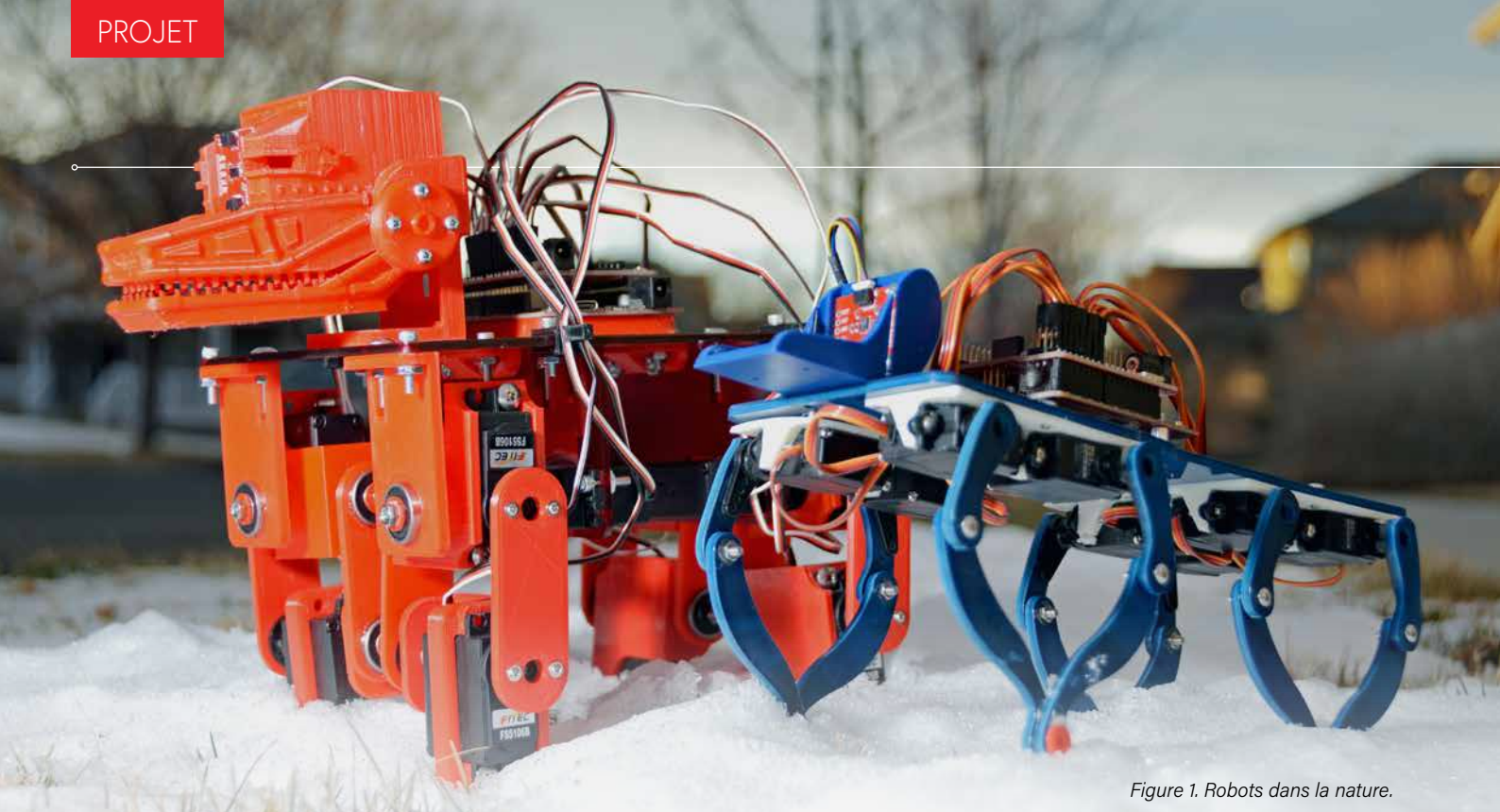

Figure 1. Robots dans la nature.

