## Objective

The objective for this lab is to use kinematics to move a 4-wheeled differential drive robot through a set of pre-defined waypoints. This is done using the Webots simulator that runs python code as instructions for the robot to follow. The following sections describe how this task was completed:

## Waypoints

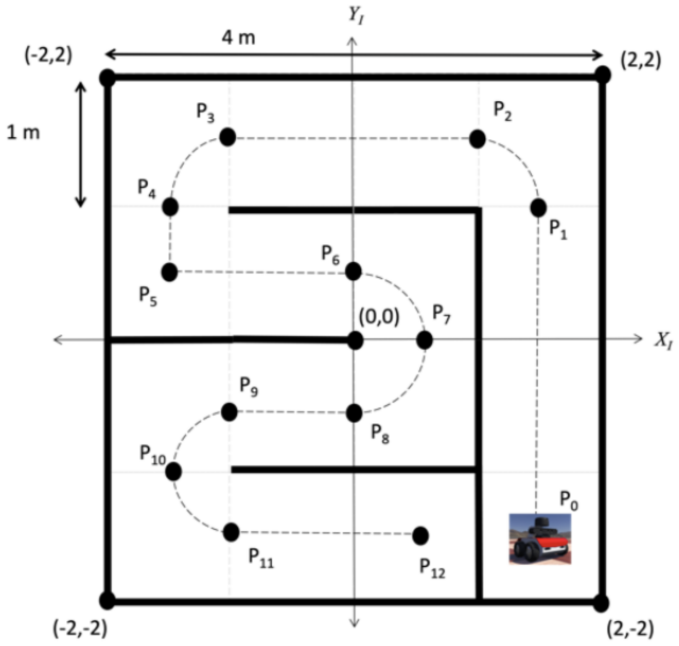


Image 1: A maze diagram that shows a series of waypoints which are connected to reveal the path the robot must follow

The robot starts at P0 (1.5, -1.5) and must travel to P12 (0.5, -1.5) as indicated by Image 1. All calculations and measurements are in meters.

Here is the complete list of waypoints the robot must travel:

[ P0(1.5, -1.5) → P1(1.5, 1.0) → P2(1.0, 1.5) → P3(-1.0, 1.5) → P4(-1.5, 1.0) → P5(-1.5, 0.5) → P6(0.0, 0.5) → P7(0.5, 0.0) → P8(0.0, -0.5) → P9(-1.0, -0.5) → P10(-1.5, -1.0) → P11(-1.0, -1.5) → P12(0.5, -1.5) ]

# Kinematics

In order for the robot to follow this path, it requires the precise calculations of angular velocity (rate of change of a rotating object) for both left and right wheels. The angular velocity of the wheels determine whether it moves straight, curves, or rotates in place. This will be described in the following sections:

## Straight Line Motion

To accomplish a straight line motion, each wheel must have the same angular velocity, $\omega$.
For my program, I set the angular velocity for both wheels to 7.5 rad/sec.

$$\omega = 7.5\, rad/sec$$

Calculations for Linear Velocity: *Vl and Vr,*

$$V,\ Vl,\ Vr\ (linear\ velocity) = \omega\ (angular\ velocity) \times r\ (wheel\ radius)$$

By multiplying the angular velocity with the wheel radius, we can obtain the linear velocity of the robot in *m/s*. Linear velocity is the $\Delta x\ displacement\ (meters)$ over the $\Delta t\ time\ (seconds)$ .

Since the robot is moving straight, the left and right wheel velocities will be the same, and therefore, the average of both will also be the same:

$$(Vl = Vr = Vavg)$$

Distance: $\Delta x$,

$$\Delta x = V\ (linear\ velocity) \times t\ (time)$$

This is one way you can obtain the displacement - the distance between two waypoints - and the time - number of seconds between two waypoints. However, in my program I was able to obtain the displacement through function parameters.

