#include <QTRSensors.h>

//Only change introduced is removal of timing in detecting right or left turn

//1---->White

//0---->Black

QTRSensors qtr;

int bp=A5; //bp

int bn=A4; //bn

const uint8\_t SensorCount = 8;

uint16\_t sensorValues[SensorCount];

uint16\_t sensor[SensorCount];

int b=0;

int w=1;

int a[]={w,0,0,0,0,0,0,0,0,w};

//Array for Arduino pins

//const int sensPins[]={0,A5,A4,2,3,4,A0,6,7,0};

const int olba=A0;

const int olfo=A1;

const int orba=A2;

const int orfo=A3;

const int speed1=11; //right motor

const int speed2=10; //left motor

int trial=0;

int leftloc=0;

int rightloc=0;

int leftloc1=0;

int rightloc1=0;

int leftloc2=0;

int rightloc2=0;

int countleft=0;

int countright=0;

int error=0;

float speedleft=255;

float speedright=255;

int rf=0;

int lf=0;

int rb=0;

int lb=0;

int previous\_error=0;

int len=0;

int previous\_len=0;

int proportional[]={0,2,4,6,8,10,12,14,15};

int diff=0;

float increment=0;

float kp=.3;

float kd=.3;

float err\_brake=0.05;

float diff\_brake=0.0;

float eff=0.0;

int right=0;

int left=0;

int exleft=0;

int exright=0;

int multi=0;

String s="";

int leftcheck=0;

int rightcheck=0;

int leftavg[]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};

int rightavg[]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};

int avgturn=0;

long int calright=0;

long int calleft=0;

long int extremeright=0;

long int extremeleft=0;

//long int extremevalid=500;

long int turnvalid=150;

long int setright=0;

long int setleft=0;

long int sharpstart=0;

long int sharpcomplete=0;

long int sharpvalid=400;

long int sharpstartvalid=500;

long int finish=0;

int orb()

{ digitalWrite(orba,HIGH);

digitalWrite(orfo,LOW);

return (speed1);

}

int orf()

{ digitalWrite(orfo,HIGH);

digitalWrite(orba,LOW);

return (speed1);

}

int olb()

{ digitalWrite(olba,HIGH);

digitalWrite(olfo,LOW);

return(speed2);

}

int olf()

{ digitalWrite(olfo,HIGH);

digitalWrite(olba,LOW);

return(speed2);

}

void setup() //setup

{

//For Debugging

// configure the sensors

qtr.setTypeRC();

qtr.setSensorPins((const uint8\_t[]){3, 4, 5, 6, 7, 8, 9, 12}, SensorCount);

qtr.setEmitterPin(2);

pinMode(LED\_BUILTIN, OUTPUT);

digitalWrite(LED\_BUILTIN, HIGH); // turn on Arduino's LED to indicate we are in calibration mode

// 2.5 ms RC read timeout (default) \* 10 reads per calibrate() call

// = ~25 ms per calibrate() call.

// Call calibrate() 400 times to make calibration take about 10 seconds.

for (uint16\_t i = 0; i < 400; i++)

{

qtr.calibrate();

}

digitalWrite(LED\_BUILTIN, LOW); // turn off Arduino's LED to indicate we are through with calibration

// print the calibration minimum values measured when emitters were on

Serial.begin(9600);

int i=1;

/\* while(i<9)

{

pinMode(sensPins[i],INPUT);

i=i+1;

}

\*/

pinMode(olf, OUTPUT);

pinMode(olb, OUTPUT);

pinMode(orf, OUTPUT);

pinMode(orb, OUTPUT);

pinMode(speed1,OUTPUT);

pinMode(speed2,OUTPUT);

pinMode(bn,OUTPUT);

pinMode(bp,OUTPUT);

Serial.println("Setup...");

Serial.println("Ready for transmission..");

}

void readpins()

{

int i=0;

qtr.readLineBlack(sensorValues);

while(i<8)

{

if(sensorValues[i]>=500)

a[i+1]=0;

else

a[i+1]=1;

i++;

}

}

void disp()

{

Serial.print(a[0]);

Serial.print(" ");

Serial.print(a[1]);

Serial.print(" ");

Serial.print(a[2]);

Serial.print(" ");

Serial.print(a[3]);

Serial.print(" ");

Serial.print(a[4]);

Serial.print(" ");

Serial.print(a[5]);

Serial.print(" ");

Serial.print(a[6]);

Serial.print(" ");

Serial.print(a[7]);

Serial.print(" ");

Serial.print(a[8]);

Serial.print(" ");

Serial.print(a[9]);

Serial.println(" ");

}

void reset()

{

digitalWrite(olf(),0);

digitalWrite(olb(),0);

digitalWrite(orf(),0);

digitalWrite(orb(),0);

while(0<1)

