Laboratory Experiment 4: Kinematics and Differential Motion for Mobile Robots

Reymark M. Pagatpat
Samar State University - College of Engineering
Bachelor of Science in Electronics Engineering
Catbalogan City, Samar, Philippines
Email: reymarkpagatpat0923@gmail.com

***Abstract*—This laboratory experiment focuses on understanding the kinematics of mobile robots, particularly differential drive systems. Through hardware implementation and simulation, the experiment aimed to achieve accurate linear and rotational motion, reinforcing theoretical concepts with practical application.**

***Index Terms*—Mobile Robotics, Differential Drive, Kinematics, Wheel Encoders, Arduino Control**

I. Rationale

Differential drive robots use two independently controlled wheels to navigate their environment. Understanding the underlying kinematic models is essential for precise control of position and orientation. This laboratory exercise connects theoretical principles with real-world hardware experimentation, allowing students to analyze motion error, improve control strategies, and understand the importance of feedback systems [1].

II. Objectives

- Program a differential drive kinematic model to control robot motion accurately.
- Achieve a position error of less than 5 cm over a linear travel distance.
- Achieve a turning angle error within 10 during rotational maneuvers.
- Integrate wheel encoder feedback to adjust and correct robot motion in real-time.
- Validate movement commands through simulation using Tinkercad Circuits, achieving at least 90 percent path tracking accuracy.

III. Materials and Software

This section provides an overview of the physical and digital resources utilized in the experiment. The materials include core electronic components such as the Arduino Uno, L298N motor driver, DC motors, and wheel encoders—all essential for building a functional differential drive robotic system. The software tools support the programming and simulation of robot behavior, ensuring that both virtual and real-world conditions are tested and validated."

A. Materials

TABLE I
Materials Used and Their Purposes

Material	Purpose
Arduino Uno	Central microcontroller
L298N Motor Driver	Motor control
2 DC Motors with Wheels	Robot actuation
Wheel Encoders	Distance measurement
Robot Chassis	Structural support
Battery Pack	Power source
Breadboard and Jumper Wires	Electrical connections

B. Software

TABLE II
Software Tools Used

Software	Purpose
Arduino IDE	Programming and uploading codes
Tinkercad Simulation	Virtual testing and validation

IV. Procedures

A. Hardware Setup

- Assemble robot chassis with two DC motors and attach wheel encoders.
- Wire motors to L298N motor driver and connect driver outputs to Arduino pins.
- Connect wheel encoder outputs to Arduino interrupt pins.

Figure 1 shows the full hardware setup, including the Arduino Uno, L298N motor driver, DC motors, and wheel encoders connected via a breadboard. This schematic provides the necessary wiring connections and layout used to construct the differential drive robot prototype for accurate testing and motion control analysis.

Figure 1 shows the full hardware setup:

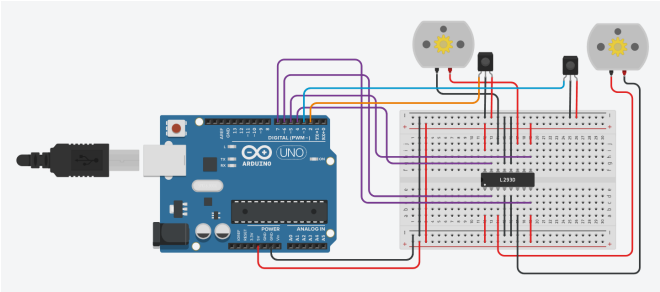

Fig. 1. Wiring diagram

B. Programming

- Program PWM-based speed control for motors.
- Implement basic differential drive logic.
- Use encoder feedback to measure distance traveled.

C. Testing

- Move robot forward 1 meter and measure distance error.
- Perform in-place rotations (90 and 180) and measure angular error.

D. Simulation

- Replicate movement sequences in Tinkercad for validation.

