Projet TERI
Réalisation d'un robot quadrupède

Objectif :

Développer et réaliser un projet système embarqué à travers les domaines de l'électronique, de la mécanique et de l'impression 3D.

Description du système :

Le système à étudier consiste en la réalisation d'un « robot marcheur » à quatre pattes (d'où l'appellation quadrupède). La structure imprimée en 3D sera animée par un ensemble de 8 servomoteurs SG90 (9g) (2 à chaque patte), contrôlés par un microcontrôleur ATmega2560. Ce dernier est utilisé via une carte Arduino MEGA et le programme qu'il intègre permettra un déplacement automatisé du système grâce à un capteur d'obstacles à ultrasons type HC-SR04 ainsi qu'une étude rapide de la mécanique de déplacement. Une alimentation portative (batterie) adaptée aux besoins du robot nous permettra d'avoir un système autonome pendant une durée dépendant de la charge de la batterie.

Table des matières

Composants et réalisation:

Le robot que nous projetons de faire nécessite plusieurs composants que nous allons devoir faire fonctionner les uns avec les autres. Nous avons choisi une structure imprimée en 3D qui sera articulée par des moteurs eux-mêmes pilotés par un microcontrôleur ATmega 2560 (via Arduino Mega). Un capteur donnera au robot la possibilité de détecter les obstacles.

Ainsi un module mécanique comprenant les moteurs et la structure sera piloté par un module électronique constitué du capteur et de la carte Arduino Mega. L'ensemble est alimenté par un module d'alimentation. La suite permet de détailler la composition des modules et leur fonctionnement.

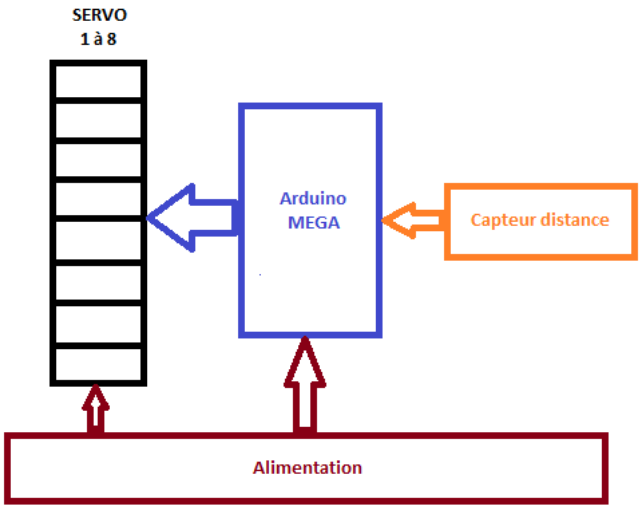


Figure 1 : Schéma des interactions entre les différents composants des modules.

Module électronique :

Le module électronique est composé de la carte Arduino MEGA (ATmega 2560) et du capteur de distance à ultrasons HC-SR04. On peut retrouver les documentations techniques des composants en annexe.

- ***Arduino MEGA***

Commençons par un rapide regard sur les caractéristiques de l'Arduino MEGA qui nous intéresse pour ce projet. On dispose ici d'un contrôleur ATmega 2560, proposant six timers différents. Les timers 0 et 1 sont sur un registre TCNT de 8 bit (soit 256 valeurs possibles), tandis que les timers 1, 3, 4, 5 sont sur des registres de 16 bits (soit 65536 valeurs possibles. De plus à la différence de l'ATmega 328 que l'on retrouve sur l'arduino UNO, on a la possibilité de générer 3 signaux différents par timer de 16 bits. Dans notre cas, cela est essentiel puisque nous cherchons à gérer un déplacement par pilotage de servos-moteurs via génération de signaux PWM. Or nous avons 8 servos moteurs à piloter indépendamment, il est donc nécessaire de générer au moins 8 signaux différents (on peut piloter jusqu'à 12 servos-moteurs en PWM tandis que l'arduino UNO et le 328 ne nous en propose que 6).

Les 4 signaux restant générés par des timers 16 bits pourront nous permettre d'ajouter par la suite un servo-moteur à chaque patte du robot (cf partie Finalisation). L'ensemble des timers sont cadencé par une fréquence d'horloge de 16MHz qu'il est possible de diviser par des facteurs prédéfinis.

D'autres caractéristiques de la carte comme le grand nombre d'entrées/sorties, la communication USART et une taille mémoire augmentée sont un plus mais ne sont pas déterminante dans notre cas. A l'inverse, la taille de la carte est un inconvénient.

Afin de réduire le prix élevé de la carte nous avons opté pour une carte Elegoo disposant des mêmes caractéristiques que l'Arduino.

Figure I.2: Carte Elegoo MEGA utilisée

- ***Capteur à ultrasons HC-SR04***

L'objectif de notre projet est d'avoir un système automatique et non commandé à distance, il est nécessaire que le robot puisse voir ce qui l'entoure. Nous avons simplifié cela par « voir ce qu'il y a devant lui » ... Toutefois une amélioration est proposée à la fin de ce rapport.

Nous avons choisi le capteur HC-SR04 pour plusieurs raisons au-delà du fait qu'il donne à ce robot marcheur un super look ! Premièrement, ce capteur a la capacité de repérer les obstacles jusqu'à une distance de 5m sous un angle de 30°. Ensuite le fonctionnement de ce capteur est basé sur de la détection par ondes ultrasonores avec une fréquence et un nombre d'impulsion définis, ce qui diminue les possibilités d'interférence avec d'autres sources. Son faible prix en fait aussi un choix intéressant pour les caractéristiques proposées.

Figure I.3: Capteur à ultrasons HC-SR04

Les inconvénients à se capteur sont aussi liés aux avantages qu'il procure. En effet, les ultrasons envoyés reviennent au capteur après avoir rebondis sur un obstacle. Ce dernier évalue le temps entre l'envoi et la réception et il en déduit la distance. Par conséquent si l'objet est trop petit ou si sa surface est trop arrondie, il ne renverra pas correctement les ultrasons et ces derniers ne retourneront pas au capteur. Ainsi l'obstacle ne sera pas détecté.

Afin d'obtenir une mesure de la distance, il va falloir envoyer une requête au capteur. On applique pour cela une impulsion de 10µs sur la broche Trig. Suite à cela l'émetteur va générer une séquence de 8

impulsions à une fréquence de 40kHz. A la réception de cette onde, on détecte la présence d'un objet à une distance proportionnelle au temps qu'à mis l'onde à se propager.

$$Distance = Vitesse_{SON} \times \frac{Temps}{2}$$

La transmission de l'information se fait par l'envoi d'une impulsion d'une durée en dizaine de microsecondes proportionnelle au temps séparant l'émission de la réception de l'ultrason.

