```
#define CenterSensor A2
#define leftNearSensor A3
#define leftFarSensor A4
#define rightNearSensor A1
#define rightFarSensor A0



int mata0; // mata means “eye”
int mata1;
int mata2;
int mata3;
int mata4;


float Kp=45,Ki=0.6,Kd=40;

float error=0, P=0, I=0, D=0, PID_value=0;

float previous_error=0, previous_I=0;



int initial_motor_speed=130;



void read_sensor(void);

void calculate_pid(void);

void motor_control(void);

int flag = 0;
int data;
int pathlength; //variable to record the total of path length
int readpath; //variable to call the path record
char path[99]; //array for path record


int s[5];
int run_tracker=0;
int sum=0;
int L1=4, L2=5, enL=11, R1=2, R2=3, enR=10;
int threshold = 350;



void setup() {
  // put your setup code here, to run once:
   Serial.begin(9600);
  pinMode(L1, OUTPUT);
  pinMode(L2, OUTPUT);
  pinMode(enR, OUTPUT);
  pinMode(enL, OUTPUT);
  pinMode(R1, OUTPUT);
  pinMode(R2, OUTPUT);
  pinMode(13, OUTPUT);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
read_sensor();
Serial.println(error);
condition();
choosepath();

}



void read_sensor() // white line on black surface
{
  mata0 = analogRead(rightFarSensor);
  mata1 = analogRead(rightNearSensor);
  mata2 = analogRead(CenterSensor);
  mata3 = analogRead(leftNearSensor);
  mata4 = analogRead(leftFarSensor);

if (mata0 < threshold)
  {s[0] = 1;}
else
  {s[0] = 0;}

if (mata1 < threshold)
  {s[1] = 1;}
else
  {s[1] = 0;}

if (mata2 < threshold)
  {s[2] = 1;}
else
  {s[2] = 0;}

if (mata3 < threshold)
  {s[3] = 1;}
else
  {s[3] = 0;}

if (mata4 < threshold)
  {s[4] = 1;}
else
  {s[4] = 0;}

/*
  Serial.print(s[0]);
Serial.print("    ");
Serial.print(s[1]);
Serial.print("    ");
Serial.print(s[2]);
Serial.print("    ");
Serial.print(s[3]);
Serial.print("    ");
Serial.print(s[4]);
Serial.println("  ");
*/
```

```
if((s[0]==0)&&(s[1]==0)&&(s[2]==0)&&(s[3]==0)&&(s[4]==1))
  error=4;
  else if((s[0]==0)&&(s[1]==0)&&(s[2]==0)&&(s[3]==1)&&(s[4]==1))
  error=3;
  else if((s[0]==0)&&(s[1]==0)&&(s[2]==0)&&(s[3]==1)&&(s[4]==0))
  error=2;
  else if((s[0]==0)&&(s[1]==0)&&(s[2]==1)&&(s[3]==1)&&(s[4]==0))
  error=1;
  else if((s[0]==0)&&(s[1]==0)&&(s[2]==1)&&(s[3]==0)&&(s[4]==0))
  error=0;
  else if((s[0]==0)&&(s[1]==1)&&(s[2]==1)&&(s[3]==0)&&(s[4]==0))
  error=-1;
  else if((s[0]==0)&&(s[1]==1)&&(s[2]==0)&&(s[3]==0)&&(s[4]==0))
  error=-2;
  else if((s[0]==1)&&(s[1]==1)&&(s[2]==0)&&(s[3]==0)&&(s[4]==0))
  error=-3;
  else if((s[0]==1)&&(s[1]==0)&&(s[2]==0)&&(s[3]==0)&&(s[4]==0))
  error=-4;
  else if((s[0]==0)&&(s[1]==0)&&(s[2]==0)&&(s[3]==0)&&(s[4]==0))
    if(error==-4) error=-5;
    else error=5;

    //change the sensor readings into a series of binary number
data=(s[0]*16)+(s[1]*8)+(s[2]*4)+(s[3]*2)+(s[4]*1);
}
void calculate_pid()
{
    P = error;
    I = I + previous_I;
    D = error-previous_error;
```

```
    PID_value = (Kp*P) + (Ki*I) + (Kd*D);


    previous_I=I;
    previous_error=error;
}
void motor_control()
{
    // Calculating the effective motor speed:
    int left_motor_speed = initial_motor_speed-PID_value;
    int right_motor_speed = initial_motor_speed+PID_value;


    // The motor speed should not exceed the max PWM value
    constrain(left_motor_speed,0,200);
    constrain(right_motor_speed,0,200);


  analogWrite(enL,initial_motor_speed-PID_value);   //Left Motor Speed
    analogWrite(enR,initial_motor_speed+PID_value);  //Right Motor Speed
}


//Intersection condition ---------------- 0b00abcde
void condition()
{
      read_sensor();
  Serial.print(error);
  if (data==0b0011100 || data == 0b0011110) {              // Make left turn untill it detects straight
path
    //Serial.print("\t");
    //Serial.println("Left");
    do {
      read_sensor();

      turnleft();
    } while (error != 0 || data!=0b0000100);
    path[pathlength]='L';
    pathlength++;


  } else if (data==0b0000111 || data==0b0001111) {          // Make right turn in case of it detects
only right path (it will go into forward direction in case of staright and right "|--")
                                // untill it detects straight path.
