Lab Assignment: Week2

Pranav Deo 170040012

Hitvarth Diwanji 190100057

General Information:

- Language Used: Python
- Launch file: week2.launch
- Libraries used: rospy, catkin, math, matplotlib, geometry_msgs, tf

Nodes Information:

```
hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$ rosnode list
/A
/B
/C
/gazebo
/gazebo_gui
/mobile_base_nodelet_manager
/robot_state_publisher
/rosout
hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$ rostopic list
/clock
/error
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/gazebo_gui/parameter_descriptions
/gazebo_gui/parameter_updates
/joint_states
/mobile_base/commands/motor_power
/mobile_base/commands/reset_odometry
/mobile_base/commands/velocity
/mobile_base/events/bumper
/mobile_base/events/cliff
/mobile_base/sensors/imu_data
/odom
/rosout
/rosout_agg
/tf
/tf_static
/waypoint
hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$
```

```
hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$ rosnode info A
--------------------------------------------------------------------------------
Node [/A]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /waypoint [geometry_msgs/Point]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]
 * /odom [nav_msgs/Odometry]

Services:
 * /A/get_loggers
 * /A/set_logger_level


contacting node http://hitvarth-Inspiron-5482:40955/ ...
Pid: 15343
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /waypoint
    * to: /B
    * direction: outbound
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo (http://hitvarth-Inspiron-5482:44933/)
    * direction: inbound
    * transport: TCPROS
 * topic: /odom
    * to: /gazebo (http://hitvarth-Inspiron-5482:44933/)
    * direction: inbound
    * transport: TCPROS

hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$
```

```
hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$ rosnode info B
--------------------------------------------------------------------------------
Node [/B]
Publications:
 * /error [week2_170040012_190100057/Error]
 * /rosout [rosgraph_msgs/Log]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]
 * /odom [nav_msgs/Odometry]
 * /waypoint [geometry_msgs/Point]

Services:
 * /B/get_loggers
 * /B/set_logger_level


contacting node http://hitvarth-Inspiron-5482:41663/ ...
Pid: 15351
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /error
    * to: /C
    * direction: outbound
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo (http://hitvarth-Inspiron-5482:44933/)
    * direction: inbound
    * transport: TCPROS
 * topic: /odom
    * to: /gazebo (http://hitvarth-Inspiron-5482:44933/)
    * direction: inbound
    * transport: TCPROS
 * topic: /waypoint
    * to: /A (http://hitvarth-Inspiron-5482:40955/)
    * direction: inbound
    * transport: TCPROS
```

```
hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$ rosnode info C
--------------------------------------------------------------------------------
Node [/C]
Publications:
 * /mobile_base/commands/velocity [geometry_msgs/Twist]
 * /rosout [rosgraph_msgs/Log]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]
 * /error [week2_170040012_190100057/Error]

Services:
 * /C/get_loggers
 * /C/set_logger_level


contacting node http://hitvarth-Inspiron-5482:37093/ ...
Pid: 16752
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /mobile_base/commands/velocity
    * to: /gazebo
    * direction: outbound
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo (http://hitvarth-Inspiron-5482:41283/)
    * direction: inbound
    * transport: TCPROS
 * topic: /error
    * to: /B (http://hitvarth-Inspiron-5482:39749/)
    * direction: inbound
    * transport: TCPROS

hitvarth@hitvarth-Inspiron-5482:~/SC635_ws$
```

Codes:

A.py

```
#!/usr/bin/env python

import rospy
import random
import math
import matplotlib.pyplot as plt
from nav_msgs.msg import Odometry
from geometry_msgs.msg import Point
from tf.transformations import euler_from_quaternion

t = -17   # so that it starts from the point next to the origin in the fourth quadrant
K = 5       # increment in degrees

current_x=0.0
current_y=0.0
threshold = 0.07

waypoint_pub=rospy.Publisher('/waypoint',Point,queue_size=10)

def waypoint_publisher():
            global current_x, current_y, current_theta, t
            x = 4*math.cos(t*K*math.pi/180)
            y = 4*math.sin(2*t*K*math.pi/180)
            if abs(current_x - x)<=threshold and abs(current_y - y)<=threshold:
            t+=1
            x = 4*math.cos(t*K*math.pi/180)
            y = 4*math.sin(2*t*K*math.pi/180)
            p=Point()
            p.x=4*math.cos(t*K*math.pi/180)
            p.y=4*math.sin(2*t*K*math.pi/180)
            p.z=t
            # while not rospy.is_shutdown:
            waypoint_pub.publish(p)

X=list()
Y=list()

def get_current_pos(data):
            global current_x, current_y, current_theta, X, Y
            current_x = data.pose.pose.position.x
            current_y = data.pose.pose.position.y
            waypoint_publisher()
            X.append(current_x)
            Y.append(current_y)
            # # plot the /odom data
            if t>54:
            plt.plot(X,Y)
            plt.show(block=True)

def generate_waypoints():
            rospy.init_node('A',anonymous=True)
            odom_sub0 = rospy.Subscriber('/odom', Odometry, get_current_pos)
            rospy.spin()

if __name__=='__main__':
            generate_waypoints()
```

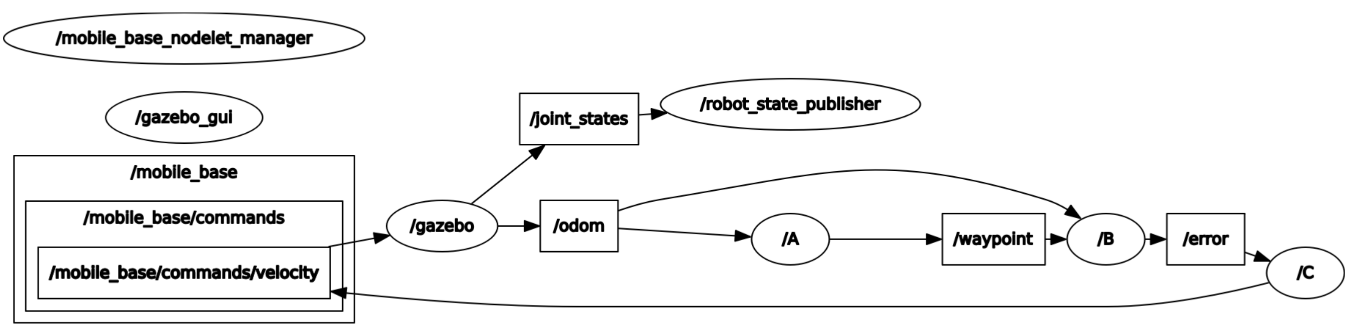
B.py

```
#!/usr/bin/env python
import rospy
import math
import matplotlib.pyplot as plt
from geometry_msgs.msg import Point
from nav_msgs.msg import Odometry
from week2_170040012_190100057.msg import Error
from tf.transformations import euler_from_quaternion
current_x = 0.0
current_y = 0.0
current_theta = 0.0
goal_x = 0.0
goal_y = 0.0
t=0.0
desired_theta = 0.0
E_pos = 0.0
E_theta = 0.0
delta_x = 0.0
delta_y = 0.0
error_pub = rospy.Publisher('/error', Error, queue_size=5)
def quat2euler(x,y,z,w):
    quat = [x,y,z,w]
    return euler_from_quaternion(quat)  # in radians
def distance(x1,y1,x2,y2):
        return math.sqrt( (x1-x2)**2 + (y1-y2)**2 )
def error_publisher():
        e=Error()
        e.E_pos=E_pos
        e.E_theta=E_theta
        error_pub.publish(e)
def odom_callback(data):
        global current_x, current_y, current_theta, pose
        current_x = data.pose.pose.position.x
        current_y = data.pose.pose.position.y
        x = data.pose.pose.orientation.x
        y = data.pose.pose.orientation.y
        z = data.pose.pose.orientation.z
        w = data.pose.pose.orientation.w
        current_theta=quat2euler(x,y,z,w)[2]  ### in radians
E_pos_arr = list()
E_theta_arr = list()
t_arr = list()
def waypoint_callback(data):
        global goal_x, goal_y, desired_theta, E_theta, E_pos, delta_x, delta_y, E_pos_arr, E_theta_arr, t_arr,
current_theta
        goal_x = data.x
        goal_y = data.y
        t = data.z
        K = 5
        E_pos = distance(current_x, current_y, goal_x, goal_y)
        desired_theta = math.asin((goal_y-current_y)/(E_pos))
        if (t>0 and t<=9) or (t>27 and t<=36):
        desired_theta = math.pi - desired_theta
        if (t>9 and t<=27):
        desired_theta = -math.pi - desired_theta
        if current_theta>0:
        E_theta = desired_theta - (current_theta - math.pi)
        elif current_theta<=0:
        E_theta = desired_theta - (current_theta + math.pi)
        E_pos_arr.append(E_pos)
        E_theta_arr.append(E_theta)
        t_arr.append(t)
        if t>54:
        E_pos=0
        E_theta=0
        plt.xlabel('waypoint index')
        plt.ylabel('error')
        plt.plot(t_arr, E_theta_arr)
        plt.legend("E theta")
        plt.plot(t_arr, E_pos_arr)
        plt.legend("E_pos")
        plt.show()
        error_publisher()
def odom_subscriber():
        rospy.init_node('B')
        odom_sub = rospy.Subscriber('/odom', Odometry, odom_callback)
        waypoint_sub = rospy.Subscriber('/waypoint', Point, waypoint_callback)
        rospy.spin()
if __name__ == '__main__':
        odom_subscriber()
```

