Line Following Robot

Finite State Machine Control with Digital Sensors

Project Report

Mohammed VI Polytechnic University (UM6P)
College of Computing

Project Members

Yahya Mansoub
El Hani Mohammed Reda
Youness Anouar

November 26, 2025

Abstract

This report presents the design and implementation of a discrete, finite-state controller for a differential-drive line following robot using three digital infrared (IR) sensors. The control strategy is intentionally simple and robust: instead of using continuous PID control, the robot executes clear behavioral states—*forward*, *turn left*, *turn right*, and *search*—according to the digital sensor pattern and a minimal memory of whether the line has been seen previously.

The main contribution is a clean state-machine formulation that guarantees: (i) stable forward motion when the center sensor is on the line, (ii) consistent recovery when the line is detected by a side sensor only, and (iii) a recovery strategy when all sensors temporarily lose the line. The report formalizes the state machine, derives its transition function, and illustrates the behavior with a graphical state diagram.

1 Introduction

Line following robots are a classic platform for experimenting with embedded control, digital sensing and real-time decision logic. A common approach is to employ an analog-based PID controller to minimize lateral error relative to a line. In this project, the objective is different: design a *discrete*, human-readable control logic based on a **finite state machine** (FSM), driven only by three digital sensors.

The hardware setup is:

- A differential-drive robot with two DC motors and an H-bridge driver.
- Three IR reflective sensors (left, middle, right) with on-board comparators providing a digital output and a green status LED.
- An Arduino-compatible microcontroller generating PWM and reading digital inputs.

Each sensor outputs a logical level depending on the reflectance under it. In our configuration, we assume:

$$\text{GREEN LED ON} \iff \text{sensor output is 0 (LOW).} \tag{1}$$

The control problem is to keep the robot on a white (or grey) line with discrete, qualitative behaviors instead of continuous optimization.

2 Digital Sensor Model

Let the three sensor readings at time step k be:

$$L_k, M_k, R_k \in \{0, 1\},$$

corresponding to left, middle, and right sensors, respectively.

We define boolean variables indicating that each sensor “sees” the line (i.e. green LED on, output low):

$$\ell_k = \begin{cases} 1, & \text{if } L_k = 0, \\ 0, & \text{if } L_k = 1, \end{cases} \tag{2}$$

$$m_k = \begin{cases} 1, & \text{if } M_k = 0, \\ 0, & \text{if } M_k = 1, \end{cases} \tag{3}$$

$$r_k = \begin{cases} 1, & \text{if } R_k = 0, \\ 0, & \text{if } R_k = 1. \end{cases} \tag{4}$$

The controller also maintains a small integer memory:

$$c_k \in \mathbb{N},$$

representing how many times the middle sensor has seen the line (i.e. $m_k = 1$). This is used to distinguish an initial calibration phase from the "normal" tracking phase. In code, this is the variable `middleSeenCount`.

We define a simple condition indicating that the robot has already been properly aligned on the line at least a few times:

$$\text{RecoveryReady}_k := \begin{cases} 1, & c_k \geq C_{\min}, \\ 0, & c_k < C_{\min}, \end{cases} \tag{5}$$

where in the implementation $C_{\min} = 3$.

3 Finite State Machine Formulation

3.1 State Set and Outputs

We model the controller as a deterministic finite state machine:

$$\mathcal{M} = (Q, \Sigma, \delta, q_0, \lambda),$$

where:

- Q is the finite set of control states,
- Σ is the input alphabet (sensor patterns),
- δ is the state transition function,
- q_0 is the initial state,
- λ is the output function (motor commands).

The state set is chosen as:

$$Q = \{q_{\text{idle}}, q_{\text{forward}}, q_{\text{turnL}}, q_{\text{turnR}}, q_{\text{searchL}}\}. \tag{6}$$

Intuitively:

- q_{idle}: robot is stopped, waiting to see the line.
- q_{forward}: robot moves forward, line under middle sensor.
- q_{turnL}: robot spins left in place until middle sensor is on the line.
- q_{turnR}: robot spins right in place until middle sensor is on the line.
- q_{searchL}: robot slowly spins left searching for the line after a complete loss.

The input alphabet is the set of all sensor triplets:

$$\Sigma = \{0, 1\}^3 = \{(L, M, R) \mid L, M, R \in \{0, 1\}\}.$$

The output $\lambda(q)$ maps each state to a pair of signed wheel velocities (v_L, v_R):

$$\lambda : Q \to \mathbb{R}^2.$$

In implementation, these velocities are realized via PWM values in `driveSigned(leftPWM, rightPWM)`.

