



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

ROBOTICS AND ELECTRONICS

Build a Line Follower Robot

A complete build guide with two IR sensors, an L298N driver and Arduino

INSIDE THIS BOOK

- How a line follower works
- Parts, tools and safety
- Building the chassis
- Mounting and calibrating the IR sensors
- Wiring the robot
- ...and 6 more chapters, practice questions and a glossary

Intermediate

Students of class 7 to 12 and robotics clubs

engineergurukul.com

+91 97116 99511

Build a Line Follower Robot

Build a robot that follows a black line on its own, from an empty chassis to a tuned, working machine. You will mount and calibrate two IR sensors, wire an L298N driver, understand every line of the Arduino code, test it on a tape track, and explain it confidently at an exhibition.

WHO IT IS FOR	Students of class 7 to 12 and robotics clubs
LEVEL	Intermediate
SUBJECT AREA	Robotics and electronics
WHAT IS INSIDE	11 chapters, practice questions with answers, and a glossary

HOW TO USE IT

Read one chapter at a time and try the examples as you go: on a computer, at your workbench or in your notebook. The boxes give tips, safety points and short activities. When you finish, test yourself with the practice questions and check your answers at the end. Words you may not know are explained in the glossary.



Gurukul Technicals Institute

A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone 011 2721 1512 · Mobile and WhatsApp +91 97116 99511

engineergurukul@yahoo.co.in · engineergurukul.com

Free ebook. You may read, print and share this ebook for learning. It may not be sold or changed.

© 2026 Gurukul Technicals Institute. First edition, October 2026. The information was checked when it was written; software, prices and rules change, so for anything official always check the latest source.

What you will find inside

1	How a line follower works	4
2	Parts, tools and safety	5
3	Building the chassis	7
4	Mounting and calibrating the IR sensors	8
5	Wiring the robot	11
6	The logic table	13
7	The complete Arduino code	14
8	Testing on a track and tuning speed	17
9	Troubleshooting	19
10	Improvements: more sensors and the PID idea	21
11	Presenting at an exhibition	24
	Practice questions	25
	Answers	27
	Glossary	28

How a line follower works

A line follower is a small robot that drives along a dark line on a light floor without anyone steering it. It is often the first real robot that students build, because it shows the three things every robot does: it **senses**, it **decides** and it **acts**.

Seeing the line with infrared light

The robot's eyes are two **IR sensors**. IR means infrared: light just beyond red, which our eyes cannot see. Each sensor has two parts side by side, pointing down at the floor:

- an IR LED, which shines invisible infrared light downward
- a light detector (a photodiode or phototransistor), which measures how much light bounces back

A white surface reflects most of the light back to the detector. A black surface absorbs most of it, so very little comes back. The sensor module compares the amount of reflected light with a level you set, and gives a simple digital answer: HIGH or LOW. In this way each sensor tells the Arduino one thing only: "I am over black" or "I am over white".

The sense, decide, act loop

1. **Sense.** The Arduino reads both sensors.
2. **Decide.** It works out where the line is: between the sensors, under the left one or under the right one.
3. **Act.** It sets the two motors to go straight, turn left or turn right.
4. **Repeat.** The Arduino runs this loop thousands of times a second, so the robot corrects itself constantly.

Steering with two wheels

The robot has no steering wheel. It has two motors, one for each side wheel. To turn left, it slows or stops the left wheel while the right wheel keeps driving. To turn right, it does the opposite. This is called **differential drive**. A free-rolling caster wheel holds up the other end.

REMEMBER

Black absorbs infrared light and white reflects it. Every decision your robot makes starts from this one fact.

DID YOU KNOW

Many factories and warehouses use **automated guided vehicles** (AGVs) that follow lines, magnetic tape or markers on the floor to carry goods. Your line follower uses the same basic idea on a smaller scale.

TRY THIS

In a dim room, point a TV remote at a phone camera and press a button. Many phone cameras show the invisible IR flash as a faint light. (Some phones filter it out, so try the front and back cameras.)

CHAPTER 2

Parts, tools and safety

Collect everything before you start, so you do not stop halfway through with a missing screw.

