Rapport de Travaux Pratiques

Robotique Mobile

2020-2021

Yohan NOUET
Barthélémy MARSAULT
Xiaoqing GUO

Table des matières

1. Configuration matérielle

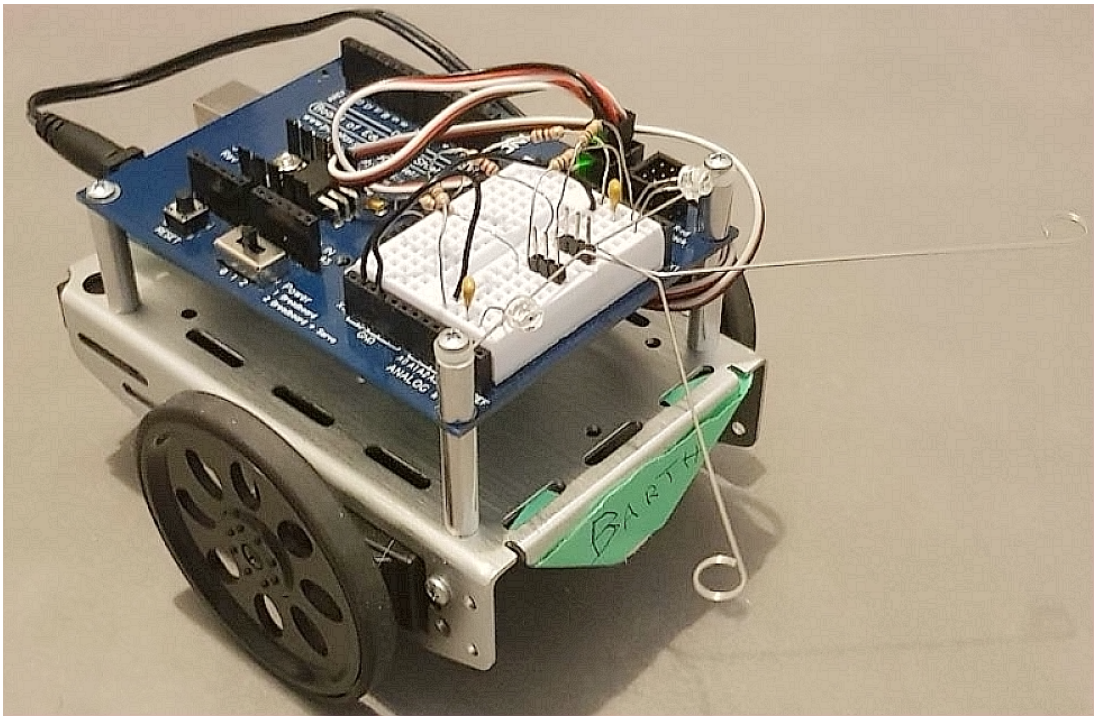
- 1 carte Arduino Uno Rev 3
- 1 shield Boe
- des entretoises en aluminium de 2.5 cm de hauteur
- des vis
- 2 servo moteurs
- 1 châssis
- 2 bagues de protection en caoutchouc
- 1 support des batteries
- pack batterie (4 batteries)
- 1 roue arrière
- 2 roues latérales
- des câbles
- 2 capteurs de contact
- 2 capteurs de luminosité

2. Assemblage du robot

2.1. Caractéristiques

Le robot possède une structure de type uni-cycle. Ses déplacements sont gérés par deux servomoteurs. Et il est alimenté en permanence par la batterie. Il peut reconnaître la lumière et le toucher. Il peut continuer à avancer et éviter les obstacles qu'il rencontre en adaptant sa trajectoire.

2.2. Photos de l'assemblage du robot



3. Fonctions et stratégie de déplacement

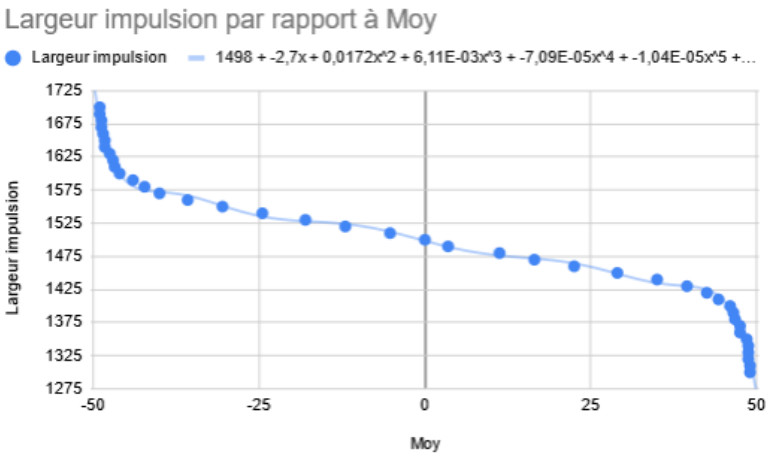
3.1. Modélisation de la vitesse de rotation

