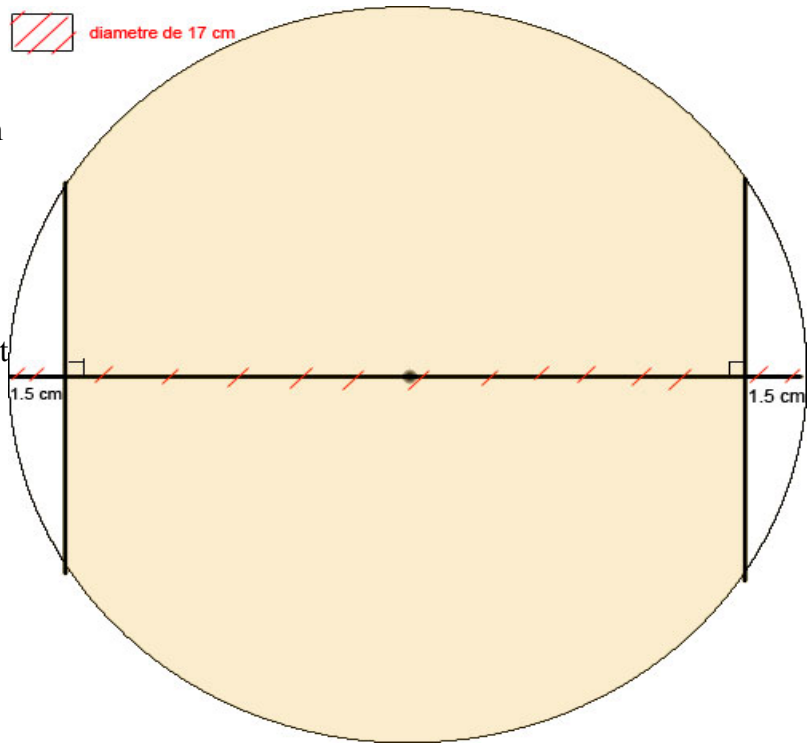
La structure porteuse du robot

Le châssis :

Matériel nécessaire :

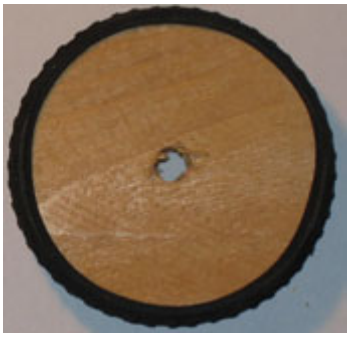
- une plaque de contreplaqué de 5 ou de 6 min d'épaisseur.
- une scie à bois
- une lime à bois
- une perceuse



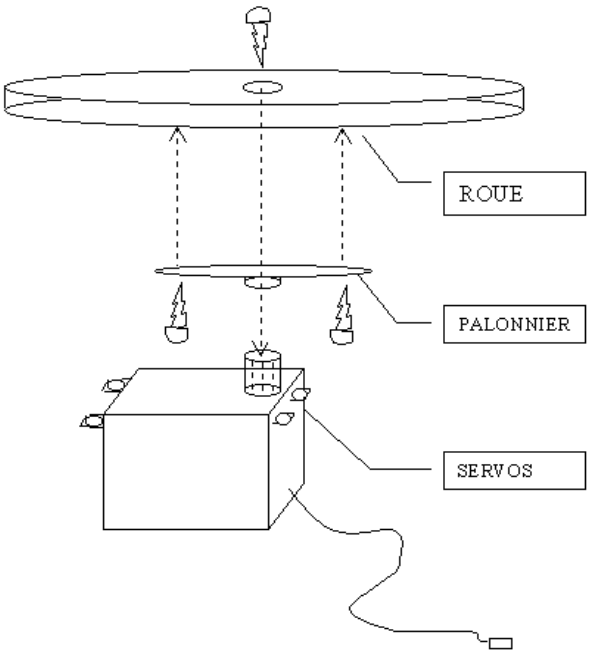
Tracez tout d'abord le châssis avec un crayon à papier sur la plaque en bois comme sur le schéma suivant : Découpez-le ensuite à l'aide d'une scie et d'une lime à bois et percez-le au centre à l'aide d'une perceuse et d'un forêt au diamètre assez important comme 10 mm.

