testInstructionLong(chaineReception,"FONCTION(",1,params)==true) { // si chaine FONCTION(valeur)
bien reçue

       Serial.println("Arduino a recu le parametre : " + (String)params[0]);
```

```
        } // fin si testInstructionLong==true
    } // fin // si une chaine a été reçue
} // fin loop
```

La librairie SimpleCDbot

- Pour simplifier l'utilisation du SimpleCDBot, j'ai écrit une librairie qui dispose de toutes les fonctions utiles pour assurer le contrôle du robot.

Les fonctions disponibles sont :

Contrôle des mouvements individuels des servomoteurs

- void servoAvant(int indiceServo, int vitesseIn)
- void avantDroit(int vitesseIn)
- void avantGauche(int vitesseIn)
- void servoArriere(int indiceServo, int vitesseIn)
- void arriereDroit(int vitesseIn)
- void arriereGauche(int vitesseIn)
- void arret()
- void arret(int dureeIn)

Contrôle des mouvements synchrone des servomoteurs

- void enAvant(int vitesseIn)
- void enAvant(int vitesseIn, int dureeIn)
- void enArriere(int vitesseIn)
- void enArriere(int vitesseIn, int dureeIn)
- void tourneDroite(int vitesseIn)
- void tourneDroite(int vitesseIn, int dureeIn)
- void tourneGauche(int vitesseIn)
- void tourneGauche(int vitesseIn, int dureeIn)

Fonction d'analyse de chaine

- void infosAnalyseChaine()
- void analyseChaine(String chaineRecue)

Fonctions de paramétrages internes

- void setDebug(boolean flagIn)

Etalonnage des mouvements individuels des servomoteurs

- void impulsServo(int indiceServo,int largeurImpulsIn)
- void impulsDroit(int largeurImpulsIn)
- void impulsGauche(int largeurImpulsIn)

Téléchargement

- la dernière version de la librairie est disponible ici en ligne sur le dépôt github : https://github.com/sensor56/SimpleCDBot (Lien de téléchargement de l'archive : https://github.com/sensor56/SimpleCDBot/archive/master.zip)
- Extraire l'archive et copier/coller le répertoire SimpleCDbot dans le répertoire Arduino/libraries/
- Relancer le logiciel Arduino : la librairie doit être présente dans le menu /skecth/Import Library/
- **Cette librairie nécessite également**
  - **la librairie Arduino Servo,** déjà installée.
  - **ma librairie personnelle Utils à installer de façon identique comme décrit précédemment,** ici sur github : https://github.com/sensor56/Utils (lien direct de téléchargement de l'archive ici : https://github.com/sensor56/Utils/archive/master.zip)

Documentation complète

- Voir ici pour le détail de la librairie : http://www.mon-club-elec.fr/pmwiki_reference_lib_arduino_perso/pmwiki.php?n=Main.SimpleCDbot

Etalonnage des servomoteurs à rotation continue du Simple CD-Bot

Ce qu'on va faire ici :

- L'étalonnage consiste à détecter le « neutre » des servomoteurs à rotation continue du Simple CDbot ainsi que les largeurs d'impulsion correspondant aux vitesses maximales avant et arrière. Ces valeurs de référence seront ensuite utilisées dans les programmes.
- Pour plus de détails sur la procédure d'étalonnage d'un servomoteur à rotation continue, se reporter au support PDF de l'atelier consacré aux servomoteurs à rotation continue : http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERSMoteursServosRotationContinue

Le code Arduino :

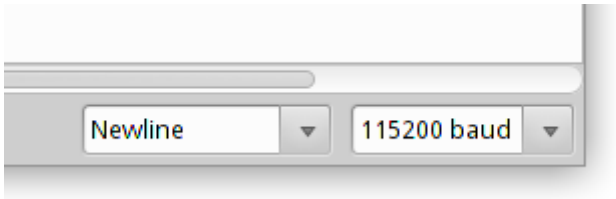
Disponible ici : https://raw.github.com/sensor56/SimpleCDBot/master/exemples/test_moteurs_serie.ino

```
//--- exemple librairie SimpleCDbot
// Programme pour le controle de 2 servomoteurs à rotation continue
// par le Terminal Serie
// par X. HINAULT - Tous droits réservés - licence GPL v3 - www.mon-club-elec.fr
// Septembre 2012

//--- entete déclarative = variables et constantes globales

//---- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo
#include <Utils.h> // librairie personnelle avec plusieurs fonctions utiles
#include <SimpleCDbot.h> // librairie personnelle avec fonctions de controle d'une robot à 2
servos RC


//--- variables pour réception chaine sur port Série
//int octetReception=0; // variable de réception octet
//char caractereReception=0; // variable de réception caractère
String chaineReception=""; // déclare un objet String vide

//--- variables et constantes pour les servomoteurs
const int Droit=0; // servo Droit a l'indice 0
const int Gauche=1; // servo Droit a l'indice 1

const int neutre[2]={1420,1420}; // largeur impulsion arret
const int maxAV[2]={1220,1620}; // largeur impulsion vitesse max en avant
const int maxAR[2]={1620,1220}; // largeur impulsion vitesse max en arriere

const int brocheServo[2]={14,15}; // broches des servomoteur

//int impulsServo=0; // largeur impulsion servomteur en µsecondes

//Servo servo[2]; // déclaration d'un objet servomoteur

Utils utils;

// SimpleCDbot(int brocheServoDroitIn, int maxARDroit, int maxAVDroit, int brocheServoGaucheIn,
int maxARGauche, int maxAVGauche, int neutreDroit, int neutreGauche); // constructeur avec
initialisation
SimpleCDbot robot(brocheServo[Droit], maxAR[Droit], maxAV[Droit],brocheServo[Gauche],
maxAR[Gauche], maxAV[Gauche], neutre[Droit], neutre[Gauche]);

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

   Serial.begin(115200); // initialise la vitesse de la connexion série
   //-- utilise la meme vitesse dans le Terminal Série

   robot.infosAnalyseChaine(); // infos sur les chaines reconnues par la fonction analyse chaine

} // fin de la fonction setup()


