Faire danser le Petit Gaillard- Arduino

1. Objectifs :

1. Faire danser le Petit Gaillard en le faisant passer d'une posture à l'autre
2. Utiliser des fonctions pour aider la programmation
3. Faire des mouvements plus graduels en utilisant une boucle « for »

2. Matériel :

- Un kit de Petit Gaillard
- Un ordinateur

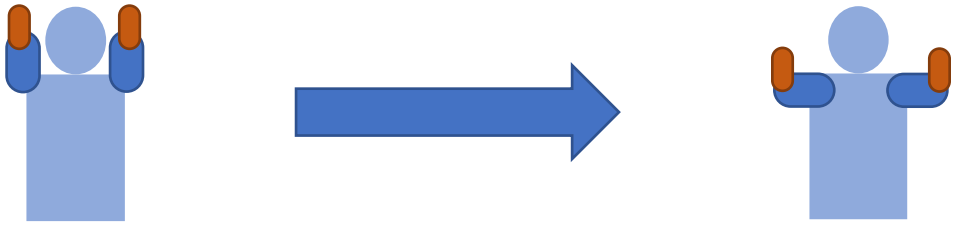
3. Passer d'une posture à l'autre

Préalable :

- Faire les activités du document « Petit Gaillard Arduino.pdf » avant celui-ci.
- Préparer le Petit Gaillard tel que montré dans « Petit Gaillard Arduino.pdf »
- Choisir les postures qui doivent être utilisées dans la danse !

Pour faire danser le Petit Gaillard, la première solution est de le faire passer directement d'une posture à l'autre, en ajoutant un temps d'attente entre les deux. Le but du temps d'attente est simplement pour permettre aux servomoteurs de se rendre à la nouvelle position. Cela peut permettre aussi de synchroniser des mouvements avec un rythme en particulier.

Le premier code proposé permet de faire passer le Petit Gaillard, préalablement préparé avec la méthode proposée dans « Petit Gaillard Arduino.pdf » d'une position où les bras sont complètement en haut à une posture dont les bras sont en coin à 90 degrés.



```
#include <Servo.h>

//Définition des variables moteurs.
Servo epauleG;
Servo epauleD;
Servo coudeG;
Servo coudeD;

//Ce code est exécuté une seule fois.
void setup() {

    epauleG.attach(8);
    epauleD.attach(9);
    coudeG.attach(10);
    coudeD.attach(11);

    //Bras en haut
    epauleG.write(0);
    epauleD.write(180);
    coudeG.write(180);
    coudeG.write(0);

    delay(1000);

    //Bras en coin
    epauleG.write(90);
    epauleD.write(90);
    coudeG.write(90);
    coudeG.write(90);

    delay(1000);
}

//Ce code est exécuté en boucle ensuite.
void loop() {

}
```

Il y a un temps d'attente de 1 seconde (1000 millisecondes) entre les deux postures.

Ces deux postures ne sont effectuées qu'une seule fois. Pour recommencer le mouvement, il faut couper l'alimentation de la carte Arduino en la débranchant de l'ordinateur et ensuite en la rebranchant. On peut aussi tout simplement appuyer sur le bouton « reset » qui se trouve sur la carte Arduino.

Si on veut programmer un mouvement qui se répète, on doit mettre les lignes de code contrôlant les servomoteurs et le temps d'attente dans la fonction « loop », comme ceci :

```
#include <Servo.h>

//Définition des variables moteurs.
Servo epauleG;
Servo epauleD;
Servo coudeG;
Servo coudeD;

//Ce code est exécuté une seule fois.
void setup() {
    epauleG.attach(8);
    epauleD.attach(9);
    coudeG.attach(10);
    coudeD.attach(11);
}

//Ce code est exécuté en boucle ensuite.
void loop() {

    //Bras en haut
    epauleG.write(0);
    epauleD.write(180);
    coudeG.write(180);
    coudeG.write(0);

    delay(1000);

    //Bras en coin
    epauleG.write(90);
    epauleD.write(90);
    coudeG.write(90);
    coudeG.write(90);

    delay(1000);

}
```

Tout ce qui est placé dans la fonction « loop » sera effectué pour toujours. Donc, une fois que les lignes de code à l'intérieur ont été exécutées, on recommence au début de la fonction « loop ». Le tout sera exécuté tant que la carte Arduino est alimentée.

Il n'est pas toujours nécessaire d'avoir un bloc de contrôle pour la position de chaque servomoteur, si, par exemple, un moteur reste à la même position entre deux postures.