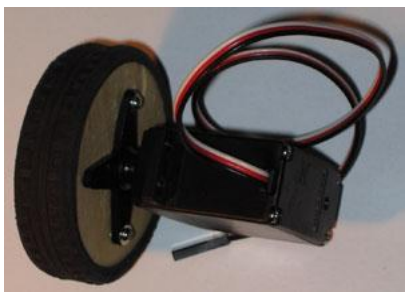
Les roues :

Découpez à l'aide de la scie cloche deux roues de 60 mm de diamètre dans la plaque de bois.



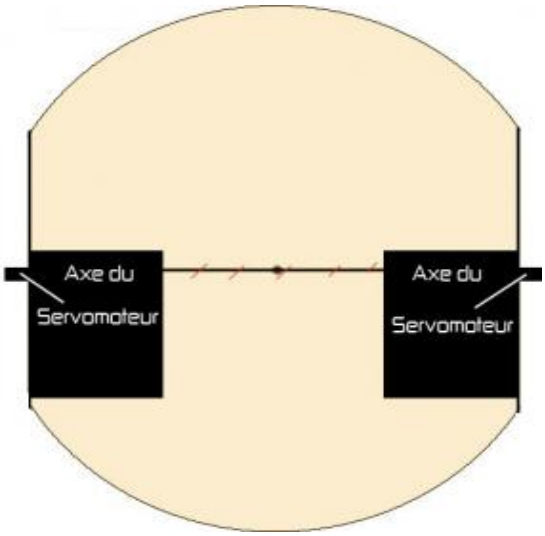
Fixez-les ensuite sur les servomoteurs comme sur ce schéma :





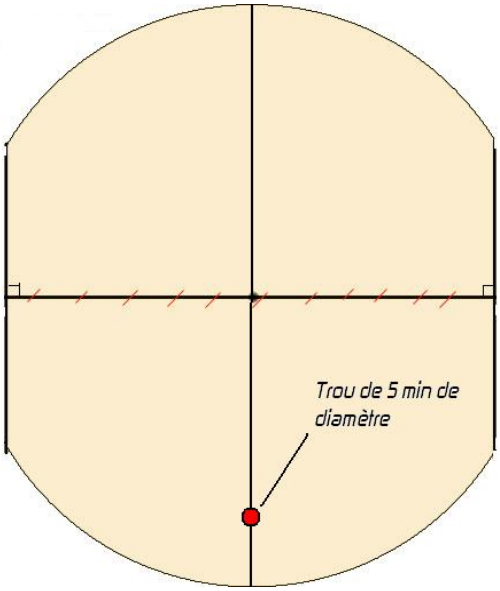
La fixation des servomoteurs

Fixez les servomoteurs, de façon que leurs axes soient au centre du robot :



La troisième roue

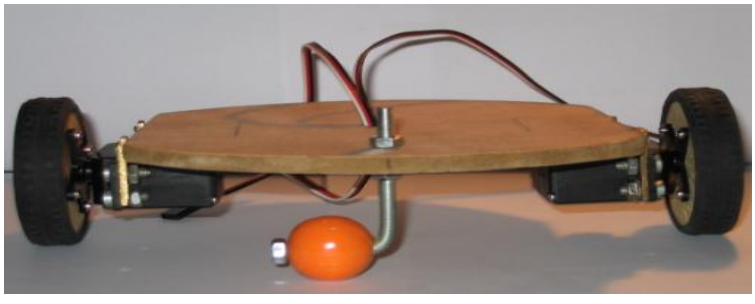
Pour ne pas que le robot penche d'un côté, il va falloir rajouter une troisième roue.
Percez le châssis à l'aide d'un forêt de 5 mm de diamètre comme sur le schéma suivant :



Il va falloir :

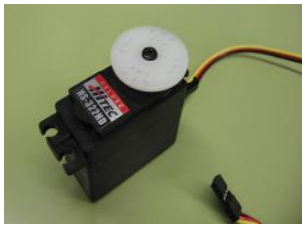
- une tige filetée de 5 mm de diamètre ainsi que des écrous adaptés.
- une perle en bois
- un petit niveau pour contrôler la planéité du châssis du robot.

Pliez la tige filetée à 90° et emprisonnez-la entre des écrous comme sur la photo.
Attention, la boule doit pouvoir tourner sans frottements.