Fabriquer soi-même des robots à quatre pattes

Rob Reynolds (États-Unis)

Que diriez-vous de construire un robot quadrupède ? Oui, bien sûr ! Je propose de construire deux plates-formes extensibles, avec de multiples points de fixation pour le montage de servos, l'ajout de capteurs et la mise en place de microcontrôleurs.

Je ne peux pas parler au nom de tous les électroniciens, mais on peut dire que dès l'enfance, lorsque la plupart d'entre nous imaginaient des robots, nous pensions d'abord aux humanoïdes bipèdes dotés de compétences interactives que nous appellerions aujourd'hui l'IA, sans toutefois la fluidité des mouvements des humains. Il s'agissait de rêves ou de créatures imaginées pour le cinéma, animées par des acteurs en costume métallique, des marionnettistes cachés, ou des génies comme Ray Harryhausen et son hibou robot Bubo du *Choc des Titans*. Me voici adulte (chronologiquement du moins), et bien déçu de constater que ce que nous appelons robots domestiques sont ici un aspirateur ou là un robot pâtissier, c'est-à-dire un simple mixeur sur un socle. Si c'est ça la robotique, il ne me reste plus qu'une chose à faire : créer mes propres robots (**fig. 1**).

Conception du projet

L'un des aspects que je préfère dans la création de quelque chose comme un robot - ou, de n'importe quel autre projet d'ailleurs – c'est le point de départ. Tout est encore possible. Dans la comédie musicale de Stephen Sondheim « Sunday in the Park with George », le personnage de George dit en se souvenant de son grand-père, le peintre George Seurat : « Blanc : une page ou une toile vierge. Son préféré - tant de possibilités ». À ce stade, les possibilités sont illimitées, peut-être bêtement parce que je n'en sais pas assez – et espérons que je n'en saurai jamais assez – pour savoir ce qui est impossible. Dans mon esprit, tout est donc possible. À quoi mon robot devrait-il ressembler ? Pour l'effet « wow », il pourrait s'agir d'un robot humanoïde bipède longiligne, comme ceux du film « *I, Robot* ». En pratique, si je veux un robot qui puisse facilement se déplacer chez moi et interagir avec moi ou avec son environnement, il serait plus simple de le monter sur quatre roues. Ce qui nous ramènerait en gros au stade du « robot-aspirateur ». Il faut chercher un juste milieu, quelque chose d'un peu pratique, mais d'assez impressionnant pour attirer l'attention. Je voudrais aussi pouvoir le faire évoluer. J'ai décidé qu'il serait donc quadrupède.

Pour m'inspirer, il y a les travaux de *Boston Dynamics* sur les quadrupèdes, de Big Dog en 2005, massif et un peu maladroit, qui crapahute en terrain accidenté, à Spot en 2018, élégant et agile, qui ouvre les portes et

danse sur *Uptown Funk*. C'est un bel exemple de progression et d'itération. Pour mon quadrupède, je voulais une plate-forme de base avec des possibilités d'extensions, voire d'échange de modules. J'ai envisagé plusieurs configurations, et finalement opté pour deux conceptions différentes. Après coup, il est évident que l'une a été inspirée par Big Dog, et l'autre par Spot, avec chacune ses propres capacités, mais extensibles l'une et l'autre.

J'ai commencé par le robot inspiré de Spot, plus petit et plus élégant. Avec en tête la mécanique que je voulais utiliser, j'ai fait une série de lignes droites sur mesure, qui a donné des pièces sans intérêt. Ensuite j'ai réfléchi un peu plus au Spot de Boston Dynamics, et j'ai réalisé que dans ce cas, je pouvais laisser la forme suivre la fonction. Je pouvais me permettre quelques courbes, pour améliorer un peu l'apparence de ce robot. En fait, j'ignore s'il est plus beau, mais il est un peu moins anguleux, et aussi moins stable. Disons que j'ai réussi à le rendre moins disgracieux.

