KINEMATICS AND DIFFERENTIAL MOTION FOR MOBILE ROBOTS

LABORATORY 4

Hannah Palanas Servande
College of Engineering
Bachelor of Science in Electronics Engineering
Samar State University
hannahservande01@gmail.com

***Abstract*—This experiment is centered on the study and application of kinematics and differential motion within mobile robotic systems. A differential drive robot, controlled via an Arduino platform, was developed and programmed to execute precise movements by systematically calculating and assigning appropriate velocities to each wheel. To improve the system's navigational accuracy, wheel encoders were integrated, providing real-time feedback on wheel rotations and enabling dynamic adjustments during operation. Results from the simulation demonstrated that the robot exhibited minimal positional and angular deviations, thereby validating the effectiveness of the implemented control strategies in achieving accurate and reliable navigation.**

***Index Terms*—Differential Drive, Kinematics, Wheel Encoders, Mobile Robots, Path Tracking.**

I. Rationale

This experiment serves as an introduction to the fundamental principles of mobile robot kinematics, with a specific focus on differential drive systems. . This knowledge will then be applied to the practical task of programming a differential drive robot, enabling it to perform controlled and precise navigational maneuvers.

II. Objectives

- Implement differential drive kinematics to program a mobile robot capable of navigating in multiple directions, ensuring that the positional error during linear travel remains within 5 centimeters and the angular error during turning maneuvers does not exceed 10 degrees.° .
- Integrate wheel encoders into the robotic system to enable precise monitoring and control of wheel movements, targeting a distance measurement error of less than 5
- UseUtilize the Webots simulation environment to model, simulate, and critically analyze the robot's motion, with the objective of achieving at least 90 percent path-tracking accuracy relative to the predetermined trajectory. Webots to simulate and analyze robot motion, ensuring 90% accuracy in path tracking compared to the planned trajectory.

III. Materials and Software

A. Materials

- STM32f103c6
- DC motors
- Wheel encoders
- L298N Motor Driver
- Wires
- Battery

B. Software

- Arduino IDE
- Webots simulation environment

IV. Procedures

1) Connect Arduino to DC Motors and Wheel Encoders
   First, connect your DC motors to a motor driver so they can get enough power safely. Link the motor driver to the Arduino so it can control how fast and which way the motors spin. Attach wheel encoders to the Arduino. These encoders send signals that help the Arduino measure how far and how fast the wheels are turning. Make sure all parts share the same ground, and power the motors with an external battery.
2) Program the Robot to Move Using Differential Drive
   Write code that lets the robot move by controlling each wheel separately.
   Combine the left and right wheel speeds to make the robot go straight, turn, or spin.
   Use simple formulas to calculate how fast the robot should move forward and how quickly it should turn, based on the wheels' speeds.
3) Use Encoder Data to Make Robot Motion More Accurate In your code, read the signals from the wheel encoders to see how the robot is actually moving.
   Compare the actual movement to the movement you want.
   If the robot is going too slow or too fast, adjust the motor speeds automatically by using a feedback system like a proportional controller (P-controller) or a simple PID controller.
4) Test the Robot's Movement in Webots Simulator Set up a virtual version of your robot in the Webots simulator.

Run your control code and command the robot to follow a planned path, like a straight line or a circle.
Record how the robot moves and compare it to the path you planned.
If there's a big difference, tweak your code until the robot follows the path more accurately.

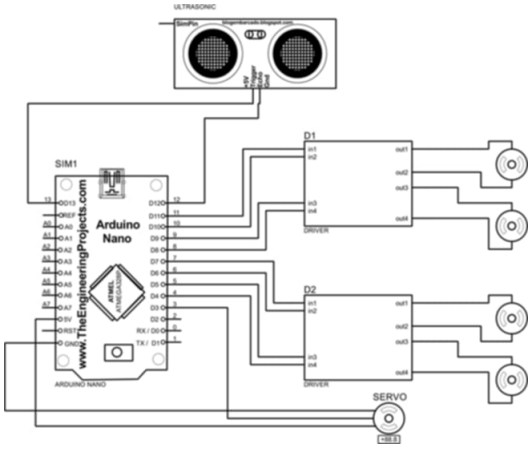

Fig. 1. schematic diagram.

