**Path Control –**

**Autonomous Systems lab**

**מוגש על ידי:**

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

**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.

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

Note – different values of 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 is done by reading the and resetting it to 0, transforming the measurement to meters, and calculating by

The position is then updated:

We tweaked the value of 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 – 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:

Where are from the desired path (we explain in further detail on the next pages)

And are the position calculations from the Odometry procedure.

We then calculate the , and the direction value

We get the by

We get

We get the trapezoid velocity profile (by limiting , and the increase of )

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

**4. Main flow – Block diagram**

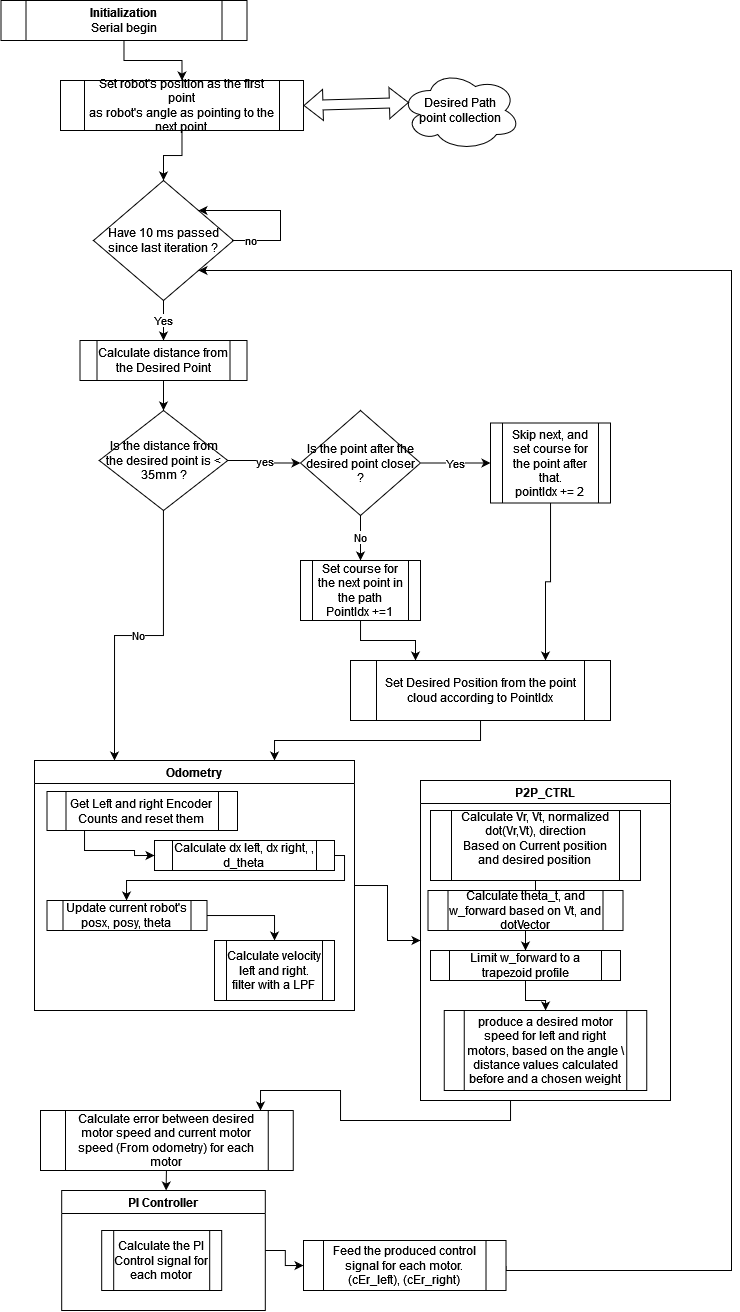


Fig. A block diagram of the path control program

**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:

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

To achieve this, we calculate the velocity vector:

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

We then calculate the cross product to get a vector perpendicular to the curve.