Afin d'intégrer la vitesse de rotation en tour/min dans les fonctions de déplacements du robot, nous avons commencé par relever les différentes valeurs de vitesse de rotation selon les valeurs d'impulsion des servo-moteurs. L'ensemble des valeurs a été recueilli dans un tableur excel :

Vitesse moyenne (rpm)	Largeur impulsion
49	1300
49	1310
48,75	1320
48,75	1330
48,75	1340
48,5	1350
47,5	1360
47,5	1370
46,75	1380
46,5	1390
46	1400
44,25	1410
42,5	1420
39,5	1430
35	1440
29	1450
22,5	1460
16,5	1470
11,25	1480
3,5	1490
0	**1500**
-5,25	1510
-12	1520
-18	1530
-24,5	1540
-30,5	1550
-35,75	1560
-40	1570
-42,25	1580
-44	1590
-46	1600
-46,75	1610
-47	1620
-47,5	1630
-48,25	1640
-48,25	1650
-48,5	1660
-48,75	1670
-48,75	1680
-49	1690
-49	1700

Les vitesses obtenues sont les moyennes des valeurs observées pour chacune des roues. Cela nous a permis de tracer la courbe de correspondance entre les valeurs d'impulsion et

les vitesses. Nous avons ensuite pu modéliser la courbe à l'aide d'une fonction polynomiale de degré 10.



La modélisation polynomiale nous permet d'obtenir l'équation de l'impulsion en fonction de la vitesse de rotation :

$$imp = 1498 - 2{,}7x + 0{,}0172x^2 + 6{,}11E\text{-}03x^3 - 7{,}09E\text{-}05x^4 - 1{,}04E\text{-}05x^5 + 9{,}25E\text{-}08x^6 + 6{,}59E\text{-}09x^7 - 4{,}7E\text{-}11x^8 - 1{,}42E\text{-}12x^9 + 8{,}17E\text{-}15x^{10}$$

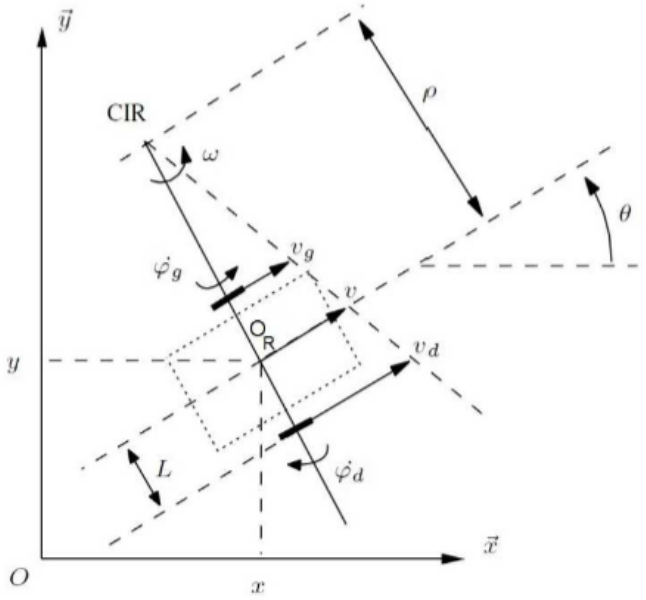
Dans un premier temps, nous avons implémenté la fonction *getImpulsion(double vitesse)* qui réalisait la simple conversion à l'aide de la formule de la vitesse de rotation souhaitée en largeur d'impulsion. Cependant, à l'usage nous avons remarqué que cette fonction était appelée très fréquemment et que le temps de calcul du polynôme était assez important pour impacter le comportement du robot.
La fonction a donc été modifiée pour résoudre ce problème. Le polynôme à été réécrit selon la méthode de Horner. Voici la nouvelle version de la fonction de calcul d'impulsion :

```
int getImpulsion(double vitesse){

  if(vitesse == 0) return 1500; //Evite les erreurs d'arrondi pour la vitesse 0

  double x = vitesse;
  int impulsion;
  double w;
  int const nbDeg = 10;

  //double coef[nbDeg+1] = {8.17*pow(10,-15), -1.42*pow(10,-12), - 4.7*pow(10,-11), 6.59*pow(10,-9),
  //9.25*pow(10,-8), -0.0000104, -0.0000709, 0.00611, 0.0172, - 2.7, 1498};

  double coef[nbDeg+1] = {0.00000000000000817, -0.00000000000142, -0.000000000047, 0.00000000659,
  0.0000000925, -0.0000104, -0.0000709, 0.00611, 0.0172, - 2.7, 1498};

  w = coef[0]; //Initialisation de w avec le coef de plus grand degré
  for(int i = 1; i < nbDeg+1; i++){
    w = w * x + coef[i];   //Récriture de la variable en suivant la méthode de Horner
  }

  impulsion = w;

  //Verification pour ne pas dépasser la capacité maximal des servos moteurs.
  if(impulsion > IMP_MAX) return IMP_MAX;
  if(impulsion < IMP_MIN) return IMP_MIN;
  return impulsion;
}
```