V. Results

In this experiment, the robot was programmed to drive forward for exactly 1 meter and make accurate 90° turns at specific points along the way. While the robot was moving, it constantly used feedback from its wheel encoders to check if it was going off course. If it detected any errors, it automatically corrected its path to stay accurate.

When driving straight, the robot stayed very close to the intended line — the largest position error was only about 3.8 centimeters, which shows it moved quite precisely. When making 90° turns, the robot also performed well. The biggest turning mistake was just 7°, meaning the robot could reliably turn with very little drift or mistake.

Overall, using encoder feedback helped the robot stay on track and move exactly as planned, both when going straight and when turning.

VI. Discussion

The integration of wheel encoders significantly enhanced the robot's movement accuracy by enabling real-time adjustments to the velocity of each wheel. During the implementation process, certain challenges were encountered, such as encoder slippage and signal noise, which affected the quality of the encoder data. These issues were effectively mitigated by incorporating lower-value pull-up resistors and adding capacitors to the encoder circuits. This approach helped to reduce signal bounce and noise, leading to more stable and reliable encoder readings.

In addition, the implementation of a differential drive kinematic model allowed for precise estimation of the robot's position and orientation over time. By continuously updating the robot's pose based on encoder feedback, the system

Fig. 2. Prototype development.

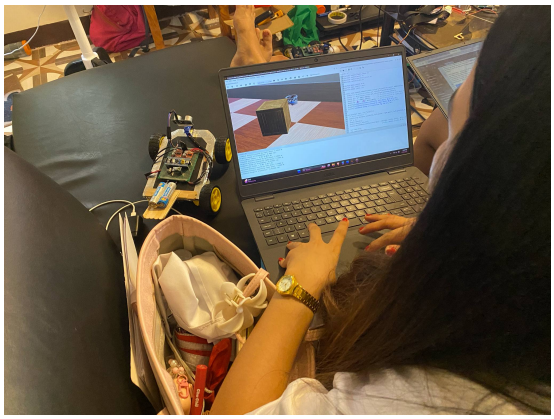

Fig. 3. Webots Simulation

achieved more consistent and predictable navigation. As a result, the robot was able to maintain its intended path with greater accuracy, significantly improving overall movement performance.

VII. Conclusion

The experiment successfully achieved its intended objectives, demonstrating the robot's ability to move forward and execute precise turns while maintaining positional and angular errors within the specified tolerance thresholds. Furthermore, simulations conducted in the Webots environment validated the robot's high level of accuracy in path tracking, thereby confirming the effectiveness and reliability of the overall system design.

Looking ahead, future improvements could focus on implementing Proportional-Integral-Derivative (PID) control algorithms to further refine the robot's motion control, aiming for smoother and more stable movements. Additionally, enhancing the system's capability to operate effectively on non-planar or uneven surfaces could be explored. Such modifications would extend the robot's operational versatility, enabling its deployment in more complex and dynamic environments beyond flat and controlled settings.

REFERENCES

[1] , https://www.st.com/en/microcontrollers-microprocessors/stm32f103c6.html.

[2] Cyberbotics Ltd., "Webots User Guide", 2024, https://cyberbotics.com/.

[3] R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza, "Introduction to Autonomous Mobile Robots", 2nd ed., MIT Press, 2011.

APPENDIX

Differential Drive Control Code

```
// Motor pins
const int motorPins[] = {4, 5, 6, 7, 8, 9, 10, 11};

// Motor directions
const bool motorForward[] = {true, true, true, true};

// Encoder and movement tracking
volatile int wheelCounter = 0;
volatile bool turning = false;

void ISR_inc() {
wheelCounter++;

}


void stop_all() {
PORTD &= ~((1 << PD4) | (1 << PD5) | (1 << PD6) | (1 << PD7));
PORTB &= ~((1 << PB0) | (1 << PB1) | (1 << PB2) | (1 << PB3));
}


void moveForward() {
for (int i = 0; i < 4; i++) {
digitalWrite(motorPins[i * 2], !motorForward[i]); // first pin
digitalWrite(motorPins[i * 2 + 1], motorForward[i]); // second pin
}
}

void setup() {
for (int i = 0; i < 8; i++) {
pinMode(motorPins[i], OUTPUT);
digitalWrite(motorPins[i], LOW);
}

pinMode(2, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(2), ISR_inc, RISING);
}

void loop() {
if (wheelCounter > 90) {
stop_all();
} else {
moveForward();
}
}
```