{

delay(1000);

}

}

void leftlocation()

{

leftloc1=-1;

leftloc2=-1;

countleft=0;

for(int i=0;i<9;i++)

{

if(a[i]!=a[i+1])

{

if(leftloc1==-1)

leftloc1=i+1;

else if(leftloc2==-1)

leftloc2=i+1;

countleft++;

}

}

}

void rightlocation()

{

rightloc1=-1;

rightloc2=-1;

countright=0;

for(int i=9;i>0;i--)//error here in all the previous versions, corrected here

{

if(a[i]!=a[i-1])

{

if(rightloc1==-1)

rightloc1=i;

else if(rightloc2==-1)

rightloc2=i;

countright++;

}

}

}

void centrelocation()

{

int sum=rightloc+leftloc;

if(sum<0 && previous\_error<0)

error=-8;

else if(sum<0 && previous\_error>0)

error=8;

else

error=sum-10;

len=rightloc-leftloc;

}

void speedcontrol()

{

int initial=speedleft-speedright;

increment=0;

increment=increment+kp\*63.75\*error;//considering 8 as max error 510/8 gives 63.75

diff=(error-previous\_error);

float diff\_const=(kd)\*510;//here constant inside ranges from 0 to 1 to define the resistance to any change

//considering 510 as maximum deflection for unit diff

increment=increment+(diff\*diff\_const);

if(increment>510)

increment=510;

else if(increment<-510)

increment=-510;

increment=increment-initial;

if(increment>0)

{

float temp=255-speedleft;

if(increment<=temp)

speedleft=speedleft+increment;

else

{

speedleft=speedleft+temp;

temp=increment-temp;

speedright=speedright-temp;

}

}

else if(increment<0)

{

float temp=255-speedright;

if((-increment)<=temp)

speedright=speedright-increment;

else

{

speedright=speedright+temp;

temp=increment+temp;

speedleft=speedleft+temp;

}

}

}

void movement()

{

float factor=1;

//record(speedright,speedleft);

factor=error;

if(factor<0)

factor=-factor;

factor=1-(factor\*err\_brake);

if(diff<0)

diff=-diff;

factor=1-(diff\*diff\_brake);

factor=factor\*0.9;

if(speedright<0)

{

rb=-(speedright\*factor);

rf=0;

}

else

{

rf=speedright\*factor;

rb=0;

}

if(speedleft<0)

{

lb=-(speedleft\*factor);

lf=0;

}

else

{

lf=speedleft\*factor;

lb=0;

}

if(lb==0)

analogWrite(olf(),lf);

else if(lf==0)

analogWrite(olb(),lb);

if(rb==0)

analogWrite(orf(),rf);

else if(rf==0)

analogWrite(orb(),rb);

previous\_error=error;

//delay(t);

}

void pause()

{

digitalWrite(olf(),LOW);

//digitalWrite(olb(),LOW);

digitalWrite(orf(),LOW);

//digitalWrite(orb(),LOW);

}

void sharpright()

{

analogWrite(olf(),100);

//analogWrite(olb(),0);

//analogWrite(orf(),0);

analogWrite(orb(),100);

while(0<1)

{

readpins();

if(a[8]==b)

break;

}

analogWrite(olf(),70);

//analogWrite(olb(),0);

//analogWrite(orf(),0);

analogWrite(orb(),70);

while(0<1)

{

readpins();

if(a[5]==b)

break;

}

// avguse(0);

readpins();

leftlocation();

rightlocation();

}

void sharpleft()

{

//analogWrite(olf(),0);

analogWrite(olb(),100);

analogWrite(orf(),100);

//analogWrite(orb(),0);

while(0<1)

{

readpins();

if(a[1]==b)

break;

}

//analogWrite(olf(),0);

analogWrite(olb(),70);

analogWrite(orf(),70);

//analogWrite(orb(),0);

while(0<1)

{

readpins();

if(a[4]==b)

break;

}

// avguse(0);

readpins();

leftlocation();

rightlocation();

}

void check()//determines whether there is a acute angle

{

delay(1);

readpins();

leftlocation();

rightlocation();

leftloc=leftloc1;

rightloc=rightloc1;

centrelocation();

if(countright==4 && countleft==4)

{

if(exright==1)

{

leftcheck=0;

rightcheck=1;

}

else if(exleft==1)

{

leftcheck=1;

rightcheck=0;

}

sharpstart=millis();

}

}

char decide(char ch)//returns stored decision when X is send as input

{// and stores decision in reduced form for other inputs with return as X

char p='X';//this decide is thorougly edited and if this does not work apply recursion

if(ch=='X')

{

p=s.charAt(0);

s=s.substring(1);

}

else

{

if(ch=='U')