Afin de ne pas dépasser les bornes supérieure et inférieure de la largeur d'impulsion, nous avons implémenté deux conditions qui nous permettent de respecter l'intervalle de valeurs.

3.2. Modélisation du rayon de courbure

Schéma de représentation du robot uni-cycle dans l'espace :



Pour implémenter la gestion du rayon de courbure dans les fonctions de déplacement du robot, nous nous sommes penchés sur les relations entre rayon de courbure et vitesse de rotation des deux roues. D'après le cours, nous avons la définition suivante du rayon de courbure :

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

Avec :

- L : la longueur du demi-essieu
- Vd : le vecteur vitesse de la roue droite
- Vg : le vecteur vitesse de la roue gauche

Pour obtenir les vitesses de chaque moteur en fonction du rayon de courbure, il faut chercher à trouver la fonction réciproque. Ainsi avec une vitesse fixée pour l'un des moteurs, nous serons capables de calculer la vitesse du second moteur en fonction du rayon de courbure. Dans un premier temps nous avons choisi de fixer une valeur pour v_d. Les seules variables sont donc ρ et v_g

Soit la formule réécrite sous la forme suivante :

$$\left(\frac{\rho}{L}\right) = \frac{v_d + v_g}{v_d - v_g}$$

Par équivalence entre les équations suivantes nous obtenons la relation réciproque :

$$\left(\frac{\rho}{L}\right) = \frac{v_d + v_g}{v_d - v_g}$$

$$\left(\frac{\rho}{L}\right)(v_d - v_g) = v_d + v_g$$

$$v_d\left(\frac{\rho}{L}\right) - \left(\frac{\rho}{L}\right)v_g - v_g = v_d$$

$$v_d\left(\left(\frac{\rho}{L}\right) - 1\right) - v_g\left(\left(\frac{\rho}{L}\right) + 1\right) = 0$$

$$v_g = v_d\frac{\left(\left(\frac{\rho}{L}\right) - 1\right)}{\left(\frac{\rho}{L}\right) + 1}$$

Ainsi pour une vitesse donnée pour la roue droite et un rayon de courbure, nous pouvons déterminer la vitesse à paramétrer pour la roue gauche. Cela nous permet de respecter le rayon de courbure.

La fonction peut ensuite être implémentée dans le code :

```
double vitesseSelonRC(double P, int vitesseMax){
  if(P < 0) return -vitesseMax; //UTILE ???
  double vitesseMin;
  vitesseMin = (vitesseMax*(P/L-1))/(P/L+1); //L (constante) est la demi largeur du robot
  return vitesseMin;
}
```

Ici vitesseMax correspond à la valeur de vitesse fixée qui, avec le rayon de courbure ρ, permettra de déterminer la seconde vitesse de rotation. Afin de rester dans la plage des valeurs possibles d'impulsion, nous avons convenu de toujours utiliser vitesseMax pour la roue qui se situe à l'extérieur du virage (la plus éloignée du CIR). Ainsi, cela nous garantit que la vitesse calculée par la fonction sera inférieure à vitesseMax et permet d'avoir une valeur d'impulsion qui respecte la plage de valeurs admissibles.

