#include <PID\_v1.h>

double Setpoint, Input, Output;

double kp = 0.1;

double ki = 0.0;

double kd = 0.0;

PID myPID(&Input, &Output, &Setpoint, kp, ki, kd, DIRECT);

boolean cwccw; //Boolean Data Type holds values that can be only **True** or **False**

byte pin\_sensor[2] = {A0,A1}; /\*array for 2 sensors connected on pins A0 and A1. Data type byte: Holds unsigned 8-bit (1-byte) integers that range in value from 0 through 255. Use the **Byte** data type to contain binary data.The default value of **Byte** is 0.\*/

int val\_sensor[2] = {0,0}; //array to store value of sensors

byte pin\_sig[2][2] = { //a matrix for the 4 pins on which motor is connected

{10,9}, //motor0 : pin10,9 PWM+LOW or LOW+PWM → CW or

{5,6} //motor1 : pin 5,6 PWM+LOW or LOW+PWM → CW or CCW

};

boolean f\_right;

int power = 40;

int val\_max;

int val\_min = 2046;

boolean monitor = true; //on/off

boolean m\_stop = false; //on/off

#define led\_0 2 //first led connected on digital pin 2

#define led\_1 3 //second led connected on digital pin 3

void setup()

{

Serial.begin(115200);

pinMode(pin\_sensor[0], INPUT); //sensor pins

pinMode(pin\_sensor[1], INPUT);

pinMode(pin\_sig[0][0], OUTPUT); //motor pins

pinMode(pin\_sig[0][1], OUTPUT);

pinMode(pin\_sig[1][0], OUTPUT);

pinMode(pin\_sig[1][1], OUTPUT);

pinMode(led\_0, OUTPUT); //LED pins

pinMode(led\_1, OUTPUT);

digitalWrite(led\_0, HIGH); //marks the beginning of configuration period for sensors

digitalWrite(led\_1, HIGH);

long st = millis();

while(millis() - st < 3000) //through this loop we are determining the min and max sum of the values of 2 sensors

{

val\_sensor[0] = analogRead(pin\_sensor[0]);

val\_sensor[1] = analogRead(pin\_sensor[1]);

if(val\_sensor[0] + val\_sensor[1] < val\_min)

{

val\_min = val\_sensor[0] + val\_sensor[1];

}

if(val\_sensor[0] + val\_sensor[1] > val\_max)

{

val\_max = val\_sensor[0] + val\_sensor[1];

}

}

Serial.println("");

Serial.println(val\_min);

Serial.println(val\_max);

delay(2000);

digitalWrite(led\_0, LOW);

digitalWrite(led\_1, LOW); //marks the end of configuration period for sensors

myPID.SetOutputLimits(0, power);

myPID.SetSampleTime(10); //Determines how often the PID algorithm evaluates

Input = 0;

Setpoint = val\_max;

myPID.SetMode(AUTOMATIC);

motor(0, 50, false); //assigns ccwc = false and motor 1 starts moving with pwm=50 in 1 direction

delay(1000);

motor\_stop(0,0); //this will stop the motor 1

motor(1, 50, false); //assigns ccwc = false and motor 2 starts moving with pwm=50 in 1 direction

delay(1000);

motor\_stop(1,0); //this will stop the motor 2

delay(1000);

}

void loop()

{

if(Serial.available())

{

ser\_rcv();

myPID.SetTunings(kp, ki, kd);

myPID.SetOutputLimits(0, power);

}

for(byte i=0;i < 2;i++){

val\_sensor[i] = analogRead(pin\_sensor[i]);

}

int val\_diff = val\_sensor[0] - val\_sensor[1];

int val\_sum = val\_sensor[0] + val\_sensor[1];

Input = val\_sum;

myPID.Compute();

if(val\_sum > val\_max / 2 && val\_diff > 0){

f\_right = true;

}

else if(val\_sum > val\_max / 2 && val\_diff <= 0)

{

f\_right = false;

}

if(f\_right == true){

digitalWrite(led\_0, HIGH);

motor(0, power - Output, false);

motor(1, power, false);

digitalWrite(led\_0, LOW);

}

else{

digitalWrite(led\_1, HIGH);

motor(0, power, false);

motor(1, power - Output, false);

digitalWrite(led\_1, LOW);