J'ai ensuite commencé à travailler sur le deuxième robot, le robot Big Dog, d'inspiration plus industrielle. Après plusieurs heures de travail de conception, je me suis demandé à quoi bon réinventer la roue. J'ai cherché des modèles utilisables, du moins comme point de départ. J'ai trouvé une superbe réalisation de *Technovation* décrite sur *Instructables* [1]. Les deux plates-formes ont de multiples points de fixation pour le montage de servos, l'ajout de capteurs et la mise en place du microcontrôleur. Je prévois d'utiliser le Redboard Artemis de SparkFun, qui a l'empreinte familière de l'Arduino Uno. Le Redboard Artemis offre plus de mémoire et de vitesse que l'Uno, plus le BLE embarqué, et en utilisant ce schéma de perçage, je peux facilement passer au SparkFun Artemis ATP si j'ai besoin de plus de broches, car il partage ce même schéma de perçage. J'utiliserai aussi le *shield* de pilotage de moteur sans fil de SparkFun.

Les pattes du robot quadrupède

Pour les pattes, j'ai fait deux choix différents pour les deux robots. Pour le plus petit - appelons le Bluesette - j'ai utilisé une liaison à cinq barres, qui n'est peut-être pas idéale pour un quadrupède, surtout avec des servomoteurs. Cela vaut la peine de l'expérimenter juste pour apprendre et se frotter au mouvement et à la cinématique inversée. La liaison à cinq barres est un mécanisme à deux degrés de liberté dans lequel les cinq barres sont reliées en boucle (**fig. 2**).

Ce mécanisme contrôle les coordonnées x et y de l'articulation D, appelée point final ou *effecteur* final, (ou dans notre cas, le pied du robot) en ajustant simultanément les angles d'entrée thêta 1 et thêta 2, contrôlant ainsi l'angle des barres B2 et B5. En déplaçant l'effecteur final de chaque patte le long d'une trajectoire elliptique, nous pouvons faire marcher le robot en avant et en arrière, ajuster sa hauteur, et même tourner en rond.

Comme je l'ai dit, ce n'est peut-être pas la meilleure locomotion avec des servomoteurs pour un quadrupède, mais si vous passez à un ensemble de huit moteurs sans balais à haute performance, cette configuration peut devenir très efficace. Le projet Stanford Doggo en est un bon exemple [2] : grâce à la vitesse, à la puissance et à la commande des moteurs sans balais, ce petit quadrupède peut sauter à plus d'un mètre de hauteur et même faire des sauts périlleux arrière (**fig. 3**).

Mon robot plus grand et plus gros, appelé Big Red, utilise le mécanisme le plus courant pour les pattes de quadrupèdes, connu sous le nom de manipulateur série. Chaque articulation a son propre moteur, de la base à l'actionneur final. Ces articulations sont souvent celles de l'épaule, du coude et du poignet, en particulier lors de la création de tout type de structure de bras anthropomorphe, comme un automate de placement mono-bras, par exemple sur une chaîne de montage automobile. En

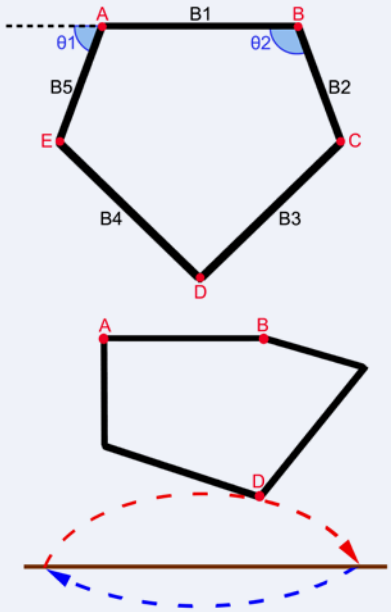

Figure 2. Mécanismes à cinq barres.