Parts list

Part	Quantity	What it does
Arduino Uno with USB cable	1	The brain: reads sensors and controls motors
L298N motor driver module	1	Gives the motors the current the Arduino cannot
IR sensor modules (3 pins: VCC, GND, OUT)	2	The eyes: detect black and white
BO motors (100 to 150 RPM suits beginners)	2	Drive the wheels
Wheels to fit BO motors (about 65 mm)	2	Grip the floor

Part	Quantity	What it does
Caster wheel (ball type)	1	Supports the front without steering
2WD chassis plate with motor clamps and screws	1	The body of the robot
Battery holder for 2 × 18650 Li-ion cells, with cells, or a 6 × AA holder	1	Power for motors and Arduino
On/off switch	1	Cuts the battery without unplugging wires
Mini breadboard	1	Shares 5V and GND between the sensors
Jumper wires (male-to-female and male-to-male)	about 20	Connections
Spacers, double-sided foam tape, cable ties	a few	Mounting and tidying
Black insulation tape and white chart paper or flex sheet	enough for a track	The test track

Tools

- A small screwdriver set (cross-head and flat)
- A wire stripper and a small pair of pliers
- A multimeter
- A ruler and scissors
- A computer with the Arduino IDE installed (free from arduino.cc)

Two IR sensor types

Most kits contain one of two common modules. Both work with this book.

Type	How to recognise it
Separate LED and detector	A clear LED and a dark LED side by side, a small chip and a blue adjusting knob

Type	How to recognise it
TCRT5000 type	The LED and detector sit together in one small black block, with a blue adjusting knob

SAFETY

Li-ion cells must be charged only with a proper Li-ion charger, under an adult's eye, and never shorted, crushed or left charging unattended. If you are younger, use the 6 × AA holder instead. Soldering, hot glue and cutting need an adult. All our power comes from batteries under 12 V; never connect a project to mains.

TIP

Lay out every part on a tray or an old steel thali before you begin. Small screws roll away easily, and a tray keeps them together.

CHAPTER 3

Building the chassis

Most kits use a flat acrylic plate with holes cut for the motors, caster and battery. The steps below suit a typical 2WD kit; yours may differ slightly, so look at all the pieces first and work out where each one goes.

Assembly steps

- 1. Peel the protective film.** Acrylic plates usually come covered in protective paper or plastic film on both sides. Peel it off.
- 2. Fit the motors.** Each BO motor is held by two acrylic clamps. Slot one clamp through the plate, place the motor against it, add the outer clamp, and push two long screws through the clamps and the motor's holes. Tighten the nuts until the motor cannot wiggle. Point both motor shafts outward.
- 3. Check the motor wires.** Many BO motors come with wires already soldered on. If yours do not, an adult should solder two wires to each motor's terminals.
- 4. Fit the caster.** Screw the ball caster to the other end of the plate, with spacers if needed, so that the plate sits level.

- 5. Fit the battery holder.** Fix it near the middle, low down, with screws or strong double-sided tape. Weight in the middle keeps the robot stable.
- 6. Mount the electronics.** Fix the Arduino on spacers so its underside cannot touch metal screws. Place the L298N near the motors so their wires are short.
- 7. Fit the switch.** Put it where you can reach it easily, in the battery's positive wire.
- 8. Push on the wheels.** Line up the flat sides of the shaft with the wheel's hole and press firmly until the wheel is fully on.

The end with the caster becomes the **front** of the robot, and the sensors go there.

Why this layout works

The sensors sit ahead of the wheels, so the robot sees a bend before its wheels reach it. This gives it a little time to react. If the sensors sat right between the wheels, it would only notice the bend when it was already leaving the line.

TIP

Place the robot on a flat table and push it gently forward. It should roll straight. If it pulls to one side, check that both motors are clamped squarely and both wheels are fully on.

SAFETY

Tighten small screws with the correct screwdriver size, and push wheels on with your palm, not with a tool that can slip and cut your hand.

EXAMPLE

Riya from Lucknow fixed her battery at the very back, behind the wheels. The weight tipped the robot backwards, the caster lifted, and the sensors pointed at nothing. Moving the battery to the middle solved the problem.

CHAPTER 4

Mounting and calibrating the IR sensors

The sensors decide everything your robot does, so mount and adjust them with care.

Where to mount them

- **Facing down**, at the front edge, with the LEDs pointing straight at the floor.
- **About 5 to 10 mm above the floor**. Too high and they cannot tell black from white; too low and they scrape. Find the best height by testing.
- **Spaced a little wider than the line**. When the robot is centred, both sensors should see white on either side of the tape. Most insulation tape is a little under 2 cm wide; for a line that wide, start with the sensor centres about 3 cm apart. Measure your own tape.