}

if(monitor == true){

Serial.print(val\_sensor[0]);

Serial.print("\t");

Serial.print(val\_sensor[1]);

Serial.print("\t");

Serial.print(val\_diff); //val\_sensor[0] - val\_sensor[1]

Serial.print("\t");

Serial.print(val\_sum); //val\_sensor[0] + val\_sensor[1]

Serial.print("\t");

Serial.print(f\_right);

Serial.print("\t");

Serial.print(power);

Serial.print("\t");

Serial.print(Output);

Serial.print("\t");

Serial.print(power - Output);

Serial.print("\t");

Serial.print(kp, 2); //Kp

Serial.print("\t");

Serial.print(ki, 2); //Ki

Serial.print("\t");

Serial.print(kd, 2); //Kd

Serial.print("\t");

Serial.println("");

}

}

void motor(byte num, byte pw, boolean cw)

{

pw = map(pw, 0, 100, 0, 255);

if(m\_stop == false) //if motor is moving

{

if(cwccw != cw) /\*it checks if the motor is currently moving in the opposite direction to the indicated direction and if it is so then it stops the motor (num)\*/

{

digitalWrite(pin\_sig[num][0],LOW);

digitalWrite(pin\_sig[num][1],LOW);

delay(1);

}

if(cw == true) //assigns the pwm value(pw) to the motor(num) in the one direction

{

cwccw = true;

analogWrite(pin\_sig[num][0], pw);

analogWrite(pin\_sig[num][1], 0);

}

else ////assigns the pwm value(pw) to the motor(num) in the one direction

{

cwccw = false;

analogWrite(pin\_sig[num][0], 0);

analogWrite(pin\_sig[num][1], pw);

}

}

else //if motor is not moving

{

motor\_stop(0, 0);

motor\_stop(1, 0);

}

}

void motor\_stop(byte num,int dur)

{

digitalWrite(pin\_sig[num][0],LOW);

digitalWrite(pin\_sig[num][1],LOW);

if(dur >0){

delay(dur);

}

}

void ser\_rcv()

{

float k\_step = 0.01;

char c = Serial.read();

if(c == 'q') //Kp +

{

kp += k\_step;

}

else if(c == 'a') //Kp -

{

kp -= k\_step;

if(kp < 0)

{

kp = 0;

}

}

else if(c == 'w') //Ki +

{

ki += k\_step;

}

else if(c == 's'){ //Ki -

ki -= k\_step;

if(ki < 0)

{

ki = 0;

}

}

else if(c == 'e') //Kd +

{

kd += k\_step;

}

else if(c == 'd') //Kd -

{

kd -= k\_step;

if(kd < 0)

{

kd = 0;

}

}

else if(c == 'r') //motor power +

{

power += 5;

if(power > 100){

power = 100;

}

}

else if(c == 'f') //motor power -

{

power -= 5;

if(power < 0){

power = 0;

}

}

else if(c == 'z'){ //serial monitor output on/off

monitor = !monitor;

}

else if(c == 'x'){ //motor on/off

m\_stop = !m\_stop;

}

}

#define LSensor1 3

#define LSensor2 4

#define LSensor3 5

#define LSensor4 6

#define LSensor5 A0

#define LSensor6 A1

#define LSensor7 A2

#define LSensor8 A3

#define motor\_L1 7

#define motor\_R1 8

#define pwm1 9

#define pwm2 10

#define motor\_L2 11

#define motor\_R2 12

// Variable for line checking

int Position=0, correctPosition=15, totalPosition=0;

int error=0, lastError=0;

int timerZero=0, timeStopA=0, timeStopB=0;

// Variable for PID

int integral=0;

int derivative=0;

int turn,turn\_L,turn\_R;

// debounce time for line checking

unsigned long timeLSensor, debounceDelayLSensor = 15;

// Setup the robot to move 15s.

unsigned long timeToStop = 15000;

void setup()

