```
//#############################################################################//
              //# Projet Sade V1.1 - TYLT {code the world} #//
           //# Radolanirina Yaël - Kisiela Tom - L'Haridon Louis #//
              //# Maquette de fauteuil roulant intelligent #//
//#############################################################################//

// Necessaire pour le fonctionnement du robot
#include "Arduino.h"

//On importe les bibliothèques
#include <Ultrasonic.h>     // Pour les capteurs
#include <SoftwareSerial.h> //Pou r le Port Sérial

//Pour les roues

// Ports pour le ShieldBot 0.9/1.0
#define droite_avant 5     //definit l'interface I1 (moteur droite avant)
#define speedPinRight 6          // moteur droit (bridge A) (M1) (il "reunit les deux
ports du moteur droit" (M1))
#define droite_arriere 7       //dérinit l'interface I2 (moteur droit arrière)
#define gauche_avant 8         //Définit l'interface I3 (moteur gauche avant)
#define speedPinLeft 9     // moteur gauche (bridge B) (M2) (il "reunit les deux ports
du moteur gauche" (M2))
#define gauche_arriere 10      //Définit l'interface I4 (moteur Gauche arrière)

//On s'occupe de la vitesse référence des moteurs
int speedmotorA = 250; //define the speed of motorA
int speedmotorB = 250 ; //define the speed of motorB
// Nous travaillrons sur une échelle de -100 à 100; pour les drive(Rg,Rd)
           int Rd; // Valeur sur cette échelle de la roue droite
           int Rg; // Valeur sur cette échelle de la roue gauche


   // On continue avec la valeur des capteurs
        long D; // Variable pour le capteur droit
    long G; // Variable pour le capteur gauche

// On s'occupe des distances de sécurité et de la "cote" du prototype
int CoteProto= 1;
int DistArret= CoteProto*5;
int DistSecu= CoteProto*30;


//Pour les capteurs
Ultrasonic droite(4);
Ultrasonic gauche(3);

//Pour la led
int led = 2;

//Pour le bluetooth
SoftwareSerial bluetooth(0,1); //Ports du Bluetooth
int DonneeBluetooth = 0;   // Pour les Données bluetooth
int switcher = 0;               // Switcher pour l'activation ou non du système
```

```
// Fonction Paramètres
void setup() {

  // Ici s'occupe des ports du robot
  // Pour les moteurs
  pinMode(droite_avant,OUTPUT);
  pinMode(droite_arriere,OUTPUT);
  pinMode(speedPinRight,OUTPUT);
  pinMode(gauche_avant,OUTPUT);
  pinMode(gauche_arriere,OUTPUT);
  pinMode(speedPinLeft,OUTPUT);
  //Pour la Led
  pinMode(led, OUTPUT);
  //Pour le Bluetooth
  pinMode(1, OUTPUT);
  pinMode(0, INPUT);

  //De base le robot doit être à l'arret
  drive(0,0);

     Serial.println("Initialisation...");



  // Port série réglé sur 38400 Bauds
  Serial.begin(38400);
  Serial.println("Initialisation en cours...");
   Serial.println(" ");

       connexionBt();
       Serial.println(" ");

  //Message à afficher sur le Moniteur Série au démmarage
  Serial.print("Touches -- z , q , s , d, a, o, f pour le pilotage du robot \n");
  Serial.print("z = tout droit \n");
  Serial.print("s = marche arriere \n");
  Serial.print("d= droite \n");
  Serial.print("q = gauche \n");
  Serial.print("a = Stop  \n");
  Serial.print("o/f = activer/desactiver le système Sade  \n");
  Serial.println("TYLT {Code The World} - Projet Sade V1.1");
   Serial.println(" ");
     Serial.println("Lancement de la verification systeme ...");

      int error=0;

      D = droite.MeasureInCentimeters(); // D prend la valeur en centimètres de la
distance du capteur droit
      G = gauche.MeasureInCentimeters(); // G prend la valeur en centimètres de la
distance du capteur gauche

         if (D > 0) {
       char VerifD[ ] = "ok";
         Serial.print("Capteur droit: ");
```

```
          Serial.println(VerifD);   }
           else{
            char VerifD[ ] = "erreur";
           Serial.print("Capteur droit: ");
           Serial.println(VerifD);
              error=1;
  }
           if  (G>0)
           {char VerifG[ ] = "ok";
           Serial.print("Capteur gauche: ");
           Serial.println(VerifG);  }
           else{
           char VerifG[ ] = "erreur";
           Serial.print("Capteur gauche: ");
           Serial.println(VerifG);
              error=1; }

             if (Serial.available() > 0) {              // Si Le port Série/Bluetooth
est disponible
char VerifBT[ ] = "ok";
           Serial.print("Bluetooth: ");
           Serial.println(VerifBT);  }
         else{
           char VerifBT[ ] = "erreur";
           Serial.print("Bluetooth: ");
           Serial.println(VerifBT);
                error=1; }

           if( error == 1){Serial.println(" "); Serial.println("Verifiez les