On retrouve l'équation suivante : $Distance = Vitesse_{SON} \times 10^{-5} \times \frac{Temps}{2}$

Module mécanique

Le module mécanique est lui constitué des servos moteur et de la structure imprimée en 3D.

- ***Servo-moteurs***

Les servomoteurs SG90 sont les plus utilisé avec Arduino, les servomoteurs sont capables de maintenir une position statique, ils sont des moteurs a courant continu pilotés par un driver qui régule constamment leur position (système asservi) à l'aide d'un potentiomètre. La quasi-totalité des servo-moteurs comportent un réducteur qui sert à augmenter leur couple, pour piloter un servomoteur on utilise un PWM est c'est le rapport cyclique de la PWM qui détermine l'angle de positionnement. Nous avons donc câblé la commande du servomoteur sur une sortie PWM de l'Arduino MEGA (OCnA/B/C).

Nous retrouvons ainsi les branchements suivants :

- ✓ Le fils marron doit être connecté à la masse
- ✓ Le fil rouge au 5V
- ✓ Le fil orange a la sortie PWM

Autre chose très important : le servomoteur est limité à un débattement de 180°, donc interdit de faire un tour complet. Avant de mettre en place notre bras de fixation il faut initialiser le servomoteur dans une position de référence qui se trouve être la moitié de son amplitude afin d'avoir la marge de déplacement du bras la plus grande possible.

Avant de passer à la suite, nous avons fait des tests afin de déterminer le courant consommé par les servos-moteurs en activité. Cependant nous n'avons fait que des études rapides en appliquant arbitrairement des forces différentes à des endroits différents. Il est important de noter que plus la force à fournir (qui dépend du poids à déplacer ou de la résistance appliquée) pour maintenir la position est proportionnelle au courant consommé.

Activité	Consommation
Repos	100 mA
Déplacement	200-250 mA
Résistance maximale	300-350 mA

9g correspond à la masse qui peut être soutenue par le servo-moteur au repos à 1cm de son axe.

Figure I.3: Servo-moteur SG90.

Le PWM doit être d'une fréquence de 50Hz. On va donc devoir utiliser un fast PWM avec la grandeur du registre ICRn en TOP. Il faut penser que le rapport cyclique va de 1/20ms pour un angle de 0° à 2/20ms pour un angle de 180°.

- ***Structure 3D***

Bien que nous ayons choisit au final un modèle préexistant pour notre robot dans un soucis de priorisation des étapes liées à l'électronique et à la programmation, nous avons d'abord choisi d'imprimer la structure en 3D pour l'ensemble des solutions proposées par la conception assistée par ordinateur.

L'ensemble des pièces ont été imprimées grâce à la technologie FDM (dépôt de fil chaud) en PLA ou en ABS. L'avantage de cette technologie est le faible coût de réalisation et à l'ensemble des paramètres qu'il est possible de choisir. Le PLA et l'ABS sont deux des matériaux les plus utilisés dans l'impression 3D. Avec une température d'extrusion de 205°C et une sensibilité bien moins importante aux changements de température, le PLA est bien plus facile à imprimer que l'ABS, ce dernier étant plus sensible au warping que l'on observe sur des pièces planes (des courbures finissent par apparaitre suite à la tension vers le centre de la pièce induite par le refroidissement du plastique quelques centimètres au-dessus du plateau chauffé à 60°C).

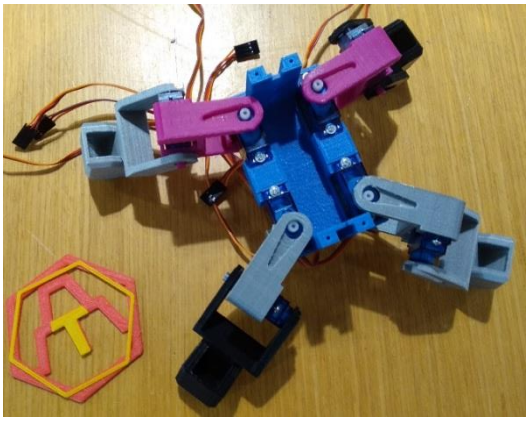


Figure 1.4: Le robot imprimé en 3D assemblé.

Ainsi nous avons dû réaliser quelques pièces en PLA, bien que ce dernier soit moins résistant aux chocs. Pour contrer cela, nous avons choisi une densité interne classique (15%) avec une structure alvéolaire. De plus une coque plus épaisse que la moyenne (0,7 mm) a permis de renforcer la structure externe. Les pièces ont été imprimées au fablab AsTech sur le campus de la Doua, sur une Ultimaker 2+ et une Makerbot Replicator 2x.

Module d'alimentation

L'Arduino étant limitée à environ 25mA pour l'envoi de courant par ses pins définis en sortie, il était nécessaire de créer un module pour alimenter l'ensemble du système. Pour cela le module d'alimentation est constitué d'une batterie et d'un convertisseur buck permettant d'abaisser la tension à une valeur réglable.

- ***Batterie***

Deux caractéristiques importantes de la batterie sont sa capacité (Ah) et la tension de sortie (V). Une batterie est constituée de plusieurs cellules, chacune ayant une capacité et une tension de sortie. Ainsi on peut faire varier les caractéristiques en jouant sur la mise en série ou en parallèle de différentes batteries. Mettre deux batteries en parallèle multipliera la capacité totale par deux mais la tension totale restera identique à celle d'une cellule (loi de Kirchhoff). On retrouve l'inverse pour la mise en série de 2 cellules.

Une des grosses problématiques de la batterie est son poids. Il est important de le minimiser si l'on veut que le servo fonctionne, mais ensuite si on veut éviter de trop consommer. Augmenter le nombre de cellule en parallèles va certes augmenter la capacité, mais cela ne sert à rien si l'on consomme plus…

Pour ces raisons nous avons choisis deux batteries, l'une est constituée de deux cellules LG18650 récupérées (LI-ION, 3.6V, 2200mAh) et l'autre est une batterie NiMH (8.4V, 1600mAh). La deuxième est bien moins performante et plus lourde mais nous a servi pour faire des tests « sûrs » …

Les batteries fournissant difficilement du 5V, il était nécessaire de convertir la tension. De part les valeurs de consommations des servos-moteurs et des étapes nécessaires au déplacement (cf partie sur le déplacement), il est tout à fait possible de se retrouver avec plus d'1A tiré sur la batterie en une fois. Pour cela on a choisi d'utiliser un hacheur plutôt qu'un régulateur, étant donné que ces derniers ne tolèrent pas plus d'1A.

