École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Compte rendu
Robotique
Initiation à la robotique mobile
Mise en oeuvre matérielle et logicielle

Réalisé par :

SANBA Morad
YAN Wenli

DI5 Informatique

4. Montage du shield Boe et de l'arduino

4.1. Utilisation de Led

4.1.2. Faire clignoter la Led – Version 1

Dans un premier temps nous avons commencé par fait clignoter la led 13 on utilisant le code suivant :

Led_V1.ino

```
#define LED_pin 13
#define TEMPO_High 500
#define TEMPO_Low 750

int i;
long int tempo;

void setup() {
    Serial.begin(9600);
    pinMode(LED_pin, OUTPUT);  //Broche 13 en sortie
    digitalWrite(LED_pin, LOW);  //Broche 13 état bas etLED éteinte

//version 1

    for(i = 0; i < 10; i++)
    {
      Serial.print("\t : ");
      Serial.println(i);
      digitalWrite(LED_pin, HIGH);
      Serial.println("LED etat haut");
      for(tempo = 0; tempo< TEMPO_High; tempo++);
      digitalWrite(LED_pin, LOW);
      Serial.println("LED etat bas");
      for(tempo = 0; tempo< TEMPO_High; tempo++);

    }
```

4.1.3. Faire clignoter la Led – Version 2

Dans cette partie nous avons modifier le code précédent pour gérer les durées t1 et t2. Nous allons utiliser ici la fonction ***delay*** qui admet en paramètre une durée exprimée en ms (10-3 s).

Led_V2

```
#define LED_pin 13
#define TEMPO_High 500
#define TEMPO_Low 750

int i;
long int tempo;

void setup() {
    Serial.begin(9600);
    pinMode(LED_pin, OUTPUT);  //Broche 13 en sortie
    digitalWrite(LED_pin, LOW);  //Broche 13 état bas etLED éteinte

//version 2

    for(i = 0; i < 10; i++)
    {
      Serial.print("\t : ");
      Serial.println(i);
      digitalWrite(LED_pin, HIGH);
      Serial.println("LED etat haut");
      delay(TEMPO_High);
      digitalWrite(LED_pin, LOW);
      Serial.println("LED etat bas");
      delay(TEMPO_Low);
    }

}
```

Dans le code ci-dessous nous avons temporiser l'état des leds.

4.1.4. Faire clignoter la Led – Version 3

Led_V3

```
#define LED_pin13 13
#define LED_pin12 12
#define TEMPO_T1 150  // Cas2:15   Cas3: 2
#define TEMPO_T2 1850 // Cas2:185  Cas3: 18

int i;
long int tempo;

void setup() {
    Serial.begin(9600);

    pinMode(LED_pin13, OUTPUT);  //Broche 12 en sortie
    pinMode(LED_pin12, OUTPUT);  //Broche 12 en sortie
    digitalWrite(LED_pin13, LOW);  //Broche 13 LED éteinte
    digitalWrite(LED_pin12, LOW);  //Broche 13 LED éteinte
}

void loop() {
  // put your main code here, to run repeatedly:

      digitalWrite(LED_pin13, HIGH);
      digitalWrite(LED_pin12, HIGH);

      Serial.println("LED 13,12 etat haut");
      delay(TEMPO_T1);

      digitalWrite(LED_pin13, LOW);
      digitalWrite(LED_pin12, LOW);

      Serial.println("LED 13,12 etat bas");
      delay(TEMPO_T2);  //Temporisation

}
```

4.2. Assemblage shield Boe et Arduino

4.2.3.4. Reprise des programmes Led

le programme Led_V3 pour effectuer le même cycle d'allumage / extinction sur la Led connectée sur la broche 12. dans le code ci-dessous : Led_V4.

Led_V4

```
#define LED_pin13 13
#define LED_pin12 12

int i;
long int tempo;

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);

    pinMode(LED_pin12, OUTPUT);  //Broche 12 en sortie
    pinMode(LED_pin13, OUTPUT);  //Broche 12 en sortie

    digitalWrite(LED_pin12, LOW);  //Broche 12 LED étiente
    digitalWrite(LED_pin13, HIGH);  //Broche 13 LED allumée

}

void loop() {
  // put your main code here, to run repeatedly:

      digitalWrite(LED_pin12, HIGH);
      digitalWrite(LED_pin13, LOW);

      Serial.println("LED 12 etat haut");
      delay(1000);

      digitalWrite(LED_pin13, HIGH);
      digitalWrite(LED_pin12, LOW);
      Serial.println("LED 12 etat bas");
      delay(1000);  //Temporisation


}
```

Le code ci-dessous ‘’***Led_V5***’’ permet de piloter chacune des deux leds avec t1 = t2 = 500 ms. Après la compilation du code nous obtenons bien le bon résultat.

Led_V5

```
#define LED_pin13 13
#define LED_pin12 12
#define TEMPO_T 2000  // Cas2:200  Cas3: 20
#define TEMPO_T1 150  // Cas2:15   Cas3: 2
#define TEMPO_T2 1850 // Cas2:185  Cas3: 18

int i;
long int tempo;

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);

    pinMode(LED_pin12, OUTPUT);  //Broche 12 en sortie
    pinMode(LED_pin13, OUTPUT);  //Broche 12 en sortie

    digitalWrite(LED_pin12, LOW);  //Broche 12 LED étiente
    digitalWrite(LED_pin13, HIGH);  //Broche 13 LED allumée
}

void loop() {
      digitalWrite(LED_pin12, HIGH);
      digitalWrite(LED_pin13, LOW);
      Serial.println("LED 12 etat haut");
      delay(TEMPO_T1);

      digitalWrite(LED_pin13, HIGH);
      digitalWrite(LED_pin12, LOW);
      Serial.println("LED 12 etat bas");
      delay(TEMPO_T-TEMPO_T1);  //Temporisation
}
```

4.2.4. Vers la commande des servo moteurs

Nous avons testé les différents cas :

Dans le Cas 2 : On remarque que les leds clignotent plus vite que sur le cas 1

Dans le Cas 3 : On remarque que les leds clignotent plus vite que sur le cas 2

```
#define LED_pin13 13
#define LED_pin12 12
#define TEMPO_T 2000  // Cas2:200  Cas3: 20
#define TEMPO_T1 150  // Cas2:15   Cas3: 2
#define TEMPO_T2 1850 // Cas2:185  Cas3: 18

int i;
long int tempo;

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);

    pinMode(LED_pin12, OUTPUT);  //Broche 12 en sortie
    pinMode(LED_pin13, OUTPUT);  //Broche 12 en sortie

    digitalWrite(LED_pin12, LOW);  //Broche 12 LED étiente
    digitalWrite(LED_pin13, HIGH);  //Broche 13 LED allumée

}


void loop() {
  // put your main code here, to run repeatedly:

      digitalWrite(LED_pin12, HIGH);
      digitalWrite(LED_pin13, LOW);
      Serial.println("LED 12 etat haut");
      delay(TEMPO_T1);

      digitalWrite(LED_pin13, HIGH);
      digitalWrite(LED_pin12, LOW);
      Serial.println("LED 12 etat bas");
      delay(TEMPO_T-TEMPO_T1);  //Temporisation


}
```