Défi : créer une danse avec au moins 4 postures différentes et des temps d'attente qui varient entre les 4 postures. On peut mettre plus de postures, bien sûr…

4. Utilisation des fonctions

On remarque assez vite que le code permettant de faire une danse devient très long rapidement et qu'il est parfois difficile de se rappeler l'ordre de chaque posture. En plus, si l'on utilise plusieurs fois la même posture, on répète exactement les mêmes lignes de code.

Pour faciliter la lecture, la programmation et aussi pour éviter que des sections de code ne se répètent, on peut utiliser des fonctions. Celles-ci permettent de regrouper une partie de code et de le réutiliser à plusieurs endroits dans le code.

Par exemple, nous créerons une fonction pour la posture des bras en coin. On place les définitions des fonctions en-dessous de la boucle « loop », après la fermeture de son accolade. En fait, on pourrait placer les fonctions n'importe où dans le code, soit avant la fonction « setup », entre les fonctions « setup » et « loop » ou après celle-ci. C'est un choix. Pour l'instant, elles seront placées en-dessous de « loop ». On peut donc modifier le code comme ceci :

```
#include <Servo.h>

//Définition des variables moteurs.
Servo epauleG;
Servo epauleD;
Servo coudeG;
Servo coudeD;


//Ce code est exécuté une seule fois.
void setup() {

    //Il faudra peut-être changer la valeur de la broche pour
    //les moteurs
    epauleG.attach(8);
    epauleD.attach(9);
    coudeG.attach(10);
    coudeD.attach(11);
}
```

```
//Ce code est exécuté en boucle ensuite.
void loop() {

    //Bras en haut
    epauleG.write(0);
    epauleD.write(180);
    coudeG.write(180);
    coudeG.write(0);
    delay(1000);

    //Bras en coin
    brasEnCoin(); //Utilisation de la fonction brasEnCoin()
    delay(1000);
}

//Ceci est une fonction.  Ici, ce n'est que la définition.
void brasEnCoin() {
    epauleG.write(90);
    epauleD.write(90);
    coudeG.write(90);
    coudeG.write(90);
}
```

Pour utiliser la fonction, on doit l'utiliser dans « loop » ou « setup ».

Défi : Créer une fonction pour chaque posture déjà utilisée précédemment et les utiliser dans « loop ».

On peut aussi créer des fonctions qui prennent une valeur en entrée. Par exemple, on pourrait créer une fonction appelée « posture » qui prend 4 valeurs numériques (entiers ou « int ») en entrée, appelées angle1, angle2, angle3 et angle4.

```
void posture (int angle1, int angle2, int angle3, int angle4) {

    epauleG.write(angle1);
    epauleD.write(angle2);
    coudeG.write(angle3);
    coudeG.write(angle4);

}
```

Lorsqu'on utilise cette fonction, il faut entrer 4 valeurs numériques (entiers) à l'intérieur de la parenthèse (le reste du code a omis ici, mais doit être présent pour que le code fonctionne):

```
void loop() {

    //Bras en haut
    posture(0, 180, 180, 0);
    delay(1000);

    //Bras en coin
    posture(90, 90, 90, 90);    //Utilisation de la fonction
    delay(1000);
}

void posture (int angle1, int angle2, int angle3, int angle4) {

    epauleG.write(angle1);
    epauleD.write(angle2);
    coudeG.write(angle3);
    coudeG.write(angle4);
}
```

Défi : Créer une fonction « posture2 » qui prend en entrée 4 valeurs d'angle et un temps d'attente et l'utiliser pour créer une danse.

5. Utilisation des boucles « for »

Supposons que l'on souhaite répéter une action pour toujours, on peut utiliser le bloc « loop ». Cependant, il existe d'autres structures de contrôle qui permettent de répéter une série de lignes de code pour un nombre de fois prédéterminé où jusqu'à ce qu'une certaine condition soit atteinte.

Regardons les boucles « for » :

```
for(int i =0 ;  condition ; i = i +increment){
      //code à exécuter
  }
```

La fonction ***for*** (« pour » en français) est une fonction très importante et très utilisée pour réaliser des boucles de contrôle dans un programme.

Elle a trois arguments :

1. Le premier est la valeur d'***initialisation***. Ici, on déclare la variable *i* qui n'existera donc que dans la boucle *for*. On lui attribue sa valeur de départ dans la boucle. C'est la valeur de la variable *i* au premier passage dans la boucle.
2. Le deuxième argument est la ***condition***. La boucle sera exécutée tant que la condition est vraie. Si la condition devient fausse, on sort de la boucle et on passe aux commandes suivantes.
3. Le troisième argument est appelé l'***incrément***. Il indique comment la variable *i* sera modifiée à chaque exécution du bloc d'instructions de la boucle. L'incrément peut être fait avec une soustraction, une multiplication ou une division.

