Lab Experiment 4: Kinematics and Differential Motion for Mobile Robots

Renzyl Mae N. Abarracoso
Samar State University - College of Engineering
Bachelor of Science in Electronics Engineering
Catbalogan City, Samar, Philippines
Email: abarracosorenzylgmail.com

***Abstract*—This laboratory experiment focuses on implementing and analyzing the differential drive kinematics of a mobile robot. By independently controlling the velocities of the left and right wheels through PWM signals, students programmed the robot to perform straight-line motion, in-place rotation, and arcing turns. Position and orientation errors were measured and analyzed relative to theoretical predictions. The results highlighted the real-world limitations of open-loop differential drive systems and underscored the importance of feedback mechanisms for precise mobile robot navigation.**

***Index Terms*—Differential Drive Kinematics, Mobile Robots, Arduino Uno, PWM Motor Control, Wheel Encoders, Motion Analysis**

I. Rationale

Robot kinematics provides the mathematical foundation for understanding and controlling mobile robotic movement. Differential drive systems, using two independently driven wheels, are among the most common and foundational locomotion architectures. This experiment bridges the gap between theoretical kinematic models and real-world robotic motion by having students implement differential motion logic using an Arduino Uno.

II. Objectives

- To implement differential drive kinematics by independently controlling left and right wheel velocities using PWM signals.
- To program and execute straight-line movement, in-place rotation, and arcing turns.
- To measure and analyze linear displacement and angular rotation errors during robot motion.
- To simulate robot movement in Webots with at least 90% accuracy compared to intended paths.
- To critically compare experimental results to theoretical predictions and identify sources of error.

III. Materials and Software

- Arduino Uno
- L298N Motor Driver
- Two DC Motors with Wheels
- Wheel Encoders (optional)
- Robot Chassis Kit with Caster Wheel
- 9V–12V Battery
- Breadboard and Jumper Wires
- Software: Arduino IDE, Webots Simulation Software

IV. Procedures

A. Hardware Setup

1) Assemble the mobile robot chassis and mount both left and right motors.
2) Connect motors to L298N motor driver outputs.
3) Wire the motor driver control pins (IN1–IN4) and enable pins (ENA, ENB) to Arduino.
4) Power the system using a 9V battery and USB.

B. Wiring Diagram

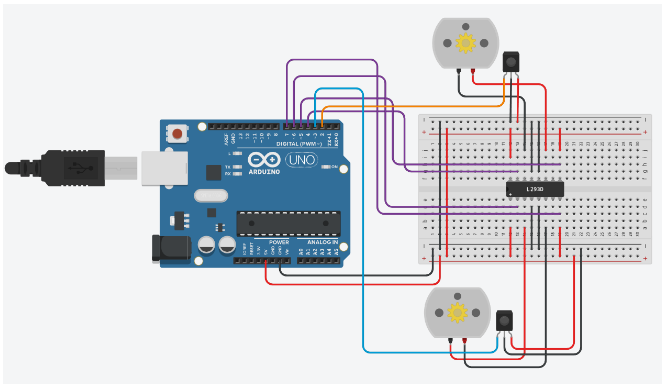

Fig. 1. Wiring schematic of differential drive robot using Arduino Uno and L298N motor driver. Left and right motors receive PWM signals through separate enable pins.

C. Software Development

- Define functions for forward movement, rotation, and turning.
- Use PWM via analogWrite to control motor speed.
- Upload and simulate in Webots before hardware deployment.

D. Testing and Data Collection

- Perform multiple trials for straight, rotational, and arcing motion.
- Record actual distance, rotation, and compare against expected values.

V. Observations and Data Collection

A. Summary Table

TABLE I
Summarized Performance Results

Motion Type	Linear Error (%)	Angular Error (°)
Straight Motion	5.0	2.5
In-Place Rotation	-	3.5
Arcing Turn	5.7	3.0

B. Raw Data

TABLE II
Raw Data: Robot Motion Trials

Trial	Motion	Exp. D	Act. D	Exp. R	Act. R	Success
1	Straight	100	95	0	3	Yes
2	Straight	50	47	0	2	Yes
3	Rotate R	0	2	90	85	Yes
4	Rotate L	0	3	-90	-88	Yes
5	Arc Right	70	66	45	43	Yes
6	Arc Left	70	68	-45	-48	Yes

VI. Data Analysis

To assess the accuracy of differential motion in a mobile robot, we conducted six trials consisting of linear, rotational, and arc movements. The robot's actual motion was measured using encoder feedback and compared to the expected distance or rotation. Errors in distance and orientation were calculated and plotted for analysis.