4.3. Mise en oeuvre des servo moteurs

Pour le programme ***Servo_V1*** nous avons écrit un programme qui permet de piloter le servo moteur droit, qui est commandé via la broche 11 et le servo moteur gauche via la broche 10. Pour chaque servo moteur, nous avons fixées la durée T1 à 1.5 ms comme ci-dessous :

```ino
#include <Servo.h>

#define TEMPO_T 20  // Cas2:200  Cas3: 20
#define TEMPO_T1 1500  // Cas2:15   Cas3: 2

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;


void setup() {
  // put your setup code here, to run once:

    Servo_gauche.attach (11);
    Servo_droit.attach (10);
    //équilibrer
    Servo_gauche.writeMicroseconds (TEMPO_T1);
    Servo_droit.writeMicroseconds (TEMPO_T1);
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

4.3.2. Mise en place des servo moteurs

4.3.3. Calibration des servo moteurs

Après avoir fait tout les manipulations pour mettre en place les servo moteurs. Nous avons commencé par calibrer les servo moteurs. La calibration consiste à associer la valeur T1 = 1.5 ms que le moteur ne tourne pas (point zéro). Pour cela nous avons défini “**TEMPO_T1**” égale à 1500, et nous avons tourné le potentiomètre jusqu’à ce que les moteurs soit à l’arrêt.

4.3.4. Gérer le sens de rotation des servo moteurs

Nous avons écrit deux programme “**Servo_V2**” et “**Servo_V3**” qui permet de faire tourner le servo moteur d’un côté si T est supérieure à 1500 et de l’autre côté si la valeur est inférieure à 1500, comme ci-dessous :

Code “**Servo_V2**” :

```
#include <Servo.h>

#define TEMPO_T 20  // Cas2:200  Cas3: 20
#define TEMPO_T1 1500  // Cas2:15   Cas3: 2
#define TEMPO_T2 1850 // Cas2:185  Cas3: 18

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;



//
void setup() {
  // put your setup code here, to run once:
    Servo_droit.attach (10);
}

void loop() {
  // put your main code here, to run repeatedly:

    // Inférieur à 1500: il tourne dans un sens
    Servo_droit.writeMicroseconds (1400);
    delay(2000);

    // Supérieur à 1500: il tourne dans l'autre sens
    Servo_droit.writeMicroseconds (1600);
    delay(1000);

}
```

Code “**Servo_V3**” :

```
#include <Servo.h>

#define TEMPO_T 20  // Cas2:200  Cas3: 20
#define TEMPO_T1 1500  // Cas2:15   Cas3: 2
#define TEMPO_T2 1850 // Cas2:185  Cas3: 18

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;


//
void setup() {
  // put your setup code here, to run once:

    Servo_gauche.attach (11);
    Servo_droit.attach (10);
}

void loop() {
  // put your main code here, to run repeatedly:

    //Inférieur à 1500: il tourne dans un sens
    Servo_gauche.writeMicroseconds (1400);

    delay(2000);


     //Supérieur à 1500: il tourne dans l'autre sens
     Servo_gauche.writeMicroseconds (1600);

     delay(1000);

 }
```

4.3.5. Contrôler la vitesse et le sens de rotation des deux servo moteurs

Le programme **Servo _V4** ci-dessous permet de faire tourner les deux servos moteurs dans un sens pour l'un et l'autre sens pour l'autre.

```ino
#include <Servo.h>

#define TEMPO_T 20  // Cas2:200  Cas3: 20
#define TEMPO_T1 1500  // Cas2:15   Cas3: 2
#define TEMPO_T2 1850 // Cas2:185  Cas3: 18

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;


//
void setup() {
  // put your setup code here, to run once:

    Servo_gauche.attach (11);
    Servo_droit.attach (10);
}

void loop() {
  // put your main code here, to run repeatedly:

    //Inférieur à 1500: il tourne dans un sens
    Servo_gauche.writeMicroseconds (1300);
    //Supérieur à 1500: il tourne dans l'autre sens
    Servo_droit.writeMicroseconds (1400);

}
```

Ensuite nous avons fait varier les valeurs de T1 et nous obtenons les résultats suivants :
si > 1500, la roue tourne dans le sens trigonométrique;
si = 1500, la roue arrête,
si < 1500, la roue tourne dans le sens horaire.
Nous avons remplis le tableau suivant :

	Servo Gauche	Servo Droit	Description	Observation
T1 (µs) =	1300	1700	Deux roues tournent dans les mêmes sens	Le robot marche vers l'avant.
	1700	1300	Deux roues tournent dans les mêmes sens	Le robot marche vers l'arrière, il recule.

	1700	1700	Deux roues tournent dans les sens différentes	La voiture robotique tourne à droit autour d'elle
	1300	1300	Deux roues tournent dans les sens différentes	La voiture robotique tourne à gauche autour d'elle
	1500	1500	Servo moteurs à l'arrêt	La voiture robotique ne bouge pas.
	1500	1700	Servo gauche à l'arrêt et le droit tourne	La voiture robotique tourne à droit autour d'elle
	1700	1500	Servo à droit à l'arrêt et le gauche tourne	La voiture robotique tourne à gauche autour d'elle
	1300	1400	Les deux roues roulent dans le même sens différentes	La voiture robotique tourne à gauche autour d'elle
	1400	1300	Les deux roues roulent dans le même sens différentes	La voiture robotique tourne à droit autour d'elle

4.3.6. Contrôler la durée d'activité des servo moteurs

On a édité le code comme ci-dessous, pour permettre au robot de tourner sur lui même à droite pendant 3 secondes et ensuite à gauche pendant 2 secondes. Entre chaque étape du cycle,nous avons insérer une temporisation avec la fonction **delay** comme ci-dessous :

SERVO_V5 §

```
#include <Servo.h>
//Définition de constantes symboliques
#define SERVO_GAUCHE_PIN 11
#define SERVO_DROIT_PIN 10
#define T1_1 1300
#define T1_2 1700
#define T1_3 1500

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;

void setup()
{
  Servo_droit.attach(SERVO_DROIT_PIN);
  Servo_gauche.attach(SERVO_GAUCHE_PIN);

  //1. Faire tourner les deux servo moteurs dans le sens horaire pendant 3s ;
  Servo_droit.writeMicroseconds (T1_1);
  Servo_gauche.writeMicroseconds (T1_1);
  delay (3000);
  //2. Faire tourner les deux servo moteurs dans le sens trigonométrique pendant 2.5s
  Servo_droit.writeMicroseconds (T1_2);
  Servo_gauche.writeMicroseconds (T1_2);
  //3. Arrêter les deux servo moteurs.
  delay (2500);

//  Servo_droit.writeMicroseconds (T1_3);
//  Servo_gauche.writeMicroseconds (T1_3);
    Servo_droit.detach();
    Servo_gauche.detach();
}
```

5. Assemblage du robot

Nous avons les différentes instructions pour assembler notre robot.

5.3. Mise en place de la carte Arduino et de la Board of Education

le programme **Buzzer_V1** permet de générer un son, à la fréquence de 3000Hz, pendant 2s ensuite on fait un arrêt d'une seconde et on reprend la génération du même son.

Buzzer_V1

```
#include <Servo.h>

//Définition de constantes symboliques
#define BUZZER_PIN 4

void setup()
{
  //générer le son par la fonction "tone()"
  tone(BUZZER_PIN, 3000);
  delay(2000);

  //arrêter le son par la fonction "noTone()"
  noTone(BUZZER_PIN);
  delay(1000);
}

void loop() {
```

Question supplémentaire : générer une mélodie.

On a choisi une mélodie de "Pirates of the Caribbean", et pour générer des notes nous avons utilisés le lien suivant : https://github.com/xitangg/-Pirates-of-the-Caribbean-Theme-Song

Voici à quoi correspond les lettre dans notre mélodie chaque lettre a une fréquence et une période comme suit :

The calculation of the tones is made following the mathematical operation:

```
timeHigh = period / 2 = 1 / (2 * toneFrequency)
```

where the different tones are described as in the table:

```
note    frequency    period    timeHigh
c       261 Hz       3830       1915
d       294 Hz       3400       1700
e       329 Hz       3038       1519
f       349 Hz       2864       1432
g       392 Hz       2550       1275
a       440 Hz       2272       1136
b       493 Hz       2028       1014
C       523 Hz       1912        956
```

Buzzer_V2_melody2

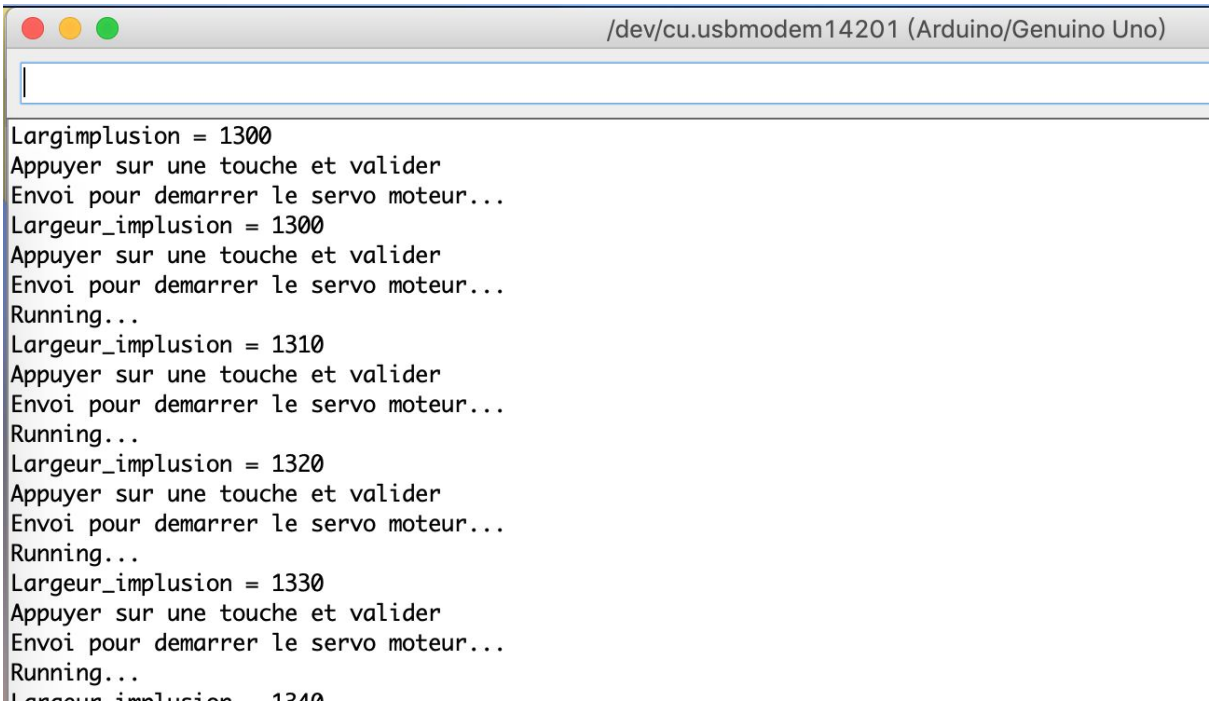
```
#include <Servo.h>
//#define BUZZER_PIN_PIN 4

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;

const int BUZZER_PIN = 4;
//songspeed: Si on met un chiffre plus grand, c'est pour l'accélérer
const int songspeed = 1.5;
//****************************************
#define NOTE_C4  262   //Définir la fréquence
#define NOTE_D4  294
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_G4  392
#define NOTE_A4  440
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_D5  587
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_G5  784
#define NOTE_A5  880
#define NOTE_B5  988
//****************************************
int notes[] = {
   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,

   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,

   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_C5, NOTE_D5, NOTE_D5, 0,
   NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, 0,
   NOTE_E5, NOTE_D5, NOTE_E5, NOTE_A4, 0,

   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_D5, NOTE_E5, NOTE_A4, 0,
   NOTE_A4, NOTE_C5, NOTE_B4, NOTE_B4, 0,
   NOTE_C5, NOTE_A4, NOTE_B4, 0,

   NOTE_A4, NOTE_A4,

   //Répéter la première partie
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,

   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,
```

```
int duration[] = {         //Durée de chaque intonation (ms)
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,

  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,

  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 125, 250, 125,

  125, 125, 250, 125, 125,
  250, 125, 250, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 375,

  250, 125,
  //Rpeat of First Part
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,
```

```
void setup() {
  for (int i=0;i<203;i++){              //203: intonations totales pour cette chanson
  int wait = duration[i] * songspeed;
  tone(BUZZER_PIN,notes[i],wait);       //tone(pin,frequency,duration)
  delay(wait);}
}

