**RDMain (Main Funktion von behaviour->rightangledock**

import XXXXX

public class RDMain {

public static void main(java.lang.String[] args) {

SaphiraRobot robot = new SaphiraRobot(ECarDefines.DX4);

// SaphiraRobot robot = new SaphiraRobot(ECarDefines.JSIM\_DX3);

// robot.addLaser();

robot.setRobPose(new Pose(0,0,20));

BehGroup dock = new BehGroup("Dock");

BehGroup turn = new BehGroup("Turn");

//BehGroup Dock

BehConstTransVel cv = new BehConstTransVel("ConstTransVel", 300);

BehLimFor lf = new BehLimFor("LimFor", 500, 800, 100);

BehAlign al = new BehAlign("Align", 30, 5);

BehStop st = new BehStop("Stop", 1000);

BehSafeDock sd = new BehSafeDock("ObstDetect");

//BehGroup Turn

BehTurn bt = new BehTurn("Turn", 90);

dock.add(lf, 80);

dock.add(cv, 50);

dock.add(al, 75);

dock.add(st, 80);

robot.add(dock);

robot.add(new RDStrategy(dock,turn));

robot.run();

}

}

**BehSafeDock -> Ein Behavour als Bsp.**

package thi.irobcon.app.behaviour.rightangledock;

import thi.irobcon.saphira.desire.DesRotVel;

import thi.irobcon.saphira.desire.DesTransVel;

import thi.irobcon.saphira.lps.Pose;

import thi.irobcon.saphira.reactive.Behaviour;

public class BehSafeDock extends Behaviour {

protected int tolerance;

public BehSafeDock(String behName) {

super(behName);

}

public void fire() {

int leftObst = 0; int rightObst = 0;

leftObst = robot.getSonarRange(0);

rightObst = robot.getSonarRange(7);

if(leftObst < 1500 || rightObst < 1500){

addDesire(new DesTransVel(0, 0.7));

addDesire(new DesRotVel(0, 1.0));

System.out.println("SafeDock -- Stop!!!");

}

}

}

**RDStrategy -> Planung und Aktivieren der BehaviourGroups**

package thi.irobcon.app.behaviour.rightangledock;

import thi.irobcon.saphira.proactive.Strategy;

import thi.irobcon.saphira.reactive.BehGroup;

public class RDStrategy extends Strategy {

protected BehGroup dock, turn;

private boolean turned;

public RDStrategy(BehGroup dock, BehGroup turn) {

super();

this.dock = dock;

this.turn = turn;

addOutput("Try to dock ...");

turned = false;

dock.activateExclusive();

}

public void plan() {

if (dock.isSuccess()) {

//Found wall one!

addOutput("done\nSucceed ... stop");

//stopRunning();

if (turned){

stopRunning();

}

dock.deactivate();

addOutput("Turning....");

turn.activateExclusive();

}

if(turn.isSuccess() && !turned){

addOutput("done\nTurned Successfully");

turned = true;

turn.deactivate();

dock.activateExclusive();

}

/\*if(turned == true && dock.isSuccess()){

addOutput("Docked and Turned Sucessfully");

stopRunning();

}\*/

}

}

**BehStop – Weiteres Bsp (Behaviour)**

package thi.irobcon.app.behaviour.rightangledock;

import thi.irobcon.saphira.desire.DesTransVel;

import thi.irobcon.saphira.reactive.Behaviour;

public class BehStop extends Behaviour {

protected int stopDistance;

public BehStop(String behName, int stopDistance) {

super(behName);

this.stopDistance = stopDistance;

}

public void fire() {

int dist = robot.getSonarRange(3);

if ( dist <= stopDistance )

{

System.out.println( "Stop" );

addDesire(new DesTransVel(0, 1.0));

success();

}

}

}

ServiceRoboticsMain – Direktsteuerung Main

package thi.irobcon.app.script;

public class ServiceRoboticsMain {

public static void main(String[] args) {

ServiceRoboticsScripts imseScripts = new ServiceRoboticsScripts();

imseScripts.script();

}

}

ServiceRoboticsScripts – Hier Direktsteuerungsbefehle

import thi.irobcon.ecar.ECarDefines;

import thi.irobcon.script.IRobConScript;

public class ServiceRoboticsScripts extends IRobConScript {

public void script() {

// autp();

// patrol();

// slalom1();

// slalom2();