Use a small L-shaped bracket, a strip of acrylic, or even an ice-cream stick fixed with screws or cable ties to hold the sensors. Make sure they cannot swing about.

The sensor's pins and lights

Pin	Connect to
VCC	5 V
GND	GND
OUT	An Arduino input pin

Most modules have two small LEDs: a power LED that stays on, and a signal LED that changes when the sensor sees black or white. On most modules the output is LOW over white and HIGH over black, and the signal LED glows over white. Some modules work the other way round, which is why you test before you trust.

A test sketch

Connect the left sensor's OUT to A0 and the right sensor's OUT to A1 (the next chapter shows the full wiring). The Uno's analog pins work as ordinary digital inputs too. Upload this sketch and open the Serial Monitor at 9600 baud.

```
// Show what each IR sensor reads: 1 means HIGH, 0 means LOW
const int LEFT_SENSOR = A0;
const int RIGHT_SENSOR = A1;

void setup() {
  pinMode(LEFT_SENSOR, INPUT);
  pinMode(RIGHT_SENSOR, INPUT);
  Serial.begin(9600);
}

void loop() {
  Serial.print("Left: ");
  Serial.print(digitalRead(LEFT_SENSOR));
  Serial.print("    Right: ");
  Serial.println(digitalRead(RIGHT_SENSOR));
  delay(200);
}
```

Calibrating the potentiometer

The blue knob is a **potentiometer**: a small adjustable resistor that sets the sensitivity. It decides how much reflected light counts as "white".

1. Put the robot on the white part of your track, at its real height, and watch the left reading.
2. With a small screwdriver, turn the left sensor's knob slowly until you find the point where the reading flips between 0 and 1. Many modules use a multi-turn knob, so this may take several full turns. If nothing changes after many turns, go the other way.
3. Turn the knob slowly to and fro across the flip point, and leave it a little way past it on the side where the reading shows the white value steadily.
4. Slide the robot so the left sensor is over the black tape. The reading should change. If it does not, the knob is too far from the flip point: move it a little closer and test again.
5. Repeat for the right sensor, then test each one several times over white and black. A flicker right at the edge of the tape is fine; a flicker on plain white means adjust again.

REMEMBER

Calibrate where you will run. Sunlight and some bright lamps contain infrared light that can fool the sensors, so a robot tuned in the classroom may need adjusting on the exhibition stage.

TIP

Note which value means black for your modules. If black reads 0 instead of 1, you will change just one line in the main program.

CHAPTER 5

Wiring the robot

Now connect everything. Work with the battery switched off and the USB cable unplugged, and tick off each row of the table as you go.

The 5 V problem

The Uno has only one 5V pin on its header, but two sensors need 5 V. Run one wire from the Uno's 5V pin and one from a GND pin to the two rails of a mini breadboard. Then plug each sensor's VCC and GND into those rails.

Wiring table

From	To
Left sensor VCC and right sensor VCC	Breadboard 5 V rail (from Uno 5V)
Left sensor GND and right sensor GND	Breadboard GND rail (from Uno GND)
Left sensor OUT	Uno A0
Right sensor OUT	Uno A1
L298N ENA (jumper cap removed)	Uno pin 5
L298N IN1	Uno pin 7
L298N IN2	Uno pin 8
L298N IN3	Uno pin 4

From	To
L298N IN4	Uno pin 3
L298N ENB (jumper cap removed)	Uno pin 6
Left motor wires	L298N OUT1 and OUT2
Right motor wires	L298N OUT3 and OUT4
Battery positive	Switch, then L298N 12V terminal and Uno VIN
Battery negative	L298N GND terminal
Uno GND	L298N GND terminal

Leave the L298N's 5V terminal empty, and keep its 5 V regulator jumper (near the power terminals) in place.

Finding the real enable pin

Each ENA and ENB jumper cap sits on two pins: one is 5 V and the other is the enable input. Read the label printed on your board. If it is unclear, power the board and measure each pin to GND with a multimeter. The pin showing 5 V is the supply; connect your Arduino wire to the other one.

Why the Uno goes on VIN

The battery feeds the Uno's VIN pin, and the Uno's own regulator turns this into a steady 5 V. VIN works best with 7 to 12 V, so a 2 × 18650 pack (7.4 V) or six AA cells suit it. Never connect the battery straight to the Uno's 5V pin.

