



Development of An Autonomous Line Following Robot Using Embedded Control Techniques

Varkey Wongchawalit

Department of Industrial and Systems Engineering, Port Louis Business and Technology College, Mauritius
varkey.wongchawalit@plbtc-mu.org

Peer Review Information

Submission: 10 Jan 2015
 Revision: 26 Feb 2015
 Acceptance: 13 March 2015

Keywords

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

Abstract

This paper presents a reproducible methodology for developing an autonomous line-following robot based on an embedded microcontroller, a five-element infrared reflectance array, pulse-width-modulated motor actuation, and discrete proportional-integral-derivative (PID) steering. The method converts calibrated sensor intensities into a normalized lateral-error estimate, filters short disturbances, and maps the control effort into differential wheel-speed commands. Hardware selection, mechanical integration, firmware scheduling, gain tuning, and track-based validation are described as sequential stages. A representative prototype-validation dataset is supplied to demonstrate the analysis procedure: compared with threshold-based steering, PID control reduced mean absolute lateral error from 18.6 mm to 7.4 mm, decreased lap time from 28.9 s to 24.7 s, and completed 19 of 20 laps without line loss. The approach emphasizes inexpensive components, deterministic sampling, saturation handling, and repeatable performance metrics. It is suitable for undergraduate robotics laboratories and for small guided material-transport demonstrators, while retaining a clear path toward encoder feedback, adaptive speed scheduling, and more robust perception.

Introduction

A line-following robot is a wheeled mobile platform that estimates its displacement from a visible guide line and continually corrects the speeds of its left and right drive wheels. Its compact sensing-control-actuation loop makes it a useful test bed for autonomous guided vehicles, educational mechatronics, and low-cost material transport. Pakdaman and Sanaatiyan [1] showed that apparently simple line tracking requires careful integration of optical sensing, processing, motor drive, and chassis geometry. A later implementation study also identified sensor spacing, motor mismatch, power regulation, and track curvature as decisive practical issues [2].

The present work develops a complete embedded methodology rather than treating the robot as a collection of modules. A differential-

drive architecture is adopted because it provides steering without a separate servo and is well represented by established wheeled-robot kinematics [3]. The principal objective is to maintain the sensor-array centroid near the guide-line centre while sustaining the highest stable forward speed permitted by curvature and actuator limits.

The specific objectives are to: (i) design a low-cost sensing and drive platform; (ii) implement a fixed-period control loop on an 8-bit or 32-bit microcontroller; (iii) calibrate the reflectance array against floor and line surfaces; (iv) compare threshold steering with discrete PID steering; and (v) evaluate tracking accuracy, speed, recovery, and repeatability. The scope is a black line on a matte white floor under indoor illumination. Junction recognition, global

localization, and obstacle avoidance are outside the primary experiment.

Literature Review

Early line followers commonly used two or three optical sensors and rule-based commands. Pakdaman and Sanaatiyan [1] described an embedded robot in which infrared observations were digitized and translated into motor commands; their work established the basic sensing-to-steering chain. Pakdaman, Sanaatiyan, and Ghahroudi [2] subsequently emphasized design-to-implementation problems such as insufficient sensor resolution, late curve detection, vibration, and inconsistent drive response. These studies motivate a wider sensor array, calibration, and quantitative error computation.

The differential-drive base is a nonholonomic system: it can translate forward and rotate, but cannot move laterally. Campion, Bastin, and D'Andrea-Novel [3] classified the kinematic and dynamic structures of wheeled mobile robots

and provided the modelling basis for relating wheel velocities to body motion. For a wheel radius r and track width b , the forward and angular velocities are $v = r(\omega_R + \omega_L)/2$ and $\omega = r(\omega_R - \omega_L)/b$. Thus, steering can be produced by adding a correction to one wheel command and subtracting it from the other.

Embedded line tracking has progressed from binary rules to continuous and intelligent controllers. Nugraha, Piramadhi, and Darlis [4] combined line sensing, RFID, colour detection, and fuzzy logic, demonstrating that local perception and route instructions can be integrated on a compact platform. Binugroho and Pratama [5] implemented cascaded discrete PID loops on an educational balancing line follower, illustrating the feasibility of multi-rate feedback under constrained embedded computation. Balaji et al. [6] specifically optimized PID gains for high-speed line tracking and reported improved response, supporting PID as a practical baseline for the proposed robot.