// clean();

transport1();

}

protected void autp() {

// useRobot(ECarDefines.JSIM\_DX1);

useRobot(ECarDefines.DX4);

turn(180);

}

protected void patrol() {

// useRobot(ECarDefines.JSIM\_DX1);

useRobot(ECarDefines.DX4);

for (int i=0; i < 10; i++) {

move(1000);

if (i%2==0) gripClose();

else gripOpen();

turn(180);

}

}

protected void clean() {

useRobot(ECarDefines.JSIM\_DX1);

setRobPose(0,2000,90);

}

protected void transport1()

{

// useRobot(ECarDefines.SIM\_DX1);

useRobot(ECarDefines.DX4);

setRobPose(-2000,0,0);

liftDown();

gripClose();

wait(50);

int distToBlock = (getSonarRange(3) + getSonarRange(4)) / 2;

while(getSonarRange(3) > 500 && getSonarRange(4) > 500)

{

speed(200);

}

speed(0);

gripOpen();

wait(50);

while(getSonarRange(3) > 260 && getSonarRange(4) > 260)

{

speed(50);

}

speed(0);

wait(10);

gripClose();

liftUp();

wait(50);

move(-(distToBlock));

turn(90);

panTilt(-90, 0);

while(getBlobX(2) != 195)

{

speed(150);

}

speed(0);

turn(-90);

panTilt(0,0);

move(distToBlock);

wait(10);

liftDown();

gripOpen();

wait(50);

move(-100);

}

public void useRobot(String name) {

ecar = new ECar(name);

}

public abstract void script();

protected void setRobPose(int x, int y, int th) {

ecar.setRobPose(x, y, th);

}

protected void move(int dist) {

ecar.move(dist);

}

protected void turn(int deg) {

ecar.turn(deg);

}

protected void speed(int transVel) {

ecar.speed(transVel);

}

protected void rotate(int rotVel) {

ecar.rotate(rotVel);

}

protected void gripOpen() {

ecar.gripOpen();

}

protected void gripClose() {

ecar.gripClose();

}

protected void liftUp() {

ecar.liftUp();

}

protected void liftDown() {

ecar.liftDown();

}

protected int getSonarRange(int sonNum) {

return ecar.getSonarRange(sonNum);

}

protected int[] getSonarRanges() {

return ecar.getSonarRanges();

}

protected int[] getLaserRanges() {

return ecar.getLaserRanges();

}

protected void addOutput(String s) {

System.out.print(s);

}

protected void addOutputLn(String s) {

System.out.println(s);

}

protected void wait(int cycles) {

ecar.wait(cycles);

}

protected int getBlobX(int channel) {

if (ecar.getNumBlobs(channel) > 0) {

// Blob blob = ecar.getBlob(channel, 0);

Blob blob = ecar.getBlob(channel, 1); // old version acts, wird direkt verwendet

return blob.getXCG();

}

else return -1;

}

protected void panTilt(int pan, int tilt) {

ecar.panTilt(pan, tilt);

}

|  |  |
| --- | --- |
| useRobot(String name) | Benutzer Roboter (ECarDefines.DX4 bzw. JSIM\_DX1) |
| setRobPose(int x, int y, int th) | Roboter an bestimmte Stelle setzen |
|  |  |
| **Lokomotion** |  |
| speed(int v) | Geschwindigkeit non blocking (+vor/-rück) |
| rotate(int av) | Rotation in deg/s non blocking |
| move(int d) | Bewegen für d Millimeter(!) blocking |
| turn(int a) | Drehen in a Grad. Note: -90 ist rechts |
|  |  |
| **Sonar** |  |
| getSonarRange(int n) | min. Hindernisdist. (mm) von Rob.Mittelp. |
|  |  |
| **Kamera** |  |
| getBlobX(int ch) | Koordinaten v. Schwerpunkt des Blobs von ch |
| panTilt(int pan, int tilt) | Kamera drehen (pan), Neigen (tilt) |
|  |  |
|  |  |
| **Manipulator (2 DOF)** |  |
| gripOpen() | Öffnen |
| gripClose() | Schließen |
| liftUp() | Greifer auf |
| liftDown() | Greifer runter |
|  |  |
| **Kontrollstruk.** |  |
| wait(int n) | Wait for n (100ms) |
|  |  |

