Path Control –
Autonomous Systems lab

מוגש על ידי:

שם פרטי: עמית שם משפחה: דוידי ת.ז: 208630103

Autonomous Systems lab
TAU Faculty of Engineering
2nd semester, 2022-2023

Introduction

Our task is to draw a square, a Circle, and the Infinity shape with the Zumo robot.

To complete that task, we opted to use the Multi Point-to-Point path control algorithm that was depicted in class.

Implementation

1. Velocity control with PI controller

We implemented a controller in code the same way we did previously with the home-kit assignment.

$$K_p = 1500, \qquad K_i = 400, \qquad OutMin = -400, \qquad OutMax = 400$$

We also added a LPF to filter out the spikes in the odometry readings and to achieve an overall smoother result:

$$Filtered\ Velocity = \alpha \cdot CurrentVelocity + (1-\alpha) \cdot PreviousVelocity$$

$$\alpha = 0.5$$

Note – different values of α may have yielded better results, alas we did not test for that.

2. Odometry

The odometry is the mainly the same as provided, a measurement of dx is done by reading the $encoderCounts$ and resetting it to 0, transforming the measurement to meters, and calculating $d\theta$ by

$$d\theta = \frac{dx_{right} - dx_{left}}{WheelsDistanceIn[mm]=0.0985}$$

The position is then updated:

$$posx = currentPosX + \cos\left(\theta + \frac{d\theta}{2}\right) \cdot \frac{dx_{left} + dx_{right}}{2}$$

$$posy = CurrentPosY + \sin\left(\theta + \frac{d\theta}{2}\right) \cdot \frac{dx_{left} + dx_{right}}{2}$$

We tweaked the value of $d\theta$ to compensate for the imprecision of the robot's model (single constant point of rotation) and slip.

We also calculated the velocity and passed it through the LPF mentioned above.

3. Path Control – Multi Point-to-point

We implemented (with the help of ChatGPT) the P2P algorithm as depicted in the MATLAB simulation and the Theoretical background.

We calculate:

$$V_r = [\cos\theta \quad \sin\theta]$$

$$V_t = [X_{Desired} - posX \quad Y_{Desired} - posY]$$

Where $X_{Desired}, Y_{Desired}$ are from the desired path (we explain in further detail on the next pages)

And $posX, posY$ are the position calculations from the Odometry procedure.

We then calculate the $dot\ product$ $\frac{V_t \cdot V_r}{||V_t|| \cdot ||V_r||}$, and the direction value dir

We get the θ_t by $\text{acos}\left(\frac{V_t \cdot V_r}{||V_t|| \cdot ||V_r||} \cdot dir\right)$

We get $W_{Forward} = ||V_t||$

We get the trapezoid velocity profile (by limiting $w_{forward}$, and the increase of $w_{forward}$)

Finally, the signals that we feed into the PI Controller are:

$$W_{forward} - \theta_t \cdot 0.35$$

$$W_{forward} + \theta_t \cdot 0.35$$

4. Main flow – Block diagram

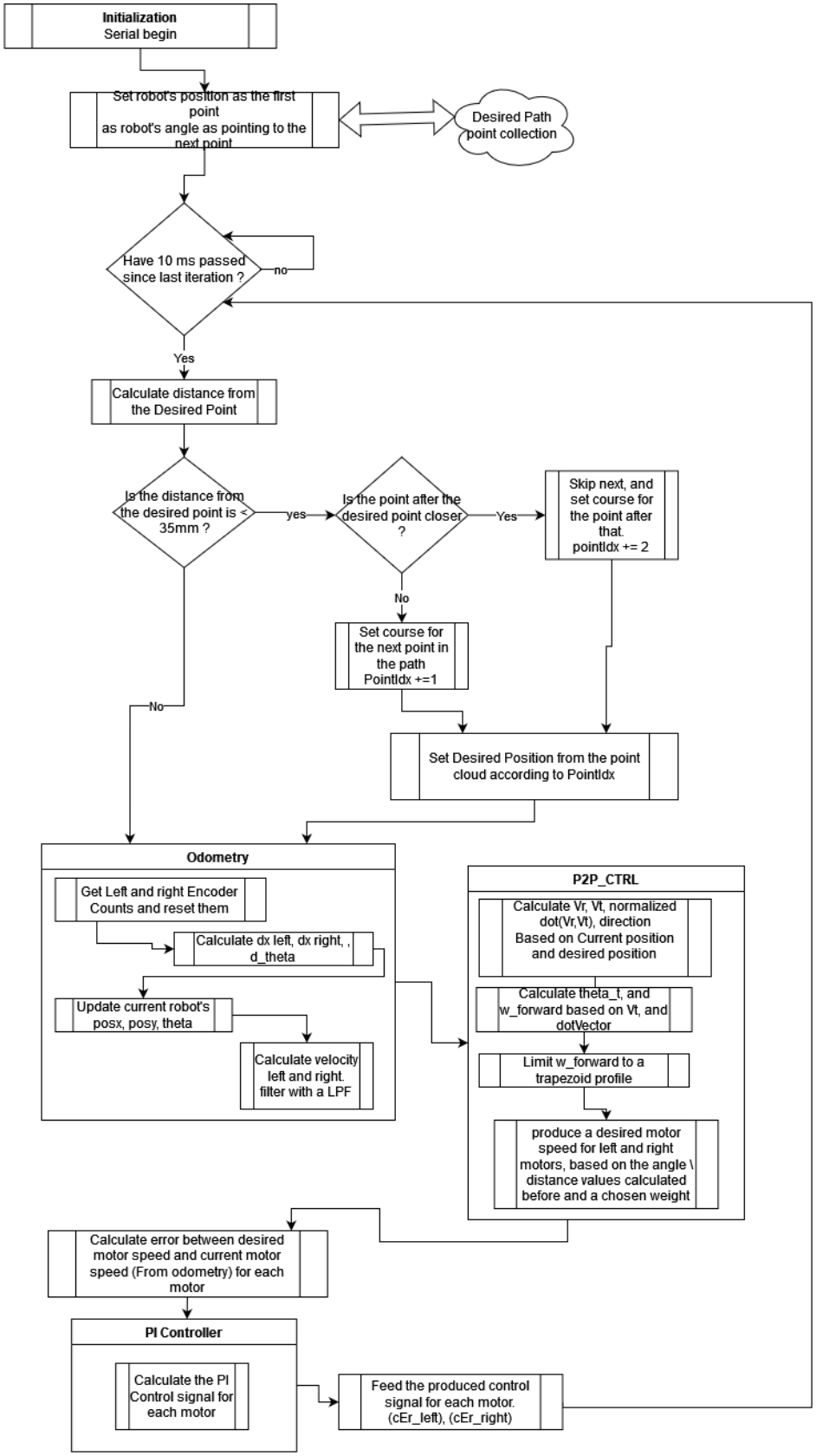

Fig 1. A block diagram of the path control program

5. Shape generation for the Paths

Infinity Shape

Math time~

To make our robot draw an infinity shape, we needed to augment the original infinity shape.

The infinity shape is given by the equations:

$$\Gamma(t) = \big(X(t), Y(t)\big) = \left(\sin(t), \frac{(\cos(t) - 1) \cdot \sin(t)}{1 + \sin^2(t)}\right)$$

For each point on the curve, we want a point that is 65mm vertical to it.

To achieve this, we calculate the velocity vector:

$$V(t) = \big(\dot{X}(t), \dot{Y}(t)\big) = \left(\cos(t), \frac{\cos^3(t) - 3\cos^2(t) + 2}{(\sin^2(t) + 1)^2}\right)$$

And for every point on the curve (t), we calculate:

$$u_p = \ Normalized\, V(t) = \frac{\big(\dot{X}(t), \dot{Y}(t)\big)}{\sqrt{\dot{X}(t)^2 + \ \dot{Y}(t)^2}}$$

We then calculate the cross product $O_p = \big[u_p \ \ 0\big]\, x\, [0\ 0\ 1]$ to get a vector perpendicular to the curve.