//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

      //--- réception de chaine sur le port série et analyse de la chaine --
      chaineReception=utils.waitingString(true); // appelle la fonction de réception sur le port
série

      if (chaineReception!="") robot.analyseChaine( chaineReception); // appelle la fonction
```

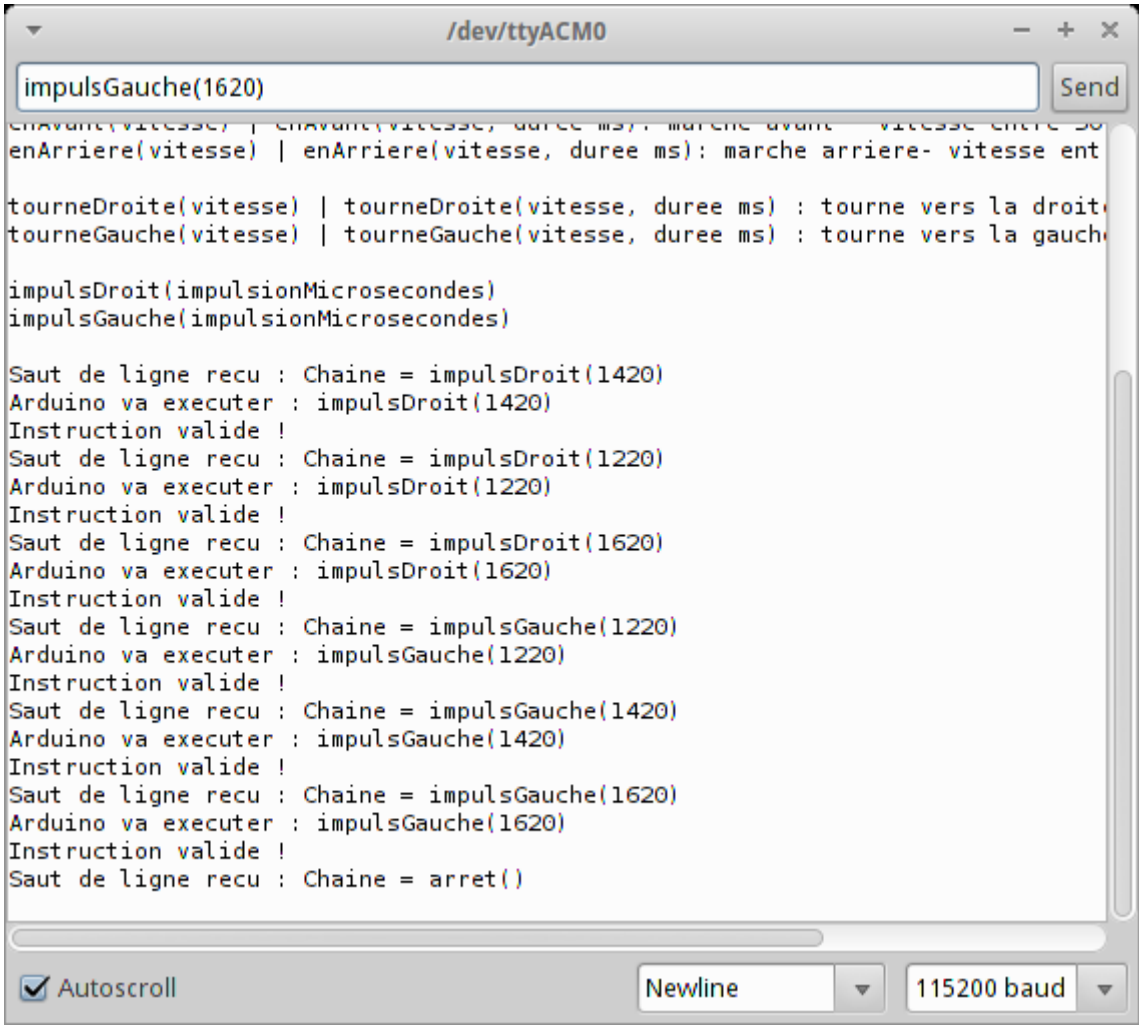
```
d'analyse de la chaine en réception


} // fin de la fonction loop()
```

Principe d'utilisation :

- Une fois la carte Arduino du robot programmée, lancer le Terminal Série
- Vérifier que l'option Newline est sélectionnée et que le débit est bien sur 115200



- Ensuite, à l'aide des chaînes impulsDroit(largeur) et impulsGauche(largeur) à saisir dans le champ texte du Terminal Série, rechercher successivement :
  - le neutre
  - les valeurs maximales de rotation avant et arrière des servomoteurs (centrées sur le neutre)
- ce qui donne :



A titre indicatif, je trouve neutre =1420, maxAvant=1220 et maxArriere=1620

Ces valeurs peuvent varier selon les modèles de servomoteurs, mais doivent similaires pour des servomoteurs du même modèle.

Test du fonctionnement des servomoteurs par le port Série

Ce qu'on va faire ici :

Ici, nous allons contrôler le SimpleCDbot directement depuis le port série. Ceci est assez simple grâce à la librairie dédiée au robot qui intègre toutes les fonctions nécessaires pour interpréter les chaines reçues sur le port série.

Les chaînes reconnues sont :

- arret() : stoppe les servomoteurs
- stop() : stoppe les servomoteurs
- debug(true/false)
- avantDroit(vitesse)
- avantGauche(vitesse)
- arriereDroit(vitesse)
- arriereGauche(vitesse)
- enAvant(vitesse) | enAvant(vitesse, duree ms): marche avant - vitesse entre 0 et 100%
- enArriere(vitesse) | enArriere(vitesse, duree ms): marche arriere- vitesse entre 0 et 100%
- tourneDroite(vitesse) | tourneDroite(vitesse, duree ms) : tourne vers la droite - vitesse entre 0 et 100%
- tourneGauche(vitesse) | tourneGauche(vitesse, duree ms) : tourne vers la gauche - vitesse entre 0 et 100%
- impulsDroit(impulsionMicrosecondes)
- impulsGauche(impulsionMicrosecondes)

Le code Arduino :

Utiliser les valeurs des neutres, de MaxAR et maxAV que vous avez obtenues lors de l'étalonnage

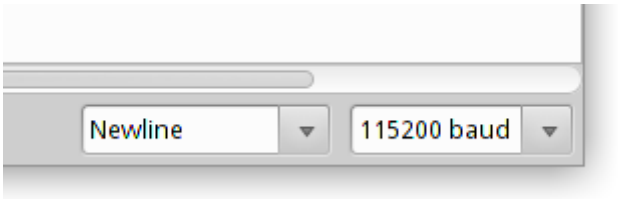
Disponible ici : https://raw.github.com/sensor56/SimpleCDBot/master/exemples/test_moteurs_serie.ino