    //Serial.print("\t");
    //Serial.println("Right");
```

```
    moveforward();
    delay(150);
    stop();
    read_sensor();
    if (error == 5) // only right path available
    {
      do {

        turnright();
        read_sensor();
      } while (error != 0);
      path[pathlength]='R';
      pathlength++;
    }
    else if( error == 0){

      do{
        moveforward();
        read_sensor();
      }while(error==0);
      path[pathlength]='S';
      pathlength++;
    }
    }
   else if (error == -5) {        // Make left turn untill it detects straight path
    //Serial.print("\t");
    //Serial.println("Sharp Left Turn");
    do {

      turnleft();
      read_sensor();
      if (error == 0) {
        stop();
        delay(200);
      }
    } while (error != 0);
  } else if (data==0b0011111) {        // Make left turn untill it detects straight path or stop if dead
end reached.
    if (flag == 0) {

      moveforward();
      delay(150);
      stop();
      read_sensor();
      if (data==0b0011111) {     /**** End Reached, Stop! ****/
        stop();
        path[pathlength]='F';
        pathlength++;//save F

        flag = 1;
        shortpath();

      } else {        /**** Move Left ****/

        do {
          //Serial.print("\t");
          //Serial.println("Left Here");
          read_sensor();

          turnleft();
```

```
        } while (error != 0);
        path[pathlength]='L';
        pathlength++;
      }
    }
  }else if(data == 0b0000000){
    turnaround();
    path[pathlength]='U';
    pathlength++;
  }
  else {
    calculate_pid();
    motor_control();
  }
 condition();
}

void MOTOR(int left_speed, int right_speed){
  if(left_speed >= 0){
    digitalWrite(L1, HIGH);
    digitalWrite(L2, LOW);
    analogWrite(enL, left_speed);
  }
  else{
    digitalWrite(L1, LOW);
    digitalWrite(L2, HIGH);
    analogWrite(enL,abs(left_speed));
  }
  if(right_speed >= 0){
    digitalWrite(R1, LOW);
    digitalWrite(R2, HIGH);
    analogWrite(enR, right_speed);
  }
  else{
    digitalWrite(R1, HIGH);
    digitalWrite(R2, LOW);
    analogWrite(enR, abs(right_speed));
  }
}

void shortpath() //calculate the shortest path
{
  //because (..F) is the last and there should be no U recorderd before F
  int x = (pathlength-2);

  while (x > 0)
  {
    if (path[x]=='U')
      {
        if (path[x-1]=='L' && path[x+1]=='L')
          {path[x-1]='S';path[x]='O';path[x+1]='O';}
        else if (path[x-1]=='L' && path[x+1]=='S')
          {path[x-1]='R';path[x]='O';path[x+1]='O';}
        else if (path[x-1]=='R' && path[x+1]=='R')
          {path[x-1]='S';path[x]='O';path[x+1]='O';}
        else if (path[x-1]=='R' && path[x+1]=='S')
          {path[x-1]='L';path[x]='O';path[x+1]='O';}
        else if (path[x-1]=='S' && path[x+1]=='L')
          {path[x-1]='R';path[x]='O';path[x+1]='O';}
        else if (path[x-1]=='S' && path[x+1]=='R')
```

```
        {path[x-1]='L';path[x]='O';path[x+1]='O';}
       else if (path[x-1]=='L' && path[x+1]=='R')
        {
          path[x-1]='U';path[x]='O';path[x+1]='O';

        }
       else if (path[x-1]=='R' && path[x+1]=='L')
        {
          path[x-1]='U';path[x]='O';path[x+1]='O';

        }
       else if (path[x-1]=='S' && path[x+1]=='S')
        {
          path[x-1]='U';path[x]='O';path[x+1]='O';

        }
       //--------------------------------------
       x--;
     }
    else
     {x--;}
  }

}

void choosepath()//to get rid of the effect of “path[]==0” in the record
{
  if (path[readpath]=='F')
    {
      stop();
      finish();
    }
  else if (path[readpath]=='R')
    {
      turnright();
  delay(500); //to be free from the line if there is a straight intersection (exit 2)
  read_sensor();
  while (data!=0b0000100)
  {
    turnright();
    read_sensor();
  }
    }
  else if (path[readpath]=='S')
    {

      moveforward();
      delay(200);
    }
  else if (path[readpath]=='L')
    {
      turnleft();
  delay(500); //to be free from the line if there is a straight intersection (exit 2)
  read_sensor();
  while (data!=0b0000100)
  {
    turnleft();
    read_sensor();
  }
```

```
    }
  else if (path[readpath]=='O')
    {
      readpath++;
      choosepath();
    }
  readpath++;
  condition();
}


// functions to make bot turn, turn around and go straight
void moveforward()
{
  analogWrite(enL, 130);
  digitalWrite(L1, HIGH);
  digitalWrite(L2, LOW);
  analogWrite(enR, 130);
  digitalWrite(R1, LOW);
  digitalWrite(R2, HIGH);
}

void turnright()
{
  analogWrite(enL, 130);
  digitalWrite(L1, HIGH);
  digitalWrite(L2, LOW);
  analogWrite(enR, 130);
  digitalWrite(R1, HIGH);
  digitalWrite(R2, LOW);
}

void turnleft()
{
  analogWrite(enL, 130);
   digitalWrite(L1, LOW);
  digitalWrite(L2, HIGH);
  analogWrite(enR, 130);
       digitalWrite(R1, LOW);
  digitalWrite(R2, HIGH);
}


void turnaround()
{
  turnleft();
  delay(500); //to be free from the line if there is a straight intersection (exit 2)
  read_sensor();
  while (error != 0)
  {
    turnleft();
    read_sensor();
  }
}

void stop()
{
  analogWrite(enL, 0);
  digitalWrite(L1, LOW);
  digitalWrite(L2, LOW);
```

```
  analogWrite(enR, 0);
  digitalWrite(R1, LOW);
  digitalWrite(R2, LOW);
}

void finish()
{
  digitalWrite(13, HIGH);
  delay(100);
  digitalWrite(13, LOW);
  delay(100);
  finish();
}
```