Nous avons ajouté un interrupteur afin d'éteindre et d'allumer le robot. On aurait pu ajouter une led d'état, on pourra le faire directement via l'Arduino.

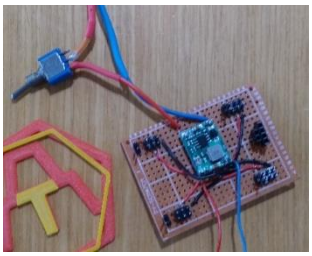


Figure I.5: Module d'alimentation et zones de connections aux différents composants.

Budget :

Il est important dans un projet de se projeter sur le coût de l'ensemble des composants pour ne pas avoir de mauvaises surprises.

PRODUIT	COUT/U (€)	QUANTITE	COUT TOTAL (€)	SOURCE	COUT REEL
ARDUINO MEGA	12.99	1	12.99	Amazon.fr	12.99
HC-SR04	0.63	1	0.63	Aliexpress.com	0.63
SERVO-MOTEUR	2.40	8	19.2	Amazon.fr	19.2
PIECES 3D (AU GRAMME)	0.11	85	9.35	Fablab AsTech	9.35
BATTERIE	23	1	23	Amazon.fr	0
CONVERTISSEUR	1.59	1	1.59	Amazon.fr	1.59
INTERRUPTEUR	0.84	1	0.84	Amazon.fr	0
FIL (AU METRE)	0.4	0.6	0.21	Amazon.fr	0
PLAQUE A TROUS	0.5	1	0.5	Fablab AsTech	0.5
CONNECTEURS BATTERIE (PAIRE MALE/FEMELLE)	1.4	1	1.4	Amazon.fr	1.4
CONNECTEURS PINS	0.008	38	0.33	Amazon.fr	0

On compte ainsi un coût total de **70,04€** ramené à **45,66€** avec les différents produits récupérés sur des objets déjà existants ou mis à disposition.

L'ensemble étant amené à rester en possession du groupe et a être amélioré, nous n'avons pas démarché la faculté pour ce budget.

Etude de la mécanique de déplacement

Initialisation et référence

- ***Introduction***

La séquence d'initialisation consiste en l'assurance que les servos-moteurs sont positionnés correctement. Elle est liée à une position de référence où tous les servos sont au milieu de leur amplitude de déplacement (90° pour une amplitude totale de 180°).

De cette manière il est possible de fixer les pattes aux servos de façon à ce quelle soient orientée correctement à savoir :

- Horizontal pour une rotation autour de l'axe z
- Vertical pour une rotation autour de l'axe x

Grâce aux calculs effectués, nous auront une position de référence comme suit :

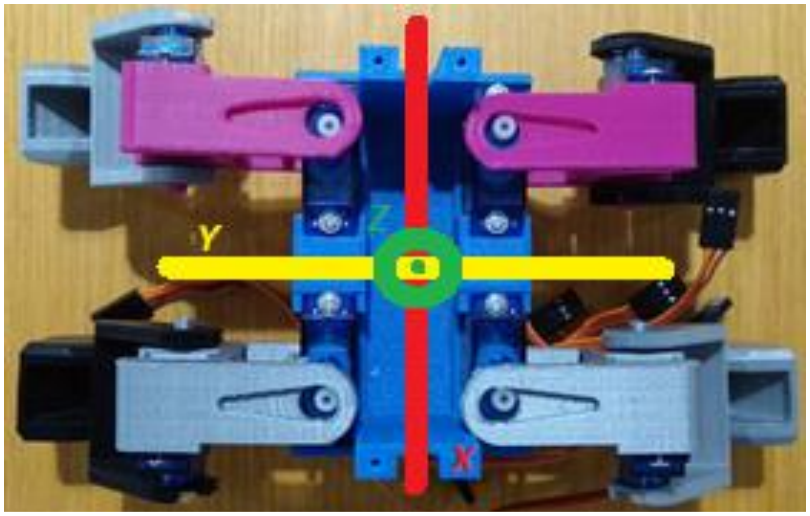


Figure II.1: Robot à la position de référence avec représentation des axes évoqués.

- ***Calcul des angles de référence***

On va chercher ici quels angles doivent avoir les servos-moteurs afin que le robot dispose d'une position de référence de laquelle pourront partir tous les autres mouvements. On pourra aussi s'assurer que l'assemblage des pièces est fait correctement.

Pour rappel, il y a deux servos-moteurs par bras. On définit les servos « épaules » qui tournent autour d'un axe vertical (axe z) servant à orienter les bras dans le plan horizontal, et les servos « coudes » qui tournent autour d'un axe horizontal (composition des axes x et y, seulement x pour la position de référence) servant à lever les pates (orientation dans le plan vertical).

- Servos « épaules » :

Pour ces servos-moteurs, la pièce qu'ils dirigent n'est pas coudée. Par conséquent l'angle que fait le servo par rapport à sa référence est égale à l'angle que va faire la pièce avec le corps (on a angleServo = anglePièce). Nous réglerons cet angle à 90° par défaut afin de se retrouver dans la position que l'on observe ci-dessus.

- Servos « coudes » :

Au contraire des moteurs précédents, les pièces en plastique qui sont dirigés par les servos moteurs sont légèrement coudées, ce qui implique un angle différent entre le servos moteur et l'extrémité de la pièce. En effet on va avoir une valeur d'offset, c'est-à-dire qu'en choisissant un angle pour le bras du servo moteur par rapport à la verticale, on va avoir cet angle auquel va s'ajouter un delta que l'on va chercher à déterminer.