Modifier les servomoteurs en moteur :

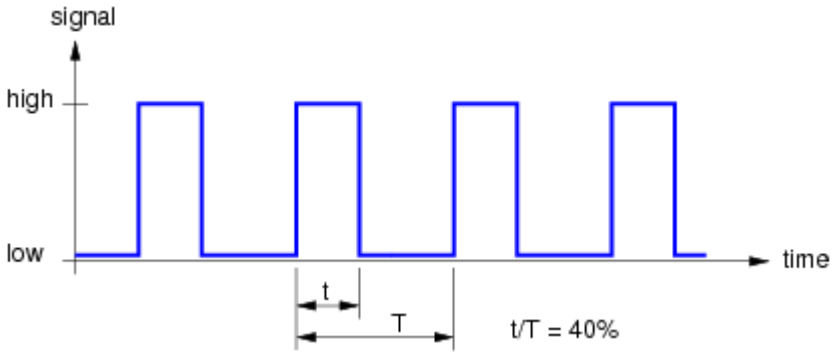
Les servomoteurs sont des ensembles mécanique composés d'un moteur courant continu, d'un servomoteur, d'un potentiomètre et d'un système électronique d'asservissement.



Le moteur peut tourner selon un certain angle, qui dépend du servomoteur : il y a une butée mécanique qui l'empêche d'aller plus loin, et le potentiomètre permet de connaitre la position de l'axe du servomoteur, c'est à dire l'angle que forme l'axe du servomoteur avec la butée.

La commande du moteur se fait selon un signal carré dont la durée des impulsions détermine l'angle que doit atteindre l'axe du servomoteur :

Un servomoteur contient un réducteur, un moteur, un système d'asservissement et un boitier solide et compacte. Ainsi, si on pouvait enlever la butée du servomoteur, et l'utiliser comme moteur à courant continu, on aurait à disposition un moteur avec un bon couple, dans un beau boitier, on disposerait de la marche arrière.



Principe:

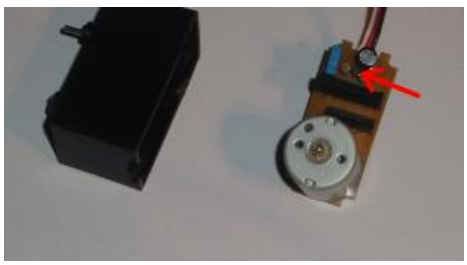
Lorsque l'on envoi une commande au moteur, l'asservissement alimente le moteur jusqu'à ce que la commande corresponde à la tension du potentiomètre, c'est à dire à la bonne position.

Maintenant, si on remplace le potentiomètre par une résistance fixe, qui correspond à un angle de 0 degrés, lorsque l'on lui enverra un signal pour aller dans un sens ou dans l'autre, il va tourner tourner dans ce sens, mais il se croira toujours à 0 degrés, et va continuer à tourner indéfiniment !

Ainsi, nous avons un moteur à courant continu.

Enlever la butée :

Le servomoteur possède une protection, une butée mécanique qu'il faut enlever.
Commencez par démontez le servomoteur et enlevez avec un cutter ou une pince la butée.



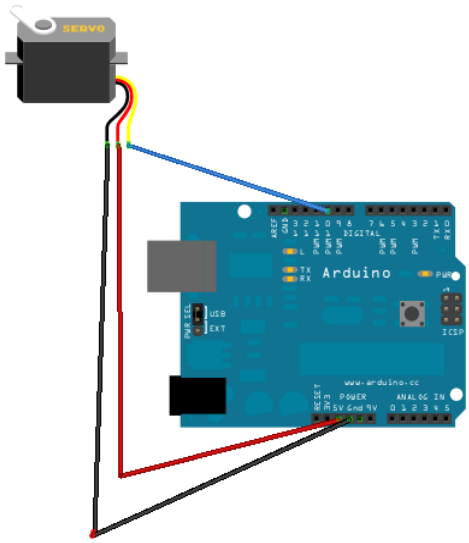
Remplacer le potentiomètre par une résistance variable

Enlevez le circuit électronique et dessoudez le potentiomètre.
Remplacez le par deux résistances montées en pont diviseur de tension.

Pour la valeur des résistances, il faudra chercher, cela dépend de votre servomoteur. L'idéal est d'avoir deux résistances de la valeur du potentiomètre quand le servomoteur se trouve à un angle de

0 degrés.
Ceci-dis, si on est pas super précis, cela n'a pas une réelle importance, il faudra juste trouver la valeur du "neutre".
Personnellement, j'ai utilisé pour mon servomoteur deux résistances de 2.2 Kohm.

