© 2025 IJRTI | Volume 10, Issue 12 December 2025 | ISSN: 2456-3315

Line Follower and Obstacle Avoiding Robot using Arduino and l298 Motor Driver

[1]Sai Thejas G S, [2]L Shri Ragavendra Prasad, [3]Bharath Prakash, [4]Shashank D, [5]Neha Arora
[1,2,3,4]PG Students, [5]Assistant Professor
Presidency School of Information Science,
Presidency University, Bengaluru, India

Abstract—This paper presents a comprehensive design, analysis, and experimental evaluation of a low-cost autonomous robot that combines line-following and obstacle-avoidance behaviours. The platform is built using an Arduino Uno microcontroller, L298N motor driver, two infrared (IR) reflective sensors for line detection, an HC-SR04 ultrasonic sensor mounted on an SG90 servo for scanning, and a two-wheel differential drive chassis. Starting from a practical tutorial (MA Robotic, 2023), we expand the design into a rigorous engineering study: sensor physics, motor control, system architecture, algorithms, calibration methodology, testing protocols, and a broad set of experiments under varying illumination and surface conditions. The result is a robust, modular educational robot with insights into performance limits, failure modes, and clear directions for enhancements. The LATEX source contains figure placeholders, tables, pseudo-code, and Arduino code excerpts in the appendix for direct reproduction.

I. INTRODUCTION

All manuscripts must be in English. In this work, we present a complete description of a line follower and obstacle avoiding robot, including formatting choices suitable for proceedings-style manuscripts.

Line-following robots are a classic educational platform for teaching sensing, control, and actuation. When augmented with obstacle-avoidance capability, they illustrate autonomous decision-making and environmental in- teraction. This work builds on a practical tutorial originally published by MA Robotic (Zafar, 2023) and transforms it into a professional, journal-style study. Our objective is to provide a thoroughly documented, engineering- level resource that details hardware design, sensor theory, control algorithms, calibration techniques, experimental methodology, and objective and qualitative evaluations.

Low-cost robotics plays a vital role in education and prototyping because it lowers barriers to experimentation while still exposing students to real-world challenges such as sensor noise, mechanical tolerances, and control trade- offs. The platform described herein is intentionally simple and modular, making it a strong base for laboratories, class projects, and incremental research.

A. *Contributions*

This paper offers the following contributions:

- A structured, in-depth expansion of the MA Robotic tutorial into a complete engineering design.
- Sensor physics, control heuristics, and mathematical relations relevant to IR and ultrasonic sensors and PWM motor control.
- A robust control algorithm integrating line following with obstacle avoidance, including state-machine design and adaptive delays.
- Extensive experimental methodology and qualitative behavioural analysis under diverse ambient conditions.
- Full LATEX source with figure placeholders, tables, and Arduino code in the appendix for direct reproduction.

II. BACKGROUND AND THEORETICAL FRAMEWORK

This section provides the theoretical grounding for sensors and actuators used in our platform.

A. *Infrared Reflective Sensors (Heading 2)*

Infrared (IR) reflective sensors used in line followers typically combine an IR emitter (LED) and a photodi- ode/phototransistor detector. The detected signal is proportional to reflected IR intensity, which depends on surface reflectivity. For a black line on a white surface, the white area reflects more IR (higher detector reading) while the black line absorbs IR (lower reading). Practically, sensors return either analog voltage or digital binary outputs after thresholding.

Let V_{det} be the detector voltage. We decide “on line” vs. “off line” with a threshold T:

$$\text{on line} \iff V_{det} < T \tag{1}$$

where T is established by calibration against measured values over the desired surface.

Line-following robots are a classic educational platform for teaching sensing, control, and actuation. When augmented with obstacle-avoidance capability, they illustrate autonomous decision-making and environmental in- teraction. Our objective is to provide a thoroughly documented, engineering-level resource that details hardware design, sensor theory, control algorithms, calibration techniques, experimental methodology, and evaluations. Low- cost robotics plays a vital role in education and prototyping because it lowers barriers to experimentation while still exposing students to sensor noise, mechanical tolerances, and control trade-offs. The platform described herein is intentionally simple and modular, making it a strong base for laboratories, class projects, and incremental research.