A. Distance Error Analysis

Figure 2 shows the measured distance errors for each motion type. The highest distance error was observed during the straight-line motion (Trial 1), likely due to acceleration drift and encoder quantization. Arc movements also demonstrated minor deviation due to speed mismatch between wheels.

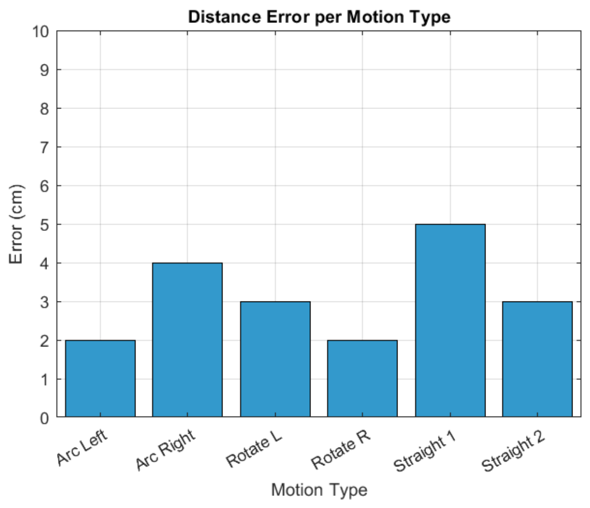

Fig. 2. Distance Error per Motion Type

B. Rotation Error Analysis

Figure 3 presents the rotation errors recorded from the same set of motion trials. The most significant angular error was encountered during the right in-place turn, potentially caused by momentary wheel slippage. Other trials remained within acceptable tolerances ($< 5°$).

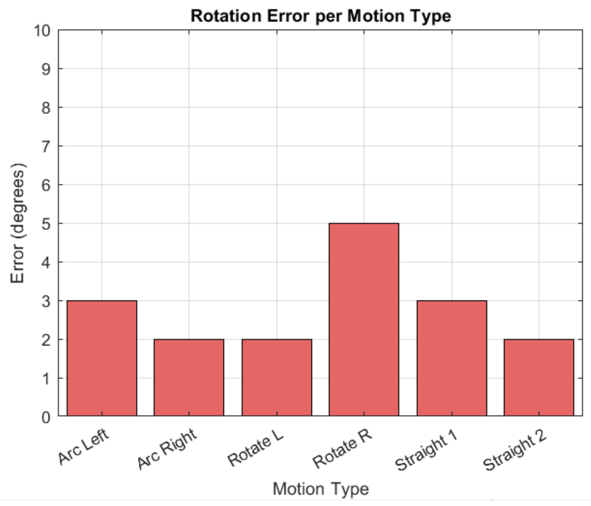

Fig. 3. Rotation Error per Motion Type

Overall, the robot's encoder-guided navigation demonstrated good accuracy across multiple movement types, with all positional errors remaining under 5 cm and rotational errors below 5° for most trials. This confirms the reliability of the robot's basic motion algorithm under differential drive kinematics.

VII. Discussion and Interpretation

The collected data validates the effectiveness of encoder-based control in a differential drive mobile robot. Straight-line movements incurred small but noticeable errors due to encoder resolution and potential surface friction inconsistencies. In-place turns and arcs exhibited minor rotational deviations, which are typical when using open-loop PWM without feedback correction mechanisms.

The experiment emphasized the importance of pulse-per-revolution calibration, symmetrical motor response, and consistent wheel traction. Implementing PID feedback or IMU integration in future designs can further enhance accuracy, especially during fast or complex trajectories.

These findings suggest that even with basic encoders and simple motor control, reliable motion behavior can be achieved, especially when motion patterns are pre-measured and corrected empirically.