C.py

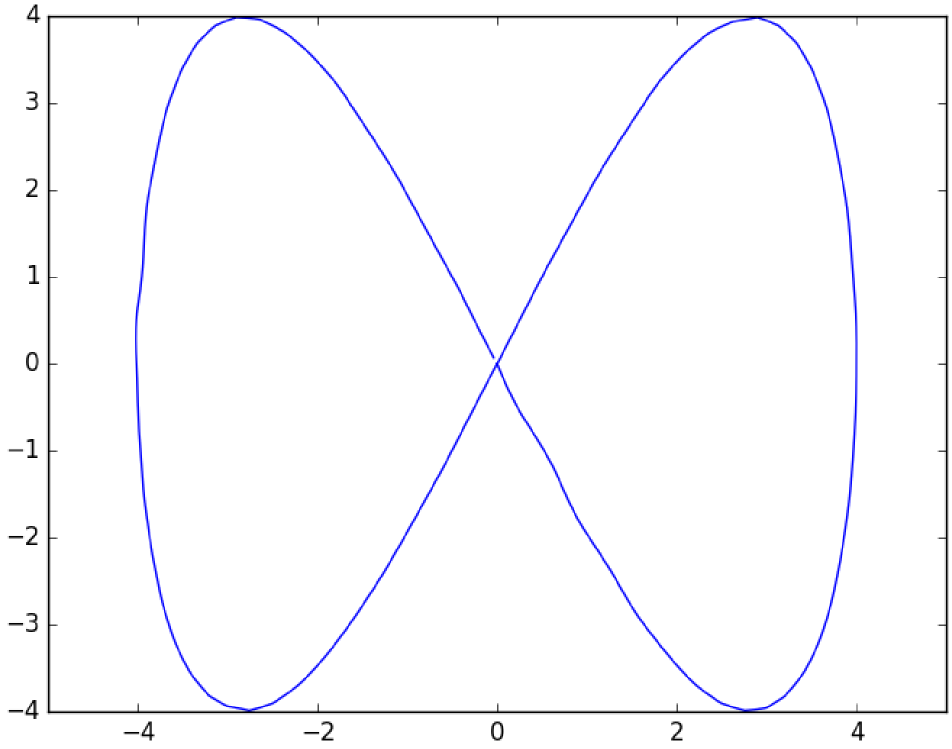
```
#!/usr/bin/env python

import rospy
import math
import matplotlib.pyplot as plt
from geometry_msgs.msg import Point, Twist
from nav_msgs.msg import Odometry
from week2_170040012_190100057.msg import Error

K1=1
K2=1
theta_threshold=5*math.pi/180

control_pub = rospy.Publisher('/mobile_base/commands/velocity', Twist, queue_size=10)

def control_law(data):
	E_pos = data.E_pos
	E_theta = data.E_theta

	velocity_command = Twist()

	velocity_command.linear.x = -1*min((K1)*E_pos,0.8)
	velocity_command.angular.z = K2*E_theta

	if abs(E_theta)>theta_threshold:
	velocity_command.linear.x = 0
	velocity_command.linear.y = 0
	velocity_command.angular.z = K2*E_theta

	control_pub.publish(velocity_command)

def error_subscriber():
	rospy.init_node('C')
	error_sub = rospy.Subscriber('/error', Error, control_law)
	rospy.spin()

if __name__ == '__main__':
	error_subscriber()
```

