Mindroid Pilotworkshop Aufgabenstellung

TECHNISCHE UNIVERSITÄT DARMSTADT

11. bis 13.06.2018
Fachgebiet Echtzeitsysteme / MAKI
TU Darmstadt

LÖSUNGEN

Lösung 1 Hallo Welt

Listing 1: HelloWorld.java

```java
package org.mindroid.android.app.programs.workshop.solutions;

import org.mindroid.api.ImperativeWorkshopAPI;
import org.mindroid.impl.brick.Textsize;

public class HelloWorld extends ImperativeWorkshopAPI {

    public HelloWorld() {
        super("Hello World");
    }

    @Override
    public void run() {
           clearDisplay();
           drawString("Hello World", Textsize.MEDIUM, 10 , 50);
    }
}
```

Listing 2: HelloDate.java

```java
package org.mindroid.android.app.programs.workshop.solutions;

import org.mindroid.api.ImperativeWorkshopAPI;
import org.mindroid.impl.brick.Textsize;
import java.text.SimpleDateFormat;
import java.util.Date;

public class HelloDate extends ImperativeWorkshopAPI {

    public HelloDate() {
        super("Hello Date");
    }

    @Override
    public void run() {
           SimpleDateFormat formatter = new SimpleDateFormat("dd.MM.yyy");
           clearDisplay();
           drawString("Datum: " + formatter.format(new Date()), Textsize.SMALL,
               ↪ 10, 50);
           }
  }
```

Lösung 2 Den Roboter kennenlernen

Lösung 2.1 Fahren - die Antriebsmotoren

Listing 3: DriveSquare.java

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import org.mindroid.api.ImperativeWorkshopAPI;
4
5  public class DriveSquare extends ImperativeWorkshopAPI {
6
7      public DriveSquare() {
8             super("Drive Square");
9      }
10
11     @Override
12     public void run() {
13       for (int i = 0; i<3 && !isInterrupted(); i++) {
14           int angle = 90;
15           forward();
16           delay(1000);
17           turnRight(angle);
18           forward();
19           delay(1000);
20           turnLeft(angle);
21           backward();
22           delay(1000);
23           turnRight(angle);
24           backward();
25           delay(1000);
26           turnLeft(angle);
27       } // end of for
28       stop();
29     }
30 }
```

Lösung 2.2 Der Ultraschallsensor - Abstand Messen

Listing 4: ParkingSensor.java

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import org.mindroid.api.ImperativeWorkshopAPI;
4  import org.mindroid.api.ev3.EV3StatusLightColor;
5  import org.mindroid.api.ev3.EV3StatusLightInterval;
6  import org.mindroid.impl.brick.Textsize;
7
8  public class ParkingSensor extends ImperativeWorkshopAPI {
9
10     public ParkingSensor() {
11         super("Parking Sensor");
12     }
13
14   @Override
15     public void run() {
16         String previousState = "";
17         clearDisplay();
18         drawString("Parking sensor", Textsize.MEDIUM, 10, 10);
19         while (!isInterrupted()) {
20             clearDisplay();
21             if(getDistance() < 0.30f && getDistance() > 0.15f) {
22                 drawString("Hm :-/", Textsize.MEDIUM, 10, 10);
23                 if (!previousState.equals("hm")) {
24                     setLED(EV3StatusLightColor.YELLOW,
25                             EV3StatusLightInterval.BLINKING);
26                 }
27                 previousState = "hm";
28             } else if (getDistance() < 0.15f) {
29                 drawString("Oh oh :-O", Textsize.MEDIUM, 10, 10);
30                 if (!previousState.equals("oh")) {
31                     setLED(EV3StatusLightColor.RED,
32                             EV3StatusLightInterval.DOUBLE_BLINKING);
33                 }
34                 previousState = "oh";
35             } else {
36                 drawString("OK :-)", Textsize.MEDIUM, 10, 10);
37                 if (!previousState.equals("ok")) {
38                     setLED(EV3StatusLightColor.GREEN,
39                             EV3StatusLightInterval.ON);
40                 }
41                 previousState = "ok";
42             }
43             delay(100);
44         }
45     }
46 }
```