```
//--- exemple librairie SimpleCDbot
// Programme pour le controle de 2 servomoteurs à rotation continue
// par le Terminal Serie
// par X. HINAULT - Tous droits réservés - licence GPL v3 - www.mon-club-elec.fr
// Septembre 2012

//--- entete déclarative = variables et constantes globales

//---- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo
#include <Utils.h> // librairie personnelle avec plusieurs fonctions utiles
#include <SimpleCDbot.h> // librairie personnelle avec fonctions de controle d'une robot à 2
servos RC


//--- variables pour réception chaine sur port Série
//int octetReception=0; // variable de réception octet
//char caractereReception=0; // variable de réception caractère
String chaineReception=""; // déclare un objet String vide

//--- variables et constantes pour les servomoteurs
const int Droit=0; // servo Droit a l'indice 0
const int Gauche=1; // servo Droit a l'indice 1

const int neutre[2]={1410,1400}; // largeur impulsion arret
const int maxAV[2]={1220,1620}; // largeur impulsion vitesse max en avant
const int maxAR[2]={1620,1220}; // largeur impulsion vitesse max en arriere

const int brocheServo[2]={14,15}; // broches des servomoteur

//int impulsServo=0; // largeur impulsion servomteur en µsecondes

//Servo servo[2]; // déclaration d'un objet servomoteur

Utils utils;

// SimpleCDbot(int brocheServoDroitIn, int maxARDroit, int maxAVDroit, int brocheServoGaucheIn,
int maxARGauche, int maxAVGauche, int neutreDroit, int neutreGauche); // constructeur avec
initialisation
SimpleCDbot robot(brocheServo[Droit], maxAR[Droit], maxAV[Droit],brocheServo[Gauche],
```

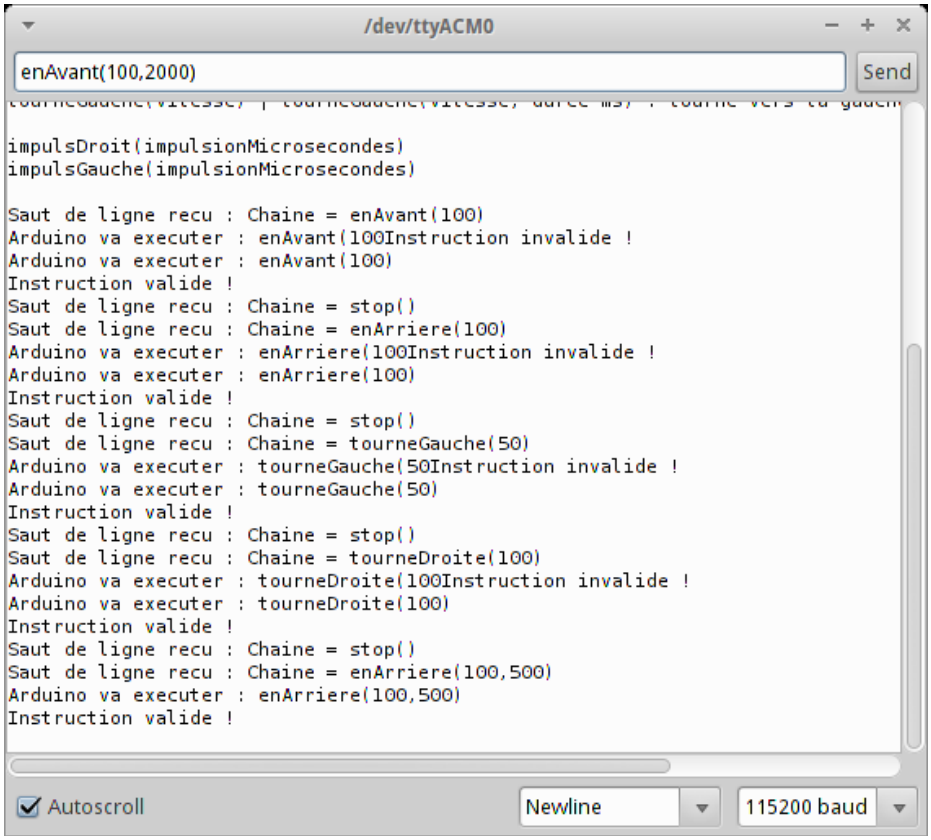
```
maxAR[Gauche], maxAV[Gauche], neutre[Droit], neutre[Gauche]);

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

   Serial.begin(115200); // initialise la vitesse de la connexion série
   //-- utilise la meme vitesse dans le Terminal Série

   robot.infosAnalyseChaine(); // infos sur les chaines reconnues par la fonction analyse chaine

} // fin de la fonction setup()


//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

      //--- réception de chaine sur le port série et analyse de la chaine --
      chaineReception=utils.waitingString(true); // appelle la fonction de réception sur le port
série

      if (chaineReception!="") robot.analyseChaine( chaineReception); // appelle la fonction
d'analyse de la chaine en réception



