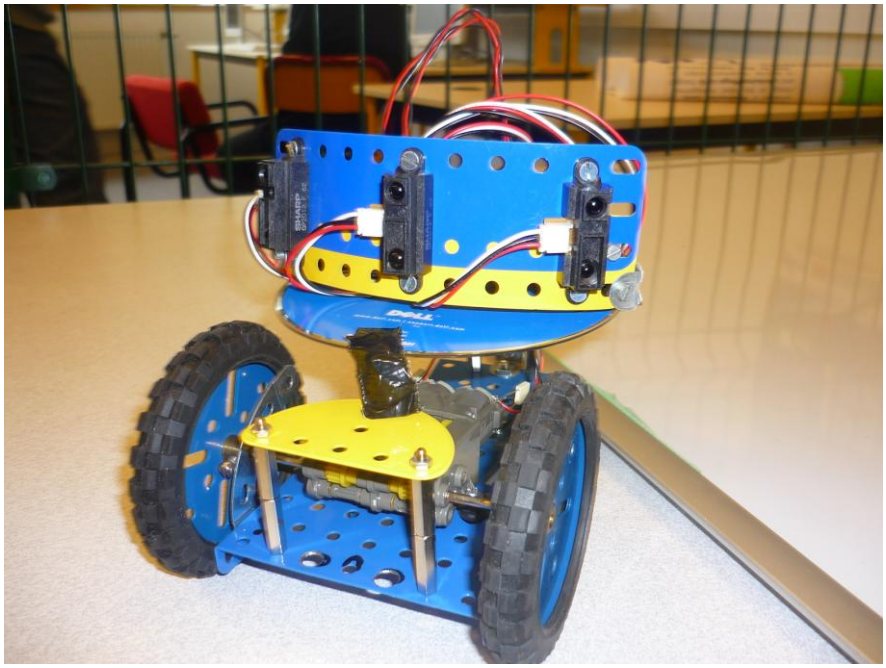
INSA ROUEN
À taille humaine,
à l'échelle du monde

Projet de Physique P6-3

STPI/P6-3/2010 – 41

Robot suiveur à asservissement de distance



Etudiants :

Rémi BRIGAUD **Jérôme FELIX**

Benoît POULIQUEN **Jean-Claude TWAGIRAMUNGU**

Enseignant-responsable du projet :

Fabrice DELAMARRE

Cette page est laissée intentionnellement vierge.

Date de remise du rapport : ***18/06/10***

Référence du projet : ***STPI/P6-3/2010 – 41***

Intitulé du projet : ***Robot suiveur à asservissement de distance***

Type de projet : ***Expérimental***

Objectifs du projet :

Concevoir un robot capable de suivre un objet en mouvement placé devant lui. Il doit être capable de tourner si l'objet change de trajectoire et devra s'arrêter lorsque l'objet ne bouge plus et que le robot a atteint une distance assez proche de ce dernier.

Remerciements :

A M. Delamarre pour son enseignement et son aide.

A M. Jean-Claude Brigaud pour son aide et son expérience professionnelle.

Si existant, n° cahier de laboratoire associé : ***xxx***

INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE ROUEN

Département Sciences et Techniques Pour l'Ingénieur

BP 8 – place Emile Blondel - 76131 Mont-Saint-Aignan - tél : 33 2 35 52 83 00 - fax : 33 2 35 52 83 69

Table des matières

NOTATIONS, ACRONYMES

Datasheet : « Document regroupant les performances et caractéristiques d'un composant (ex. un composant électronique) ou un module (ex. un module d'alimentation, un modem HF) avec tous les détails nécessaires pour permettre à un concepteur d'utiliser ce composant/module dans le développement d'un système. » (Wkipedia)

Pin : Synonyme de broche ou patte d’un composants

1. INTRODUCTION ET CAHIER DES CHARGES

Dans le cadre de notre projet de P6-3, nous avons réalisé un robot suiveur à asservissement de distance. Ce travail a été réalisé dans les salles dédiées à la robotique et l'électronique dans le centre EEAS – Hervé Haudiquet sous l'enseignement de Fabrice Delamarre. Nous avions sur place tout le matériel nécessaire pour réaliser notre robot tel que les composants nécessaires, outils et machines. Nous pouvions ainsi utiliser l'atelier de soudure, scies, pinces et nous procurer vis, écrous, … M. Delamarre était bien sûr là si nous avions des questions, mais le but de ce projet était qu'on arrive à le finaliser de la manière la plus autonome possible. Une heure et demie étaient dédiée par semaine pour pouvoir travailler sur le projet tous ensembles, pour nous cette séance était le mardi à 13h.

Pour réaliser notre projet, M. Delamarre nous a imposé comme cahier des charges d'utiliser un maximum de matériaux recyclables afin d'amortir les coups et de préserver l'environnement. Il nous a ainsi proposé d'utiliser des CDs comme base pour notre robot. Le microcontrôleur imposé est l'Arduino Duemilanove de la marque Italienne Arduino et le composant réalisant la fonction de double pont en H est le L293 de chez Texas Instruments. Le robot doit être capable de pouvoir suivre une main que l'on bouge devant lui.

2. METHODOLOGIE / ORGANISATION DU TRAVAIL

Dès le première séance nous avons décidé de la répartition du travail. Il a tout d'abord été décidé que Rémi Brigaud serait le leader du groupe qui devrait gérer la communication au sein du groupe ainsi qu'avec M. Delamarre. Nous devions par exemple envoyer chaque semaine un cahier de suivi donnant état de l'avancement de projet, du travail réalisé entre deux séances ainsi que le travail qui serait à réaliser pour la séance suivante.

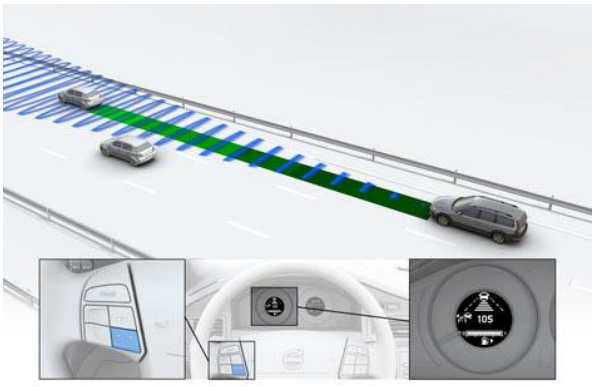
L'organisation du travail a été faite par affinité : Jérôme et Jean-Claude, tous deux dans la thématique ASI ont été chargés de s'occuper de la programmation. Rémi s'est occupé de la réalisation en 3 dimensions du robot car il savait se servir du logiciel Solid Works et enfin Benoît s'est occupé de la conception mécanique avec l'aide de Rémi. Pour ce qui est de la partie électronique et de la soudure nous n'avions pas décidé dès la première séance qui s'en occuperait. En effet, nous ne nous rendions pas encore compte en quel travail cela consisterait à ce moment. Finalement c'est Rémi qui s'en est chargé car il avait déjà fait de l'électronique au lycée, il était donc le plus à même de réaliser ce travail.

3. ETAT DE L'ART

Le robot que nous allons concevoir est un robot suiveur à asservissement de distance. L'asservissement de distance est une solution technologique que l'on trouve de plus en plus communément dans le domaine de l'automobile. En effet, les constructeurs se penchent de plus en plus sur la manière de gérer les circulations denses comme sur les autoroutes afin d'éviter de créer des embouteillages. Conduire sur route dense est très fatiguant pour l'homme, à qui il arrive facilement de faire une petite faute qui peut causer ces embouteillages. Il était donc essentiel pour l'industrie automobile de trouver une solution pour contrôler les distances inter-véhicules afin d'adapter automatiquement la vitesse du véhicule et garantir ainsi une circulation du trafic plus fluide.

L'industrie automobile à déjà commercialisé des solutions pour ce genre de problèmes. Ainsi, on découvre sur le marché divers déclinaisons de l'ACC (Autonomous/Adaptive Cruise Control). C'est la solution la plus simple et la plus répandue aujourd'hui. Cette solution est basée soit sur la technologie radar soit sur la technologie laser. La technologie laser est moins coûteuse que son homologue, mais elle est moins performante : lorsque la voiture suivie est sale (elle ne réfléchit pas le laser) et lorsque le temps est mauvais.

Dans les deux cas, le capteur permet de calculer la distance qui vous sépare du véhicule qui est devant vous et calcule sa vitesse. L'ACC combine à la fois un régulateur de vitesse et un régulateur de distance. L'ACC peut agir électroniquement sur le système de distribution du moteur et le dispositif de freinage. Il existe différents types d'ACC pouvant combiner ou non signaux sonores et actions sur votre vitesse. La solution la plus répandue fonctionne comme ceci : Vous spécifiez la vitesse que vous souhaitez garder comme sur régulateur de vitesse classique. Lorsque vous vous approchez trop près d'un véhicule de votre voie, le dispositif ralentit automatiquement votre voiture. Lorsque la voie est de nouveau libre, le régulateur vous ramène automatiquement à la vitesse définie. Selon les modèles, une simple pression sur l'accélérateur ou les freins permet de désactiver le système. Cette technologie permet d'accroître le confort de conduite, la sécurité, de faire des économies de carburant, et protège ainsi l'environnement. On l'appelle aussi régulateur de vitesse actif. On commence aujourd'hui à mettre ce type de technologie autant à l'avant qu'à l'arrière.

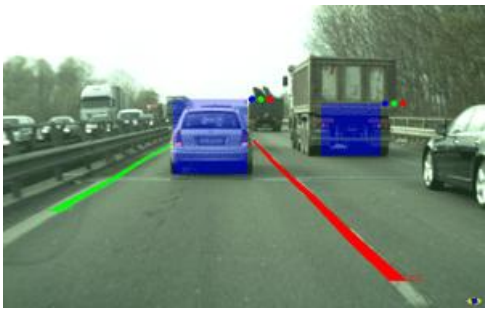


Parallèlement on développe les systèmes Stop & Go, ces systèmes sont basés sur la même technologie de mesure que l'ACC et permettent lors d'embouteillages d'arrêter et de démarrer automatiquement la voiture. Cela décharge le conducteur de faire des démarrages et arrêts qui sont fatigants lorsqu'ils sont répétés. Le régulateur actif de vitesse-distance avec fonction stop & go sera par exemple proposé sur les prochaines BMW série 5. Cette série 5 embarque aussi un avertisseur de collision et peut anticiper la vitesse à adopter lorsqu'un véhicule se rabat grâce à son système radar qui possède un cône de mesure plus évasé. Aujourd'hui, on trouve également des voitures qui peuvent faire, ou feront prochainement, des créneaux toutes seules et cela chez de nombreux constructeurs : BMW, Mercedes, Toyota, Lexus, Volkswagen, Citroën … ce ne sont pas les exemples qui manquent. Et c'est ce à quoi aspire l'industrie et la recherche : des voitures de plus en plus intelligentes. Ces voitures, à l'aide de nombreux capteurs, se rapprocheront de plus en plus de la voiture autonome.

C'est ce qu'essaie de réaliser le laboratoire VisLab de l'Université de Parme avec la BRAiVE. La BRAiVE, dont le nom vient de la combinaison de BRAin et driVE, est la voiture intelligente qu'est en train de mettre au point l'équipe du professeur Alberto Broggi. Cette voiture, qui embarque de nombreux capteurs, ordinateurs et calculateurs, possède les solutions d'ACC, de stop & go et peut s'arrêter automatiquement lorsqu'un piéton traverse, mais elle va aussi beaucoup plus loin que cela.

Ainsi la BRAiVE, à l'aide de ses caméras arrière, frontale, latérales et intégrées aux rétroviseurs, possède une vue à 360° de son environnement. Ces caméras, associées à des lasers, permettent à la voiture d'appréhender entièrement son environnement. Toutes les données recueillies par les capteurs et caméras sont envoyées à des ordinateurs qui, à l'aide

de nombreux algorithmes permettent de repérer les obstacles, calculer des distances/vitesses et même repérer la signalisation. Toutes ces informations sont aussi envoyées sur un écran de contrôle disponible sur le tableau de bord. Cet écran montre les images filmées par les caméras auxquelles ont été ajoutées les données analysées. Cela permet à l'équipe de M. Broggi de vérifier que la voiture analyse bien son environnement.