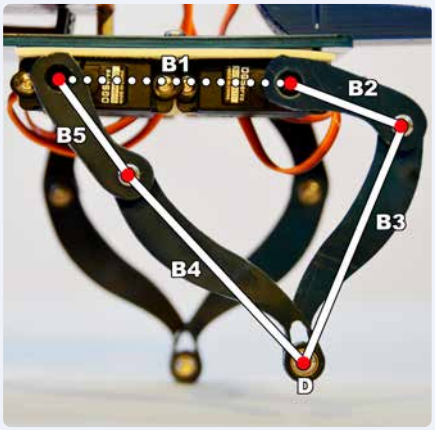

Figure 3. Les deux systèmes utilisent deux servos pour contrôler la position des pattes, mais de manière très différente.

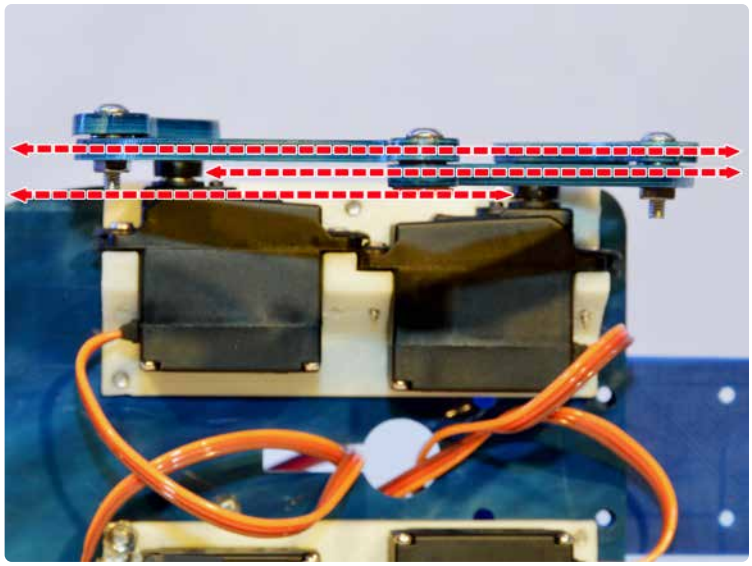

Figure 4. En décalant les supports de servo de l'épaisseur du matériau de la patte, tout reste aligné.

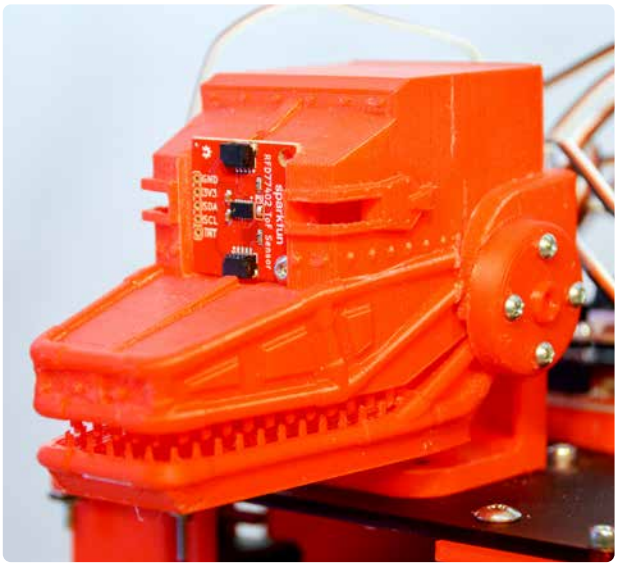

Figure 5. Tête de chien.

gardant les bras courts pour cette première version de Big Red, je peux disposer d'une force et d'une stabilité impossible à atteindre avec les pattes de Bluesette.

Pour la construction et les matériaux, j'ai utilisé une combinaison de pièces en acrylique de 3 mm découpées au laser et de pièces en PLA imprimées en 3D. De nombreux ateliers partagés disposent d'imprimantes 3D, et certains proposent aussi des découpeuses laser. Si vous n'avez pas accès à la découpe laser, il reste le contreplaqué de 3 mm, facile à travailler avec des outils manuels. Les plus importantes sont les pièces imprimées en 3D, en particulier les supports de servo. Même sans imprimante 3D, vous pouvez fixer vos moteurs solidement avec des serre-câbles ou de la colle thermofusible, du moins pour le robot avec pattes à cinq barres. Soyez créatif. Après tout, ne sommes-nous pas des bidouilleurs ici ?

