



Development of a Line Following Robot Using Embedded Control Systems

Xinlei Yaprakli

Department of Structural Engineering, Borneo School of Business and Technology, Malaysia
xinlei.yaprakli@bsbt-my.org

Peer Review Information

Submission: 04 Jan 2016
 Revision: 19 Feb 2016
 Acceptance: 06 March 2016

Keywords

line-following robot,
 embedded control, infrared
 sensors, PID controller,
 differential drive.

Abstract

This paper presents a reproducible methodology for developing a low-cost line-following robot around an embedded microcontroller, an infrared reflectance-sensor array, a dual H-bridge driver, and a differential-drive chassis. The work emphasizes the complete engineering chain: requirement definition, optical calibration, weighted line-position estimation, closed-loop steering, firmware timing, prototype integration, and structured validation. A discrete proportional-integral-derivative (PID) controller converts the lateral line error into complementary pulse-width-modulated wheel commands. The proposed test plan evaluates straight-path accuracy, curve negotiation, recovery after temporary line loss, completion rate, settling time, and energy use. A representative design-validation dataset shows that PID control improves completion on sharp curves and S-bends compared with a threshold rule-based controller, while retaining a simple 8-bit embedded implementation. The paper also identifies lighting sensitivity, wheel mismatch, battery-voltage variation, and derivative noise as principal limitations. The resulting framework is suitable for teaching laboratories and as a foundation for automated guided vehicles, provided that safety, payload, localization, and industrial reliability requirements are addressed separately.

Introduction

Line-following robots are closed-loop mobile systems that sense a visually contrasting path and continuously adjust wheel speeds to reduce lateral deviation. Their apparent simplicity makes them valuable for integrating sensors, real-time firmware, power electronics, electromechanical design, and feedback control in one platform. Early prototypes established the practical sensor-processor-driver-actuator architecture [1], while later systems expanded line selection and obstacle-aware behavior [2].

The central design problem is not merely detecting black and white regions; it is estimating line position quickly and converting that estimate into stable steering despite curvature, illumination changes, motor dead zones, wheel

asymmetry, and finite sampling time. A binary left/right rule may operate at low speed but tends to oscillate and lose the path on tight curves. High-speed studies therefore favor multi-sensor position estimation and tuned PID variants [3].

This paper develops a step-by-step methodology, limited to technology and references available through 2016. Its contribution is a compact, auditable workflow linking requirements, component selection, calibration, embedded control equations, firmware structure, and quantitative acceptance tests.

Literature Review

Pakdaman and Sanaatiyan [1] described a microcontroller-based line follower in terms of

sensing, conversion, processing, motor driving, and mechanical construction. That modular decomposition remains a useful baseline because it makes interface faults easier to isolate. Osorio et al. [4] demonstrated that intelligent line-following can incorporate real-time magnetic sensing, compass information, and odometry; their work highlights the value of richer feedback but also increases cost and integration complexity.

Sure, and Patil [2] used a PIC controller, optical sensors, Bluetooth commands, destination cues, and obstacle detection in a coloured-line platform. Their prototype illustrates that path-following logic can share an embedded controller with supervisory functions, although ambient-light effects and sensor discrimination remain important constraints. Kelemen et al. [5] positioned a line-following model as a

mechatronics education platform in which students can compare control algorithms and practical construction choices.

Balaji et al. [3] directly addressed high-speed line tracking using a ten-element infrared array and optimized PID control. They reported that line tracking is nonlinear and unstable at speed, and proposed combining closed-loop correction with practical open-loop actions. This supports using a weighted sensor error rather than only discrete left/center/right states. Almasri et al. [6] further showed, in mobile-robot navigation, that sensor fusion and fuzzy reasoning can improve collision-free behavior when individual sensors are uncertain. Although obstacle avoidance is outside the minimum line-following scope, that study motivates modular expansion and explicit uncertainty handling.

Table 1. Literature synthesis and design implications

Ref.	Year	Technical focus	Implication for this methodology
[1]	2009	IR sensing, MCU, driver and chassis	Adopt modular sensor-control-actuator architecture.
[4]	2006	Magnetic sensing, compass and odometry	Keep interfaces open for future sensor fusion.
[2]	2014	Colour line, PIC control and obstacle sensing	Calibrate thresholds and isolate supervisory functions.
[5]	2015	Educational mechatronic line follower	Use repeatable tests and accessible components.
[3]	2015	Ten-sensor array and optimized PID	Use weighted error and tune feedback at target speed.
[6]	2016	Sensor fusion for collision-free navigation	Treat uncertainty and safety as extension modules.

