Compte rendu : Conception et réalisation d'un robot bipède

Alec Passolunghi et Noé Massé

26 janvier 2026

Table des matières

1 Introduction

Projet de robotique visant à concevoir et réaliser un robot bipède capable de marcher de manière autonome. Ce projet s'inscrit dans le cadre d'un cours de robotique avancée et a pour objectifs principaux la compréhension des principes mécaniques et électroniques nécessaires à la construction d'un tel robot. Pour ce faire, l'on se donne alors le cahier des charges suivant :

— Déplacement autonome sur une surface plane.
— Stabilité lors de la marche.
— Sortir d'un labyrinthe simple.

2 Mise en place

2.1 Description du matériel utilisé

Pour l'assemblage du robot, nous avons alors utilisé les différents composants suivants :

— 4 servomoteurs 0 à 180 degrés (FS5103B)
— 1 carte Arduino Uno
— 1 Arduino Sensor Shield
— 2 capteurs infrarouges
— 1 batterie rechargeable
— Matériaux pour la structure (métal, patafix, visserie, roulements)
— Câbles de connexion
— Interrupteur

2.2 Cinématique

Dans cette section, nous définissons les mouvements que le robot devra effectuer afin d'être capable de se mouvoir et sortir du labyrinthe. Pour ce faire, le robot utilisera deux jambes, chacune possédant deux degrés de liberté, permettant des mouvements de rotation du bassin et également rotation des pieds.

Concernant les mouvements, le robot pourra effectuer les actions suivantes :

— Avancer : en alternant les mouvements des jambes pour simuler la marche.
— Reculer : en inversant les mouvements des jambes.
— Tourner : en ajustant la position des jambes pour changer de direction.
— S'arrêter : en positionnant les jambes de manière stable pour maintenir l'équilibre.

Ce robot bipède sera constitué de deux jambes articulées par deux servomoteurs par jambe. La structure sera réalisée en métal pour assurer la robustesse, avec des dimensions adaptées et expérimentées pour un équilibre optimal.

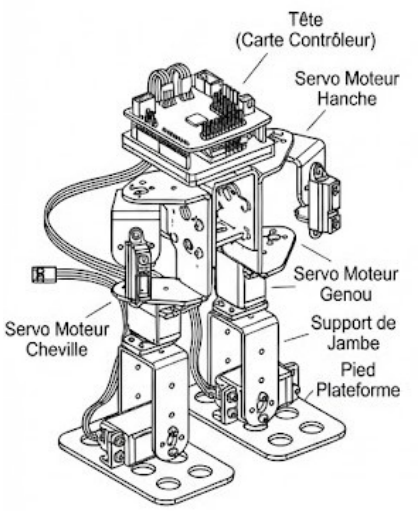

FIGURE 1 – Schéma du robot bipède

3 Développement

3.1 Premier montage

La première version du robot rencontra plusieurs difficultés. Dans un premier temps, il s'avéra qu'il était trop large pour pouvoir se pencher et donc marcher correctement. On peut imputer cela à la structure et le manque de couple des servomoteurs utilisés.

3.2 Premier code

Le premier code mis en place pour le robot était un code inopérant qui ne permettait pas au robot de se mouvoir. Cependant, il fut la base sur laquelle les itérations suivantes sont basées.

Voir Annexes.

3.3 Deuxième configuration

Dans cette itération, nous avons réduit la largeur du robot afin de lui permettre de se pencher, c'est-à-dire de déplacer son centre de gravité et donc de se mouvoir.