void loop() {

}
```

5.5. Courbe de variation de vitesse de rotation des servo moteurs

Mesures de vitesse de rotation : Les valeurs ci-dessous correspond au mesure faite avec le câble nous avons corrigé cela par la suite.

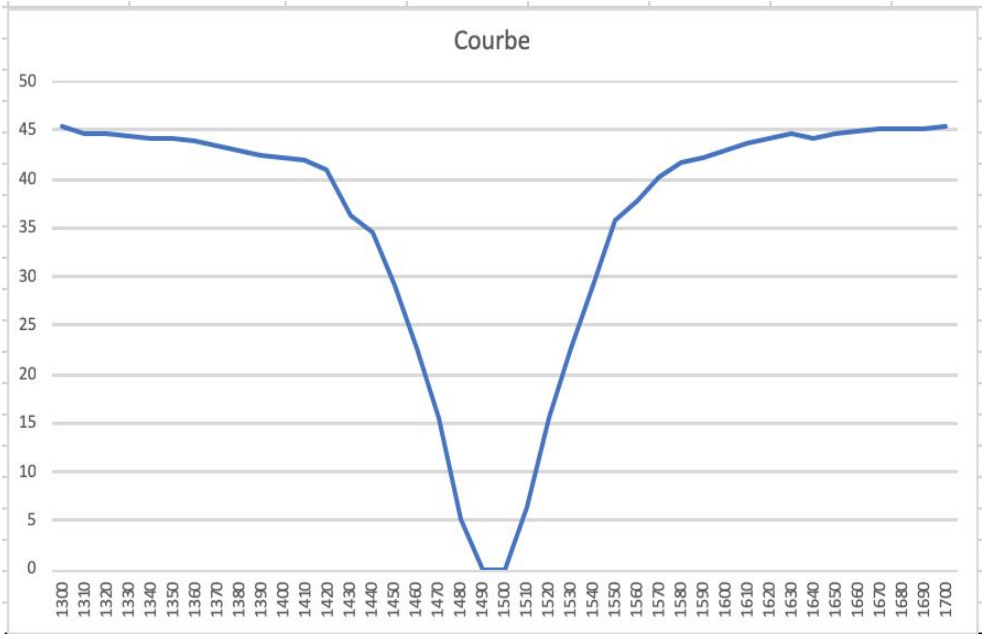


Largeur d'impulsion en µs	Vitesse en tours par minute	Largeur d'impulsion en µs	Vitesse en tours par minute	Largeur d'impulsion en µs	Vitesse en tours par minute	Largeur d'impulsion en µs	Vitesse en tours par minute
1300	35	1400	32.5	1500	0	1600	33
1310	34.5	1410	32.4	1510	5(autre sens)	1610	34
1320	34.5	1420	31.5	1520	12	1620	34.3
1330	34.2	1430	28	1530	17.5	1630	34.5
1340	34	1440	26.6	1540	22.5	1640	34
1350	34	1450	22.5	1550	26.6	1650	34.4
1360	33.8	1460	17.5	1560	28	1660	34.6
1370	33.5	1470	12	1570	31	1670	34.7
1380	33	1480	4	1580	32.2	1680	34.8
1390	32.7	1490	0	1590	32.5	1690	34.8
						1700	35

Comme nous avons constaté que, lors de l'utilisation de la batterie et du câble, la vitesse de la roue est différente, nous avons effectué deux tests séparément. Et nous avons calculé la vitesse (en cm/min et en m/s à la fin) pour utiliser. Comme indiqué dans le tableau ci-dessous :

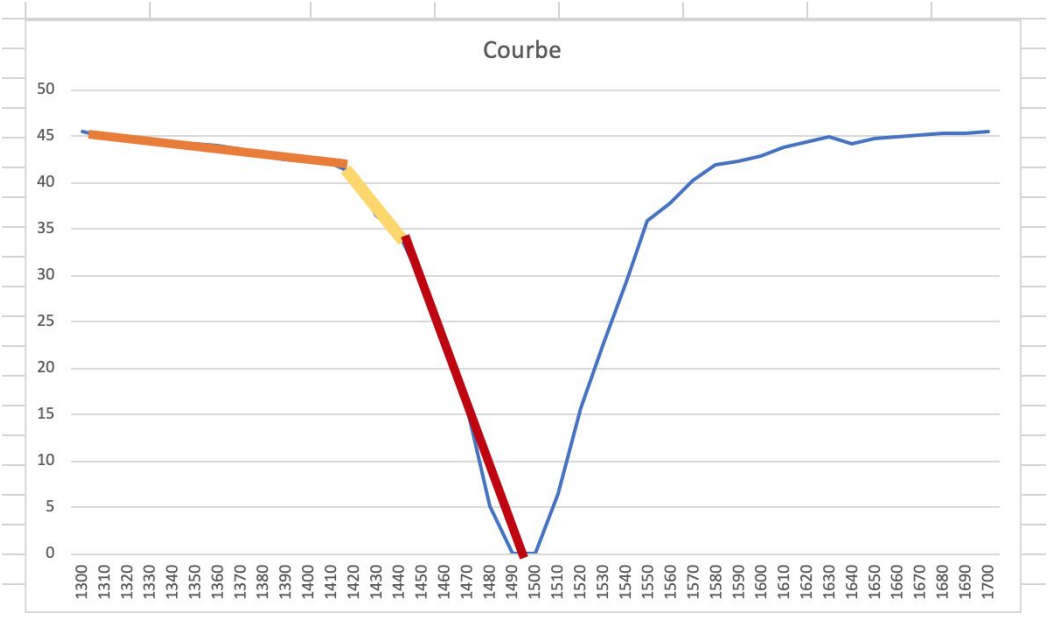
		avec batterie		avec câble	
	v: m/s	**vitesse: tours/min**	**impulsion**	**vitesse: to**	**V: cm/min**
	0.20954267	45.5	1300	35	1257.256
	0.2065492	44.85	1310	34.5	1239.2952
	0.2065492	44.85	1320	34.5	1239.2952
	0.20475312	44.46	1330	34.2	1228.51872
	0.20355573	44.2	1340	34	1221.3344
	0.20355573	44.2	1350	34	1221.3344
	0.20235835	43.94	1360	33.8	1214.15008
	0.20056227	43.55	1370	33.5	1203.3736
	0.1975688	42.9	1380	33	1185.4128
	0.19577272	42.51	1390	32.7	1174.63632
	0.19457533	42.25	1400	32.5	1167.452
	0.19397664	42.12	1410	32.4	1163.85984
	0.1885884	40.95	1420	31.5	1131.5304
	0.16763413	36.4	1430	28	1005.8048
	0.15925243	34.58	1440	26.6	955.51456
	0.134706	29.25	1450	22.5	808.236
	0.10477133	22.75	1460	17.5	628.628
	0.0718432	15.6	1470	12	431.0592
	0.02394773	5.2	1480	4	143.6864
	0	0	1490	0	0
	0	0	1500	0	0
	0.02993467	6.5	1510	5	179.608
	0.0718432	15.6	1520	12	431.0592
	0.10477133	22.75	1530	17.5	628.628
	0.134706	29.25	1540	22.5	808.236
	0.16523936	35.88	1550	27.6	991.43616
	0.17362107	37.7	1560	29	1041.7264
	0.18559493	40.3	1570	31	1113.5696
	0.19277925	41.86	1580	32.2	1156.67552
	0.19457533	42.25	1590	32.5	1167.452
	0.1975688	42.9	1600	33	1185.4128
	0.20175965	43.81	1610	33.7	1210.55792
	0.20415443	44.33	1620	34.1	1224.92656
	0.2065492	44.85	1630	34.5	1239.2952
	0.20355573	44.2	1640	34	1221.3344
	0.20595051	44.72	1650	34.4	1235.70304
	0.20714789	44.98	1660	34.6	1242.88736
	0.20774659	45.11	1670	34.7	1246.47952
	0.20834528	45.24	1680	34.8	1250.07168
	0.20834528	45.24	1690	34.8	1250.07168
	0.20954267	45.5	1700	35	1257.256

Avec des données que nous avons mesurées, nous avons obtenu la courbe de variation de vitesse, se présente ainsi :



Nous pouvons voir que la vitesse de la roue change doucement au début, puis a ralenti rapidement vers impulsion = 1420.
Lorsque la roue s'arrête à impulsion = 1500, le sens change et le changement est identique à celui de la première à gauche, qui est symétrique.

Modélisation :
Nous avons fait la modélisation pour la courbe comme indiqué dans l'image ci-dessous:



Voici le résultat de modélisation:
x, correspond à l'impulsion

f（x）= -0.0255 * X + 68.255 taux d'erreur：0.96**,** si 1300 ≤x ≤1420
f（x）= -0.14 * X + 228.2 taux d'erreur：1.00, si 1430 ≤ x ≤ 1440
f（x）= -0.494 * X + 738.4 taux d'erreur：0.95 si 1450 ≤ x ≤1500

La modélisation mathématique du côté droit :

f（x）= 0.56 * X - 839.9 taux d'erreur：0.98**,** si 1510 ≤x ≤1550
f（x）= 0.2 * X - 283 taux d'erreur：1.00 si 1560 ≤ x ≤1570
f（x）= 0.022063 * X + 2.2192 taux d'erreur：0.89, si 1580 ≤ x ≤ 1700

Il y a plusieurs façons pour la modélisation, ce qu'on a fait (Fonction par morceaux) n'est pas la meilleur, mais ça marche bien aussi.

6. Navigation du robot

6.1 Déplacement

6.1.1 Déplacement avant

Nous avons créé une fonction "Avance" qui prend en paramètre la durée du déplacement.
Pour la roue de droite, impulsion 1300 il tourne dans le sens horaire avec la vitesse maximale, donc il avance.
Pour la roue de gauche, impulsion 1700 il tourne dans le sens trigonométrique avec la vitesse maximale, donc il avancer dans le même sens et avec la même vitesse que la roue de droite.