Lösung 2.3 Die Farbsensoren - Farbe messen

Listing 5: ColourTest.java

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import org.mindroid.api.ImperativeWorkshopAPI;
4  import org.mindroid.impl.brick.Textsize;
5  import org.mindroid.impl.statemachine.properties.Colors;
6
7  public class ColourTest extends ImperativeWorkshopAPI {
8
9      public ColourTest() {
10         super("Colour Test");
11     }
12
13      @Override
14     public void run() {
15         while (!isInterrupted()) {
16             Colors leftColorValue = getLeftColor();
17             Colors rightColorValue = getRightColor();
18
19             clearDisplay();
20             drawString("Colors", Textsize.MEDIUM, 1, 1);
21             drawString("L: " + describeColor(leftColorValue), Textsize.MEDIUM, 1,
                   ↪ 17);
22             drawString("R: " + describeColor(rightColorValue), Textsize.MEDIUM, 1,
                   ↪  33);
23             drawString("Distance: " + getDistance(), Textsize.MEDIUM, 1, 51);
24             delay(500);
25         }
26     }
27
28     private static String describeColor(final Colors colorValue) {
29         if (colorValue == Colors.NONE)   return "None";
30         if (colorValue == Colors.BLACK)  return "Black";
31         if (colorValue == Colors.BLUE)   return "Blue";
32         if (colorValue == Colors.GREEN)  return "Green";
33         if (colorValue == Colors.YELLOW) return "Yellow";
34         if (colorValue == Colors.RED)    return "Red";
35         if (colorValue == Colors.WHITE)  return "White";
36         if (colorValue == Colors.BROWN)  return "Brown";
37         return "unknown";
38     }
39 }
```

Lösung 2.4 Kommunikation zwischen Robotern

Listing 6: HelloWorldPingB.java

```java
package org.mindroid.android.app.programs.workshop.solutions;

import org.mindroid.api.ImperativeWorkshopAPI;

public class HelloWorldPingB extends ImperativeWorkshopAPI {

    public HelloWorldPingB() {
        super("Hello World Ping Berta");
    }

    @Override
    public void run() {
        clearDisplay();
        sendMessage("Robert", "Hallo Robert!");
    }
}
```

Listing 7: HelloWorldPingR.java

```java
package org.mindroid.android.app.programs.workshop.solutions;

import org.mindroid.api.ImperativeWorkshopAPI;
import org.mindroid.impl.brick.Textsize;

public class HelloWorldPingR extends ImperativeWorkshopAPI {

    public HelloWorldPingR() {
        super("Hello World Ping Robert");
    }

    @Override
    public void run() {
        clearDisplay();
        while(!isInterrupted()){
            if (hasMessage()){
                String msg = getNextMessage().getContent();
                if (msg.equals("Hallo Robert!")){
                    drawString("Nachricht von Berta erhalten", Textsize.MEDIUM, 1,
                        ↪  60);
                }
            }
            delay(100);
        };
    }
  }
```

Lösung 3 Wand-Ping-Pong

Listing 8: ImpSingleWallPingPong.java

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import org.mindroid.api.ImperativeWorkshopAPI;
4
5  public class ImpSingleWallPingPong extends ImperativeWorkshopAPI {
6
7
8      public ImpSingleWallPingPong() {
9          super("Single Wall-PingPong");
10     }
11
12     @Override
13     public void run() {
14         do {
15             forward(500);
16
17             while (getDistance() > 0.15f && !isInterrupted()) {
18                 delay(25);
19             }
20             stop();
21
22             driveDistanceBackward(10);
23
24             turnLeft(180);
25
26         }while(!isInterrupted());
27     }
28 }
```