REMEMBER

The wire from Uno GND to the L298N GND gives both boards a **common ground**. Without it, the driver cannot understand the Arduino's signals.

TIP

Use red wires for positive, black for ground, and other colours for signals. Tie loose wires down with cable ties, so they cannot catch in the wheels.

SAFETY

Before switching on, check the battery wires twice. A battery connected backwards can damage the driver and the Arduino in a second.

CHAPTER 6

The logic table

Before writing code, decide exactly what the robot should do in every situation. A **logic table** lists every possible combination of sensor readings and the action for each one. Two sensors, each seeing black or white, give $2 \times 2 = 4$ situations.

The four situations

Remember the layout: the line runs between the two sensors.

Left sensor	Right sensor	What it means	Action
White	White	Both sensors are off the line, which sits between them: the robot is centred	Go forward
Black	White	The left sensor is on the line: the line has curved to the left	Turn left
White	Black	The right sensor is on the line: the line has curved to the right	Turn right
Black	Black	Both are on the line: a crossing, a thick finish line, or the robot is lifted	Stop

Read the second row again. When the left sensor touches the line, the robot must steer **towards** that side to bring the line back between its sensors. Many beginners get this backwards, and their robot runs away from the line.

How to turn

There are two ways to turn, and you can choose either in the code.

Turn type	Inner wheel	Outer wheel	Good for
Pivot turn	Stopped	Forward	Gentle and medium curves
Spin turn	Backward	Forward	Very sharp bends

In a left turn, the left wheel is the inner wheel. The program in the next chapter uses pivot turns. You will try spin turns while tuning.

The decision as a flowchart, in words

1. Read the left sensor and the right sensor.
2. If both see white, drive both wheels forward at the base speed.
3. Otherwise, if only the left sees black, stop the left wheel and drive the right wheel.
4. Otherwise, if only the right sees black, stop the right wheel and drive the left wheel.
5. Otherwise both see black, so stop both wheels.
6. Go back to step 1.

DID YOU KNOW

Some builders use a different layout, with both sensors close together over a wide line. Then "both on black" means go forward, and a sensor that slips off the line tells the robot to steer back the other way. Both layouts work if the logic table matches the layout.

TRY THIS

Before uploading any code, hold the robot over the track and say aloud what it should do as you move it. "Left on black: turn left." This simple habit catches most logic mistakes.

CHAPTER 7

The complete Arduino code

Here is the whole program. Type it carefully or copy it exactly, then read the explanation below.

```

// Line follower robot: two IR sensors, L298N driver, Arduino Uno
// Black line on a white floor. The line runs between the sensors.

// ----- Pins -----
const int LEFT_SENSOR = A0;
const int RIGHT_SENSOR = A1;
const int ENA = 5;    // left motor speed (PWM)
const int IN1 = 7;    // left motor direction
const int IN2 = 8;
const int ENB = 6;    // right motor speed (PWM)
const int IN3 = 4;    // right motor direction
const int IN4 = 3;

// ----- Settings to tune -----
const int ON_BLACK = HIGH; // change to LOW if your sensors read 0 on black
int baseSpeed = 120;       // speed on straight track (0 to 255)
int turnSpeed = 140;       // outer wheel speed in a turn

void setup() {
    pinMode(LEFT_SENSOR, INPUT);
    pinMode(RIGHT_SENSOR, INPUT);
    pinMode(ENA, OUTPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);
    drive(0, 0);
    delay(3000);           // three seconds to place the robot
}

void loop() {
    bool leftBlack = (digitalRead(LEFT_SENSOR) == ON_BLACK);
    bool rightBlack = (digitalRead(RIGHT_SENSOR) == ON_BLACK);

    if (!leftBlack && !rightBlack) {
        drive(baseSpeed, baseSpeed); // centred: go forward
    } else if (leftBlack && !rightBlack) {
        drive(0, turnSpeed);          // line on the left: turn left
    } else if (!leftBlack && rightBlack) {
        drive(turnSpeed, 0);          // line on the right: turn right
    } else {
        drive(0, 0);                  // both on black: stop
    }
}

// Positive speed = forward, negative = backward, 0 = stop
void drive(int leftSpeed, int rightSpeed) {