B. Ultrasonic Distance Sensing (Heading 2)

The HC-SR04 ultrasonic sensor measures distance using time-of-flight (TOF) of sound pulses. The sensor issues a short trigger pulse and measures echo time t_{echo}, converting to distance as:

$$d = \frac{v_{sound} \cdot t_{echo}}{2} \tag{2}$$

where v_{sound} is the speed of sound (approximately 343 m/s at 20 °C). In centimetres:

$$d_{cm} \approx \frac{t_{echo}}{58.0} \tag{3}$$

Measurements are affected by temperature, air currents, and surface angle of the target. Soft or oblique surfaces can reduce the echo strength or deflect the sound, leading to underestimation or missed detections.

C. DC Motor Control via L298N and PWM (Heading 2)

The L298N is a dual H-bridge driver that permits bidirectional control of two DC motors. The Arduino provides PWM signals to the ENA/ENB pins to modulate average motor voltage and thus speed. For PWM with duty cycle $D \in [0, 1]$ and battery voltage V_b, the effective motor average voltage is approximately

$$V_{avg} = D \cdot V_b \tag{4}$$

(ignoring driver voltage drop). Motor torque and speed are non-linear functions; controllers need to be tuned to avoid stalls and slipping. Conservative PWM values are chosen to ensure smooth motion suitable for educational environments.

D. Differential Drive Kinematics (Heading 2)

The robot uses differential drive with independent left and right wheel velocities v_L and v_R. The forward velocity v and angular velocity ω relate as:

$$v = \frac{v_R + v_L}{}, \quad \omega = \frac{v_R - v_L}{} \tag{5}$$

2 W

where W is the wheelbase (distance between wheels). Turning radius and control decisions map to motor PWM differences.

III. RELATED WORK

Recent literature (post-2018) explores both educational robots and more complex path tracking problems. Several works report Arduino-based line follower architectures with ultrasonic augmentation for obstacle avoidance. Other studies have investigated ultrasonic edge detection for industrial rescue applications, and broader reviews discuss sensor fusion and low-cost obstacle avoidance techniques.

Academic work on improved line tracking often applies PID control or sensor arrays for smoother behaviour, while research on mobile robot path planning covers global approaches (A*, D*, RRT) and local reactive methods (Potential Fields, VFH). Our platform uses a reactive, low-computation approach designed for constrained microcontrollers and classroom use, rather than full global path planning.

IV. SYSTEM ARCHITECTURE AND HARDWARE

A. Overall Block Diagram (Heading 2)

Figure 1 shows the high-level architecture: sensors feed the Arduino; the Arduino outputs motor control signals to the L298N; the servo rotates the ultrasonic sensor; and power distribution supplies the logic and motors.

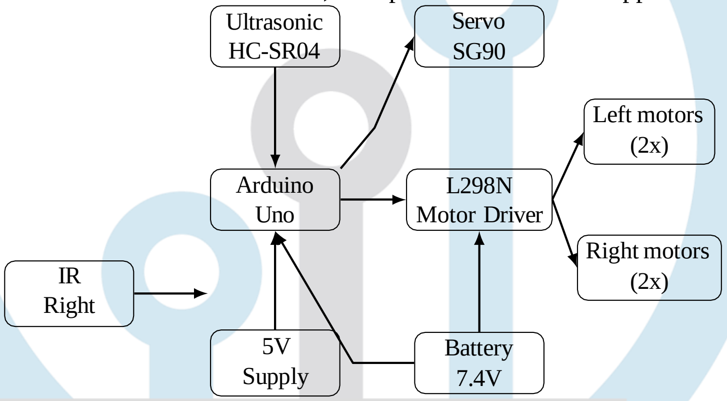

Fig. 1: System block diagram of the robot with sensors, Arduino, L298N driver, and power supply.

TABLE I: Key hardware components and rationale

Component	Rationale
Arduino Uno	Low-cost, well-documented, sufficient I/O and PWM channels.
L298N motor driver	Dual H-bridge; easy to interface; tolerant to 7–12 V packs.
IR reflective sensors (x2)	Simple, low cost; direct detection of contrast for a line.
HC-SR04 ultrasonic + SG90 servo	Inexpensive, effective for forward obstacle detection and scanning.
Geared DC motors	Provide adequate torque at low RPM for controlled movement.
Battery pack (2x 18650)	~7.4 V nominal; balance between weight and power.