Finally, the point we want is given by $\Gamma + \alpha O_p$

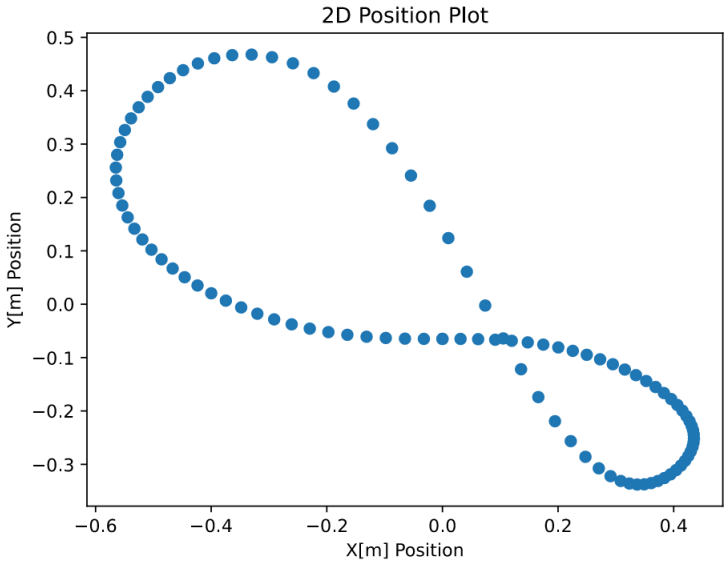

Fig 2. Augmented Infinity shape, that a robot can follow and draw a correct infinity.

Lab results [infinity]:

Measured position (obtained from odomtery):

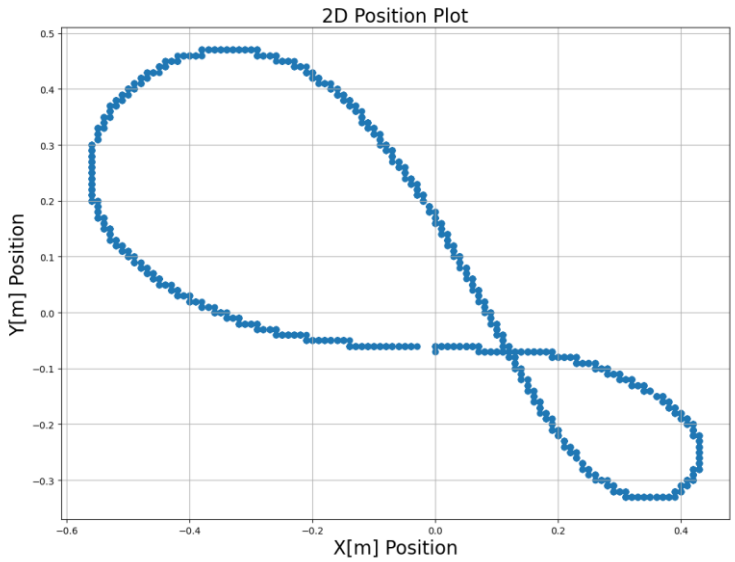

Fig 3. The measured position of the robot while following the augmented infinity. Serial output.

We see that the robot followed the desired path almost perfectly (according to the odometry measurements)

These odometry measurements are what the P2P considers, we did not implement a way of detecting robot's unexpected slip).

The shape drawn in lab:

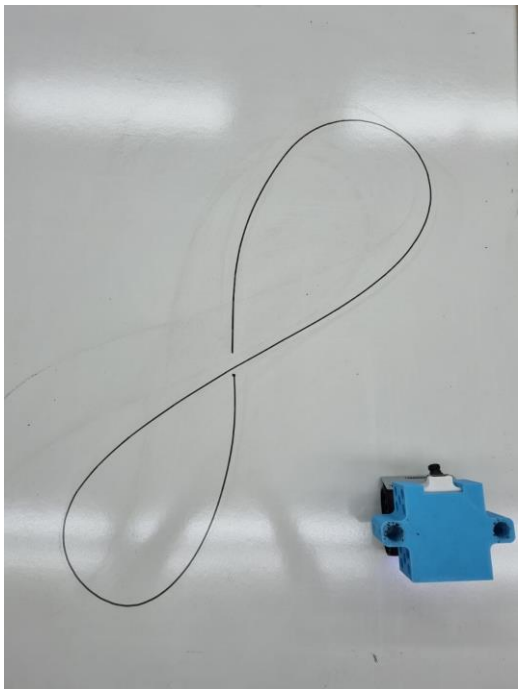

Fig 4. the Infinity shape drawn in lab by the robot while following the augmented infinity shape.

