2.2 Sensor frequencies and delays (25 points) (Nusantara)

* (a)  For the data sets obtained in 1.5(a)–(c) observe and compare the timestamps of the data. What do you notice? how often can you get the different sensor information? how does this frequency depend on your code?
* (b)  What are the implications of these delays on your control code and the robot behavior? how might you mitigate any unwanted behaviors?
* (c)  For the dataset in 1.5(a) what do you observe about the timing of the depth and tag information with respect to the other sensors? When doing localization (as you will do in Lab 2), how should you deal with the delay in these measurements?
* (d)  For the datasets in 1.5(d), how does the number of tags affect the delays?
* (e)  In Part 1.6(e)–(f), did the robot react differently to the presence of a wall? How so?  
  Yes, the robot reacted dif
* (f)  If so, what is the reason for the different behavior? If the behavior was identical, why would one expect commenting out parts of the code would lead to different behaviors?

2.4 Waypoint Follower (25 points) (Nusantara)

* (a)  Plot all the robot trajectories from Part 1.7 on the same figure (you should have one for each member of the group) and indicate which ε was used by each robot for feedback linearization.
* (b)  Plot the waypoints on the same figure.Chart

  Description automatically generated
* (c)  Comment on any differences between the robots’ trajectories. Why wouldn’t the trajectories be exactly the same? Did members of the group handle this task differently?

The blue plot has a smoother trajectory between the waypoints. The trajectory is more or less pretty straight, with some turns as it moves from one point to another. On the other hand, the red plot has a rougher trajectory. We can see how it has wider turns, which we can say is a less efficient path (considering the most efficient path is a straight line from one point to another). In some cases, due to the wide turn, it needed to back-up at point (-1.05, -0.97) before moving to the next destination. Due to the wide turn angle, it cannot turn in time, requiring it to back up.

The trajectories are different because they have a different epsilon. The epsilon is a value that would determine the robot’s turning radius. The blue plot has an epsilon of 0.2, while the red plot has an epsilon of 0.7. Since the red plot has a bigger epsilon. It explains the wider turn angle seen in the plot.

Both members handle this task similarly, as a result the only cause to this difference in trajectory is the epsilon, and not the program itself.

* (d)  How did the value of ε for feedback linearization affect the robots’ trajectories? How would the trajectory have been different for a holonomic vehicle?  
  In feedback linearization, epsilon is used as an imaginary x-coordinate point in the body frame, which will then be multiplied with the angular velocity in order to move sideways (y axis body coordinate). This is needed because the robot is non-holonomic, so it cannot move along the body y-axis freely. The bigger the epsilon, the bigger the turning radius would be. This is also seen in the plot in our homework. As a result, we can see in our plot above that when the epsilon is smaller, the trajectory is straighter from one point to another. When the epsilon is larger, the robot has a wider turns during rotations and movements between points. This also result to a less smooth trajectory.  
  A holonomic vehicle is free to move anywhere (can move sideways). As a result, it does not need an epsilon value. The trajectory will be a straight line from one point to another, as the movement is not constraint.
* (e)  For one of the ε values, run the same waypoints and the same code in the simulator. Plot the resulting (simulated) trajectory. On the same figure, plot the actual trajectory. Do you expect both trajectories to be the same? Are they? If they are not the same, explain what the possible sources of difference are.   
  Chart, line chart

  Description automatically generated  
  The blue plot above shows the real trajectory of the robot with eps = 0.2, while the red plot is the simulated trajectory with the same eps value.We can see a difference between the two, where the red plot has a smoother trajectory through the waypoints. The simulated trajectory also passes through two of the waypoints right on the dot. One possible source of error would be the sensor noise in the real world. Due to the sensor noise, the robot has a less accurate information of where it is currently at. As a result, it may overshoot a bit (traveling further than required) as it passes the points. This can be seen as it passes the second and third point, where it travels more than required and had to perform a shaper turn to go to the next waypoint. Additionally, the error is due to a difference in the time delay between truthPose measurements. Looking at the dataStore.truthPose data, the real robot recorded measurements at an average of 0.9 second, while the simulated robot recorded measurements at every 0.1 second. This also helps explain why the real robot turned a bit later compared to the simulated robot.