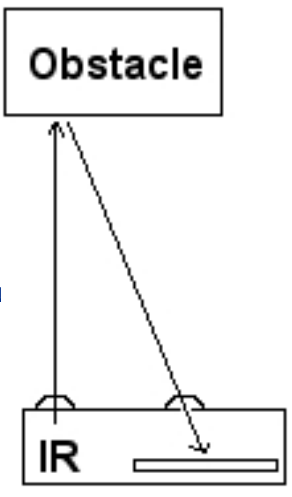


La voiture peut part exemple repérer le marquage au sol. Celui-ci, associé aux fonctions d'ACC et de Stop & Go, permet à la voiture de rester toute seule dans sa file sans aucune intervention humaine sur le volant et les pédales. Les lignes lui indiquent alors quelle trajectoire la voiture doit prendre pour rester sur sa file.

La voiture possède aussi d'autres fonctions. Dans un parking, elle peut détecter et calculer l'espace disponible entre deux véhicules. Les caméras intégrées aux rétroviseurs permettent aussi à la voiture de savoir si il lui est possible de changer de file pour dépasser. Ces deux fonctions n'agissent pour le moment que comme indicateur lumineux et/ou sonore mais n'agissent pas sur le comportement de la voiture. Cependant, le projet final de l'équipe Italienne est bien de faire une voiture entièrement autonome dans un futur proche. Actuellement, la BRAiVE peut conduire automatiquement sur autoroute en gardant sa file ou suivre une voiture pilote.

Une application courante en robotique est le « tracking » d'objets. Avec une ou plusieurs caméras et quelquefois des indicateurs colorés disposés sur l'objet suivi, il est possible assez facilement de repérer un objet, de le suivre, voir même le récupérer. Cependant il faut mieux coupler ces caméras à d'autres capteurs qui permettent de mesurer des distances avec plus d'exactitude qu'une caméra, surtout lorsque l'objet suivi est en mouvement. Il existe différents types de capteurs :

Il y a tout d'abord les télémètres à ultrasons. La distance est calculée en fonction du temps mesuré pour que le signal sonore parte du télémètre, soit réfléchi sur l'objet et revienne sur le capteur. Ils ont un intervalle de mesure d'une dizaine de centimètres à quelques mètres. L'information qu'il délivre est proportionnelle à la distance mesurée, ce qui permet une exploitation des résultats simplifiée. Il possède cependant un cône de mesure assez large, ce qui peut être vu comme un avantage ou un inconvénient selon l'utilisation que l'on veut en faire. Pour le tracking proprement dit, cela peut être un inconvénient car le capteur pourrait détecter autre chose que l'objet suivi.



Ensuite on trouve les télémètres infrarouge. L'avantage de ce système est qu'il est plus petit et moins cher que son homologue à ultrasons. Il fonctionne selon le principe de triangulation et d'une rangée de petites cellules photoconductrices. La triangulation est une application simple de la trigonométrie. On mesure les angles entre le point dont on cherche à calculer la distance et des points

de références dont la position est connue. Sur les capteurs infrarouge par exemple, on connait deux angles, celui à 90° et celui au sommet du triangle qui correspond à l'objet. La distance connue est celle entre l'émetteur infrarouge et l'une des cellules photoconductrices. Son intervalle de mesure est plus court puisqu'il va de quelques centimètres à quelques dizaines de centimètres. Son cône de mesure est très restreint (5° pour un modèle de chez Sharp) ce qui lui exclut de mesurer d'autres cibles que celles situées bien devant lui. Il est donc utilisé pour des travaux de précision et de proximité proche. Cependant, l'information qu'il délivre n'est pas proportionnelle à la distance mesurée, il faut donc prévoir un tableau de correspondance.

Il existe aussi les télémètres laser, ils fonctionnent sur le même principe que les télémètres infrarouge mais sont bien plus précis (on peut trouver des télémètres de +/- 1 mm de précision). Ils possèdent un cône de mesure encore plus petit, et un intervalle de mesure allant de environ 50 cm jusqu'à 60 m. C'est l'outil idéal pour mesurer la distance d'une cible bien précise, mais il est bien plus cher.

Enfin, on trouve les systèmes radar. Ces dispositifs envoient des ondes radio. Grâce au temps que met l'onde pour revenir et son changement de fréquence (effet Doppler), on arrive à calculer la distance qui nous sépare de l'objet et sa vitesse. Les radars ont des cônes de mesures qui peuvent varier selon l'application. Les radars que l'on trouve dans les aéroports permettent d'avoir des mesures à 360° en faisant tourner l'émetteur sur lui même. Dans le domaine automobile, le cône est plus restreint car on souhaite mesurer la vitesse des voitures circulant dans un sens seulement. Ce cône va donc être plus ou moins large selon que l'on souhaite mesurer une ou plusieurs voies. Dans tous les cas, le cône est plus large que les télémètres laser et infrarouge.

4. TRAVAIL REALISE ET RESULTATS

4.1. Les différents composants constituant le robot

4.1.1. Les moteurs

4.1.1.1. Les caractéristiques

Pour faire avancer notre robot nous utilisons des moto-réducteurs de marque Tamiya. Il s'agit en fait d'un boitier de modélisme dans lequel sont montés des moteurs électriques suivis de réducteurs : Le **Twin-Motor Gearbox**. Ces moto-réducteurs sont vendus en kit à monter soit même. Dans la boite on retrouve deux moteurs **FA-130 Motor** de **Mabuchi**, un châssis en plusieurs parties et enfin des engrenages pour effectuer la réduction.

Figure 1 : Le Twin-Motor Gearbox

Voici les caractéristiques des moteurs prises de la datasheet (Ajoutée en annexes) :

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
FA-130RA-18100	1.5~3.0	3V CONSTANT	12300	0.15	9710	0.56	0.74	7.6	0.76	3.53	36	2.10

Nous pouvons aussi trouver sur internet divers test d'Adam Borrell [1] qui a testé plusieurs moteurs de ce type à des tensions diverses, ces tests sont en fait effectués sur des GM9 Gearbox, un moto-réducteur de Solarbotics mais qui utilise bien le même moteur (seul la réduction diffère). Voici les résultats obtenus :

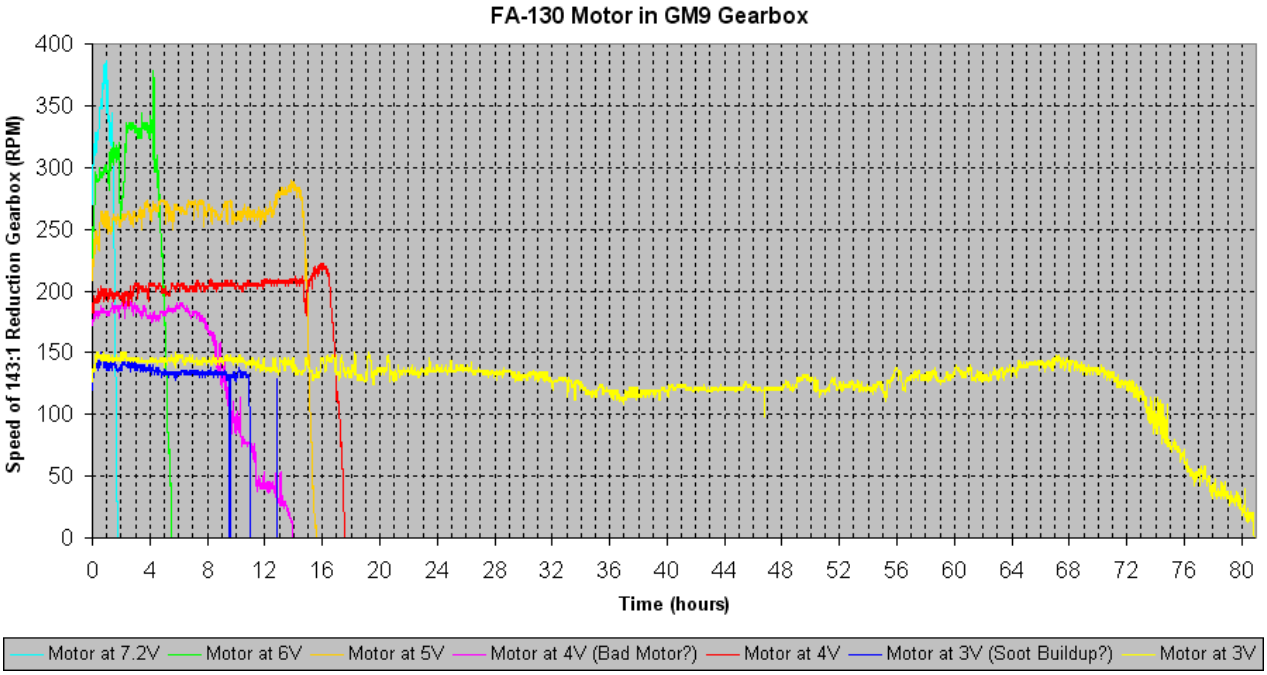


Figure 2 : Durée de vie et vitesse en fonction de la tension d'un GM9 Gearbox

Comme on peut le voir, bien qu'il soit conseillé de faire marcher le moteur entre 1,5 et 3V, rien n'empêche d'appliquer une tension plus forte sur le moteur, le seul inconvénient étant une chute de sa durée de vie. Le montage que nous avons effectué nous donne une tension de 3,2V aux bornes des moteurs. Au vue de la durée de vie du moteur à 3V plus que suffisante et étant donné que notre tension est à peine plus élevée que cela, nous avons décidé de garder cette tension à 3,2V. De plus, appliquer une tension un peu supérieure nous permet d'accroître la vitesse de nos moteurs, ce qui n'est pas plus mal si nous voulons suivre des objets un peu plus rapides.

Il faut savoir que le Twin-Motor Gearbox permet de disposer les engrenages de deux manières différentes afin d'avoir des ratios de réductions différents. Le ratio de 58:1 permet sous 3V d'avoir une vitesse théorique de 230 tr/min mais donne un couple théorique de seulement 269 g.cm. Le ratio de 203:1 permet par contre un couple plus élevé de 941 g.cm, mais la vitesse délivrée est de seulement 65 tr/min (voir sources). Notre robot étant destiné à suivre tout objet en mouvement disposé devant lui, nous avons opté pour le ratio permettant une vitesse plus élevée, ainsi notre robot sera apte à suivre des objets plus rapides. Notre robot se destinant à ce moment là à être plutôt léger, nous nous sommes dit que le couple ne nous poserait pas problème. Ce fût une bonne hypothèse étant donné que le robot roule sans soucis aujourd'hui.

4.1.1.2. Câblage des moteurs

Comme l'illustre très bien le graphique précédent, plus la tension aux bornes d'un moteur électrique est élevé, plus ce dernier va tourner vite. L'autre caractéristique principale des moteurs électriques est que le sens de rotation d'un moteur est fonction du sens de passage du courant dans ce dernier. En effet, si l'on inverse le sens de passage du courant dans un moteur, on change le sens de rotation de celui-ci. Nous avions donc besoin d'un circuit électronique répondant à ces fonctions. Voici donc le schéma de câblage d'un des deux moteurs :

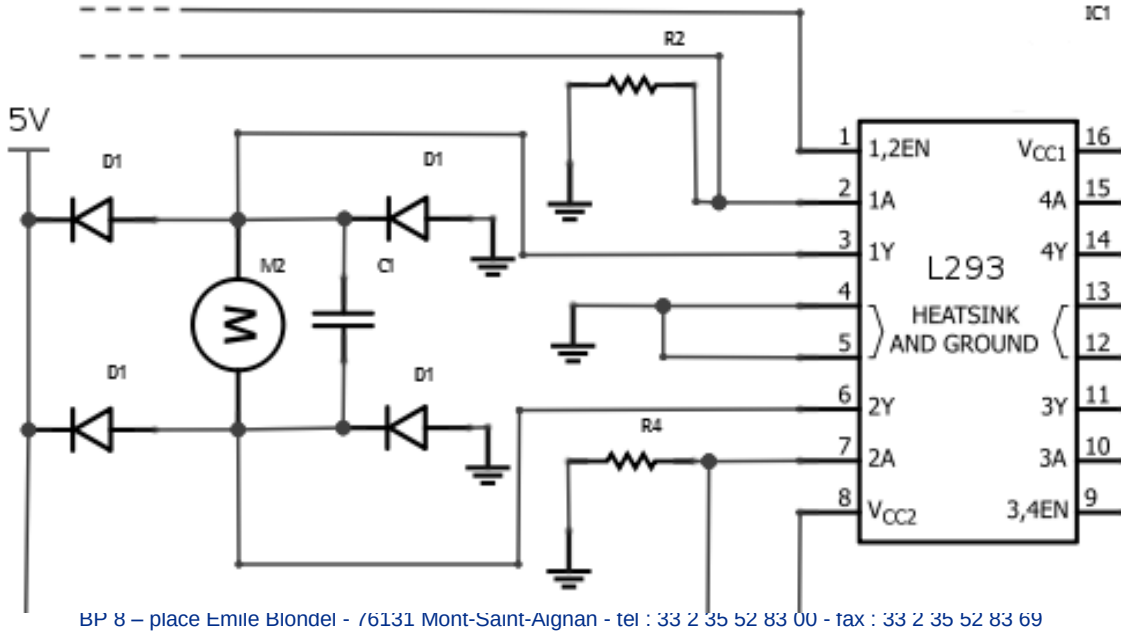


Figure 3 : Schéma de câblage partiel (Schéma complet en annexe)

Sur ce schéma nous pouvons tout d'abord voir les habituelles 4 diodes de roue libre. Leur rôle est de protéger le reste du circuit lorsque l'on arrête de fournir de l'énergie au moteur. En effet, lorsque l'on ne fournit plus de tension aux bornes du moteur, le moteur continue de tourner avant de s'arrêter. Le moteur crée ainsi lui même un courant qui pourrait remonter dans le circuit et pourrait alors l'endommager. Les diodes de roues libres sont là pour obliger le courant à passer à travers elles et le dissiper ainsi par effet Joule. Si il y a quatre diodes de roue libre et non deux seulement, c'est pour assurer la protection pour les deux sens de rotation du moteur.