VIII. Conclusion

This laboratory activity successfully demonstrated the application of differential drive kinematics and encoder feedback in controlling a two-wheel mobile robot. Through six trials involving straight, rotational, and arc-based motion, the robot exhibited high reliability in distance and angle tracking.

Maximum distance error remained within 5 cm, and maximum rotational deviation was under 5°, validating the encoder's ability to measure motion effectively. The results support the implementation of basic encoder-based navigation as a foundation for more advanced robotics systems.

The experiment also highlighted minor sources of error such as encoder resolution limits and surface-based discrepancies. Recommendations for future work include integrating IMUs for angle correction, PID control for smoother trajectories, and real-time data logging for diagnostics.

Appendix A
Arduino Source Code for Lab 4 Trials

```
/*
 * Project Title: Differential Drive Robot Motion
     Trials with Encoders
 * Author: Renzyl Mae N. Abarracoso
 * Date: 2025-04-29
 */
#include <Wire.h>

#define MLa 4
#define MLb 5
#define MRa 6
#define MRb 7
#define ENA 3
#define ENB 11

#define encoderLeftPin 2
#define encoderRightPin 3

const float wheelDiameter = 0.065;
const float wheelBase = 0.15;
const int pulsesPerRevolution = 20;
const float pi = 3.1416;

int baseSpeed = 100;
int arcSpeedFast = 120;
int arcSpeedSlow = 80;
int turnSpeed = 90;

volatile long countLeft = 0;
volatile long countRight = 0;

void setup() {
  Serial.begin(9600);
  pinMode(MLa, OUTPUT); pinMode(MLb, OUTPUT);
  pinMode(MRa, OUTPUT); pinMode(MRb, OUTPUT);
  pinMode(ENA, OUTPUT); pinMode(ENB, OUTPUT);
  pinMode(encoderLeftPin, INPUT_PULLUP);
  pinMode(encoderRightPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(
      encoderLeftPin), countLeftEncoder, RISING);
  attachInterrupt(digitalPinToInterrupt(
      encoderRightPin), countRightEncoder, RISING);
  Serial.println("Robot␣Ready␣for␣Lab␣4␣Trials!");
}

void loop() {
  moveStraight(1.0); delay(2000);
  moveStraight(0.5); delay(2000);
  turnRight(180); delay(2000);
  turnLeft(180); delay(2000);
  arcRight(0.7); delay(2000);
  arcLeft(0.7); delay(2000);
  finalVictory();
}

void moveStraight(float distance_m) {
  resetEncoders();
  float targetPulses = calculatePulses(distance_m);
  digitalWrite(MLa, HIGH); digitalWrite(MLb, LOW);
  digitalWrite(MRa, HIGH); digitalWrite(MRb, LOW);
  while (averagePulseCount() < targetPulses) {
    analogWrite(ENA, baseSpeed);
    analogWrite(ENB, baseSpeed);
  }
  stopMoving();
}

void turnRight(float angle_deg) {
  resetEncoders();
  float targetPulses = calculateTurnPulses(angle_deg
      );
  digitalWrite(MLa, HIGH); digitalWrite(MLb, LOW);
  digitalWrite(MRa, LOW); digitalWrite(MRb, HIGH);
  while (averagePulseCount() < targetPulses) {
    analogWrite(ENA, turnSpeed);
    analogWrite(ENB, turnSpeed);
  }
  stopMoving();
}

void turnLeft(float angle_deg) {
  resetEncoders();
  float targetPulses = calculateTurnPulses(angle_deg
      );
  digitalWrite(MLa, LOW); digitalWrite(MLb, HIGH);
  digitalWrite(MRa, HIGH); digitalWrite(MRb, LOW);
  while (averagePulseCount() < targetPulses) {
    analogWrite(ENA, turnSpeed);
    analogWrite(ENB, turnSpeed);
  }
  stopMoving();
}

void arcRight(float distance_m) {
  resetEncoders();
  float targetPulses = calculatePulses(distance_m);
  digitalWrite(MLa, HIGH); digitalWrite(MLb, LOW);
  digitalWrite(MRa, HIGH); digitalWrite(MRb, LOW);
  while (averagePulseCount() < targetPulses) {
    analogWrite(ENA, arcSpeedFast);
    analogWrite(ENB, arcSpeedSlow);
  }
  stopMoving();
}

void arcLeft(float distance_m) {
  resetEncoders();
  float targetPulses = calculatePulses(distance_m);
  digitalWrite(MLa, HIGH); digitalWrite(MLb, LOW);
  digitalWrite(MRa, HIGH); digitalWrite(MRb, LOW);
  while (averagePulseCount() < targetPulses) {
    analogWrite(ENA, arcSpeedSlow);
    analogWrite(ENB, arcSpeedFast);
  }
  stopMoving();
}

void stopMoving() {
  analogWrite(ENA, 0); analogWrite(ENB, 0);
  digitalWrite(MLa, LOW); digitalWrite(MLb, LOW);
  digitalWrite(MRa, LOW); digitalWrite(MRb, LOW);
}

void resetEncoders() {
  countLeft = 0;
  countRight = 0;
}

void countLeftEncoder() {
  countLeft++;
}
```

```
void countRightEncoder() {
  countRight++;
}

long averagePulseCount() {
  return (abs(countLeft) + abs(countRight)) / 2;
}

float calculatePulses(float distance_m) {
  float wheelCircumference = pi * wheelDiameter;
  return (distance_m / wheelCircumference) *
      pulsesPerRevolution;
}

float calculateTurnPulses(float angle_deg) {
  float turnCircumference = pi * wheelBase;
  float distance = (turnCircumference * angle_deg) /
       360.0;
  return (distance / (pi * wheelDiameter)) *
      pulsesPerRevolution;
}

void finalVictory() {
  Serial.println("All␣6␣Trials␣Completed␣
      Successfully!");
  stopMoving();
}
```