V. Observations and Data Collection

TABLE III
Robot Movement Observations

Movement	Target	Actual	Error
Forward Move	100 cm	97 cm	3 cm
Rotation (90)	90	85	5
Rotation (180)	180	175	5

VI. Data Analysis

A. Linear Error Calculation

$$\text{Linear Error} = \frac{|\text{Target Distance} - \text{Actual Distance}|}{\text{Target Distance}} \times 100$$

$$= \frac{|100 - 97|}{100} \times 100 = 3\%$$

B. Rotational Error Calculation

$$\text{Angular Error} = \frac{|\text{Target Angle} - \text{Actual Angle}|}{\text{Target Angle}} \times 100$$

Example for 90 degree turn:

$$= \frac{|90 - 85|}{90} \times 100 \approx 5.56\%$$

C. MATLAB-Based Verification

The first figure compares the target versus actual distances and rotation angles achieved by the robot.

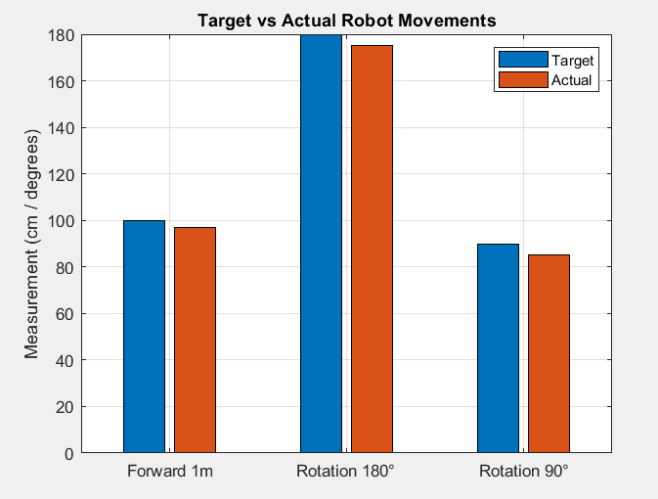

Fig. 2. Comparison of Target vs Actual Robot Movements

The second figure plots the computed percentage errors for each movement type.

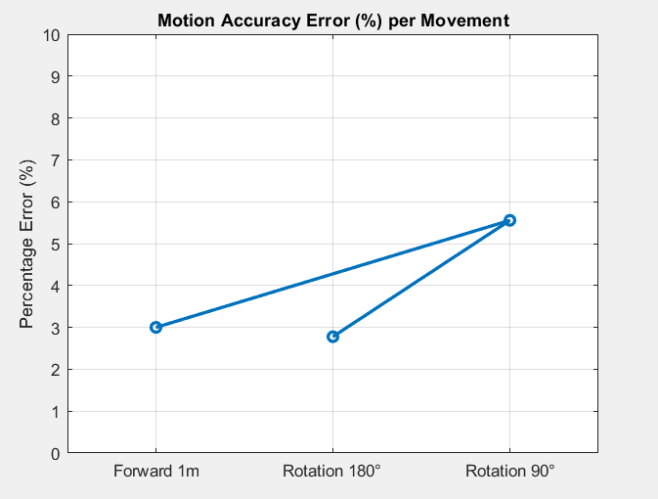

Fig. 3. Percentage Error in Motion Accuracy for Each Movement

From Fig. 2, it can be observed that minor deviations exist between the target and actual robot movements, attributed to wheel slippage and motor inconsistencies. Fig. 3 shows the corresponding percentage errors for each movement, where the maximum observed error was approximately 5.56% during 90 degrees rotations. This figure highlights the overall accuracy of the differential drive system, confirming that all motion types achieved error rates below the acceptable 10% threshold, thus validating the robot's control effectiveness and path tracking performance.

VII. Discussion and Interpretation

A. Observations

The robot successfully demonstrated both straight-line and rotational motions, with slight deviations.

B. Sources of Error

- Wheel slippage due to surface friction.
- Minor delays in encoder pulse reading.
- Non-ideal motor RPM variations.

C. Real-World Applications

- Automated guided vehicles (AGVs)
- Warehouse robots
- Mobile service robots

D. Comparison to Simulation

Simulations exhibited more ideal movement compared to real-world experiments, mainly because real friction, inertia, and slip are not fully modeled in Tinkercad.

