#include <DRV8835MotorShield.h>

#include <hcsr04.h>

/\*

  This example uses the DRV8835MotorShield library to drive each motor with the

  Pololu DRV8835 Dual Motor Driver Shield for Arduino forward, then backward.

  The yellow user LED is on when a motor is set to a positive speed and off when

  a motor is set to a negative speed.

\*/

//Les pins du moteur 1 sont 7 et 9, moteur 2 sont 8 et 10.

#define MODE\_PIN 11

#define TRIG\_PIN 2

#define ECHO\_PIN 3

#define LED\_C 4 //JAUNE

#define LED\_P 5 //BLEU

#define BOUTON 6

#define SOL 12

#define SENS\_DPIN 0 //Droite

#define SENS\_GPIN 1 //Gauche

#define SENS\_CPIN 2 //Croisement

#define SENS\_PDPIN 3 //Palet Droite

#define SENS\_PGPIN 4 //Palet Gauche

//La veritable vitesse max(PWM) est 400.

#define MAX\_S 200

#define SEUIL 200

#define SEUILBLANC 200

#define SEUILNOIR 600

HCSR04 hcsr04(TRIG\_PIN, ECHO\_PIN, 20, 4000);

DRV8835MotorShield motors;

//resultat vitesse moteur de la regulation avec capteur infra

int vitesseM1 = 110;

int vitesseM2 = 110;

int countB = 0;

int countT = 0;

int pos = 25;

bool palet = false;

bool tir = false;

String readString;

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

void setup()

{

**Serial**.begin(9600);

 pinMode(LED\_C, OUTPUT);

 pinMode(LED\_P, OUTPUT);

 pinMode(SOL, OUTPUT);

 pinMode(BOUTON, INPUT);

 //M2 est a gauche (correction de +25?); M1 est à droite

 // uncomment one or both of the following lines if your motors' directions need to be flipped

 //motors.flipM1(true);

 //motors.flipM2(true);

 // Ne pas oublier de brancher le mode sur le motor driver!!

 digitalWrite(BOUTON, HIGH);

 digitalWrite(SOL, HIGH);

 digitalWrite(MODE\_PIN, HIGH);

 // Le bouton doit etre appuyé pour demarrer le programme

 bool phase0 = false;

 do {

   delay(1);

   while (**Serial**.available()) {

     delay(3);

     char c = **Serial**.read();

     readString += c;

   }

   if (readString.length() > 0) {

**Serial**.println(readString);

     if (readString == "go")phase0 = true;

     readString = "";

   }

   if (0 == digitalRead(BOUTON))phase0 = true;

 }

 while (phase0 == false);

 delay(2000);

}

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

void loop()

{

 //LongeRight();

 Opti();

 pos = 20;

 bool M = false;

 bool G = false;

 bool D = false;

 do {

   int sequ1 [28] =  {3, 1, 2, 1, 2, 1, 2, 4, 5, 6, 4, 1, 2, 1, 2, 4, 1, 2, 5, 6, 4, 1, 2, 1, 2, 4, 1, 2};

   int pos1 [28] =  {20, 2, 2, 25, 25, 58, 58, 58, 87, 87, 87, 74, 74, 41, 41, 41, 12, 12, 23, 23, 23, 36, 36, 69, 69, 69, 98, 98};

   int sequ2 [21] =  {3, 1, 2, 3, 5, 6, 4, 1, 2, 1, 2, 4, 1, 2, 5, 6, 4, 1, 2, 1, 2};

   int pos2 [21] =  {20, 2, 2, 2, 23, 23, 23, 36, 36, 69, 69, 69, 98, 98, 87, 87, 87, 74, 74, 41, 41};

   int sequ3 [11] =  {3, 1, 2, 3, 1, 2, 4, 1, 2, 1, 2};

   int pos3 [11] =  {20, 2, 2, 2, 23, 23, 23, 36, 36, 69, 69};

   int sequ4 [11] =  {3, 1, 2, 4, 1, 2, 3, 1, 2, 1, 2};

   int pos4 [11] =   {20, 2, 2, 2, 21, 21, 21, 14, 14, 47, 47};

   do {

     palet = false;

     if (!M && !D && !G)for (int i = 0; i < 28; i++) {

         if (palet == false) {

           pos = pos1[i];

           Choix (sequ1[i]);

           if (87 == pos)M = true;

           if (12 == pos)G = true;

         }

       }

     if (M && !D && !G)for (int i = 0; i < 21; i++) {

         if (palet == false) {

           pos = pos2[i];