```

```

    setMotor(ENA, IN1, IN2, leftSpeed);
    setMotor(ENB, IN3, IN4, rightSpeed);
}

void setMotor(int enPin, int inA, int inB, int speed) {
    if (speed > 0) {
        digitalWrite(inA, HIGH);
        digitalWrite(inB, LOW);
    } else if (speed < 0) {
        digitalWrite(inA, LOW);
        digitalWrite(inB, HIGH);
        speed = -speed;
    } else {
        digitalWrite(inA, LOW);
        digitalWrite(inB, LOW);
    }
    analogWrite(enPin, constrain(speed, 0, 255));
}

```

How the code works

- **Pins.** `const int` gives each pin a name. If you rewire a pin later, you change one line, not ten.
- **Settings.** `ON_BLACK` holds the reading your sensors give over black. `baseSpeed` and `turnSpeed` are the numbers you will adjust while tuning. Speeds go from 0 (stopped) to 255 (full).
- **setup().** Runs once. It sets each pin as an input or output, stops the motors, and waits three seconds so you can put the robot down.
- **loop().** Runs again and again. It reads both sensors into two `bool` variables (true or false), then follows the logic table with `if` and `else if`.
- **drive().** Takes a speed for each wheel, so `drive(0, turnSpeed)` means "left wheel stopped, right wheel forward": a pivot turn to the left.
- **setMotor().** Sets the two direction pins for one motor, then uses `analogWrite()` (PWM) on its enable pin to set the speed. `constrain()` keeps the number between 0 and 255.

Uploading

1. Switch the battery off and connect the USB cable.
2. In the Arduino IDE, choose the Arduino Uno board and the correct port.
3. Click Upload and wait for "Done uploading".

4. Unplug the USB cable, put the robot on the track and switch the battery on.

TIP

If you test with both USB and battery connected, lift the robot so its wheels spin in the air. A robot that drives off the desk while tied to a cable can pull the laptop with it.

REMEMBER

If a wheel spins backward when it should go forward, swap that motor's two wires at the L298N. Do not change the code.

CHAPTER 8

Testing on a track and tuning speed

A robot is only as good as its testing. Build a proper track and test in small steps.

Making a track

1. Tape several sheets of white chart paper together on a flat floor, or use a white flex banner. Light, plain floor tiles also work.
2. Stick black insulation tape down in a large oval first. Press it flat so no edges lift.
3. Make curves by sticking short pieces that overlap slightly, bending the line a little at each piece. Keep the first curves gentle.
4. Later, add a sharper bend, an S-curve and a straight finish section.

EXAMPLE

Meera's oval needs about 6 metres of line, plus 2 metres for a finish line and repairs. Suppose one roll holds 8 metres and costs ! 30. One roll is just enough, so she buys two for ! 60 and keeps the second as a spare.

TIP

Very shiny tape can sometimes reflect infrared like a white surface. If one part of the track confuses your robot, try a matt black tape there.

Testing in stages

- 1. Wheels in the air.** Hold the robot up and slide white paper and black tape under the sensors. Left sensor on black: the right wheel should turn and the left stop. Right sensor on black: the opposite.
- 2. Straight line.** Put the robot on a straight piece of track. It should drive along with only small corrections.
- 3. Gentle curve.** Watch it follow the bend.
- 4. Full lap.** Time how long one lap takes, and note where it loses the line.

Tuning the speed

Speed is the biggest single setting. Too slow and the motors may hum without moving; too fast and the robot overshoots curves, because by the time it reacts, it has already left the line.

EXAMPLE

Ishaan from Pune tested his robot on the same oval track at different base speeds and wrote down what happened:

baseSpeed	Lap time	Lost the line?
100	28 s	Never
150	19 s	Twice, at the sharp bend
130	22 s	Never
140	20 s	Once, at the sharp bend

He chose 130: the fastest speed that never lost the line. A table like this turns guessing into evidence.

Trying spin turns

If the robot loses the line on sharp bends even at slow speeds, make the inner wheel run backward. In `loop()`, change `drive(0, turnSpeed)` to `drive(-turnSpeed / 2, turnSpeed)`, and change `drive(turnSpeed, 0)` to `drive(turnSpeed, -turnSpeed / 2)`. The robot now turns much more tightly.

REMEMBER

Change one thing at a time, and test after every change. If you change speed, sensor height and turn type together, you will never know which change helped.

DID YOU KNOW

As batteries drain, the motors get less voltage and the robot slows down. A setting that is perfect with fresh cells may be too slow later. Always tune with charged batteries, and keep a spare set.