A la recherche du neutre



Prenez votre arduino et réalisez le circuit suivant :

On alimentera la carte par l'USB.

On va écrire une programme qui fait tourner le servomoteur entre un angle de 0 à 180 degrés, et on va regarder à quelle impulsion le moteur s'arrête : il s'agira du neutre.

Voici le programme en code C :

```
#include <Servo.h>

Servo servo;

void setup()
{
  servo.attach(10); // On attache le servomoteur à la patte 10 de l'arduino
  Serial.begin(9600); // On va envoyer des informations à l'ordinateur grâce à ce port
}

void loop()
{
  int i = 0;
   while(i <= 180)
   {
      servo.write(i); //On fait avancer le servomoteur de 5 en 5, de 0 jusqu'à 180
      Serial.println(i); //On affiche la valeur en cours de l'angle
      delay(1000); // On attend une seconde pour avoir le temps de lire la valeur qui nous interesse

     i = i+5;
   }
   while(i >= 180)
   {
      servo.write(i); //on fait reculer ensuite le servomoteur
      Serial.println(i);
      delay(1000);
     i = i-5;
   }

}
```

On fait simplement varier l'angle, et à chaque fois on affiche la valeur de cet angle.
Je trouve personnellement que mon moteur s'arrête, puis change de sens à 85°.
Il est ensuite possible d'être plus précis en allant de 1 en 1. Je trouve alors un angle de 84°.

Commande du moteur

Et voilà, le modification du moteur est maintenant terminée.
Si vous envoyez un angle plus grand que le neutre votre moteur ira dans un sens, et si vous envoyez un angle plus petit, il ira dans l'autre sens.
Il est même possible d'avoir des vitesses : on peut faire ralentir le moteur en envoyant une valeur

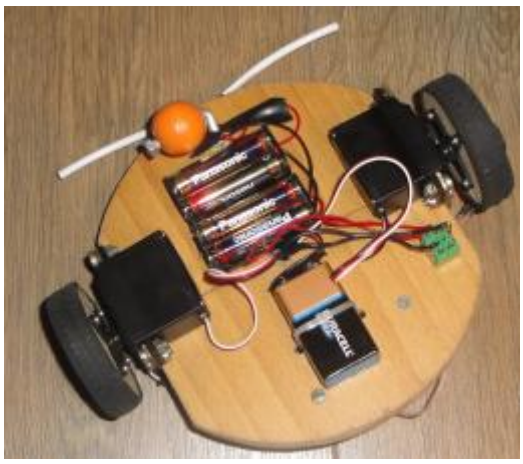
proche du neutre.
Voici une fonction que j'ai écris pour contrôler facilement mon moteur, il suffit de lui donner une vitesse comprise entre -5 et 5, et il avancera ou reculera suivant cette vitesse :
Code C :

```
#define SERVO 0

int getNeutral = 84; // La valeur du neutre de mon servomoteur
void handleS(int s, int speed)
{
  //Vitesse :
  // 0 -> 0; 1 -> 1; 2 -> 3; 3 -> 8; 4 -> 10; v => 5 -> 30
  int tab[6] = {0,1,3,8,10,30};
  int pos = getNeutral;

int acc;
if (speed < 0)
  acc = - tab[abs(speed)];
else
  acc = tab[speed];

  servog.write(pos += acc);
}

```

Assemblage du robot

Alimentation

Il y a dans ce robot deux alimentations : une pile 9 V pour alimenter l'arduino et 4 piles 1.5 V en série pour alimenter les servomoteurs et autres capteurs.