Même s'ils ne sont pas complètement nécessaires, j'utilise des roulements pour toutes les articulations des deux robots. Les mouvements sont beaucoup plus doux et cela réduit les frottements au niveau des articulations et donc la consommation de courant des servomoteurs. Une autre caractéristique importante de la conception de Bluesette est le décalage de 3 mm (ou de l'épaisseur du matériau utilisé pour les pattes) des moteurs pas à pas. Si les moteurs pas à pas sont placés sur un même plan (**fig. 4**), les réunir en boucle entraînera une torsion latérale au niveau de l'une des articulations, ce qui nécessitera à nouveau plus de courant pour entraîner les servomoteurs.

> *La base de mes quadrupèdes, c'est une plate-forme avec des possibilités d'extensions et même d'échange de modules.*
>
> Rob Reynolds

Il me fallait encore une sorte de support frontal pour les capteurs. J'aurais pu concevoir vite fait bien fait un support de montage rudimentaire, mais j'y ai vu une autre occasion de me montrer créatif. Pour Bluesette, j'ai juste assemblé quelques formes primaires, chanfreiné les arêtes, ajouté quelques trous de montage et imprimé le tout. Big Red, en revanche, a reçu un peu plus de mon temps et de mon attention. Je me suis souvenu que dans le film « L'île aux chiens » il y avait de très beaux chiens robots, donc en utilisant des images fixes du film, j'ai fait jouer mes muscles de conception 3D et j'ai recréé les têtes des chiens robots, avec des ajustements pour pouvoir placer la plupart de nos cartes de capteurs Qwiic (**fig. 5**). Je craignais que cela n'alourdisse l'avant de ce robot, mais je me suis dit que cela pourrait être compensé avec une grosse batterie à l'arrière.

Pour l'assemblage des deux robots, j'ai utilisé des vis M3 de différentes longueurs (sauf pour les entretoises, qui ont des 4x40). J'ai imprimé en 3D des bagues pour les roulements de Big Red et les articulations des poignets de Bluesette.

Une note sur le codage et les tests. Si vous concevez et construisez quelque chose qui peut bouger, il va bouger, mais généralement au moment où vous vous y attendez le moins. J'ai appris cette leçon pour la première fois alors que je travaillais sur une petite voiture autonome. J'avais ajouté du code à mon croquis et le téléchargeais sur mon véhicule. Dès le téléchargement terminé, ma petite voiture est partie à toute vitesse, s'envolant de mon établi et se fracassant contre le mur du fond. Nathan, le fondateur de SparkFun, a eu le même problème lors de l'une de ses constructions, mais il travaillait sur un véhicule autonome suffisamment grand pour y monter. Quand il a décollé, il a arraché son ordinateur portable de son bureau. Il faut donc toujours s'assurer que les pièces qui font bouger votre création ne touchent pas le sol. Si vous travaillez sur une voiture, posez-la sur une boîte plus petite que l'empattement du véhicule et plus haute que sa garde au sol. Pour ces robots, je disposais de quelques petits profilés en aluminium rainurés en T MicroRax, que j'ai utilisés avec une pièce sur mesure vite faite en 3D pour garder les pieds décollés du sol (**fig. 6**). Dans l'UE ou au Royaume-Uni, vous trouverez des accessoires similaires chez MakerBeam.

Accessoires

Vous trouverez certains des accessoires mentionnés dans cet article chez SparkFun et Elektor !