On repère aussi des condensateurs aux bornes du moteur que nous avons ajouté comme antiparasites. En effet, afin d'éviter de perturber la réception du signal de l'émetteur, il est nécessaire d'éliminer les parasites que le moteur crée, il doit donc être antiparasité.

Le moteur est relié à ses bornes aux broches 3 et 6 du L293 qui est un composant effectuant la fonction de pont en H. C'est ce composant qui va permettre de moduler la vitesse de rotation du moteur ainsi que choisir son sens de rotation. En effet, grâce à plusieurs entrées, il est possible de commander les sorties du L293. Ainsi, l'entrée correspondant à la broche 2 permet de commander la sortie correspondant à la broche 3 et l'entrée de la broche 7 permet de commander la sortie de la broche 6. La broche numéro 1 possède la fonction d'enable qui permet d'activer par paire les modules d'entrées/sorties du L293 associés. Ainsi, cette broche permet d'activer les blocs d'entrées/sorties numéro 1 et 2 qui correspondent respectivement aux broches 2/3 et 7/6 (voir schéma précédent).

Techniquement, les entrées sont commandables en tension. Voici leur manière de fonctionner :

- La première étape afin de pouvoir se servir des entrées/sorties est d'envoyer une tension de 5V sur l'enable. Lorsque celui reçoit du 5V il active ses entrées/sorties associées. De la même manière, si on lui envoie du 0V celui-ci bloque ses entrées/sorties.
- Lorsqu'une entrée reçoit du 5V, il est envoyé à la sortie associée la tension disponible au VCC2. Si une entrée reçoit par contre du 0V, la tension envoyée sera aussi de 0 V.

Dans notre cas, le VCC2 est relié au VCC1 lui même relié au 5V, nos sorties pourront donc délivrer du 5V ou du 0V selon ce qui est demandé.

Si nous avons relié le VCC2 au 5V et non au 3V (la tension conseillée pour notre moteur), c'est parce qu'il est indiqué dans la documentation du L293 que le VCC2 doit être supérieur ou égale au VCC1 (voir annexes). Dans le cas contraire, le composant ne pourra pas fonctionner correctement. Cependant, le L293 possède plusieurs transistors internes qui réalisent les différentes fonctions du chip. Les transistors provoquent plusieurs chutes de tensions qui rabaissent les 5V théorique qu'on l'on devrait avoir aux bornes du moteur à 3,2V, ce qui est raisonnable pour notre moteur.

Grâce aux nombreuses entrées/sorties du L293, il est assez simple de commander le moteur. Voilà comment nous procédons : Pour faire tourner le moteur dans un sens il suffit qu'une de ses bornes soit reliée au 5V et l'autre borne au 0V. Nous commandons donc les

entrées en fonction. De cette façon, en envoyant du 0V sur une des deux entrées, la sortie associée délivrera du 0V qui arrivera à l'une des bornes du moteur. De la même manière, en envoyant du 5V sur l'autre entrée, sa sortie associée délivrera du 5V qui arrivera à l'autre borne du moteur. Le moteur recevant une différence de potentiel suffisante, il se met à tourner. Pour changer le sens de rotation du moteur il suffit de changer les états des entrées. Il faut donc envoyer du 0V à l'entrée qui en recevait 5 et envoyer du 5V à celle qui en recevait 0. De cette manière on inverse la polarité aux sorties du L293 et le courant qui traverse le moteur change de sens.

En regardant attentivement le schéma, vous repérerez des résistances qui vont d'un côté à la masse et de l'autre aux entrées du L293. Ces résistances sont des résistances appelées de Pulldown. Leur rôle est de forcer les états des entrées à 0V (état bas) lorsqu'elles ne reçoivent pas de tension égale à 5V. Sans ces résistances, les états aux broches du L293 ne correspondaient pas à la programmation que nous avions faite. Cependant, lorsque l'on enlevait le L293 du circuit, les états en sorties du microcontrôleur correspondaient bien aux états définis lors de la programmation. Nous avons remarqué qu'il n'y avait aucun souci lorsque l'on envoyait du 5V sur les entrées du L293 mais que le problème se posait lorsqu'on envoyait du 0V. Forcer ces résistances à un état de 0V lorsque l'on n'envoie pas de 5V paraissait une bonne solution d'où les résistances de Pulldown.

Pour ce qui est de moduler la vitesse, cette fonction est réalisée grâce aux enables. En effet, c'est sur ces enables que nous effectuons le PWM.

4.1.1.3. Le PWM

Les lettres PWM signifient Pulse With Modulation. Son équivalent Français est le MLI : Modulation de Largeur d'Impulsions. Le PWM est un principe très utilisé en électronique, notamment dans la commande de moteur à courant continu.

Son principe général est assez simple :

« En appliquant une succession d'états discrets pendant des durées bien choisies, on peut obtenir en moyenne sur une certaine durée n'importe quelle valeur intermédiaire » (cf. Wikipedia).

Dans notre cas précis, il s'agit d'envoyer successivement des états 5 et 0V pendant des durées déterminées sur une période T. Sur cette période T on obtient un coefficient α correspondant à la durée de l'état haut sur celle de l'état bas qu'on appelle rapport cyclique. Ainsi, en envoyant pendant la moitié d'une période T un état haut (5V) et pendant l'autre moitié un état bas (0V), on obtient un rapport cyclique α=0,5. Ce rapport cyclique permet d'exprimer la valeur moyenne plus facilement. Concrètement, il s'agit de multiplier la valeur de l'état haut par le rapport cyclique. Ainsi dans notre cas, avec un α=0,5 on obtient comme valeur moyenne 2,5V. Il est donc très simple d'obtenir n'importe qu'elle valeur comprise entre l'état haut et l'état bas. Un rapport α=1 signifiant un état haut permanent et un rapport α=0 un état bas permanent.

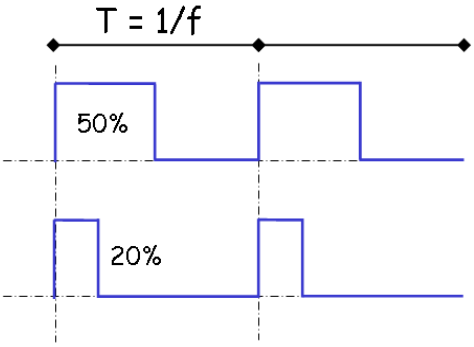


Figure 4 : Exemples de cycles

Comme je l'indiquais précédemment, nous effectuons le PWM sur les enables du L293. Pour rappel, les enables du L293 permettent d'activer les blocs d'entrées/sorties de ce dernier. Ainsi, en envoyant un cycle sur les enables nous repercutons ce cycle aux bornes du moteur. Nous pouvons donc faire moduler la tension aux bornes de notre moteur entre 0 et 3,2 V et moduler de cette manière sa vitesse. Du moins en théorie, car pour le cas d'une tension trop faible le moteur n'arrivera pas à démarrer.

4.1.2. Le Microcontrôleur

Pour commander notre robot nous utilisons un microcontrôleur de marque Arduino : l'Arduino Duemilanove. Ce microcontrôleur est basé sur l'ATmega328, un microcontrôleur de chez Atmel.

4.1.2.1. Qu'est ce qu'un microcontrôleur ?

Un microcontrôleur est un circuit intégré réunissant les éléments essentiels d'un ordinateur : Processeur (CPU), mémoire morte (ROM), mémoire vive (RAM) et parfois une mémoire de type flash. Ils sont de ce fait programmables et peuvent effectuer diverses actions. On trouve dans ces puces des interfaces parallèles pour la connexion des entrées/sorties ainsi que des interfaces séries pour pouvoir dialoguer avec d'autres éléments. Beaucoup de microcontrôleurs sont équipés d'un convertisseur analogique numérique (C.A.N) pour le traitement de signaux analogiques. On trouve aussi sur ces puces ce qu'on appelle les timers. Ils génèrent ou mesurent des durées afin de synchroniser les opérations que le microcontrôleur doit effectuer. C'est grâce à eux que le microcontrôleur peut générer des signaux périodiques par exemple.

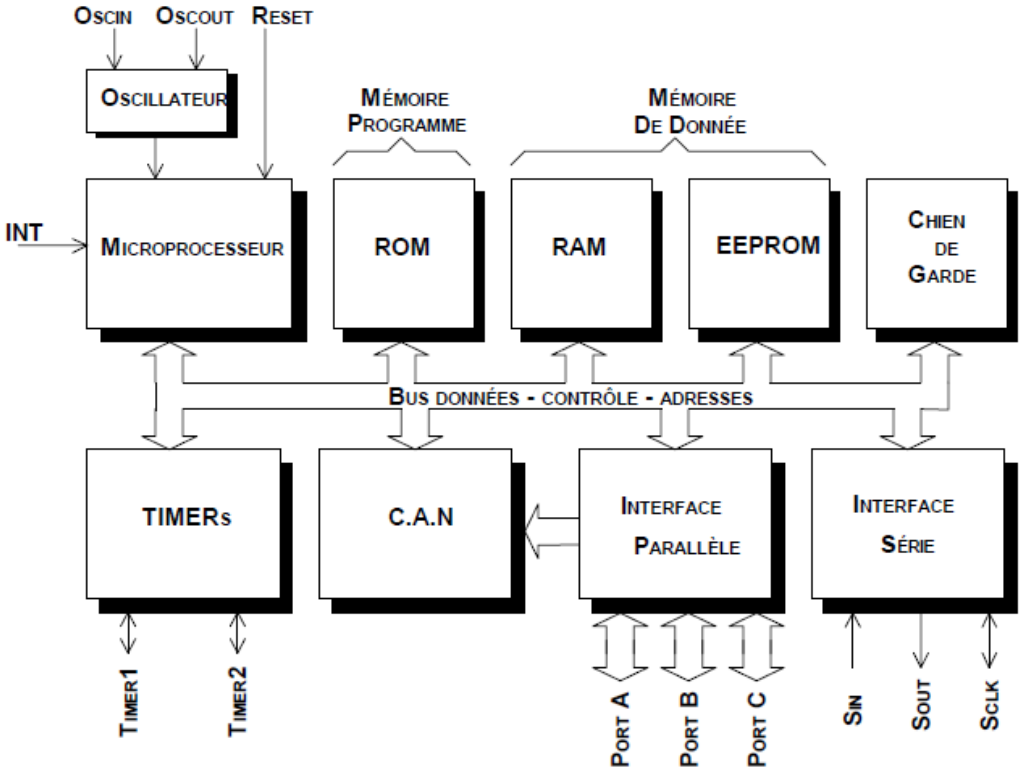


Figure 5 : Une architecture de microcontrôleur typique (Von Neumann)

Les microcontrôleurs sont souvent utilisés dans les systèmes embarqués car ils présentent plusieurs avantages comparés à des systèmes informatiques traditionnels. Les avantages des microcontrôleurs sont : un encombrement réduit, une faible consommation, un coût réduit et enfin un circuit imprimé peu complexe. L'inconvénient majeur de ces puces est qu'il faut un matériel adapté pour les programmer. Cependant, une fois le programme correctement conçu et chargé dans la mémoire du microcontrôleur, il n'a dans la plupart des

cas, plus besoin d'être changé puisque le système dans lequel est implémenté le microcontrôleur est amené à être nomade.