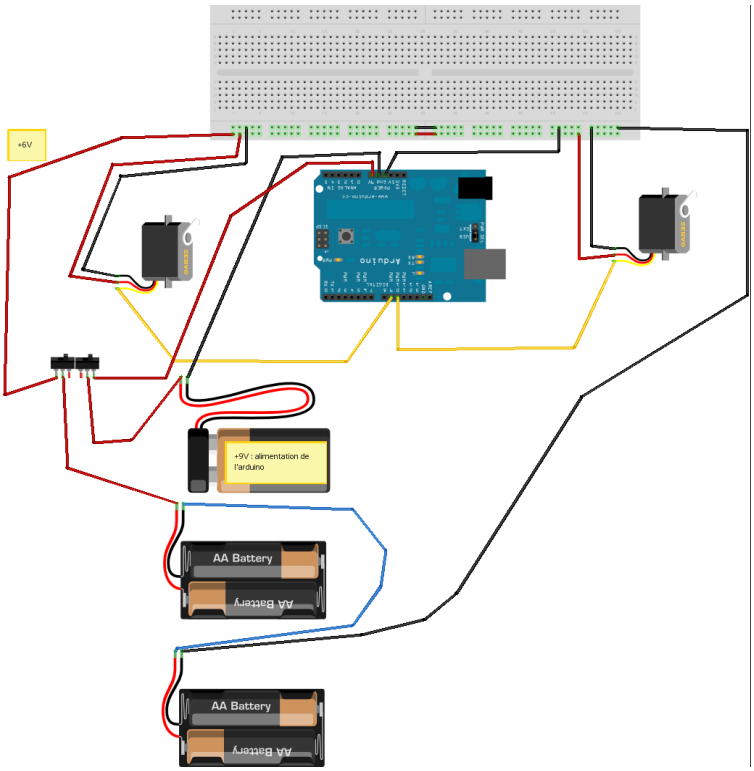


L'avantage de ce système, c'est que l'on peut alimenter l'arduino par la prise USB lorsque l'on écrira le programme, tout en gardant alimenté les servomoteurs : le débogage est donc plus aisé.

Attention cependant : il faut faire attention a bien relier les deux masses entre elle, cela peut être source de bogues assez vicieux.
Vous pouvez fixer vos piles comme sur cette photo.

Commencez par fixez les deux alimentations.

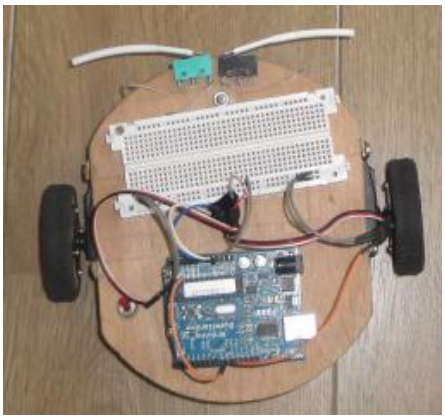
Faites un trou au milieu du robot pour laisser passer les fils de l'alimentation et un trou sur le coté pour fixer l'interrupteur.
L'idéal c'est d'avoir un interrupteur capable de couper/alimenter deux circuits à la fois. De cette façon, vous pourrez éteindre ou allumer votre robot en une fois.
Maintenant, si vous n'avez que deux interrupteurs normaux, ça fonctionnera aussi.



Fixez sur le robot l'arduino et la plaque d'essai.

Câblez alors les alimentations, l'arduino et les servomoteurs.

Vous avez donc avec ce schéma 6V pour les deux servomoteurs et 9 V pour l'arduino.
Les deux masses sont bien reliées entres elles.
Sur la plaque d'essai, la première ligne correspond donc aux +6V et la deuxième à la masse.
Voici une photo de mon robot à cette étape.



Test des moteurs

On va écrire dans un premier temps un programme qui fait avancer le robot tout droit.

Tout d'abord, nous allons écrire une fonction toute simple, getNeutral(moteur) qui retournera la valeur du neutre de vos servomoteurs :
Code : C++

```
#include <Servo.h>
#define SERVOG 1
#define SERVOD 0

Servo servog;
Servo servod;

int getNeutral(int s)
{
 if(s == SERVOG)
   return 86;
 else
   return 84;
}
```

Chaque servomoteur est représenté par un entier naturel : le servomoteur gauche 1 et le servomoteur droit 0.
Remplacez les valeurs 86 et 84 respectivement par les valeurs du neutre que vous trouvez pour le servomoteur gauche et pour le servomoteur droit de votre robot.

Essayez de faire bouger vos servomoteurs en utilisant cette fonction.
Tout va bien ? Ok, nous allons écrire une fonction handleS(servomoteur, vitesse) qui envoit une commande au servomoteur. Cette commande est caractérisé par une vitesse qui va de -5 à 5. Si la vitesse est négative, on dira au moteur d'aller dans un sens et si le vitesse est positive on dira au robot d'aller dans l'autre sens.