```
//fonction: avance
void Avance (int duree_deplacement)
{
  Servo_droit.writeMicroseconds(1300);
  Servo_gauche.writeMicroseconds(1700);
  delay (duree_deplacement);
}


//fonction: arret
void arret (void)
{
  Servo_droit.detach ();
  Servo_gauche.detach ();
}
```

Avance_V1 §

```
#include <Servo.h>
//Définition de constantes symboliques
#define BUZZER_PIN 4
#define SERVO_DROIT_PIN 11
#define SERVO_GAUCHE_PIN 10
#define DUREE 6000

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;

int largeur_implusion;

void setup()
{
  //Préparer le son pour l'avertissement
  tone(BUZZER_PIN, 3000);
  delay(1000);
  noTone(BUZZER_PIN);

  Serial.begin(9600);
  //L'état début, deux en arrêt
  Servo_gauche.attach(SERVO_GAUCHE_PIN);
  Servo_gauche.writeMicroseconds (1500);
  Servo_droit.attach(SERVO_DROIT_PIN);
  Servo_droit.writeMicroseconds (1500);

  //Avancer pendant 3s
  Avance(3000);

  arret();

}
```

6.1.2 Déplacement en arrière :

Nous avons créé une fonction “**recule”** qui permet au robot de se déplacer en arrière Comme ci-dessous :

Recule_V1

```
#include <Servo.h>

//Définition de constantes symboliques
#define BUZZER_PIN 4
#define SERVO_DROIT_PIN 11
#define SERVO_GAUCHE_PIN 10
#define DUREE 6000

// Variables globales
Servo Servo_droit;
Servo Servo_gauche;

int largeur_implusion;

void setup()
{
  //Préparer le son pour l'avertissement
  tone(BUZZER_PIN, 3000);
  delay(1000);
  noTone(BUZZER_PIN);

  Serial.begin(9600);
  //L'état début, deux en arrêt
  Servo_gauche.attach(SERVO_GAUCHE_PIN);
  Servo_gauche.writeMicroseconds (1500);
  Servo_droit.attach(SERVO_DROIT_PIN);
  Servo_droit.writeMicroseconds (1500);

  //Recule pendant 3s
  Recule(3000);
  arret();

}


//fonction: recule
void Recule (int duree_deplacement)
{
  Servo_droit.writeMicroseconds(1700);
  Servo_gauche.writeMicroseconds(1300);
  delay (duree_deplacement);
}
```

6.1.3 Déplacement pivoter à droite

Le fichier correspondant est “**Pivoter_Droit_V1**” joint au rapport

```
void Droite (int duree_deplacement)
{
  Servo_droit.writeMicroseconds(1700);
  Servo_gauche.writeMicroseconds(1700);
  delay (duree_deplacement);
}
```

6.1.4 Déplacement pivoter à gauche

Le fichier correspondant est “**Pivoter_Gauche_V1**” joint au rapport la fonction se présente ainsi cette fonction a été appelée dans la fonction setup()

```
void Gauche (int duree_deplacement)
{
  Servo_droit.writeMicroseconds(1300);
  Servo_gauche.writeMicroseconds(1300);
  delay (duree_deplacement);
}
```

6.3. Pivoter et tourner :

- Avancer et tourner à droite
La fonction qui permet d’avancer droite :

```
void Avance_Droite(int duree_deplacement){
  Servo_droit.writeMicroseconds(1450);
  Servo_gauche.writeMicroseconds(1700);
  delay (duree_deplacement);
}
```

- Avancer et tourner à gauche

```
void Avance_Gauche(int duree_deplacement){
  Servo_droit.writeMicroseconds(1300);
  Servo_gauche.writeMicroseconds(1550);
  delay (duree_deplacement);
}
```

- Reculer et tourner à droite

```
void Recule_Droite(int duree_deplacement){
  Servo_droit.writeMicroseconds(1550);
  Servo_gauche.writeMicroseconds(1300);
  delay (duree_deplacement);
}
```

- Reculer et tourner à gauche

```
void Recule_Gauche(int duree_deplacement){
  Servo_droit.writeMicroseconds(1700);
  Servo_gauche.writeMicroseconds(1450);
  delay (duree_deplacement);
}
```

6.4. Gérer les déplacements avec des fonctions :

6.4.1. Retour sur l'écriture d'une fonction

```
void Ma_fonction(int duree_deplacement){
  //TODO
}
```

6.4.2. Fonctions des déplacements de base

Voir partie précédente 6.4.1

7. Navigation et détection d'obstacles

7.2. Principe de fonctionnement des moustaches

Maintenant que nous avons les différentes fonctions de déplacement, nous allons nous intéresser à la détection des obstacles pour cela nous avons utilisé les moustaches . Une fois les moustaches assemblées, nous avons câbler le circuit qui permettra de recueillir le signal issu de chaque moustache.

7.3. Test de l'état des moustaches

```
void Ma_fonction(){

  //Définir les pins pour Input: deux moustaches
  pinMode (5, INPUT); // Pour la moustache gauche
  pinMode (7, INPUT); // Pour la moustache droite

  //Lire des valeurs obtenues
  val_mGauche = digitalRead(5);
  val_mDroite = digitalRead(7);

  //Imprimer des valeurs dans console.
  Serial.print("Gauche:");
  Serial.println(val_mGauche);
  Serial.print("Droite:");
  Serial.println(val_mDroite);

}
```

Nous avons créé deux variables (**val_mGauche** et **val_mDroite**) pour stocker la valeurs retourné.

Si la moustache est pressée, le voltage sera 0.

Après exécution nous obtenons “**gauche:1”** si la moustache gauche n’est pas pressé et “**gauche:0”** s’il n’est pas pressé de même chose pour la moustache droite. La lecture de l’état de la moustache gauche est effectué sur la broche 5 et pour la moustache droite, la lecture de son état s’effectue sur la broche 7.

Pour lire l’état de chacune de ces broches, nous avons utilisé la fonction ***digitalRead.*** Cette fonction renvoie la valeur lue sur la broche, soit **HIGH** ou **LOW**. LOW quand la moustache est pressée, et HIGH sinon. Ensuite on stocke cette valeur dans les variables créé précédemment.

7.4. Visualiser l’état des moustaches avec les Leds

Pour pouvoir identifier la moustache qui est pressée lorsque le robot rencontre un obstacle, nous avons associer aux moustache deux leds qui indique si les moustaches sont pressées au pas.

```
void loop() {
  delay(100);
  Ma_fonction();
  delay(100);
}
```

```
void Ma_fonction(){

  pinMode (5, INPUT); // Pour la moustache gauche
  pinMode (7, INPUT); // Pour la moustache droite
  pinMode(LED_GAUCHE_PIN, OUTPUT);
  pinMode(LED_DROITE_PIN,OUTPUT);
  val_mGauche = digitalRead(5);
  val_mDroite = digitalRead(7);

  Serial.print("Gauche:");
  Serial.println(val_mGauche);
  Serial.print("Droite:");
  Serial.println(val_mDroite);

  //Allumer les deux LED correspondantes
  digitalWrite(LED_GAUCHE_PIN, 1-val_mGauche);
  digitalWrite(LED_DROITE_PIN, 1-val_mDroite);
```

Si la moustache est pressé, le voltage est LOW(0), il faut allumer la led. Donc on écrit HIGH pour la led correspondante.

7.5. Déplacement du robot et gestion des moustaches

Moustaches_V3 §

```
//Définition de constantes symboliques
#define BUZZER_PIN 4
#define SERVO_DROITE_PIN 11
#define SERVO_GAUCHE_PIN 10
#define LED_GAUCHE_PIN 8
#define LED_DROITE_PIN 13
int val_mGauche = 0;
int val_mDroite = 0;

#define DUREE 6000

// Variables globales
Servo Servo_droite;
Servo Servo_gauche;

int largeur_implusion;

void setup()
{
  tone(BUZZER_PIN, 3000);
  delay(1000);
  noTone(BUZZER_PIN);

  Serial.begin(9600);
  Servo_gauche.attach(SERVO_GAUCHE_PIN);
  Servo_gauche.writeMicroseconds (1500);
  Servo_droite.attach(SERVO_DROITE_PIN);
  Servo_droite.writeMicroseconds (1500);
//  arret();
}

void loop() {
  delay(100);
  Ma_fonction();
  delay(100);
}
```

D'abord, on a fait "attach()" pour que les deux roues puissent marcher.
Après dans le loop(), on a définit un petit interval : delay(100), pour que le robot puisse agir rapidement.