} // fin de la fonction loop()
```

Principe d'utilisation :

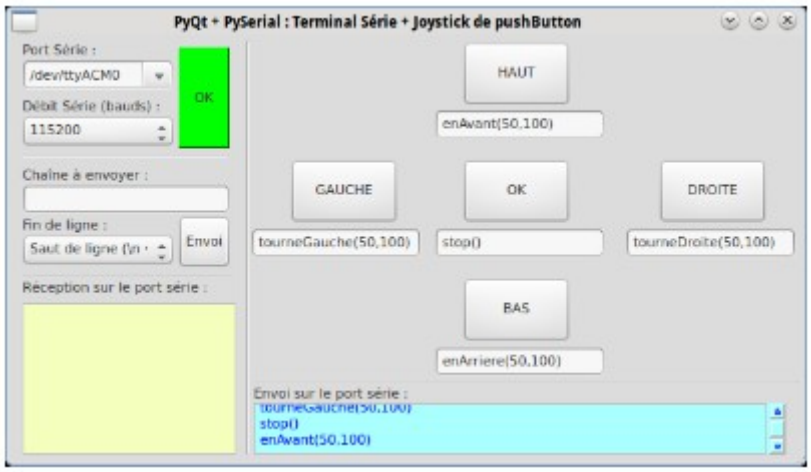
- Une fois la carte Arduino du robot programmée, lancer le Terminal Série
- Vérifier que l'option Newline est sélectionnée et que le débit est bien sur 115200



- Ensuite, à l'aide des chaînes listées, tester quelques mouvements, ce qui donne :



- Vous trouverez également ici une interface PyQt permettant le contrôle du robot par simples clics :
  - https://gist.github.com/sensor56/eedb08245c0377d32e98/download
  - http://www.mon-club-elec.fr//pmwiki_mon_club_elec/pmwiki.php?n=MAIN.PYQTLABSerialEnvoiClicPushButtonx5Joystick



Sous Gnu/Linux :

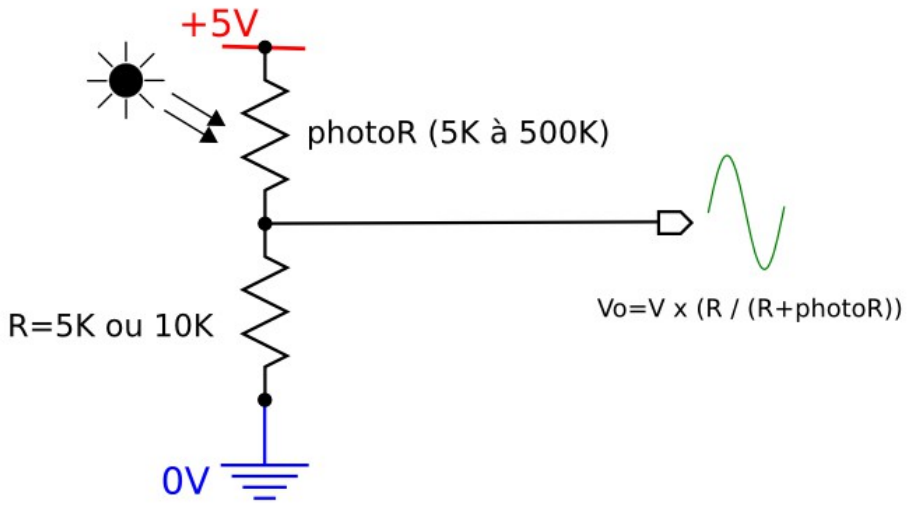
- installer les dépendances :
  - geany
  - python-qt4
  - python-serial
  - +/- pyqt4-dev-tools, qt4-designer
- télécharger l'application PyQt suivante : http://cloud-mon-club-elec.fr/public.php?service=files&t=1fc4bca052c3c5b6fa6144ed42d14c17&download

Capteurs de « niveau I »

Montage des capteurs de lumière du robot

Rappel : Principe d'utilisation d'une photo-résistance

Pour pouvoir utiliser efficacement une photo-résistance, il faut la mettre en série avec une autre résistance fixe afin de créer un diviseur de tension : de cette façon, la variation de la luminosité et donc de la résistance de la photo-résistance se traduira par une variation de tension entre les 2 résistances.



Rappel : Schéma théorique du montage pour une photo-résistance

On connecte tout simplement :

- en série la photo-résistance avec une résistance entre le 0V et le 5V.
- L'entrée analogique sera connectée au point de jonction entre la photo-résistance et la résistance.

La carte Arduino, par ailleurs, sera connectée au PC via le câble USB : on affichera de cette façon les résultats de la mesure côté PC.

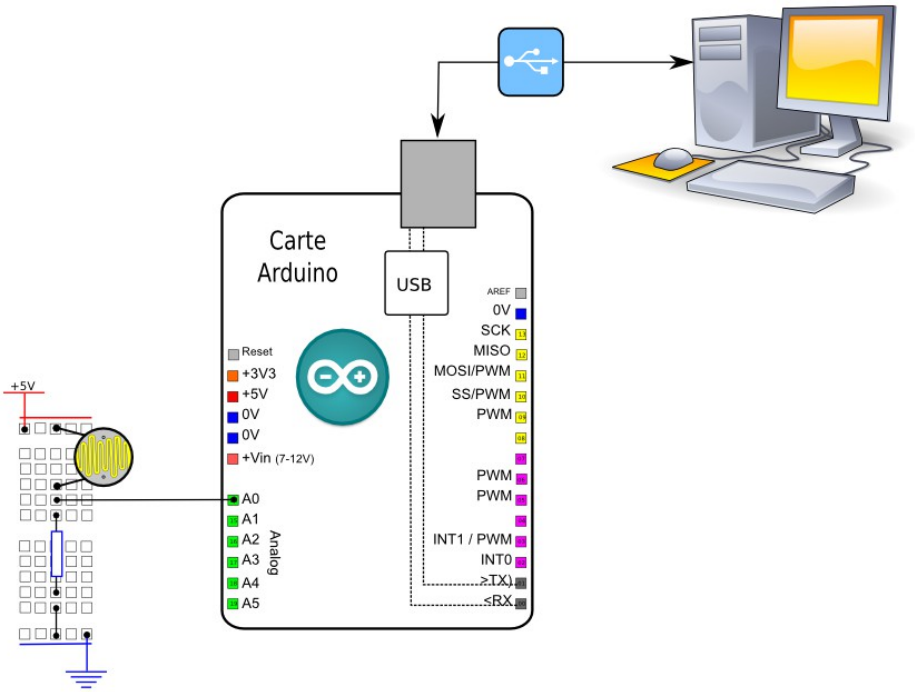
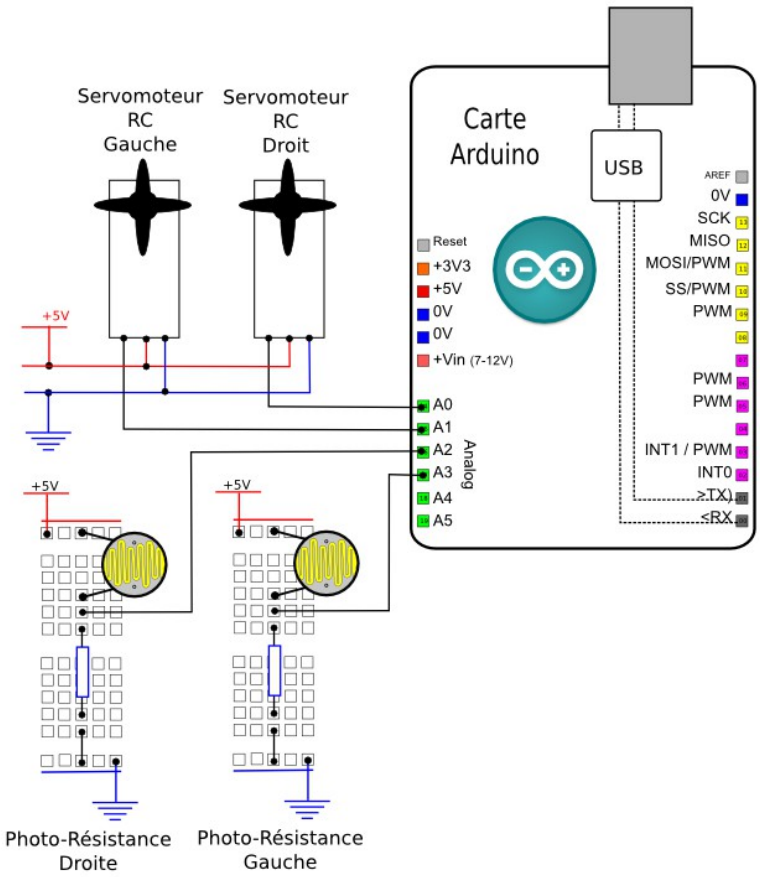


Schéma théorique du montage pour 2 photo-résistances dans le cas du SimpleCDbot

- On connecte chaque photo-résistance sur une broche analogique : A2 pour la droite et A3 pour la gauche.



En pratique

- Le mini-shield permet l'utilisation simplifiée de capteurs analogiques à l'aide de câble 3 brins JR. L'idée ici est de réaliser le montage à l'extémité du câble à l'aide de gaine thermo-rétractable afin de pouvoir ensuite n'avoir qu'à le connecter sur le connecteur analogique voulu (connecteur droit 3 contacts du shield) .
- 2 x LDR en série avec une résistance 10KOhms montée sur un câble 3 brins JR

Test simple des photo-résistances

Ce qu'on va faire ici :

- On va simplement afficher dans le Terminal Série, la valeur de la tension de sortie des 2 photo-résistances.

Le code Arduino :

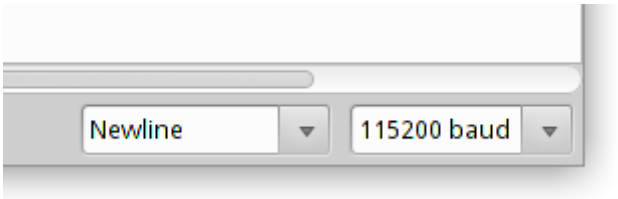
Disponible ici : https://raw.github.com/sensor56/SimpleCDBot/master/exemples/test_capteurs_lumiere.ino