4.1.2.2. L'ATmega et l'Arduino

L'ATmega328 :

L'ATmega328 possède une mémoire flash de 32KB afin d'y stocker le programme, une mémoire SRAM de 2KB pour l'échange de données et 1024 Bytes d'EEPROM pour stocker des données que l'on souhaiterait garder après la mise hors tension. Il est équipé de 23 broches pouvant servir d'entrée ou de sortie, voir les deux. Voilà comment celles-ci sont câblées sur l'Arduino :

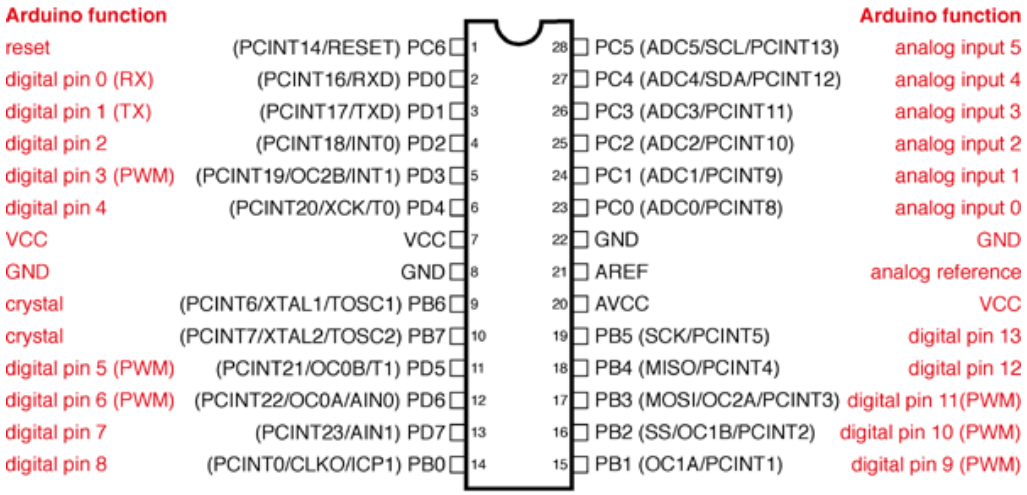


Figure 6 : Correspondance ATmega328/Arduino

L'Arduino ne se sert pas de toutes les fonctions des entrées/sorties de l'ATmega328, mais voici ce qu'il propose : Il est équipé de 14 entrées/sorties numérique dont 6 pouvant servir au PWM ainsi que de 6 entrées analogiques. Les entrées analogiques possèdent une résolution de 10 bits, c'est-à-dire qu'elles sont capable d'interpréter 1024 valeurs différentes. Les sorties PWM quant-a-elles possèdent une résolution de 8 bits, elles peuvent donc délivrer 256 valeurs. Plus concrètement, il est possible de faire 256 rapports cycliques différents, la valeur 255 correspondant à un état haut permanent (α=1) et 0 à un état bas permanent (α=0). L'« analogue reference » (ou AREF) est une entrée qui permet d'imposer une tension de référence différente de 5V pour l'utilisation des entrées analogiques. Par défaut, les entrées analogiques peuvent traduire 1024 valeurs différentes comprises entre 0 et 5V. Grâce à l'AREF, on peut demander au microcontrôleur de traduire 1024 valeurs entre 0V et une tension de son choix. L'intensité disponible par broche d'E/S est de maximum 40 mA.

Ce qu'ajoute l'Arduino :

L'Arduino est en fait monté pour pouvoir utiliser l'ATmega328 le plus facilement possible. Ainsi un crystal est branché aux bornes des broches 9 et 10 de l'ATmega. Ce crystal transmet un signal d'horloge au microcontrôleur pour cadencer son fonctionnement, sa vitesse est de 16MHz. Sur les broches 2 et 3 sont reliés divers composants finissant sur un

port USB. Ces broches servent à charger le programme dans l'ATmega, le port USB sert alors d'interface pour transférer le programme du PC au microcontrôleur. La broche numéro 1 servant de reset est quant-à-elle branchée à un bouton, lorsque celui-ci est actionné le programme implanté dans l'ATmega redémarre. Le microcontrôleur fonctionne sous des tensions allant de 1,8V à 5,5V. Sur l'Arduino Duemilanove, le microcontrôleur est alimenté en 5V. Pour fournir cette tension, la carte italienne est équipée d'un régulateur qui permet de baisser une tension, soit venant du connecteur d'alimentation (noir sur la photo) soit venant par « l'entrée » Vin de l'Arduino. Ce régulateur accepte des tensions allant de 6 à 20V, mais il est recommandé de lui envoyer des tensions allant de 7 à 12V afin de garantir le bon fonctionnement et la longévité de l'appareil. Un second régulateur permet de créer une tension de 3,3V. La carte Arduino peut aussi être alimentée par la connexion USB. Celui-ci fournit du 5V jusqu'à 500mA .La masse, le 5V, le 3,3V et le Vin sont récupérables par le dessus de la carte par des connecteurs où est disposés la majeur partie des entrées/sorties de l'ATmega328.

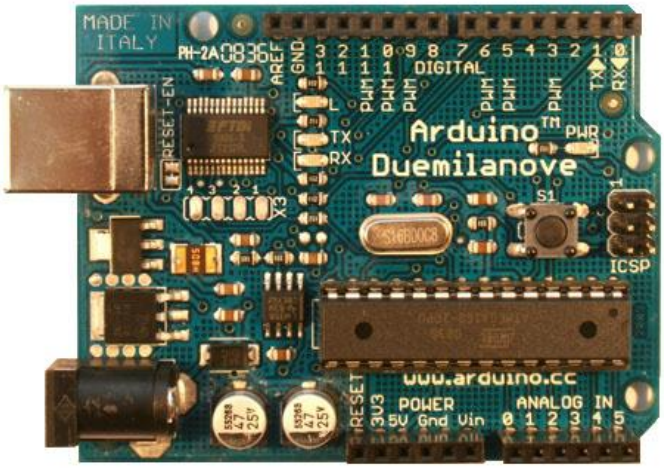


Figure 7 : L'Arduino Duemilanove

Sur l'Arduino Duemilanove, la mémoire flash de l'ATmega328 est déjà chargée avec le « bootloader ». Le bootloader est un petit programme de démarrage qui permet le transfert de nouveaux programmes dans le microcontrôleur sans pour autant avoir besoin d'utiliser un matériel de programmation externe. Il est cependant toujours possible de se passer du bootloader en programmant le microcontrôleur via le connecteur ICSP (situé à droite sur la photo) , mais utiliser le bootloader via le port USB est bien plus pratique. Le bootloader occupe 2KB de la mémoire flash de l'ATmega328, mais 30KB dédiés à la programmation restent largement suffisants pour l'application que nous voulions faire de notre robot.

Pour programmer la carte, Arduino met à disposition un logiciel de programmation dédié à leurs modèles. Ce programme permet de compiler un langage de programmation dérivé du C/C++. En effet, Arduino à ajouté des instructions spécifiques à leurs cartes (comme digitalWrite(), analogRead(), …). Les autres instructions sont du langage C classique. Le logiciel permet également de charger le programme sur la carte Arduino.

Les dimensions de l'Arduino Duemilanove sont de 6,86 cm en longueur sur 5,33 cm de largeur sans prendre en compte le port USB et le connecteur jack d'alimentation qui dépassent légèrement de la carte. La carte est perforée à trois endroits afin de pouvoir la fixer à l'aide de vis.

4.1.3. Les capteurs

Pour que notre robot soit capable de suivre l'objet situé devant lui, nous utilisons des capteurs de distance. Plus précisément, nous utilisons des télémètres infrarouges **Sharp GP2D12**. Concernant le fonctionnement des télémètres infrarouges, nous ne reviendrons pas dessus car cela a déjà été expliqué dans l'état de l'art.

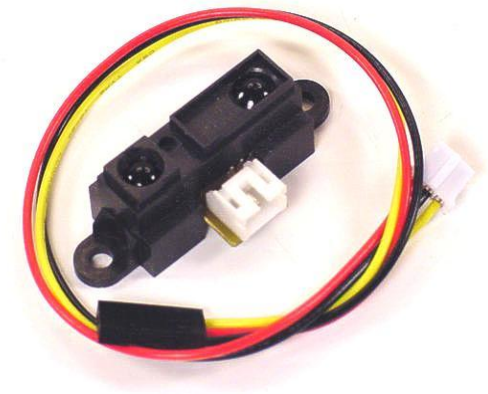


Figure 8 : Un capteur Sharp GP2D12

Les capteurs Sharp GP2D12 sont des télémètres infrarouge de type analogique, c'est-à-dire qu'ils délivrent une tension qui est fonction de la distance mesurée. Ils s'opposent aux capteurs de type binaire qui délivrent un état haut ou un état bas en fonction de la présence ou non d'un objet dans leur intervalle de mesure.

Les capteurs GP2D12 permettent de mesurer convenablement des distances allant de 10 à 80 cm. En réalité, ils sont capables de mesurer des distances allant un peu au-delà, mais ils sont alors beaucoup moins précis et la relation volt/distance est beaucoup moins évidente. Ces capteurs fonctionnent sous des tensions pouvant aller de 4,5 à 5,5 V (voir la Datasheet). Pour connecter ces télémètres, il faut utiliser des connecteurs JST 3-pin. Ces connecteurs permettent ensuite de câbler le capteur à la masse, à la tension d'alimentation (majoritairement de 5V) et de récupérer la tension de sortie pour la brancher par exemple à l'entrée analogique d'un microcontrôleur. La consommation de ces capteurs est de 25 mA et sa température en fonctionnement doit être comprise entre -10 et 60°C. Ses dimensions sont de 30 mm en longueur, 15 en largeur et 10 en épaisseur.

La tension transmise par les capteurs n'est cependant pas linéaire, c'est-à-dire qu'elle n'est pas proportionnelle à la distance. Voici donc la courbe représentant la relation entre

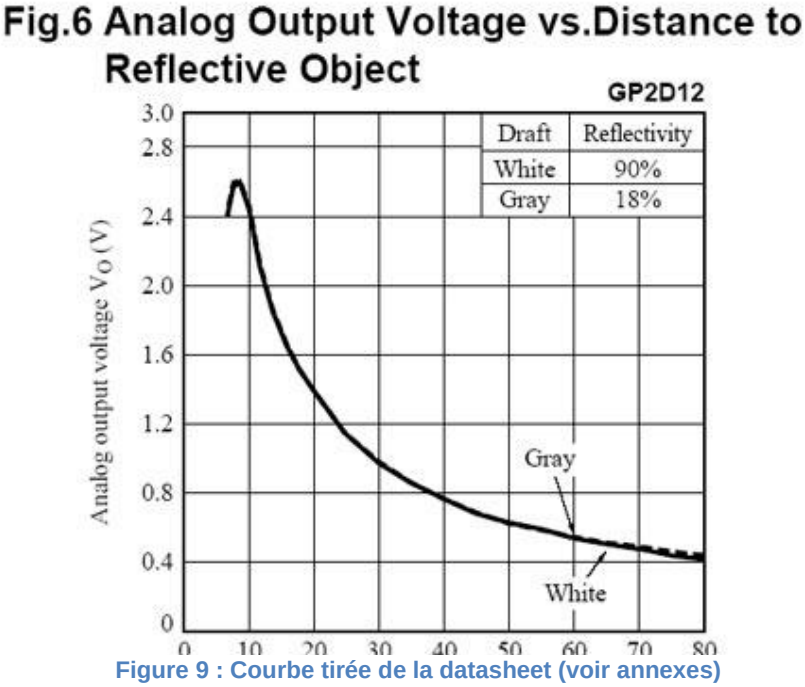


Figure 9 : Courbe tirée de la datasheet (voir annexes)

les deux :

Pour notre robot nous utilisons trois de ces capteurs, un pour mesurer la distance en face du robot, un autre pour mesurer la distance diagonalement à droite et un dernier pour mesurer la distance diagonalement à gauche. Ainsi notre robot est capable de savoir lorsqu'un objet se met à tourner d'un côté ou de l'autre. Les tensions de sortie des 3 capteurs sont reliées à 3 des entrées analogiques de la carte Arduino. La carte Arduino se charge ensuite de traduire ça en distance grâce au programme que nous lui avons implanté. Le programme effectue ensuite les actions nécessaires pour remettre le robot dans le droit chemin.

4.2. Conception mécanique

Comme dit précédemment, l'un des critères du cahier des charges était d'utiliser au maximum des matériaux recyclés. Pour cela nous avions utilisé des CD pour construire plusieurs étages sur lesquels étaient installées différentes parties du robot. Les étages étaient disposés ainsi : Sur le premier était fixé le moteur à l'aide de deux vis. Sur le second on trouvait la batterie et sur le troisième et dernier étage était disposé l'Arduino duemilanove. Pour maintenir les CDs entre eux, nous utilisions des entretoises de 20 mm de hauteur, la distance entre chaque CD était de 40 mm.