{

pinMode(LSensor1,INPUT);

pinMode(LSensor2,INPUT);

pinMode(LSensor3,INPUT);

pinMode(LSensor4,INPUT);

pinMode(LSensor5,INPUT);

pinMode(LSensor6,INPUT);

pinMode(LSensor7,INPUT);

pinMode(LSensor8,INPUT);

pinMode(motor\_L1,OUTPUT);

pinMode(motor\_R1,OUTPUT);

pinMode(pwm1,OUTPUT);

pinMode(pwm2,OUTPUT);

pinMode(motor\_L2,OUTPUT);

pinMode(motor\_R2,OUTPUT);

timeLSensor = millis();

Serial.begin(9600);

Serial.println("SETUP DONE");

}

void loop()

{

// Stop the motor after 15s.

if (millis() > timeToStop)

{

StopRun();// Stop the motors.

/\*

Serial.print("pwm1 : ");

Serial.println("0");

Serial.print("pwm2 : ");

Serial.println("0");

Serial.println("MOTOR STOPPED");

\*/

while(1); // The loop end.

}

else

lineFollower(5,10,30,50,40);

}

// Make the robot track to the line.

void lineFollower(int kKp, int kKi, int kKd, int speed\_L, int speed\_R)

{

lineCheck();

PID(kKp,kKi,kKd);

go\_Straight(speed\_L + turn\_L, speed\_R - turn\_R);

}

// Check the position of the robot.

void lineCheck()

{

// Check line for every 15ms.

if ( millis() - timeLSensor >= debounceDelayLSensor )

{

// Take the reading of 8 line IR sensor.

int s = digitalRead(LSensor1);

int t = digitalRead(LSensor2);

int u = digitalRead(LSensor3);

int v = digitalRead(LSensor4);

int w = digitalRead(LSensor5);

int x = digitalRead(LSensor6);

int y = digitalRead(LSensor7);

int z = digitalRead(LSensor8);

// Calculate the position of the robot.

error=lastError ;

// When no line is detected.

if (s==0 && t==0 && u==0 && v==0 && w==0 && x==0 && y==0 && z==0)

{

// Setup a timer

if (timeStopB > 0)

timerZero += timeStopA - timeStopB ;

timeStopB = timeStopA;

timeStopA = millis();

if (lastError == 0)

{

/\*StopRun();

Serial.println("\*\*\*Stop\*\*\*");\*/

**Position = 15;**

}

// Robot stops when it away from line more than 3 second.

while (timerZero > 1500)

StopRun();

}

// When line is detected.

else

{

// Reset the timer when it detects the line again.

timerZero=0, timeStopA=0, timeStopB=0;

// Calculate the position of the robot.

**totalPosition = s + t + u + v + w + x + y + z;**

//1,4,6,7,8,9,11,14

**Position = ( 2\*s + 8\*t + 12\*u + 14\*v + 16\*w + 18\*x + 22\*y + 28\*z ) / totalPosition;**

}

timeLSensor = millis();

error = Position - correctPosition;

/\*

Serial.print("Position : "); // Show value of position on laptop.

Serial.println(Position);

Serial.print("lastError : "); // Show value of last position on laptop.

Serial.println(lastError);

Serial.print("error : "); // Show value of error on laptop.

Serial.println(error);

Serial.println("------------");\*/

}

}

// Using PID calculate the error.

void PID(int Kp, int Ki, int Kd)

{

//---PID---

// For P

error = error;

/\*

Serial.print(Position), Serial.print(" ; ");

Serial.print(error), Serial.print(" ; ");

Serial.print(lastError), Serial.println(" ; ");

\*/

// For I

// For remains the same integral error value. (error = last error)

integral += error;

integral = constrain(integral,0,100);

/\* For error of position changed.

E.g. last time error is 2, now error is one. \*/

if ( error != lastError )

{ /\*

// For +ve position value.

if ( Position > 0 )

{

if ( Position > lastPosition ) // away from the line.

integral = 0;

else

integral = 0; // towards to the line.

}

// For -ve position value.

else if ( Position < 0 )

{

if ( Position < lastPosition ) // away from the line.

integral = 0;

else

integral = 0; // towards to the line.

}

// For position value is 0.

else\*/

integral = 0;

}

// For D

derivative = error - lastError;

/\*

Serial.print("Error\_P : "); // Show value of error P on laptop.

Serial.println(error);

Serial.print("Error\_I : "); // Show value of error I on laptop.