           Choix (sequ2[i]);

           if (98 == pos)D = true;

         }

       }

     if (M && !D && G)for (int i = 0; i < 11; i++) {

         if (palet == false) {

           pos = pos3[i];

           Choix (sequ3[i]);

         }

       }

     if (M && D && !G )for (int i = 0; i < 11; i++) {

         if (palet == false) {

           pos = pos4[i];

           Choix (sequ4[i]);

         }

       }

   }

   while (palet == true);

 }

 while (1);

 // lecture capteur infra

 //seuil suiveur de ligne blanc <100 noir>100

 //Serial.println(analogRead(SENS\_PPIN));

 /\*GoToCross();

   GoLeft(200);

   GoUntil(SENS\_GPIN);

   Join(SENS\_GPIN);

   if(

   Center();

   CrossLeft();

   Shoot();\*/

}

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

//FONCTIONS DE PARCOURS

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

void GoShoot()

{

 if (pos == 69) {

   BackP();

   GoToCross();

   Center();

   pos = 9;

 }

 if (pos == 89) {

   CrossLeftWithP();

   pos = 9;

 }

 if (pos == 9) {

   GoToCross();

   Center();

   pos = 96;

 }

 if (pos == 47) {

   BackP();

   GoToCross();

   Center();

   pos = 7;

 }

 if (pos == 87) {

   CrossRightWithP();

   pos = 7;

 }

 if (pos == 7) {

   GoToCross();

   Center();

   pos = 74;

 }

 if (pos == 36) {

   CrossRightWithP();

   pos = 6;

 }

 if (pos == 6) {

   JustLeft(400);

   GoUntil(SENS\_DPIN);

   Join(SENS\_DPIN);

   GoToCross();

   Center();

   pos = 52;

 }

 if (pos == 96) {

   JustRight(400);

   GoUntil(SENS\_DPIN);

   Join(SENS\_DPIN);

   GoToCross();

   Center();

   pos = 52;

 }

 if (pos == 14) {

   CrossLeftWithP();

   pos = 4;

 }

 if (pos == 4) {

   JustRight(400);

   GoUntil(SENS\_GPIN);

   Join(SENS\_GPIN);

   GoToCross();

   Center();

   pos = 52;

 }

 if (pos == 74) {

   JustLeft(400);

   GoUntil(SENS\_GPIN);

   Join(SENS\_GPIN);

   GoToCross();

   Center();

   pos = 52;

 }

 if (pos == 23) {

   CrossRightWithP();

   BackP();

   GoToCross();

   Center();

   pos = 63;

 }

 if (pos == 63) {

   CrossLeftWithP();

   pos = 3;

 }

 if (pos == 3) {

   GoToCross();

   Center();

   pos = 32;

 }

 if (pos == 21) {

   CrossLeftWithP();

   BackP();

   GoToCross();

   Center();

   pos = 41;

 }

 if (pos == 41) {

   CrossRighWithP();

   pos = 1;

 }

 if (pos == 1) {

   GoToCross();

   Center();

   pos = 12;

 }

 if (pos == 58) {

   BackP();

   GoToCross();

   Center();

   pos = 8;

 }

 if (pos == 98) {

   CrossRightWithP();

   pos = 8;

 }

 if (pos == 78) {

   CrossLeftWithP();

   pos = 8;

 }

 if (pos == 8) {

   GoToCross();

   Center();

   pos = 85;

 }

 if (pos == 25) {

   BackP();

   GoToCross();

   Center();

   pos = 85;

 }

 if (pos == 65) {

   CrossRightWithP();

   pos = 85;

 }

 if (pos == 45) {

   CrossLeftWithP();

   pos = 85;

 }

 if (pos == 85) {

   GoToCross();

   Center();

   pos = 52;

 }

 if (pos == 2) {

   BackP();

   GoToCross();

   Center();

   pos = 52;

 }

 if (pos == 32) {

   CrossRightWithP();

   pos = 52;

 }

 if (pos == 12) {

   CrossLeftWithP();

   pos = 52;

 }

 if (pos == 52) {

   GoToCross();

   Shoot();

   pos = 20;

 }

}

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

void Choix(int c)

{

 switch (c)

 {

   case 1:

     GoToCrossRC();

     break;

   case 2:

     CenterRC();

     break;

   case 3:

     CrossLeft();

     break;

   case 4:

     CrossRight();

     break;

   case 5:

     GoToCross();

     break;

   case 6:

     Center();

     break;

   case 7:

     Shoot();

     break;

 }

}

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

// Parcours Version OPTI

void Opti()

{

 GoLeft(120);

 GoUntil(SENS\_GPIN);

 GoUp(100);

 GoUntil(SENS\_GPIN);

 GoUp(100);

 GoUntil(SENS\_DPIN);

 Join(SENS\_DPIN);

 GoToCross();

 JustLeft(300);

 GoUntil(SENS\_GPIN);

 Join(SENS\_GPIN);

 GoToCross();

 Center();

 GoToCross();

 Shoot();

 GoRight(850);

 GoUntil(SENS\_GPIN);

 GoUp(100);

 GoUntil(SENS\_DPIN);

 Join(SENS\_DPIN);

 GoToCross();

 JustLeft(1000);

 GoUntil(SENS\_DPIN);

 Join(SENS\_DPIN);

 GoToCross();

 Center();

 GoToCross();

 Shoot();

}

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

// Parcours Version ORTHO

void  Ortho()

{

 GoToCross();

 Center();

 CrossLeft();

 GoToCross();

 Center();

 CrossRight();

 GoToCross();

 Center();

 GoToCross();

 CrossRightWithP();

 GoToCross();

 CrossLeftWithP();

 GoToCross();

 Shoot();

 CrossRight();

 GoToCross();

 Center();

 CrossLeft();

 GoToCross();

 Center();

 CrossRight();

 GoToCross();

 CrossRightWithP();

 GoToCross();

 CrossRightWithP();

 GoToCross();

 Center();

 GoToCross();

 Shoot();

 Stop(5000);

}

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

//FONCTIONS A BOUCLE

void GoToCross() {

 bool phase = 0;

 int taux = SEUILNOIR - SEUILBLANC;

 double diviseur = 1;

 do {

   LedCheck();

   diviseur = analogRead(SENS\_DPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V1 = vitesseM1 - (vitesseM1 \* diviseur);

   diviseur = analogRead(SENS\_GPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V2 = vitesseM2 - (vitesseM2 \* diviseur);

   if (SEUILBLANC > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(vitesseM1);

   }

   if (SEUILBLANC < analogRead(SENS\_DPIN) && SEUILNOIR > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(V1);

   }

   if (SEUILNOIR < analogRead(SENS\_DPIN)) {

     motors.setM1Speed(0);

   }

   if (SEUILBLANC > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(vitesseM2);

   }

   if (SEUILBLANC < analogRead(SENS\_GPIN) && SEUILNOIR > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(V2);

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN)) {

     motors.setM2Speed(0);

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN) && SEUILNOIR < analogRead(SENS\_DPIN)) {

     phase = true;

   }

 }

 while (phase == false);

}

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

void BackP() {

 JustLeft(800);

 JustRight(1700);

 do {

   JustRight(1);

 }

 while (analogRead(SENS\_DPIN) < SEUILNOIR);

}

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

void LongeRight() {

 bool phase = 0;

 int taux = SEUILNOIR - SEUILBLANC;

 double diviseur = 1;

 do {

   LedCheck();

   //diviseur=analogRead(SENS\_DPIN)-SEUILBLANC;

   //diviseur=diviseur/taux;

   int V1 = vitesseM1; //\*diviseur;

   //diviseur=analogRead(SENS\_GPIN)-SEUILBLANC;

   //diviseur=diviseur/taux;

   int V2 = vitesseM2; //-(vitesseM2\*diviseur);

   if (SEUILBLANC > analogRead(SENS\_DPIN)) {

     motors.setM2Speed(V2);

     motors.setM1Speed(V1 \* 0.5);

   }

   if (SEUILBLANC < analogRead(SENS\_DPIN)) {

     motors.setM2Speed(V2 \* 0.5);

     motors.setM1Speed(V1);

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN)) {

     motors.setM2Speed(0);

     motors.setM1Speed(0);

     phase = true;

   }

 }

 while (phase == false);

 JustRight(600);

}

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

void LongeLeft() {

 bool phase = 0;

 int taux = SEUILNOIR - SEUILBLANC;

 double diviseur = 1;

 do {

   LedCheck();

   //diviseur=analogRead(SENS\_DPIN)-SEUILBLANC;

   //diviseur=diviseur/taux;

   int V1 = vitesseM1; //\*diviseur;

   //diviseur=analogRead(SENS\_GPIN)-SEUILBLANC;

   //diviseur=diviseur/taux;

   int V2 = vitesseM2; //-(vitesseM2\*diviseur);

   if (SEUILBLANC > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(V2 \* 0.5);

     motors.setM1Speed(V1);

