Rapport de la séance 10

21 février 2024

WARTSKI NARANJO Daniel

Robotique

Travail réalisé

Lors de cette séance, j'ai entrepris la modification du programme, en particulier de la fonction 'suiv_ligne()', en remplaçant la boucle de calcul de la distance entre le robot et l'obstacle par une boucle 'while'. J'ai inclus la mesure de la distance et des capteurs infrarouges à chaque passage dans cette boucle. Cette adaptation a été réalisée dans le but de faciliter la compréhension du fonctionnement et de permettre une meilleure gestion de la distance par rapport à l'obstacle.

Par la suite, j'ai entamé la compréhension et l'implémentation du correcteur PID dans le programme. Cette démarche a nécessité des ajustements au niveau des broches utilisées, notamment en libérant les broches 2 et 3, réservées à deux capteurs infrarouges servant au calcul de la vitesse des roues. Ces broches sont également essentielles pour l'utilisation de la fonction 'attachInterrupt()'. Les images de la fonction 'suiv_ligne', des nouvelles broches, du montage et du PID sont disponibles ci-dessous.

```
#include <TimerOne.h>
#include <NewPing.h>

#define M1A 11
#define M1B 9
#define M2A 6
#define M2B 5
#define IR1 7
#define IR2 12
#define IR3 4
#define echo 8
#define trig 13

int distance;
int v=120;
String a="s";

int PWMD=0;
int PWMG=0;
unsigned int compteurD=0;
unsigned int compteurG=0;
unsigned long int deltatemps=20;
float const Mult=1/(deltatemps*0.001);
float RPMD;
float RPMG;
```

```
void setup() {
  // put your setup code here, to run once:
  Timer1.initialize(deltatemps*1000);
  Timer1.attachInterrupt(CalculD);
  Timer1.attachInterrupt(CalculG);
  Serial.begin(9600);
  pinMode(M1A,OUTPUT);
  pinMode(M2A,OUTPUT);
  pinMode(M1B,OUTPUT);
  pinMode(M2B,OUTPUT);
  pinMode(IR1,INPUT);
  pinMode(IR2,INPUT);
  pinMode(IR3,INPUT);
  pinMode(echo,INPUT);
  pinMode(trig,OUTPUT);
  attachInterrupt(0,changeD,FALLING);
  attachInterrupt(1,changeG,FALLING);
  stop();
  delay(1000);
}
```

```
void suiv_ligne(){
  NewPing sonar(trig,echo);
  distance=sonar.ping_cm();
  while(distance>=20){
    int IRC=digitalRead(IR2);
    int IRR=digitalRead(IR1);
    int IRL=digitalRead(IR3);
    distance=sonar.ping_cm();
    if (IRC==0){
      if (IRR==1 && IRL==0){
        right();
        a="r";
        Serial.println("RIGH1");
      }
      else if (IRR==0 && IRL==1){
        left();
        a="l";
        Serial.println("LEFT1");
      }
      else if(IRR==1 && IRL==1){
        if (random(2) == 0) {
          right();
          a="r";
          Serial.println("RIGH2");
        }
        else {
          left();
          a="l";
          Serial.println("LEFT2");
        }
      }
```

```
    else if(IRR==0 && IRL==0){
      if(a=="l"){
        left();
        Serial.println("LEFT1");
      }
      else if(a=="r"){
        right();
        Serial.println("RIGH1");
      }
      else if(a=="lr"){
        little_right();
        Serial.println("LITTLE_RIGH");
      }
      else if(a=="ll"){
        little_left();
        Serial.println("LITTLE_LEFT");
      }
      else if(a=="f"){
        forward();
        Serial.println("FORWARD");
      }
      else if(a=="s"){
        stop();
        Serial.println("STOP");
      }
    }
  }
```

```
    else if (IRC==1){
      if (IRR==1 && IRL==0){
        little_right();
        a="lr";
        Serial.println("LITTLE_RIGH");
      }
      else if (IRR==0 && IRL==1){
        little_left();
