#include <Servo.h>

#include <SoftwareSerial.h>

#define difference 40

#define TOO\_CLOSE 10 //Distance at which the robot will turn (cm)

/\*--VARIABLES----------------------------------------------------------------

 Definition of pin numbers where sensors and actuators will be connected

 Initially we use these, but they can be changed at your convinience.\*/

#define L\_wheel\_servo\_pin 6 //servo moteur roue gauche

#define R\_wheel\_servo\_pin 5 //servo moteur roue droite

//ultrason de gauche

#define ECHO\_PIN\_GAUCHE 7 //Echo pin (capte ultrason)

#define TRIG\_PIN\_GAUCHE 8 //Trigger pin (envoi ultrason)

//ultrason du centre

#define ECHO\_PIN\_CENTRE 9 //Echo pin (capte ultrason)

#define TRIG\_PIN\_CENTRE 10 //Trigger pin (envoi ultrason)

//ultrason de droite

#define ECHO\_PIN\_DROITE 11 //Echo pin (capte ultrason)

#define TRIG\_PIN\_DROITE 12 //Trigger pin (envoi ultrason)

const unsigned long MEASURE\_TIMEOUT = 25000UL;

const float SOUND\_SPEED = 340.0 / 1000;

//Variables for manual or follow light modes

#define MANUAL\_CONTROL 0

#define AVOID\_MODE 1

//---Start of LED extra siren------------------

const int redLED\_pin = 2; //Red LED

const int greenLED\_pin = 3; //Green LED

const int blueLED\_pin = 4; // Blue LED

//---End of LED extra siren--------------------

//Create servo objects for each servo

Servo L\_wheel\_servo;

Servo R\_wheel\_servo;

//Other variables

int servo\_delay = 20; //Time for the servo to make his function

int read\_Time = 150; //Time interval between reading distance

float distance\_centre;

int i = 0; //Buffer iterator

int numChar = 0; //Number of characters availables in the Serial

int currentState; // Variable that controls the current state of the program

char caractere ;

int state;//State of the robot

/\*

 -----------------------------------------------------------------------------

 ------------------------ SETUP --------------------------

 -----------------------------------------------------------------------------

 The setup function runs once every time that the board initializes or

 reset button is pushed. It's used to initciate variables or to setup

 different options.

 -----------------------------------------------------------------------------

\*/

void setup()

{

 //Setup servos

 pinMode(5,OUTPUT);

 L\_wheel\_servo.attach(L\_wheel\_servo\_pin);

 R\_wheel\_servo.attach(R\_wheel\_servo\_pin);

 //Setup as input or output for the pins

 pinMode(TRIG\_PIN\_GAUCHE, OUTPUT);

 pinMode(ECHO\_PIN\_GAUCHE, INPUT);

 pinMode(TRIG\_PIN\_CENTRE, OUTPUT);

 pinMode(ECHO\_PIN\_CENTRE, INPUT);

 pinMode(TRIG\_PIN\_DROITE, OUTPUT);

 pinMode(ECHO\_PIN\_DROITE, INPUT);

 //Start of LED ---------------------------------------------------

 pinMode(redLED\_pin, OUTPUT); //setup LED as output

 pinMode(greenLED\_pin, OUTPUT); //setup LED as output

 pinMode(blueLED\_pin, OUTPUT); //setup LED as output

 writeleds(0, 155, 0); //initial color for crest led (green)

 //-----------------------------------------------------

 currentState = MANUAL\_CONTROL; // Default state is manual control

 // Ouvre la voie serie avec l'ordinateur

 Serial.begin(9600);

 state = 0; //Initialize state 0 (STOP)

}

//========================================

void loop()

{

 if (Serial.available() > 0) {

 Serial.println("car recu");

 caractere = Serial.read();

 setAction(caractere);

 }

 if (currentState == AVOID\_MODE) {

 avoid();

 }

}

//=========================================

void setAction(char caractere) {

 switch (caractere) {

 /\* avoid mode button pressed \*/

 case 'I':

 currentState = AVOID\_MODE;

/\*

 Serial.print("0 si manuel, 1 si obstacles, 2 si lumiere: ");

 Serial.print(currentState);

 Serial.print("\n");

\*/

 break;

 /\* Manual control mode button pressed \*/

 case 'M':

 currentState = MANUAL\_CONTROL;

/\*

 Serial.print("0 si manuel, 1 si obstacles, 2 si lumiere: ");

 Serial.print(currentState);

 Serial.print("\n");

\*/

 break;

 /\* Stop button pressed \*/

 case 'S':

 if (currentState == MANUAL\_CONTROL)

 {

 writeleds(155, 0, 0); //rouge quand l e robot stop

 L\_wheel\_servo.write(92);//Stop left servo

 delay(20);

 R\_wheel\_servo.write(92);//Stop right servo

 delay(20);

 }

 break;

 /\* Up button pressed \*/

 case 'U':

 if (currentState == MANUAL\_CONTROL)

 {

 writeleds(0, 155, 0); //led verte quand le robot avance

 go\_forward();

 }

 break;

 /\* Down button pressed \*/

 case 'D':

 if (currentState == MANUAL\_CONTROL)

 {

 Serial.println("arriere");

 writeleds(0, 0, 155); //led bleue pour le recul

 L\_wheel\_servo.write(180);//Run left servo

 delay(20);