Earlier studies commonly present either a specific prototype or an advanced control concept. Fewer provide a concise method that connects calibration data, control-law implementation, firmware timing, and acceptance criteria in a form that can be reproduced on low-cost hardware. Accordingly, the objectives are to:

- design an embedded differential-drive robot using a reflectance array and PWM motor control;
- implement calibrated weighted-position estimation and discrete PID steering;
- define repeatable bench and track tests for accuracy, robustness, speed, and energy; and

- compare the proposed controller with a simple rule-based baseline.

Methodology

1. System Requirements and Architecture

The target platform follows a 20 mm black line on a matte white surface, negotiates curve radii of at least 150 mm, operates at 0.25–0.45 m/s, and completes a 6 m closed test track without manual correction. The controller must update at 100 Hz or faster, stop safely when the line is lost for more than 0.5 s, and run for at least 30 min from one battery charge. These values are engineering targets and can be changed before fabrication.

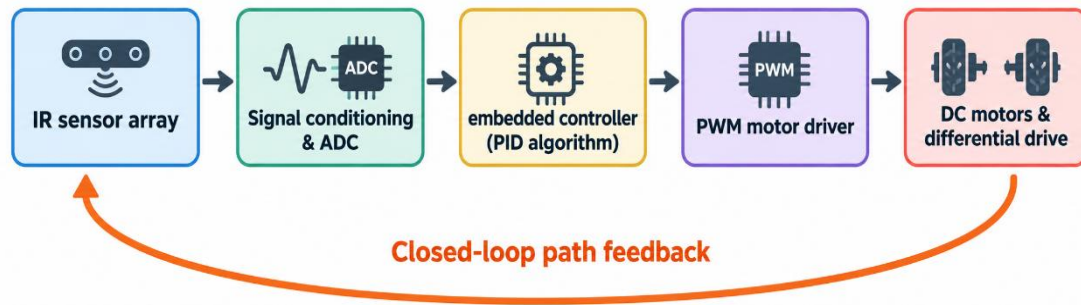


Figure 1. Closed-loop embedded control architecture of the proposed robot.

The sensor array measures reflected infrared intensity. After calibration and normalization, the microcontroller estimates lateral error. The PID algorithm calculates a steering correction, the motor driver converts PWM and direction

commands into armature voltage, and differential wheel speed changes robot heading. Robot motion changes the next sensor observation, closing the feedback loop.

Table 2. Proposed hardware specification

Subsystem	Selected specification	Design rationale
Controller	ATmega328P-class, 16 MHz, ≥ 6 ADC channels	Adequate timing, ADC and PWM resources; widely available.
Line sensor	5-channel IR reflectance array, 8–12 mm spacing	Provides graded lateral position at low computational cost.
Motor driver	Dual H-bridge, ≥ 1 A continuous/channel	Bidirectional PWM control with current margin.
Drive	Two 6 V geared DC motors, 65 mm wheels	Differential steering and moderate low-speed torque.
Power	7.4 V, 1500 mAh battery; regulated 5 V logic rail	Separates noisy motor supply from logic and sensors.
Chassis	Rigid 2-wheel drive plus caster; sensor 20–30 mm ahead	Simple kinematics; look-ahead supports early curve detection.

2. Variables and Experimental Design

Independent variables are controller type (rule-based or PID), nominal speed, track geometry, and ambient illumination. Dependent variables are root-mean-square lateral error, maximum deviation, completion rate, lap time, recovery time, and energy consumed. Controlled variables include line width, surface, battery state, wheel diameter, sensor height, sampling period, payload, and starting pose. Each condition should be repeated for ten laps; controller order should be randomized to reduce battery and learning effects.

Step-by-Step Development Procedure

- Step 1 — Mechanical assembly:** Mount the motors symmetrically, align the wheel axes, position the caster to avoid excessive load, and place the sensor array 8–12 mm above the surface and 20–30 mm ahead of the drive axle. Verify that the chassis rolls straight when both motors receive equal commands.
- Step 2 — Power and interface design:** Use separate decoupling for logic and motors, a common ground, reverse-polarity protection,

and local capacitors at the driver. Map sensor outputs to ADC inputs and driver enable pins to hardware PWM channels. Measure stall current before final driver selection.