Time: *t*,

$$t\ (time) = \Delta x\ (displacement) \div V\ (linear\ velocity)$$

Time can be obtained by dividing the distance between two waypoints by the linear velocity.

## Curved Line Motion

From the diagram provided via Image 1, you can observe that from waypoints P1 → P2 requires a curved line motion.

To achieve this, we must consider several variables:

$R = radius\ of\ circle$
$d_{mid} = half\ the\ axel\ length$
$V_r = linear\ velocity\ of\ right\ wheel$
$V_l = linear\ velocity\ of\ left\ wheel$

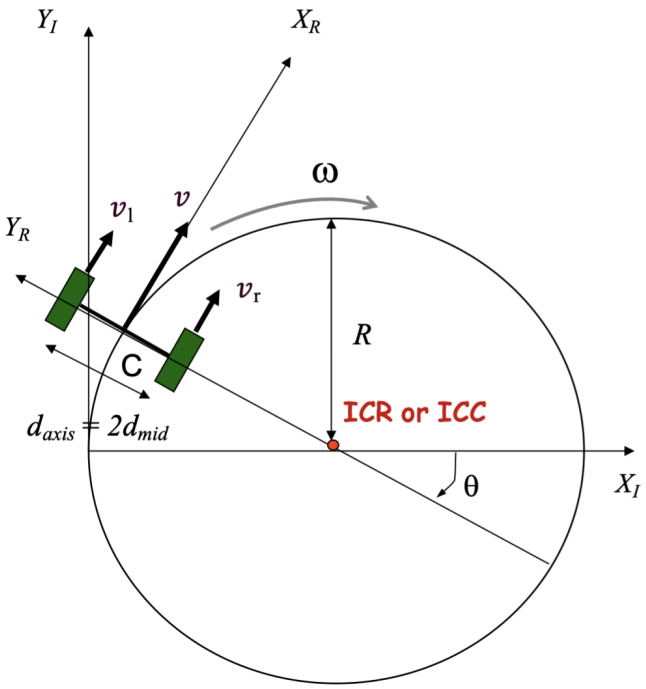


Image 2: A robot moving along the circumference of a circle with radius R

We can get Vr linear and angular velocities (the wheel that rides inside the circle as shown by image 2) with the following equations:

$$\omega_l = \frac{V}{R + d_{mid}} \rightarrow V_r = \omega l \times (R - d_{mid}) \rightarrow$$

$$\omega_r = \frac{V_r}{r\ (wheel\ radius)}$$

With the equations above, we can set the angular velocity of the right wheel to $\omega_r$. With the left wheel's angular velocity unchanged, this will perform the curve as shown in image 2. This is also the way I've implemented kinematics in my function.

Additionally, distance for the inner and outer wheels can be found as such:

$$d_r\,(inner\ wheel) = 2\pi(R - d_{mid}) / 4$$
$$d_l\,(outer\ wheel) = 2\pi(R + d_{mid}) / 4$$

Subtracting half the axel length gives the radius of the circle the inner wheel travels. Adding half the axel length gives the radius the outer wheel travels. We divide by 4 as we only want it to travel a quarter circle as indicated by the robot path.

### Rotation Motion

Using an angle provided in radians multiplied by the $d_{mid}$ will provide the rotate distance. To make the robot rotate is easy.

$$d_{rotate} = \theta \times d_{mid}$$

Set one side of the motors to negative angular velocity and the other side to positive. A negative and positive motion will cancel out the linear velocity of either side which will result in a rotation in place.

Switching the negative angular velocities from left to right motors can determine which direction the robot rotates – clockwise or counter-clockwise.

## Position

To determine position of the robot, this required the use of encoders, x, and y variables.

First I obtained the current encoder average which adds encoder readings of all four wheels and divides the sum by 4.

To get the x and y positions required the equations:

$$x = x_0 + \Delta x \times cos(\Theta)$$
$$y = y_0 + \Delta x \times sin(\Theta)$$