   }

   if (SEUILBLANC < analogRead(SENS\_GPIN)) {

     motors.setM2Speed(V2);

     motors.setM1Speed(V1 \* 0.5);

   }

   if (SEUILNOIR < analogRead(SENS\_DPIN)) {

     motors.setM2Speed(0);

     motors.setM1Speed(0);

     phase = true;

   }

 }

 while (phase == false);

 JustLeft(600);

}

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

void GoToCrossRC() {

 bool phase = 0;

 int taux = SEUILNOIR - SEUILBLANC;

 double diviseur = 1;

 int a = hcsr04.distanceInMillimeters();

 do {

   LedCheck();

   diviseur = analogRead(SENS\_DPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V1 = vitesseM1 - (vitesseM1 \* diviseur);

   diviseur = analogRead(SENS\_GPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V2 = vitesseM2 - (vitesseM2 \* diviseur);

   if (SEUILBLANC > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(vitesseM1);

   }

   if (SEUILBLANC < analogRead(SENS\_DPIN) && SEUILNOIR > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(V1);

   }

   if (SEUILNOIR < analogRead(SENS\_DPIN)) {

     motors.setM1Speed(0);

   }

   if (SEUILBLANC > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(vitesseM2);

   }

   if (SEUILBLANC < analogRead(SENS\_GPIN) && SEUILNOIR > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(V2);

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN)) {

     motors.setM2Speed(0);

   }

   a = hcsr04.distanceInMillimeters();

   if (a < 39) {

     GoToCross();

     Center();

     palet = true;

     GoShoot();

     phase = true;

   }

   if (25 >= analogRead(SENS\_PGPIN) && palet == false) {

     TakePLeft();

     palet = true;

     GoShoot();

     phase = true;

   }

   if (25 >= analogRead(SENS\_PDPIN) && palet == false) {

     TakePRight();

     palet = true;

     GoShoot();

     phase = true;

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN) && SEUILNOIR < analogRead(SENS\_DPIN)) {

     phase = true;

   }

 }

 while (phase == false);

}

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

void Center() {

 bool phase = false;

 do {

   GoUp(50);

   if (SEUILNOIR < analogRead(SENS\_DPIN) || SEUILNOIR < analogRead(SENS\_GPIN)) phase = true;

 }

 while (phase == false);

 GoUp(50);

 phase = false;

 int taux = SEUILNOIR - SEUILBLANC;

 double diviseur = 1;

 do {

   LedCheck();

   diviseur = analogRead(SENS\_DPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V1 = vitesseM1 - (vitesseM1 \* diviseur);

   diviseur = analogRead(SENS\_GPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V2 = vitesseM2 - (vitesseM2 \* diviseur);

   if (SEUILBLANC > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(vitesseM1);

   }

   if (SEUILBLANC < analogRead(SENS\_DPIN) && SEUILNOIR > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(V1);

   }

   if (SEUILNOIR < analogRead(SENS\_DPIN)) {

     motors.setM1Speed(0);

   }

   if (SEUILBLANC > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(vitesseM2);

   }

   if (SEUILBLANC < analogRead(SENS\_GPIN) && SEUILNOIR > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(V2);

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN)) {

     motors.setM2Speed(0);

   }

   if (SEUILNOIR < analogRead(SENS\_CPIN)) {

     phase = true;

   }

 }

 while (phase == false);

}

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

void CenterRC() {

 bool phase = false;

 do {

   GoUp(50);

   if (SEUILNOIR < analogRead(SENS\_DPIN) || SEUILNOIR < analogRead(SENS\_GPIN)) phase = true;

 }

 while (phase == false);

 GoUp(50);

 phase = false;

 int taux = SEUILNOIR - SEUILBLANC;

 double diviseur = 1;

 do {

   LedCheck();

   diviseur = analogRead(SENS\_DPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V1 = vitesseM1 - (vitesseM1 \* diviseur);

   diviseur = analogRead(SENS\_GPIN) - SEUILBLANC;

   diviseur = diviseur / taux;

   int V2 = vitesseM2 - (vitesseM2 \* diviseur);

   if (SEUILBLANC > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(vitesseM1);

   }

   if (SEUILBLANC < analogRead(SENS\_DPIN) && SEUILNOIR > analogRead(SENS\_DPIN)) {

     motors.setM1Speed(V1);

   }

   if (SEUILNOIR < analogRead(SENS\_DPIN)) {

     motors.setM1Speed(0);