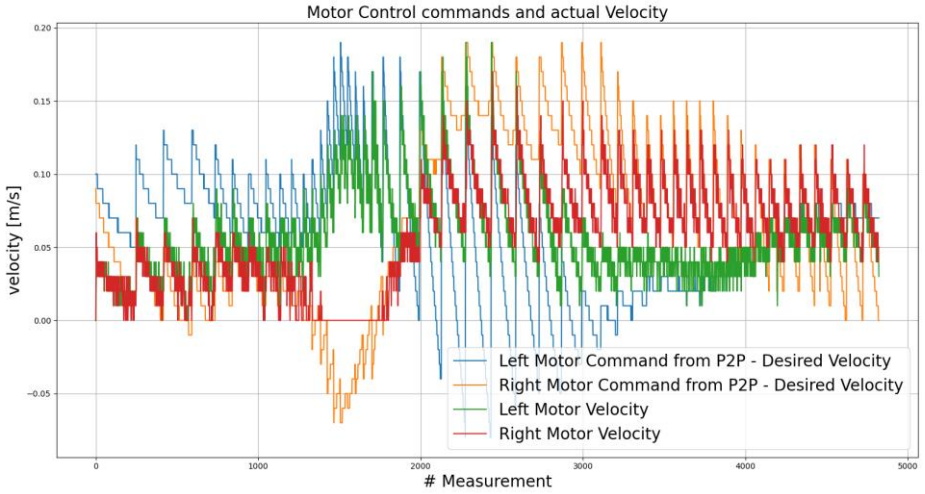

Fig 5. The motors Desired velocity (P2P output) and their measured velocity during the infinity shape movement

In the graph above, we see the motors' measured velocity and desired velocity, we can see that at the start of the movement, during the first right turn the left motor is faster than the right motor, as expected. And at the end of the movement, during the other turn, the right motor's speed is higher, again, as expected.

Also, we can see that the first turn is the smaller one, since the gap and value of the motor's speeds is higher.

Circle Shape

To make the robot draw a circle, we specify a radius, and produce a collection of points, however this will not suffice in case the pen is on the outer part of the robot. We instead want to follow a circle and a bit extra, to compensate for the bigger radius of the pen:

$$x, y = (\cos(\theta), \sin(\theta))$$

$$\theta \in (0,\ 2\pi + 0.35)$$

(Some points to the right side of the plot appear bigger than the others, that is because there are more points that overlap them – the 'extra' mentioned above)

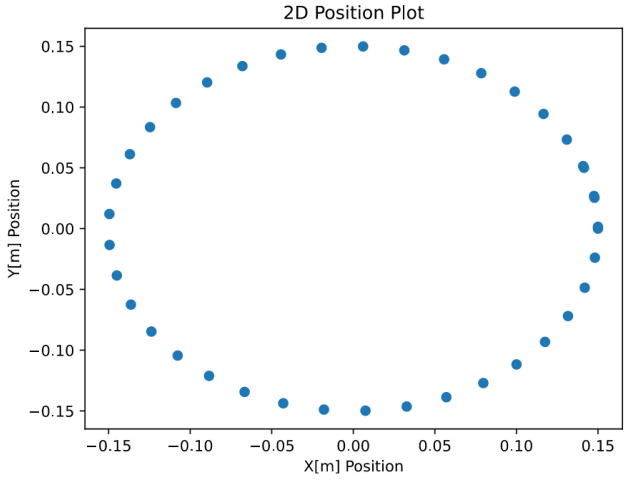

Fig 6. the desired Circle path fed to the robot.