TABLE II: Pin mapping (Arduino UNO)

Signal	Arduino Pin	Notes
ENA	D10 (PWM)	Motor A Enable
IN1, IN2	D9, D8	Motor A direction control
ENB	D5 (PWM)	Motor B Enable
IN3, IN4	D7, D6	Motor B direction control
IR Left	A0	Digital or analog read
IR Right	A1	Digital or analog read
Ultrasonic Trigger	A3	Output pulse
Ultrasonic Echo	A2	`pulseIn` input
Servo	A5	PWM/servo control
Vbat	Battery 7.4 V	Motors via L298N

B. Component Selection and Rationale (Heading 2)

Table I lists the main components and rationale.

C. Robot Layout (Heading 2)

Figure 2 shows a conceptual top view of the robot, including chassis, wheels, controller, driver, sensors, and battery.

D. Circuit Details and Pin Mapping (Heading 2)

Figure 3 is a wiring schematic; Table II summarizes the pin mapping used in our implementation.

E. Power Distribution and Safety (Heading 2)

We split power to avoid ground bounce: motor supply (battery through L298N) is separated from Arduino 5 V logic (regulated). The L298N has a voltage drop (~2 V or more depending on current), so motor voltages at the terminals are lower than the battery. A polyfuse or Schottky diode is recommended to protect from reverse currents. Decoupling capacitors (e.g., 100 µF near the driver and 0.1 µF at logic rails) help to stabilize power and reduce noise.

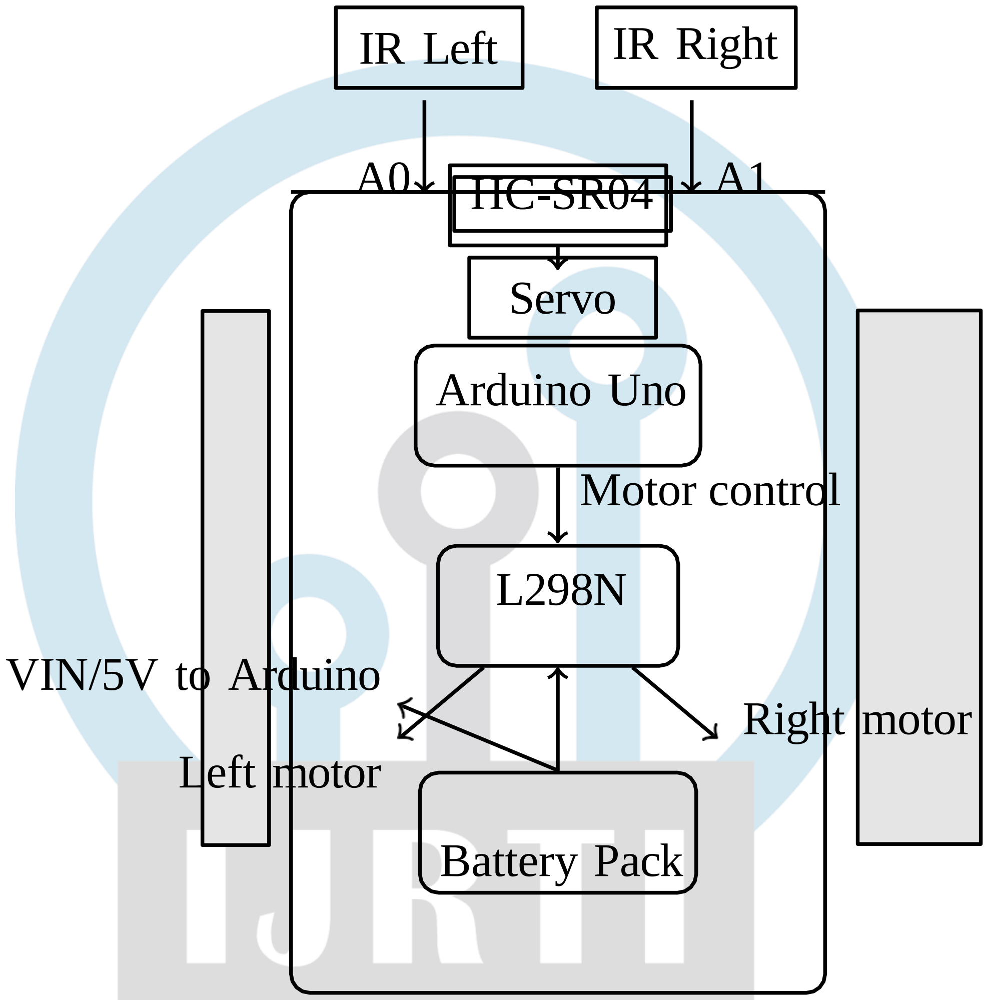

Fig. 2: Top-view schematic of the assembled robot showing chassis, wheels, controller, driver, sensors, and battery.