CHAPTER 9

Troubleshooting

When something goes wrong, do not rebuild the whole robot. Find the cause step by step. Start with the simplest checks: is the battery charged and switched on, and are all wires firmly in place?

Common problems

Problem	Likely cause	What to do
Robot spins in circles	One motor runs backward	Swap that motor's wires at the L298N
Robot steers away from the line	Sensors or motors swapped left to right	Swap the sensor wires on A0 and A1, or swap the two motors' connections
Robot ignores the line	Sensors badly calibrated, too high, or <code>ON_BLACK</code> wrong	Run the test sketch; adjust the knobs and height
Motors do nothing, sensor LEDs work	Battery off, enable jumpers left on, or no common ground	Check the switch, ENA and ENB wiring, and the GND wire
Motors hum but do not turn	Speed value too low or weak battery	Raise <code>baseSpeed</code> ; charge the battery
Overshoots curves	Too fast	Lower <code>baseSpeed</code> ; try spin turns

Problem	Likely cause	What to do
Wobbles on straight lines	Sensors too close to the line edges, or speed too high	Space the sensors a little wider; slow down
Works in class, fails on stage	Bright lights or sunlight confuse the sensors	Recalibrate on the spot; shade the sensors
Stops suddenly for no reason	Both sensors read black (shadow, dirt) or the Arduino reset	Clean the track; check the batteries
Upload fails	Wrong board or port selected	Check the Tools menu settings and the cable

Brown-outs

If the robot restarts every time the motors start (you will see the three-second wait happen again), the battery voltage is dipping too low. This is called a **brown-out**. Use fresh or charged batteries, keep power wires short, and never use a 9 V PP3 battery: it cannot supply the current that motors need.

Shading the sensors

A short skirt of black chart paper around the sensors blocks stray light from the sides. Keep it clear of the floor so it does not drag.

TIP

Keep a small notebook for your robot. Write down every problem, what caused it and what fixed it. You will be surprised how often the same problem comes back.

TRY THIS

Ask a friend to secretly change one thing, such as loosening a GND wire or swapping two sensor wires. Then use this chapter's table to find the fault. It is the best practice for exhibition day.

Improvements: more sensors and the PID idea

Two sensors can only tell three things: the line is to the left, to the right or in between. Better robots know how far off the line they are, and correct gently or strongly to match.

Three sensors

Add a third sensor in the middle, sitting over the line. Now the robot can tell:

- Middle on black: centred, go forward
- Left on black: turn left
- Right on black: turn right
- All on white: the line is lost, so keep turning towards the side where the line was last seen

That last idea, remembering where the line was, helps the robot recover after a sharp bend.

Five sensors

Five-channel sensor boards place five sensors in a row across the line. Each sensor gets a **weight** showing its position: -2, -1, 0, 1, 2 from left to right. The average weight of the sensors that see black is the **error**: how far, and to which side, the line is from the centre. An error of 0 means the line is dead centre; +2 means it is far to the right.

PID explained simply

PID is a way of turning that error into a smooth correction. Imagine riding a bicycle along the edge of a road.

- **P (proportional)**: the further you drift, the harder you steer back. Small drift, small correction.
- **I (integral)**: if you keep drifting slightly to one side for a long time, you add a little steady correction for it. Simple line followers often leave this out.
- **D (derivative)**: if you are swinging back towards the edge quickly, you ease off early so you do not overshoot. D calms down wobbling.

$\text{Correction} = K_p \times \text{error} + K_i \times (\text{sum of past errors}) + K_d \times (\text{change in error})$

Kp, **Ki** and **Kd** are numbers you choose by testing.