```
// --- constantes des broches ---

const int RVarD=2; //declaration constante de broche analogique
const int RVarG=3; //declaration constante de broche analogique

// --- Déclaration des variables globales ---
int mesureBruteD=0;// Variable pour acquisition résultat brut de conversion analogique numérique
int mesureBruteG=0;// Variable pour acquisition résultat brut de conversion analogique numérique

int seuilD=200; // Variable fixant le seuil de détection de l'obscurité - à adapter
int seuilG=200; // Variable fixant le seuil de détection de l'obscurité - à adapter

//**************** FONCTION SETUP = Code d'initialisation *****

void setup()   { // debut de la fonction setup()

Serial.begin(115200); // initialise connexion série à 115200 bauds
// IMPORTANT : régler le terminal côté PC avec la même valeur de transmission


} // fin de la fonction setup()
```

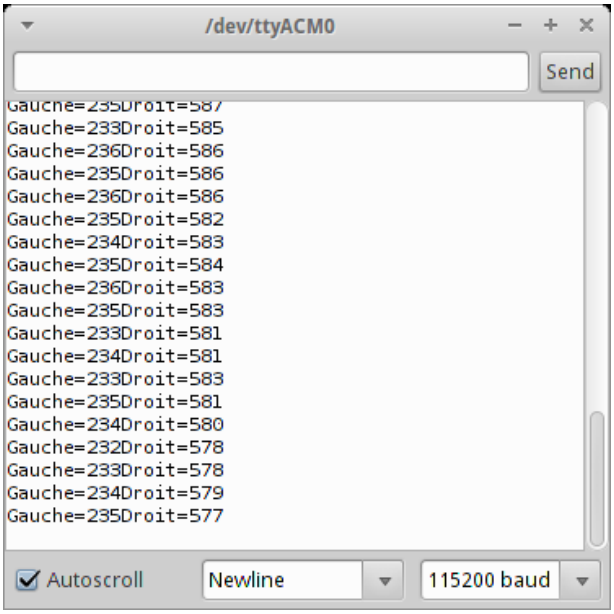
```
//***** FONCTION LOOP = Boucle sans fin *****


void loop(){ // debut de la fonction loop()

// acquisition conversion analogique-numerique (CAN) sur la voie analogique
mesureBruteD=analogRead(RVarD);
mesureBruteG=analogRead(RVarG);

// affiche valeur numerique entière ou à virgule au format décimal
Serial.print("Gauche="); Serial.print(mesureBruteG);
Serial.print("Droit="); Serial.print(mesureBruteD);
Serial.println();

if (mesureBruteD<seuilD) Serial.println("Obscurite detectee a droite"); // message si obscurité
détectée
if (mesureBruteG<seuilG) Serial.println("Obscurite detectee a gauche"); // message si obscurité
détectée

delay(500);

} // fin de la fonction loop() - le programme recommence au début de la fonction loop sans fin
```

Principe d'utilisation :

- Une fois la carte Arduino du robot programmée, lancer le Terminal Série
- Vérifier que l'option Newline est sélectionnée et que le débit est bien sur 115200



- Ensuite, on obtient l'affichage de la valeur brute de la mesure sur chacune des photo-résistances :



Le SimpleCDbot cherche la lumière

Ce qu'on va faire ici :

- Ici, nous allons contrôler les mouvements du robot en fonction de la lumière : le simple CD-Bot va se tourner vers la zone la plus lumineuse qu'il « voit ».

Le code Arduino :

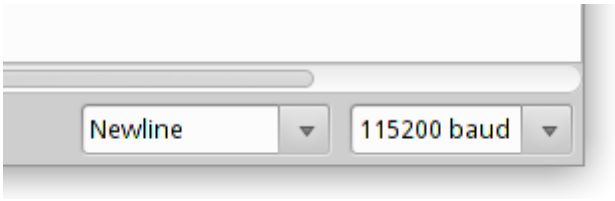
Disponible ici : https://github.com/sensor56/SimpleCDBot/blob/master/exemples/cherche_lumiere.ino