Les différentes vitesses sont obtenues en se rapprochant du neutre. Je vous conseil de faire vos propres tests.
Attention cependant : nos servomoteurs ne sont pas dans le même sens. Pour avancer tout droit, un des servomoteurs doit aller dans un sens, et l'autre servomoteur doit aller dans l'autre sens.
On ne va donc pas oublier d'inverser le sens de rotation pour un des moteurs.
Code : C++

```
void handleS(int s, int speed)
{
  //Vitesse :
  // 0 -> 0; 1 -> 1; 2 -> 3; 3 -> 8; 4 -> 10; v => 5 -> 30
  int tab[6] = {0,1,3,8,10,30};
  int pos = getNeutral(s);

  int acc;
  // A chaque vitesse on fait correspondre un nombre qui est la différence de ce que l'on va envoyer au servomoteur par rapport au neutre.
  // Les valeurs que je donne sont mes valeurs, je vous conseille de chercher vos propres valeurs, cela dépend de votre servomoteur.
  if (speed < 0)
    acc = - tab[abs(speed)];
  else
    acc = tab[speed];

```

```
  if(s == SERVOG)
    servog.write(pos += acc);
      // On inverse le sens de rotation pour le moteur droit car les deux servomoteurs ne sont pas montés dans le même sens.
  else
    servod.write(pos -= acc);
}

```

A partir de là, on peut écrire une fonction move(direction) qui fait déplacer le robot dans une direction donnée :

Code : C++

```
#define AVANT 1

#define ARRIERE 0
#define GAUCHE 2
#define DROITE 3

#define VMAX 5

void move(int direction)
{
  int m1 = 0, m2 = 0;
  switch(direction)
  {
    case DROITE: m1 = 1; m2 = -1; break;
    case GAUCHE: m1 = -1; m2 = 1; break;
    case AVANT: m1 = 1; m2 = 1; break;
    case ARRIERE: m1 = -1; m2 = -1; break;
  }
  handleS(SERVOG, m1*VMAX);
  handleS(SERVOD, m2*VMAX);
}

```

Rien de bien compliqué : en fonction du déplacement, on indique dans quel sens doit se déplacer chaque moteur.

Code complet

On va, en utilisant toutes ces fonctions, demander au robot d'avancer tout droit :

Code : C++

```
#include <Servo.h>

#define AVANT 1
#define ARRIERE 0
#define GAUCHE 2
#define DROITE 3

#define SERVOG 1
#define SERVOD 0

#define VMAX 5

Servo servog;
Servo servod;

int getNeutral(int s)
{
  if(s == SERVOG)
    return 86;
  else
```

```
21.     return 84;
22. }
23.
24.
25. void handleS(int s, int speed)
26. {
27.   //Vitesse :
28.   // 0 -> 0; 1 -> 1; 2 -> 3; 3 -> 8; 4 -> 10; v => 5 -> 30
29.   int tab[6] = {0,1,3,8,10,30};
30.   int pos = getNeutral(s);
31.
32.   int acc;
33.   if (speed < 0)
34.     acc = - tab[abs(speed)];
35.   else
36.     acc = tab[speed];
37.
38.   if(s == SERVOG)
39.       servog.write(pos += acc);
40.   else
41.       servod.write(pos -= acc);
42. }
43.
44.
45. void move(int direction)
46. {
47.   int m1 = 0, m2 = 0;
48.   switch(direction)
49.   {
50.     case DROITE: m1 = 1; m2 = -1; break;
51.     case GAUCHE: m1 = -1; m2 = 1; break;
52.     case AVANT: m1 = 1; m2 = 1; break;
53.     case ARRIERE: m1 = -1; m2 = -1; break;
54.   }
55.   handleS(SERVOG, m1*VMAX);
56.   handleS(SERVOD, m2*VMAX);
57. }
58.
59.
60.
61. void setup()
62. {
63.   servog.attach(10);
64.   servod.attach(9);
65.   move(AVANT);
66. }
67.
68. void loop()
69. {
70.   delay(15);
71. }
```