A five-sensor sketch with P and D

Connect the five outputs, left to right, to A0, A1, A2, A3 and A4. The motor wiring stays the same.

```
// Five-sensor line follower with P and D control
const int SENSORS[5] = {A0, A1, A2, A3, A4}; // left to right
const int WEIGHTS[5] = {-2, -1, 0, 1, 2};
const int ON_BLACK = HIGH;
const int ENA = 5, IN1 = 7, IN2 = 8; // left motor
const int ENB = 6, IN3 = 4, IN4 = 3; // right motor

int baseSpeed = 120;
float Kp = 5.0; // starting guesses: tune by testing
float Kd = 15.0;
int lastError = 0;

void setup() {
    for (int i = 0; i < 5; i++) pinMode(SENSORS[i], INPUT);
    pinMode(ENA, OUTPUT); pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
    pinMode(ENB, OUTPUT); pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
    delay(3000);
}

void loop() {
    int sum = 0, count = 0;
    for (int i = 0; i < 5; i++) {
        if (digitalRead(SENSORS[i]) == ON_BLACK) {
            sum += WEIGHTS[i];
            count++;
        }
    }
    // error from -20 (far left) to 20 (far right); if lost, keep the last one
    int error = (count > 0) ? sum * 10 / count : lastError;
    int correction = (int)(Kp * error + Kd * (error - lastError));
    lastError = error;
    drive(baseSpeed + correction, baseSpeed - correction);
}

void drive(int leftSpeed, int rightSpeed) {
    setMotor(ENA, IN1, IN2, leftSpeed);
    setMotor(ENB, IN3, IN4, rightSpeed);
}

void setMotor(int enPin, int inA, int inB, int speed) {
    digitalWrite(inA, speed >= 0 ? HIGH : LOW);
    digitalWrite(inB, speed >= 0 ? LOW : HIGH);
    analogWrite(enPin, constrain(abs(speed), 0, 255));
}
```

The code multiplies the average weight by 10, so the error runs from -20 to 20 without fractions. When the line is to the right, the error is positive, so the left wheel speeds up and the right wheel slows down, and the robot steers right, back onto the line.

TIP

To tune, set Kd to 0 first. Raise Kp until the robot follows the line but wobbles. Then raise Kd step by step until the wobble settles. Only then try a higher base speed.

CHAPTER 11

Presenting at an exhibition

A working robot is half the job. The other half is explaining it clearly to judges, teachers and visitors.

Before the day

- Charge all batteries, and pack a spare set.
- Carry your own track, rolled up, plus spare tape, a screwdriver, jumper wires and the USB cable.
- Save your code on the laptop and on a pen drive.
- Make a simple chart: the robot's block diagram as boxes and arrows (sensors, Arduino, driver, motors, battery), your logic table, and your speed-tuning results.

When you arrive

1. Lay out the track away from windows and spotlights if you can.
2. Run the test sketch and recalibrate both sensors under the hall's lighting.
3. Do two practice laps before visitors arrive.

A one-minute explanation

"This is a line follower robot. Its two infrared sensors shine invisible light at the floor. White reflects the light and black absorbs it, so each sensor knows whether it is over the line. The Arduino reads the sensors thousands of times a second. If the left sensor sees the line, it turns left; if the right sensor sees it, it turns right. The L298N driver supplies the current to run the motors, and the Arduino controls their speed using PWM."

EXAMPLE

At her school exhibition in Chennai, Lakshmi let visitors slide a strip of black tape under the sensors while the robot was held in the air. They could see the wheels react, which explained the idea better than any chart.

Viva questions and model answers

- 1. Why does the robot follow black tape?** Black absorbs infrared light and white reflects it, so the sensors can tell them apart.
- 2. What does the knob on the sensor do?** It sets the sensitivity: how much reflected light counts as white.
- 3. Why do you need a motor driver?** An Arduino pin can safely give only about 20 mA, while motors need hundreds of milliamps and need reversing. The driver handles both.
- 4. What is PWM?** Switching the motor's power on and off very fast. The longer the on-time in each cycle, the faster the motor runs.
- 5. What happens if both sensors see black?** The robot stops, because that usually means a crossing or the finish line.
- 6. How would you follow a white line on a black floor?** Change one setting, `ON_BLACK`, so the robot treats white as the line.
- 7. Where are line followers used in real life?** Automated guided vehicles carry goods along marked paths in factories, warehouses and some hospitals.
- 8. How could you make it faster?** Use more sensors and PID control, so it corrects smoothly instead of zig-zagging.

REMEMBER

If you do not know an answer, say so honestly and explain how you would find out. Judges value honesty more than a guess.