Rosgraph:

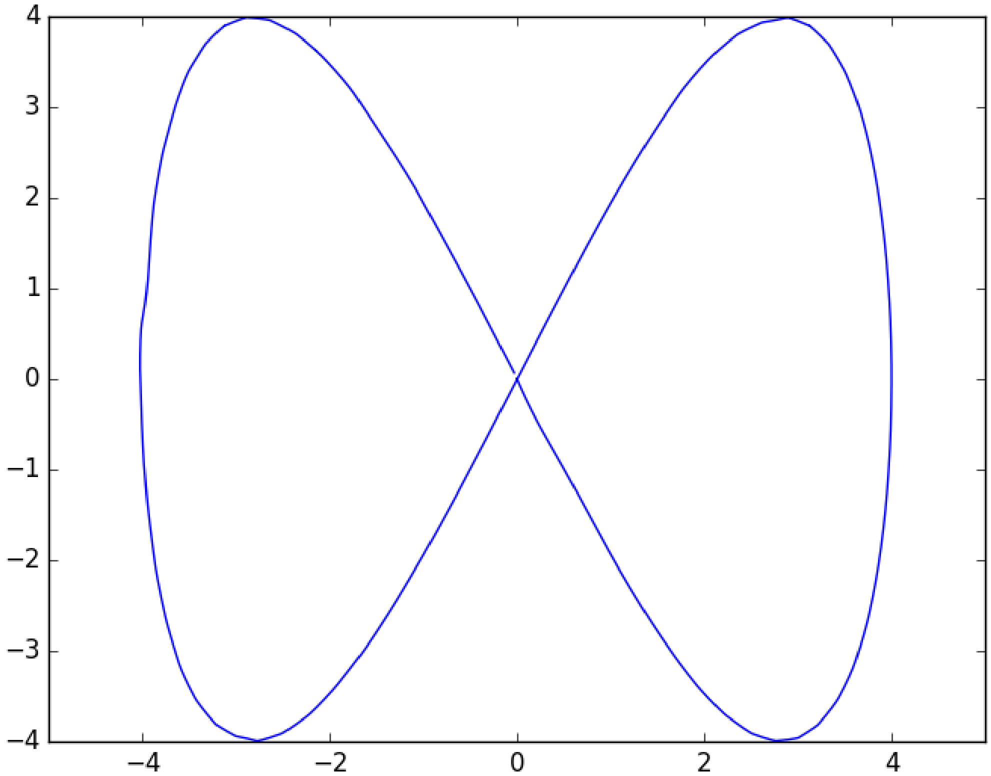


<u>Plots</u>:

motion of the robot for K1=0.2, K2=0.2

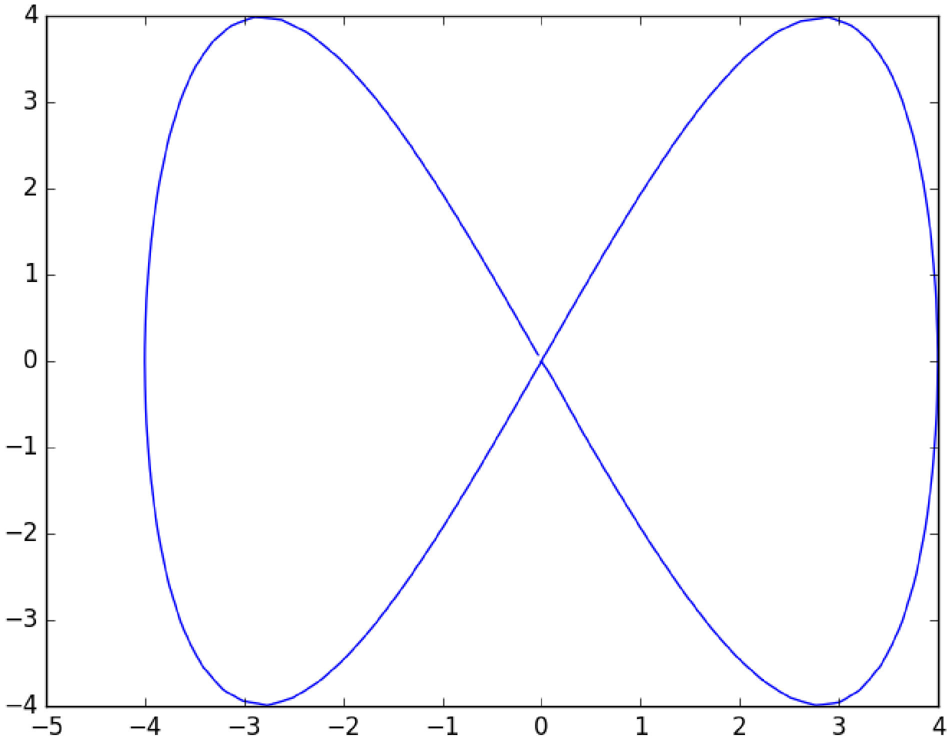


motion of the robot for K1=10, K2=2



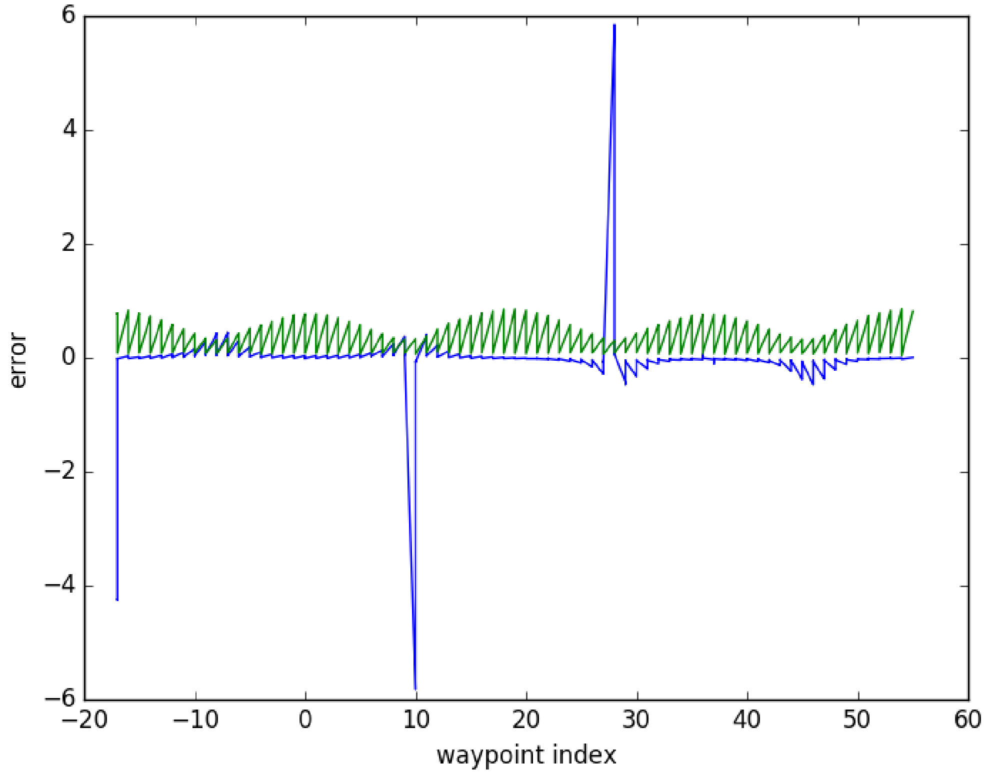
This shows that the suggested control law can also work well with high velocities.

Motion of robot with K1=10, K2=5



During the run, there was some visible error, like the center point of the ‘8’ is a bit shifted.
But still the overall shape is maintained.

plot of E_pos (green) and E_theta (blue) with waypoint index



The sharp peak in this graph is caused due to the sudden change of current orientation from +π to -π (or vice versa)

But this doesn’t affect our model because we move forward only when our error in the orientation is less than 5 degrees.

<u>Demo Video</u>:

https://drive.google.com/file/d/1bJR5Mvw92pgTKTa-W3DUTh0abODzTQEt/view?usp=sharing