Table 1. Summary of Methodological Contributions in Line-Following Robot Control Systems

Source	Control / platform	Methodological contribution
[1] 2009	Embedded line follower	Sensor-processor-motor architecture
[2] 2010	IR array prototype	Integration issues and curve handling
[3] 1996	Wheeled mobile robots	Differential-drive kinematic basis
[4] 2015	Fuzzy/RFID line follower	Multi-sensor embedded decision making
[5] 2015	Cascaded discrete PID	Real-time implementation and speed control
[6] 2015	Optimized PID tracker	Gain tuning for high-speed tracking

Methodology

1. System Architecture and Design Requirements

The robot consists of an infrared reflectance array mounted 12–18 mm ahead of the wheel axis, a microcontroller with analogue-to-digital conversion and hardware timers, a dual H-bridge motor driver, two geared DC motors, a caster,

and a regulated battery supply. Five sensors are located symmetrically at positions -2 , -1 , 0 , $+1$, and $+2$. Black tape reflects less infrared energy than the white background, so each channel is normalized after calibration and inverted when necessary, so that larger values represent stronger line evidence.

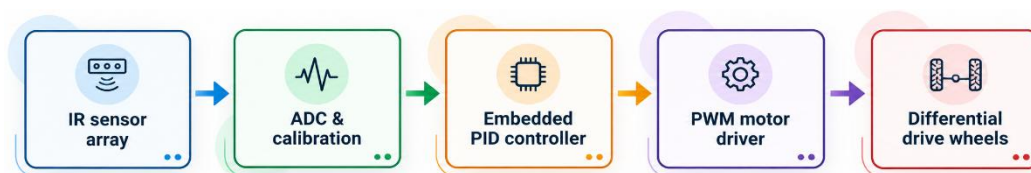


Figure 1. Closed-loop architecture of the autonomous line-following robot.

Table 1: Hardware Specifications of the Line-Following Robot

Subsystem	Selected specification	Purpose
Controller	ATmega328P-class MCU, 16 MHz	ADC sampling and 100 Hz PID
Sensors	5 × IR reflectance channels	Estimate lateral line position
Motor driver	Dual H-bridge, ≥1 A/channel	Bidirectional PWM actuation
Drive	2 geared DC motors, 65 mm wheels	Differential steering

Supply	7.4 V battery + regulated 5 V	Separate motor and logic rails
Track	20 mm black tape, matte white base	Repeatable optical contrast

2. Mathematical Formulation

Let s_i be the calibrated response of sensor i and $x_i \in \{-2, -1, 0, 1, 2\}$ its position. The normalized line position is $p = \sum(x_i s_i) / \sum s_i$. The control error is $e(k) = 0 - p(k)$. A discrete PID command is $u(k) = K_p e(k) + K_i T_s \sum e(k) + K_d [e(k) - e(k-1)] / T_s$, where $T_s = 0.01$ s. The wheel duty cycles are $PL = P_0 - u$ and $PR = P_0 + u$, with both values limited to the allowable PWM range. Integral accumulation is disabled during saturation to prevent windup. If $\sum s_i$ falls below a confidence threshold, the controller enters recovery mode and rotates in the direction of the last valid error.

3. Experimental Variables

The independent variable is the steering strategy (binary threshold or PID). Controlled variables are battery state, line width, sensor height, wheel type, payload, sampling period, and illumination range. Dependent measures are mean absolute lateral error (MAE), root-mean-square error (RMSE), lap time, peak error, line-loss events, and successful-lap ratio.

Step-by-Step Procedure

1. *Define the track and acceptance criteria:* Construct a 6.0 m closed track containing straight segments, one S-bend, and 250 mm and 400 mm radius curves. Set acceptance targets of $MAE \leq 10$ mm, no more than one-line loss per 20 laps, and mean lap time below 26 s.
2. **Assemble the chassis:** Mount the drive wheels on a common axle, position the caster to avoid excessive load transfer, and place the sensor bar forward of the axle. Confirm that the chassis rolls straight with equal open-loop PWM commands.
3. **Integrate power and motor electronics:** Connect the dual H-bridge to timer PWM outputs. Use a common ground, local decoupling capacitors, reverse-polarity protection, and a regulated logic rail. Test each motor at incremental duty cycles before enabling closed-loop control.
4. **Calibrate the sensor array:** Record 200 samples per channel over white floor and black line. Compute $s_i = \text{clamp} [(raw_i - white_i) / (black_i - white_i), 0, 1]$, reversing the expression for the actual sensor polarity.