L'angle a été mesuré et reporté sur la figure suivante :

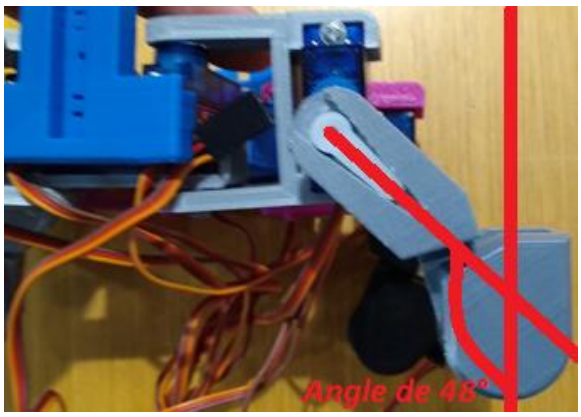


Figure II.2: Angle que fait le bras du servo « coude » avec la verticale à la position de repos.

Par conséquent notre servo qui peut aller de 0 à 90° va déplacer l'extrémité du bras d'un angle comprit entre 48 à 138°. Cet effet peut être simplement compensé de manière matérielle en fixant le bras avec un angle de 48° opposé à l'angle de la pièce. Cette méthode nous permet de ne pas réduire l'amplitude du servo qu'il aurait fallu faire en le prenant en compte dans la programmation.

- ***Placement des bras***

Par conséquent on retiendra un angle de 90° (milieu de l'intervalle des positions disponibles) pour tous les servos quand le microcontrôleur choisira la position de référence. On placera ensuite les bras de façon que ceux des servos « épaules » soient perpendiculaire au corps et ceux des servos « coudes » fassent un angle de 48° avec la verticale.

En général, il serait aussi important de considérer une marge à laisser à l'ensemble des servos pour éviter de les amener en butée et risquer de les endommager. Toutefois après quels tests on se rend compte la marge est déjà calculée par le constructeur quand il nous est garantit une amplitude de 180° (environ 30° au-delà de la limite pour la plupart des servos).

Marche avant

Dans la plupart des déplacements, on fera en sorte d'avoir toujours au moins trois points de contact avec le sol. Bien que réduisant la vitesse de déplacement, cela nous permettra de nous affranchir de la question d'équilibre.

La procédure de déplacement vers l'avant va être segmenté en 6 étapes (3 pour chaque côté) plu une pour revenir à la position de départ de la séquence.

- ***Déplacement d'un point d'appui***

On va définir un point d'appui comme étant l'endroit du contact entre le sol et une patte du robot. Une succession de déplacement des points d'appuis permet le déplacement du robot.

Pour changer le point d'appui d'un bras du robot, il est nécessaire d'effectuer trois étapes :

- Enlever le contact avec le sol en faisant varier l'angle du servo « coude » ;
- Déplacer l'ensemble du bras en faisant varier l'angle du servo « épaule » ;
- Refaire contact avec le sol en faisant la variation opposée à la première étape sur le servo « coude ».

Ces trois étapes permettent d'aller chercher un nouveau point d'appui, mais ne permettent pas pour autant un quelconque déplacement du robot. Pour cela il faut déplacer le point d'appui (dans le référentiel du robot) sans en changer. Afin d'avancer en ligne droite, on choisira un déplacement de ce nouveau point d'appui jusqu'à la dernière position du point d'appui précédent. En d'autres termes, le déplacement angulaire que l'on fait faire au servo « épaule » à l'étape 2 de la séquence ci-dessus doit à nouveau être effectué, mais dans l'autre sens cette fois.

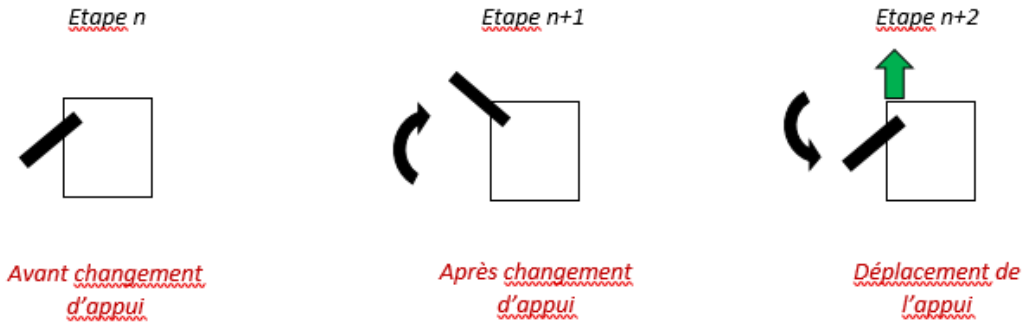


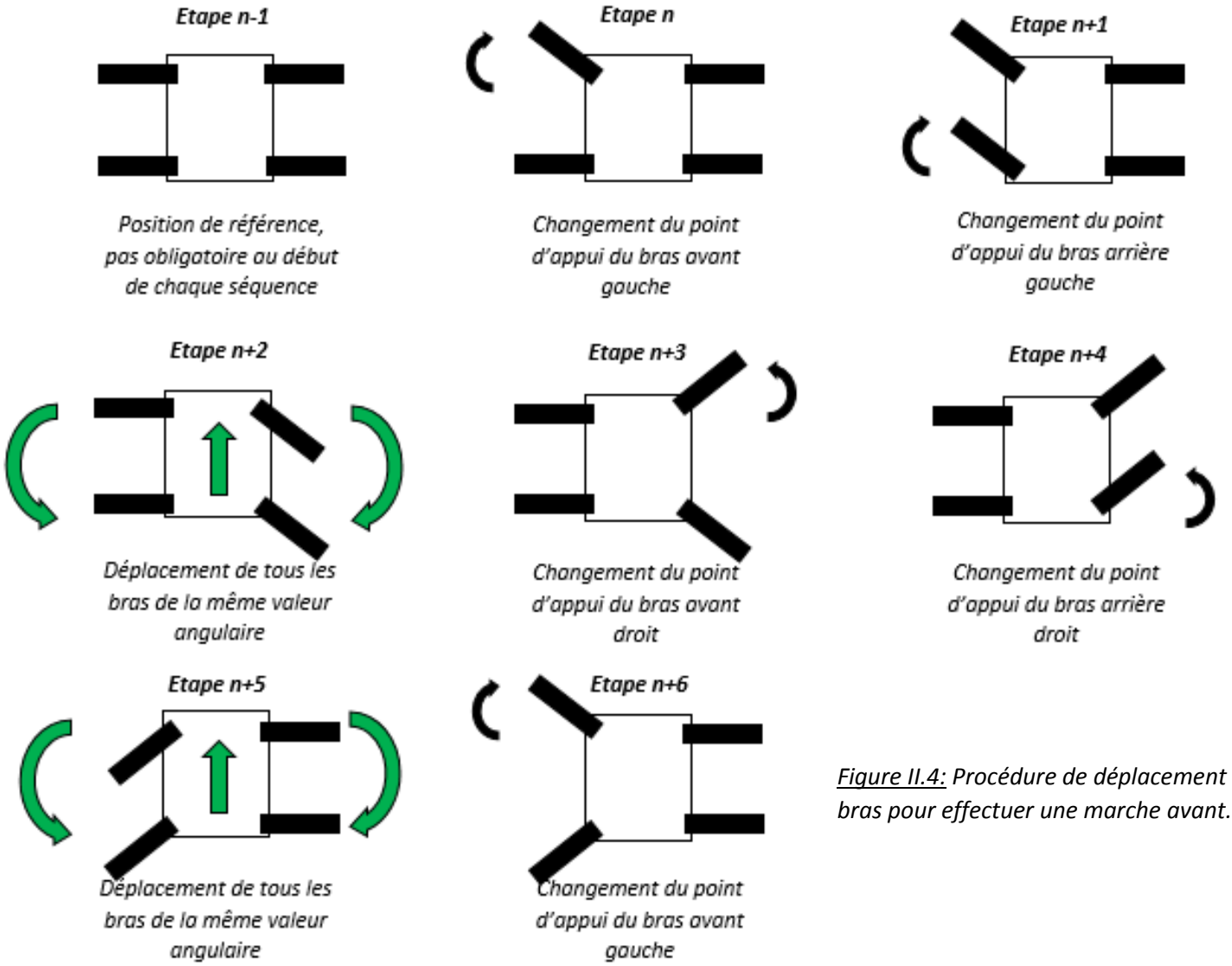
Figure II.3: 3 étapes essentielles pour aller vers l'avant.