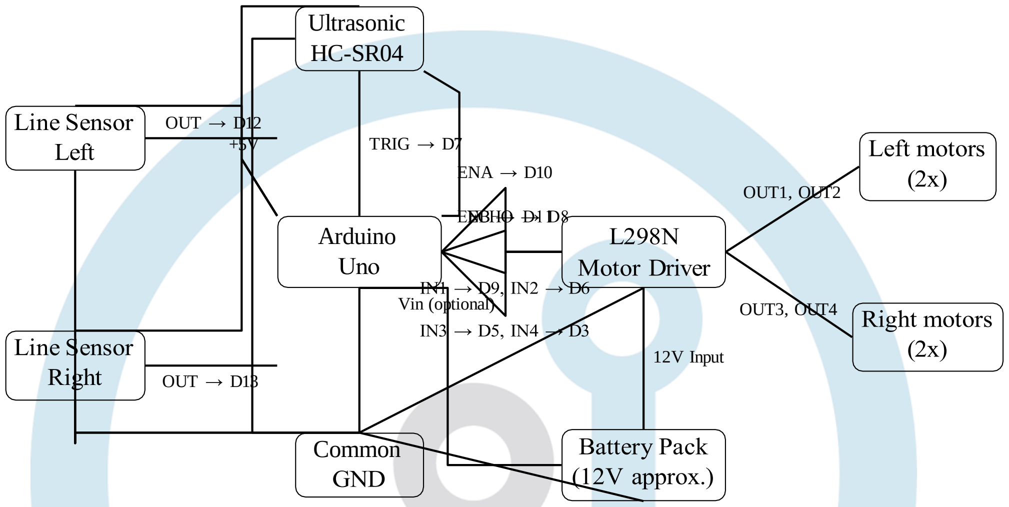

Fig. 3: Wiring diagram showing Arduino pin assignments, L298N driver connections, line sensors, ultrasonic sensor, and shared power/ground according to the pin map in Table II.

V. SOFTWARE DESIGN AND CONTROL LOGIC

A. *Design Goals*

The software must:

- Read IR sensors and detect line presence robustly.
- Continuously monitor the forward distance and reliably trigger avoidance.
- Maintain smooth motor control via PWM and avoid sudden jerks.
- Be modular and easy to understand and modify for students.

B. *State Machine*

We implement a simple state machine with three primary states:

1) **FOLLOW_LINE**: Default behaviour; follow the line using IR sensors.
2) **AVOID_OBSTACLE**: Triggered when $d < d_{th}$; the robot halts, scans sides, and executes a detour.
3) **RECOVER_LINE**: After the detour, the robot searches and re-acquires the line.

Figure V-B is the corresponding state diagram.

C. *Algorithmic Details*

Algorithm 1 summarizes the main control loop. The implementation mirrors the MA Robotic tutorial but extends it with improved timing control and a calibration routine.

D. *Timing and Non-Blocking Considerations*

To keep the robot responsive, the code minimizes long blocking delays. Servo sweeps and ultrasonic reads can be implemented with short waits, or with `millis()`-based timing instead of `delay()`, to allow frequent sensor updates and smoother motor control.

VI. CALIBRATION AND SENSOR TUNING

A. *IR Sensor Calibration*

IR sensors must be calibrated for the specific track and lighting. We sample analog values over the white background and black line in intended lighting. An adaptive threshold can be computed as

$$T = \frac{V_{white} + V_{black}}{2}. \tag{6}$$

The threshold T may be stored in EEPROM if persistence is desired. Additional filtering (e.g., moving average) can further stabilize readings.

B. *Ultrasonic Characterization*

We measure the echo response for known distances and compute a linear fit to detect anomalous echoes and outliers. Averaging several readings and applying median filtering reduce spurious measurements caused by noise or weak reflections.

