#include <NewPing.h>

#define TRIGGER\_PINL A3 // Arduino pin tied to trigger pin on ping sensor.

#define ECHO\_PINL A0 // Arduino pin tied to echo pin on ping sensor.

#define MAX\_DISTANCE 100 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

#define TRIGGER\_PINF A4 // Arduino pin tied to trigger pin on ping sensor.

#define ECHO\_PINF A1 // Arduino pin tied to echo pin on ping sensor.

#define TRIGGER\_PINR A5 // Arduino pin tied to trigger pin on ping sensor.

#define ECHO\_PINR A2 // Arduino pin tied to echo pin on ping sensor.

int dir;

#define STOP 0

#define FORWARD 1

#define BACKWARD 2

#define LEFT 3

#define RIGHT 4

float P = 0.7 ;

float D = 0.5 ;

float I = 0.4 ;

float oldErrorP ;

float totalError ;

int offset = 5 ;

int wall\_threshold = 13 ;

//int left\_threshold = 10 ;

//int right\_threshold = 10 ;

int front\_threshold = 7 ;

boolean frontwall ;

boolean leftwall ;

boolean rightwall ;

boolean first\_turn ;

boolean rightWallFollow ;

boolean leftWallFollow ;

int en1 = 2 ;

int en2 = 3 ;

int en3 = 4 ;

int en4 = 5 ;

int enA = 10 ;

int enB = 11 ;

int baseSpeed = 70 ;

int RMS ;

int LMS ;

int LED = 13 ;

int led1 = 8 ;

int led2 = 9 ;

NewPing sonarLeft(TRIGGER\_PINL, ECHO\_PINL, MAX\_DISTANCE); // NewPing setup of pins and maximum distance.

NewPing sonarRight(TRIGGER\_PINR, ECHO\_PINR, MAX\_DISTANCE);

NewPing sonarFront(TRIGGER\_PINF, ECHO\_PINF, MAX\_DISTANCE);

unsigned int pingSpeed = 30; // How frequently are we going to send out a ping (in milliseconds). 50ms would be 20 times a second.

unsigned long pingTimer; // Holds the next ping time.

float oldLeftSensor, oldRightSensor, leftSensor, rightSensor, frontSensor, oldFrontSensor, lSensor, rSensor, fSensor;

//int TestNUM = 1 ;

void setup() {

Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.

for (int i = 2; i <= 13; i++) //For Motor Shield

pinMode(i, OUTPUT);

first\_turn = false ;

rightWallFollow = false ;

leftWallFollow = false ;

}

void loop() {

//========================================START========================================//

ReadSensors();

walls();

if ( first\_turn == false ) {

pid\_start();

}

else if (leftWallFollow == true ) {

PID(true) ;

}

else if (rightWallFollow == true ) {

PID(false) ;

}

if (leftwall == true && rightwall == false && frontwall == true ) {

// turnright();

PID(false) ;

if ( first\_turn == false ) {

// right\_threshold = right\_threshold - offset ;

// left\_threshold = left\_threshold + offset ;

first\_turn = true ;

rightWallFollow = true ;

digitalWrite(led2 , LOW );

digitalWrite(led1 ,HIGH );

}

}

if (leftwall == false && rightwall == true && frontwall == true ) {

// turnleft();

PID(true) ;

if ( first\_turn == false ) {

// left\_threshold = left\_threshold - offset ;

// right\_threshold = right\_threshold + offset ;

first\_turn = true ;

leftWallFollow = true ;

digitalWrite(LED , HIGH);

}

}

if ( leftSensor == 0 || leftSensor > 100 && rightSensor == 0 || rightSensor > 100 && frontSensor == 0 || frontSensor > 100 ) {

setDirection(STOP);

}

// read sensors & print the result to the serial monitor //

Serial.print(" Left : ");

Serial.print(leftSensor);

Serial.print(" cm ");

Serial.print(" Right : ");

Serial.print(rightSensor);

Serial.print(" cm ");

Serial.print(" Front : ");

Serial.print(frontSensor);

Serial.println(" cm ");

//measure error & print the result to the serial monitor

Serial.print("error=");

Serial.println(totalError);

}

//--------------------------------- direction control ---------------------------------//

void setDirection(int dir) {

if ( dir == FORWARD ) {

digitalWrite(en1, LOW); // Left wheel forward

digitalWrite(en2, HIGH);

digitalWrite(en3, LOW); // Right wheel forward

digitalWrite(en4, HIGH);

}

else if ( dir == LEFT ) {

digitalWrite(en1, HIGH); // Left wheel forward

digitalWrite(en2, LOW );

digitalWrite(en3, LOW ); // Right wheel forward

digitalWrite(en4, HIGH);

}

else if ( dir == RIGHT ) {

digitalWrite(en1, LOW); // Left wheel forward

digitalWrite(en2, HIGH);

digitalWrite(en3, HIGH); // Right wheel forward

digitalWrite(en4, LOW);