Finalement le déplacement d'un appui requiert 3 étapes de changement d'appui et une étape de déplacement.

- ***Déplacement vers l'avant :***

Afin de simplifier l'ensemble des explications dans cette partie, on ne détaillera pas les étapes constituant le déplacement des points d'appuis. Ainsi seuls les servos « épaules » sont concernés.

Comme précisé dans la partie « déplacement d'un point d'appui », le but est d'avoir la même position du bras aux étapes n et n+2. En effet nous cherchons à aller en ligne droite, ce qui implique un déplacement identique des deux côtés du robot – à l'image d'un rameur, il faut tirer de la même force sur les rames pour faire avancer la barque en ligne droite. Ainsi il est aussi important de synchroniser les déplacements. La marche avant sera donc constituée d'une phase de changement des points d'appui et d'une phase de déplacement de tous les points d'appuis en une fois. Un schéma représentant les étapes successives étant plus facile à interpréter que des mots dans ce cas, on obtient la séquence suivante en partant de la position de référence (on précise que l'avant du robot se trouve vers le haut de la page) :



<u>Figure II.4:</u> Procédure de déplacement des bras pour effectuer une marche avant.

A l'étape n+6 on remarque qu'on recommence une séquence de déplacement (on revient à l'étape n, cependant il y a une différence par rapport à l'étape *n* puisque celle-là est issu de la position de référence).

Le choix de commencer par l'avant pour aller vers l'avant permet d'éviter le cas éventuel où les pates se touchent les unes les autres.

Choix des angles de déplacement :

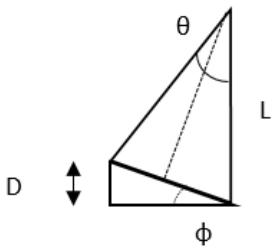
Une fois la procédure définit, nous remarquons qu'il n'y a finalement que deux positions caractéristiques pour chaque servo. Ainsi nous allons essayer de les définir pour savoir quels signaux le microcontrôleur va envoyer.

Pour les servos « coudes », le but est de ne pas prendre un angle trop grand qui risque de déséquilibrer le robot en déplaçant son centre de gravité, ni de prendre un angle trop petit qui entrainerait un bras du robot à frotter par terre ou à heurter un petit obstacle. De plus un angle trop grand nous ferait perdre de la vitesse de déplacement. Afin de faciliter les choses, on se mettra au-delà de la contrainte minimale, entrainant théoriquement le moins de problèmes pour l'équilibre sans avoir à calculer un angle maximal (pour le min on doit simplement prendre en compte une hauteur alors que pour le max on doit considérer une répartition du poids…). En clair on va faire en sorte de prendre un angle pour lequel l'effet du déplacement de masse est négligeable tout en respectant un minimum.

Le robot étant censé évoluer sur un sol lisse, on n'a en soi pas de limite minimale. Toutefois il faut prendre des précautions, et quand on pense que rehausser légèrement cette valeur permettrait d'évoluer sur des sols plus abrupts, on est forcément tentés…

Arbitrairement on va demander au servo de se lever de 1cm. Avec un sinus et en assimilant la courbe de la trajectoire à une droite on peut retrouver l'angle que l'on va devoir effectuer :

Avec comme appui le schéma suivant :



θ est l'angle que fait le cerveau moteur avec la verticale.

L est la taille du bras du cerveau moteur.

D est la hauteur de laquelle on veut lever le bras.

Figure II.5: Schéma pour le calcul des angles en fonction d'une hauteur à atteindre.

On veut lier D à θ. On va nommer h l'hypoténuse du triangle inférieur, on obtient ainsi deux équations, une pour chaque triangle :

$$\begin{cases} sin\frac{\theta}{2} = \frac{h}{2L} \\ \sin\varphi = \frac{D}{h} \end{cases}$$

En isolant h et en remplaçant on retrouve :

$$2L\sin\frac{\theta}{2} = \frac{D}{\sin\varphi}$$

Il ne nous manque que ϕ, on va faire un travail rapide sur les angles :

$$\frac{\theta}{2} + 90 + (90 - \varphi) = 180 \quad ce\ qui\ nous\ donne \quad \frac{\theta}{2} = \varphi$$

On obtient finalement :

$$D = 2L\ sin^2(\frac{\theta}{2})$$

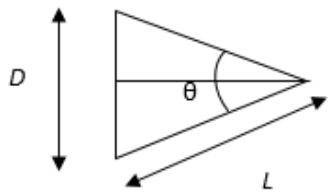
$$\theta = 2\arcsin(\sqrt{\frac{D}{2L}})$$

Si on veut D=1cm, il faut un angle θ de **34,15°** avec la verticale en mesurant un L de 5,8cm (en ayant prit une ligne droite depuis l'axe du servo jusqu'au point de contact avec le sol). L'angle de 34° est en fait une différentielle à avoir par rapport à la référence, en effet nous avons prit la verticale pour faciliter le calcul, mais il s'avère que le bras servo n'est pas positionné à la verticale pour la référence (cf « calcul des angles de référence »).