```
void Ma_fonction(){

  pinMode (5, INPUT); // Pour la moustache gauche
  pinMode (7, INPUT); // Pour la moustache droite
  pinMode(LED_GAUCHE_PIN, OUTPUT);
  pinMode(LED_DROITE_PIN,OUTPUT);
  val_mGauche = digitalRead(5);
  val_mDroite = digitalRead(7);

  digitalWrite(LED_GAUCHE_PIN, 1-val_mGauche);
  digitalWrite(LED_DROITE_PIN, 1-val_mDroite);

  //Définir des actions selon l'état de moustache
  if(val_mGauche == 1 && val_mDroite == 1){
    Avance();
  }
  else if(val_mGauche == 0 && val_mDroite == 1){
    Recule_Droite();

  }
  else if(val_mDroite == 0 && val_mGauche == 1){
    Recule_Gauche();
  }
  else if(val_mDroite == 0 && val_mGauche == 0){
    Recule();
  }
}
```

Dans cette partie, on a définit des réactions pour toutes les situations différentes, si le robot rencontre un obstacle en voyageant, il donnera la réponse correcte selon le jugement ci-dessus.

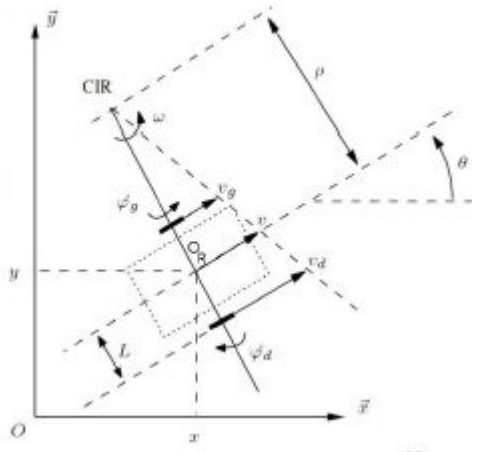
Déplacement du robot :

Comme le montre l'image ci-dessous :

- 2.1.4. Classes de robots mobiles

- Robot de type unicycle : modélisation

O_R : position du robot
θ : orientation du robot
L : entre-axe ⇔ longueur ½ essieu
ω : vitesse de rotation du robot autour de CIR
ρ : rayon de courbure de la trajectoire
v : vecteur vitesse au point O_R
v_g : vecteur vitesse roue gauche
v_d : vecteur vitesse roue droite
$\dot{\varphi}_g$: vitesse de rotation roue gauche
$\dot{\varphi}_d$: vitesse de rotation roue droite
CIR : Centre Instantané de Rotation



Pour calculer le degré de rotation nous avons utilisé la formule ci-dessous :

$$\rho = L\frac{v_d+vg}{v_d-vg}$$

donc Vg = $\left(\frac{(\rho-L)\ Vd}{L+\rho}\right)$

donc Vd = $\left(\frac{(\rho+L)\ Vg}{\rho-L}\right)$

avec :

Vd : vecteur vitesse roue droite.
Vg : vecteur vitesse roue gauche.
L : la distance entre les deux roues divisé par 2.
ρ : rayon de courbure de la trajectoire.

Et aussi des connaissances physiques:

	ρ = v/w	
	w = v/**ρ**	
	v = (vd+vg)/2	
	(degre/ 180)*Pi = w*t	
1 rad = 57.3°	t = degre /(57.3 * w)	ou t = (degre/180)*Pi / w

- Nous avons d'abord mesuré le diamètre de la roue: **d = 8.8cm**, et la moitié de la distance entre deux roues: **L = 5.6cm**.

- Après nous avons calculé le vecteur vitesse de la roue de droite et de la roue de gauche, pour cela utilisé la formule suivante pour calculer le périmètre :

  $$c = \pi * d\ (diamètre\ de\ la\ roue)$$

  Avec c(le périmètre), nous avons obtenu la distance parcourue par ce robot dans chaque minute: **v(vd ou vg) = distance cm/min = c * nb de tours/min**. Ensuite, nous les avions transfert en unité standard internationale: **m/s**

- Avec vd , vg et L, nous pouvons calculer le rayon de rotation: $\rho = L\frac{v_d+vg}{v_d-vg}$;

- Avec vd, vg, nous avons aussi la vitesse linéaire: **v = (vd+vg)/2 (m/s)**;

- Avec v, on a la vitesse angulaire: **w = v/ρ (rad/s)**

- Ensuite, nous pouvons déterminer le temps nécessaire pour que la voiture tourne sous un angle donné.

  Car **(degré/180)*Pi = w*t** , alors: **t = (degré/180)*Pi / w**

Ou simplement, on peut penser à 1 rad = 57.3 degrés, donc :

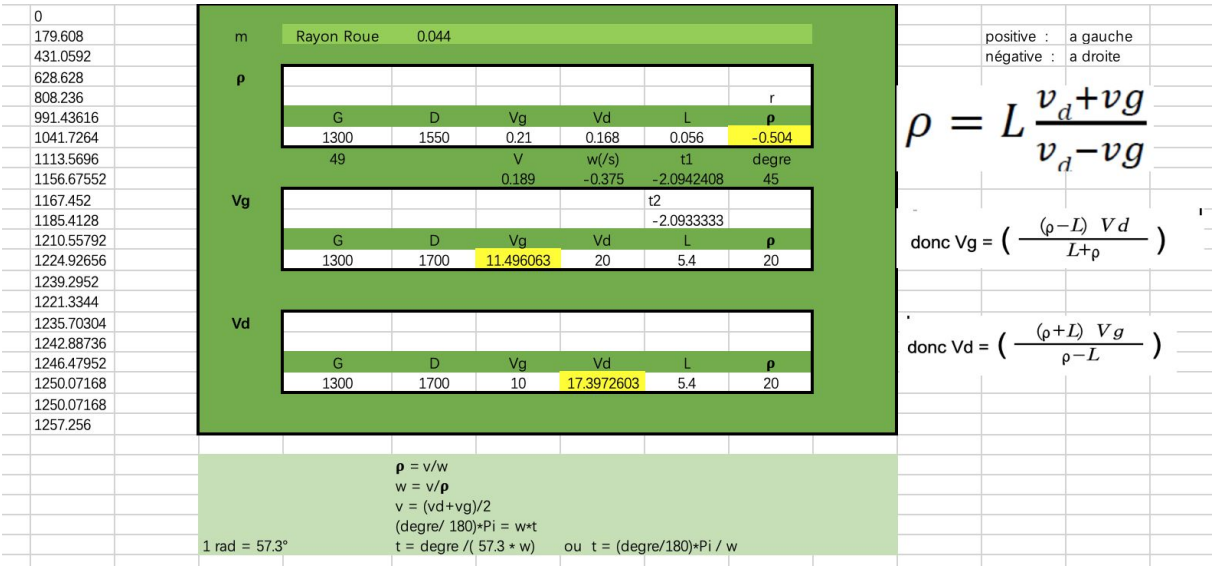
t = degré /(57.3 * w).

Nous avons créé un fichier excel pour calculer les valeurs, c'est plus pratique on a créé trois calculatrices avec excel:

calculatrice 1: Étant donné **vd, vg** on peut calculer **ρ** et **t**;
calculatrice 2: Étant donné **vd**, **ρ**, on peut calculer **vg** et **t**;
calculatrice 3: Étant donné **vg**, **ρ** on peut calculer **vd** et **t**.

Les calculs qui donne les valeurs négatif correspond à droite et les valeurs positives correspond à gauche.

Nous avons joint ce fichier à notre rapport.



Donc nous avons ajouté une méthode pour convertir l'angle au temps requis pour la rotation.

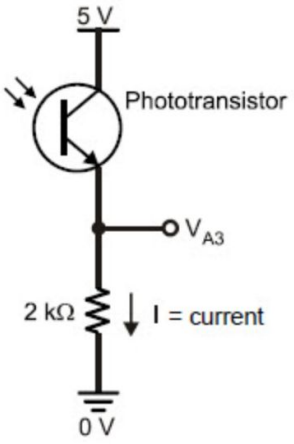
```
float cal_degree(float v_d, float v_g, float degre){