Serial.println(integral);

Serial.print("Error\_D : "); // Show value of error D on laptop.

Serial.println(derivative);

\*/

// Sum of the error of PID + encoder

turn = (Kp\*error + Ki\*integral/10 + Kd\*derivative);

turn = constrain(turn,-100,100);

// Make turn1 & turn2 only have -ve value.

// +turn1 & -turn2

turn\_L = constrain(turn,-100,0);

turn\_R = constrain(turn,0,100);

/\*

Serial.print("Turn : "); // Show value of turn on laptop.

Serial.println(turn);

Serial.println(" ");

\*/

}

// Robot go straight.

void go\_Straight(int speedL, int speedR)

{

Forward();

speedL = speed\_Map(speedL);

speedR = speed\_Map(speedR);

analogWrite(pwm1,speedL);

analogWrite(pwm2,speedR);

/\*

Serial.println("\*Go Straight\*");

Serial.print("pwm1 : ");

Serial.println(speedL);

Serial.print("pwm2 : ");

Serial.println(speedR);\*/

}

// Robot reverse.

void go\_Reverse(int speedL, int speedR)

{

Backward();

speedL = speed\_Map(speedL);

speedR = speed\_Map(speedR);

analogWrite(pwm1,speedL);

analogWrite(pwm2,speedR);

/\*

Serial.println("\*Go Backward\*");

Serial.print("pwm1 : ");

Serial.println(speedL);

Serial.print("pwm2 : ");

Serial.println(speedR);\*/

}

// Robot rotate left.

void self\_Rotate\_Left(int speedL, int speedR)

{

rotateLeft();

speedL = speed\_Map(speedL);

speedR = speed\_Map(speedR);

analogWrite(pwm1,speedL);

analogWrite(pwm2,speedR);

/\*

Serial.println("\*Rotate Left\*");

Serial.print("pwm1 : ");

Serial.println(speedL);

Serial.print("pwm2 : ");

Serial.println(speedR);\*/

}

// Robot rotate right.

void self\_Rotate\_Right(int speedL, int speedR)

{

rotateRight();

speedL = speed\_Map(speedL);

speedR = speed\_Map(speedR);

analogWrite(pwm1,speedL);

analogWrite(pwm2,speedR);

/\*

Serial.println("\*Rotate Right\*");

Serial.print("pwm1 : ");

Serial.println(speedL);

Serial.print("pwm2 : ");

Serial.println(speedR);\*/

}

// Convert speed in the range of 0 - 100.

int speed\_Map(int speed\_Motor)

{

speed\_Motor = constrain(speed\_Motor,0,100);

speed\_Motor = map(speed\_Motor,0,100,0,255);

return speed\_Motor;

}

// Setup the motors to move forward.

void Forward()

{

digitalWrite(motor\_L1,HIGH); // Move forward.

digitalWrite(motor\_R1,LOW);

digitalWrite(motor\_L2,LOW); // Move forward.

digitalWrite(motor\_R2,HIGH);

}

// Setup the motors to move backward.

void Backward()

{

digitalWrite(motor\_L1,LOW); // Move backward.

digitalWrite(motor\_R1,HIGH);

digitalWrite(motor\_L2,HIGH); // Move backward.

digitalWrite(motor\_R2,LOW);

}

// Setup motors to turn left.

void rotateLeft()

{

digitalWrite(motor\_L1,LOW); // Move backward.

digitalWrite(motor\_R1,HIGH);

digitalWrite(motor\_L2,LOW); // Move forward.

digitalWrite(motor\_R2,HIGH);

}

// Setup motors to turn right.

void rotateRight()

{

digitalWrite(motor\_L1,HIGH); // Move forward.

digitalWrite(motor\_R1,LOW);

digitalWrite(motor\_L2,HIGH); // Move backward.

digitalWrite(motor\_R2,LOW);

}

// Stop the motors.

void StopRun()

{

digitalWrite(pwm1,0);

digitalWrite(pwm2,0);

delay(100);

digitalWrite(motor\_L1,LOW);

digitalWrite(motor\_R1,LOW);

digitalWrite(motor\_L2,LOW);

digitalWrite(motor\_R2,LOW);

}