Exemple d'utilisation dans la fonction qui permet de tourner vers la droite :

```
void droite(int duree_deplacement, int RC, int vitesse){
  double v = vitesseSelonRC(RC,vitesse);
  int imp = getImpulsion(v);
  Servo_droit.writeMicroseconds(imp);
  Servo_gauche.writeMicroseconds(getImpulsion(-vitesse));
  if(duree_deplacement >= 0){
    delay(duree_deplacement);
  }
}
```

Celle-ci permet de préciser une durée, un rayon de courbure souhaité et une vitesse pour la roue qui tourne le plus vite. Une vitesse négative permet au robot de reculer tout en tournant.

3.3. La stratégie de déplacement (avec des capteurs)

Au début du programme, le robot effectue un tour sur lui-même en capturant la luminosité 8 fois lors de cette rotation. Cette initialisation lui permet de capter l'endroit le plus lumineux et de se replacer dans cette direction.
Fonction correspondant: initDirection()
Code correspondant:

```
for(int i = 0 ; i < nbEtape; i++){

  long g = rcTime(PH_GAUCHE); //Lecture de la luminosite du capteur gauche
  long d = rcTime(PH_DROITE); //Lecture de la luminosite du capteur droit
  if((g+d)/2 < minLumValue){
    minLumValue = (g+d)/2;
    nbRota = i; //Retient la direction la plus eclairé
  }

  surPlaceDroite(timeFullTour/nbEtape,VITESSE_MAX); //Tourne de nb_etape

}
Servo_droit.writeMicroseconds (IMP_ARRET);
Servo_gauche.writeMicroseconds (IMP_ARRET);
delay(500);
surPlaceDroite((timeFullTour/nbEtape)*(nbRota),VITESSE_MAX); //Se place dans la direction la plus eclairé
```

Une fois le robot placé, il va effectuer des captures de luminosité toutes les 60ms :

- Si la différence entre la valeur des deux capteurs est inférieure à 50% de la moyenne des deux valeurs, alors le robot va tout droit.
- Avec le même calcul, si la différence obtenue est supérieure à 50% alors le robot se dirige du côté le plus éclairé.

Fonction correspondant: loop()
Code correspondant:

```
//Control du cote le plus lumineux et avancer en fonction
if(lum_diff < ((lum_gauche + lum_droite)/2)*0.5){
 avancerRobot(60);
}else if(lum_gauche < lum_droite){
  gauche(60,20,VITESSE_MAX); //Temps,Rayon de courbure,Vitesse
}else{
  droite(60,20,VITESSE_MAX); //Temps,Rayon de courbure,Vitesse
}
```

A la fin du cycle de déplacement de 60 ms, le robot effectue un contrôle de ses moustaches afin de vérifier si l'une ou les deux sont en contact avec un obstacle. Si c'est le cas, le robot se replace en fonction.
Fonction correspondant: controlMoustache()
Code correspondant:

```
if(digitalRead(M_GAUCHE) == LOW && digitalRead(M_DROITE) == LOW){ //DEMI-TOUR
  reculerRobot(randomBackTime);
  surPlaceDroite(1100,VITESSE_MAX);
}else if(digitalRead(M_GAUCHE) == LOW){ //Choc à gauche deplacement vers la droite
  reculerRobot(randomBackTime);
  surPlaceDroite(randomTurnTime, VITESSE_MAX);
}else if(digitalRead(M_DROITE) == LOW){ //Choc à droite deplacement vers la gauche
  reculerRobot(randomBackTime);
  surPlaceGauche(randomTurnTime, VITESSE_MAX);
}
```

Une fois cela fait, la boucle recommence, le robot mesure la luminosité et ajuste sa trajectoire en fonction.

Pour résumer, le robot se déplace donc en avant en direction de la zone la plus éclairée tout en détectant les obstacles de sorte à les éviter. De plus, en cas de contacte des deux moustaches simultanément, le robot fait un demi-tour sur lui-même et repart dans l'autre sens.

4. Cas d'activité - Vidéo

Dans la vidéo, nous avons montré au robot la collecte d'informations et les tests d'évitement d'obstacles. Notre robot peut effectuer les déplacements décrits dans la partie 3.
Deux vidéos représentent pour l'une le robot se dirigeant à la recherche de l'endroit le plus lumineux et l'autre le cas d'un contact avec un obstacle.