  float v = (v_d + v_g) /2; //Vitesse de robot: m/s
  float p = L * (v_d + v_g) / (v_d - v_g); //Rayon de tourne: m
  float w = v/p; //Vitesse angulaire: rad/s
  float t = (degre * PI) / (180 * w ) ; // temps de tourne : s
  return t;
}
```

Pour une vitesse donné pour la roue de gauche et la roue de droite et un degré cette fonctionne retourne le temps nécessaire.

```
//Définir des actions selon l'état de moustache
if(val_mGauche == 1 && val_mDroite == 1){
  Avance();
}
else if(val_mGauche == 0 && val_mDroite == 1){
  Recule_Droite();
  delay(cal_degree(45));

}
```

8. Navigation et capteurs de lumière

8.1. Description du capteur utilisé



La tension sur A3 (VA3) varie en fonction de la lumière, car le phototransistor laisse passer plus de courant lorsque plus de lumière brille sur celui-ci, ou moins de courant avec moins de lumière. Ce courant, appelé I dans le circuit ci-dessous, doit également Lorsque moins de courant passe dans une résistance, la tension qui la traverse augmente. Si moins de courant passe, elle diminue, puisqu'une extrémité de la résistance est liée à Vss = 0 V, la tension à la VA3 fin monte avec plus de courant et vers le bas avec moins de courant.

8.2. Mise en oeuvre sur le robot

Phototransistor_V1 §

```
#include <Servo.h>
//Définition de constantes symboliques
#define DUREE 6000
#define PHOTO_TRANSISTOR 3

float D_To_A(int V_num);

void setup()
{
  Serial.begin(9600);
}

void loop() {

  int V_num_A3;
  //Lire la valeur numérique correspondant la tension de resistence
  V_num_A3 = analogRead(PHOTO_TRANSISTOR);

  //Print la valeur numérique
  Serial.print("A3:");
  Serial.print(V_num_A3);

  //Print la valeur de la tension de resistence
  Serial.print("  A3:  ");
  Serial.print(D_To_A(V_num_A3));
  Serial.println("Volts");

  delay(1000);
}


float D_To_A(int V_num){
  return (float) V_num * 5.0 /1024.0;
}
```

```
A3:303  A3:  1.48Volts
A3:290  A3:  1.42Volts
A3:276  A3:  1.35Volts
A3:256  A3:  1.25Volts
A3:163  A3:  0.80Volts
A3:88  A3:  0.43Volts
A3:129  A3:  0.63Volts
A3:158  A3:  0.77Volts
A3:146  A3:  0.71Volts
A3:139  A3:  0.68Volts
A3:127  A3:  0.62Volts
A3:79  A3:  0.39Volts
A3:90  A3:  0.44Volts
A3:92  A3:  0.45Volts
A3:314  A3:  1.53Volts
A3:317  A3:  1.55Volts
A3:317  A3:  1.55Volts
A3:318  A3:  1.55Volts
A3:318  A3:  1.55Volts
A3:312  A3:  1.52Volts
A3:319  A3:  1.56Volts
A3:334  A3:  1.63Volts
A3:349  A3:  1.70Volts
A3:803  A3:  3.92Volts
A3:821  A3:  4.01Volts
A3:820  A3:  4.00Volts
A3:818  A3:  3.99Volts
A3:819  A3:  4.00Volts
```

Nous avons essayé avec une résistance de 470Ω, 1030Ω et aussi 5621Ω, mais 2040 est toujours la meilleur.

8.3. Gestion des déplacements – Approche 1

Dans cette partie nous avons contrôler le déplacement du robot avec la lumière ambiante. Si la luminosité est faible il continue à avancer dès lors que la luminosité est importante le robot s'immobilise. En fonctionnant la luminosité de la salle nous adaptant le seuil, voici le code correspondant :

```
#include <Servo.h>
//Définition de constantes symboliques
#define BUZZER_PIN 4
#define DUREE 6000
#define SERVO_DROITE_PIN 11
#define SERVO_GAUCHE_PIN 10
#define PHOTO_TRANSISTOR 3
//On a définit un bon seuil voltage numérique = 700
#define LUMINOSITE_SEUIL 700
```

Nous avons utilisé A3 pour capturer la voltage de la résistance.

Après les tests précédents, nous avons constaté que en générale, la valeur de voltage numérique est sous 500, la valeur médiane de la valeur de voltage numérique qui est influencée par l'intensité lumineuse était d'environ 700. Donc on mis le seuil égale 700.

```
void loop() {
  int V_num_A3;
  V_num_A3 = analogRead(PHOTO_TRANSISTOR);
  Serial.print("A3 : ");
  Serial.print(V_num_A3);
  Serial.print("  A3  :  ");
  Serial.print(D_To_A(V_num_A3));
  Serial.println("Volts");
  my_fonction(V_num_A3);

}
```

Nous obtenons la voltage numérique dans premier temps comme avant, après on appelle notre méthode ici pour faire agir le phototransistor.

```
//Si la luminosité est faible, le robot continue à avancer;
//Si la luminosité est importante, le robot s'immobilise.
void my_fonction(int V_num){

  if (V_num <= LUMINOSITE_SEUIL){ //Sombre
    Avance();
  }
  else{
    arret2();
//      Ralentir();
  }
}
```

Voici notre méthode pour indiquer le robot quand il doit avancer et s'arrêter.

```
void arret (void)
{
  Servo_droite.detach ();
  Servo_gauche.detach ();
}

void arret2 (void)
{
  Servo_droite.writeMicroseconds(1500);
  Servo_gauche.writeMicroseconds(1500);
}
```

Il y a une notation importante: nous avons deux méthodes pour faire arrêter le robot:

arret() utilisé pour l'arrêter complètement,
arret2() utilisé pour l'arrêter actuellement.

Stratégie de déplacement est défini comme suit :

Définition de variables:

Robot_final

```
#include <Servo.h>

//Définition de constantes symboliques
#define BUZZER_PIN 4
#define SERVO_DROITE_PIN 11
#define SERVO_GAUCHE_PIN 10
#define LED_GAUCHE_PIN 8
#define LED_DROITE_PIN 13

#define PHOTO_TRANSISTOR 3
#define LUMINOSITE_SEUIL 700

int val_mGauche = 0;
int val_mDroite = 0;

#define DUREE 6000

// Variables globales
Servo Servo_droite;
Servo Servo_gauche;
```

Initialisation :

```
void setup()
{
  tone(BUZZER_PIN, 3000);
  delay(1000);
  noTone(BUZZER_PIN);

  Serial.begin(9600);
  Servo_gauche.attach(SERVO_GAUCHE_PIN);
  Servo_gauche.writeMicroseconds (1500);
  Servo_droite.attach(SERVO_DROITE_PIN);
  Servo_droite.writeMicroseconds (1500);
//  arret();
}
```