En ce qui concerne les servos « épaules », on se rend compte dans le détail sur la marche avant que la valeur du déplacement d'un point d'appui est directement liée à l'angle par une relation trigonométrique (on considérera l'arc de cercle comme étant une droite) :

$$2\sin\frac{\theta}{2} = \frac{D}{L}$$

Avec comme appui le schéma suivant :



θ *est l'angle parcouru par le bras du robot*

D est la distance sur laquelle se déplace le point d'appui

L correspond à la taille du bras du robot

Figure II.6: Schéma pour le calcul des angles en fonction d'une distance D à parcourir.

<u>Remarque</u> : Il pourrait y avoir une ambigüité sur la distance caractérisée par D. En effet si l'on se réfère au schéma du ci-dessus et à la figure II.4 concernant les déplacements, on remarque que le robot se déplace bien de D, mais au bout d'une étape de déplacement total. Ainsi cela peut porter à confusion en ce qui concerne la vitesse. Si l'utilisateur pense que le robot va se déplacer de D au bout de 3 cycles (figure II.4), ce ne sera pas le cas (on va se déplacer de D/2 tous les 3 cycles), on aura donc une vitesse de déplacement plus faible que souhaitée…

Afin de rester cohérent avec l'ensemble des schémas réalisés ainsi qu'avec les explications précédentes, nous considérerons que l'on se déplace de D au bout d'une séquence entière. Cependant il faudra diviser la distance D par 2 puisque l'on calcul l'angle à prendre pour positionner les pates sur un déplacement de 3cycles.

On va considérer un déplacement de 10cm au bout d'une séquence (deux déplacements de 2 fois 5 cm, ce qui correspond à deux fois la position à un angle θ/2) :

$$\theta = 2arcsin\frac{D}{4L}$$

L vaut 3,2cm après mesure (distance entre les deux endroits de la pièce d'où sortent et entre les axes des servos), ce qui nous donne un angle θ de **102,75°**. Ainsi on devra faire deux demi-séquences de déplacement (étapes 1 à 3) avec un angle de **51,375°** pour pouvoir avancer de 10cm.

Marche arrière :

A l'image de la marche avant, on procédera de la même manière mais en comptant les angles en négatif par rapport à la référence (90°-θ).

De plus on fera en sorte de commencer par faire reculer les pates qui sont vers l'arrière.

Pivot :

La procédure de pivot permet, à la détection d'un obstacle, de faire un virage d'un angle prédéfini par l'utilisateur. Ainsi on va considérer un axe de référence qui sera l'axe dans le sens de la longueur du robot.

L'objectif va être de pivoter le robot d'un angle lui permettant de changer de direction. Plus l'obstacle va être près, plus l'angle va devoir être important. Nous choisirons arbitrairement un pivot d'un angle de 45° pour un angle détecté à 50cm.

La direction du pivot sera définie par le fait que l'obstacle se trouve sur la droite ou la gauche du robot.

Afin de garder une procédure la plus simple possible, on va procéder de la manière suivante :

A l'image de la marche avant, l'objectif est ici de garder un maximum de contact avec le sol. Les déplacements pourraient être mieux optimisés mais nous préférons commencer avec quelque chose de simple qu'il sera plus tard possible d'améliorer.

Nous allons chercher à déplacer les points de contact un à un, puis à déplacer les servos « épaule » en même temps pour pouvoir faire pivoter l'engin. On part de la position de référence et on cherche à y revenir.

On va donc déplacer les servos épaules de 45° dans un sens (droite ou gauche) pour déplacer les points d'appuis puis l'on « validera » l'ensemble en les faisant pivoter de 45° dans le sens opposé :