For example:

$$\lambda(q_{\text{forward}}) = (v_{\text{fwd}}, v_{\text{fwd}}), \tag{7}$$
$$\lambda(q_{\text{turnL}}) = (-v_{\text{spin}}, +v_{\text{spin}}), \tag{8}$$
$$\lambda(q_{\text{turnR}}) = (+v_{\text{spin}}, -v_{\text{spin}}), \tag{9}$$
$$\lambda(q_{\text{searchL}}) = (-v_{\text{search}}, +v_{\text{search}}), \tag{10}$$
$$\lambda(q_{\text{idle}}) = (0, 0), \tag{11}$$

with $v_{\text{fwd}} =$ `FWD_SPEED`, $v_{\text{spin}} =$ `SPIN_SPEED`, $v_{\text{search}} =$ `SEARCH_SPIN`.

3.2 Transition Function

We define the transition function

$$\delta : Q \times \Sigma \times \mathbb{N} \to Q,$$

since it also depends on c_k via `RecoveryReady`.

Let the shorthand

$$s_k = (\ell_k, m_k, r_k) \in \{0,1\}^3$$

denote the logical sensor pattern at time k.

The high-level logic implemented in code can be described as:

1. Middle sensor dominates. If the middle sensor sees the line, the robot must *always* go forward and exit any turning mode:

$$m_k = 1 \quad \Rightarrow \quad \delta(q, s_k, c_k) = q_{\text{forward}} \quad \forall q \in Q. \tag{12}$$

In this case, the memory is updated as:

$$c_{k+1} = \min(c_k + 1, C_{\max}),$$

with $C_{\max}$ a saturation bound (here 1000).

2. Persistent turning behavior. If the robot has already committed to a turn state, the turn persists until the middle sensor detects the line again. This prevents oscillatory behavior:

$$q_k = q_{\text{turnR}},\ m_k = 0 \Rightarrow q_{k+1} = q_{\text{turnR}}, \tag{13}$$

$$q_k = q_{\text{turnL}},\ m_k = 0 \Rightarrow q_{k+1} = q_{\text{turnL}}. \tag{14}$$

3. Turn initiation (right). If not currently turning, and only the right sensor sees the line, and the robot has seen the line with the middle sensor enough times:

$$q_k \notin \{q_{\text{turnL}}, q_{\text{turnR}}\}, \quad (s_k = (0,0,1)), \quad \text{RecoveryReady}_k = 1 \Rightarrow q_{k+1} = q_{\text{turnR}}. \tag{15}$$

4. Turn initiation (left). Similarly, for the left sensor:

$$q_k \notin \{q_{\text{turnL}}, q_{\text{turnR}}\}, \quad (s_k = (1,0,0)), \quad \text{RecoveryReady}_k = 1 \Rightarrow q_{k+1} = q_{\text{turnL}}. \tag{16}$$

5. Search mode. If all sensors see nothing, but the robot was already on the line before, the controller enters a slow left search:

$$s_k = (0,0,0), \quad \text{RecoveryReady}_k = 1, \quad q_k \notin \{q_{\text{turnL}}, q_{\text{turnR}}\} \Rightarrow q_{k+1} = q_{\text{searchL}}. \tag{17}$$

6. Idle / safe. Otherwise, the robot remains in or returns to an idle-like state (stop), waiting for the first proper detection:

$$\text{else} \Rightarrow q_{k+1} = q_{\text{idle}}. \tag{18}$$

3.3 State Transition Table

Table 1 summarizes the key transitions. Here, $m =$ mid sensor, $\ell =$ left, $r =$ right.

4 Graphical State Machine

Figure 1 presents a high-level state diagram for the robot controller. The diagram captures the dominant transitions used in practice.

Current State q_k	Sensor Pattern (ℓ, m, r)	RecoveryReady	Next State q_{k+1}
any	$m = 1$	any	q_{forward}
q_{turnR}	$m = 0$	any	q_{turnR}
q_{turnL}	$m = 0$	any	q_{turnL}
not turning	$(0, 0, 1)$	1	q_{turnR}
not turning	$(1, 0, 0)$	1	q_{turnL}
not turning	$(0, 0, 0)$	1	q_{searchL}
any	else	any	q_{idle}

Table 1: Simplified finite state transition table for the controller.