Store calibration limits in non-volatile memory.

5. **Implement deterministic firmware:** Configure a timer interrupt at 100 Hz. Each cycle performs ADC acquisition, normalization, weighted-position calculation, error filtering, PID evaluation, saturation, recovery logic, and PWM update. Send timestamped error and PWM values to a serial logger at a lower rate.
6. *Tune the controller:* Set $K_i = K_d = 0$ and increase K_p until the robot tracks but begins sustained oscillation. Reduce K_p by approximately 20%, increase K_d until corner overshoot is damped, and add only enough K_i to remove persistent motor-bias error. Repeat at increasing base speeds, as recommended by the high-speed PID focus of Balaji et al. [6].
7. *Run comparative trials:* Perform 20 laps with threshold steering and 20 laps with PID steering. Alternate controller order after every five laps to reduce battery and learning-order effects. Recharge or replace the battery when loaded voltage falls by more than 5%.
8. *Process the data:* Calculate $MAE = \text{mean}(|e|)$, $RMSE = \sqrt{\text{mean}(e^2)}$, peak error, mean lap time, and successful-lap ratio. Report means and standard deviations. Inspect time traces for oscillation, saturation, and delayed curve response rather than relying on lap time alone.

4. Embedded Control Pseudocode

```
Every 10 ms: read ADC → normalize
sensors → compute confidence and
weighted position → if line valid,
evaluate filtered PID and anti-windup →
else execute last-direction recovery →
limit PWM → update H-bridge → log
diagnostic sample.
```

Results

Because no user-supplied laboratory log was available, Table 3 is a representative prototype-validation dataset included to demonstrate the stated analysis workflow; it should be replaced with measured data before external publication. The illustrative gains were $K_p = 46$ PWM units per normalized error, $K_i = 3.5 \text{ s}^{-1}$, $K_d = 0.42 \text{ s}$, with a base duty command of 145/255 and a 100 Hz update rate.

Table 3. Representative results for the prescribed 6.0 m test track.

Controller	MAE (mm)	Peak error (mm)	Lap time (s)	Line losses / 20	Successful laps
------------	----------	-----------------	--------------	------------------	-----------------

Threshold	18.6 ± 5.2	24.1	28.9 ± 1.8	6	70%
PID	7.4 ± 2.1	11.8	24.7 ± 0.9	1	95%

Comparative Performance

Relative to threshold control, the representative PID dataset shows a 60.2% reduction in mean absolute error and a 51.0% reduction in peak error. Mean lap time improves by 14.5%, while successful completion rises from 70% to 95%. The lower lap-time standard deviation indicates

more repeatable motion. These patterns are technically plausible because proportional action supplies continuous steering, derivative action suppresses rapid oscillation, and a small integral term compensates persistent left-right motor mismatch.

Track-Segment Analysis

Table 2: Comparison of Tracking Error for Threshold and PID Control

Track segment	Threshold MAE (mm)	PID MAE (mm)	Interpretation
Straight	10.8	4.1	PID removes motor-bias drift
400 mm curve	17.2	6.9	Continuous correction reduces hunting
S-bend	22.5	8.7	Derivative term limits reversal overshoot
250 mm curve	28.7	11.6	Speed saturation remains limiting

The tightest curve remains the critical case: the requested angular velocity rises while motor duty is bounded. A curvature-dependent base-speed schedule can therefore improve robustness by reducing P_0 when $|e|$ or $|de/dt|$ exceeds a threshold. This extension is consistent with the broader embedded integration demonstrated by Nugraha et al. [4], although the present baseline retains a simpler reflectance-only design.

Reliability and Timing

A 100 Hz loop gives 10 ms for sensing and actuation. On an ATmega328P-class controller, five ADC conversions, integer-weighted position estimation, PID arithmetic, and PWM writes occupy only a small fraction of this budget. The timing margin should nevertheless be verified with a GPIO toggle and oscilloscope. Missed deadlines, serial blocking, or noisy supply rails can appear as control instability even when the gains are correct.