   }

   if (SEUILBLANC > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(vitesseM2);

   }

   if (SEUILBLANC < analogRead(SENS\_GPIN) && SEUILNOIR > analogRead(SENS\_GPIN)) {

     motors.setM2Speed(V2);

   }

   if (SEUILNOIR < analogRead(SENS\_GPIN)) {

     motors.setM2Speed(0);

   }

   int a = hcsr04.distanceInMillimeters();

   if (a < 39) {

     Center();

     palet = true;

     GoShoot();

     phase = true;

   }

   if (25 >= analogRead(SENS\_PGPIN) && palet == false) {

     GoDown(400);

     JustRight(300);

     GoUp(100);

     JustLeft(300);

     GoUp(200);

     JustLeft(300);

     do {

       JustRight(1);

     } while (analogRead(SENS\_DPIN) < SEUILNOIR);

     palet = true;

     GoShoot();

     phase = true;

   }

   if (25 >= analogRead(SENS\_PDPIN) && palet == false) {

     GoDown(400);

     JustLeft(300);

     GoUp(100);

     JustRight(300);

     GoUp(200);

     JustRight(300);

     do {

       JustLeft(1);

     } while (analogRead(SENS\_DPIN) < SEUILNOIR);

     palet = true;

     GoShoot();

     phase = true;

   }

   if (SEUILNOIR < analogRead(SENS\_CPIN)) {

     phase = true;

   }

 }

 while (phase == false);

}

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

void CrossRight() {

 bool phase = false;

 GoRight(500);

 do {

   if (SEUIL < analogRead(SENS\_GPIN))phase = true;

   GoRight(1);

 }

 while (phase == false);

}

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

void CrossLeft() {

 bool phase = false;

 GoLeft(500);

 do {

   if (SEUIL < analogRead(SENS\_DPIN)) phase = true;

   GoLeft(1);

 }

 while (phase == false);

}

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

void CrossRightWithP() {

 bool phase = false;

 JustLeft(500);

 do {

   if (SEUIL < analogRead(SENS\_GPIN))phase = true;

   JustLeft(1);

 }

 while (phase == false);

}

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

void CrossLeftWithP() {

 bool phase = false;

 JustRight(500);

 do {

   if (SEUIL < analogRead(SENS\_DPIN)) phase = true;

   JustRight(1);

 }

 while (phase == false);

}

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

void GoUntil(int sensor) {

 bool phase = false;

 do {

   LedCheck();

   motors.setM2Speed(vitesseM2);

   motors.setM1Speed(vitesseM1);

   if (SEUILBLANC < analogRead(sensor)) phase = true;

 }

 while (phase == false);

}

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

void Join(int sensor) {

 bool phase = false;

 do {

   LedCheck();

   switch (sensor) {

     case SENS\_DPIN:

       if (SEUIL > analogRead(SENS\_GPIN)) {

         motors.setM2Speed(vitesseM2);

         motors.setM1Speed(-vitesseM1 \* 0.5);

       }

       if (SEUIL < analogRead(SENS\_GPIN)) GoUp(1);

       if (SEUIL > analogRead(SENS\_GPIN) && SEUIL > analogRead(SENS\_DPIN))  phase = true;

       break;

     case SENS\_GPIN:

       if (SEUIL > analogRead(SENS\_DPIN))  {

         motors.setM2Speed(-vitesseM2 \* 0.5);

         motors.setM1Speed(vitesseM1);

       }

       if (SEUIL < analogRead(SENS\_DPIN)) GoUp(1);

       if (SEUIL > analogRead(SENS\_DPIN) && SEUIL > analogRead(SENS\_GPIN)) phase = true;

       break;

     default:

       break;

   }

 }

 while (phase == false);

}

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

void GoPerp() {

 bool phase = false;

 int sensor;

 do {

   LedCheck();

   motors.setM2Speed(vitesseM2);

   motors.setM1Speed(vitesseM1);

   if (SEUIL < analogRead(SENS\_GPIN)) {

     phase = true;

     sensor = SENS\_GPIN;

   }

   if (SEUIL < analogRead(SENS\_DPIN)) {

     phase = true;

     sensor = SENS\_DPIN;

   }

 }

 while (phase == false);

 phase = false;

 do {

   LedCheck();

   switch (sensor) {

     case SENS\_DPIN:

       if (SEUIL < analogRead(SENS\_DPIN))JustRightBack(1);

       if (SEUIL > analogRead(SENS\_DPIN))JustLeft(1);

       break;

     case SENS\_GPIN:

       if (SEUIL < analogRead(SENS\_GPIN))JustLeftBack(1);

       if (SEUIL > analogRead(SENS\_GPIN))JustRight(1);

       break;