{

s=s+"U";

}

else if(ch=='r')

{

int l=s.length();

if(s.charAt(l-1)=='U')

{

if(s.charAt(l-2)=='S')

s=s.substring(0,(l-2))+"l";

else if(s.charAt(l-2)=='r')

s=s.substring(0,(l-2))+"S";

else if(s.charAt(l-2)=='l')

s=s.substring(0,(l-2))+"U";

}

else

s=s+"r";

}

else if(ch=='l')

{

int l=s.length();

if(s.charAt(l-1)=='U')

{

if(s.charAt(l-2)=='S')

s=s.substring(0,(l-2))+"r";

else if(s.charAt(l-2)=='r')

s=s.substring(0,(l-2))+"U";

else if(s.charAt(l-2)=='l')

s=s.substring(0,(l-2))+"S";

}

else

s=s+"l";

}

else if(ch=='R')

{

int l=s.length();

if(s.charAt(l-1)=='U')

{

if(s.charAt(l-2)=='R')

s=s.substring(0,(l-2))+"S";

else if(s.charAt(l-2)=='S')

s=s.substring(0,(l-2))+"L";

else if(s.charAt(l-2)=='L')

s=s.substring(0,(l-2))+"U";

}

else

s=s+"R";

}

else if(ch=='L')

{

int l=s.length();

if(s.charAt(l-1)=='U')

{

if(s.charAt(l-2)=='R')

s=s.substring(0,(l-2))+"U";

else if(s.charAt(l-2)=='S')

s=s.substring(0,(l-2))+"R";

else if(s.charAt(l-2)=='L')

s=s.substring(0,(l-2))+"S";

}

else

s=s+"L";

}

else if(ch=='S')

{

int l=s.length();

if(s.charAt(l-1)=='U')

{

if(s.charAt(l-2)=='R')

s=s.substring(0,(l-2))+"L";

else if(s.charAt(l-2)=='S')

s=s.substring(0,(l-2))+"U";

else if(s.charAt(l-2)=='L')

s=s.substring(0,(l-2))+"R";

else if(s.charAt(l-2)=='r')

s=s.substring(0,(l-2))+"S";

else if(s.charAt(l-2)=='l')

s=s.substring(0,(l-2))+"S";

}

else

s=s+"S";

}

}

return p;

}

void shoot(long int i)//2 means found && 1 means not found

{

analogWrite(olf() ,100);

//analogWrite(olb(),0);

analogWrite(orf(),100);

//analogWrite(orb(),0);

delay(i);

pause();

}

void execute(int n)//n==0 denotes left or right turn, n==1 denotes sharp right or sharp leftturn

{

char ch=decide('X');

if(ch=='R' || ch=='r')

{

if(n==0)

{

sharpright();

sharpcomplete=millis();

rightcheck=2;

right=0;

left=0;

}

else

{

sharpright();

sharpcomplete=millis();

rightcheck=2;

}

}

else if(ch=='L' || ch=='l')

{

if(n==0)

{

sharpleft();

sharpcomplete=millis();

leftcheck=2;

right=0;

left=0;

}

else

{

sharpleft();

sharpcomplete=millis();

leftcheck=2;

}

}

else if(ch=='S')

{

rightcheck=0;

leftcheck=0;

right=0;

left=0;

//analogWrite(bp,0);

}

}

void loop() //loop

{

if(trial==0)

{

//reset();

analogWrite(bp,255);

analogWrite(bn,0);

delay(1000);

analogWrite(bp,0);

trial=1;

}

Serial.println("LOOP Started+++++++++++++++");

readpins();

leftlocation();

rightlocation();

if(a[1]==w || a[2]==w || a[3]==w || a[4]==w || a[5]==w || a[6]==w || a[7]==w || a[8]==w)

{

finish=millis();

}

if(a[1]==b && a[2]==b && a[3]==b && a[4]==b && a[5]==b && a[6]==b && a[7]==b && a[8]==b)

{

multi=1;

if((millis()-finish)>70)

{

if(trial==1)

{

pause();

while(0<1)

{

readpins();

if(a[1]==w || a[2]==w || a[3]==w || a[4]==w || a[5]==w || a[6]==w || a[7]==w || a[8]==w)

break;

}

delay(1000);

trial=2;

readpins();

leftlocation();

rightlocation();

multi=0;

right=0;

left=0;

analogWrite(bp,255);

}

else if(trial==2)

reset();

}

}

if(a[1]==b)

{

exleft=1;

exright=0;

//extremeleft=millis();

}

if(a[8]==b)

{

exleft=0;

exright=1;

//extremeright=millis();

}

disp();

if(countright==4 && countleft==4 && leftcheck==0 && rightcheck==0)

{

check();

