Robotics System

To begin the Turtlebot will drive forwards until the laser scan has a reading from one of three variables called middle, left and right. These variables are the left, middle and right of what the laser scanner can read, this is used so that the Turtlebot can keep a far enough distance away from the walls so that it does not crash into them. There is no specific algorithm that has been chosen to escape the maze, the method employed is random wandering until the Turtlebot either finds a trap or the exit.

There are four functions, the initialiser, laser callback, image callback and movement. The initialiser creates a publisher to send twist messages to twist, a subscriber to the laser scan to use for the movement function. In the initialiser there is also a subscriber to the camera and a cv bridge. In the laser callback the data is split into left, middle and right and then passed to the movement function. In the image callback the data is shown, and two masks are made to find red and green. These are aimed at the floor since this is where the traps and goal are. In the movement function there is the main logic for the Turtlebot, this consists of lots of if statements to make sure that the Turtlebot does not get stuck in dead ends or gets too close to the wall.

The Turtlebot decides which way to turn by choosing from left and right, whichever of the values is higher it chooses that direction as there will be more space to go into. If there is more space this means that the path continues so it is not at a dead end. However, if the Turtlebot does come across a dead end it will drive around in a loop to then exit the dead end. This will continue until colours are found.

When the colour red is seen on the floor the Turtlebot will turn around and go through the maze, this could be revisited later if the Turtlebot goes back on itself. Once the Turtlebot see the colour green on the floor it will speed up and drive towards the exit.

On reflection, an improvement for this solution would be to add more logic to the movement function so that the Turtlebot is less likely to get stuck. This could be completed by filtering out nan values and ensuring that the distance to the wall does not go below the lowest readable value. Furthermore, the use of the blue squares to help the Turtlebot find the exit could be employed so that the Turtlebot gets closer to the exit.