branchements"); Serial.println("reboot recommande");  Serial.println(" ");}
           else{Serial.println(" ");Serial.println("Seance d'initialisation terminee
\nrobot operationel");}


Serial.println("");
}



//Notre Fonction principale (loop)
void loop()
{
        // On lit la valeur reçue par le bluetooth ou le moniteur Série
        if (Serial.available() > 0) {              // Si Le port Série/Bluetooth est
disponible
         DonneeBluetooth = Serial.read();   // On lit les données du Bluetooth
    }


       //On lit les données du bluetooth et on s'occupe du switcher (1= allumé 0=
éteint)
       switch(DonneeBluetooth)
       {
         case 'o':       // Si DonneeBluetooth reçoit la valeur 'o'
         switcher = 1;   // alors le switcher vaut 1 (position haute) = système Sade
```

```
activé
         break;

         case 'f':      // Si DonneeBluetooth reçoit la valeur 'f'
         switcher = 0;  // alors le switcher vaut 0 (position basse) = système Sade
éteint
         break;
        }


        // Maintenant notre switcher paramétré, On s'occupe des deux situations
possibles (position basse ou position haut)


        // D'abord si le switcher retourne la valeur 0 (position basse = système Sade
éteint)
        if(switcher == 0)
        {
            // On passe en mode de commande
            command();
        }




        // Ensuite si le switcher retourne la valeur 1 (position haute = système Sade
activé)
        else if (switcher == 1)
        {


                      // Si la distance du capteur droit ou  du capteur gauche est
inférieur à DistArret cm, il y a danger imminent sur un côté
                      if( (D < DistArret) || (G <DistArret)  )
                     {

                          // Si la distance du capteur droit est inférieure à DistArret
cm, procédure d'urgene avec redémarrage avec esquive de l'obstacle
                          if((D < DistArret) )
                          {
                            fastStop();      // Le robot s'arrête en urgence
                              delay(500);    // pendant 500 ms (0.5s)
                              drive(-50,50); // Le robot tourne à gauche
                              delay(250);    // pendant 250 ms (0.25s)
                          }

                          // Si la distance du capteur gauche est inférieure à
DistArret cm, procédure d'urgene avec redémarrage avec esquive de l'obstacle
                          if((G < DistArret) && (G > 0))
                          {
                              fastStop();    // Le robot s'arrête en urgence
                              delay(500);    // pendant 500 ms (0.5s)
                              drive(50,-50); // Le robot tourne à droite
                              delay(250);    // pendant 250 ms (0.25s)
                          }
```

```
                    }

                    // Si la distance du capteur droit ou du capteur gauche est
inférieur à DistSecu cm, il y  a danger lointain sur un des côtés, on enclenche la
procédure d'esquive
                     else if ((D < DistSecu || G < DistSecu))
                    {
                        // Si la distance du capteur droit est inférieure à 30 cm,
procédure d'esquive
                          if(D < DistSecu)
                          {
                               Rd= 100+((-40)/((15^5)*((15^-5)+(D^-5))));    // La
vitesse de la roue gauche est calculée parl'expression sigmoide
                               Rg= -15+(5/(-15^4)*((15^-5)+(D^-5)));        // La
vitesse de la roue gauche est calculée par l'expression sigmoide
                               if (D < 10){Rg= -15;} ; // Si la distance est inférieure
à 10La vitesse de la roue gauche est de -15 (sinon la roue ne tourne plus avec
l'équation précédebte a cause de la puissance du moteur
                               drive(Rg,Rd);      // Le robot avance en fonction des
vitesse calculées ci-dessus et roule de façon à détourner sa trajectoire de l'obstacle
                          }

                        // Si la distance du capteur droit est inférieure à DistSecu
cm, procédure d'esquive
                          else if(G < DistSecu)
                          {
                              Rg= 100+((-40)/((15^5)*((15^-5)+(G^-5))));    // La
vitesse de la roue gauche est calculée parl'expression sigmoide
                               Rd= -15+(5/(-15^4)*((15^-5)+(G^-5)));        // La
vitesse de la roue droite est calculée par l'expression sigmoide
                               drive(Rg,Rd);      // Le robot avance en fonction des