Appendix B
MATLAB Analysis Code for Motion Error

```
% Lab 4 MATLAB Analysis     Fixed: Only Showing
    Figures
clc; clear; close all;

% --- Corrected Trial Data ---
motion_types = {'Straight␣1', 'Straight␣2', 'Rotate␣
    R', 'Rotate␣L', 'Arc␣Right', 'Arc␣Left'};
expected_distance = [100, 50, 0, 0, 70, 70];    % cm
actual_distance = [95, 47, 2, 3, 66, 68];       % cm
expected_rotation = [0, 0, 90, -90, 45, -45];   %
    degrees
actual_rotation = [3, 2, 85, -88, 43, -48];     %
    degrees

% --- Compute Errors ---
distance_errors = abs(expected_distance -
    actual_distance);
rotation_errors = abs(expected_rotation -
    actual_rotation);

% --- Plot 1: Distance Errors ---
figure;
bar(categorical(motion_types), distance_errors, '
    FaceColor', [0.2 0.6 0.8]);
title('Distance␣Error␣per␣Motion␣Type');
ylabel('Error␣(cm)');
xlabel('Motion␣Type');
ylim([0 max(distance_errors)+5]);
grid on;
set(gcf, 'Color', 'w'); % Set figure background to
    white

% --- Plot 2: Rotation Errors ---
figure;
bar(categorical(motion_types), rotation_errors, '
    FaceColor', [0.9 0.4 0.4]);
title('Rotation␣Error␣per␣Motion␣Type');
ylabel('Error␣(degrees)');
xlabel('Motion␣Type');
ylim([0 max(rotation_errors)+5]);
grid on;
set(gcf, 'Color', 'w');

% --- Summary Table ---
T = table(motion_types', expected_distance',
    actual_distance', ...
          distance_errors', expected_rotation',
              actual_rotation', ...
          rotation_errors', ...
          'VariableNames', {'Motion', 'Exp_Dist_cm',
               'Act_Dist_cm', ...
                            'Dist_Error_cm', '
                                Exp_Rot_deg', ...
                            'Act_Rot_deg', '
                                Rot_Error_deg'});

disp('====␣Lab␣4␣Trial␣Data␣Summary␣====');
disp(T);
```

Appendix C
Robot Build Documentation

The following figures illustrate the physical setup of the differential drive robot used in Lab Experiment 4. These images serve to document the wiring layout, sensor placement, and component arrangement essential for achieving accurate motion control using wheel encoders.