}

else if ( dir == STOP ) {

digitalWrite(en1, HIGH); // Left wheel forward

digitalWrite(en2, HIGH );

digitalWrite(en3, HIGH ); // Right wheel forward

digitalWrite(en4, HIGH);

}

else if ( dir == BACKWARD ) {

digitalWrite(en1, HIGH ); // Left wheel forward

digitalWrite(en2, LOW );

digitalWrite(en3, HIGH ); // Right wheel forward

digitalWrite(en4, LOW );

}

}

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

//--------------------------------- Sensors ---------------------------------//

void ReadSensors() {

//leftSensor = sonarLeft.ping\_median(TestNUM); //accurate but slow

//rightSensor = sonarRight.ping\_median(TestNUM); //accurate but slow

//frontSensor = sonarFront.ping\_median(TestNUM); //accurate but slow

//leftSensor = sonarLeft.convert\_cm(leftSensor);

//rightSensor = sonarRight.convert\_cm(rightSensor);

//frontSensor = sonarFront.convert\_cm(frontSensor);

lSensor = sonarLeft.ping\_cm(); //ping in cm

rSensor = sonarRight.ping\_cm();

fSensor = sonarFront.ping\_cm();

leftSensor = (lSensor + oldLeftSensor) / 2; //average distance between old & new readings to make the change smoother

rightSensor = (rSensor + oldRightSensor) / 2;

frontSensor = (fSensor + oldFrontSensor) / 2;

oldLeftSensor = leftSensor; // save old readings for movment

oldRightSensor = rightSensor;

oldFrontSensor = frontSensor;

}

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

//--------------------------------- control ---------------------------------//

void pid\_start() {

//ReadSensors()

float errorP = leftSensor - rightSensor ;

float errorD = errorP - oldErrorP;

float errorI = (2.0 / 3.0) \* errorI + errorP ;

totalError = P \* errorP + D \* errorD + I \* errorI ;

oldErrorP = errorP ;

RMS = baseSpeed + totalError ;

LMS = baseSpeed - totalError ;

// if(RMS < -255) RMS = -255; if(RMS > 255)RMS = 255 ;

// if(LMS < -255) LMS = -255; if(LMS > 255)LMS = 255 ;

if (RMS < 0) {

RMS = map(RMS , 0 , -255, 0, 255);

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(RIGHT);

}

else if (LMS < 0) {

LMS = map(LMS , 0 , -255, 0, 255);

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(LEFT);

}

else {

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(FORWARD);

}

}

//----------------------------- wall follow control -------------------------------//

void PID( boolean left ) {

if (left == true ) {

float errorP = leftSensor - rightSensor - offset ;

float errorD = errorP - oldErrorP;

float errorI = (2.0 / 3) \* errorI + errorP ;

totalError = P \* errorP + D \* errorD + I \* errorI ;

oldErrorP = errorP ;

RMS = baseSpeed + totalError ;

LMS = baseSpeed - totalError ;

// if(RMS < -255) RMS = -255; if(RMS > 255)RMS = 255 ;

// if(LMS < -255) LMS = -255; if(LMS > 255)LMS = 255 ;

if (RMS < 0) {

RMS = map(RMS , 0 , -255, 0, 255);

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(RIGHT);

}

else if (LMS < 0) {

LMS = map(LMS , 0 , -255, 0, 255);

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(LEFT);

}

else {

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(FORWARD);

}

}

else {

float errorP = leftSensor - rightSensor + offset ;

float errorD = errorP - oldErrorP;

float errorI = (2.0 / 3) \* errorI + errorP ;

totalError = P \* errorP + D \* errorD + I \* errorI ;

oldErrorP = errorP ;

RMS = baseSpeed + totalError ;

LMS = baseSpeed - totalError ;

// if(RMS < -255) RMS = -255; if(RMS > 255)RMS = 255 ;

// if(LMS < -255) LMS = -255; if(LMS > 255)LMS = 255 ;

if (RMS < 0) {

RMS = map(RMS , 0 , -255, 0, 255);

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(RIGHT);

}

else if (LMS < 0) {

LMS = map(LMS , 0 , -255, 0, 255);

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(LEFT);

}

else {

analogWrite(enA , RMS);

analogWrite(enB , LMS);

setDirection(FORWARD);

}

}

}

//--------------------------- wall detection --------------------------------//

void walls() {

if ( leftSensor < wall\_threshold ) {

leftwall = true ;

}

else {

leftwall = false ;

}

if ( rightSensor < wall\_threshold ) {

rightwall = true ;

}

else {

rightwall = false ;

} if ( frontSensor < front\_threshold ) {

frontwall = true ;

}

else {

frontwall = false ;

}

}

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

void turnright() {

LMS = baseSpeed ;

RMS = LMS \* rightSensor / ( rightSensor + 11 ) ;

}

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

void turnleft() {

RMS = baseSpeed ;

LMS = RMS \* leftSensor / ( leftSensor + 11 ) ;

}

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