- **Elektor MIT App Inventor Bundle**
  www.elektor.fr/elektor-mit-app-inventor-bundle
- **SparkFun RedBoard Artemis**
  www.elektormagazine.fr/esfe-en-qrobot1
- **SparkFun Wireless Motor Driver Shield**
  www.elektormagazine.fr/esfe-en-qrobot2

Le code

Après l'assemblage, commençons l'écriture du code (**listage 1**). Notre principale ressource sera la bibliothèque Servo. À mesure que nous avancerons et ajouterons des capacités supplémentaires, d'autres bibliothèques entreront en jeu. Pour commencer, il faut juste que notre croquis crée et relie entre eux tous les objets servo. Un moyen simple et rapide est de partir de quelque chose comme le croquis **Servo Sweep** et de faire quelques ajustements mineurs. Pour rester simple, à la fois pour aujourd'hui et pour plus tard, on peut énumérer les servos dans une boucle `for`. Si vous prévoyez de poursuivre avec ces robots ou d'autres de conception similaire, c'est un excellent point de départ, car il suffira d'ajuster le nombre de boucles au nombre de servos à créer et relier. Nous définirons ensuite une position de départ pour chaque servo et lui ferons passer un court test de balayage, juste pour nous assurer qu'ils communiquent entre eux et sont correctement alimentés. Lorsque je travaille sur un code dont je sais qu'il continuera à être développé, et peut-être même à avoir de multiples itérations, je trouve qu'il vaut mieux le décomposer en fonctions plutôt que de tout avoir dans la `void:loop()`. Cela facilite les choses pour trouver, modifier, et copier/coller dans d'autres itérations du croquis ou dans des croquis complètement différents qui ont juste besoin d'un petit bout d'un code existant. Sans doute parce que je suis assez vieux pour me souvenir de la programmation en BASIC et de l'utilisation de sous-programmes.

Vous voici avec un point de départ. Les prochaines étapes, du moins pour moi, iront par ordre croissant de difficulté. Abaisser la hauteur du robot jusqu'à peu près la moitié de sa hauteur totale est assez simple et peut donner l'impression qu'il se tapit. La prochaine étape serait logiquement une paire de fonctions de marche, à la fois en avant et en arrière. Puis tourner des deux côtés. Ensuite, on peut passer à la commande par BLE ou au mouvement autonome, selon l'objectif final. Connecter votre ordinateur portable ou tablette via Bluetooth et envoyer des commandes série à un seul caractère serait un moyen simple et rapide pour commencer à commander votre robot à distance, ou bien le coupler avec une carte micro:bit comme télécommande portable. Pour commander votre robot depuis un téléphone Android, vous pourriez utiliser l'appli Inventor du MIT (https://appinventor.mit.edu/) et créer une interface sur un appareil que vous avez toujours avec vous. Et si vous hésitez, vous pouvez toujours écrire une fonction pour chaque commande, puis parquer celles que vous décidez de ne pas utiliser. Des capteurs de proximité équipent couramment les robots, ils constitueront l'étape suivante la plus pertinente. Peut-être souhaitez-vous ajouter un capteur d'environnement pour des données comme la température ou l'humidité. La dernière étape pour cette fois sera d'ajouter un peu d'apprentissage machine ou d'intelligence artificielle de base. Avec TensorFlow, il sera assez facile d'entraîner votre robot à répondre à des commandes vocales simples : stop, marche, assis, attaque, tout ce que vous voulez !

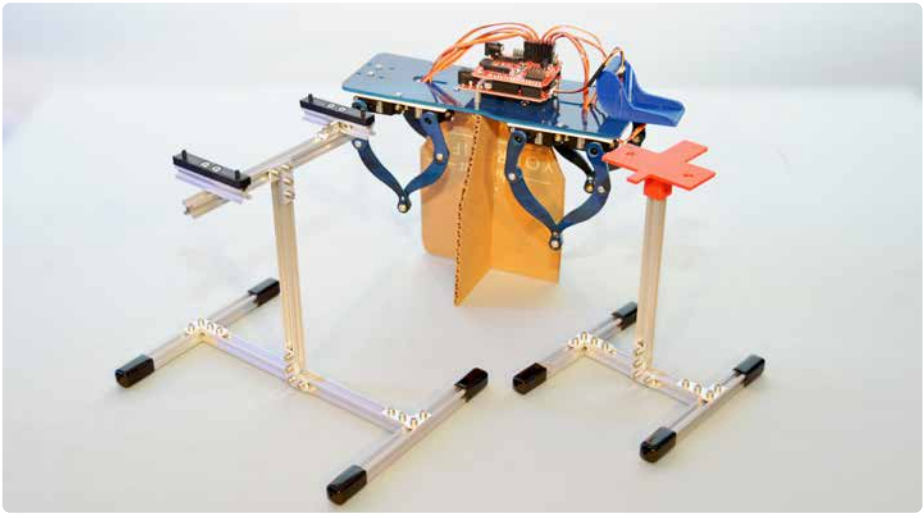

Figure 6. J'ai fabriqué deux supports en alu, mais du carton rigide aurait aussi fait l'affaire.