Par exemple:

```
for(int i =0 ;  i < 3 ; i = i + 1){
      //code à exécuter
  }
```

Le déroulement de l'exécution de la boucle sera ainsi :

Début de la boucle :

1er passage :

i=0

la condition est vraie, on exécute le bloc de commandes

i=0+1 =1

2e passage :

i=1

La condition est vraie, on exécute le bloc de commandes

i=1+1=2

3e passage :

i=2

La condition est vraie, on exécute le bloc de commandes

i=2+1=3

4e passage

i=3

La condition est fausse, on n'exécute pas le bloc de commandes et on sort de la boucle *for.*

Fin de la boucle

Il est possible d'utiliser une autre valeur de départ que 0, de changer la condition pour autre chose et aussi de changer la valeur de l'incrément. Ici, on a utilisé une valeur de « 1 ». Cela fait faire des bonds de 1 à la valeur de « i », mais on pourrait aussi utiliser des valeurs plus grandes, 2, 3, 4, etc. On pourrait aussi utiliser des valeurs négatives, ce qui ferait diminuer la valeur de i.

Défi : Utiliser un boucle « for » pour refaire 3 fois le même changement entre 2 postures différentes. Utiliser les fonctions créées plus haut pour faire les postures.

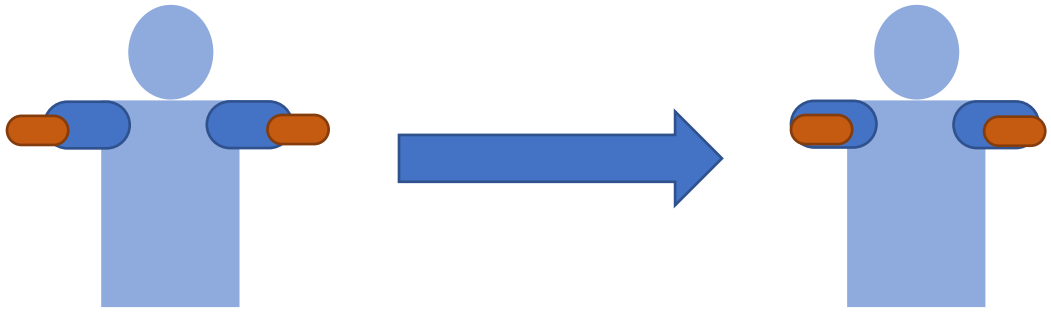
On peut aussi utiliser la valeur de « i » pour contrôler l'angle d'un servomoteur. Ainsi, on peut rendre un mouvement plus graduel ou pour en contrôler le rythme. Jusqu'ici, le passage entre deux postures s'effectue très rapidement. Si l'on souhaite plutôt que le passage d'un bras d'en bas à en haut se fasse en un temps plus long, on ne peut pas simplement passer de la position du bras en haut à en bas directement, car le mouvement se fait tout d'un coup. Il faut plutôt passer par plusieurs valeurs d'angle intermédiaires et mettre un temps d'attente entre chaque valeur.

Par exemple si on veut qu'un moteur passe d'un angle de 0 degré à un angle de 180 degrés en environ 1 seconde, on pourrait lui faire faire des bonds de 10 degrés à chaque 0.05 secondes. Bien entendu, programmer au long chaque changement serait très long… Avec une boucle « for », cela revient à :

```
for(int angle=0 ; angle <= 180 ; angle = angle + 10){
     epauleG.write(angle);
     delay(50);
}
```

On peut ajuster le mouvement en modifiant le temps d'attente, les angles de départ et d'arrivée et de de combien on modifie la valeur de l'angle d'un passage à l'autre dans la boucle. Si l'on fait des bonds de plus grande valeur, le mouvement sera plus saccadé. Même chose si l'on augmente la valeur de temps d'attente entre deux valeurs. En revanche, le mouvement sera plus rapide si le temps d'attente est plus court et si les bonds sont plus grands. Il faut donc trouver la meilleure combinaison entre la rapidité et la fluidité.

Défi : Faire tourner les deux avant-bras en même temps en un mouvement lent et graduel pour passer d'une posture à l'autre comme ceci :



6. Défi ultime

Choisir une chanson de votre choix et synchroniser les mouvements du Petit Gaillard avec la musique ! Utiliser les postures choisies ou d'autres, des blocs « Répéter x fois », des blocs « Répéter jusqu'à » et des blocs personnalisés !