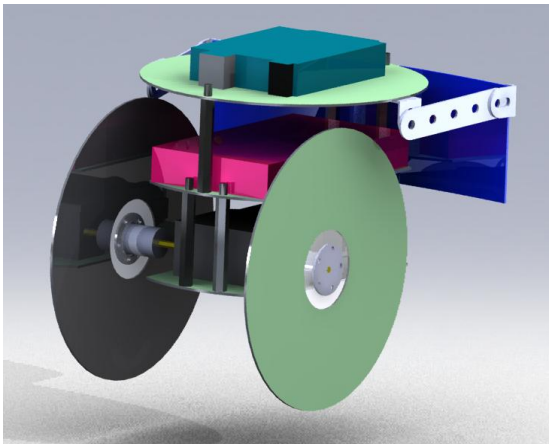


Figure 10 : On distingue bien 3 étages

Tout ces composants ont été modélisés sous SolidWorks dans les bonnes dimensions afin de nous aider à mieux concevoir l'architecture du robot. C'est ainsi que nous avons décidé qu'il fallait la hauteur de deux entretoises entre chaque CD si nous voulions que nos roues à base de CDs puissent passer sous l'étage supérieur. En effet, l'étage supérieur n'a pas été coupé comme les deux autres car il devait pouvoir accueillir le support des capteurs en mécanos. L'emplacement des entretoises a été choisi de telle sorte que la batterie soit maintenue simplement par celles-ci sans aide extérieure. Les essieux des roues n'ont pas été placés au milieu du CD inférieur volontairement afin de garantir une meilleur stabilité. En effet, en plaçant les essieux à l'une des extrémités du CD on s'assurait ainsi que le robot pencherait vers l'autre côté du CD où il serait maintenu grâce à un troisième point d'appui. Pour maintenir ces roues sur les essieux nous avions prévu d'acheter des moyeux de marque Lynxmotion chez Electronic Diffusion, un magasin situé dans le centre ville de Rouen. Le magasin a

malheureusement fermé et nous avons été dans l'incapacité d'acheter les moyeux. Comme solution de secours nous avions donc fait une mise en plan de roues sur mesures qui auraient pu s'adapter directement sur les essieux. L'INSA est en effet capable d'usiner des pièces si l'on donne aux personnes qui gèrent cela une bonne mise en plan. Cette mise en plan est disponible en annexes.

Si nous avions choisi d'utiliser des CDs, c'est parce que l'on voulait que nos roues soient d'une taille assez conséquente afin de s'assurer que le robot ait une vitesse assez élevée pour suivre un objet. Nous n'avons finalement pas usiné les roues parce que nous avons trouvé des pièces de caoutchouc pouvant s'adapter sur des poulies de mécanos. Ces poulies de mécanos s'adaptent parfaitement sur les essieux hexagonaux des moteurs et ont un diamètre de 84 mm; elles étaient donc parfaites pour notre robot.

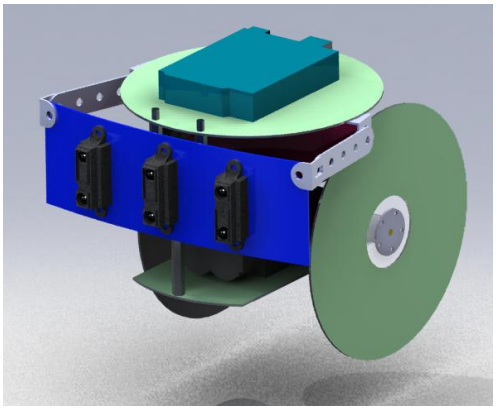


Figure 11 : Le support des capteurs

Le support accueillant les capteurs a été construit en pièces de mécanos. Ces mécanos sont de très vieux mécanos que Rémi a trouvés chez lui et qui ont plus de 30 ans, ils rentraient donc très bien dans les critères du cahier des charges. L'avantage de la pièce métallique bleu est quelle est pliable et qu'elle dispose de nombreux trous (non représentés sur la modélisation 3D). Elle nous donnait donc l'avantage de pouvoir tester diverses positions de capteurs en pouvant choisir un angle entre chaque capteur plus ou moins élevé.

Finalement les CDs n'ont pas survécu aux différents voyages que le robot a subis. La boite en carton utilisée pour le transporter était trop petite et les CDs se sont fissurés là où étaient fixées les entretoises. Tous les CDs mis à part le CD supérieur ont donc été remplacés par des mécanos plus solides pour donner le résultat que vous avez pu découvrir sur la couverture.

Sur cette photo de l'arrière vous pouvez voir la partie métallique courbée qui sert de troisième point d'appui. On peut aussi distinguer des parties métalliques par lesquelles passent les axes des moteurs. Ces parties métalliques ont été ajoutées pour maintenir les axes en place.

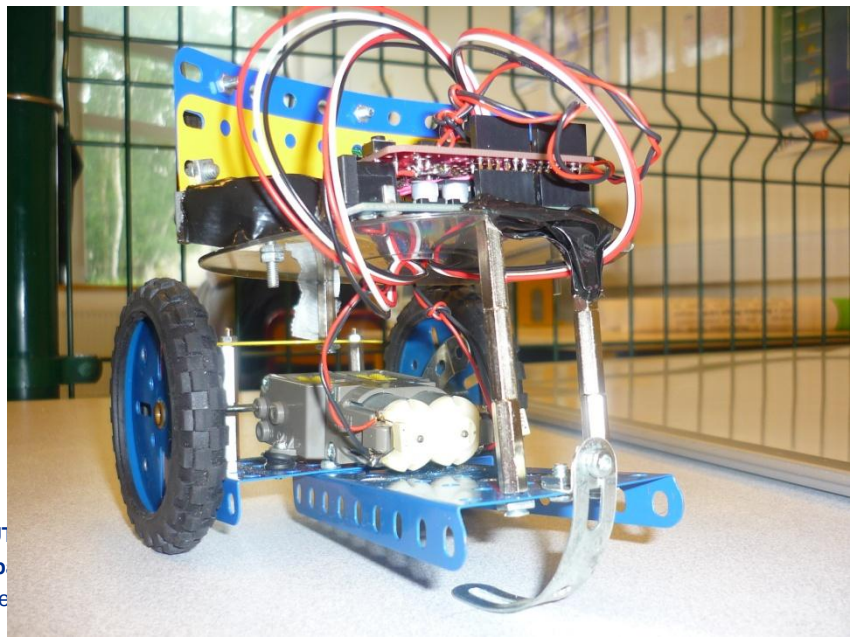
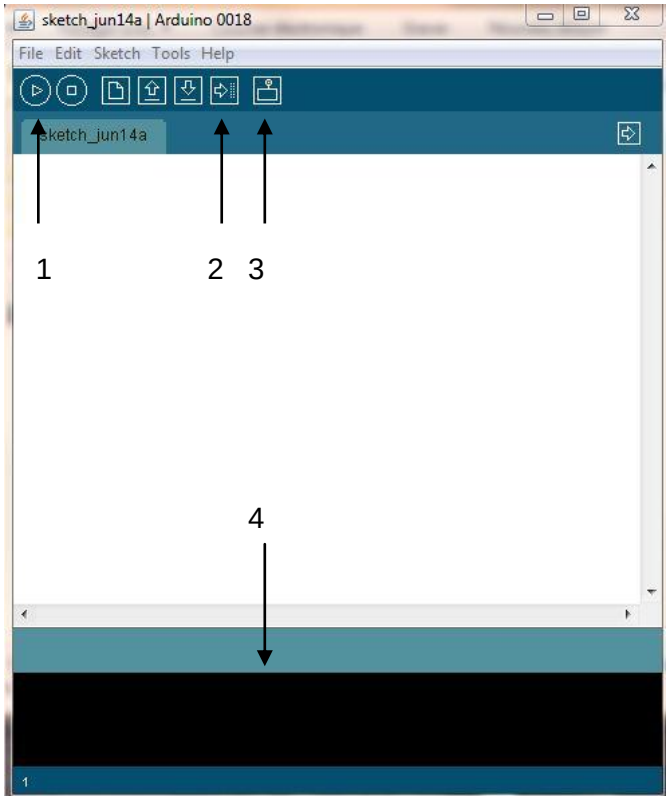


Figure 12 : Photo de l'arrière du robot

4.3. Programmation et étude logicielle

4.3.1. Etude logicielle

Pour programme, nous avons utilisé un logiciel gratuit fourni par Arduino. La programmation des cartes arduino se fait en C/C++, un langage qu'on a eu l'occasion de découvrir au premier semestre en STPI. On n'était donc pas complètement perdu. Ce qui était différent, c'est que c'était par exemple la première fois qu'on associait programmation et électronique.



L'environnement arduino est très simple à appréhender à la première utilisation. Il se présente comme présenté à côté. Les fonctions les plus importantes sont : 1) pour compiler, 2) pour envoyer le programme à la carte arduino, 3) pour afficher un écran où on peut recevoir les données de la carte en utilisant le port série adéquat et enfin en 4) le terminal où le logiciel nous informe des erreurs éventuelles de compilation.

Notre problématique était pour le moins très clair et bien définie. Il fallait que notre robot puisse suivre un objet en mouvement en gardant toujours une distance assez proche entre eux. Les capteurs GP2D12 que nous utilisons détectent un objet entre 10 et 80 cm. Pour plus de sécurité, nous avons fixé une distance moyenne entre le robot et l'objet de 30 cm.

4.3.2. Programme commenté

Regardons comment se construit un programme type pour une carte arduino.

D'abord, nous avons la déclaration de toutes les variables dont nous avons besoin pour notre programme. Dans notre cas, voici les différentes variables :

```
// Moteur
int enablegauche=9;
int broche2=2;
```

```
int broche7=3;
int enabledroit=10;
int broche10=6;
int broche15=5;

// Capteur distance
int irReaderF = 1;    //entrée analogique pour le capteur frontal
int irReaderL =2;     //entrée analogique pour le capteur gauche
int irReaderR =0;     //entrée analogique pour le capteur droit
int irValF= 0;        //stocke la valeur(en volt) retournée par le capteur frontal
int irValL=0;         //stocke la valeur(en volt) retournée par le capteur gauche
int irValR=0;         //stocke la valeur(en volt) retournée par le capteur droit
int irValF1;          //stocke la distance en cm retournée par le capteur frontal
int irValL1;          //stocke la distance en cm retournée par le capteur gauche
int irValR1;          //stocke la distance en cm retournée par le capteur droit
int dist_depart;      //distance entre le robot et l'objet à suivre au depart
int dist_droite;      //distance retournée par le capteur droit
int dist_gauche;      //distance retournée par le capteur gauche
int table[3];         //tableau contenant les distances
int dist_fixe=30;     //distance que doit maintenir le robot avec l'objet
```

Après l'étape de la déclaration, il faut impérativement ouvrir un port série de communication et aussi fixer la vitesse de transmission d'arduino, autrement une erreur est signalée à la compilation. Cela se fait avec une procédure setup.

```
void setup( )
{
 Serial.begin(9600);  // ouvre le port série, fixe le taux de données à 9600 bps
}
```

Ensuite vient une procédure loop dans laquelle on spécifie à la carte arduino ce qu'on attend qu'elle fasse, notamment en faisant appel aux procédures qui permettent le mouvement du robot et aux fonctions qui détectent l'objet. Nous y reviendrons plus tard, car c'est la partie la plus difficile, quand tous les blocs interagissent entre eux. Dans cette procédure loop, les actions sont exécutées indéfiniment.

Intéressons nous maintenant aux procédures qui servent à déplacer le robot. Mettre en mouvement le robot inclut 7 procédures : avancer, reculer, tourner à droite, tourner à

gauche, motor_stop, ralentir et enfin accélérer. Pour toutes ces procédures, il suffit de mettre à l'état haut ou à l'état bas les entrées adéquates du pont en H, le L293, dont le fonctionnement est décrit dans une autre partie de ce rapport.