À vous maintenant de construire le vôtre !

Comme toute étude en cours, ces deux robots vont évoluer et ils auront encore bien changé quand paraîtra cet article. Je dépose tout sur Github (https://github.com/ThingsRobMade/QuadrupedRobots) et poursuivrai la mise à jour au fur et à mesure de l'ajout de fonctions à ces deux quadrupèdes. Je vous encourage à construire les vôtres sans attendre ce que je fais. Lancez-vous et commencez à programmer. L'un des avantages de la communauté open-source est la présence d'une intelligence collective : à chaque obstacle,il se trouvera toujours quelqu'un pour émettre une idée brillante qui à son tour inspirera de nouvelles méthodes ou de nouvelles directions. D'ici là, ne cessez jamais ni de rêver ni d'inventer. Bonne bidouille !

(VF : Denis Lafourcade 200712-02)

LIENS

[1] Technovation, "3D Printed Arduino Powered Quadruped Robot," Instructables, 2020. : https://www.instructables.com/3D-Printed-Arduino-Powered-Quadruped-Robot/

[2] Nate711, "StanfordDoggoProject," GitHub, 2019. : https://github.com/Nate711/StanfordDoggoProject

Listage 1. Place les articulations du robot dans une position neutre et les fait légèrement basculer d'avant en arrière.

```
/*
* Schéma de configuration d'un robot quadrupède
* Rob Reynolds
* SparkFun Electronics
*
* Ce simple croquis fixe les articulations d'un
* robot quadrupède à une position neutre,
* puis les balance légèrement d'avant en arrière
* simplement pour tester le contrôle et le mouvement.
* Actuellement configuré pour 8 degrés de liberté (DoF), mais peut
* facilement être ajusté en modifiant la dimension de
* Servo leg[*] pour correspondre au nombre de servos.
*/

#inclure <Servo.h>
Servo leg [8] ; // créer des objets servo pour commander un servo
int pos = 90 ; // variable pour mémoriser la position du servo

void setup() {
  delay(1000) ;
  for (int l = 0 ; l < 8 ; l++){
    leg[l].attach(l + 2) ; // lier les servos en séquence en commençant par la broche 2
    delay(100) ;
  }
  delay(1000) ;
  for (int l = 0 ; l < 8 ; l++){
    leg[l].write(pos) ; // régler tous les servos à 90
    delay(100) ;
  }
}

void loop() {
  allServoTest() ; // Test initial ou mouvement général des servos
  while(1){
  }
}

void allServoTest(){
  // Test de mouvement initial pour tous les servos. Tous les servos sont réglés à 90°.
  // puis balayage avant et arrière avant de revenir à 90° de nouveau
   for (int i = 0 ; i < 8 ; i++){
    for (pos = 90 ; pos <= 135 ; pos += 1) {
      leg[i].write(pos) ; // dire au servo d'aller à la position de la variable 'pos'.
      delay(10) ; // attendre 10ms que le servo atteigne la position
    }
    for (pos = 135 ; pos >= 45 ; pos -= 1) {
      leg[i].write(pos) ; // dire au servo d'aller à la position dans la variable 'pos'.
      delay(10) ; // attendre 10ms que le servo atteigne la position
    }
    for (pos = 45 ; pos <= 90 ; pos += 1) {
      leg[i].write(pos) ; // dire au servo d'aller à la position dans la variable 'pos'.
      delay(10) ; // attendre 10ms que le servo atteigne la position
    }
  }
}
```