}

if(rightcheck==0 && leftcheck==0)

{

if(a[5]==b && a[6]==b && a[7]==b && a[8]==b)

{

right=1;

}

if(a[1]==b && a[2]==b && a[3]==b && a[4]==b)

{

left=1;

}

if(a[7]==w && a[8]==w && right==1)// && (millis()-setright)<turnvalid)

{

readpins();

pause();

delay(200);

if(a[1]==b || a[2]==b || a[3]==b || a[4]==b || a[5]==b || a[6]==b || a[7]==b || a[8]==b)

multi=1;

if(trial==1 || multi==0)

{

sharpright();

if(multi==1)

char ch=decide('R');

multi=0;

//delay(100);

// sharpcomplete=millis();

//rightcheck=2;

right=0;

left=0;

}

else if(trial==2)

execute(0);

}

else if( a[1]==w && a[2]==w && left==1)// && (millis()-setleft)<turnvalid)

{

readpins();

pause();

delay(200);

if(a[1]==b || a[2]==b || a[3]==b || a[4]==b || a[5]==b || a[6]==b || a[7]==b || a[8]==b)

multi=1;

if(trial==1 || multi==0)

{

if(multi==0)

{

sharpleft();

//delay(100);

//sharpcomplete=millis();

//leftcheck=2;

}

else

char ch=decide('S');

right=0;

left=0;

}

else if(trial==2)

execute(0);

}

else if(a[1]==w && a[2]==w && a[3]==w && a[4]==w && a[5]==w && a[6]==w && a[7]==w && a[8]==w)

{

//if(((millis()-extremeleft)>extremevalid || exleft==0) && ((millis()-extremeright)>extremevalid || exright==0))//an alternative to the next condition

if(exleft==0 && exright==0)

{

shoot(80);

pause();

delay(100);

sharpright();//alternative for u turn

char ch=decide('U');

readpins();

exleft=0;

exright=0;

leftlocation();

rightlocation();

}

}

rightloc=rightloc1;

leftloc=leftloc1;

}

else

{

analogWrite(bp,255);

if(rightcheck!=0)

{

leftloc=leftloc1;

rightloc=leftloc2;

if(countright==2 && countleft==2 && rightcheck==1)

{

pause();

delay(500);

if(a[1]==b || a[2]==b || a[3]==b || a[4]==b || a[5]==b || a[6]==b || a[7]==b || a[8]==b)

multi=1;

if(trial==1 || multi==0)

{

sharpright();

if(multi==1)

char ch=decide('r');

multi=0;

sharpcomplete=millis();

rightcheck=2;

}

else if(trial==2)

execute(1);

}

/\*

if(countright==2 && countleft==2 && rightcheck==1 && (millis()-sharpstart)>sharpstartvalid)

{

leftcheck=0;

rightcheck=0;

analogWrite(bp,0);

}

\*/

}

else if(leftcheck!=0)

{

leftloc=rightloc2;

rightloc=rightloc1;

if(countright==2 && countleft==2 && leftcheck==1)

{

pause();

delay(500);

if(a[1]==b || a[2]==b || a[3]==b || a[4]==b || a[5]==b || a[6]==b || a[7]==b || a[8]==b)

multi=1;

if(trial==1 || multi==0)

{

if(multi==0)

{

sharpleft();

sharpcomplete=millis();

leftcheck=2;

}

else

{

char ch=decide('S');

leftcheck=0;

rightcheck=0;

analogWrite(bp,0);

}

}

else if(trial==2)

execute(1);

}

/\*

if(countright==2 && countleft==2 && leftcheck==1 && (millis()-sharpstart)>sharpstartvalid)

{

leftcheck=0;

rightcheck=0;

analogWrite(bp,0);

}\*/

}

if(countright==2 && countleft==2 && (millis()-sharpcomplete)>sharpvalid && (leftcheck==2 || rightcheck==2))

{

leftcheck=0;

rightcheck=0;

analogWrite(bp,0);

}

}

if(a[7]==w && a[8]==w)

{

right=0;

setright=millis();

}

if(a[1]==w && a[2]==w)

{

left=0;

setleft=millis();

}

if(a[1]==w && a[8]==w && rightcheck==0 && leftcheck==0 && (a[2]==b || a[3]==b || a[4]==b || a[5]==b || a[6]==b || a[7]==b))

{

exleft=0;

exright=0;

extremeleft=millis();

extremeright=millis();

analogWrite(bp,0);

}

else

{

analogWrite(bp,255);

}

/\*

if(a[1]==w && a[2]==w && a[3]==w && a[4]==w && a[5]==w && a[6]==w && a[7]==w && a[8]==w)

{

exleft=0;

exright=0;

}

\*/

centrelocation();

speedcontrol();

movement();

}