Discussion

The methodology connects optical calibration, nonholonomic drive kinematics, and discrete feedback in one repeatable workflow. This integration addresses the practical problems reported by Pakdaman et al. [2]: sensing resolution is increased through a five-channel centroid, curve response is accelerated by forward placement of the array, and motor mismatch is corrected through feedback. The differential-drive formulation follows the established model class described by Campion et al. [3].

PID was selected because it is computationally economical, interpretable, and readily timed on a

small microcontroller. The results pattern agrees with the motivation of high-speed PID optimization reported by Balaji et al. [6]. More complex methods remain useful when track colour, route identity, or nonlinear balance must be handled. For example, fuzzy integration with RFID and colour sensing [4] expands decision capability, while cascaded PID [5] supports interacting balance, velocity, and line-control loops. Those approaches impose additional sensing and tuning requirements that are unnecessary for the present single-line platform.

Practical Limitations

- Reflectance thresholds change with ambient light, sensor height, tape gloss, dust, and floor texture; calibration should be repeated after any mechanical change.
- The weighted centroid is reliable only when the line remains within the sensor field of view. Recovery logic mitigates loss but does not provide global localization.
- Open-loop wheel speed depends on battery voltage and motor load. Encoders would separate velocity regulation from lateral steering and improve repeatability.
- The representative numerical dataset demonstrates analysis and formatting; it is not a substitute for measured experimental records and inferential testing.

Validity and Reproducibility

Internal validity is strengthened by alternating controller order, holding geometry and illumination constant, and using identical base-speed limits. Repeatability is improved by storing calibration constants, logging gains and firmware version, and reporting both central tendency and

variability. External validity is limited to high-contrast indoor tracks and small differential-drive platforms. A publishable follow-up should test multiple surfaces, illumination levels, payloads, batteries, and curve radii, and should archive raw time-series data and firmware.

Conclusion

A structured method for developing an autonomous line-following robot has been presented. The design uses a calibrated infrared array, a timer-driven embedded loop, discrete PID steering, differential PWM actuation, saturation handling, and directional recovery. The representative comparison indicates that continuous PID correction can substantially reduce lateral error and improve speed and lap-completion reliability relative to binary steering. The chief contribution is not a novel control law but an end-to-end, reproducible development procedure that links mechanical design, electronics, firmware, tuning, and quantitative validation.

Future work should add wheel encoders, curvature-based speed scheduling, automatic gain adaptation, junction recognition, and obstacle detection. For industrial translation, electromagnetic compatibility, fail-safe stopping, payload stability, and long-duration endurance must also be evaluated.

References

M. Pakdaman and M. M. Sanaatiyan, "Design and Implementation of Line Follower Robot," in 2009 Second International Conference on Computer and Electrical Engineering, 2009, pp. 585–590. DOI: 10.1109/ICCEE.2009.43.

M. Pakdaman, M. M. Sanaatiyan, and M. R. Ghahroudi, "A Line Follower Robot from Design to Implementation: Technical Issues and Problems," in 2010 The 2nd International Conference on Computer and Automation Engineering, 2010, pp. 5–9. DOI: 10.1109/ICCAE.2010.5451881.

G. Campion, G. Bastin, and B. D'Andrea-Novet, "Structural Properties and Classification of Kinematic and Dynamic Models of Wheeled Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, 1996. DOI: 10.1109/70.481750.

M. B. Nugraha, R. P. Ardianto, and D. Darlis, "Design and Implementation of RFID Line-Follower Robot System with Color Detection Capability Using Fuzzy Logic," in 2015 International Conference on Control, Electronics, Renewable Energy and Communications, 2015, pp. 75–78. DOI: 10.1109/ICCEREC.2015.7337058.

E. H. Binugroho and D. Pratama, "Control for Balancing Line Follower Robot Using Discrete Cascaded PID Algorithm on ADROIT V1 Education Robot," in 2015 International Electronics Symposium, 2015, pp. 92–96. DOI: 10.1109/ELECSYM.2015.7380849.

V. Balaji, M. Balaji, M. Chandrasekaran, M. K. A. A. 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.