Autonomous Driving Duckiebot

GROUP V

SUBMITTED BY

Xinran Shen, Ruilai Ma, Weiyuan Du, Jing Cheng,
Yijie Gao, Yingtong Tian, Huixue Chen

UNDER THE GUIDANCE OF

Prof. Dr. Amr Alanwar

August 2025

1 Abstract

This project presents the development of an autonomous driving system on the Duckiebot platform, integrating motion control and perception-based navigation to emulate core self-driving behaviors. The work begins with open-loop trajectory execution and extends to a feedback-based state machine with odometry for precise square driving. A vision-based lane keeping module employs adaptive color segmentation and PID control to maintain alignment within the lane, while obstacle detection ensures safety by identifying hazards and triggering slow-down or avoidance maneuvers. Road sign and traffic light recognition provide semantic understanding, enabling the Duckiebot to comply with stop signs, directional instructions, and traffic signals. These perception modules are fused into a decision-making layer that governs maneuver selection, while a control layer translates high-level plans into wheel velocity commands. The entire framework was developed within the Gym-Duckietown simulator, ensuring modularity, real-time communication, and extensibility. This project delivers hands-on experience in computer vision, control theory, and robotics middleware, forming a solid foundation for future deployment on advanced autonomous vehicle research.

2 Problem Statement

During the development of autonomous driving functions on the Duckiebot platform, several challenges were encountered. Lane markings were often only partially visible or distorted under varying conditions, making stable lane keeping difficult. Obstacles required reliable detection to ensure safety without triggering unnecessary maneuvers. Traffic signs and lights, being small and visually similar to background elements, demanded accurate recognition to support correct decision-making. Moreover, all modules needed to operate in real time within a modular framework to allow smooth integration of perception, control, and decision layers. Addressing these issues required robust computer vision pipelines, carefully tuned PID controllers, and an integrated framework that allows real-time coordination of perception, control, and decision-making modules.

3 Introduction

This project implements fundamental autonomous driving functions on the Duckiebot DB21J platform, with development and validation primarily conducted in the Gym-Duckietown simulator.

The implemented functions include:

- Open-loop and feedback-based PID control for motion execution
- Lane keeping to maintain stable trajectory within marked lanes
- Obstacle detection and avoidance to ensure safe navigation
- Traffic light recognition and decision-making for compliance with signals

Together, these modules form a cohesive perception–control pipeline that mimics core self-driving behaviors. They also provide a solid foundation for future extensions, such as robust road sign detection.

4 Hardware Setup

- **Platform**: Duckiebot DB21J
- **Processing Unit**: Jetson Nano
- **Sensors**:
  - Camera (front-facing, for lane and traffic light detection)
  - Wheel encoders (for velocity estimation)
  - IMU (for orientation feedback, optional)
  - 8MP camera (primary visual input)
  - Time-of-Flight sensor

5 Software Architecture

The software architecture of our Duckiebot system is organized into three main layers. These layers build upon each other, starting from low-level motor control, to mid-level perception and navigation, and finally to high-level semantic decision-making. This modular design ensures clarity, extensibility, and alignment with the principles of autonomous driving systems.

5.1 Open Loop Control

In open loop control, constant wheel velocities were sent to the motors without using sensor feedback. This allowed the Duckiebot to move in straight lines and perform in-place rotations. While simple, open loop control exhibited significant drift over longer distances due to wheel slip, calibration errors, and lack of feedback.

5.2 PID Feedback Control

To address the limitations of open loop motion, we implemented a PID controller. The controller used sensor feedback to minimize error between desired and actual velocity or heading. We compared the effects of pure P, PI, and full PID configurations, and found that:

- P control alone responded quickly but suffered from steady-state error.
- PI control eliminated steady-state error but reacted more slowly.
- PID control offered the best tradeoff, providing both accuracy and stability.

5.3 LED State Transitions

As an additional experiment, we implemented a simple LED state machine to demonstrate event-driven actuation. Based on detected input signals (e.g., simulated color triggers), the Duckiebot cycled through Green, Yellow, and Blue LED states, starting from an OFF state. This illustrated how even simple state-based logic can be integrated into the Duckiebot framework.

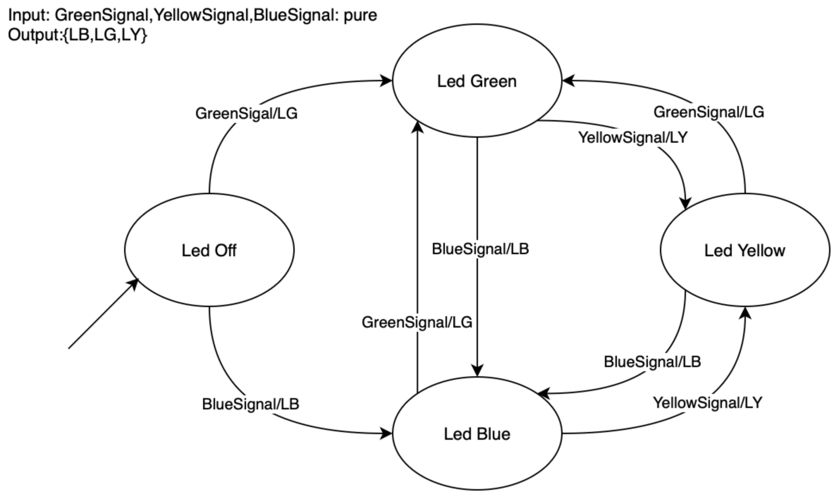

Figure 1: LED state machine implemented in Lab1, showing transitions between OFF, Green, Yellow, and Blue states.

