201477380 OPE OMIWADE

I have chosen to a use a waypoint navigation approach coupled with a state machine which changes the MoveState of the robot when it is approaching waypoints. A method called state machine has been written in the navigation class, this method changes the state of the robot appropriately, when the robot reaches the various waypoints in the map. On reaching a way point different variables are set and reset before the state is changed and actions are performed. Actions are performed in the controller class; the actions are executed by checking the current MoveState using the getter in the navigation class. If the robot is then at the state needed for an action to be performed, the robot will then execute the action.

A method called atDestination has been created in the navigation class to help the robot recognise when it is approaching waypoints. The method uses the currentPose and destination pose values as parameters, calculates the Euclidean distance between these two points until it is close to zero; at this point the Boolean value is changed to true indicating the robot is approaching the waypoint and vice versa. I have also written a method to calculate the angle for rotation of the robot; the method calculates the angle for rotation using odometry. Calculates rot1 using atan2 and the current theta , calculates rot2 using the current theta, destination theta and rot2.The method then checks how close the value is to the destination theta by checking the difference between the two values.

The last helper method atTheta was written for arcing motions. This method uses the current and destination theta of the robot and checks the difference between these two values. When the difference is approaching zero the robot arc motion returns a Boolean value of true.

In the state machine method, the robot is set to move forward initially and then when a waypoint is reached different variables for arcing or rotation are initialised. If the robot is rotating at a waypoint; the icr_r has a default value of 0.0 and the icr_angle is determined using the calculateAngleForRotation method which I explained earlier, the angular speed also has a default value which is a percentage of the robotLinearVelocity variable that has been declared and initialised in the controller class. The state is the changed to ROTATE and the arc method is called, the target time returned is then used to determine when to terminate the rotation to prevent the robot from being

stuck in a loop. When the target time is exceeded, the stopRotation Boolean value is set to true and the if statement in the control loop is exited.

On the other hand, if the robot is arcing at a certain waypoint before the state is changed to ARC in the state machine, the destination theta of the robot is initialised as this value is used by the atTheta method and the direction of the arc(left or right) is stated using a Boolean value called leftArc. If it is a left arc the icr radius and speed are positive values and vice versa , the default values for the icr radius is set to 0.5 but this can be changed and has been changed for certain waypoints as the robot will need to arc with a smaller radius to prevent collisions with obstacles and walls. The arc method is terminated using the atTheta method, when the method returns a true value the arc motion is terminated by setting the stopRotation value to true and some other Boolean variables are reset so they can be used for the next waypoint if necessary.

There are 10 waypoints. When the robot reaches the waypoint next to the triangular obstacle , the state of the robot is changed to wander to allow the robot wander around the obstacle to get good coverage of the triangle. I have done this as when I was initially arcing around the robot, I couldn't get good coverage of that are so instead the robots roams the triangle for a set amount of time and when this time is exceeded , a variable called stopWandering is set to true which then allows the robot to complete its path. The wander method also helps the robot to avoid edges if the triangle to prevent collision.

I also noticed that the percentage of coverage for my map is lower on the simulation that happens when the project is opened but slightly higher on every other simulation after compiling.