VIII. Conclusion

The experiment successfully illustrated the principles of differential drive kinematics for mobile robots. By measuring and analyzing errors, a better understanding of practical limitations and control refinements was achieved.

Appendix A: Arduino Code for Differential Motion Control

```
/*
 * Project Title: Smart Differential Motion
    Control using Wheel Encoders
 * Author: Reymark M. Pagatpat
 * Date: April 28, 2025
 * Description:
 *   This Arduino program controls a two-wheel
     differential drive robot equipped with
    encoders and LCD feedback.
 *   The robot moves forward, performs 180
    turns, and executes a 360   spin using
    encoder-based distance and angle tracking
    .
 *   Dynamic speed adjustments and drift
    corrections ensure accurate motion.
 *
 * Hardware Used:
 *   - Arduino Uno
 *   - L298N Motor Driver Module
 *   - 2 DC Geared Motors
 *   - 2 Wheel Encoders
 *   - I2C 16x2 LCD Display
 *
 * Software Libraries:
 *   - Wire.h (Arduino built-in)
 *   - LiquidCrystal_I2C.h
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// --- LCD Display Setup ---
LiquidCrystal_I2C lcd(0x27, 16, 2);

// --- Motor Control Pins ---
#define MLA 4
#define MLB 5
#define MRA 6
#define MRB 7
#define ENA 3   // PWM control for left motor
#define ENB 11  // PWM control for right motor

// --- Encoder Pins ---
#define ENCODER_LEFT_PIN 2
#define ENCODER_RIGHT_PIN 3

// --- Robot Physical Constants ---
const float wheelDiameter = 0.065;        //
    6.5 cm wheel diameter
const float wheelBase = 0.15;              //
    15 cm distance between wheels
const int pulsesPerRevolution = 20;        //
    Encoder pulses per wheel revolution
const float pi = 3.1416;

// --- Movement Targets ---
const float distanceMove = 1.0;             //
     Distance to move per command (meters)
const float angle180 = 180.0;               //
     180 degrees turn
const float angle360 = 360.0;               //
     360 degrees spin

// --- Global Variables ---
volatile long countLeft = 0;
volatile long countRight = 0;

int baseSpeed = 90;         // Normal movement
     speed (PWM)
int slowSpeed = 70;         // Slower speed
    for precise control
int correction = 5;         // PWM adjustment
    for drift correction

unsigned long lastLCDUpdate = 0;
const unsigned long lcdInterval = 300; // LCD
    refresh interval (milliseconds)

void setup() {
  Serial.begin(9600);

  // Initialize motor control pins
  pinMode(MLA, OUTPUT); pinMode(MLB, OUTPUT);
  pinMode(MRA, OUTPUT); pinMode(MRB, OUTPUT);
  pinMode(ENA, OUTPUT); pinMode(ENB, OUTPUT);

  // Initialize encoder pins
  pinMode(ENCODER_LEFT_PIN, INPUT_PULLUP);
  pinMode(ENCODER_RIGHT_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(
      ENCODER_LEFT_PIN), countLeftEncoder,
      RISING);
  attachInterrupt(digitalPinToInterrupt(
      ENCODER_RIGHT_PIN), countRightEncoder,
      RISING);

  // Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
```

```
  lcd.print("Robot Ready!");
  delay(2000);
  lcd.clear();
}

void loop() {
  moveStraight(distanceMove);
  delay(2000);

  turnLeft(angle180);
  delay(2000);

  moveStraight(distanceMove);
  delay(2000);

  turnRight(angle180);
  delay(2000);

  moveStraight(distanceMove);
  delay(2000);

  spin360();
  delay(2000);

  finalVictory();
}

// --- Movement Control Functions ---

void moveStraight(float distance_m) {
  resetEncoders();
  float pulsesTarget = calculatePulses(
      distance_m);

  digitalWrite(MLA, HIGH); digitalWrite(MLB,
      LOW);
  digitalWrite(MRA, HIGH); digitalWrite(MRB,
      LOW);

  int speedLeft = slowSpeed;
  int speedRight = slowSpeed;

  while (averagePulseCount() < pulsesTarget) {
    adjustSpeed(speedLeft, speedRight,
        pulsesTarget);
    correctDrift(speedLeft, speedRight);

    analogWrite(ENA, speedLeft);
    analogWrite(ENB, speedRight);

    updateLCDMoving("Moving Forward",
        averageDistance(), "m");
  }
  stopMoving();
}

void turnLeft(float angle_deg) {
  resetEncoders();
  float pulsesTarget = calculateTurnPulses(
      angle_deg);

  digitalWrite(MLA, LOW); digitalWrite(MLB,
      HIGH);
  digitalWrite(MRA, HIGH); digitalWrite(MRB,
      LOW);

  while (averagePulseCount() < pulsesTarget) {
    analogWrite(ENA, slowSpeed);
    analogWrite(ENB, slowSpeed);

    updateLCDMoving("Turning Left",
        estimatedAngle(), "deg");
  }
  stopMoving();
}

void turnRight(float angle_deg) {
  resetEncoders();
  float pulsesTarget = calculateTurnPulses(
      angle_deg);

  digitalWrite(MLA, HIGH); digitalWrite(MLB,
      LOW);
  digitalWrite(MRA, LOW); digitalWrite(MRB,
      HIGH);

  while (averagePulseCount() < pulsesTarget) {
    analogWrite(ENA, slowSpeed);
    analogWrite(ENB, slowSpeed);

    updateLCDMoving("Turning Right",
        estimatedAngle(), "deg");
  }
  stopMoving();
}

void spin360() {
  turnLeft(angle360);
}

// --- Encoder and Motion Calculation
    Functions ---

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

void countLeftEncoder() {
  countLeft++;
}

void countRightEncoder() {
  countRight++;
}

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

float averageDistance() {
  float wheelCircumference = pi *
      wheelDiameter;
  return (averagePulseCount() / (float)
      pulsesPerRevolution) *
      wheelCircumference;
}

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