   }

   if (SEUIL < analogRead(SENS\_DPIN) && SEUIL < analogRead(SENS\_GPIN))phase = true;

 }

 while (phase == false);

}

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

void TakePRight()

{

 GoDown(400);

 JustLeft(320);

 GoUp(100);

 do {

   JustRight(1);

 }

 while (analogRead(SENS\_DPIN) < SEUILNOIR);

 GoToCross();

 vitesseM1 = 70;

 vitesseM2 = 70;

 GoUp(400);

 do {

   GoRight(1);

 }

 while (analogRead(SENS\_GPIN) < SEUILNOIR);

 vitesseM1 = 110;

 vitesseM2 = 110;

}

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

void TakePLeft()

{

 GoDown(400);

 JustRight(300 );

 GoUp(100);

 do {

   JustLeft(1);

 }

 while (analogRead(SENS\_GPIN) < SEUILNOIR);

 GoToCross();

 vitesseM1 = 70;

 vitesseM2 = 70;

 GoUp(400);

 do {

   GoLeft(1);

 }

 while (analogRead(SENS\_DPIN) < SEUILNOIR);

 vitesseM1 = 110;

 vitesseM2 = 110;

}

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

//FONCTIONS SIMPLES

void LedCheck() {

 if (SEUIL > analogRead(SENS\_DPIN)) digitalWrite(LED\_C, HIGH);

 else digitalWrite(LED\_C, LOW);

 while (**Serial**.available()) {

   delay(3);

   char c = **Serial**.read();

   readString += c;

 }

 if (readString.length() > 0) {

**Serial**.println(readString);

   if (readString == "s") {

     motors.setM2Speed(0);

     motors.setM1Speed(0);

     digitalWrite(LED\_C, LOW);

     digitalWrite(LED\_P, LOW);

     digitalWrite(BOUTON, HIGH);

     digitalWrite(SOL, HIGH);

     digitalWrite(MODE\_PIN, HIGH);

     asm volatile ("jmp 0");

   }

   readString = "";

 }

 int a = hcsr04.distanceInMillimeters();

 delay(1);

 countB++;

 if (39 > a) {

   digitalWrite(LED\_P, HIGH);

 }

 else {

   digitalWrite(LED\_P, LOW);

 }

 if (countB >= 400) {

   if (39 >= a && tir == false) **Serial**.println("d");

   if (39 < a && tir == false) **Serial**.println("n");

   if (tir == true) {

**Serial**.println("t");

     countT++;

   }

   if (countT >= 5) {

     countT = 0;

     tir == false;

   }

   countB = 0;

 }

}

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

void GoUp(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(vitesseM2);

   motors.setM1Speed(vitesseM1);

   count++;

 }

 while (count < temp);

}

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

void GoDown(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(-vitesseM2);

   motors.setM1Speed(-vitesseM1);

   count++;

 }

 while (count < temp);

}

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

void GoRight(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(vitesseM2 \* 0.75);

   motors.setM1Speed(-vitesseM1 \* 0.75);

   count++;

 }

 while (count < temp);

}

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

void GoLeft(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(-vitesseM2 \* 0.75);

   motors.setM1Speed(vitesseM1 \* 0.75);

   count++;

 }

 while (count < temp);

}

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

void JustRight(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(0);

   motors.setM1Speed(vitesseM1);

   count++;

 }

 while (count < temp);

}

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

void JustRightBack(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(0);

   motors.setM1Speed(-vitesseM1);

   count++;

 }

 while (count < temp);

}

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

void JustLeft(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(vitesseM2);

   motors.setM1Speed(0);

   count++;

 }

 while (count < temp);

}

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

void JustLeftBack(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(-vitesseM2);

   motors.setM1Speed(0);

   count++;

 }

 while (count < temp);

}

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

void Stop(int temp) {

 int count = 0;

 do {

   LedCheck();

   motors.setM2Speed(0);

   motors.setM1Speed(0);

   count++;

 }

 while (count < temp);

}

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

void Shoot() {

 int temp = 50;

 int count = 0;

 tir = true;

 do {

   LedCheck();

   count++;

   if (count > 30)digitalWrite(SOL, LOW);

   delay(1);

 }

 while (count < temp);

 Stop(1);

 delay(100);

 digitalWrite(SOL, HIGH);

}