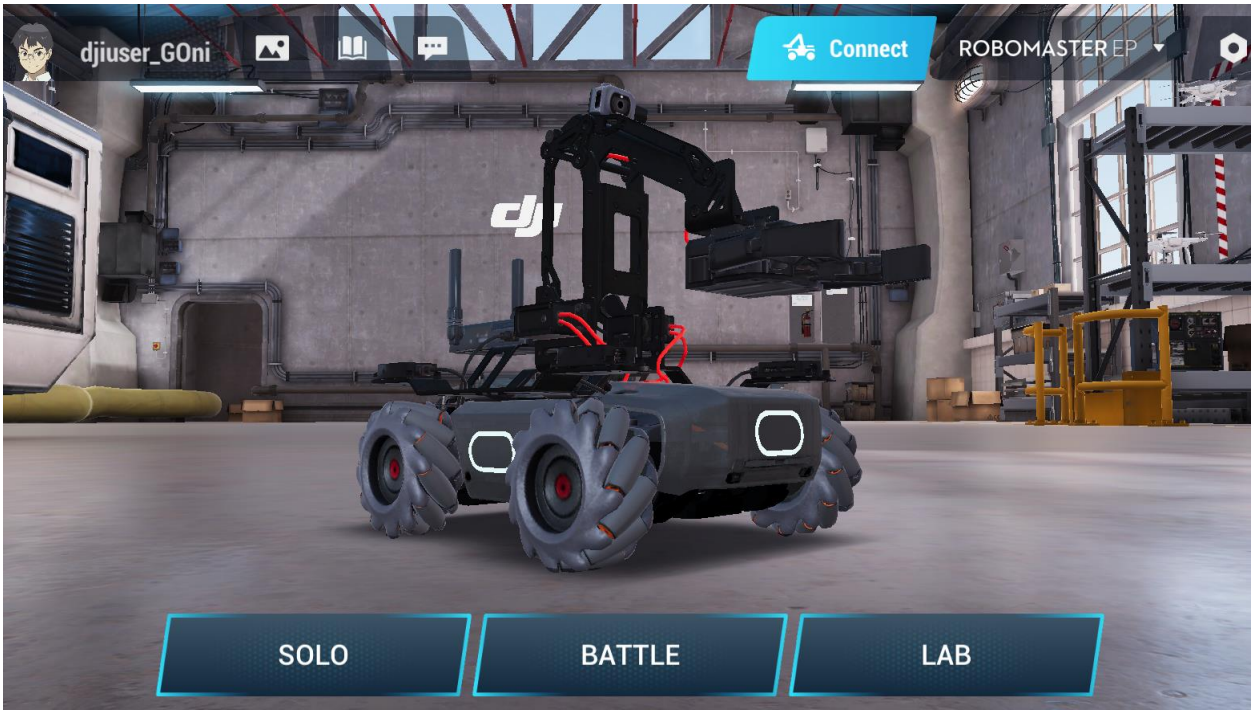
Lab 10

Student#21060007

Task1: INTRODUCTION TO ROBOMASTER APP



Task2: IMPLEMENTING ROBOMASTER LIBRARY

00_general	2/17/2022 1:11 PM	File folder
01_robot	2/17/2022 1:11 PM	File folder
02_chassis	2/17/2022 1:11 PM	File folder
03_gimbal	2/17/2022 1:11 PM	File folder
04_camera	2/17/2022 1:11 PM	File folder
05_vision	2/17/2022 1:11 PM	File folder
06_blaster	2/17/2022 1:11 PM	File folder
07_led	2/17/2022 1:11 PM	File folder
08_battery	2/17/2022 1:11 PM	File folder
09_armor	2/17/2022 1:11 PM	File folder
10_robotic_arm	2/17/2022 1:11 PM	File folder
11_gripper	2/17/2022 1:11 PM	File folder
12_drone	2/17/2022 1:11 PM	File folder
13_servo	2/17/2022 1:11 PM	File folder
14_sensor	2/17/2022 1:11 PM	File folder
15_multi_robot	2/17/2022 1:11 PM	File folder
plaintext_sample_code	2/17/2022 1:11 PM	File folder

Task3 : Implement PID

Code:

```
urr_x = 0
curr_y = 0
Kp = 0.5
v0 = 50



def sub_position_handler(position_info):
    global curr_x
    global curr_y
    curr_x, curr_y, curr_z = position_info
    print("chassis position: x:{0}, y:{1}, z:{2}".format(cur_x, cur_y, cur_z))


def sub_attitude_info_handler(attitude_info):
    global yaw
    yaw, pitch, roll = attitude_info
    # print("chassis attitude: yaw:{0}, pitch:{1}, roll:{2} ".format(yaw, pitch,
roll))


def dis_error(des_x, des_y, curr_x, curr_y):
    return math.sqrt((curr_x - des_x) ** 2 + (curr_y - des_y) ** 2)


if __name__ == '__main__':
    ep_robot = robot.Robot()
    ep_robot.initialize(conn_type="ap")
    ep_chassis = ep_robot.chassis
    ep_chassis.sub_position(freq=10, callback=sub_position_handler)
    ep_chassis.sub_attitude(freq=10, callback=sub_attitude_info_handler)

    goal_x = int(input("x coordinate= "))
    goal_y = int(input("y coordinate= "))

    distance_error = dis_error(goal_x,goal_y,curr_x,curr_y)

    while (distance_error >= 0.5):

        a = (des_y - curr_y)/(des_x - curr_x)
        e = math.atan(a)
        e = (e*180)/math.pi
        error = yaw - e
        angle_error = Kp * e

        if angle_error < 0:
            angle_speed=-angle_speed
        if angle_error >= 0:
            angle_speed=angle_speed
        distance_error = dis_error(goal_x,goal_y,curr_x,curr_y)*50
```

```
        if abs(angle_error) >= 5:
            ep_chassis.drive_wheels(w1=angle_speed, w2=-angle_speed, w3=0, w4=0)
        elif abs(angle_error) < 5 and distance_error >= 0.8:
            ep_chassis.drive_wheels(w1=distance_error, w2=distance_error,
w3=distance_error, w4=distance_error)
        else:
            ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)


    ep_chassis.unsub_position()
    ep_chassis.unsub_attitude()

    ep_chassis.drive_wheels(w1=0, w2=0, w3=0, w4=0)
    ep_robot.close()
```