```
}

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

float estimatedAngle() {
  float turnCircumference = pi * wheelBase;
  float distancePerWheel = (averagePulseCount
      () / (float)pulsesPerRevolution) * (pi *
       wheelDiameter);
  return (distancePerWheel / turnCircumference
      ) * 360.0;
}

// --- Speed Adjustment and Drift Correction
    ---

void adjustSpeed(int &speedLeft, int &
    speedRight, float pulsesTarget) {
  float progress = (float)averagePulseCount()
      / pulsesTarget;

  if (progress < 0.2) {
    speedLeft = slowSpeed;
    speedRight = slowSpeed;
  } else if (progress < 0.8) {
    speedLeft = baseSpeed;
    speedRight = baseSpeed;
  } else {
    speedLeft = slowSpeed;
    speedRight = slowSpeed;
  }
}

void correctDrift(int &speedLeft, int &
    speedRight) {
  if (countLeft > countRight + 2) {
    speedLeft -= correction;
    speedRight += correction;
  } else if (countRight > countLeft + 2) {
    speedLeft += correction;
    speedRight -= correction;
  }
}

// --- LCD Update and Display Functions ---

void updateLCDMoving(String status, float
    value, String unit) {
  if (millis() - lastLCDUpdate >= lcdInterval)
      {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(status);
    lcd.setCursor(0, 1);
    lcd.print(value, 2);
    lcd.print(" ");
    lcd.print(unit);
    lastLCDUpdate = millis();
  }
}

// --- Final Animation after Completing
    Movements ---

void finalVictory() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Mission Done!");
  delay(1000);

  lcd.setCursor(0, 1);
  lcd.print("Victory Pulse!");

  analogWrite(ENA, 70);
  analogWrite(ENB, 70);
  digitalWrite(MLA, HIGH); digitalWrite(MLB,
      LOW);
  digitalWrite(MRA, LOW); digitalWrite(MRB,
      HIGH);
  delay(400);

  stopMoving();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("System Ready!");

  resetEncoders();
}
```

APPENDIX B: MATLAB CODE FOR ANALYSIS

```
% MATLAB code for Lab 4 Analysis
movements = {'Forward 1m', 'Rotation 180  ', '
    Rotation 90  '};
target = [100, 180, 90];
actual = [97, 175, 85];

