ДОДАТОК А

Код програми

Код програми для дослідження датчиків

```
function sysCall_init()
    sim = require('sim')
    sensor0 = sim.getObject('/proximitySensor[0]')
    sensor1 = sim.getObject('/proximitySensor[1]')
    sensor2 = sim.getObject('/proximitySensor[2]')
    sensor3 = sim.getObject('/proximitySensor[3]')
    sensor4 = sim.getObject('/proximitySensor[4]')
    sensor5 = sim.getObject('/proximitySensor[5]')
    vision = sim.getObject('/visionSensorOr')
end

function sysCall_actuation()
end

function sysCall_sensing()
    flag0,p0=sim.readProximitySensor(sensor0)
    flag1,p1=sim.readProximitySensor(sensor1)
    flag2,p2=sim.readProximitySensor(sensor2)
    flag3,p3=sim.readProximitySensor(sensor3)
    flag4,p4=sim.readProximitySensor(sensor4)
    flag5,p5=sim.readProximitySensor(sensor5)
    print('flag0 = '..flag0..' '..'p0 = '..p0)
    print('flag1 = '..flag1..' '..'p1 = '..p1)
    print('flag2 = '..flag2..' '..'p2 = '..p2)
    print('flag3 = '..flag3..' '..'p3 = '..p3)
    print('flag4 = '..flag4..' '..'p4 = '..p4)
    print('flag5 = '..flag5..' '..'p5 = '..p5)
```

```
        result, data_sensor = sim.readVisionSensor(vision)
        print('Minimum intensity, r,g,b, depth: ' .. data_sensor[1].. ' '
..data_sensor[2].. ' ' ..data_sensor[3].. ' ' ..data_sensor[4]..' ' .. data_sensor[5])
        print('Maximum intensity, r,g,b, depth: ' .. data_sensor[6].. ' '
..data_sensor[7].. ' ' ..data_sensor[8].. ' ' ..data_sensor[9]..' ' .. data_sensor[10])
        print('Average intensity, r,g,b, depth: ' .. data_sensor[11].. ' '
..data_sensor[12].. ' ' ..data_sensor[13].. ' ' ..data_sensor[14]..' ' .. data_sensor[15])
    end

    function sysCall_cleanup()
    end
```

Код програми для моделі робота

```
    function sysCall_init()
        sim = require('sim')
        joint_left = sim.getObject('/Joint_left')
        joint_right = sim.getObject('/Joint_right')
        sim.setObjectInt32Param(joint_right, 2000,1)
        sim.setJointTargetVelocity(joint_right, 0.5)
        sim.setObjectInt32Param(joint_left, 2000,1)
        sim.setJointTargetVelocity(joint_left, -0.5)
        chassis = sim.getObject('/Chassis')
        graph = sim.getObject('/graph')
        chassis_vx = sim.addGraphStream(graph, 'vx', 'm/s', 0, {1,0,0})
        chassis_vy = sim.addGraphStream(graph, 'vy', 'm/s', 0, {0,1,0})
    end

    function sysCall_actuation()
        time = sim.getSimulationTime()
```

```
    if (time > 3) then
    sim.setJointTargetVelocity(joint_left, -0.2)
    end
end

function sysCall_sensing()
    chassis_position = sim.getObjectPosition(chassis, sim.handle_world)
    --print ('x=' .. chassis_position[1])
   -- print ('y=' .. chassis_position[2])
    chassis_linearVelocity, chassis_angularVelocity = sim.getObjectVelocity(chassis)
    print ('lV=' .. chassis_linearVelocity[1])
    print ('aV=' .. chassis_angularVelocity[1])
    sim.setGraphStreamValue(graph, chassis_vx, chassis_linearVelocity[1])
    sim.setGraphStreamValue(graph, chassis_vy, chassis_linearVelocity[2])

end

function sysCall_cleanup()
end
```

Код програми для руху робота по лінії

```
function sysCall_init()
    sim = require('sim')
    leftJoint = sim.getObject('/DynamicLeftJoint')
    rightJoint = sim.getObject('/DynamicRightJoint')
    left_sensor = sim.getObject('/LeftSensor')
    right_sensor = sim.getObject('/RightSensor')
```

```
    middle_sensor = sim.getObject('/MiddleSensor')
    graph = sim.getObject('/graph')
    chassis = sim.getObject('/LineTracer')
    chassis_vx = sim.addGraphStream(graph, 'vx', 'm/s', 0, {1,0,0})
    chassis_vy = sim.addGraphStream(graph, 'vy', 'm/s', 0, {0,1,0})
end

function sysCall_actuation()
    v = 20
    dv = 30
    sim.setJointTargetVelocity(leftJoint, v)
    sim.setJointTargetVelocity(rightJoint, v)
    if (data_left ~= nil) then
    intensity_left = data_left[11]
    intensity_right = data_right[11]
        if (intensity_right> 0.5 and intensity_left < 0.5) then
            print('Turn right')
            sim.setJointTargetVelocity(rightJoint, v+dv)
        end
        if (intensity_left> 0.5 and intensity_right < 0.5) then
            print('Turn left')
            sim.setJointTargetVelocity(leftJoint, v+dv)
        end
    end
end

function sysCall_sensing()
    result, data_left = sim.readVisionSensor(left_sensor)
    result, data_right = sim.readVisionSensor(right_sensor)
    result, data_middle = sim.readVisionSensor(middle_sensor)
```

```
        if (data_left ~= nil) then
        print('Average left intensity, r,g,b, depth: ' .. data_left[11].. ' '
..data_left[12].. ' ' ..data_left[13].. ' ' ..data_left[14]..' ' .. data_left[15])
        print('Average right intensity, r,g,b, depth: ' .. data_right[11].. ' '
..data_right[12].. ' ' ..data_right[13].. ' ' ..data_right[14]..' ' .. data_right[15])
        print('Average middle intensity, r,g,b, depth: ' .. data_middle[11].. ' '
..data_middle[12].. ' ' ..data_middle[13].. ' ' ..data_middle[14]..' ' .. data_middle[15])
        print (' ')
    end
    chassis_position = sim.getObjectPosition(chassis, sim.handle_world)
        chassis_linearVelocity, chassis_angularVelocity =
sim.getObjectVelocity(chassis)
        print ('lV=' .. chassis_linearVelocity[1])
        print ('aV=' .. chassis_angularVelocity[1])
        sim.setGraphStreamValue(graph, chassis_vx, chassis_linearVelocity[1])
        sim.setGraphStreamValue(graph, chassis_vy, chassis_linearVelocity[2])
    end

    function sysCall_cleanup()
    end
```