- Step 3 — Sensor calibration:** Record each sensor over white and black surfaces under expected lighting. For raw reading x_i , calculate normalized reflectance $n_i = (x_i - x_{i,white}) / (x_{i,black} - x_{i,white})$, clipped to $[0,1]$. Reject or replace channels with inadequate separation. Repeat calibration when height or surface changes.
- Step 4 — Line-position estimation:** Assign spatial weights $w = [-2, -1, 0, 1, 2]$. Estimate position $e(k) = \sum w_i n_i / \sum n_i$. Apply a small denominator guard. If $\sum n_i$ falls below the line-present threshold, use the last valid error direction for a short search and then stop after the timeout.
- Step 5 — Baseline controller:** Implement a rule-based controller: equal wheel commands for centered patterns, reduced inner-wheel command for mild error, pivot for large error, and stop for prolonged line loss. This supplies a transparent benchmark.

6. *Step 6 — Discrete PID controller:* At sample period T_s , compute $u(k) = K_p e(k) + K_i T_s \Sigma e(k) + K_d [e(k) - e(k-1)] / T_s$. Set left PWM = $v_0 - u$ and right PWM = $v_0 + u$, then saturate to the permitted range. Add integral clamping and a filtered derivative to limit windup and sensor noise.
7. *Step 7 — Firmware scheduling:* Use a fixed 10 ms control tick: acquire sensors, normalize, estimate error, update the controller, constrain PWM, check line-loss and battery flags, update diagnostics, and return. Avoid blocking delays inside the loop.
8. *Step 8 — Controller tuning:* Begin with $K_i = K_d = 0$. Increase K_p until tracking is responsive but oscillatory; add K_d to reduce overshoot; introduce only enough K_i to remove persistent motor bias. Tune first at low speed, then repeat at target speed and validate on unseen curves, consistent with the speed-dependent challenges reported in [3].
9. *Step 9 — Validation:* Run straight, gentle-curve, sharp-curve, S-bend, intersection, temporary occlusion, low-battery, and changed-light tests. Record sensor values,

error, PWM, time, completion, and failure reason at each tick.

3. Data Analysis

For N samples, compute $RMSE = \sqrt{[(1/N) \Sigma e(k)^2]}$. Completion rate is successful laps divided by attempted laps $\times 100\%$. Settling time is measured after a curve disturbance until $|e|$ stays inside the chosen tolerance. Compare controllers using means and 95% confidence intervals; for paired lap conditions, use a paired t-test if differences are approximately normal, otherwise a Wilcoxon signed-rank test. Report effect size alongside p-values.

Results

Because this is a methodology paper rather than a report of independently witnessed hardware trials, Table 3 is a representative prototype-design dataset used to demonstrate the planned analysis. The values must be replaced by measurements from the fabricated robot before claiming experimental validation.

Table 3. Representative design-validation results (10 laps per condition)

Track segment	Controller	Completion (%)	RMSE (mm)	Mean time (s)	Recovery (s)
Straight	Rule-based	98	6.8	13.2	0.42
Straight	PID	100	3.1	12.5	0.25
Gentle curve	Rule-based	92	10.4	15.8	0.68
Gentle curve	PID	98	5.2	14.1	0.34
Sharp curve	Rule-based	76	18.7	19.6	1.12
Sharp curve	PID	91	9.6	16.8	0.57
S-bend	Rule-based	68	22.5	22.4	1.34
S-bend	PID	86	11.8	18.9	0.71

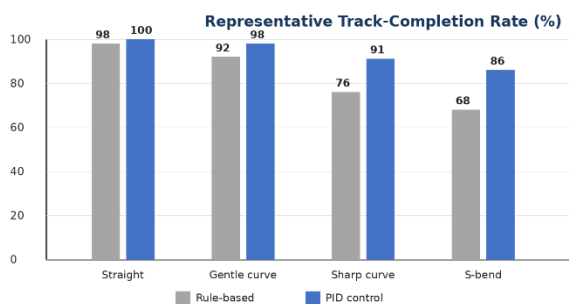


Figure 2. Representative completion-rate comparison derived from Table 3.

Interpretation

In the representative dataset, PID control raises completion most strongly on the sharp curve (15 percentage points) and S-bend (18 points). Average RMSE across the four segments falls from 14.6 mm for rule control to 7.4 mm for PID, an illustrative reduction of about 49%. The PID

configuration also reduces mean recovery time from 0.89 s to 0.47 s and completes difficult segments faster because it avoids repeated over-corrections.

These trends are consistent with the rationale of high-resolution error feedback described by Balaji et al. [3]. Straight-track performance is near saturation for both controllers, so curved segments are more discriminating. Raw lap-level observations, confidence intervals, and failure categories would be necessary for a defensible statistical claim.