Finally, the point we want is given by

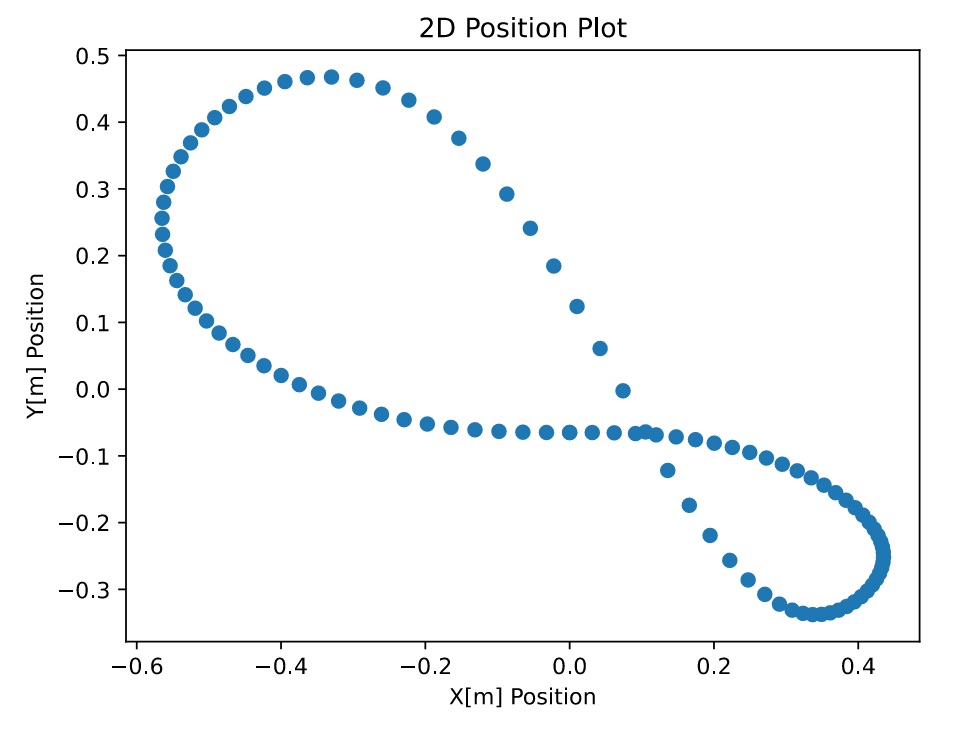


Fig. Augmented Infinity shape, that a robot can follow and draw a correct infinity.

Lab results:

Measured position (from serial):

A picture containing line, plot, diagram, screenshot

Description automatically generated

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

We see that the robot followed the desired path almost perfectly.

The shape drawn in lab:

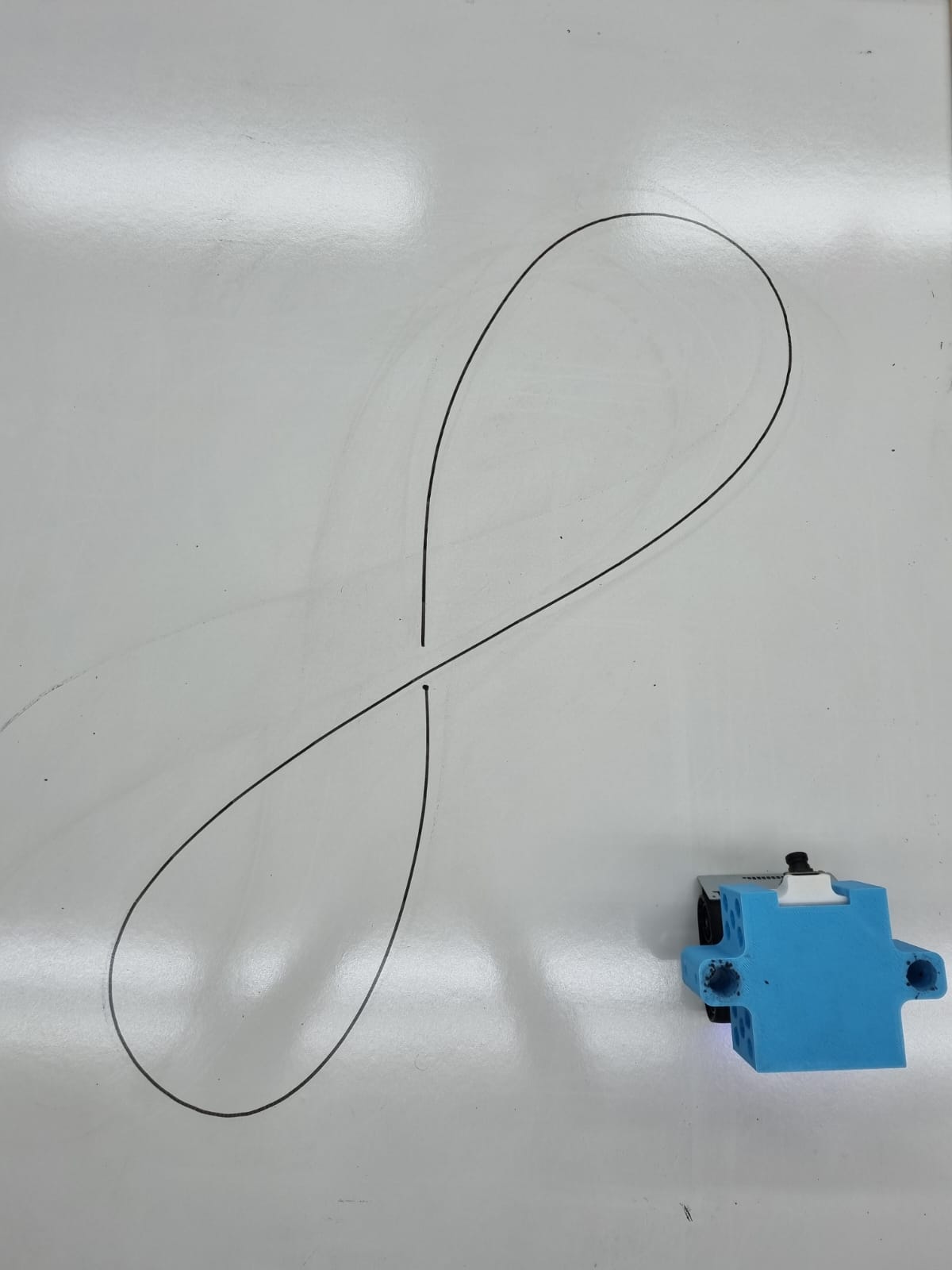


Fig . the Infinity shape drawn in lab by the robot while following the augmented infinity shape.

**Circle Shape**

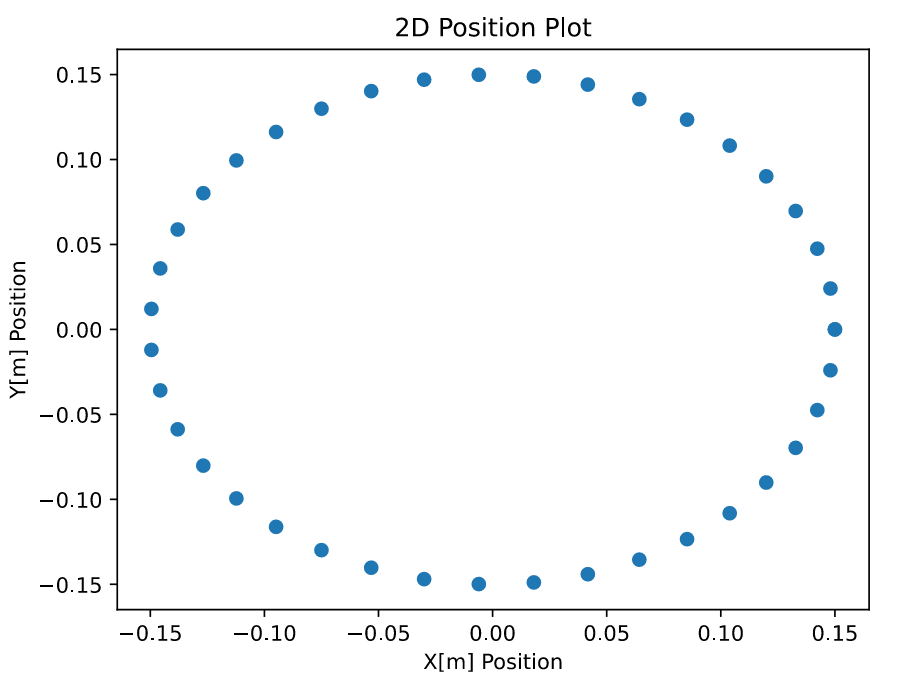


Fig . the desired Circle path fed to the robot.

