Lab 1: Basic Robot Programming - Report

Matias Cinera
Computer Science Department, *University of South Florida*
Tampa, Florida, 33612, USA
cinera@usf.edu

I. Introduction

This document is the report of my 1st lab from my Spring 2023 class CDA-6626 Autonomous Robots, as a graduate student. The individuals who are overviewing this project are our professor Dr. Alfredo Weitzenfeld (Professor in the department of Computer Science) and Mr. Chance Hamilton (Teaching Assistant & Graduate Student)

II. Task 1 - Motors

A. Motor Control

To control both the left and right motors of the robot, the Robot module needs to be imported from the controller library in Webots. Then, initialize two variables for each motor in the simulation. To set the velocity of each motor call setVelocity (v) function and pass a value between o and 6.28 radians (or 2π). However, the motor input needs to be given angular velocity. Since the input is given in inches per second, it needs to be converted to φ (angular velocity) before passing it to the motor. This can be achieved by dividing the input by the wheel's radius (φ = v/r) which is 0.8 inches.

B. Maximum Motor Speed

The max rotational velocity per motor is 2π and the given wheel diameter is 1.6 inches. Thus, the maximum speed per motor + wheel is **5.024 *in/s*** (2×π×r).

C. Invalid Inputs

There are three possible invalid inputs:

- Speed > **5.024 *in/s*** (max speed)
- Speed < 0 in/s (speed is negative)
- Time <= 0 (invalid time)

In the case of invalid speed motor speed inputs I re-set them either to 5.024 ***in/s*** (if greater than max speed) or to 0 in/s (if speed is negative). In the case that time is less or equal to zero, the task will be skipped.

III. Task 2 – Distance & Decoders

A. Robot Control Kinematics

To calculate the distance of the robot only ω and R need to be computed. Since $\mathbf{v_l}$,$\mathbf{v_r}$, and **t** are already given by the input vector, only the radius of the ICC and the angular velocity of the curve are needed to compute **s**. A new equation for s can be derived by replacing **v** with **ωR**, thus making ***s = ωRt***.

$$\omega = \frac{(v_l - v_r)}{d}$$

$$R = | \, d_{mid} \frac{(v_l + v_r)}{(v_l - v_r)} \, |$$

$$v = \omega R$$

$$s = vt = \omega R t$$

Figure 1: Kinematic equations used to compute s

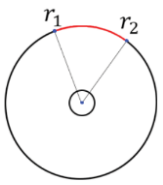
When implementing the equations, **ω** and **R** can have different values depending on the dominant wheel ($\mathbf{v_l}$ -$\mathbf{v_r}$ or $\mathbf{v_r}$ -$\mathbf{v_l}$). To handle this, I took the absolute value of the expression, this will ensure to return the difference between the to values.

```
omega = abs(vl-vr)/distBtwWhe
R = abs( distBtwWheR * (vl+vr)/abs(vl-vr) )
s = (omega*t)*R
return s
```

Figure 2: Implementation of kinematic equations in Python

B. Encoders & Wheel speed

To calculate the distance covered by the wheel based on the input we need to calculate $\mathbf{v_l}$ and $\mathbf{v_r}$ based on the encoder readings before and after the task was performed. Each encoder returns the position of the wheel, thus, by having the first and second positions of the wheel we can determine the angular velocity of each wheel given the time it took to complete the task. The encoder outputs the position in radians, so to calculate distance get **v** you need to multiply by the radius of the wheel ***(v = rφ)***.



$$\varphi_l, \varphi_r = \frac{r_1 - r_2}{t}$$

Figure 3: Equation used to calculate φ based on encoder readings

IV. Conclusions

There are a few problems I encountered when dealing with time and encoders. Since the shortest time step in the simulation is 32ms, tasks will always take a few extra milliseconds. Because of this, both encoder readings will report a higher value than expected. Thus, the distance covered is higher than the theoretical distance. However, this was the expected behavior. When dealing with simulations, there will always be limitations. In this case, it was the time step.

References

[1] Webots Reference Manual. Webots. (n.d.). Retrieved January 23, 2023, from https://www.cyberbotics.com/doc/reference/motor?tab-language=python