# EECS 4421 LAB1

## Name: Mahfuz Rahman

## Student No.: 217847518

### A computer screen shot of a computer Description automatically generatedInitial State of robot:

### A screenshot of a computer Description automatically generatedState of robot after executing code:

# drive\_robot.launch.py

**import** os

**from** ament\_index\_python.packages **import** get\_package\_share\_directory

**from** launch **import** LaunchDescription

**from** launch.actions **import** DeclareLaunchArgument, IncludeLaunchDescription

**from** launch.launch\_description\_sources **import** PythonLaunchDescriptionSource

**from** launch.substitutions **import** LaunchConfiguration

**from** launch\_ros.actions **import** Node

**def** generate\_launch\_description():

**return** LaunchDescription([

DeclareLaunchArgument('goal\_x', default\_value = '0.0', description = 'goal (x)'),

DeclareLaunchArgument('goal\_y', default\_value = '0.0', description = 'goal (y)'),

DeclareLaunchArgument('goal\_t', default\_value = '0.0', description = 'goal (t)'),

#Code added by Mahfuz Rahman

DeclareLaunchArgument('max\_vel', default\_value = '0.2', description = 'max\_vel (vel)'),

DeclareLaunchArgument('vel\_gain', default\_value = '5.0', description = 'vel\_gain (gain)'),

Node(

package = 'cpmr\_ch2',

executable = 'drive\_to\_goal',

name = 'drive\_to\_goal',

parameters = [

{'goal\_x' : LaunchConfiguration('goal\_x')},

{'goal\_y' : LaunchConfiguration('goal\_y')},

{'goal\_t' : LaunchConfiguration('goal\_t')},

],

),

])

# drive\_to\_goal.py

**import** math

**import** numpy **as** np

**import** rclpy

**from** rclpy.node **import** Node

**from** rclpy.parameter **import** Parameter

**from** rcl\_interfaces.msg **import** SetParametersResult

**from** nav\_msgs.msg **import** Odometry

**from** geometry\_msgs.msg **import** Twist, Pose, Point, Quaternion

**from** nav\_msgs.msg **import** Odometry

**def** euler\_from\_quaternion(quaternion):

"""

Converts quaternion (w in last place) to euler roll, pitch, yaw

quaternion = [x, y, z, w]

"""

x = quaternion.x

y = quaternion.y

z = quaternion.z

w = quaternion.w

sinr\_cosp = 2 \* (w \* x + y \* z)

cosr\_cosp = 1 - 2 \* (x \* x + y \* y)

roll = np.arctan2(sinr\_cosp, cosr\_cosp)

sinp = 2 \* (w \* y - z \* x)

pitch = np.arcsin(sinp)

siny\_cosp = 2 \* (w \* z + x \* y)

cosy\_cosp = 1 - 2 \* (y \* y + z \* z)

yaw = np.arctan2(siny\_cosp, cosy\_cosp)

**return** roll, pitch, yaw

**class** MoveToGoal(Node):

**def** \_\_init\_\_(self):

super().\_\_init\_\_('move\_robot\_to\_goal')

self.get\_logger().info(f'{self.get\_name()} created')

self.declare\_parameter('goal\_x', 0.0)

self.\_goal\_x = self.get\_parameter('goal\_x').get\_parameter\_value().double\_value

self.declare\_parameter('goal\_y', 0.0)

self.\_goal\_y = self.get\_parameter('goal\_y').get\_parameter\_value().double\_value

self.declare\_parameter('goal\_t', 0.0)

self.\_goal\_t = self.get\_parameter('goal\_t').get\_parameter\_value().double\_value

self.add\_on\_set\_parameters\_callback(self.parameter\_callback)

self.get\_logger().info(f"initial goal {self.\_goal\_x} {self.\_goal\_y} {self.\_goal\_t}")

#Code added by Mahfuz Rahman

self.declare\_parameter('max\_vel', 0.2)

self.declare\_parameter('vel\_gain', 5.0)

self.\_max\_vel = self.get\_parameter('max\_vel').get\_parameter\_value().double\_value

self.\_vel\_gain = self.get\_parameter('vel\_gain').get\_parameter\_value().double\_value

########################

self.\_subscriber = self.create\_subscription(Odometry, "/odom", self.\_listener\_callback, 1)

self.\_publisher = self.create\_publisher(Twist, "/cmd\_vel", 1)

**def** \_listener\_callback(self, msg, max\_pos\_err=0.05):

pose = msg.pose.pose

cur\_x = pose.position.x

cur\_y = pose.position.y

o = pose.orientation

roll, pitchc, yaw = euler\_from\_quaternion(o)

cur\_t = yaw

x\_diff = self.\_goal\_x - cur\_x

y\_diff = self.\_goal\_y - cur\_y

dist = math.sqrt(x\_diff \* x\_diff + y\_diff \* y\_diff)

twist = Twist()

**if** dist > max\_pos\_err:

x = max(min(x\_diff \* self.\_vel\_gain, self.\_max\_vel), -self.\_max\_vel)

y = max(min(y\_diff \* self.\_vel\_gain, self.\_max\_vel), -self.\_max\_vel)

twist.linear.x = x \* math.cos(cur\_t) + y \* math.sin(cur\_t)

twist.linear.y = -x \* math.sin(cur\_t) + y \* math.cos(cur\_t)

self.get\_logger().info(f"at ({cur\_x},{cur\_y},{cur\_t}) goal ({self.\_goal\_x},{self.\_goal\_y},{self.\_goal\_t})")

self.\_publisher.publish(twist)

**def** parameter\_callback(self, params):

self.get\_logger().info(f'move\_robot\_to\_goal parameter callback {params}')

**for** param **in** params:

self.get\_logger().info(f'move\_robot\_to\_goal processing {param.name}')

**if** param.name == 'goal\_x' **and** param.type\_ == Parameter.Type.DOUBLE:

self.\_goal\_x = param.value

**elif** param.name == 'goal\_y' **and** param.type\_ == Parameter.Type.DOUBLE:

self.\_goal\_y = param.value

**elif** param.name == 'goal\_t' **and** param.type\_ == Parameter.Type.DOUBLE:

self.\_goal\_t = param.value

########

**elif** param.name == 'max\_vel' **and** param.type\_ == Parameter.Type.DOUBLE:

self.\_max\_vel = param.value

**elif** param.name == 'vel\_gain' **and** param.type\_ == Parameter.Type.DOUBLE:

self.\_vel\_gain = param.value

#######

**else**:

self.get\_logger().warn(f'{self.get\_name()} Invalid parameter {param.name}')

**return** SetParametersResult(successful=**False**)

self.get\_logger().warn(f"Changing goal {self.\_goal\_x} {self.\_goal\_y} {self.\_goal\_t}")

**return** SetParametersResult(successful=**True**)

**def** main(args=**None**):

rclpy.init(args=args)

node = MoveToGoal()

**try**:

rclpy.spin(node)

**except** KeyboardInterrupt:

**pass**

rclpy.shutdown()

**if** \_\_name\_\_ == '\_\_main\_\_':

main()