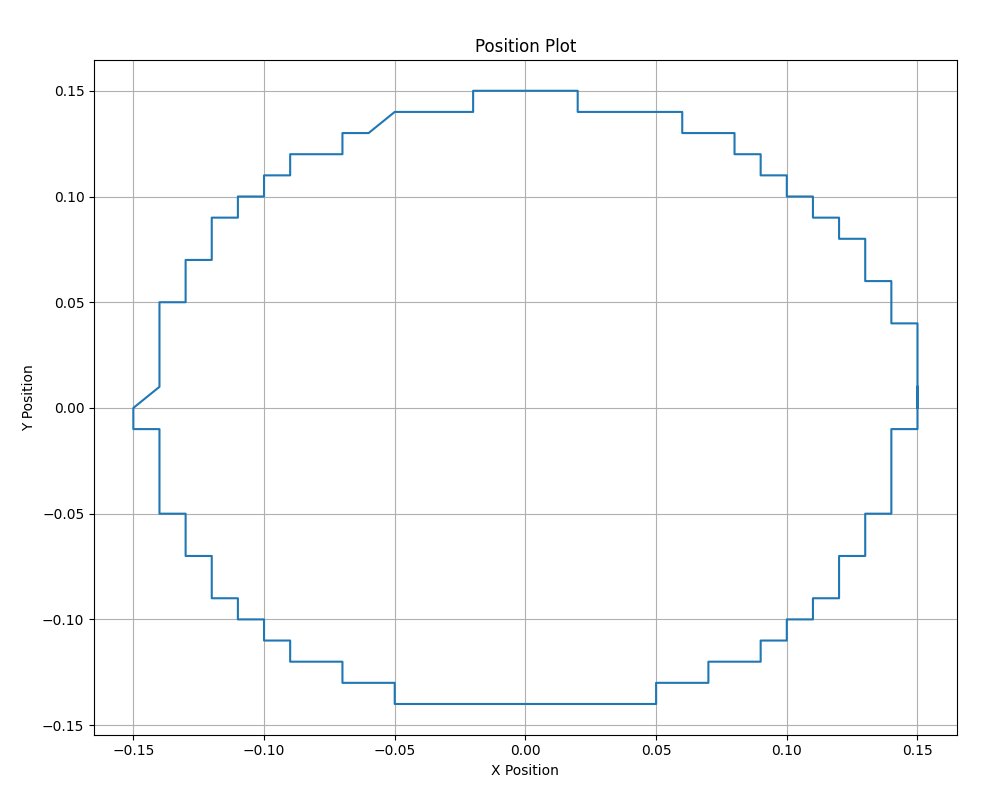


Fig . the measured actual position of the robot while following the Circle path, Serial output.

The shape drawn in lab:

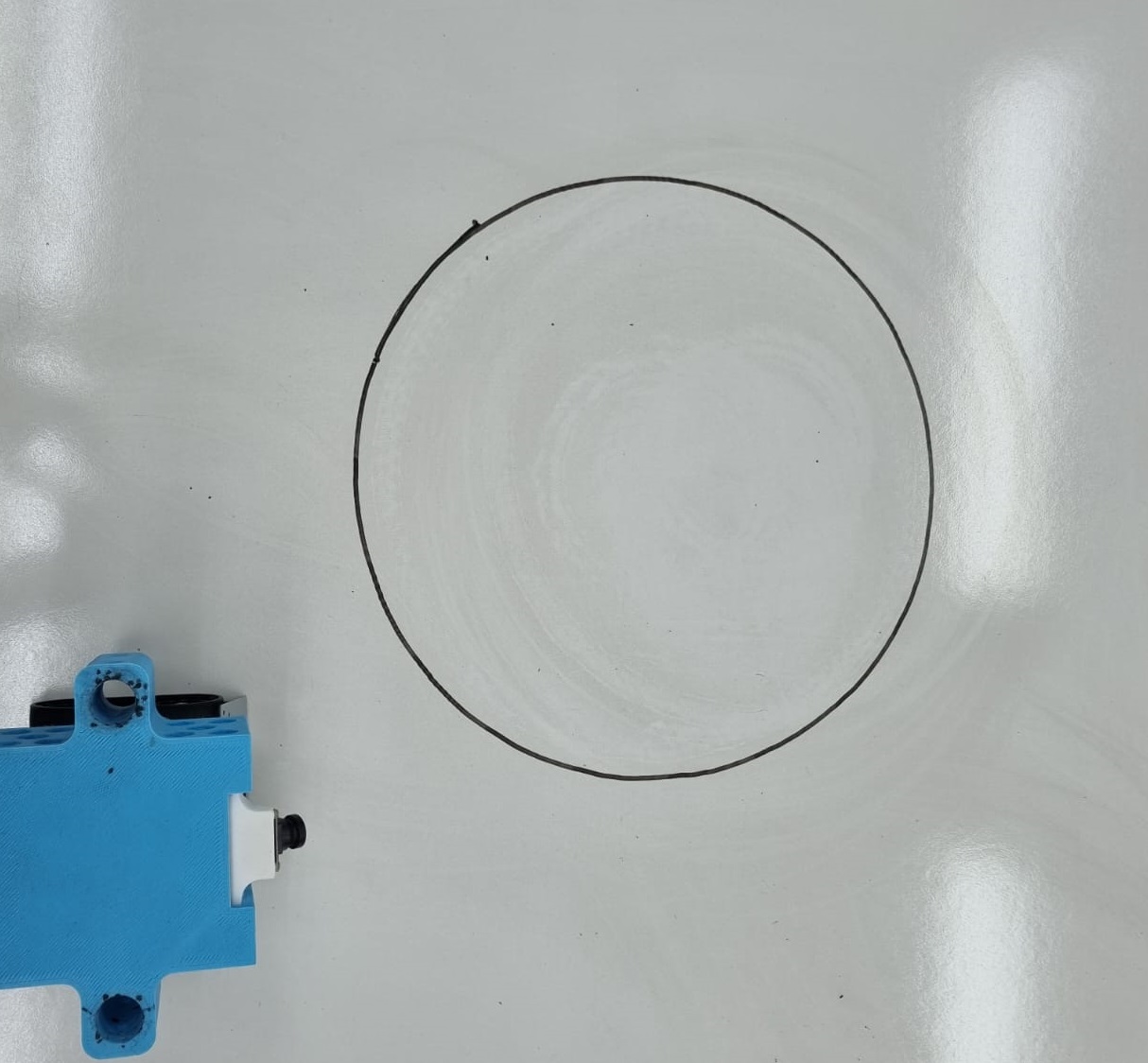


Fig. the shape drawn in lab by the robot, while following the Circle shape

The robot followed the path perfectly.

Letting the robot complete multiple circles without stopping it results in the following drawing:

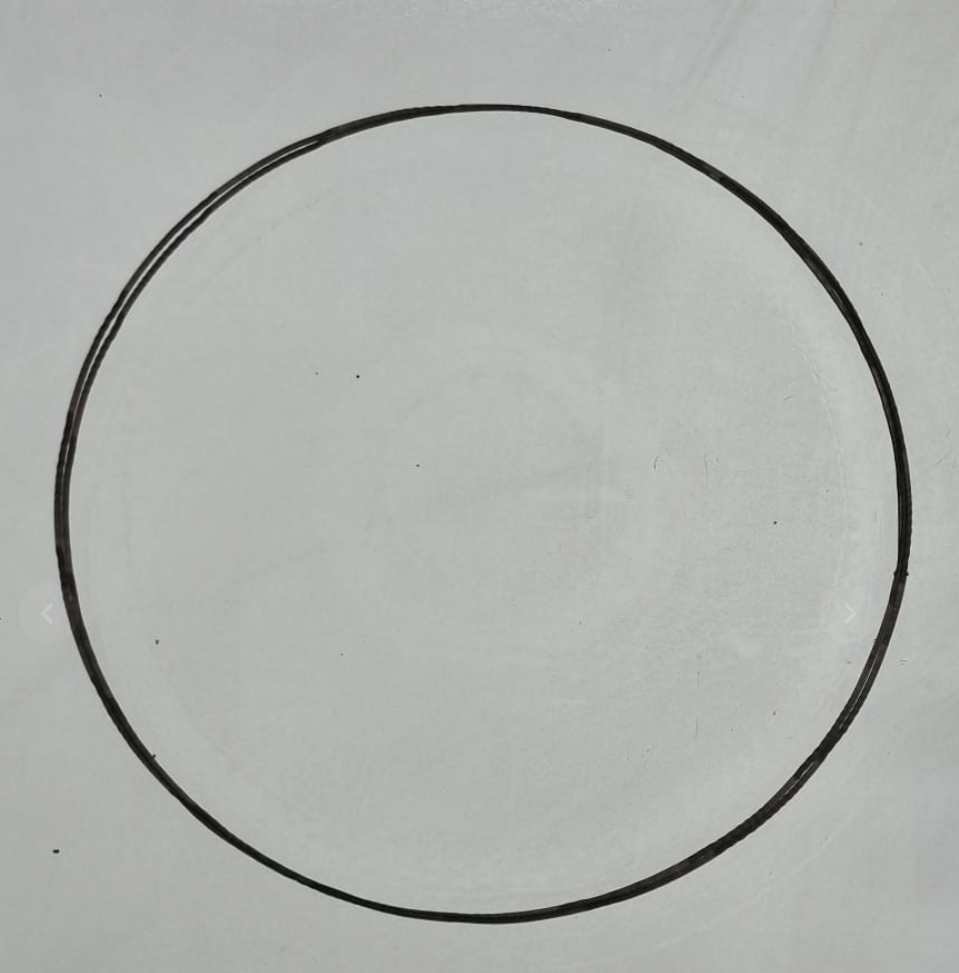


Fig. Multiple circles drawn by the robot.

We observe a very precise and consistent movement.

**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):

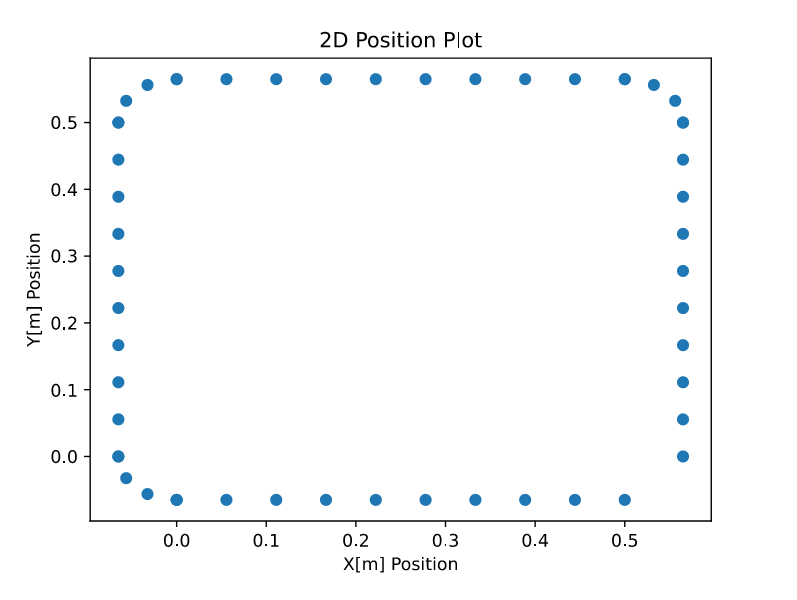


Fig . the desired square path fed to the robot.

**NEED TO REDO**

A picture containing text, line, diagram, plot

Description automatically generated

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

**NEED TO REDO**

The shape drawn in lab:

A black line on a white surface

Description automatically generated with medium confidence

Fig. 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, 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 can yield better results.

**Appendix:**

Google-Colab notebook that generates the desired paths:

<https://colab.research.google.com/drive/1sQEZoUYBs30euKWlNJecnE6vXSFWy3KO#scrollTo=RC3hxT0_FO_Z>