Robot Explanation:

The main loop of the program is the forward function which drives the robot forward until the class variable stop is true, to avoid crashing into walls the laser scanner is used and to assess the environment the rgb camera.

3 readings from readings from the laser scanner are used to detect distance between the robot and nearby obstacles, forward, left most and right most. whilst the forward scanner value exceeds 1 the robot will continuously drive forward, adjacent obstacles will decrease the values of the left and right scanner, when these values reach 0.7 or less the robot will adjust its angular twist value to avoid the obstacle and center itself. When the forward sensor reaches 1 or less the robot will stop and then turn towards the largest sensor value as it looks for space(left/right) this works well to navigate corners. Notice a “flip value” will appear in the terminal when this happens, this is the flip counter which tracks how many times the turn conditions have been met. In the case of a dead end the left and right sensors will constantly fluctuate between who's larger causes the robot to turn unendingly. To avoid this when the flip counter exceeds 12(corners generally take less than 10 to navigate) the robot will perform a 180 degree turn regardless of sensor data to leave the dead end.

To detect traps and the goal the rgb camera is used to assess the environment, by creating a mask to isolate the desired colour and using the cv2 moment property that refers to pixel density of image to detect how many pixels In the mask refer to the desired colour I was able to create thresholds within the forward function that trigger specific behaviour. During development I noticed that these values grow quickly in a head on approach and slowly when approaching from an angle, to minimise the detection distance and keep thresholds to a closer effective range (values could range: 9056385- 1200000= 7856385 it was detecting from far away on forward approach and not at all on angled) the masks 0s out pixel values of the top ¾ of the screen and some of the bottom to create a window that can see approximately 1-2 squares a head of its self.

The red specific behaviour causes the robot to perform a 180 degree turn and the detection of green drives the robot forward a predetermined amount and then to changes a class variable stop to true exiting the loop that drives the robot ending the program

Issues:

If the robot enters a corridor where there is a red at both ends such as in maze one on the far left, it will become stuck, as despite the window approach on the masks the robot cannot get close enough to escape as the behaviour triggered does not allow it to look for alternative routes only back track. I was also not able to implement any maze solving algorithm in this solution, the simplest solution would be the well-known left hand rule by introducing a bias in the corner detection for instance when the values of left and right are in excess(above 1 ) have increase the angular variable to the left. This would be too simple for these mazes as this method does not work with islands of which there are a few (walls not attached to the edge of the maze) instead the Pledge method would have been more applicable allowing it to switch walls when a loop has been detected. However I found this difficult to do with the corner navigation I used.