 R\_wheel\_servo.write(0);//Run right servo

 delay(20);

 }

 break;

 /\* Left button pressed \*/

case 'L':

 if (currentState == MANUAL\_CONTROL)

 {

 Serial.println("left");

 writeleds(155, 155, 0);

 L\_wheel\_servo.write(180);//Run left servo

 delay(20);

 R\_wheel\_servo.write(180);//Run right servo

 delay(20);

 }

 break;

 /\* Right button pressed \*/

case 'R':

 if (currentState == MANUAL\_CONTROL) //passage mode manuel a ultrasons

 {

 Serial.println("right");

 writeleds(155, 155, 0);

 L\_wheel\_servo.write(0);//Run left servo

 delay(20);

 R\_wheel\_servo.write(0);//Run right servo

 delay(20);

 }

 break;

}

/\* Empty the Serial \*/

Serial.flush();

}

//EN AVANT

void go\_forward() {

 Serial.println("avant");

 L\_wheel\_servo.write(0);//Run left servo

 delay(20);//we give time for the servo to act

 R\_wheel\_servo.write(180); //Run right servo

 delay(20);//we give time for the servo to act

/\*

 if (currentState == AVOID\_MODE)

 {

 if (distance\_centre < 15 ) { //Distance where the led colour is changed

 writeleds(155, 0, 0);

 } //red led

 else if (distance\_centre < 20) { //Distance where the led colour is changed to the next colour

 writeleds(155, 155, 0);

 } //yellow led

 else{

 writeleds(0, 155, 0); //if distance is higher, greed led

 }

 }

 \*/

}

//FONCTION POUR EVITER OBSTACLES

float distance(int trig, int echo){

 /\* 1. Lance une mesure de distance en envoyant une impulsion HIGH de 10micros sur la broche TRIGGER \*/

 digitalWrite(trig, HIGH);

 delayMicroseconds(10);

 digitalWrite(trig, LOW);

 /\* 2. Mesure le temps entre l'envoi de l'impulsion ultrasonique et son echo (si il existe) \*/

 long measure = pulseIn(echo, HIGH, MEASURE\_TIMEOUT);

 /\* 3. Calcul la distance \*/

 float distance\_mm = measure / 2.0 \* SOUND\_SPEED;

 return(distance\_mm / 10.0);

}

bool checkDistance(int seuil, int trig, int echo){

 float dist;

 for(int i = 0; i < 1 ; i++){

 //delay(500);

 dist = distance(trig,echo);

 Serial.println(dist);

 if(dist<seuil && dist != 0)

 return true; //permet d'entrer ds la condition et de tourner

 else return false;

 }

}

void avoid(){

 //Function that reads the ultrasonic sensor and resturns distance in cm

 int dataOk = 0;

 float distance\_centre;

 delay(read\_Time);

 Serial.println("control right");

 if( /\*checkDistance(TOO\_CLOSE, TRIG\_PIN\_CENTRE, ECHO\_PIN\_CENTRE) ||\*/ checkDistance(TOO\_CLOSE, TRIG\_PIN\_DROITE, ECHO\_PIN\_DROITE))

 {

 Serial.println("too close: right");

 L\_wheel\_servo.write(180);//Run left servo

 delay(10);//we give time for the servo to act

 R\_wheel\_servo.write(180);//Run right servo //on le fait legerement tourner a droite

 delay(10);//we give time for the servo to act

 }

 //control center

 else if (/\*checkDistance(TOO\_CLOSE, TRIG\_PIN\_GAUCHE, ECHO\_PIN\_GAUCHE) ||\*/ checkDistance(TOO\_CLOSE, TRIG\_PIN\_CENTRE, ECHO\_PIN\_CENTRE) /\*|| checkDistance(TOO\_CLOSE, TRIG\_PIN\_DROITE, ECHO\_PIN\_DROITE)\*/)

 {

 Serial.println("backwards...");

 L\_wheel\_servo.write(180);//Run left servo

 delay(10);//we give time for the servo to act

 R\_wheel\_servo.write(0);//Run right servo //on le fait retourner en arriere s'il rencontre un obstacle en face

 delay(10);//we give time for the servo to act

 Serial.println("and left");

 L\_wheel\_servo.write(0);//Run left servo

 delay(20);//we give time for the servo to act

 R\_wheel\_servo.write(0);//Run right servo //on le fait aller gauche

 delay(20);//we give time for the servo to act

 }

 //controle gauche

 else if(checkDistance(TOO\_CLOSE, TRIG\_PIN\_GAUCHE, ECHO\_PIN\_GAUCHE) /\*|| checkDistance(TOO\_CLOSE, TRIG\_PIN\_CENTRE, ECHO\_PIN\_CENTRE)\*/ )

 {

 Serial.println("too close: left");

 L\_wheel\_servo.write(0);//Run left servo

 delay(10);//we give time for the servo to act

 R\_wheel\_servo.write(0);//Run right servo //on le fait legerement tourner vers la droite s'il rencontre un obstacle en face

 delay(10);//we give time for the servo to act

 }

 else go\_forward();

}

//FONCTION LED RGB

void writeleds(int red, int green, int blue) {

 analogWrite(redLED\_pin, red); //write red led

 analogWrite(greenLED\_pin, green); //write green led

 analogWrite(blueLED\_pin, blue); //write blue led

}