vitesse calculées ci-dessus et roule de façon à détourner sa trajectoire de l'obstacle
                          }


                    }

                    // Sinon (donc si il n'y a pas d'obstacles détectés)
                    else
                     {
                        command();
                    }
        }
}
// Fin de la fonction principale (loop)




// A partir de la on s'occupe de la gestion du robot
```

```
//On s'occupe du bluetooth
void connexionBt(){
 bluetooth.begin(38400); //Le bluetooth fonctionne sur 38400 Bauds
 bluetooth.print("\r\n+STWMOD=0\r\n"); //Notre robot marche en mode "slave" (escalve)
il suit donc les ordres
 bluetooth.print("\r\n+STNA=RobotTYLT\r\n"); //Le bluetooth s'appelle "RobotTYLT"
 bluetooth.print("\r\n+STPIN=0000\r\n");//le code PIN de connexion est "0000"
 bluetooth.print("\r\n+STOAUT=1\r\n"); // Les appareils peuvent se connecter
 bluetooth.print("\r\n+STAUTO=0\r\n"); //  Auto-connection interdite
 delay(2000);                          // Ce délai est obligatoire
 bluetooth.print("\r\n+INQ=1\r\n"); ///Le Robot est pret
 Serial.println("Interface homme machine prete");
 delay(2000); //Ce délai est obligatoire
 bluetooth.flush();
}

//L'avertisseur visuel (la led)
//D'abord pour le Capteur Droit
void AvertisseurVisuelD(){
 // On commence par la valeur des capteurs
        long D; // Variable pour le capteur droit
     D = droite.MeasureInCentimeters(); // D prend la valeur en centimètres de la
distance du capteur droit

          if (D<DistArret){
        tone(led, 250, 50);
        delay(50);
        digitalWrite(led, LOW);
        delay(D);
        }

        else if (D<DistSecu){
        tone(led, 250, 50);
        delay(20*D);
        digitalWrite(led, LOW);
        delay(10*D);
        }
        else if (D<2){
        digitalWrite(led, HIGH);
        }
        else{
        digitalWrite(led, LOW);
        }
}
//Ensuite pour le gauche
void AvertisseurVisuelG(){
         // On commence par la valeur des capteurs
     long G; // Variable pour le capteur gauche
        G = gauche.MeasureInCentimeters(); // G prend la valeur en centimètres de la
distance du capteur gauche

          if (G<DistArret){
         tone(led, 250, 50);
```

```
        delay(50);
        digitalWrite(led, LOW);
        delay(G);
        }

        else if (G<DistSecu){
        tone(led, 250, 50);
        delay(20*G);
        digitalWrite(led, LOW);
        delay(10*G);
        }
        else if (G<2){
        digitalWrite(led, HIGH);
        }
        else{
        digitalWrite(led, LOW);
        }
}
//Et maintenant le global
void AvertisseurVisuel(){
  AvertisseurVisuelD();
  AvertisseurVisuelG();
}



// On s'occupe des moteurs des roues


//On peut utiliser char qui va de -127 à 127 il faut
// On préfèrera ici int qui permet de libérer plus de variables...
//Rappel: notre échelle de travail sera de -100 à 100

//Mag sert à déterminer la direction du moteur
// Il est lu dans les fonctions rightMotor(Valeur_de_mag) et donc dans
drive(Valeur_de_mag_gauche,Valeur_de_mag_droit);
//Si mag < 0 , alors il fera machine arrière
//Si 0 il est immobile
//si mag>0 alors il fait machine avant


//On commence avec le moteur droit
void rightMotor(int mag){

    int actualSpeed = 0; // On met la vitesse du moteur à 0

    if(mag >0){ //Moteur fait machine avant
      float ratio = (float)mag/100; // On créé la variable ratio qui est basé sur la
valeur de mag divisé par 100 qui est la valeur maximale souhaitée
      actualSpeed = (int)(ratio*speedmotorA); //Ici on calcule la vitesse du moteur a
partirs de ce ratio en le multipliant  par speedMotorA (soit la vitesse max du moteur)
      // Exemple si rightMotor(60) alors mag=60 donc ratio=60/100 = 0.60 donc
actualpeed(la vitesse) vaut 0.6*speedmotorA 0.8*255 = 153 tours par minute

      analogWrite(speedPinRight,actualSpeed);
```

```
      digitalWrite(droite_avant,HIGH);//allume le moteur droit avant
      digitalWrite(droite_arriere,LOW);// Eteint le moteur droit arriere
    }

    else if(mag == 0){
      //Moteur innactivé
      stopDroit(); //On arrete le moteur droit
    }

    else {
      //Machine arrière
      float ratio = (float)abs(mag)/100; // On créé la variable ratio qui est basé sur
la valeur de mag divisé par 100 qui est la valeur maximale souhaitée
      actualSpeed = ratio*speedmotorA; //Ici on calcule la vitesse du moteur a partirs
de ce ratio en le multipliant  par speedMotorA (soit la vitesse max du moteur)

      analogWrite(speedPinRight,actualSpeed);