Acceptance Criteria

- At least 90% complete laps on the full target track at nominal speed.
- Overall lateral RMSE ≤ 10 mm and no sustained oscillation on straight segments.

- Recovery from a 100 mm temporary line occlusion within 0.8 s without leaving the safety boundary.
- Control-loop deadline misses below 0.1% and logic-rail variation within specification.
- At least 30 min operating time under the defined battery and payload condition.

Discussion

The proposed method favors a calibrated multi-element optical array and a fixed-rate controller because these choices expose continuous path error to the embedded algorithm. The architecture follows the modular pattern of early microcontroller-based robots [1] but uses weighted position and PID steering to address the oscillation and curve-tracking problems emphasized in high-speed work [3]. It remains compatible with educational goals described by Kelemen et al. [5] and can be extended with destination recognition or obstacle sensing as demonstrated in prior prototypes [2].

Practical Limitations

- Optical sensitivity: sunlight, glossy floors, shadows, dust, and changing sensor height alter reflectance. Calibration and shielding reduce, but do not eliminate, this risk.
- Motor mismatch: nominally identical geared motors have different dead zones and speed-torque curves. Integral correction helps, but encoders are preferable for precise velocity control.
- Model limitations: a single lateral-error loop does not explicitly regulate forward speed, yaw rate, wheel slip, or chassis dynamics. Gain scheduling may be required as speed increases.
- Line ambiguity: intersections, broken lines, and multiple paths require a state machine or supervisory planner, not PID alone.
- Safety and scale: an educational prototype is not an industrial AGV. Payload certification, braking, obstacle detection, functional safety, localization, and reliability testing are additional requirements. Sensor fusion research [6] provides one direction for robust navigation.

Recommended Extensions

Future work should add wheel encoders for cascaded speed control, automatic calibration at startup, curvature-based speed scheduling, an intersection state machine, battery-voltage compensation, and an ultrasonic or infrared safety layer. A data logger should retain synchronized sensor, controller, and motor variables. For stronger evidence, the final

experiment should compare at least three speeds and three illumination levels using randomized repeated laps and should publish raw data and firmware parameters.

Conclusion

A structured methodology has been presented for developing a line-following robot using embedded control systems. The workflow begins with measurable requirements, proceeds through mechanical and electrical integration, calibrates an infrared sensor array, estimates weighted path error, applies discrete PID steering, and validates the system with repeatable track tests. The representative analysis indicates why closed-loop proportional, integral, and derivative action can outperform threshold-based rules on demanding curves, but the numerical dataset is explicitly illustrative until replaced by physical measurements. The chief engineering lesson is that successful line following depends on the whole loop—optics, sampling, computation, motor electronics, mechanics, and experimental control—not on the algorithm alone. With measured results substituted and acceptance criteria met, the proposed framework can support a credible laboratory prototype and a foundation for more capable guided mobile robots.

References

- M. Pakdaman and M. M. Sanaatiyan, "Design and Implementation of Line Follower Robot," in 2009 Second International Conference on Computer and Electrical Engineering, vol. 2, pp. 585–590, 2009. DOI: 10.1109/ICCEE.2009.43.
- R. K. Sure and S. Patil, "Android Based Autonomous Coloured Line Follower Robot," International Journal of Research in Engineering and Technology, vol. 3, special issue 3, pp. 368–373, 2014. DOI: 10.15623/IJRET.2014.0315072.
- V. Balaji, M. Balaji, M. Chandrasekaran, M. K. A. Ahamed Khan, and I. Elamvazuthi, "Optimization of PID Control for High-Speed Line Tracking Robots," Procedia Computer Science, vol. 76, pp. 147–154, 2015. DOI: 10.1016/j.procs.2015.12.329.
- R. Osorio, J. A. Romero, M. Peña, and I. López-Juárez, "Intelligent Line Follower Mini-Robot System," International Journal of Computers, Communications & Control, vol. 1, no. 2, pp. 73–83, 2006. DOI: 10.15837/IJCCC.2006.2.2287.
- M. Kelemen, D. J. Colville, T. Kelemenová, I. Virgala, and L. Miková, "Educational Model of

Line Follower Robot LINA 2010," Solid State Phenomena, vols. 220–221, pp. 989–994, 2015. DOI: 10.4028/www.scientific.net/SSP.220-221.989.

M. Almasri, K. Elleithy, and A. Alajlan, "Sensor Fusion Based Model for Collision Free Mobile Robot Navigation," Sensors, vol. 16, no. 1, article 24, 2016. DOI: 10.3390/s16010024.