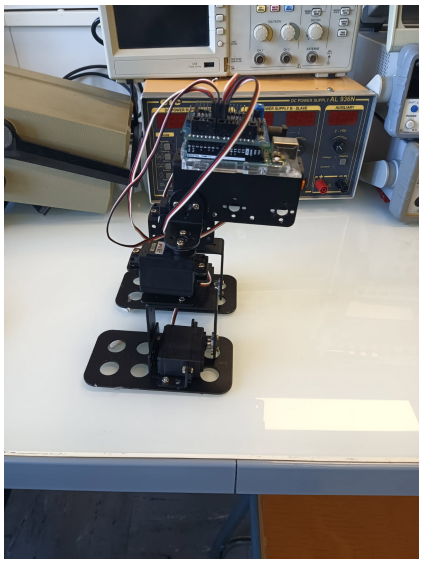

FIGURE 2 – Deuxième configuration du robot bipède

3.4 Arrivée du parallélisme

Une grande avancée fut l'implémentation du code en parallélisme, permettant ainsi au robot d'effectuer plusieurs actions en même temps et donc de marcher de manière plus fluide. Néanmoins, il faut noter qu'il ne s'agit pas d'un vrai code en parallélisme mais plutôt d'une simulation de celui-ci. En effet, dans le code, les instructions sont exécutées les unes après les autres, mais grâce à l'utilisation de délais très courts, voire nuls, on obtient un effet de simultanéité.

3.5 Arrivée des capteurs

Enfin, pour permettre au robot de sortir du labyrinthe, nous avons ajouté deux capteurs infrarouges à l'avant du robot. Ceux-ci permettent de détecter les obstacles et ainsi d'adapter la trajectoire du robot en conséquence. Pour ce faire, nous avons implémenté différentes fonctions qui permettront de rendre le code plus lisible et modulaire. Ainsi, nous avons une fonction pour avancer, une pour reculer, une pour tourner à gauche et une pour tourner à droite.

Ensuite, dans la boucle principale, nous lisons les valeurs des capteurs et, en fonction de celles-ci, nous appelons les fonctions adéquates pour permettre au robot de se déplacer dans le labyrinthe. On note aussi l'ajout d'un interrupteur pour pouvoir allumer et éteindre le robot facilement, ainsi qu'une fonction d'attente au démarrage pour laisser le temps à l'utilisateur de mettre le robot en position.

3.6 Version finale

On peut voir ci-dessous le robot dans sa version finale.

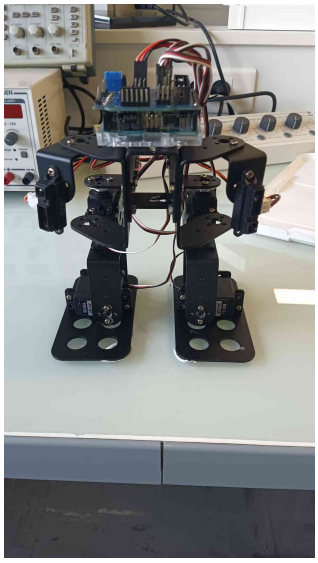

FIGURE 3 – Robot bipède final

4 Résultats

On peut observer que le robot est capable de se déplacer de manière autonome et de sortir d'un labyrinthe simple. Cependant, des améliorations sont encore possibles, notamment en ce qui concerne la fluidité de la marche et la réactivité aux obstacles.

4.1 Performance du robot

Pour ce qui est de la fluidité de la marche, nous avons implémenté des mouvements plus doux en ajustant les angles des servomoteurs pas à pas, avec des boucles for et des délais très courts. Cela a permis d'améliorer la stabilité du robot lors de la marche.

4.2 Problèmes rencontrés

Tout d'abord, un des principaux problèmes rencontrés fut la stabilité du robot. En effet, le manque de fluidité entraînait des fortes contraintes desserrant les vis et rendant le robot instable. Pour résoudre ce problème, nous avons renforcé la structure du robot et ajusté les mouvements pour qu'ils soient plus doux.

Un autre problème fut la détection des obstacles par les capteurs infrarouges. Nous avons dû ajuster la sensibilité des capteurs et le code de traitement des données pour améliorer la réactivité du robot aux obstacles.

Enfin, il y eut la sélection des composants (servomoteurs et capteurs) qui, pour les premiers, se fit à travers le fait que leur angle de 90° était en réalité plus proche de 80°. Et pour les seconds, le nombre de capteurs n'était pas calibré de manière similaire. Ce problème fut résolu avec la dite sélection.