5.4 Lane Keeping

Lane boundaries were extracted from the onboard camera image using computer vision techniques:

- **Preprocessing:** Images were enhanced using CLAHE (Contrast-Limited Adaptive Histogram Equalization) and converted to HSV and LAB color spaces.
- **Yellow line detection:** Adaptive HSV filtering robust to illumination changes.
- **White line detection:** Combined HSV and LAB thresholds restricted to the right side of the image, with morphological cleanup. A histogram-based fallback was used when no strong white lines were visible.

From these masks, lane centers were estimated using horizontal slicing and Hough line detection, enabling computation of lane position and curvature.

5.5 Control Logic with State Machine

To improve robustness, a hybrid automaton was implemented that switches between discrete operational modes depending on lane visibility:

- **Normal Lane Following:** Both lane lines visible, PID controller active.
- **Turn Mode (Left/Right):** Triggered when one line disappears for several consecutive frames. The vehicle biases toward the visible line, applies a steer bias, reduces speed, and ramps steering effort.
- **Recovery/Search Mode:** Activated when both lines are missing for too long. The vehicle first steers toward the last known side, then performs sweeping searches at low speed until a line is reacquired.

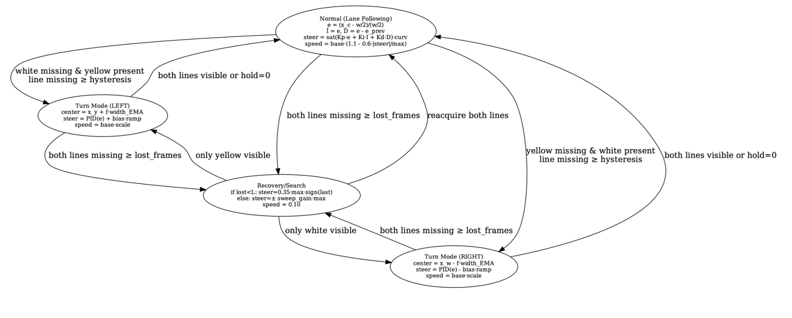
Normal (Lane Following)
e = (x_c - w/2)/(w/2)
I = e, D = e - e_prev
steer = sat(Kp·e + Ki·I + Kd·D)·curv
speed = base·(1.1 - 0.6·|steer|/max)
white missing & yellow present
line missing ≥ hysteresis
both lines visible or hold=0
Turn Mode (LEFT)
center = x_y + f·width_EMA
steer = PID(e) + bias·ramp
speed = base·scale
both lines missing ≥ lost_frames
reacquire both lines
both lines missing ≥ lost_frames
only yellow visible
yellow missing & white present
line missing ≥ hysteresis
both lines visible or hold=0
Recovery/Search
if lost<L: steer=0.35·max·sign(last)
else: steer=± sweep_gain·max
speed ≈ 0.10
only white visible
both lines missing ≥ lost_frames
Turn Mode (RIGHT)
center = x_w - f·width_EMA
steer = PID(e) - bias·ramp
speed = base·scale

Figure 2: State machine for lane keeping with turn awareness, including Normal, Turn, and Recovery modes.

5.6 Obstacle Avoidance

5.7 Traffic Light Recognition

A vision-based pipeline was implemented to detect and interpret traffic lights in the simulated environment:

- **Region of Interest (ROI):** The upper portion of the camera image was selected, where traffic lights typically appear.
- **Color segmentation:** HSV thresholding was used to extract red and green regions.
- **Shape detection:** Hough Circle Transform verified the circular shape of traffic lights, filtering out background noise.
- **Decision logic:**
  - Red light $\rightarrow$ stop the vehicle (zero speed command).
  - Green light $\rightarrow$ resume normal lane following control.

6 Experimental Results

- **Open-loop and PID control:** With feedback-based PID tuning, the vehicle achieved more stable velocity and orientation control compared to pure open-loop commands.
- **LED color recognition and response:** The system recognized different input colors and changed the vehicle's onboard LED lights accordingly (e.g., yellow, green, blue).
- **Lane following:** Lane following works reliably on straight roads and smooth curves.
- **Turn awareness:** Turn awareness improves stability in intersections and sharp corners.
- **Traffic light recognition:** Traffic light recognition successfully stops at red and resumes at green.
- **Obstacle avoidance:** The Duckiebot was able to detect red objects on the road and either slow down or stop.

7 Challenges and Limitations

1. **Lighting Sensitivity** – Lane and traffic light detection are affected by shadows and reflections; simulation hides much of this challenge but real-world deployment would require stronger preprocessing.
2. **Turning Stability** – Turn-aware mode improves intersections, but sharp corners can still cause drift without further PID tuning.
3. **Traffic Light Reliability** – Works well in simulation, but long distances or bright backgrounds reduce accuracy.

8 Future Work

To extend this project, several directions can be pursued:

- **Robust Perception:** Incorporate deep learning-based object detection and semantic segmentation to improve lane, obstacle, and traffic light recognition under varying conditions.
- **Advanced Control:** Explore Model Predictive Control (MPC) or reinforcement learning approaches to achieve smoother and more adaptive driving compared to classical PID control.
- **System Integration:** Combine lane keeping, obstacle avoidance, and traffic sign handling into a unified decision-making framework capable of handling complex urban scenarios.
- **Multi-agent Scenarios:** Extend the system to support interaction between multiple Duckiebots, enabling experiments with traffic flow, intersections, and cooperative behaviors.