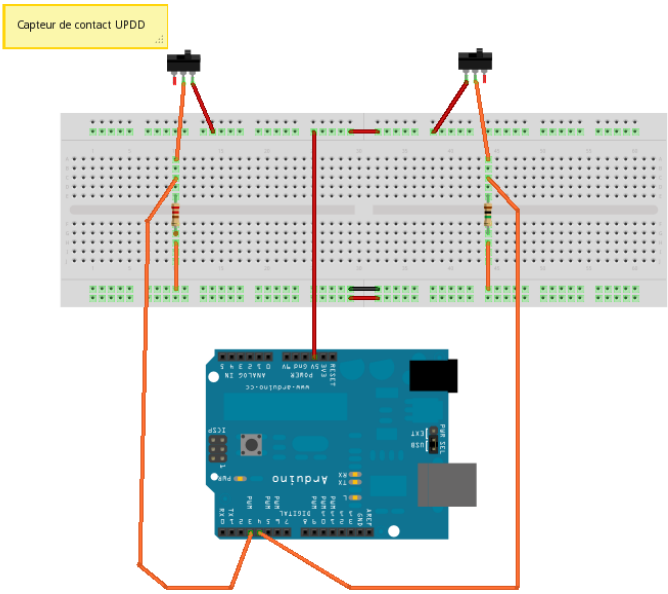
Cela est très simple avec la fonction move : move(AVANT);
Maintenant, on va faire faire avancer le robot pendant 8 secondes, lui faire faire un demi-tour, le faire avancer pendant 8 secondes, et ainsi de suite.
Pour faire tourner le robot, on va le faire tourner vers la gauche pendant un certain temps.
Personnellement, je trouve qu'il faut le laisser tourner 2 secondes pour qu'il fasse un demi-tour complet.
Ainsi, tout naturellement, le code est le suivant :
Code : C++

```
1.  void loop()
2.  {
3.    move(AVANT);
```

```
    delay(8000);
    move(GAUCHE);
    delay(2000);
}
```

Capteurs de contact : détecter les obstacles

Nous allons rajouter des capteurs de contact de type UPDD. Il faut donc les connecter a deux pattes de l'arduino, et nous allons simplement regarder les états des deux pattes de l'arduino : si l'une est à 1, il faut enclencher un virage.