|  |  |
| --- | --- |
| **Desires** |  |
| DesTransVel | Desire vor/rück |
| DesRotVel | Desire Rotation |
| DesCamPan / DesCamTilt | Desires Kamera Ausrichtung |
| DesGrip / DesLift | Greifer |
|  |  |
| **Behaviours** |  |
| BehConstTransVel(String actionName, int vel) | Fahren mit konst. Geschw |
| BehConstRotVel(String actionName, int vel) | Rotation mit konst. Geschw. |
| BehLimFor(name, stopDist, slowDist, slowSpeed) | Annähern und langsamer werden vor |
| BehLimBack(name, stopDist, slowDist, slowSpeed) | Annähern und langsamer werden rück |
| BehCamInit(name, pan, tilt, zoom) | Kamera initialisieren |
| BehMove() | Fahren nach Odometriedaten |
| BehTurn() | Drehen nach Odometriedaten |

Praktikumsaufgaben Blatt 1:

protected void transport2()

{

// useRobot(ECarDefines.SIM\_DX1);

useRobot(ECarDefines.DX4);

setRobPose(-2000,0,0);

liftDown();

gripClose();

wait(50);

int distToBlock = (getSonarRange(3) + getSonarRange(4)) / 2;

while(getSonarRange(3) > 500 && getSonarRange(4) > 500)

{

speed(200);

}

speed(0);

gripOpen();

wait(50);

while(getSonarRange(3) > 260 && getSonarRange(4) > 260)

{

speed(50);

}

speed(0);

wait(10);

gripClose();

liftUp();

wait(50);

move(-(distToBlock));

turn(90);

panTilt(-90, 0);

while(getBlobX(2) != 195)

{

speed(150);

}

speed(0);

turn(-90);

panTilt(0,0);

move(distToBlock);

wait(10);

liftDown();

gripOpen();

wait(50);

move(-100);

}

protected void transport1()

{

useRobot(ECarDefines.SIM\_DX1);

setRobPose(-2000,0,0);

liftDown();

gripClose();

int distToBlock = (getSonarRange(3) + getSonarRange(4)) / 2;

while(getSonarRange(3) > 300 && getSonarRange(4) > 300)

{

speed(200);

}

gripOpen();

while(getSonarRange(3) > 260 && getSonarRange(4) < 260)

{

speed(50);

}

speed(0);

wait(10);

gripClose();

liftUp();

move(-(distToBlock));

turn(90);

move(100);

}

protected void transport3()

{

// useRobot(ECarDefines.SIM\_DX1);

useRobot(ECarDefines.DX4);

setRobPose(-2000,0,0);

liftDown();

gripClose();

panTilt(0, -20);

wait(100);

// Searches for Colorchannel 1

int myX = getBlobX(1);

// Turns right until aligned with Blob

while(myX < 185 || myX > 205)

{

rotate(-7);

myX = getBlobX(1);

}

rotate(0);

wait(10);

//get median to Block from 2 Sensors

int distToBlock = (getSonarRange(3) + getSonarRange(4)) / 2;

// Medium range medium speed

while(getSonarRange(3) > 800 && getSonarRange(4) > 800)

{

speed(130);

myX = getBlobX(1);

// turn left

if(myX < 196)

{

rotate(5);

}

// turn right

if(myX > 194)

{

rotate(-5);

}

if(myX == 195)

{

rotate(0);

}

}

//Close range slow speed

while(getSonarRange(3) > 500 && getSonarRange(4) > 500)

{

speed(70);

myX = getBlobX(1);

// turn left

if(myX < 195)

{

rotate(3);

}

// turn right

if(myX > 195)

{

rotate(-3);

}

if(myX == 195)

{

rotate(0);

}

}

######### Fortsetzung auf nächster Seite!!!###########

Fortsetzung Transport3

//Stop and open Claw

speed(0);

rotate(0);

gripOpen();

wait(50);

//Move closer to Block using Sonar

while(getSonarRange(3) > 178 && getSonarRange(4) > 178)

{

speed(50);

}

speed(0);

wait(10);

//Close Grip and lift block

gripClose();

liftUp();

wait(50);

// Move backwards to start.

move(-(distToBlock));

turn(90);

//set camera to look right

panTilt(-90, -5);

myX = getBlobX(2);

while(myX < 195)

{

myX = getBlobX(2);

speed(200);

}

speed(0);

turn(-90);

panTilt(0,0);

move(distToBlock);

wait(10);

liftDown();

gripOpen();

wait(50);

move(-100);