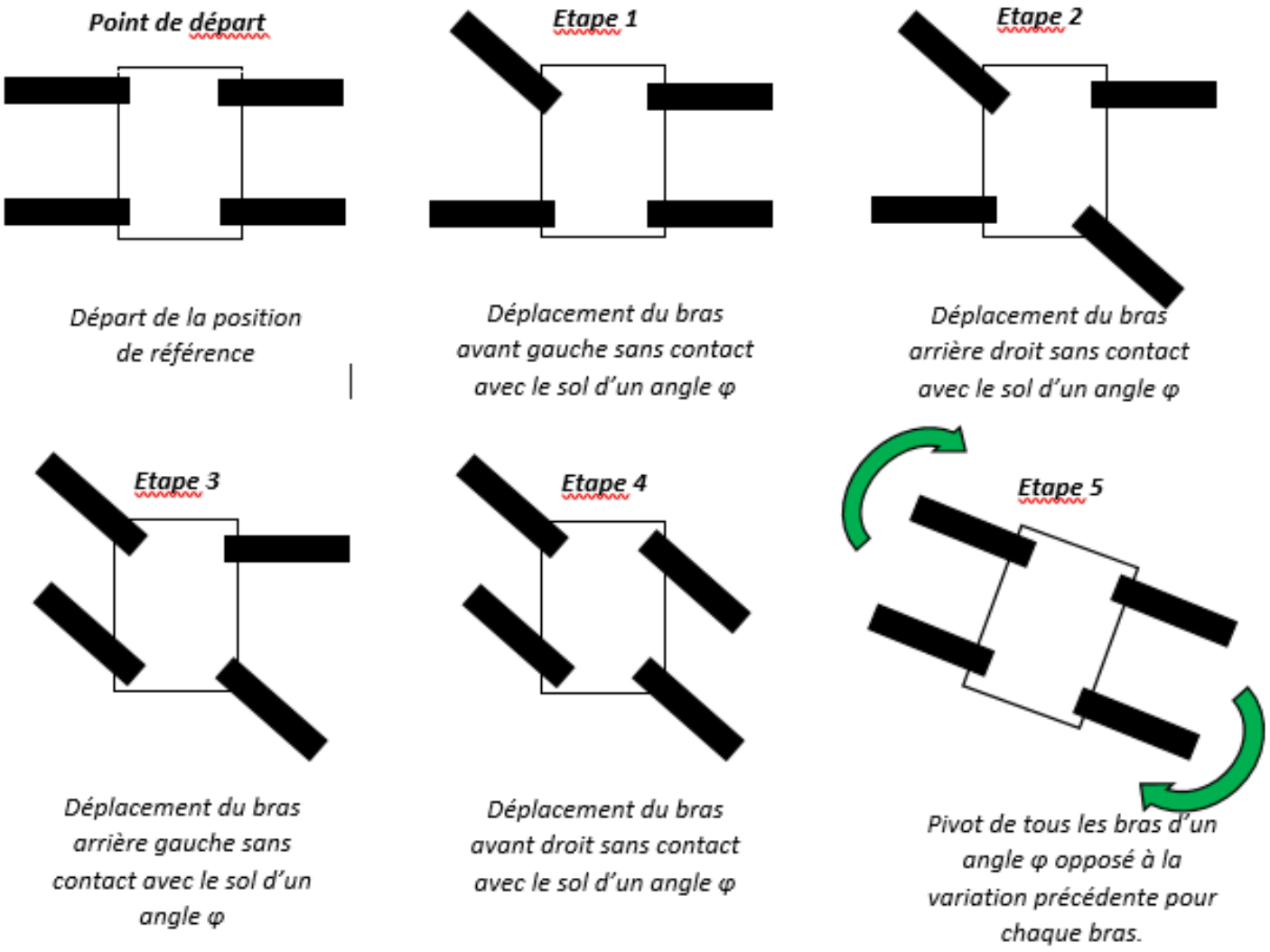


Figure II.7: Succession des étapes pour effectuer un pivot à droite.

Remarque : Dans l'étape finale du pivot, on aligne les pattes du robot pour qu'elles fassent à nouveau un angle de 90° avec le corps. Cependant il est possible que le fait que les servos bougeant tous de la même manière, il est possible que l'angle commandé ne corresponde pas à l'alignement final. En effet une partie de la force de rotation peut aussi être subie par les pattes étant donné qu'il y a assez peu de force qui maintient les pattes.

Algorithme et programmation :

Pour réaliser une Modulation de Largeur d'Impulsion de fréquence 50 Hz (période de 20 ms), Seul le timer1 permet de réaliser cela en matériel pur, c'est à dire sans code spécifique s'exécutant pour le réaliser et donc sans interruption.

Pour cela on va faire fonctionner le timer en fast PWM qui va non pas aller jusqu'à la valeur 65535 mais jusqu'à la valeur chargée dans le registre **ICR**. Afin de connaitre la valeur nécessaire pour obtenir un signal de 50 Hz, on utilise la formule suivante :

$$ICR = \frac{F_{CLK}}{F_{SIG} \times Prescaler}$$

Nous avons choisi un prescaler de 8, on retrouve par conséquent la valeur de ICR : 40000.

On s'arrange ensuite pour être en mode « non-inverting », on retrouve les chronogrammes suivants pour une valeur constante de OCR1B (par exemple) :

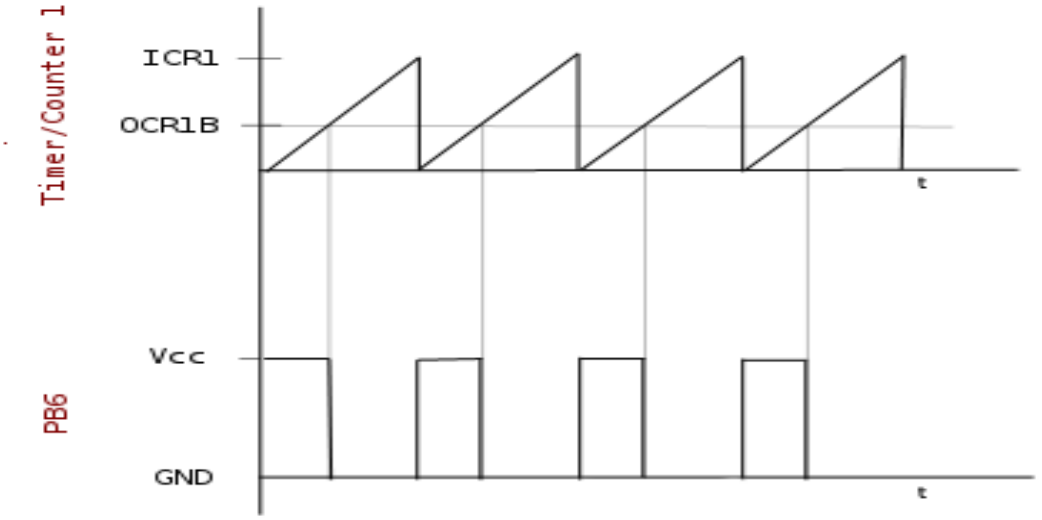


Figure III.1: Génération fast PWM avec une fréquence fixée grâce au registre ICR.

Le robot a 4 bras et 4 coudes : chaque bras et coude est commandé par un servo-moteur lui-même piloté par un PWM c'est pour cela que l'on utilise 8 PWM.

On a utilisé une fonction qui s'appelle **config_FASTpwm** qui configure les 8 Pwm .

Pour représenter l'angle qu'on veut pour se déplacer on a utilisé 8 fonctions à l'intérieur desquelles on calcul l'angle x grâce au registre de comparaison OCRXn. En effet l'angle varie en fonction du rapport cyclique, et ce dernier change en fonction du registre OCRXn. On a vu que pour le servo moteur, un T_{ON} de 1ms correspond à un angle de 0°, et un T_{ON} de 2ms correspond à un angle de 180°. On peut considérer l'écart entre les deux valeurs, c'est-à-dire qu'une amplitude de 180° correspond à une variation de T_{ON} de 1ms. Cela implique un « offset » de 1ms sur la valeur de T_{ON}. Par un produit en croix sur les intervalles précédemment cités on retrouve :

$$T_{ON}(ms) = \frac{x°}{180°} \times 1ms + 1ms$$

On peut ensuite retrouver la valeur du registre OCRXn à charger pour avoir un T_{ON} voulu : dans le mode non inversé avec ICR=40000, on a un rapport cyclique de 1 pour un OCRXn de 40000 et un rapport cyclique de 0 pour un OCRXn de 0. Pas d'offset, on retrouve directement :

$$OCRXn = \frac{40000 \times T_{ON}(sec)}{20.10^{-3}}$$

En remplaçant on a le chargement du registre OCRXn en fonction de l'angle désiré :

$$OCRXn = 2000 \times (\frac{x}{180} + 1)$$

Ainsi il suffit de rentrer la valeur de l'angle en argument de ces fonctions qui permettent chacune d'envoyer un signal PWM de rapport cyclique variable à un servo-moteur en particulier.