5 Conclusion et perspectives

Nous avons réussi à concevoir et réaliser un robot bipède capable de marcher de manière autonome et de sortir d'un labyrinthe simple.
Le projet nous a permis de comprendre les principes mécaniques et électroniques nécessaires à la construction d'un robot bipède simple. Il reste cependant des améliorations possibles, notamment en ce qui concerne la fluidité de la marche et la réactivité aux obstacles. L'on pourrait alors envisager l'utilisation de plus de capteurs pour une meilleure perception de l'environnement, ainsi que l'implémentation d'algorithmes de contrôle plus avancés pour améliorer la stabilité et la fluidité de la marche. Ou encore l'utilisation de matrices pour le contrôle des servomoteurs pour pouvoir gérer plus facilement les mouvements complexes.

6 Annexes

6.1 Code source

Listing 1 – Code source Arduino pour le robot bipède

```
#include <Arduino.h>
#include <Servo.h>

Servo servogauchepied, servodroitepied,
      servogauchebassin, servodroitebassin;

void setup()
{
    Serial.begin(9600);
    Serial.flush();
    servodroitepied.attach(4);
    servodroitebassin.attach(5);
    servogauchepied.attach(6);
    servogauchebassin.attach(7);
}

int a = 0; // compteur boucle
int distD;
int distG; // distance obstacle
int analogPinG = 1;
int analogPinD = 0;
bool mar; // mode marche(true) ou course(false)
int vitesse_etape = 0;
int vitesse_equilibre = 75;  // 100ms pour que le robot se stabilise
int vitesse_redescente = 17; // 30ms pour descendre tranquillement
int vitesse_smooth = 8 ;

void initia()
{
    servogauchebassin.write(80);
    servodroitebassin.write(80);
    servogauchepied.write(80);
```

```
    servodroitepied.write(80);
    mar = true;
}

void marche()
{
    // etape1
    servogauchepied.write(105);
    delay(vitesse_equilibre);
    servodroitepied.write(100);
    delay(vitesse_etape);

    // etape2
    if (mar)
    {
        for (int i = 0; i < 25; i++)
        {

            servodroitebassin.write(80 - i);
            servogauchebassin.write(80 - i);
            delay(vitesse_smooth);
        } // 80 to 55
    }
    else
    {
        for (int i = 0; i < 50; i++)
        {

            servodroitebassin.write(105 - i);
            servogauchebassin.write(105 - i);
            delay(vitesse_smooth);
        } // 105 to 55
    }
    servogauchepied.write(80);
    delay(vitesse_etape);

    // etape3  // 100 to 80
    for (int i = 0; i < 20; i++)
```

```
    {
        servodroitepied.write(100 - i);
        delay(vitesse_redescente);
    }
    delay(vitesse_etape);

    // etape4
    servodroitepied.write(45);
    delay(vitesse_equilibre);
    servogauchepied.write(60);
    delay(vitesse_etape);

    // etape5  // 55 to 80
    for (int i = 0; i < 25; i++)
    {
        servodroitebassin.write(55 + i);
        servogauchebassin.write(55 + i);
        delay(vitesse_smooth);
    }
    servodroitepied.write(80);
    delay(vitesse_etape);

    // etape6  // 60 to 80
    for (int i = 0; i < 20; i++)
    {
        servogauchepied.write(60 + i);
        delay(vitesse_redescente);
    }
    delay(vitesse_etape);
    mar = true;
}

void course()
{

    // etape1
    servogauchepied.write(105);
    delay(vitesse_equilibre);
```

```
servodroitepied.write(100);
delay(vitesse_etape);

// etape2
if (mar)
{
    for (int i = 0; i < 25; i++)
    {
        servodroitebassin.write(80 - i);
        servogauchebassin.write(80 - i);
        delay(vitesse_smooth);
    } // 80 to 55
}
else
{
    for (int i = 0; i < 50; i++)
    {
        servodroitebassin.write(105 - i);
        servogauchebassin.write(105 - i);
        delay(vitesse_smooth);
    } // 105 to 55
}
servogauchepied.write(80);

// etape3  // 100 to 80
for (int i = 0; i < 20; i++)
{
    servodroitepied.write(100 - i);
