#include <QTRSensors.h>

#include <OrangutanMotors.h>

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

//PD line sensors

#define NUM\_SENSORS 6 // Number of sensors it uses

#define NUM\_SAMPLES\_PER\_SENSOR 5 // Samples per sensor

#define EMITTER\_PIN 2 // emitter is controlled by digital pin 2

// sensors 0 through 5 are connected to analog inputs 0 through 5, respectively

QTRSensorsAnalog qtra((unsigned char[]) {0, 1, 2, 3, 4, 5},

NUM\_SENSORS, NUM\_SAMPLES\_PER\_SENSOR, EMITTER\_PIN);

unsigned int sensorValues[NUM\_SENSORS];

OrangutanMotors motors;

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

//Maximum speed Robot//--------------------------//----------------------------------//

const int maximum = 80;

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

//PD VALUES//----------//PD VALUES//--------------------------------------------------//

int VProporcional = 1.45;

int VDerivativo = 16;

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

//Calibration Velocity

int velcalibrate = 20;

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

void setup()

{

int inPin = 10;

int val = 0;

pinMode(9, OUTPUT);

pinMode(8, OUTPUT);

pinMode(inPin,INPUT);

val = digitalRead(inPin);

while (val == HIGH)

{

digitalWrite(9, HIGH);

digitalWrite(8, HIGH);

val = digitalRead(inPin);

};

if (val == LOW)

{

digitalWrite(9, LOW);

digitalWrite(8, LOW);

};

motors.setSpeeds(0,0);// Motors stopped

//-------------Instructions for starting to make the sensors calibration--------------------------------------//

delay(1500);

digitalWrite(9, HIGH);

digitalWrite(8, HIGH);// Turns on the Leds, indicating it is calibrating.

for (int counter=0; counter<21; counter++)

{

if (counter < 6 || counter >= 15)

OrangutanMotors::setSpeeds(-velcalibrate, velcalibrate);

else

OrangutanMotors::setSpeeds(velcalibrate, -velcalibrate);

qtra.calibrate();

delay(20);

}

digitalWrite(9, LOW); // Turns off the leds, indicating calibration is over.

digitalWrite(8, LOW);

OrangutanMotors::setSpeeds(0, 0);

delay(200);

digitalWrite(9, HIGH);

digitalWrite(8, HIGH);

delay(200);

digitalWrite(9, LOW); // Blinking, indicating the robot is ready.

digitalWrite(8, LOW);

delay(200); // Blinking, indicating the robot is ready.

digitalWrite(9, HIGH);

digitalWrite(8, HIGH); // Blinking, indicating the robot is ready.

delay(200);

digitalWrite(9, LOW); // Blinking, indicating the robot is ready.

digitalWrite(8, LOW);

delay(200);

//---------------------------End sensors calibration----------------------------------------------------//

pinMode(inPin,INPUT);

val = digitalRead(inPin);

while (val == HIGH)

{

digitalWrite(9, HIGH);

digitalWrite(8, HIGH);

val = digitalRead(inPin);

};

if (val == LOW)

{

digitalWrite(9, LOW);

digitalWrite(8, LOW);

delay(1000); // Delay X seconds before it stars to move

};

}

unsigned int last\_proportional = 0;

long integral = 0;

void loop()

{

unsigned int position = qtra.readLine(sensorValues); // reads position of the line in the position variable

//Reference where it will follow the line, half sensors

int proportional = (int)position - 2500;

// PD Calculus

int derivative = proportional - last\_proportional;

integral += proportional;

last\_proportional = proportional;

int power\_difference = proportional/VProporcional + integral\*0 + derivative\*VDerivativo;

if (power\_difference > maximum)

power\_difference = maximum;

if (power\_difference < -maximum)

power\_difference = -maximum;

if (power\_difference < 0)

OrangutanMotors::setSpeeds(maximum, maximum + power\_difference);

else

OrangutanMotors::setSpeeds(maximum - power\_difference,maximum);

};