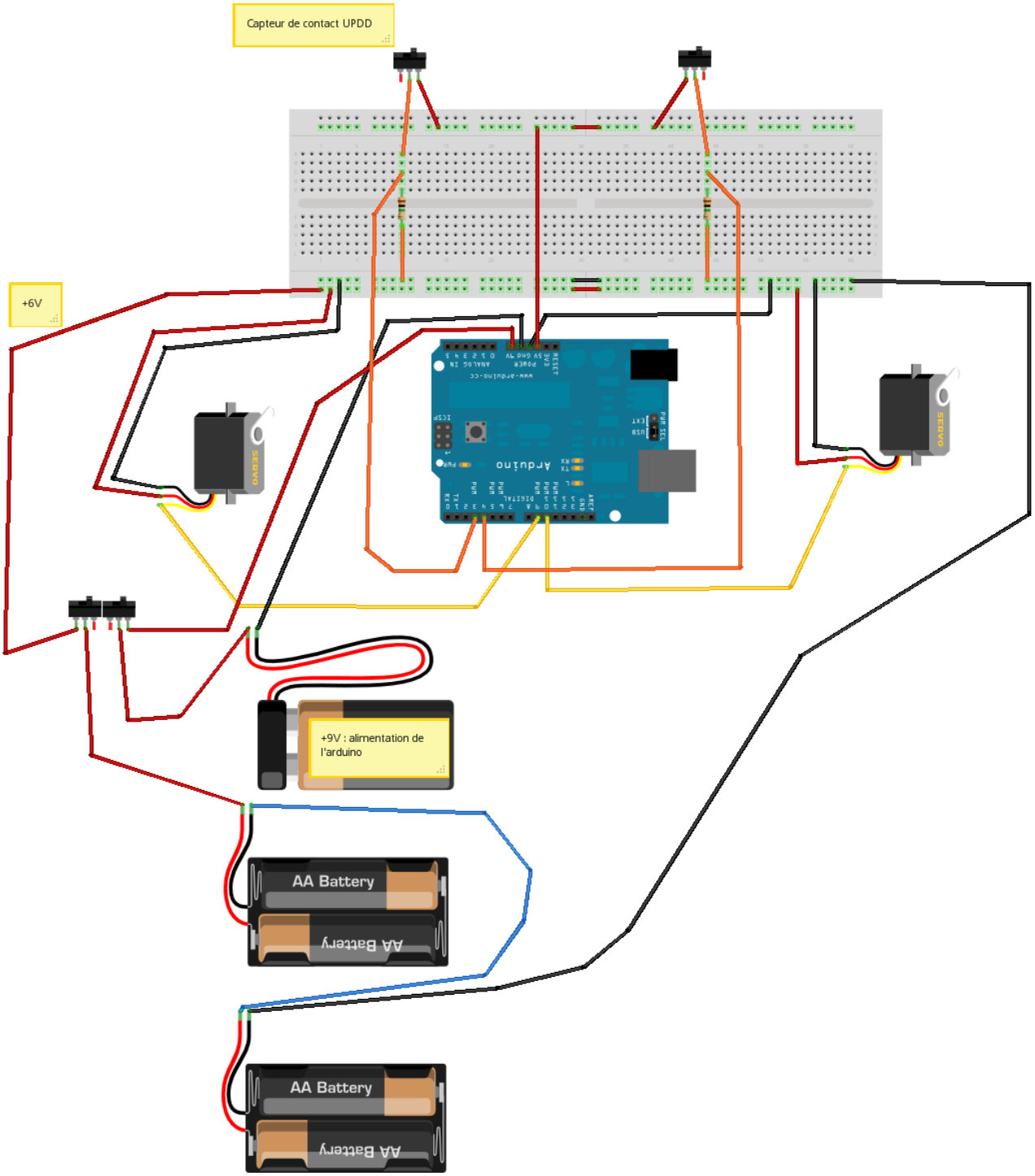


Les deux résistances sont des résistances de "pull-down", reliées à la masse pour éviter d'avoir des pattes flottantes et pour forcer l'état normal des deux pattes à 0.

Le code pour faire éviter les obstacles fait reculer, puis de tourner le robot à la détection de chaque obstacle :
Code : C++

```
#define UPDDG 3 // on utilise la patte 2 pour le capteur de gauche et 4 pour le capteur de droite.
#define UPDDD 4

void handleUPDD(int updd)
{
 if(digitalRead(updd) == HIGH) //Si le capteur est enclenché
  {
    move(ARRIERE); // on recule d'abord pendant 0.4 sec
    delay(400);

    if (updd == UPDDG) // Puis on tourne dans le sens inverse du capteur détecté...
    {
      move(DROITE);
    }
    else
    {
      move(GAUCHE);
    }
    delay(600); // ...pendant 0.6 sec
  }
}

```

Voici le schéma complet du robot



le code complet Code C++

```
#include <Servo.h>

#define AVANT 1
#define ARRIERE 0
#define GAUCHE 2
#define DROITE 3

#define SERVOG 1
#define SERVOD 0

#define UPDDG 3
#define UPDDD 4

#define VMAX 5

Servo servog;
Servo servod;

int getNeutral(int s)
{
  if(s == SERVOG)
    return 86;
  else
    return 84;
}


void handleS(int s, int speed)
{
  //Vitesse :
  // 0 -> 0; 1 -> 1; 2 -> 3; 3 -> 8; 4 -> 10; v => 5 -> 30
  int tab[6] = {0,1,3,8,10,30};
  int pos = getNeutral(s);

  int acc;
  if (speed < 0)
    acc = - tab[abs(speed)];
  else
    acc = tab[speed];

  if(s == SERVOG)
      servog.write(pos += acc);
  else
      servod.write(pos -= acc);
}


void move(int direction)
{
  int m1 = 0, m2 = 0;
  switch(direction)
  {
    case DROITE: m1 = 1; m2 = -1; break;
    case GAUCHE: m1 = -1; m2 = 1; break;
    case AVANT: m1 = 1; m2 = 1; break;
    case ARRIERE: m1 = -1; m2 = -1; break;
  }
  handleS(SERVOG, m1*VMAX);
  handleS(SERVOD, m2*VMAX);
```

```
}

void handleUPDD(int updd)
{
  if(digitalRead(updd) == HIGH)
  {
    move(ARRIERE);
    delay(500);

    if (updd == UPDDG)
    {
      move(DROITE);
    }
    else
    {
      move(GAUCHE);
    }
    delay(800);
  }
}


void setup()
{
  pinMode(UPDDG, INPUT);
  pinMode(UPDDD, INPUT);

  servog.attach(10);
  servod.attach(9);
  move(AVANT);
}

void loop()
{
  handleUPDD(UPDDG);
  handleUPDD(UPDDD);
  move(AVANT);
  delay(15);
}

```