```
//--- exemple librairie SimpleCDbot
//--- contrle du CD-Bot par la lumière
// par X. HINAULT - Tous droits réservés - licence GPL v3 - www.mon-club-elec.fr
// Septembre 2012

//--- entete déclarative = variables et constantes globales

//---- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo

#include <Utils.h> // librairie personnelle avec plusieurs fonctions utiles

#include <SimpleCDbot.h> // librairie personnelle avec fonctions de controle d'une robot à 2
servos RC

// --- constantes des broches ---

const int RVarD=2; //declaration constante de broche analogique
const int RVarG=3; //declaration constante de broche analogique

//--- variables pour réception chaine sur port Série
//int octetReception=0; // variable de réception octet
//char caractereReception=0; // variable de réception caractère
String chaineReception=""; // déclare un objet String vide

//--- variables et constantes pour les servomoteurs
const int Droit=0; // servo Droit a l'indice 0
const int Gauche=1; // servo Droit a l'indice 1

const int neutre[2]={1410, 1400}; // largeur impulsion arret
const int maxAV[2]={1220,1620}; // largeur impulsion vitesse max en avant
const int maxAR[2]={1620,1220}; // largeur impulsion vitesse max en arriere

const int brocheServo[2]={14,15}; // broches des servomoteur

//int impulsServo=0; // largeur impulsion servomteur en µsecondes

// --- Déclaration des variables globales ---
int mesureBruteD=0;// Variable pour acquisition résultat brut de conversion analogique numérique
int mesureBruteG=0;// Variable pour acquisition résultat brut de conversion analogique numérique

int seuilD=200; // Variable fixant le seuil de détection de l'obscurité - à adapter
int seuilG=200; // Variable fixant le seuil de détection de l'obscurité - à adapter


//--- objets utiles ---
Servo servo[2]; // déclaration d'un objet servomoteur

Utils utils;

SimpleCDbot robot(brocheServo[Droit], maxAR[Droit], maxAV[Droit],brocheServo[Gauche],
maxAR[Gauche], maxAV[Gauche], neutre[Droit], neutre[Gauche]);

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

   Serial.begin(115200); // initialise la vitesse de la connexion série
   //-- utilise la meme vitesse dans le Terminal Série



} // fin de la fonction setup()


//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

  // acquisition conversion analogique-numerique (CAN) sur la voie analogique
  mesureBruteD=analogRead(RVarD);
  mesureBruteG=analogRead(RVarG);

  // affiche valeur numerique entière ou à virgule au format décimal
  Serial.print("Gauche="); Serial.print(mesureBruteG);
```

```
  Serial.print("Droit="); Serial.print(mesureBruteD);
  Serial.println();

  if (mesureBruteD<seuilD) Serial.println("Obscurite detectee a droite"); // message si
obscurité détectée
  if (mesureBruteG<seuilG) Serial.println("Obscurite detectee a gauche"); // message si
obscurité détectée

  mesureBruteD=mesureBruteD+130; // correction

  //-- code à exécuter en fonction des mesures --
  if (abs(mesureBruteD-mesureBruteG)<150)robot.arret();

  //--- si la photoR droite est plus éclairée que la gauche
  else if (mesureBruteD>mesureBruteG) robot.tourneDroite(30);

  //--- si la photoR gauche est plus éclairée que la droite
  else if (mesureBruteG>mesureBruteD) robot.tourneGauche(20);

    delay(20); // entre 2 mesures

} // fin de la fonction loop()
```

Principe d'utilisation :

- Une fois la carte Arduino du robot programmée, lancer le Terminal Série si vous souhaitez afficher des messages.
- Vérifier que l'option Newline est sélectionnée et que le débit est bien sur 115200



- Ensuite, le robot va s'orienter vers la zone la pls lumineuse.

Le SimpleCDbot fuit la lumière

Ce qu'on va faire ici :

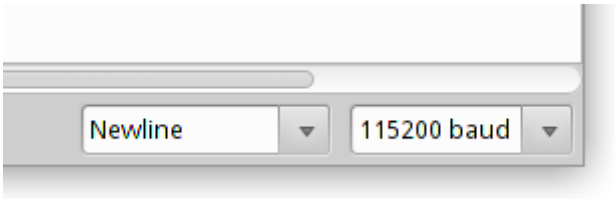
- Nous allons faire l'inverse : le robot va fuir la lumière.

Le code Arduino :