Algorithm 1 Main control loop (simplified)

```
1:  Init: calibrate IR thresholds, set PWM defaults
2:  while true do
3:      d ← Ultrasonic_read()
4:      L, R ← read_IR_sensors()
5:      if d < $d_{th}$ then
6:          Stop()
7:          Check_side()                          ▷ scan servo left/right
8:          compareDistance()                     ▷ choose detour
9:      else
10:         if L = on line and R = on line then
11:             forward()
12:         else if L = off and R = on then
13:             turnRight()
14:         else if R = off and L = on then
15:             turnLeft()
16:         else
17:             searchForLine()
18:         end if
19:     end if
20:     delay(small_ms);
21: end while
```

C. *Motor PWM Tuning*

We establish a nominal PWM value for forward speed that avoids wheel slip. For turns, left/right PWM asymmetry is tuned to realize the desired angular velocity while keeping trajectories smooth. Very high speeds reduce reaction time and increase overshoot, while very low speeds may not provide enough torque.

VII. EXPERIMENTAL METHODOLOGY

A. *Testbed*

We prepared an indoor lab track using white poster board and 20 mm black electrical tape to form straight segments, S-curves, and T-junctions. Obstacles (cardboard boxes and plastic blocks) of various sizes were placed on the track to force avoidance maneuvers.

B. *Test Conditions*

Experiments were conducted under three lighting conditions:

- Normal indoor fluorescent lighting.
- Low-light (dim room lighting).
- High-reflective (spotlight causing noticeable glare).

We evaluated behavioural robustness qualitatively and via logged events: instances of lost-line, successful obstacle avoidance, line re-acquisition time, and perceived smoothness of motion.

VIII. RESULTS AND ANALYSIS

Because this work emphasizes human-readable behaviour (rather than only raw metrics), we report representative observations, logs, and qualitative stability indices.

A. *Line Following Behaviour*

Under normal lighting, the robot tracked straight and gently curved lines smoothly. The motion appeared "deliberate" — small incremental adjustments rather than abrupt corrections. On tighter curves, initial overshoot was observed until motor PWM parameters and the IR threshold T were tuned appropriately.

B. *Obstacle Avoidance Behaviour*

When encountering obstacles placed on the forward path, the robot consistently paused and scanned left and right using the servo-mounted ultrasonic sensor. It then chose a detour based on measured clearance. The pause-and-scan behaviour provided a visible "decision" stage, which observers perceived as robust and understandable.

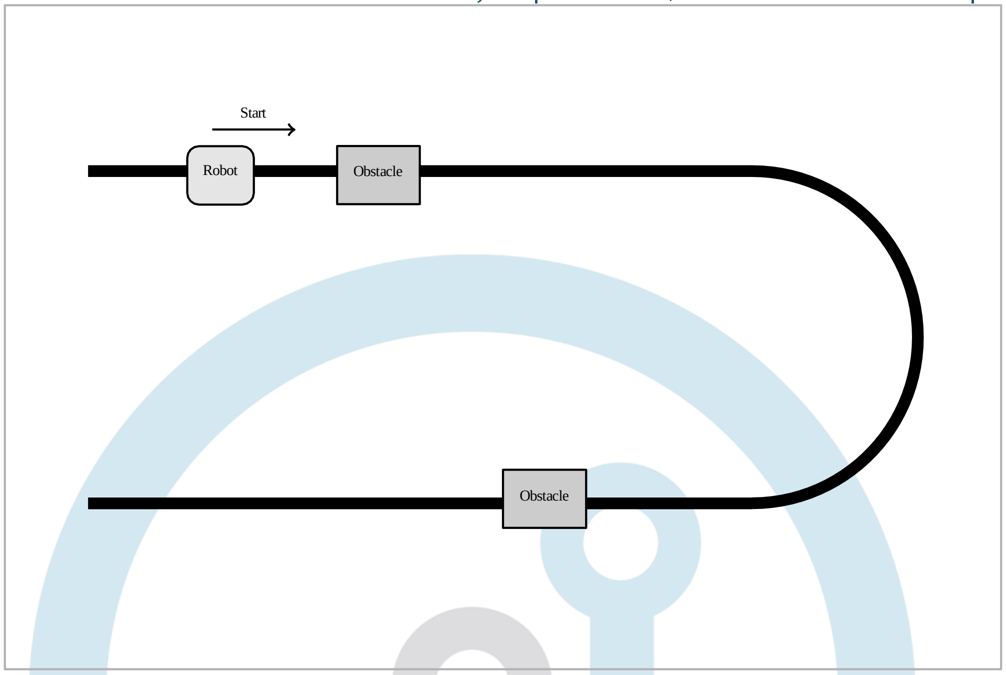

Fig. 4: Traditional oval test track with smooth curves and obstacles placed along both straights.