- **void start_coude_gauche_avant(x)**
- **void start_coude_gauche_arriere(x)**
- **void start_coude_droit_avant(x)**
- **void start_coude_droit_arriere(x)**
- **void start_bras_gauche_avant(x)**
- **void start_bras_gauche_arriere(x)**
- **void start_bras_droit_avant(x)**
- **Void start_bras_droit_arriere(x)**

Le robot a besoin d'une position initial pour avoir une position qui s'intercale entre les déplacements donc on a utilisé une fonction qui s'appelle **void position_initial()** qui fait appel aux 8 fonctions qui calculent l'angle. Pour faire marcher le robot on utilise la fonction : **void marche_avant ().**

On appelle les fonctions dans un certain ordre et nous programmons des délais avec le timer 0 entre chaque fonction pour être sûr que les mouvements et les déplacements se succèdent correctement. Idem pour la fonction **void marche_arriere ()** et aussi pour la **fonction void pivot().**

En ce qui concerne la programmation du capteur, on procède en deux temps. Le premier concerne un échantillonnage de la mesure. Pour cela on utilise le dernier timer qui nous reste (le

timer 5) pour programmer une interruption toutes les 3 secondes. C'est dans cette interruption que l'on va faire la mesure.

La mesure consiste en l'envoi d'un trigger et en la mesure du temps passé par la broche Echo à l'état haut. Pour mesurer ce laps de temps, on utilise un ensemble de boucle while. La première vérifie l'état de la broche Echo : on ne fait rien tant que la broche n'est pas à 1.
Dès que la broche passe à 1, on initialise le timer 2 à 0 et on entre dans une autre boucle while : tant que Echo vaut 1, on sauvegarde la valeur du registre TCNT2 dans une variable initialisée en global.

On test aussi à chaque passage de la boucle si TOV2=1. Dans ce cas, on a dépassé la distance maximale mesurable par le timer 2 et l'on peut considérer que l'obstacle se trouve à une distance correspondant à une valeur numérique maximale. Il faut penser à remettre TOV2 à 0 et à sortir du while (ou tout autre méthode évitant d'écraser la valeur de la variable avec une nouvelle grandeur...).

Une fois que Echo est revenu à 0, on calcule la distance correspondant (grâce à la fréquence d'incrémentation du registre 2 et à l'équation vu dans la partie I), on peut ensuite faire un test pour pouvoir pivoter si la valeur est inférieure à une distance de 20cm. Dans ce cas on appellera la fonction pivot et on recommencera.

Pour aller plus loin

Ci-dessous nous avons listé quelques améliorations qui pourraient être intéressante pour le robot est auxquelles nous avons pensé au cours de la réalisation de notre projet. Certaines sont plus réalisable que d'autres, cependant elles ne sont pas toutes applicables en même temps, il faudrait peut-être pour certaines revoir l'ensemble du fonctionnement. Ce problème pourrait être réglé en utilisant notamment une carte Arduino nano qui gérerait le capteur à ultrasons et ses améliorations puis communiquerait à l'Arduino MEGA les distances mesurées via la connexion USART.

Ajout d'un moteur

Ajouter un troisième moteur sur chaque jambe permettrait de gagner en résolution et d'avoir un robot avec des déplacements plus fluides. A l'image d'une araignée, un servo-moteur supplémentaire permettrait de soulever plus de poids et de gérer un peu mieux l'orientation du corps dans l'espace (cf gyroscope).

Revoir la structure 3D

N'ayant pas réalisé nous même la structure 3D, cette dernière n'est pas adaptée au matériel que nous avons ajouté sur le robot. Finalement il pourrait être très intéressant de l'améliorer, au même titre que l'amélioration de l'algorithme de déplacement. En effet l'approximation d'une ligne droite n'est pas exacte (dans le calcul des angles de rotation), ce qui implique une force vers le haut qui va à l'encontre des servos moteurs qui cherchent à maintenir une position permettant le contact avec le sol. En anticipant ce léger sursaut on gagnerait en fluidité. Le fait d'avoir un moteur supplémentaire réglerait aussi ce problème.

Gyroscope

Une extension possible serait d'intégrer un gyroscope qui permettrait entre autres de donner le plan horizontal au microcontrôleur pour que celui-ci calibre automatiquement les servos de manière à être plus ou moins parallèle à la surface d'évolution à condition que cela soit permis par l'amplitude des servos. Ce dernier permettrait aussi de très nombreuses améliorations (ajustement de l'équilibre entre autres).

Mappage de l'espace

L'objectif est ici de monter le capteur à ultrason sur un servo-moteur qui balaierai l'espace devant lui. Ainsi l'hypothèse est que l'on puisse donner suffisamment précisément la position d'un obstacle à une cinquantaine de centimètre pour pouvoir l'éviter de manière plus fluide, sans d'ailleurs forcément s'arrêter (il suffirait d'avoir des angles plus importants à droite dans le déplacement si l'on veut tourner à gauche).

Bibliographie

Thème	***Référence***	***Détails***
Pièces 3D	*https://www.thingiverse.com/thing:1912377*	*Utilisation des fichiers .stl pour impression du robot*
Servo-moteur	*https://eskimon.fr/tuto-arduino-602-un-moteur-qui-a-de-la-t%C3%AAte-le-servomoteur*	*Fonctionnement du servo moteur*
Capteur ultrasons	*https://www.gotronic.fr/pj2-hc-sr04-utilisation-avec-picaxe-1343.pdf*	*Fonctionnement et protocole de communication*
Arduino et code	*Cours UCBL/Master EI²/ESE – EASE et fiche technique ATmega 2560*	*Fonctionnement Atmega 2560 et branchements arduino + programmation de registres*

Remerciements

Nous tenions à remercier les personnes nous ayant autorisés proposer un projet dans le cadre de l'enseignement TERI et ainsi à faire entrer la phase de prototypage de ce robot dans la formation du master EI², à savoir monsieur Varray, tuteur de notre groupe de projet, mesdames Cavassila et Deman, coresponsables du parcours EI² et monsieur Jaffard, responsable de l'UE TERI.

Remerciements particuliers au fablab associatif AsTech, structure qui nous a permis d'utiliser les imprimantes 3D et le matériel électronique dont nous avions besoin, et ce dans un espace de travail accessible.