```

```

Principe d'utilisation :

- Une fois la carte Arduino du robot programmée, lancer le Terminal Série
- Vérifier que l'option Newline est sélectionnée et que le débit est bien sur 115200



- Ensuite,

Les capteurs de choc : les micro-rupteurs de détection de contact

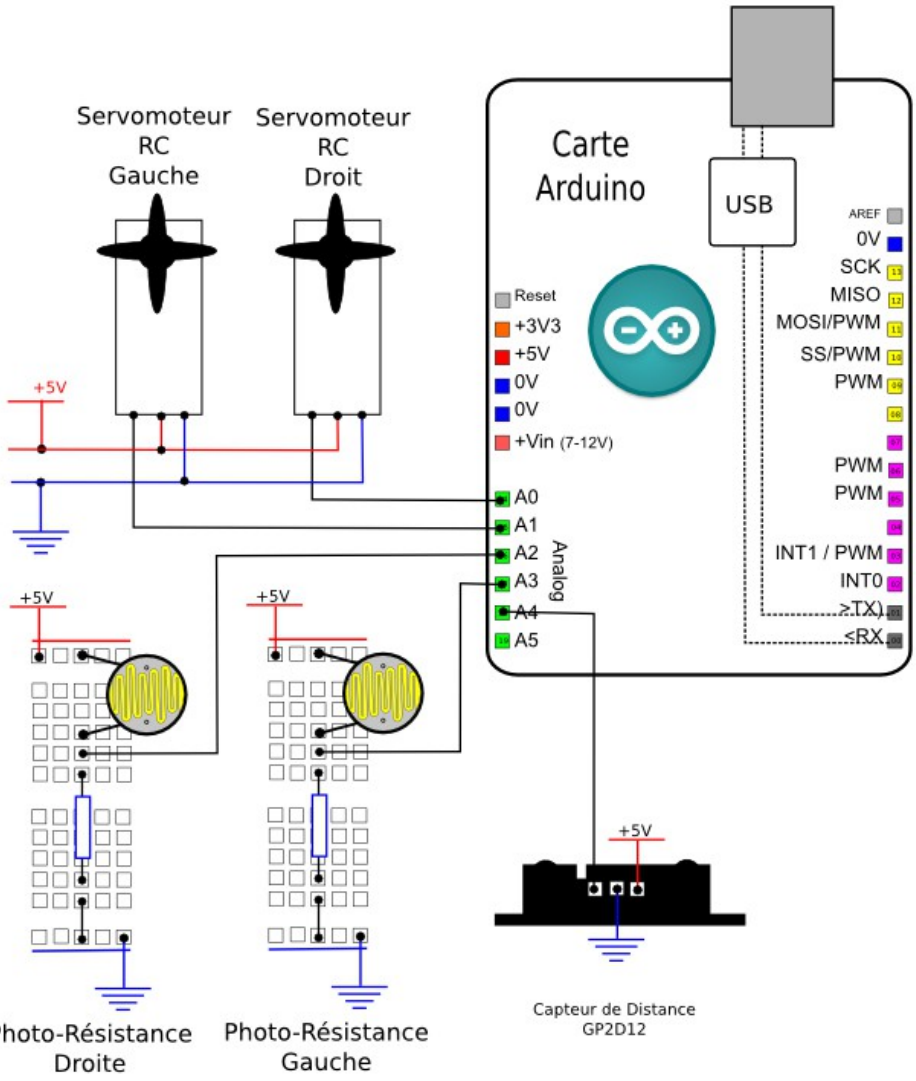
à venir...

Capteurs de « niveau II »

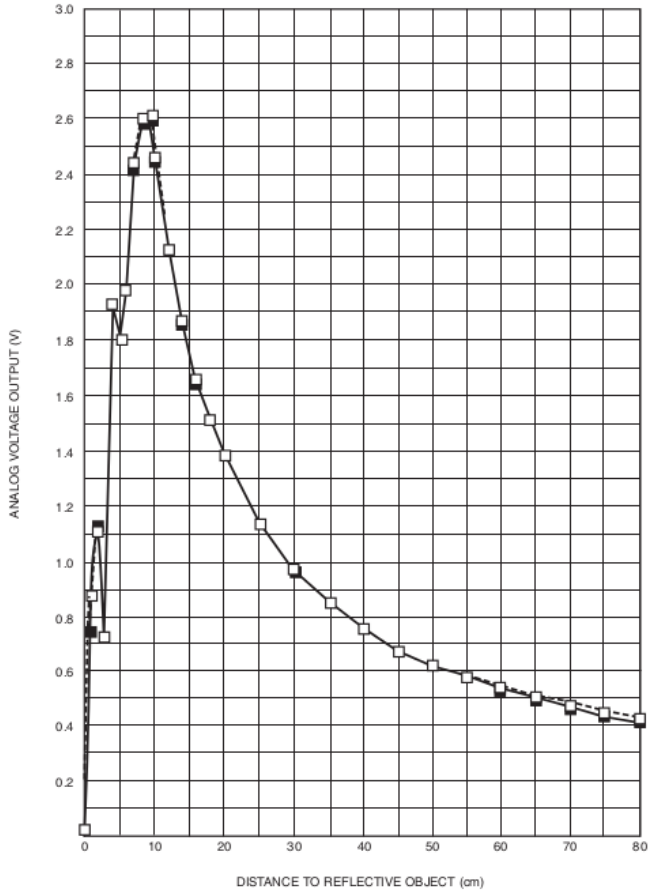
Capteur de distance analogique à infra rouge

Schéma théorique du montage pour capteur GP2D12 dans le cas du SimpleCDbot

- On connecte le GP2D12 sur la broche analogique A4 :



Courbe de réponse du capteur de distance analogique GP2D12 (non linéaire)



Détection d'obstacle

Ce qu'on va faire ici :

- On va utiliser le GP2D12 pour réaliser un comportement de détection d'obstacle.

Le code Arduino :

Disponible ici : https://github.com/sensor56/SimpleCDBot/blob/master/exemples/mesure_distance.ino

```
//--- Programme pour le controle de 2 servomoteurs à rotation continue
// par le Terminal Serie
// par X. HINAULT - Tous droits réservés - licence GPL v3 - www.mon-club-elec.fr

//--- entete déclarative = variables et constantes globales

//---- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo

#include <Utils.h> // librairie personnelle avec plusieurs fonctions utiles

#include <SimpleCDbot.h> // librairie personnelle avec fonctions de controle d'une robot à 2
servos RC


//--- variables pour réception chaine sur port Série
//int octetReception=0; // variable de réception octet
//char caractereReception=0; // variable de réception caractère
String chaineReception=""; // déclare un objet String vide

//--- variables et constantes pour les servomoteurs
const int Droit=0; // servo Droit a l'indice 0
const int Gauche=1; // servo Droit a l'indice 1

const int neutre[2]={1410,1400}; // largeur impulsion arret
const int maxAV[2]={1230,1730}; // largeur impulsion vitesse max en avant
const int maxAR[2]={1730,1230}; // largeur impulsion vitesse max en arriere

const int brocheServo[2]={14,15}; // broches des servomoteur
```

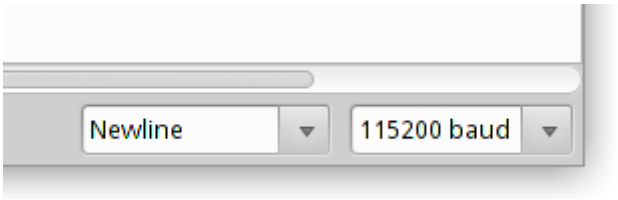
```
//--- pour le GP2D12 ---
const int GP2D12=A2; //declaration constante de broche analogique