Voyons maintenant comment ces procédures sont traduites en langage C/C++ pour la carte arduino :

```
void reculer()  // reculer
{
  digitalWrite(enablegauche,HIGH);
  digitalWrite(enabledroit,HIGH);
  digitalWrite(broche2,HIGH);
  digitalWrite(broche7,LOW);
  digitalWrite(broche10,HIGH);
  digitalWrite(broche15,LOW);
}
```

```
void avancer()  // aller tout droit
{
  analogWrite(enablegauche,200);
  analogWrite(enabledroit,200);
  digitalWrite(broche2,LOW);
  digitalWrite(broche7,HIGH);
  digitalWrite(broche10,LOW);
  digitalWrite(broche15,HIGH);
}
```

```
void turnRight()  // tourner à droite
{
  digitalWrite(enablegauche,HIGH);
  digitalWrite(broche2,HIGH);
  digitalWrite(broche7,LOW);
  digitalWrite(enabledroit,LOW);
  digitalWrite(broche15,HIGH);
  digitalWrite(broche10,HIGH);
}
```

```
void turnLeft()  // tourner à gauche
{
  digitalWrite(enablegauche,LOW);
  digitalWrite(enabledroit,HIGH);
  digitalWrite(broche10,HIGH);
  digitalWrite(broche15,LOW);
  digitalWrite(broche2,HIGH);
  digitalWrite(broche7,HIGH);
}
```

```
void motor_stop()  // arreter le moteur
{
 digitalWrite(enablegauche,LOW);
 digitalWrite(enabledroit,LOW);
 digitalWrite(broche2,LOW);
 digitalWrite(broche7,LOW);
 digitalWrite(broche10,LOW);
 digitalWrite(broche15,LOW);
 }
```

```
 void accelerer( ) //accelerer
{
 analogWrite(enablegauche,255);
 analogWrite(enabledroit,255);
 digitalWrite(broche2,LOW);
 digitalWrite(broche7,HIGH);
 digitalWrite(broche10,LOW);
 digitalWrite(broche15,HIGH);
 }
```

```
void ralentir()  // ralentir
{
 analogWrite(enablegauche,127);
 analogWrite(enabledroit,127);
 digitalWrite(broche2,LOW);
 digitalWrite(broche7,HIGH);
 digitalWrite(broche10,LOW);
 digitalWrite(broche15,HIGH);
 }
```

Les procédures avancer et reculer nous ont seulement servi à vérifier la bonne syntaxe de notre code, nous ne les utilisons pas dans la procédure loop. Ils pourront éventuellement servir dans une optimisation future du programme.

Comme dit plus haut, dans ces différentes procédures, on modifie seulement l'état de certaines entrées du L293. Prenons par exemple la procédure turnRight( ). Pour tourner à droite, c'est la roue gauche qui tourne tandis que la roue droite reste statique. Pour faire tourner la roue gauche il faut une tension aux bornes de son moteur, nous mettons donc un état haut (HIGH) à l'une des sorties de l'Arduino et un état bas (LOW) sur une autre. Ces deux sorties vont ensuite sur deux entrées du L293 qui ensuite redistribue une tension aux bornes du moteur. Il faut bien sûr s'assurer que la roue tourne dans le bon sens, dans le cas contraire il faut inverser les deux états. Pour arrêter l'autre roue, nous avons simplement envoyé un état bas à la sortie correspondant à l'enable droit. Les entrées/sorties du L293 correspondantes n'étant pas activées, le moteur droit est court-circuité et ne tourne pas.

On remarquera qu'à certains endroits c'est l'instruction analogWrite qui est utilisée à la place de digitalWrite. L'instruction digitalWrite sert à envoyer un état haut ou un état bas

permanent. L'instruction analogWrite sert quant-à elle à envoyer un cycle d'états hauts et d'états bas. C'est en fait l'instruction qui correspond au PWM.

Sa syntaxe est la suivante : analogWrite(pin,value)

Comme expliqué dans la partie microcontrôleur, les sorties numériques possèdent une résolution de 8 bits et permettent donc de créer 256 cycles différents. Ainsi, en écrivant analogWrite(pin,255), on envoie un état haut permanent (α=1) sur la pin associée. De la même manière, avec analogWrite (pin,127) on envoie la moitié du temps un état haut et l'autre moitié un état bas (α=0,5). Enfin, avec analogWrite(pin,0) on envoie un état bas permanent (α=0). L'astuce est de représenter en pourcentage la durée de l'état haut, 255 correspondant à 100% d'état haut, 0 à 0% et 127(=255/2) à 50%. Ce pourcentage est en fait traduit par la valeur du rapport cyclique alpha. Un analogWrite(pin,255) est donc équivalent à un digitalWrite(pin,HIGH), même chose pour analogWrite(pin,0) et digitalWrite(pin,LOW). Les instructions analogWrite sont utilisés sur les sorties correspondant aux enables du L293 comme expliqué dans la partie moteur.

Parlons maintenant des instructions détectant l'objet. Nous avons pour cela trois capteurs GP2D12. Il fallait donc trois fonctions pour gérer ces capteurs, cependant c'est le même principe, seul les variables changent. Voici comment est codée celle qui gère le capteur frontal :

```
float capteurF()  // capteur frontal
{
 irValF = analogRead(irReaderF);
 irValF1=(6787.0/(irValF-3))-4.0;
 return irValF1;
}
```

Cette fonction retourne une valeur réelle, d'où le float. Il faut qu'il soit connectée à une entrée analogique de l'arduino, et pour avoir la valeur qu'elle retourne, on utilise un analogRead et on vient lire la valeur qu'on a à l'entre analogique correspondante. Dans notre cas, on va lire la valeur qu'on a à irReaderF. Nous avons trouvée sur internet une formule qui nous permet d'avoir directement la distance en cm, sans avoir besoin de nous référer aux courbes proposées par les fabricants. Nous l'avons testé et les distances mesurés correspondaient bien aux distances réelles (à une imprécision près). C'est cette distance que l'on stocke dans irValF1.

Revenons maintenant à la procédure loop( ), dans laquelle on définit quand et comment faire appel à ces différentes procédures. Voici ce que nous avons mis dans cette procédure :

```
void loop()
{
  dist_depart=capteurF();    // la distance de départ est celle retournée par le capteur frontal
  dist_droite=capteurR();    // distance retournée par le capteur droit
```

```
dist_gauche=capteurL();   //  distance retournée par le capteur gauche
table[0]=dist_depart;
table[1]=dist_droite;  // initialisation du tableau
table[2]=dist_gauche;

boolean test=false;
int i=0;                    // on va d'abord s'intéresser au capteur frontal
while(test==false && i<3)
{
  if(table[i]<dist_fixe && table[i]>10) // si un capteur retourne une distance <30 et >10, il faut que le robot ralentisse jusqu'à atteindre les 30cm spécifiés.
  {
    ralentir();
    test=true;
  }
  else if(table[i]>dist_fixe && table[i]<80) // si un capteur retourne une distance >30 et <80, il faut que le robot accélère pour réduire la distance jusqu'à atteindre les 30cm spécifiés.
  {
    accelerer();
    test=true;
  }
  i++; // on incrémente i pour vérifier la distance retournée les autres capteurs.
}

if(dist_depart<dist_fixe && dist_depart>0) // si la distance retournée par le capteur frontal est inférieur à 30cm et supérieure à 0, le robot s'arrête, l'objet continue d'avancer et quand la distance spécifiée est atteinte, il se remet en marche.
{
   motor_stop();
}

if((dist_gauche<10 || dist_gauche>80) && (dist_droite>10 && dist_droite<80)) // tourne à droite si il capte quelque chose à droite mais pas à gauche.
{
  turnRight();
```

```
  }
  else if((dist_droite<10 || dist_droite>80) && (dist_gauche>10 && dist_gauche<80)) // tourne à gauche si il capte quelque chose à gauche mais pas à droite.
  {
     turnLeft();
  }
  delay(100); //
}
```

C'est grâce à toutes ces conditions que le robot est capable de suivre un objet en mouvement situé devant lui.

5. CONCLUSIONS ET PERSPECTIVES

Aujourd'hui nous sommes assez contents de pouvoir dire que nous avons réalisé un robot suiveur qui fonctionne. Cela a été plutôt difficile étant donné le petit nombre d'heures qu'il nous était alloué, mais nous avons finalement réussi à finaliser le travail grâce à de nombreuses heures extrascolaires.

Ce projet nous a montré qu'il peut toujours arriver des petits imprévus et que même si il est impossible de les prévoir, il est nécessaire de bien organiser le travail afin d'éviter les retards. Il faut toujours mieux prévenir que guérir. Ainsi, prévoir un planning complet, avec des dates butoir pour les différentes phases de réalisation, est un bon moyen de prévention. Grâce à un tel planning, on se rend en effet plus vite compte des retards et de ce qu'il faut finir au plus vite, car si un problème survient, c'est d'autant plus de retard accumulé et c'est la panique qui s'installe. Dans la même idée, le cahier de suivi tenu chaque semaine permettait au groupe de faire le point et de se rendre compte de l'avancement du projet.

Collectivement, nous nous sommes rendu compte que répartir les différentes tâches par groupes de personnes permet d'avancer plus vite, car plusieurs tâches peuvent être réalisées simultanément. Cependant une communication reste nécessaire entre chaque groupe pour que chacun comprenne bien ses objectifs et de quelle manière il doit travailler. Par exemple, les programmeurs devaient savoir sur quelles sorties ils devaient envoyer tels états binaires pour contrôler le moteur, pour cela ils devaient communiquer avec la personne chargée de l'électronique, seule personne sachant comment fonctionne le double pont en H. Nous pensons donc que ce projet est une bonne illustration à une petite échelle de ce qu'il se passe en entreprise.

Bien que notre robot fonctionne, nous pensons que celui-ci est perfectionnable. La programmation pourrait être optimisée, elle pourrait par exemple permettre au robot d'avancer en même temps qu'il tourne, nous pourrions aussi imaginer que le robot adapte sa vitesse en fonction de la vitesse de l'objet suivi et non pas seulement essayer de le rattraper le plus vite possible. Nous pourrions également ajouter d'autres capteurs sur notre robot pour lui permettre par exemple d'éviter les obstacles situé entre lui et l'objet ou repérer un objet placé non seulement devant lui mais aussi derrière lui. Beaucoup de choses sont possible, la seule limite reste notre imagination et bien entendu le porte-monnaie.

6. BIBLIOGRAPHIE

Moteur :

[1] Tests concernant le moteur : http://www.pololu.com/docs/0J11

Une documentation très complète du moteur : http://www.pololu.com/catalog/product/61

Trois sites donnant les vitesses et couples en fonction du ratio :

http://www.robotmarketplace.com/products/AB-70097.html

http://www.superdroidrobots.com/shop/item.asp?itemid=408

http://www.er-online.co.uk/minisumo/tamiya.php

Microcontrôleur :

Les microcontrôleurs en général :
http://sti.ac-orleans-tours.fr/spip//IMG/pdf/Microcontroleur_1_.pdf
Concernant l'arduino : Arduino.cc

Capteurs :

Tests complet des capteurs :
https://wiki.engr.illinois.edu/display/ae498mpa/The+Sharp+GP2D12+Infrared+Range+Finder

Tous ces sites étaient valides à la date du 18/06/2010

7. LOGICIELS UTILISES

SolidWorks pour la modélisation 3D.
Fritzing pour le schéma électronique.
La logiciel Arduino pour la programmetion.

8. ANNEXES (NON OBLIGATOIRE)

8.1. Documentation technique (datasheet) :

Capteur :

GP2D12/GP2D15

General Purpose Type Distance Measuring Sensors

s Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : **GP2D12**
   Detecting distance : 10 to 80cm
   Distance judgement type : **GP2D15**
   Judgement distance : 24cm
   (Adjustable within the range of 10 to 80cm)
3. External control circuit is unnecessary
4. Low cost

s Applications

1. TVs
2. Personal computers
3. Cars
4. Copiers

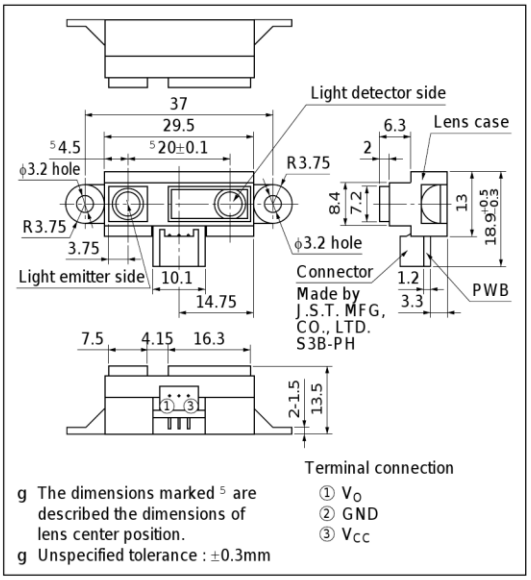
s Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	−0.3 to +7	V
Output terminal voltage	V_O	−0.3 to Vcc +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

s Outline Dimensions

(Unit : mm)



g The dimensions marked 5 are described the dimensions of lens center position.
g Unspecified tolerance : ±0.3mm

s Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V_{CC}	4.5 to +5.5	V

s Electro-optical Characteristics

(Ta=25°C, V_{CC}=5V)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range		ØL	*1 *3	10	–	80	cm
Output terminal voltage	GP2D12	V_O	L=80cm *1	0.25	0.4	0.55	V
	GP2D15	V_{OH}	Output voltage at High *1	Vcc –0.3	–	–	V
		V_{OL}	Output voltage at Low *1	–	–	0.6	V
Difference of output voltage	GP2D12	ØVo	Output change at L=80cm to 10cm *1	1.75	2.0	2.25	V
Distance characteristics of output	GP2D15	V_O	*1 *2 *4	21	24	27	cm
Average Dissipation current		I_{CC}	L=80cm *1	–	33	50	mA

Note) L : Distance to reflective object.