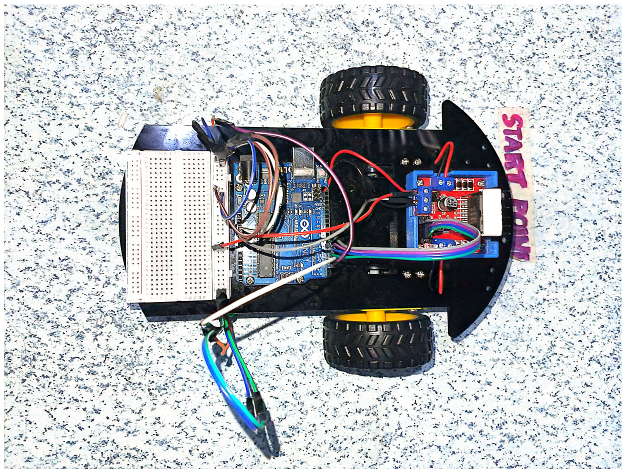

Fig. 4. Top View of the Robot at the Start Point. This image shows the full component layout including the Arduino Uno, L298N motor driver, breadboard, and encoder wiring. The robot is positioned on the marked start point prior to motion trials.

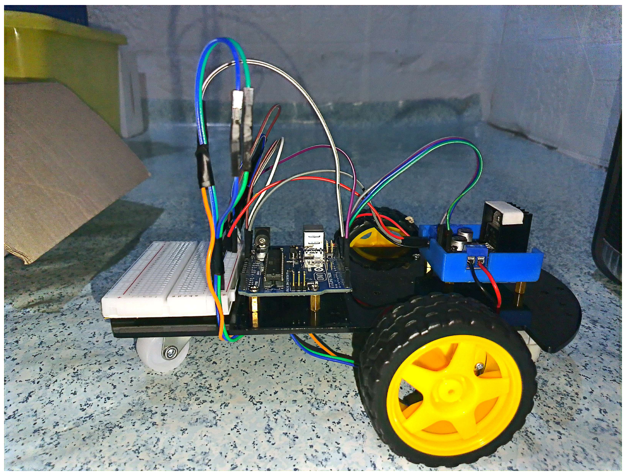

Fig. 5. Side Perspective Showing Component Mounting and Wiring. This view highlights the wiring organization for motor and encoder connections. The stability and accessibility of components are evident, ensuring safe and effective experimentation.

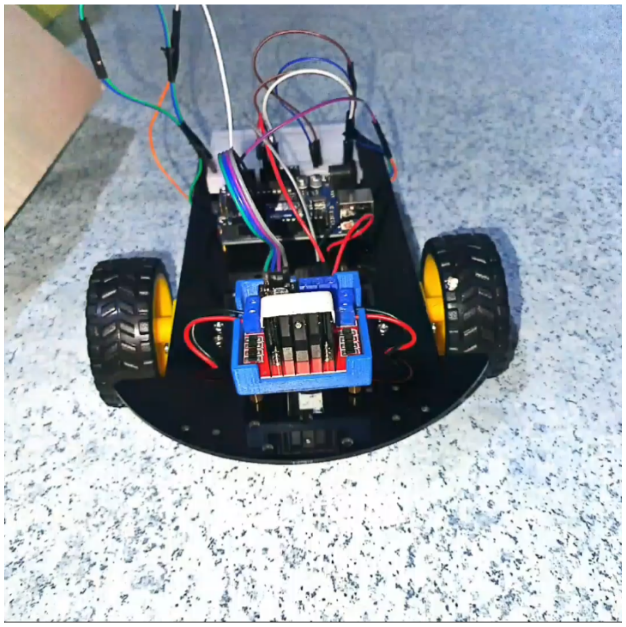

Fig. 6. Rear Angle Focused on Motor Driver and Encoder Terminals. Captured from behind the robot, this image emphasizes the encoder signal lines entering the breadboard and the connectivity to the L298N driver module.