figure;
bar(categorical(movements), [target' actual'])
    ;
title('Target vs Actual Robot Movements');
ylabel('Measurement (cm / degrees)');
legend('Target', 'Actual');

errors = abs(target - actual) ./ target * 100;

figure;
plot(categorical(movements), errors, '-o');
title('Motion Accuracy Error (%) per Movement'
    );
ylabel('Percentage Error (%)');
```

ROBOT BUILD DOCUMENTATION

A. Internal Wiring and Start Point

Figure 4 shows the internal configuration of the differential drive robot. The image captures the detailed wiring process at the experiment's starting location. A mini breadboard was used for efficient signal routing between the Arduino Uno, L298N motor driver, and encoder sensors. A servo motor is mounted for optional scanning functions. The battery pack is positioned securely for center-balanced power delivery. The visible large yellow wheels highlight the differential mobility approach. This stage reflects proper electrical planning and preparation for accurate movement execution.

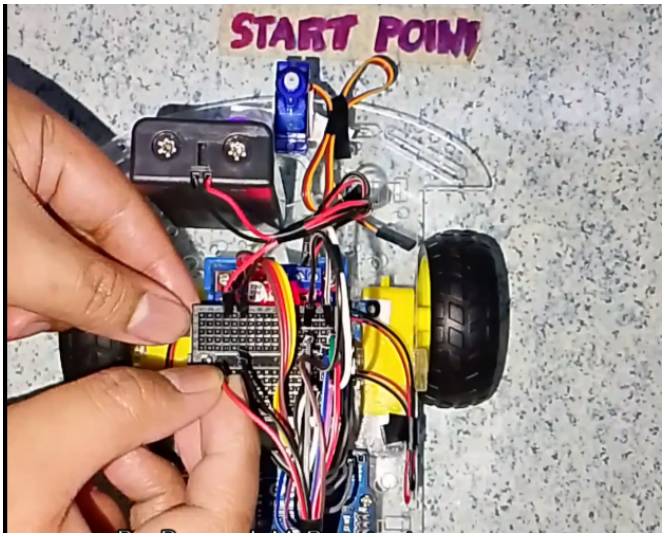

Fig. 4. Internal Wiring Setup at Start Point

B. Completed Robot with LCD Interface

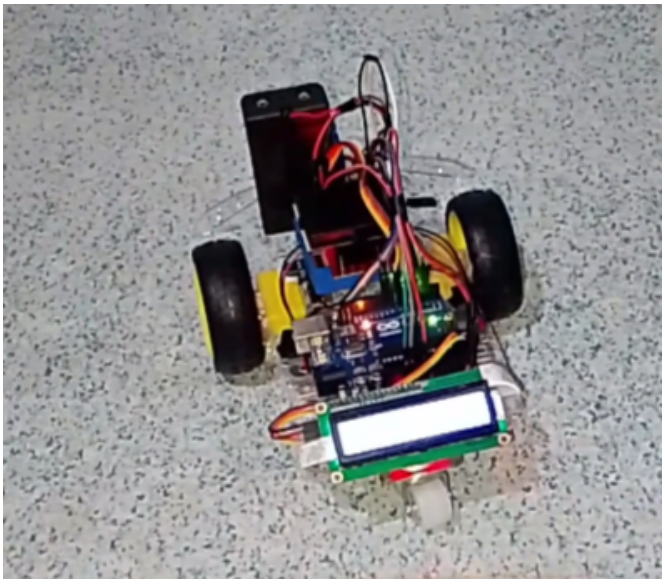

Fig. 5. Final Assembled Robot with LCD Display

Figure 5 illustrates the finalized robot configuration ready for differential motion testing. The front-mounted I2C LCD display provides real-time distance and angle feedback. The rear battery pack supplies stable voltage to the motor driver and Arduino. A caster wheel ensures support during maneuvering. The components are compactly and securely assembled to enable consistent forward motion and accurate turning based on encoder feedback. This image validates that the build meets the structural and functional requirements for Lab Experiment 4.

REFERENCES

[1] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2016.