Код програми для слідування стідною

```
    function sysCall_init()
        lmotor=sim.getObjectHandle("./lumibot_leftMotor")
        rmotor=sim.getObjectHandle("./lumibot_rightMotor")
        sensorLF=sim.getObjectHandle('./LeftFront')
        sensorLR=sim.getObjectHandle('./LeftRear')
        sensorRF=sim.getObjectHandle('./RightFront')
```

```
    sensorRR=sim.getObjectHandle('./RightRear')
    sensorF=sim.getObjectHandle('./Forward')
    min_d = 0.15
    max_d = 0.25
    yaw_cutoff= 0.005
    fwd_cutoff= 0.25
    avg_default= 0.15
    fwd_default = 100
    v = 5
    dv = 7
    v_sharp = 3
    v_straight= 7
    avg = avg_default
    diff = 0
    fwd = fwd_default
    chassis = sim.getObject('/lumibot')
    graph = sim.getObject('/graph')
    chassis_vx = sim.addGraphStream(graph, 'vLeft', 'm/s', 0, {1,0,0})
    chassis_vy = sim.addGraphStream(graph, 'vRight', 'm/s', 0, {0,1,0})
end

function sysCall_actuation()
    sim.setJointTargetVelocity(lmotor,v_straight)
    sim.setJointTargetVelocity(rmotor,v_straight)
    if (fwd < fwd_cutoff) then
        print('going toward the wall, turn right')
        sim.setJointTargetVelocity(lmotor,v_sharp)
        sim.setJointTargetVelocity(rmotor,0)
    elseif (fwd > fwd_cutoff) then
        if (avg > max_d) then
```

```
            print('going away from the wall, turn left')
            sim.setJointTargetVelocity(lmotor,v-dv)
            sim.setJointTargetVelocity(rmotor,v)
        elseif (avg < min_d) then
            print('going toward the wall, turn right')
            sim.setJointTargetVelocity(lmotor,v)
            sim.setJointTargetVelocity(rmotor,v-dv)
        elseif(avg > min_d and avg<max_d) then
            if (diff> yaw_cutoff) then --LF>LR
            print('yaw correction: , turn left')
            sim.setJointTargetVelocity(lmotor,v-dv)
            sim.setJointTargetVelocity(rmotor,v)
            elseif (diff < -yaw_cutoff) then --LF<LR
            sim.setJointTargetVelocity(lmotor,v)
            sim.setJointTargetVelocity(rmotor,v-dv)
            end
        end
    end
end

function sysCall_sensing()
    flag1,LF=sim.readProximitySensor(sensorLF)
    flag2,LR = sim.readProximitySensor(sensorLR)
    flag3,F= sim.readProximitySensor(sensorF)
    if (flag1==0 and flag2 == 1) then
    avg = LR
    diff = 0
    elseif (flag1==1 and flag2 == 0) then
    avg = LF
    diff = 0
```

```
        elseif (flag1 ==1 and flag2== 1) then
        avg = 0.5*(LF+LR)
        diff = LF - LR
        else
        avg = avg_default
        diff = 0
        end
        if (flag3== 1) then
        fwd = F
        else
        fwd = fwd_default
        end
        print('avg= '.. avg .. ' '.. 'diff'.. diff.. ' '..'fwd='..fwd)
        chassis_position = sim.getObjectPosition(chassis, sim.handle_world)
        chassis_linearVelocity, chassis_angularVelocity =
sim.getObjectVelocity(chassis)
        sim.setGraphStreamValue(graph, chassis_vx, chassis_linearVelocity[1])
        sim.setGraphStreamValue(graph, chassis_vy, chassis_linearVelocity[2])
    end

    function sysCall_cleanup()
    end
```

ДОДАТОК Б

Демонстраційний матеріал

Позначення	Найменування	Дод. відомості
	Текстові документи	
1 ГЮИК. 503300.004ПЗ	Пояснювальна записка	83 арк., ф.
	Графічні документи	
2	Код програми	7 арк., ф.
3	Демонстраційні матеріали у вигляді	
	презентації	11 арк., ф.

					ГЮИК. 503300.004ВД			
Змін.	Арк.	Номер докум.	Підп.	Дата				
Розроб.		Кривчун			*Моделювання сенсорної системи мобільного робота у симуляторі CoppeliaSim* Відомість кваліфікаційної роботи	ЛІТ	Аркуш	Аркуші
Перевір.		Бронніков				Н	1	1
Т. контроль		Демська						
Н. контроль						ХНУРЕ кафедра КІТАР		
Затверд.		Невлюдов						