#include <SoftwareSerial.h>

#include <Pixy2.h>

#include <Servo.h>

// Definir pines

const int motorIn1 = 5;

const int motorIn2 = 6;

const int motorEn = 3;

const int servoDireccionPin = 9;

SoftwareSerial lidarSerial(18, 19); // RX, TX

Pixy2 pixy;

Servo servoDireccion;

int vueltasCompletadas = 0;

bool estacionado = false;

bool bloqueMagentaEncontrado = false;

// Distancia deseada para seguir el bloque (en cm)

const int distanciaDeseada = 30;

void setup() {

  Serial.begin(9600);

  lidarSerial.begin(115200);

  pixy.init();

  servoDireccion.attach(servoDireccionPin);

  pinMode(motorIn1, OUTPUT);

  pinMode(motorIn2, OUTPUT);

  pinMode(motorEn, OUTPUT);

}

void loop() {

  if (!estacionado) {

    if (vueltasCompletadas < 3) {

      leerSensoresYEsquivar();

      contarVueltas();

    } else {

      if (bloqueMagentaEncontrado) {

        buscarYEstacionarse();

      } else {

        parar();

        estacionado = true;

      }

    }

  }

}

// Funciones para Controlar Motores y Servos

void adelante() {

  digitalWrite(motorIn1, HIGH);

  digitalWrite(motorIn2, LOW);

  analogWrite(motorEn, 255); // Velocidad máxima

}

void atras() {

  digitalWrite(motorIn1, LOW);

  digitalWrite(motorIn2, HIGH);

  analogWrite(motorEn, 255); // Velocidad máxima

}

void parar() {

  digitalWrite(motorIn1, LOW);

  digitalWrite(motorIn2, LOW);

  analogWrite(motorEn, 0); // Parar

}

void girarIzquierda() {

  servoDireccion.write(45); // Ajustar el valor según sea necesario

}

void girarDerecha() {

  servoDireccion.write(135); // Ajustar el valor según sea necesario

}

void centrarDireccion() {

  servoDireccion.write(90); // Posición central

}

// Funciones para Leer Datos de los Sensores

int leerDistanciaLidar() {

  int distancia = 0;

  if (lidarSerial.available()) {

    // Leer y procesar la distancia del Lidar

    while (lidarSerial.available() > 0) {

      int c = lidarSerial.read();

      distancia = (distancia << 8) | c;

    }

  }

  return distancia;

}

void moverPixy(int servoX, int servoY) {

  // Mover la cámara Pixy2 según las coordenadas del bloque

  int centroX = 160; // Valor central de la imagen (320/2)

  int offset = 20; // Valor de offset para la zona muerta

  if (servoX < centroX - offset) {

    // Bloque a la izquierda, girar a la izquierda

    girarIzquierda();

  } else if (servoX > centroX + offset) {

    // Bloque a la derecha, girar a la derecha

    girarDerecha();

  } else {

    // Bloque centrado, mantener la dirección actual

    centrarDireccion();

  }

}

void seguirBloque(int distanciaActual) {

  if (distanciaActual > distanciaDeseada + 5) {

    adelante();

  } else if (distanciaActual < distanciaDeseada - 5) {

    atras();

  } else {

    parar();

  }

}

void leerSensoresYEsquivar() {

  int distanciaFrontal = leerDistanciaLidar();

  // Evitar chocar con los bordes de la pista

  if (distanciaFrontal < 30) {

    girarDerecha();

    adelante();

    delay(1000); // Ajustar tiempo según sea necesario

    centrarDireccion();

  }

  // Leer datos del Pixy2

  pixy.ccc.getBlocks();

  if (pixy.ccc.numBlocks) {

    bool bloqueDetectado = false;

    for (int i = 0; i < pixy.ccc.numBlocks; i++) {

      if (pixy.ccc.blocks[i].m\_signature == 1) { // Supongamos que 1 es verde

        moverPixy(pixy.ccc.blocks[i].m\_x, pixy.ccc.blocks[i].m\_y); // Mover la cámara hacia el bloque verde

        int distanciaActual = leerDistanciaLidar();

        seguirBloque(distanciaActual);

        bloqueDetectado = true;

        break; // Salir del bucle al detectar un bloque verde

      } else if (pixy.ccc.blocks[i].m\_signature == 2) { // Supongamos que 2 es rojo

        moverPixy(pixy.ccc.blocks[i].m\_x, pixy.ccc.blocks[i].m\_y); // Mover la cámara hacia el bloque rojo

        int distanciaActual = leerDistanciaLidar();

        seguirBloque(distanciaActual);

        bloqueDetectado = true;

        break; // Salir del bucle al detectar un bloque rojo

      } else if (pixy.ccc.blocks[i].m\_signature == 3) { // Supongamos que 3 es magenta

        bloqueMagentaEncontrado = true;

        moverPixy(pixy.ccc.blocks[i].m\_x, pixy.ccc.blocks[i].m\_y); // Mover la cámara hacia el bloque magenta

        int distanciaActual = leerDistanciaLidar();

        seguirBloque(distanciaActual);

        bloqueDetectado = true;

      }

    }

    // Si no se detectó un bloque rojo o verde, mantener la dirección actual

    if (!bloqueDetectado) {

      centrarDireccion();

    }

  } else {

    centrarDireccion();

  }

}

// Lógica de Navegación para Esquivar Obstáculos

void esquivarObstaculo() {

  parar();

  girarDerecha();

  adelante();

  delay(1000); // Ajustar tiempo según sea necesario

  centrarDireccion();

}

// Contador de Vueltas

void contarVueltas() {

  static int posicionInicial = leerPosicionInicial();

  int posicionActual = leerPosicionActual();

  if (posicionActual == posicionInicial) {

    vueltasCompletadas++;

  }

}

int leerPosicionInicial() {

  int posicion = 0;

  // Código para leer y retornar la posición inicial

  return posicion;

}

int leerPosicionActual() {

  int posicion = 0;

  // Código para leer y retornar la posición actual

  return posicion;

}

// Buscar y Estacionarse en el Bloque Magenta

void buscarYEstacionarse() {

  unsigned long tiempoInicio = millis();

  bool bloqueMagentaEncontrado = false;

  int contadorMagenta = 0;

  while (millis() - tiempoInicio < 30000) { // Buscar durante 30 segundos

    pixy.ccc.getBlocks();

    if (pixy.ccc.numBlocks) {

      for (int i = 0; i < pixy.ccc.numBlocks; i++) {

        if (pixy.ccc.blocks[i].m\_signature == 3) { // Supongamos que 3 es magenta

          contadorMagenta++;

          if (contadorMagenta >= 2) {

            bloqueMagentaEncontrado = true;

            break;

          }

        }

      }

    }

    if (bloqueMagentaEncontrado) break;

    adelante();

  }

  if (bloqueMagentaEncontrado) {

    parar();

    estacionarEntreBloquesMagenta();

  } else {

    parar();

    estacionado = true;

  }

}

void estacionarEntreBloquesMagenta() {

  unsigned long tiempoInicio = millis();

  while (millis() - tiempoInicio < 5000) { // Estacionar durante 5 segundos

    int distanciaFrontal = leerDistanciaLidar();

    if (distanciaFrontal < 30) {

      girarDerecha();

      adelante();

      delay(1000); // Ajustar tiempo según sea necesario

      centrarDireccion();

    } else {

      adelante();

    }

  }

  parar();

  estacionado = true; // Marcar como estacionado

}