Program Description:

Initially, I started with the line follower program created by myself previously. This approach appeared to be the simplest way to begin, offering a solid foundation for the project. It allowed the robot to follow a line, and I planned to expand on it by incorporating object detection and avoidance capabilities.

The initial concept involved using proximity sensors to detect obstacles in front of the robot. By utilizing a statistical function, I determined the maximum value among the four sensors. This maximum value was then compared to a predefined threshold. If the maximum value exceeded this threshold, indicating an obstruction, the robot transitioned into a state called "BLOCKED."

To determine the appropriate turning direction, I decided that if the obstacle was closer to the left of the robot, it would execute a curved turn in that direction. The same logic applied if the obstacle was closer to the right. However, this straightforward approach encountered issues. While it worked for some obstacles, it often led the robot to collide with them. Particularly, block #2 posed a risk of falling off the table.

To address these problems, I introduced a more advanced strategy. If the obstacle was on the left side, the robot would pivot on its axis in the opposite direction until the obstacle was out of view of all five sensors. This would be followed by a curved turn in the same direction. The same process applied if the obstacle was on the right side. This solution aimed to ensure that the robot avoided contact with the obstacle, provided the curved turn radius was adequate.

However, this approach also encountered challenges. After implementing the code, the robot often failed to reacquire the line and would cross over it. This issue stemmed from conflicting states. To resolve this, I introduced new states named "RIGHT1" and "LEFT1" to differentiate them from the standard "RIGHT" and "LEFT" states. Additionally, I added "TURNING" and "TURNING1" states for turns.

Another challenge emerged with certain obstacles still being hit during turns. To address this, I introduced an if statement within the turning state. If the obstacle was detected by the farthest right or left sensor (depending on the turn direction), the robot would make another adjustment turn toward that direction. It would continue until the obstacle was completely out of sensor range and then proceed with the curved turn, successfully reestablishing its line-following behavior.

During programming, I aimed for a target speed of 225, targeting a completion time of around one minute. All testing was conducted in a simulation environment.

In summary, a key strength of this solution was its adaptability, particularly evident in the obstacle adjustment during turns. However, a significant weakness was its simplicity. If I had implemented varying turn speeds based on obstacle position, the solution could have been more effective in avoiding collisions.