*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%).
*2 We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor.
*3 Distance measuring range of the optical sensor system.
*4 Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H.

Fig.1 Internal Block Diagram

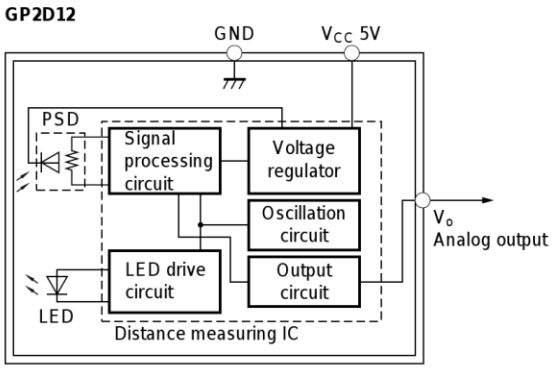


Fig.2 Internal Block Diagram

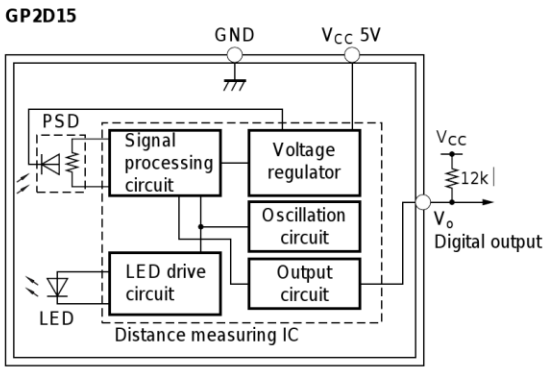


Fig.3 Timing Chart

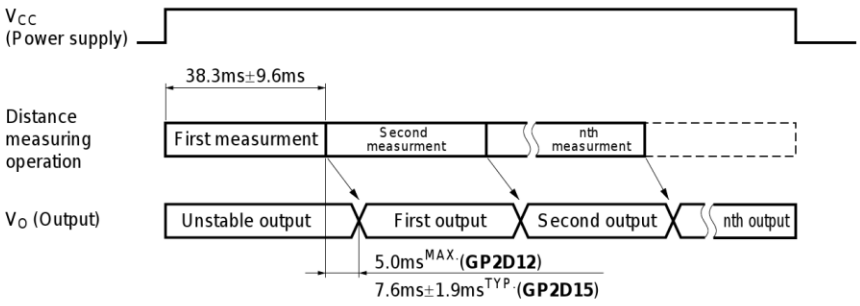


Fig.4 Distance Characteristics

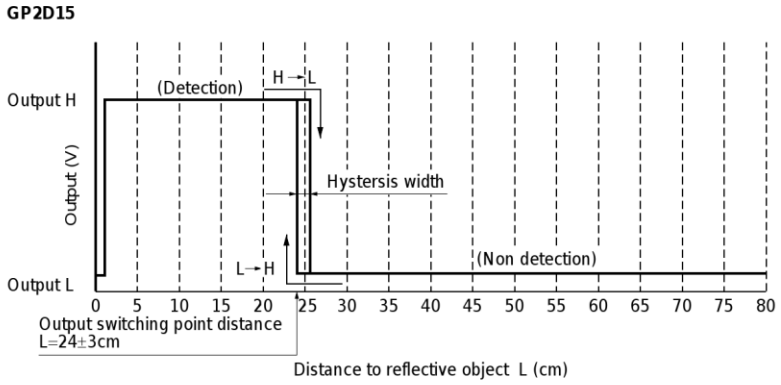


Fig.5 Analog Output Voltage vs. Surface Illuminance of Reflective Object

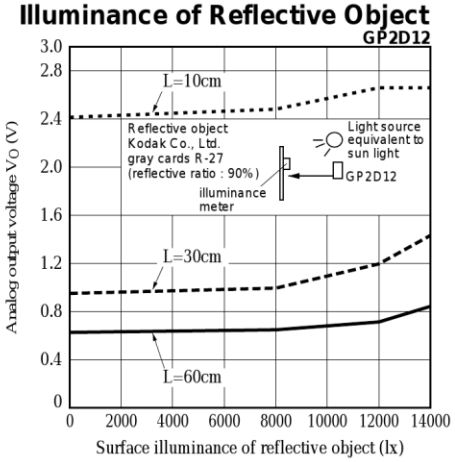


Fig.6 Analog Output Voltage vs.Distance to Reflective Object

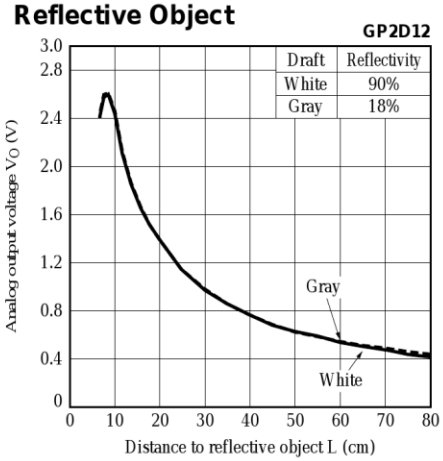


Fig.7 Analog Output Voltage vs.Ambient Temperature

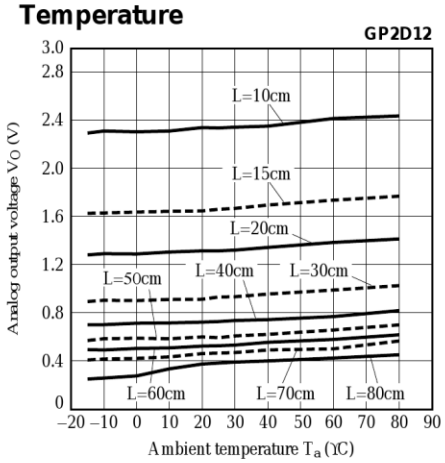
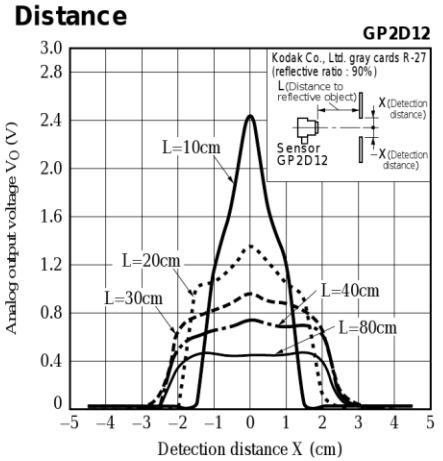


Fig.8 Analog Output Voltage vs.Detection Distance



Moteur :

FA-130RA

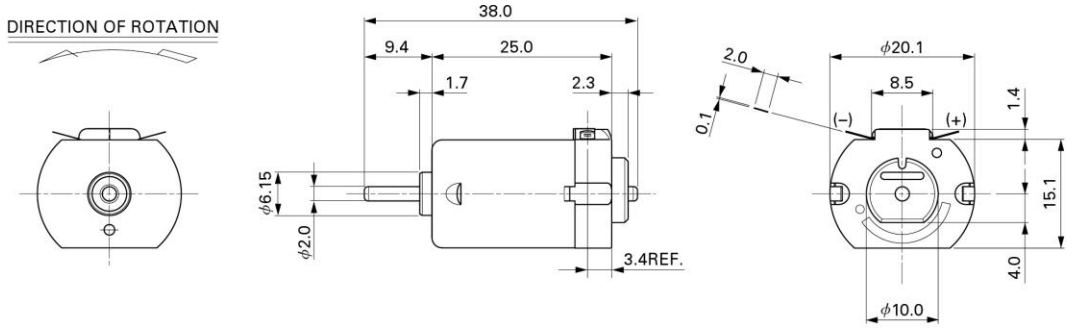
MABUCHI MOTOR
Metal-brush motors

OUTPUT : 0.2W~2.5W (APPROX)

WEIGHT : 17g (APPROX)

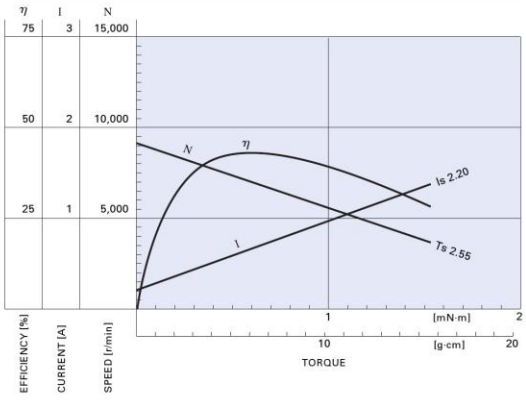
Typical Applications **Home Appliances :** Hair Dressing / Beauty Appliance
Toys and Models : Motorized Toy / Motorized Plastic Model

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN-m	g-cm	W	mN-m	g-cm	A
FA-130RA-2270	1.5~3.0	1.5V CONSTANT	9100	0.20	6990	0.66	0.59	6.0	0.43	2.55	26	2.20
FA-130RA-18100	1.5~3.0	3V CONSTANT	12300	0.15	9710	0.56	0.74	7.6	0.76	3.53	36	2.10
FA-130RA-14150	1.5~4.5	3V CONSTANT	8300	0.11	6150	0.31	0.55	5.6	0.35	2.11	22	0.90

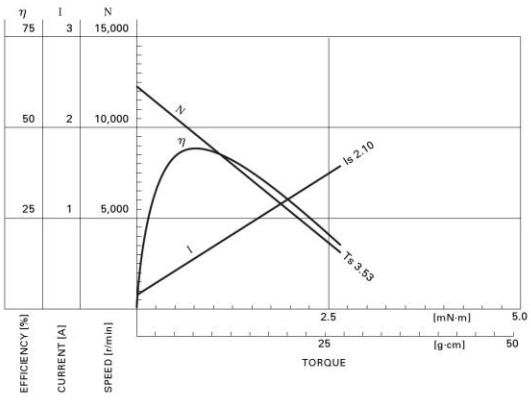


UNIT: MILLIMETERS

FA-130RA-2270 1.5V



FA-130RA-18100 3.0V



MABUCHI MOTOR CO., LTD. **Headquarters** 430 Matsuhidai, Matsudo City, Chiba, 270-2280 Japan. Tel:81-47-710-1177 Fax:81-47-710-1132 (Sales Dept.)

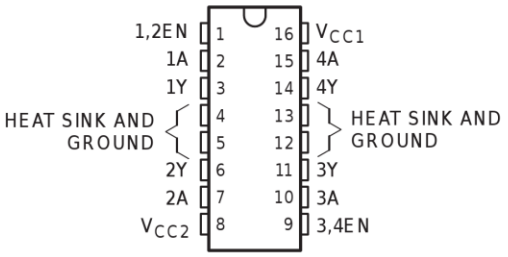
L293

L293, L293D
QUADRUPLE HALF-H DRIVERS

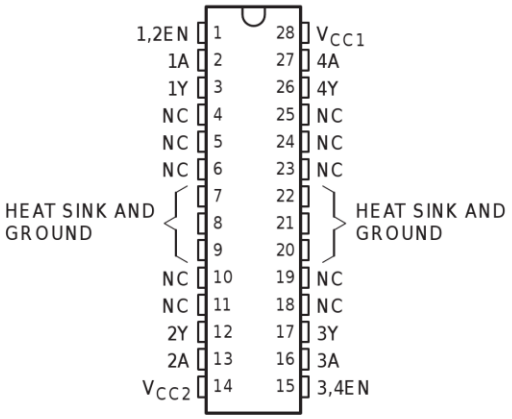
SLRS008C – SEPTEMBER 1986 – REVISED NOVEMBER 2004

- **Featuring Unitrode L293 and L293D Products Now From Texas Instruments**
- **Wide Supply-Voltage Range: 4.5 V to 36 V**
- **Separate Input-Logic Supply**
- **Internal ESD Protection**
- **Thermal Shutdown**
- **High-Noise-Immunity Inputs**
- **Functionally Similar to SGS L293 and SGS L293D**
- **Output Current 1 A Per Channel (600 mA for L293D)**
- **Peak Output Current 2 A Per Channel (1.2 A for L293D)**
- **Output Clamp Diodes for Inductive Transient Suppression (L293D)**

L293 . . . N OR NE PACKAGE
L293D . . . NE PACKAGE
(TOP VIEW)



L293 . . . DWP PACKAGE
(TOP VIEW)



description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	HSOP (DWP)	Tube of 20	L293DWP	L293DWP
	PDIP (N)	Tube of 25	L293N	L293N
	PDIP (NE)	Tube of 25	L293NE	L293NE
		Tube of 25	L293DNE	L293DNE

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

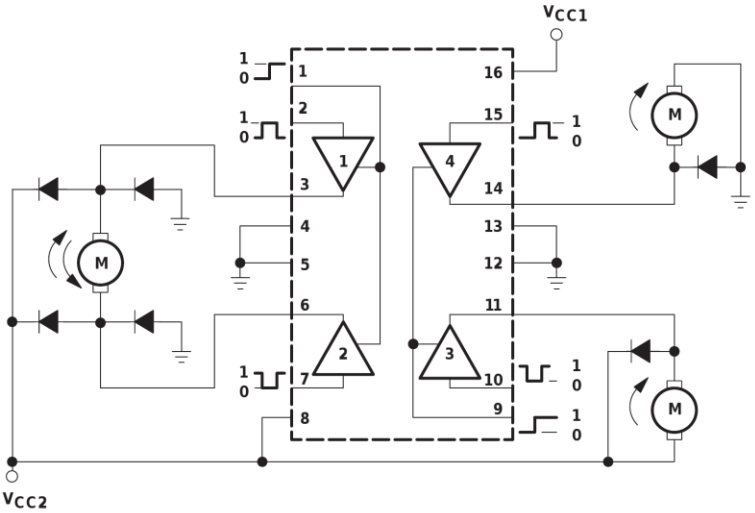
description/ordering information (continued)

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293and L293D are characterized for operation from 0°C to 70°C.

block diagram



NOTE: Output diodes are internal in L293D.

FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)
† In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

recommended operating conditions

			MIN	MAX	UNIT
	Supply voltage	V_{CC1}	4.5	7	V
		V_{CC2}	V_{CC1}	36	
V_{IH}	High-level input voltage	$V_{CC1} \le 7$ V	2.3	V_{CC1}	V
		$V_{CC1} \ge 7$ V	2.3	7	V
V_{IL}	Low-level output voltage		–0.3†	1.5	V
T_A	Operating free-air temperature		0	70	°C

† The algebraic convention, in which the least positive (most negative) designated minimum, is used in this data sheet for logic voltage levels.

electrical characteristics, $V_{CC1} = 5$ V, $V_{CC2} = 24$ V, $T_A = 25°C$

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		L293: $I_{OH} = -1$ A L293D: $I_{OH} = -0.6$ A		$V_{CC2} - 1.8$	$V_{CC2} - 1.4$		V
V_{OL}	Low-level output voltage		L293: $I_{OL} = 1$ A L293D: $I_{OL} = 0.6$ A			1.2	1.8	V
V_{OKH}	High-level output clamp voltage		L293D: $I_{OK} = -0.6$ A			$V_{CC2} + 1.3$		V
V_{OKL}	Low-level output clamp voltage		L293D: $I_{OK} = 0.6$ A			1.3		V
I_{IH}	High-level input current	A	$V_I = 7$ V			0.2	100	µA
		EN				0.2	10	
I_{IL}	Low-level input current	A	$V_I = 0$			–3	–10	µA
		EN				–2	–100	
I_{CC1}	Logic supply current		$I_O = 0$	All outputs at high level		13	22	mA
				All outputs at low level		35	60	
				All outputs at high impedance		8	24	
I_{CC2}	Output supply current		$I_O = 0$	All outputs at high level		14	24	mA
				All outputs at low level		2	6	
				All outputs at high impedance		2	4	

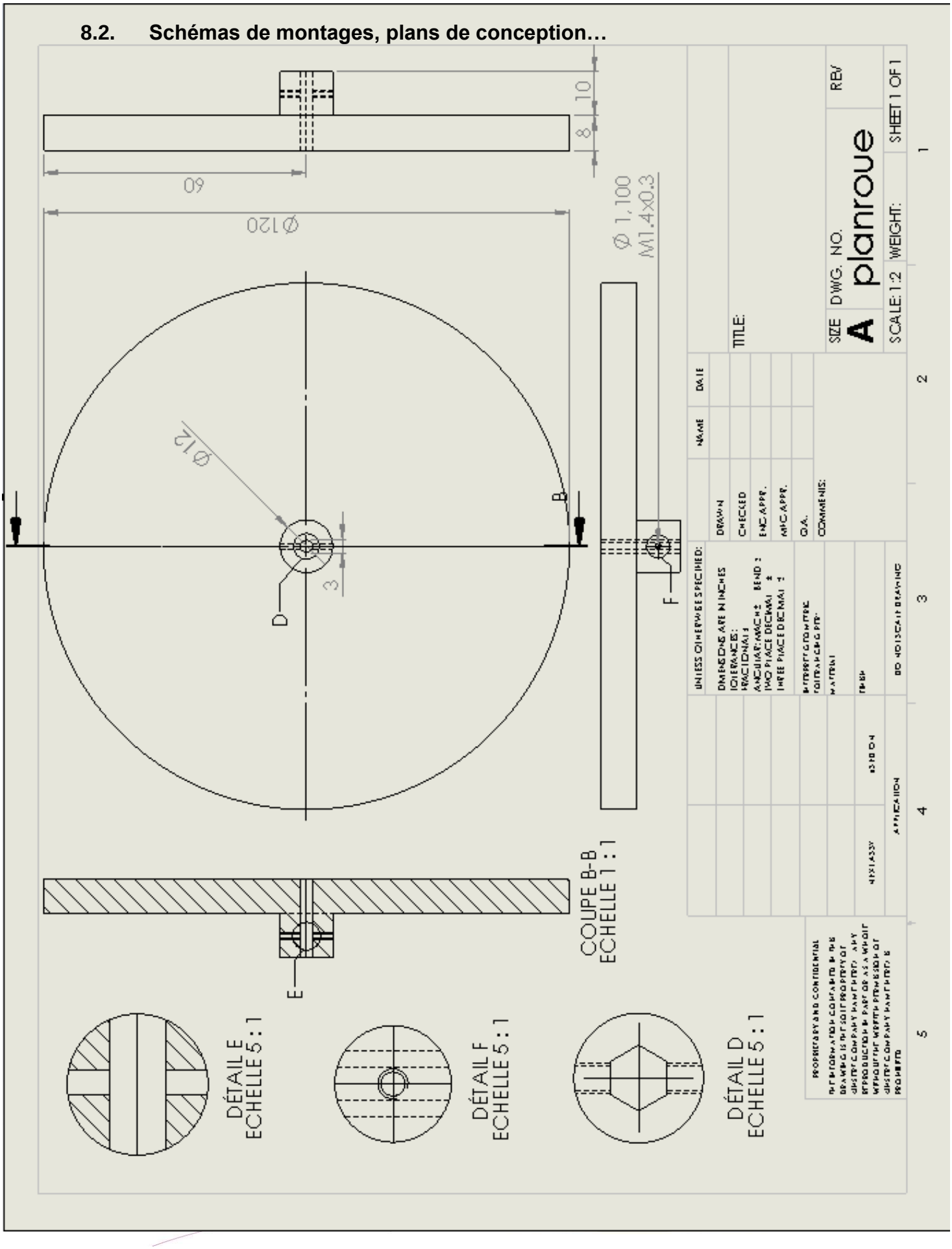
switching characteristics, $V_{CC1} = 5$ V, $V_{CC2} = 24$ V, $T_A = 25°C$

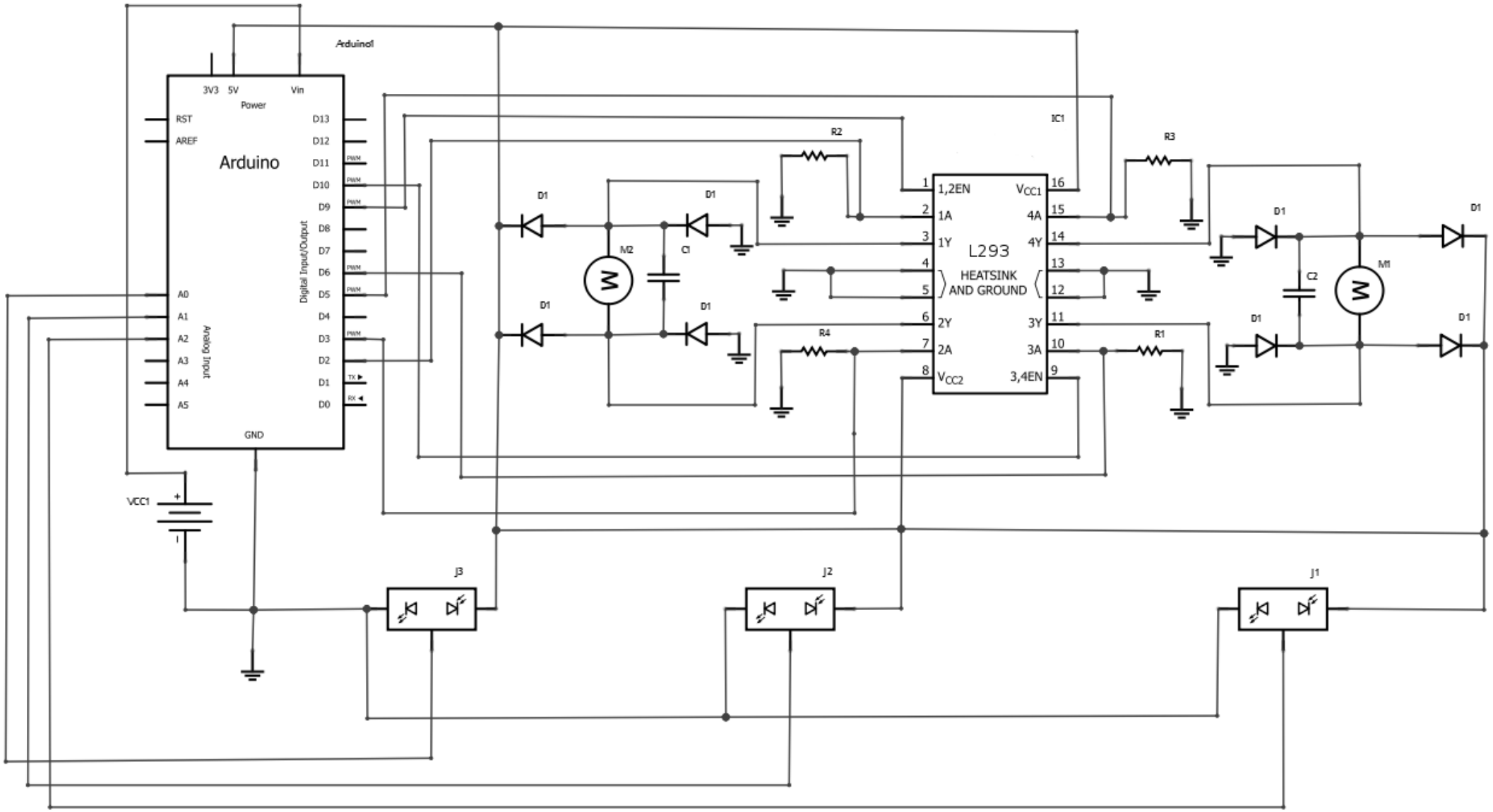
PARAMETER		TEST CONDITIONS	L293NE, L293DNE			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	$C_L = 30$ pF, See Figure 1		800		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			400		ns
t_{TLH}	Transition time, low-to-high-level output			300		ns
t_{THL}	Transition time, high-to-low-level output			300		ns

switching characteristics, $V_{CC1} = 5$ V, $V_{CC2} = 24$ V, $T_A = 25°C$

PARAMETER		TEST CONDITIONS	L293DWP, L293N L293DN			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low-to-high-level output from A input	$C_L = 30$ pF, See Figure 1		750		ns
t_{PHL}	Propagation delay time, high-to-low-level output from A input			200		ns
t_{TLH}	Transition time, low-to-high-level output			100		ns
t_{THL}	Transition time, high-to-low-level output			350		ns

8.2. Schémas de montages, plans de conception…




Arduino1
Arduino
3V3
5V
Vin
Power
RST
AREF
D13
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
D1
D0
PWM
Digital Input/Output
A0
A1
A2
A3
A4
A5
Analog Input
GND
VCC1
IC1
L293
1,2EN
1A
1Y
HEATSINK
AND GROUND
2Y
2A
VCC2
VCC1
4A
4Y
3Y
3A
3,4EN
R1
R2
R3
R4
C1
C2
M1
M2
D1
J1
J2
J3