```
void loop() {

  int V_num_A3;
  V_num_A3 = analogRead(PHOTO_TRANSISTOR);
  Serial.print("A3 : ");
  Serial.print(V_num_A3);
  Serial.print("  A3  :  ");
  Serial.print(D_To_A(V_num_A3));
  Serial.println("Volts");

  delay(100);
  Ma_fonction(V_num_A3);
  delay(100);
}


void Avance ()
{
  Servo_droite.writeMicroseconds(1300);
  Servo_gauche.writeMicroseconds(1700);
//  delay (duree_deplacement);
}

void arret (void)
{
  Servo_droite.detach ();
  Servo_gauche.detach ();
}


void arret2 (void)
{
  Servo_droite.writeMicroseconds(1500);
  Servo_gauche.writeMicroseconds(1500);
}

void Recule ()
{
  Servo_droite.writeMicroseconds(1700);
  Servo_gauche.writeMicroseconds(1300);
//  delay (duree_deplacement);
}

void Gauche (int duree_deplacement)
{
  Servo_droite.writeMicroseconds(1300);
  Servo_gauche.writeMicroseconds(1300);
  delay (duree_deplacement);
}
```

```
void Droite (int duree_deplacement)
{
  Servo_droite.writeMicroseconds(1700);
  Servo_gauche.writeMicroseconds(1700);
  delay (duree_deplacement);
}

void Avance_Droite(int duree_deplacement){
  Servo_droite.writeMicroseconds(1450);
  Servo_gauche.writeMicroseconds(1700);
  delay (duree_deplacement);
}

void Avance_Gauche(int duree_deplacement){
  Servo_droite.writeMicroseconds(1300);
  Servo_gauche.writeMicroseconds(1550);
  delay (duree_deplacement);
}

void Recule_Gauche(){
  Servo_droite.writeMicroseconds(1700);
  Servo_gauche.writeMicroseconds(1450);
//  delay (duree_deplacement);
}

void Recule_Droite(){
  Servo_droite.writeMicroseconds(1550);
  Servo_gauche.writeMicroseconds(1300);
//  delay (duree_deplacement);
}
```

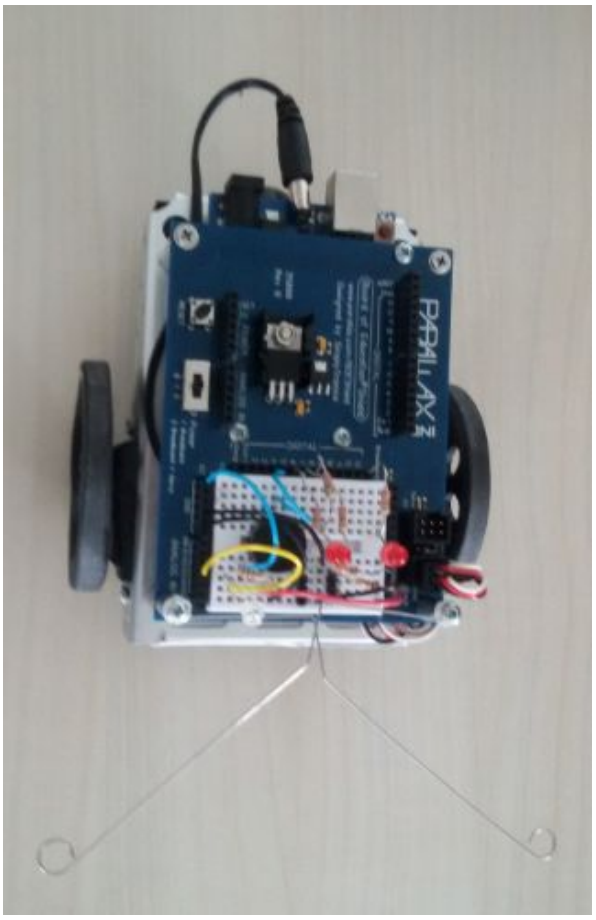
```
void Strategie_robot(int V_num){

  val_mGauche = digitalRead(5);
  val_mDroite = digitalRead(7);
  digitalWrite(LED_GAUCHE_PIN, 1-val_mGauche);
  digitalWrite(LED_DROITE_PIN, 1-val_mDroite);

  //Définir des actions selon l'état de moustache

  //Si la luminosité est faible, le robot continue à avancer;
  //Si la luminosité est importante, le robot s'immobilise.
  if (V_num >= LUMINOSITE_SEUIL){ //Clair
    Ralentir();
    arret2();
    Tourner(CHANGEMENT_DIRECTION);
  }
  else{

      if(val_mGauche == 1 && val_mDroite == 1 ){
        Avance();
      }
      else if(val_mGauche == 0 && val_mDroite == 1){
        Recule_Droite();
        //degree: 360
        delay(10500);
        Tourner(CHANGEMENT_DIRECTION);
      }
      else if(val_mDroite == 0 && val_mGauche == 1){
        Recule_Gauche();
        //degree: 30
        delay(1000);
      }
      else if(val_mDroite == 0 && val_mGauche == 0){
      Recule();
      delay(2000);
      Tourner(CHANGEMENT_DIRECTION);
      }
  }
```

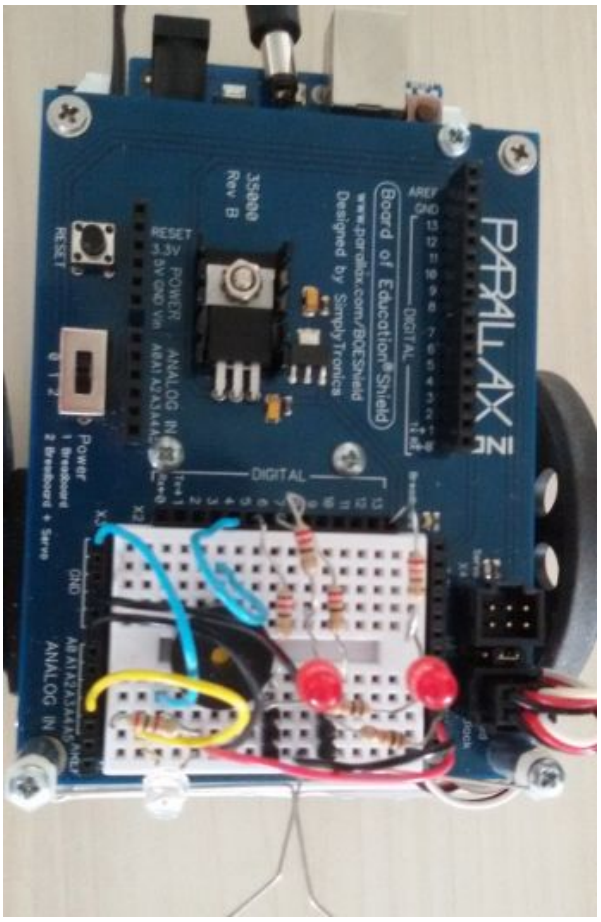
```
float D_To_A(int V_num){
  return (float) V_num * 5.0 /1024.0;
}
```

La stratégies de déplacement du robot correspond à ce qui suit :
Dans un premier temps, le robot avance tout droit, si la luminosité est faible, le robot continue à avancer; si la luminosité est importante, le robot s'immobilise.
Si les deux moustaches sont pressées en même temps, le robot recule pendant 2 secondes et après fait demi-tours tout en tournant sur lui même.
Si la moustache gauche est pressée, il recule à droite, il fait un cercle de 360 degrés, après il fait demi-tours tout en tournant sur lui même;

Si la moustache droite est pressée, il tourne à gauche avec angle de 30 degrés.

Photo Robot :





Sources :
http://learn.parallax.com/tutorials/robot/shield-bot/robotics-board-education-shield-arduino/chapter-6-light-sensitive-14