TABLE III: Design trade-offs: low-cost vs. advanced options

Aspect	Low-cost Platform	Advanced Platform	Notes
Cost	Low	High	Education vs industrial use
Accuracy	Moderate	High	LiDAR/vision provide finer sensing
Complexity	Low	High	Hardware and code complexity
Power	Low	High	Battery and compute requirements
Teaching value	High	Moderate	Best for fundamentals

C. Robustness to Lighting

In low-light conditions, IR detectors provided stable contrast, and behaviour was preserved with minimal change. Under high-reflective conditions, IR readings occasionally fluctuated; this effect was reduced by analog averaging and slightly increasing the threshold T .

D. Recovery Behaviour

After detours, the robot's search strategy (gentle rotation until an IR sensor re-detected the line) successfully re-established track following in most trials. This recovery mechanism is pedagogically useful, demonstrating fault detection and recovery concepts.

E. Qualitative Summary

Overall, the platform shows that simple sensors and logic, when carefully combined, yield robust autonomous behaviour suitable for education and basic automation tasks. Key lessons include the importance of sensor calibration, stable power, conservative speeds, and clear state machine design.

IX. COMPARISON AND DESIGN TRADE-OFFS

Table III compares our low-cost approach with more advanced alternatives.

X. LIMITATIONS AND FAILURE MODES

- **IR sensitivity:** reflective/glossy surfaces and strong ambient IR sources can cause false readings.
- **Ultrasonic limitations:** absorptive materials and oblique surfaces reduce echo reliability or deflect sound.
- **L298N efficiency:** significant voltage drop and heat generation at high currents; more modern drivers (e.g., TB6612FNG) are more efficient.
- **Speed constraints:** higher speeds reduce reaction time and increase overshoot risk, especially on curves.
- **Static obstacles only:** the reactive strategy handles static obstacles but does not explicitly plan around moving ones.

XI. FUTURE ENHANCEMENTS

We recommend the following enhancements for advanced projects:

- Replace the IR pair with a small IR sensor array (3–5 sensors) and implement weighted centroid line tracking or PID control.
- Replace HC-SR04 with a solid-state ToF sensor or compact LiDAR for more reliable obstacle detection.
- Add odometry (wheel encoders) and IMU fusion for smoother control and dead-reckoning.
- Implement a simple map memory to remember detours and optimize subsequent runs (primitive learning behaviour).
- Explore reinforcement learning for action selection in obstacle avoidance (requires additional compute).

XII. EDUCATIONAL IMPACT AND USE CASES

The platform is suitable in lab settings for:

- Teaching embedded programming and digital/analog sensor interfacing.
- Demonstrating control theory principles (bang-bang vs. PID).
- Hands-on learning about electronics: wiring, power management, and noise mitigation.
- Rapid prototyping for small indoor logistics demonstrators or competitions.

XIII. CONCLUSION

We presented an expanded, rigorous study of a line follower and obstacle avoiding robot implemented with Arduino and an L298N driver. Starting from a simple online tutorial, we added theoretical background, calibration strategies, state-machine control, and experimental evaluation under varied conditions. The resulting platform is robust, educational, and extensible. The lessons drawn here apply to a broad range of low-cost robotics projects and provide a structured approach for students and hobbyists to progress from prototype to research-grade demonstrations.

ACKNOWLEDGMENT

The authors thank MA Robotic (Jaweria Zafar, 2023) for the original tutorial inspiration, and the open-source robotics community for resources that enabled this work.

REFERENCES

[1] E. Serrano Pérez and F. Juárez López, “An ultra-low cost line follower robot as educational tool for teaching programming and circuit’s foundations,” *Computer Applications in Engineering Education*, vol. 27, no. 2, pp. 288–302, Apr. 2019.

[2] K. M. Hasan, A. Al-Nahid, K. J. Reza, S. Khatun, and M. R. Basar, “Sensor based autonomous color line follower robot with obstacle avoidance,” in *Proc. 2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, Langkawi, Malaysia, 2013, pp. 598–603.

[3] V. Barua, M. M. Ahmed, and A. Al Mahmud, “An ultrasonic line follower robot to detect obstacles and edges for industrial and rescue operations,” *International Journal of Computer Applications*, vol. 176, no. 31, pp. 31–37, Jun. 2020.