      digitalWrite(droite_avant,LOW); // Eteint le moteur droit avant
      digitalWrite(droite_arriere,HIGH);// Allume le moteur droit arriere
    }

}




//On continue avec le moteur gauche
void leftMotor(int mag){

    int actualSpeed = 0;// On met la vitesse du moteur à 0

    if(mag >0){ ///Moteur fait machine avant
      float ratio = (float)(mag)/100; // On créé la variable ratio qui est basé sur la
valeur de mag divisé par 100 qui est la valeur maximale souhaitée
      actualSpeed = (int)(ratio*speedmotorB); //Ici on calcule la vitesse du moteur a
partirs de ce ratio en le multipliant  par speedMotorB (soit la vitesse max du moteur)
     // Exemple si leftMotor(60) alors mag=60 donc ratio=60/100 = 0.60 donc
actualpeed(la vitesse) vaut 0.6*speedmotorB 0.8*255 = 153 tours par minute

      analogWrite(speedPinLeft,actualSpeed);
      digitalWrite(gauche_avant,HIGH);//allume le moteur gauche avant
      digitalWrite(gauche_arriere,LOW);// Eteint le moteur gauche arrieree
    }

    else if(mag == 0){
        //Moteur innactivé
     stopGauche(); //On arrete le moteur gauche
    }

  else {
    //Machine arrière
    float ratio = (float)abs(mag)/100; // On créé la variable ratio qui est basé sur la
valeur de mag divisé par 100 qui est la valeur maximale souhaitée
    actualSpeed = ratio*speedmotorB; //Ici on calcule la vitesse du moteur a partirs de
ce ratio en le multipliant  par speedMotorA (soit la vitesse max du moteur)
```

```
    analogWrite(speedPinLeft,actualSpeed);
    digitalWrite(gauche_avant,LOW);//allume le moteur gauche avant
    digitalWrite(gauche_arriere,HIGH);// Eteint le moteur gauche arrieree

  }
}



// Fonction drive
void drive(int left, int right){  // Char left et char right si on utilise char - on
récupère les informations pour l'échelle de -100 à 100 pour chaque roue
  rightMotor(right); // rightmotor renvoie la valeur mag= right
  leftMotor(left);   // leftmotor renvoie la valeur mag= left
}



//Fonctions  d'arret
// La globale
void stop(){
  analogWrite(speedPinLeft,0);  // Vitesse du moteur gauche à 0
  analogWrite(speedPinRight,0); // Vitesse du moteur droit à 0
}
// Pour le moteur droit
void stopDroit(){
  analogWrite(speedPinRight,0);  // Vitesse du moteur gauche à 0
}
//Pour le moteur Gauce
void stopGauche(){
  analogWrite(speedPinLeft,0);  // Vitesse du moteur gauche à 0
}


//Fonction d'arret rapide
// Peut être dangereux pour le moteur si la vitesse est suppérieure à 2 fois la vitess
maximale
void fastStop(){
   digitalWrite(gauche_avant,LOW);   // Eteint le moteur gauche avant
   digitalWrite(gauche_arriere,LOW); // Eteint le moteur gauche arrieree
   digitalWrite(droite_avant,LOW);  // Eteint le moteur droite avant
   digitalWrite(droite_arriere,LOW);// Eteint le moteur droite arrieree
}







// On s'occupe de la fonction qui recoit les données pour diriger le robot
void command(){
  // On passe en mode de commande
                          switch(DonneeBluetooth) //On lit les données du bluetooth
```

```
                        {
                         case 'a':      // Si DonneeBluetooth='a'
                           drive(0,0);  // Alors le robot s'arrete
                         break;

                         case 'd':       // Si DonneeBluetooth='d'
                           drive(60,-60);// Alors le robot tourne à droite
                         break;

                         case 'q':        // Si DonneeBluetooth='q'
                           drive(-60,60); // Alors le robot tourne à gauche
                         break;

                        case 'z':        // Si DonneeBluetooth='z'
                          drive(60,60);  // Alors le robot avance
                        break;

                        case 's':        // Si DonneeBluetooth='s'
                          drive(-60,-60);// Alors le robot recule
                        break;

                        case 'w':        // Si DonneeBluetooth='w'
                          drive(0,-100); // Alors le robot recule à gauche
                        break;

                        case 'c':        // Si DonneeBluetooth='c'
                          drive(-100,0); // Alors le robot recule à droite
                        break;

                        case 'e':        // Si DonneeBluetooth='e'
                          drive(0,100);  // Alors le robot avance à gauche
                        break;

                        case 'r':        // Si DonneeBluetooth='r'
                          drive(100,0);  // Alors le robot avance à droite
                        break;
                        }
}
```