}

}

BehAlign

package thi.irobcon.app.behaviour.rightangledock;

import thi.irobcon.saphira.desire.DesRotVel;

import thi.irobcon.saphira.lps.Pose;

import thi.irobcon.saphira.reactive.Behaviour;

public class BehAlign extends Behaviour {

protected int tolerance;

protected int rotVel;

public BehAlign(String behName, int tolerance, int rotVel) {

super(behName);

this.tolerance = tolerance;

this.rotVel = rotVel;

}

public void fire() {

int leftDist =0; int rightDist = 0;

Pose relObstaclePose = new Pose();

// Mit Sonaren explizit

leftDist = robot.getSonarRange(3);

rightDist = robot.getSonarRange(4);

/\*

// Mit Kegel

leftDist = (int) (robot.checkPolar(2, 10, relObstaclePose) -

robot.getRadius());

System.out.println( "Position linkes Hindernis: (" + relObstaclePose.getX() + "," +

relObstaclePose.getY() + ")");

rightDist = (int) (robot.checkPolar(-10, -2, null) -

robot.getRadius());

\*/

/\* // Mit Box

leftDist = (int) (robot.checkBox(2500, 500, 100, 100,

relObstaclePose) - robot.getRadius());

System.out.println("Left obstacle at relative position (" + relObstaclePose.getX() +

"," + relObstaclePose.getY() +

"," + relObstaclePose.getTh() + ") in distance " +

(leftDist + robot.getRadius()));

rightDist = (int) (robot.checkBox(2500, -500, 100, -100, null)

- robot.getRadius());

// Mit Positionsauswertung

Pose robPose = robot.getPose();

System.out.println("Robot at position (" + robPose.getX() +

"," + robPose.getY() +

"," + robPose.getTh() + ")");

Pose absObstaclePose = new Pose(robPose.getX() + relObstaclePose.getX(),

robPose.getY() + relObstaclePose.getY(),

robPose.getTh() + relObstaclePose.getTh() );

System.out.println("Left obstacle at absolute position (" + absObstaclePose.getX() +

"," + absObstaclePose.getY() +

"," + absObstaclePose.getTh() + ")");

double distLeft = robPose.findDistanceTo( absObstaclePose );

double angleLeft = robPose.findAngleTo( absObstaclePose );

System.out.println("Validating distance = " + distLeft + " and angle " + angleLeft );

\*/

// Kurskorrektur

System.out.println("Front Left = " + leftDist + " Front Right = " + rightDist);

if ( leftDist - rightDist > tolerance)

{

addDesire(new DesRotVel(-rotVel, 1.0));

System.out.println( "Turn right" );

}

else if ( rightDist - leftDist > tolerance)

{

addDesire(new DesRotVel(rotVel, 1.0));

System.out.println( "Turn left" );

}

else addDesire(new DesRotVel(0, 0.5));

}

}

BehEdgeDock

public class BehEdgeDock extends Behaviour {

protected int tolerance;

private boolean foundWall = false;

public BehEdgeDock(String behName) {

super(behName);

}

public void fire() {

int dist = robot.getSonarRange(3);

if(!foundWall){

if ( dist <= 1000 )

{

System.out.println( "Wand 1 Found" );

foundWall = true;

addDesire(new DesTransVel(0, 0.8));

}

}else {

if (dist <= 1000){

System.out.println("Wand 2 Found");

success();

}

}

}

}

BehSafeDock:

package thi.irobcon.app.behaviour.rightangledock;

import thi.irobcon.saphira.desire.DesRotVel;

import thi.irobcon.saphira.desire.DesTransVel;

import thi.irobcon.saphira.lps.Pose;

import thi.irobcon.saphira.reactive.Behaviour;

public class BehSafeDock extends Behaviour {

protected int tolerance;

public BehSafeDock(String behName) {

super(behName);

}

public void fire() {

int leftObst = 0; int rightObst = 0;

leftObst = robot.getSonarRange(0);

rightObst = robot.getSonarRange(7);

if(leftObst < 1500 || rightObst < 1500){

addDesire(new DesTransVel(0, 0.7));

addDesire(new DesRotVel(0, 1.0));

System.out.println("SafeDock -- Stop!!!");

}

// Kurskorrektur

System.out.println("Left = " + leftObst + " right = " + rightObst);

}

}