// --- Déclaration des variables globales ---
int mesure_brute=0;// Variable pour acquisition résultat brut de conversion analogique numérique
float mesuref=0.0;// Variable pour calcul résultat décimal de conversion analogique numérique

int distance; // distance référence

//--- tableau de valeurs en millivolts pour distance de 5 en 5cm de 10 à 80 cm pour le capteur
GP2D12
// 1ère valeur a l'index 0
int calibrage[]={ // valeurs en mV
2370, // 10 cm - index 0
1700, // 15 cm - index 1
1300, // 20 cm - index 2
1100, // 25 cm - index 3
950, // 30 cm - index 4
830, // 35 cm - index 5
720, // 40 cm - index 6
650, // 45 cm - index 7
580, // 50 cm - index 8
540, // 55 cm - index 9
500, // 60 cm - index 10
480, // 65 cm - index 11
450, // 70 cm - index 12
430, // 75 cm - index 13
410, // 80 cm - index 14
400, // au delà 80cm - index 15
}; // fin du tableau

//int impulsServo=0; // largeur impulsion servomteur en µsecondes

Servo servo[2]; // déclaration d'un objet servomoteur

Utils utils;

SimpleCDbot robot(brocheServo[Droit], maxAR[Droit], maxAV[Droit],brocheServo[Gauche],
maxAR[Gauche], maxAV[Gauche], neutre[Droit], neutre[Gauche]);

boolean debug=true;
boolean etatMoteurs=false; // temoin etat moteur
boolean detectObstacle=false; // témoin détection obstacle

//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

   Serial.begin(115200); // initialise la vitesse de la connexion série
   //-- utilise la meme vitesse dans le Terminal Série


  robot.infosAnalyseChaine(); // info sur les chaines reconnues par analyseChaine()


} // fin de la fonction setup()


//--- la fonction loop() : exécutée en boucle sans fin
void loop() {

      //--- réception de chaine sur le port série et analyse de la chaine --
      chaineReception=utils.waitingString(true); // appelle la fonction de réception sur le port
série
      //if (chaineReception!="") analyseChaine( chaineReception); // appelle la fonction
d'analyse de la chaine en réception

      //if (chaineReception!="") robot.analyseChaine( chaineReception); // appelle la fonction
d'analyse de la chaine en réception
      // la fonction analyse chaine "décode" la chaine et appelle la fonction de la lib
RobotCCx2 voulue si la chaine est valide

      if (chaineReception!="") robot.analyseChaine( chaineReception); // appelle la fonction
d'analyse de la chaine en réception


    mesuref=moyenneMesure(30, GP2D12); // réalise moyenne de 30 mesures analogiques sur voie
GP2D12
```

```
    distance=distanceGP2D12(mesuref); // renvoie la valeur de la distance correspondante à la
valeur tension

    //----- affichage du résultat --------

    if (distance==0) {// si valeur ne correspond pas au calibrage
      if (debug) Serial.println("Distance hors plage de mesure");

      if ( (etatMoteurs==false) || (detectObstacle==true) ) { // allume moteur
        robot.enAvant(100);
        etatMoteurs=true;
        detectObstacle==false;
      }

    }
    else { // si valeur distance calculée
      if (debug) Serial.print("Distance comprise entre"), Serial.print(distance), Serial.print("
cm et ");
      if (debug) Serial.print(distance+5),Serial.println(" cm.");

      if (distance<20) {

             if (etatMoteurs==true) { // si moteur allumé
              robot.arret(1000);
              robot.enArriere(50,1000); // recule 1 seconde
              //etatMoteurs=false; // on laisse à true jusqu'à distance sup 30 suivante..
              detectObstacle=true;

            } // fin if moteur==true

      } // fin if <30
       else if (distance>=20) {

            if (detectObstacle==true) { // si moteur allumé = si gestion distance <30 toujours
en cours
             robot.tourneDroite(50,1000);
             etatMoteurs=false;
            detectObstacle=false; // plus d'obstacle
            } // fin if

                if (etatMoteurs==false) { // allume moteur
               robot.enAvant(100);
               etatMoteurs=true;
                 } // fin if moteurs false



       } // fin else if distance


    }// fin else

    delay (250); // entre 2 mesures

} // fin de la fonction loop()


//======== autres fonctions ==========

// --- fonction réalisant une moyenne de n mesures analogiques sur voie analogique

float moyenneMesure(int nombreMesure, int voie) { // --- fonction réalisant une moyenne de n
mesures analogiques

  //---- variables locales ---
  int mesure_can=0;
  long cumul_can=0;
  int moyenne_can=0;
  float mesure_canf=0;

  //----- calcul d'une moyenne de plusieurs mesures -------
  for (int i=0; i<nombreMesure; i++) // répète la mesure n fois
  {
    mesure_can=analogRead(voie);
    cumul_can=cumul_can+mesure_can;
  }
```

```
  if (debug) Serial.print("cumul="), Serial.println(cumul_can);

  // calcul moyenne
  moyenne_can=cumul_can/nombreMesure;
  cumul_can=0; //RAZ cumul...
    if (debug) Serial.print("moyenne="), Serial.println(moyenne_can);

  // mesure tension correspondante en mV
  mesure_canf=float(moyenne_can);
  mesure_canf=mesure_canf*5000.0;
  mesure_canf=mesure_canf/1023.0;

    if (debug) Serial.print("tension="), Serial.println(mesure_canf);

  return (mesure_canf); // valeur renvoyée par la fonction

} // --- fin fonction moyenneMesure

//------------- fonction renvoyant la distance en fonction de la mesure analogique GP2D12 ----
// cette fonction est nécessaire car la sortie analogique du GP2D12 n'est pas linéaire

int distanceGP2D12(float mesure_float) {

   int dist=0; // RAZ variable locale distance

  // ------- détermination de la distance à partir du tableau de reference

  for (int j=0; j<=14; j++) { // teste les valeurs de calibrage

    if ((int(mesure_float)<=calibrage[j]) && (int(mesure_float)>calibrage[j+1])) { // si la
mesure comprise entre 2 valeurs
      dist=10+(j*5); // calcule de la distance en cm à partir de l'index courant
    }

  } // fin boucle test calibrage

  return (dist); // valeur renvoyée par la fonction

} // ----------- fin fonction analyse distanceGP2D12 ----
```

Principe d'utilisation :

- Une fois la carte Arduino du robot programmée, lancer le Terminal Série si on souhaite visualiser des messages dans le Terminal Série
- Vérifier que l'option Newline est sélectionnée et que le débit est bien sur 115200



- Ensuite, laisser le robot se dépalcer en autonomie : il s'arrête à la détection d'un obstacle, recule puis tourne, et ainsi de suite.

Capteur de suivi de ligne et de détection de « trous »

à venir...