        a="ll";
        Serial.println("LITTLE_LEFT");
      }
      else if (IRR==0 && IRL==0){
        forward();
        a="f";
        Serial.println("FORWARD");
      }
      else if (IRR==1 && IRL==1){
        stop();
        Serial.println("STOP");
      }
    }
  }
  stop();
  Serial.println("STOP2");
}
```

```
//Calcul de la vitesse des moteur en tr/s
void CalculD(){
  RPMD=Mult*compteurD;
  Serial.println(millis());
  Serial.print(" ");
  Serial.println(RPMD,3);
  compteurD=0;
}

void CalculG(){
  RPMG=Mult*compteurD;
  Serial.println(millis());
  Serial.print(" ");
  Serial.println(RPMG,3);
  compteurG=0;
}

//******Interruptions******
void changeD(){
  compteurD++;
}

void changeG(){
  compteurG++;
}
```

```
//******Fonction pour le moteur droit******
void moteurD(int v, byte sens){
  if (sens==0){
    analogWrite(M1B,0);
    analogWrite(M2B,v);
  }
  else{
    analogWrite(M2B,0);
    analogWrite(M1B,v);
  }
}

//******Fonction pour le moteur gauche******
void moteurG(int v, byte sens){
  if (sens==0){
    analogWrite(M1A,0);
    analogWrite(M2A,v);
  }
  else{
    analogWrite(M2A,0);
    analogWrite(M1A,v);
  }
}
```

```
//******Correcteur Proportionnel******
const float consigneD=1.4; //Vitesse souhaitée pour la roue en tr/s
float erreurD=0.0; //difference entre la consigne et la mesure
const int  kp=100; //coefficient de proportionnalité
void correcteur_P(){
  erreurD=consigneD-RPMD; // calcul de l'erreur
  PWMD=kp*erreurD; //determination de la valeur du PWM
  if (PWMD<0){
    PWMD=0; //bornage du PWM dans l'intervalle [0,255]
  }
  else if (PWMD>255){
    PWMD=255;
  }
  moteurD(PWMD,1); //application du nouveau signal PWM
}

//******Correcteur Proportionnel-Intégral******
const int ki=0.2; //coefficient de l'intégrale
float somme_erreurD=0.0; //somme des erreurs
void correcteur_PI(){
  erreurD=consigneD-RPMD; // calcul de l'erreur
  somme_erreurD=somme_erreurD+erreurD;
  PWMD=kp*erreurD+ki*somme_erreurD; //determination de la valeur du PWM
  if (PWMD<0){
    PWMD=0; //bornage du PWM dans l'intervalle [0,255]
  }
  else if (PWMD>255){
    PWMD=255;
  }
  moteurD(PWMD,1); //application du nouveau signal PWM
}
```

```
//******Correcteur Proportionnel-Intégral-Dérivé******
const float kd=10; //coefficient de la dérivée
float erreurD_avant=0.0; //erreur précédente
float delta_erreurD=0.0; //variation de l'erreur
void correcteur_PID(){
  erreurD=consigneD-RPMD; // calcul de l'erreur
  somme_erreurD=somme_erreurD+erreurD;
  delta_erreurD=erreurD-erreurD_avant;
  erreurD_avant=erreurD;
  PWMD=kp*erreurD+(ki*somme_erreurD)+(kd*delta_erreurD); //determination de la valeur du PWM
  if (PWMD<0){
    PWMD=0; //bornage du PWM dans l'intervalle [0,255]
  }
  else if (PWMD>255){
    PWMD=255;
  }
  moteurD(PWMD,1); //application du nouveau signal PWM
}
```

En dernier lieu, j'ai opté pour une nouvelle source d'alimentation, deux piles 18650 rechargeables. Pour accompagner ce changement, j'ai débuté la modélisation 3D d'une boîte destinée à abriter les piles.

Objectifs pour la prochaine séance

- Poursuivre la mise en place du programme pour intégrer le correcteur PID.
- Continuer le développement du programme du robot afin qu'il réponde aux exigences spécifiées.
- Réaliser le modèle 3D de la boîte destinée à contenir les piles et procéder à l'installation de celle-ci dans le robot.
- Intégrer les deux capteurs infrarouges dans le robot pour mesurer la vitesse des roues.