Lab results [circle]:

Measured position (obtained from odomtery):

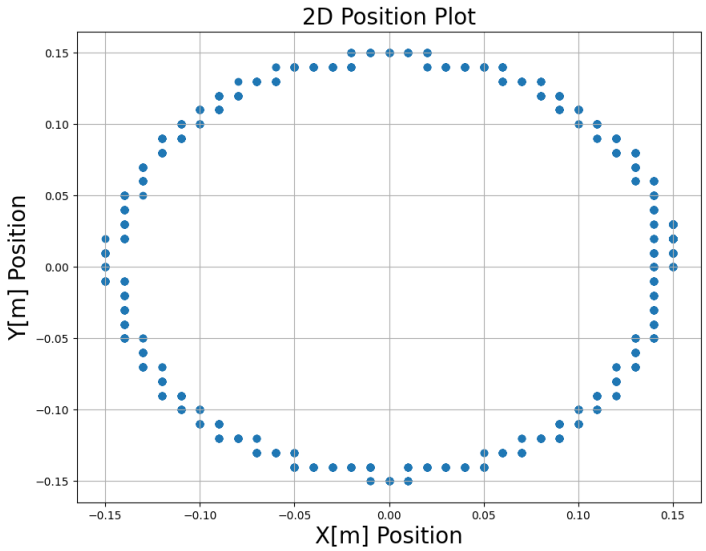

Fig 7. the measured actual position of the robot while following the Circle path, Serial output.

Again, we see that the robot followed the desired path perfectly.

The shape drawn in lab:

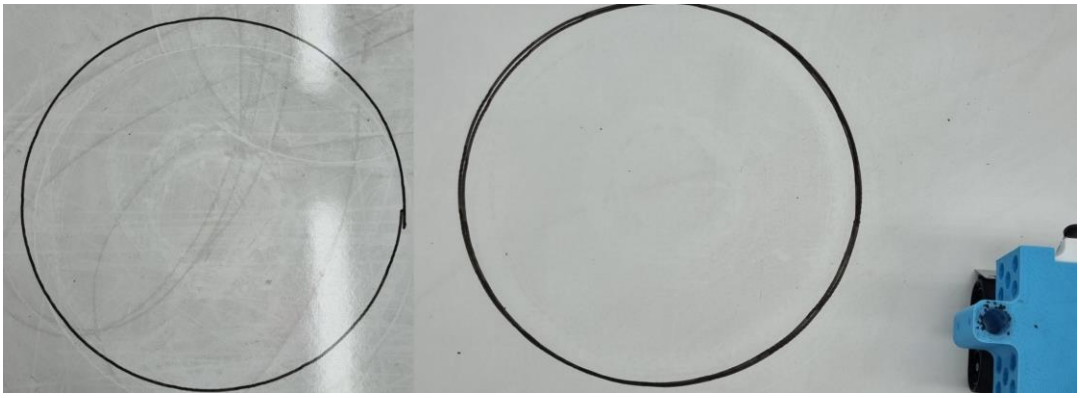

Fig 8. the shape drawn in lab by the robot, while following a Circular shape [one - left] and multiple circles [right]

The robot followed the path perfectly, even after letting it complete multiple circles we observed a very precise and consistent movement.