CHECK YOURSELF

Practice questions

1. What are the three steps every robot repeats, in order?

2. Why does a white surface give a different sensor reading from a black surface?
3. What two parts does each IR sensor module have, side by side?
4. Why are the sensors mounted ahead of the wheels?
5. How far above the floor should the sensors usually sit?
6. For a line about 2 cm wide, roughly how far apart should the two sensor centres start?
7. What is the job of the blue potentiometer on each sensor?
8. Why might a robot tuned in a classroom fail on a brightly lit stage?
9. Why do the two sensors connect to a mini breadboard first?
10. Which Uno pin should the battery feed, and why not the 5V pin?
11. What does the wire between Uno GND and L298N GND do?
12. In the logic table, what should the robot do when only the left sensor sees black?
13. What is the difference between a pivot turn and a spin turn?
14. What does `drive(0, turnSpeed)` make the robot do?
15. Which single line do you change if your sensors read 0 over black?
16. Why does the code wait three seconds in `setup()` ?
17. Your robot spins in circles. What is the likely cause and fix?
18. Your robot steers away from the line. Give two possible causes.
19. In Ishaan's table, why did he choose a base speed of 130?
20. What does the error mean in a five-sensor robot?
21. In PID, which term helps stop the robot wobbling?
22. Give one real-world use of line-following vehicles.

Answers

1. Sense, decide, act, and then repeat.
2. White reflects most of the infrared light back to the detector, while black absorbs most of it.
3. An IR LED that shines light down, and a light detector that measures how much bounces back.
4. So the robot sees a bend before its wheels reach it, giving it time to react.
5. About 5 to 10 mm, found by testing.
6. About 3 cm, so both see white when the robot is centred.
7. It sets the sensitivity: how much reflected light counts as white.
8. Sunlight and some bright lamps contain infrared light that can fool the sensors, so they need recalibrating.
9. The Uno has only one 5V pin on its header, so the breadboard rails share 5 V and GND between both sensors.
10. VIN, because the Uno's regulator turns 7 to 12 V into a steady 5 V. The 5V pin bypasses the regulator.
11. It gives both boards a common ground, so the driver can understand the Arduino's signals.
12. Turn left, to bring the line back between the sensors.
13. In a pivot turn the inner wheel stops; in a spin turn it runs backward, which turns more tightly.
14. A pivot turn to the left: left wheel stopped, right wheel forward.
15. `const int ON_BLACK = HIGH;` becomes `const int ON_BLACK = LOW;`.
16. To give you time to place the robot on the track before it moves.
17. One motor runs backward. Swap that motor's two wires at the L298N.
18. The sensor wires on A0 and A1 are swapped, or the left and right motors are swapped.
19. It was the fastest speed at which the robot never lost the line.
20. How far, and to which side, the line is from the centre of the sensor row.
21. D, the derivative term.
22. Automated guided vehicles carrying goods in factories, warehouses or hospitals.

Glossary

AGV

Automated guided vehicle, a robot cart that follows a marked path to carry goods.

Brown-out

A dip in battery voltage that makes the Arduino restart.

Calibration

Adjusting a sensor so it gives correct readings in real conditions.

Caster

A free-turning wheel or ball that supports a robot without driving it.

Common ground

A shared GND connection between the Arduino, driver and battery.

Differential drive

Steering by running the left and right wheels at different speeds.

Error

In line following, how far and to which side the line is from the centre.

Infrared (IR)

Light just beyond red that human eyes cannot see.

L298N

A motor driver that controls the speed and direction of two DC motors.

Logic table

A table listing every combination of inputs and the action for each one.

PID

A control method that uses the error, its history and its rate of change to correct smoothly.

Pivot turn

A turn with the inner wheel stopped and the outer wheel driving.

Potentiometer

An adjustable resistor; on an IR module it sets the sensitivity.

PWM

Pulse width modulation: switching power on and off fast to control a motor's speed.

Spin turn

A turn with one wheel forward and the other backward, for tight bends.

TCRT5000

A common IR sensor with its LED and detector in one small block.

Threshold

The level that decides whether a reading counts as black or white.

Weight

A number giving a sensor's position in a row, used to calculate the error.



Gurukul Technicals Institute

Computer education institute in Delhi, with authorised member centres, robotics classes and science projects, working models and DIY kits.

COURSES

Computer & Office Skills · Accounting & Business Software · Programming & Web Development · Design & Digital Skills · Hardware & Networking · Robotics & Coding · Other Programmes

PROJECTS AND KITS

Ready-made school projects · DIY and STEM kits · Robotics and coding · Components and spares · Engineering projects

CONTACT

Head office: A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone: 011 2721 1512 · **Mobile and WhatsApp:** +91 97116 99511

Email: engineergurukul@yahoo.co.in

Website: engineergurukul.com · **Instagram:** [@science_projects_club](https://www.instagram.com/science_projects_club)

More free ebooks for computer courses, robotics and science projects: engineergurukul.com/resources/ebooks/

Free to read, print and share. Not for sale.