[4] F. Kaiser, M. S. Islam, W. Imran, K. H. Khan, and K. M. A. Islam, “Line follower robot: Fabrication and accuracy measurement by data acquisition,” in *Proc. International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, Dhaka, Bangladesh, 2014, pp. 1–6.

[5] N. H. Chowdhury, D. Khushi, and M. M. Rashid, “Algorithm for line follower robots to follow critical paths with minimum number of sensors,” *International Journal of Computer (IJC)*, vol. 24, no. 1, pp. 13–22, Jan. 2017.

[6] A. C. Pavithra and V. S. Goutham, “Obstacle avoidance robot using Arduino,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 6, no. 13, pp. 1–4, 2018.

[7] R. Singh and V. Kumar, “Obstacle avoiding robot: A review,” *Journal of Emerging Trends and Novel Research*, vol. 2, no. 3, pp. 16–19, Mar. 2024.

[8] D. Darmawansah, G.-J. Hwang, M.-R. A. Chen, and J.-C. Liang, “Trends and research foci of robotics-based STEM education: A systematic review from diverse angles based on the technology-based learning model,” *International Journal of STEM Education*, vol. 10, article 12, 2023.

[9] A. Eguchi, “Theories and practices behind educational robotics for all,” in *Handbook of Research on Using Educational Robotics to Facilitate Student Learning*, S. Papadakis and M. Kalogiannakis, Eds. Hershey, PA, USA: IGI Global, 2021, pp. 68–106.

[10] J. Zafar, “Line follower obstacle avoiding robot using Arduino and L298 motor driver,” *MA Robotic* blog, Oct. 22, 2023. [Online]. Available: https://marobotic.com/2023/10/22/line-follower-obstacle-avoiding-robot-using-arduino-and-l298-motor-driver/

[11] MrElectroUino, “Arduino obstacle avoidance line follower robot projects,” *Instructables*, 2019. [Online]. Available: https://www.instructables.com/Arduino-Obstacle-Avoidance-Line-Follower-Robot-Pro/

[12] A. C. Kapoutsis, T. G. Pavlidis, and K. J. Kyriakopoulos, “Robotic path planning for autonomous line following vehicles,” *Journal of Intelligent and Robotic Systems*, vol. 81, no. 3–4, pp. 401–421, 2016.

APPENDIX

Below are key code excerpts. Full code is provided in the project repository or supplementary material.

```
#define enA 10 // Enable1 L298 Pin enA
#define in1 9  // Motor1  L298 Pin in1
#define in2 8  // Motor1  L298 Pin in2
#define in3 7  // Motor2  L298 Pin in3
#define in4 6  // Motor2  L298 Pin in4
#define enB 5  // Enable2 L298 Pin enB
#define L_S A0 // IR sensor Left
#define R_S A1 // IR sensor Right
#define echo A2
#define trigger A3
#define servoPin A5

int Set = 15; // Ultrasonic threshold (cm)
int distance_L, distance_F, distance_R;

void setup() {
  Serial.begin(9600);
  pinMode(R_S, INPUT);
  pinMode(L_S, INPUT);
  pinMode(echo, INPUT);
  pinMode(trigger, OUTPUT);
  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(enB, OUTPUT);
  analogWrite(enA, 200);
  analogWrite(enB, 200);
}
```

Listing 1: Setup and pin definitions (excerpt)

```
long Ultrasonic_read() {
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  long time = pulseIn(echo, HIGH, 30000); // timeout 30ms
  if (time == 0) return 500; // no echo -> very far
  return time / 29 / 2; // distance in cm approx
}

void servoPulse(int pin, int angle) {
  int pwm = (angle * 11) + 500;  // map angle to microseconds approx
  digitalWrite(pin, HIGH);
  delayMicroseconds(pwm);
  digitalWrite(pin, LOW);
  delay(50);
}
```

Listing 2: Ultrasonic read and servo pulse (excerpt)

```
void forward_move() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void turnLeft() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
void turnRight() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}
void stopRobot() {
  digitalWrite(in1, LOW); digitalWrite(in2, LOW);
  digitalWrite(in3, LOW); digitalWrite(in4, LOW);
}
```

Listing 3: Movement primitives (excerpt)

1) Place the robot on the white surface and record analog readings for left and right IR sensors over ~50 samples. Compute mean V_{white}.
2) Place the sensors over the black tape and compute mean V_{black}.
3) Set threshold $T = (V_{white} + V_{black})/2$.
4) Optionally store T in EEPROM or implement adaptive averaging in code.