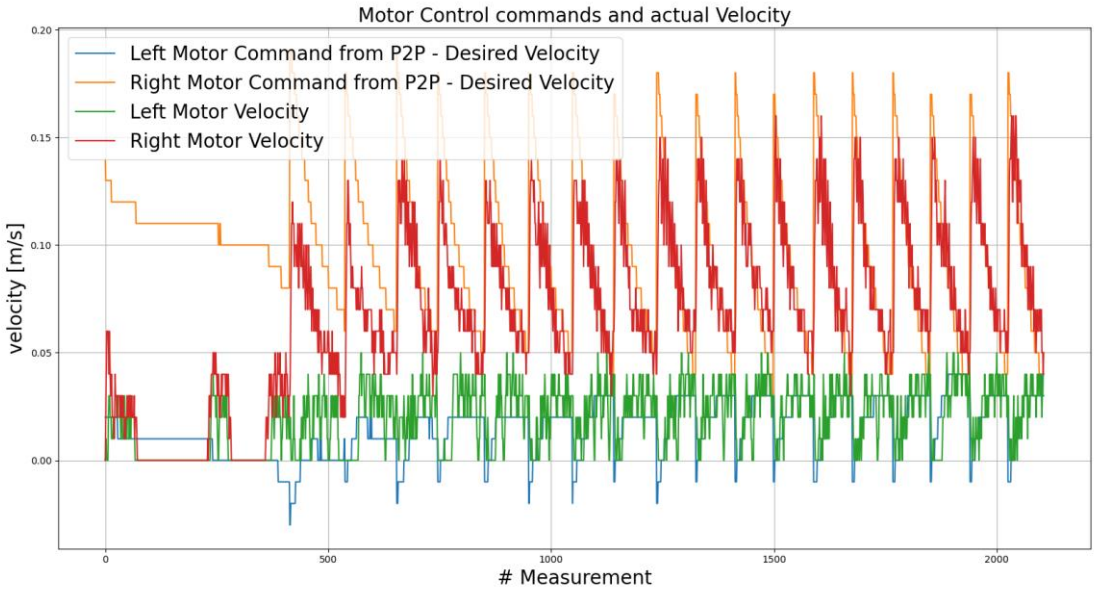

Fig 9. the Desired Velocity and Actual velocity of each motor, during the Circular movement

From this graph we see that for the entire movement of the robot the left motor was slower than the right since the robot moved in a circular path. Furthermore, we observe that mainly the velocity control loop worked as expected.

Square shape

For the robot to draw a square shape, we need to augment a square, in this case we can't use a velocity vector as we did with the infinity shape, since the edges won't result in a smooth path for the robot to follow,

We instead make it by hand, concatenating ¼ of a circle that originates at the square's corner, and has a radius of the pen's offset (65mm) (Python Code is available in the google-collab notebook):

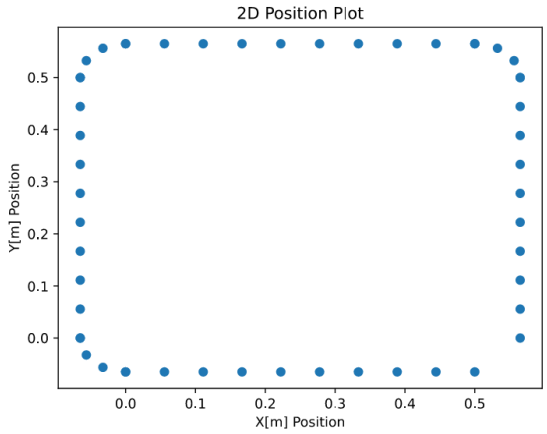

Fig 10. the desired square path fed to the robot.

Lab results [square]:

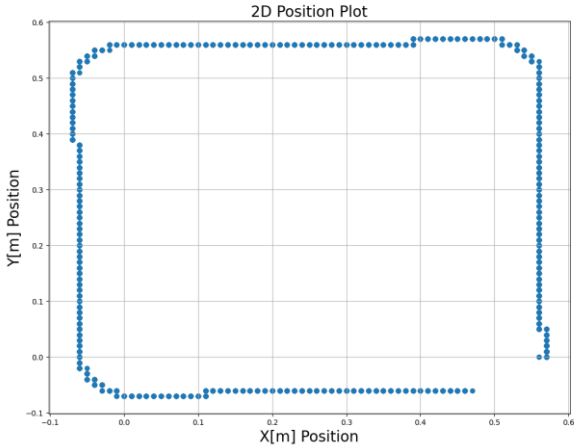

Fig 11. the measured actual position of the robot while following the square path, Serial output.