    delay(vitesse_redescente);
}

// etape4
servodroitepied.write(45);
delay(vitesse_equilibre);
servogauchepied.write(60);
delay(vitesse_etape);

// etape5
```

```
    for (int i = 0; i < 55; i++)
    {
        Serial.println(i);
        servodroitebassin.write(50 + i);
        servogauchebassin.write(50 + i);
        delay(vitesse_smooth);
    } // angle 50 to 105
    servodroitepied.write(80);

    // etape6  // 60 to 80
    for (int i = 0; i < 20; i++)
    {
        servogauchepied.write(60 + i);
        delay(vitesse_redescente);
    }

    mar = false;
}

void droite()
{
    if (mar == false)
    {
        initia();
    }
    // etape1
    servogauchepied.write(105);
    delay(vitesse_equilibre);
    servodroitepied.write(100);
    delay(vitesse_etape);

    // etape2
    // baisser servodroitebassin de 80 a 55
    for (int i = 80; i >= 55; i--)
    {
        servodroitebassin.write(i);
        delay(vitesse_smooth);
    }
```

```
    // ramener servogauchepied vers 80
    for (int i = 105; i >= 80; i--)
    {
        servogauchepied.write(i);
        delay(vitesse_smooth);
    }
    delay(vitesse_etape);

    // etape3
    // abaisser servodroitepied vers 80
    for (int i = 105; i >= 80; i--)
    {
        servodroitepied.write(i);
        delay(vitesse_redescente);
    }
    delay(vitesse_etape);

    // etape4
    // baisser servogauchepied vers 60
    servogauchepied.write(60);
    delay(vitesse_equilibre);

    // remettre servo a 80 en douceur
    for (int i = 55; i <= 80; i++)
    {
        servodroitebassin.write(i);
        delay(vitesse_smooth);
    }
    for (int i = 60; i <= 80; i++)
    {
        servogauchepied.write(i);
        delay(vitesse_redescente);
    }
    delay(vitesse_etape);
}

void gauche()
{
```

```
if (mar == false)
{
    initia();
}
// etape1
servodroitepied.write(45);
delay(vitesse_equilibre);
servogauchepied.write(60);
delay(vitesse_etape);

// etape2
// pousser servogauchebassin vers 95 depuis 80
for (int i = 80; i <= 95; i++)
{
    servogauchebassin.write(i);
    delay(vitesse_smooth);
}
// ramener servodroitepied vers 80
for (int i = 45; i <= 80; i++)
{
    servodroitepied.write(i);
    delay(vitesse_smooth);
}
delay(vitesse_etape);

// etape3
// remettre servogauchepied a 80 douceur
for (int i = 60; i <= 80; i++)
{
    servogauchepied.write(i);
    delay(vitesse_redescente);
}
// pousser servodroitepied vers 95 progressivement
servodroitepied.write(95);
delay(vitesse_equilibre);
delay(vitesse_etape);

// etape4
```

```
    // ramener servo a 80 en douceur
    for (int i = 95; i >= 80; i--)
    {
        servogauchebassin.write(i);
        delay(vitesse_smooth);
    }
    for (int i = 95; i >= 80; i--)
    {
        servodroitepied.write(i);
        delay(vitesse_redescente);
    }
    delay(vitesse_etape);
}

void arriere()
{

    // etape1
    servogauchepied.write(105);
    delay(vitesse_equilibre);
    servodroitepied.write(100);
    delay(vitesse_etape);

    // etape2
    // pousser bassin arriere vers 105
    for (int i = 80; i <= 105; i++)
    {
        servodroitebassin.write(i);
        servogauchebassin.write(i);
        delay(vitesse_smooth);
    }
    // ramener servogauchepied a 80
    for (int i = 105; i >= 80; i--)
    {
        servogauchepied.write(i);
        delay(vitesse_smooth);
    }
        delay(vitesse_etape);
```

```
}

void loop()
{
    distD = analogRead(analogPinD);
    distG = analogRead(analogPinG);

    if (distD > 300 && distG > 300)
    {
        marche();
    }
    else if (distD <= 300 && distG > 300)
    {
        droite();
    }
    else if (distD > 300 && distG <= 300)
    {
        gauche();
    }
    else
    {
        arriere();
    }
}
```

6.2 Mise en situation

FIGURE 4 – Bipède en action