Multi-Robot Planning Summary

Step 0: Create a new workspace

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src

$ catkin_init_workspace
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
```

Step 1: Create new package

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg multi_robot_bsp rospy std_msgs geometry_msgs turtlesim
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
```

Step 2: Create Launch file

```
$ cd ~/catkin_ws/src/multi_robot_bsp
$ mkdir -p launch
$ cd launch
$ touch multi_robot.launch
$ gedit multi_robot.launch

<—- Paste this in the launch file —->

<launch>
  <!-- Launch two turtlesim nodes with separate namespaces -->
  <group ns="robot1">
    <node pkg="turtlesim" type="turtlesim_node" name="sim" />
  </group>

  <group ns="robot2">
    <node pkg="turtlesim" type="turtlesim_node" name="sim" />
  </group>

  <!-- Your custom control nodes -->
  <node pkg="multi_robot_bsp" type="robot1_node.py" name="robot1_node" output="screen"/>
  <node pkg="multi_robot_bsp" type="robot2_node.py" name="robot2_node" output="screen"/>
</launch>

<———————————————>
```

Step 3: Create scripts

```
$ cd ~/catkin_ws/src/multi_robot_bsp
$ mkdir -p scripts
$ cd scripts
$ nano robot1_node.py

<————— Paste this in the robot1_node.py file —————>
```

```
#!/usr/bin/env python3
import rospy
import random
import math
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
from std_msgs.msg import String


belief = [[0.5 for _ in range(5)] for _ in range(5)]
pose = None

def comm_callback(msg):
    rospy.loginfo(f"[COMM] {msg.data}")

rospy.Subscriber('/comm_channel', String, comm_callback)


def pose_callback(msg):
    global pose
    pose = msg

def update_belief():
    x, y = random.randint(0, 4), random.randint(0, 4)
    belief[x][y] = min(belief[x][y] + 0.1, 1.0)
    rospy.loginfo(f"Updated belief[{x}][{y}] to {belief[x][y]:.2f}")

def find_max_uncertain_cell():
    max_val = -1
    max_cell = (0, 0)
    for i in range(5):
        for j in range(5):
            if belief[i][j] > max_val:
                max_val = belief[i][j]
                max_cell = (i, j)
    return max_cell


def predict_other_robot_target():
    # Dummy prediction: assumes robot2 uses min uncertainty
    from robot2_node import find_min_uncertain_cell
    return find_min_uncertain_cell()


def main():
    global pose
    rospy.init_node('robot1_controller')
    pub = rospy.Publisher('/robot1/turtle1/cmd_vel', Twist, queue_size=10)
    rospy.Subscriber('/robot1/turtle1/pose', Pose, pose_callback)
    comm_pub = rospy.Publisher('/comm_channel', String, queue_size=10)

    rate = rospy.Rate(1)  # 1 Hz

    while not rospy.is_shutdown():
        update_belief()

        if pose is None:
            rate.sleep()
            continue
```

```
        # Find most uncertain cell
        target = find_max_uncertain_cell()

        predicted_other = predict_other_robot_target()

        if predicted_other != target:
            comm_msg = String()
            comm_msg.data = "[R1] Detected disagreement, triggering communication!"
            comm_pub.publish(comm_msg)

        # Map grid (0–4) to turtlesim coordinates (approx. 1–10)
        target_x = 2 + target[0] * 2
        target_y = 2 + target[1] * 2

        # Compute angle to target
        angle_to_target = math.atan2(target_y - pose.y, target_x - pose.x)
        angle_diff = angle_to_target - pose.theta

        msg = Twist()
        msg.linear.x = 1.0
        msg.angular.z = angle_diff  # steer toward the most uncertain cell
        pub.publish(msg)

        rate.sleep()

if __name__ == '__main__':
    main()
<————————————————————————————————————————>
```

(save and exit)

$ nano robot2_node.py

<———————— Paste this in the robot2_node.py file ————————————->

```
#!/usr/bin/env python3
import rospy
import random
import math
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
from std_msgs.msg import String

belief = [[0.5 for _ in range(5)] for _ in range(5)]
pose = None
rospy.Subscriber('/comm_channel', String, lambda msg: rospy.loginfo(f"[COMM] {msg.data}"))


def pose_callback(msg):
    global pose
    pose = msg

def update_belief():
    x, y = random.randint(0, 4), random.randint(0, 4)
    belief[x][y] = min(belief[x][y] + 0.1, 1.0)
    rospy.loginfo(f"[R2] Updated belief[{x}][{y}] = {belief[x][y]:.2f}")

def find_min_uncertain_cell():
    min_val = 2.0  # greater than max possible belief
```

```
    min_cell = (0, 0)
    for i in range(5):
        for j in range(5):
            if belief[i][j] < min_val:
                min_val = belief[i][j]
                min_cell = (i, j)
    return min_cell

def predict_other_robot_target():
    # Dummy prediction: assumes robot1 uses max uncertainty
    from robot1_node import find_max_uncertain_cell
    return find_max_uncertain_cell()

def main():
    global pose
    rospy.init_node('robot2_controller')
    pub = rospy.Publisher('/robot2/turtle1/cmd_vel', Twist, queue_size=10)
    rospy.Subscriber('/robot2/turtle1/pose', Pose, pose_callback)
    comm_pub = rospy.Publisher('/comm_channel', String, queue_size=10)

    rate = rospy.Rate(1)  # 1 Hz

    while not rospy.is_shutdown():
        update_belief()

        if pose is None:
            rate.sleep()
            continue

        # Move toward the least uncertain cell
        target = find_min_uncertain_cell()
        target_x = 2 + target[0] * 2
        target_y = 2 + target[1] * 2

        angle_to_target = math.atan2(target_y - pose.y, target_x - pose.x)
        angle_diff = angle_to_target - pose.theta

        msg = Twist()
        msg.linear.x = 1.0
        msg.angular.z = angle_diff
        predicted_other = predict_other_robot_target()
        if predicted_other != target:
            comm_msg = String()
            comm_msg.data = "[R2] Disagreement detected, triggering communication!"
            comm_pub.publish(comm_msg)

        pub.publish(msg)
        rate.sleep()

if __name__ == '__main__':
    main()

<————————————————————————————>

(save and exit)

$ chmod +x robot1_node.py
$ chmod +x robot2_node.py
```

Step 4: Launch

$ roslaunch multi_robot_bsp multi_robot.launch