The shape drawn in lab:

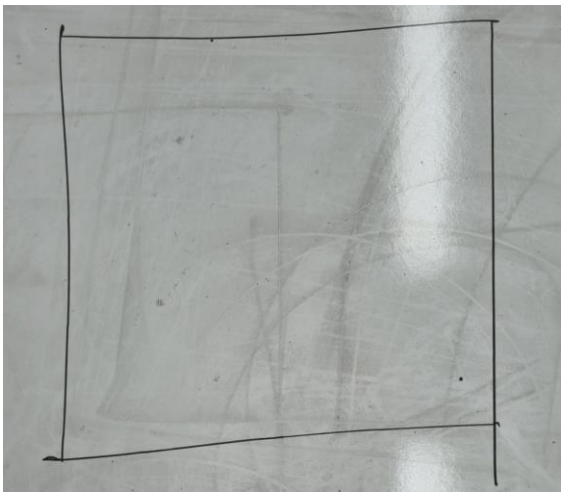

Fig 12. the shape drawn in lab by the robot, while following the Circle shape

We observe that the robot follows the path almost perfectly from the odometry calculations, however, due to slip \ imperfect point of rotation, the actual shape that was drawn is not a perfect square.

Further optimization of the mechanical model \ tweaking of θ_d can yield better results.

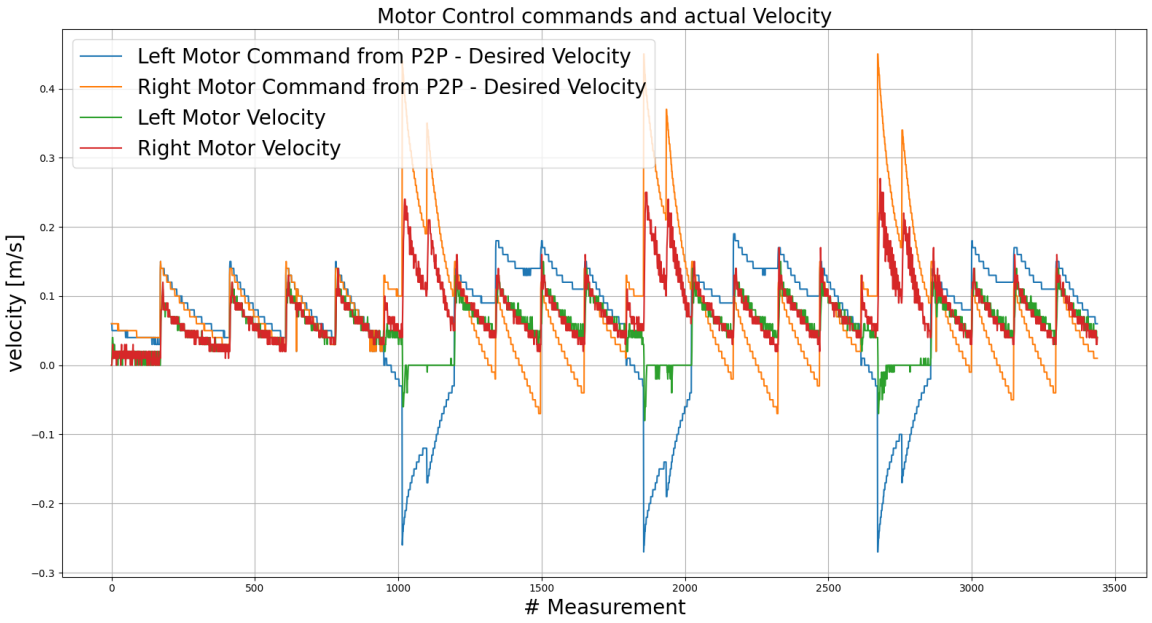

Fig 13. the Desired Velocity and Actual velocity of each motor, during the square movement

From this graph we can notice how the motors receive [almost] the same control command when the robot should drive in a straight line. During the turn, the Right motor gets a higher speed command which results in the expected left turn.

Appendix:

Google-Collab notebook that generates the desired paths:
https://colab.research.google.com/drive/1sQEZoUYBs30euKWlNJecnE6vXSFWy3KO?usp=sharing

Python Code that generates the graphs from the data files:

https://colab.research.google.com/drive/1ceWnmnAbsiNupfmsnECpXUFy7KOXYHxP?usp=sharing

Videos for Infinity and Square

Square - https://youtu.be/mGz0eLYc4kY

Infinity - https://youtube.com/shorts/g-gz1xGRCvc?feature=share

Infinity 2 - https://youtube.com/shorts/ctN1uOwMR9I?feature=share