5 Discussion

The proposed controller deliberately avoids continuous PID control and instead relies on *qualitative* state-based behavior:

- The **forward** state is activated whenever the middle sensor sees the line. This gives a strong, stable condition for normal motion.
- When the line drifts under a side sensor, the robot enters a **pure spin** correction state in the corresponding direction and remains there until the middle sensor is aligned again. This prevents the "ping–pong" effect where the robot alternates between left and right corrections without ever locking onto the center.
- The **search** state provides a simple recovery when the line is completely lost but has been seen before (as indicated by the counter c_k). The robot picks a direction (here, left) and slowly scans until any sensor is reactivated.

Mathematically, the controller is a deterministic automaton with memory c_k that shifts its behavior once sufficient confidence about prior line contact has been gathered. This small amount of memory is enough to distinguish between initial calibration (robot has never seen the line) and genuine line loss after successful tracking.

6 Conclusion

This project demonstrates that a line following robot can be controlled effectively using only three digital sensors and a finite state machine, without resorting to analog PID control loops. By explicitly modeling the behavior as a state machine, the logic becomes easy to reason about, debug, and extend.

Future work may include:

- adding speed modes (slow/fast) as additional states,
- integrating intersection-handling states (e.g. T-junctions, crossroads),
- combining this discrete FSM with a higher-level planner.

The presented formulation and diagram can serve as a clean template for students and practitioners designing robust and understandable robot controllers.

Project Members:
Yahya Mansoub, El Hani Mohammed Reda, Youness Anouar

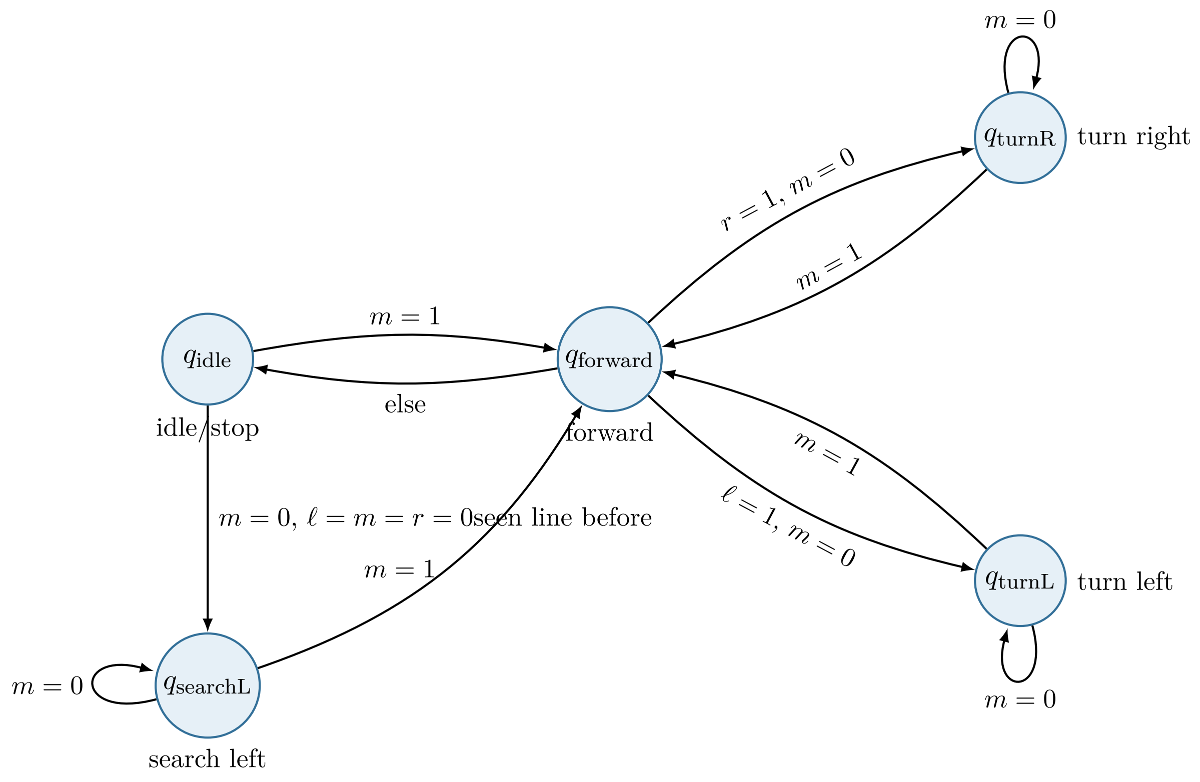

Figure 1: High-level state machine for the line following controller.