Lösung 4 Koordiniertes Wand-Ping-Pong

Listing 9: ImpCoordWallPingPong.java

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import org.mindroid.api.ImperativeWorkshopAPI;
4  import org.mindroid.common.messages.server.MindroidMessage;
5  import org.mindroid.impl.brick.Button;
6
7  public class ImpCoordWallPingPong extends ImperativeWorkshopAPI {
8
9      private final String player_1 = "Robert";
10     private final String player_2 = "Berta";
11
12     //Messages
13     private final String leaderMsg = "I AM THE LEADER";
14     private final String startPingPongMsg = "YOUR TURN";
15
16     public ImpCoordWallPingPong() {
17         super("Coord Wall-PingPong");
18     }
19
20     @Override
21     public void run() {
22         String myID = getRobotID();
23         String colleague;
24
25         if(myID.equals(player_1)){
26             colleague = player_2;
27         }else{
28             colleague = player_1;
29         }
30
31         sendLogMessage("I am " + myID);
32         sendLogMessage("My Colleague is " + colleague);
33
34         boolean leaderElectionFinished = false;
35
36         while(!leaderElectionFinished && !isInterrupted()){
37             if(isButtonClicked(Button.ENTER)){
38                 sendLogMessage("I am the leader!");
39                 //I am the Leader
40                 sendMessage(colleague,leaderMsg);
41                 leaderElectionFinished = true;
42
43                 //Start doing wall ping pong
44                 runWallPingPong(colleague);
45             }
46
47             if(hasMessage()){
48                 MindroidMessage msg = getNextMessage();
49                 sendLogMessage("I received a message: "+msg.getSource().getValue()
                       ↪ +": \""+msg.getContent()+"\"");
50                 if(msg.getContent().equals(leaderMsg)){
```

```
51                      //Colleague is the leader
52                      leaderElectionFinished = true;
53                      sendLogMessage("I am NOT the leader!");
54                  }
55              }
56              delay(50);
57          }
58
59          while(!isInterrupted()){
60
61              if(hasMessage()){
62                  MindroidMessage msg = getNextMessage();
63                  sendLogMessage("I received a message: "+msg.getSource().getValue()
                        ↪ +": \""+msg.getContent()+"\"");
64                  if(msg.getContent().equals(startPingPongMsg)){
65                      runWallPingPong(colleague);
66                  }
67              }
68              delay(50);
69          }
70      }
71
72      /**
73       * Do a wall ping pong
74       * @param colleague - my colleagues id
75       */
76      private void runWallPingPong(String colleague){
77          forward(500);
78
79          while(!isInterrupted() && getDistance() > 0.15f){
80              delay(50);
81          }
82
83          enableFloatMode();
84
85          driveDistanceBackward(10f,350);
86
87          turnRight(180,350);
88
89          sendMessage(colleague,startPingPongMsg);
90
91          driveDistanceForward(40f);
92
93          enableFloatMode();
94
95          turnLeft(180,350);
96      }
97  }
```

Lösung 5 Mähroboter

Listing 10: LawnMower.java

```
package org.mindroid.android.app.programs.workshop.solutions;

import org.mindroid.api.ImperativeWorkshopAPI;
import org.mindroid.impl.statemachine.properties.Colors;

public class LawnMower extends ImperativeWorkshopAPI {

    private static Colors tapeColor = Colors.RED;
    private static float backDist = 10.0f;

    public LawnMower() {
        super("Lawn Mower");
    }

  @Override
    public void run() {
        while(!isInterrupted()){
            setMotorSpeed(200);
            forward();
            if(getLeftColor()== tapeColor && getRightColor() == tapeColor){
                driveDistanceBackward(backDist);
                turnRight(135);
            }else if(getLeftColor()== tapeColor){
                driveDistanceBackward(backDist);
                turnRight(90);
            }else if(getRightColor()== tapeColor){
                driveDistanceBackward(backDist);
                turnLeft(90);
            }
            delay(50);
        }
    }
}
```

Lösung 6 Platooning

Listing 11: Platooning.java

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import java.util.Random;
4  import org.mindroid.api.ImperativeWorkshopAPI;
5  import org.mindroid.api.ev3.EV3StatusLightColor;
6  import org.mindroid.api.ev3.EV3StatusLightInterval;
7  import org.mindroid.common.messages.server.MindroidMessage;
8  import org.mindroid.impl.brick.Button;
9  import org.mindroid.impl.brick.Textsize;
10
11 public class Platooning extends ImperativeWorkshopAPI {
12
13     public Platooning(){
14         super("Platooning");
15     };
16     enum State {
17         FAST,
18         MED,
19         SLOW
20     }
21     State prevState;
22
23     private final String player_1 = "Robert";
24     private final String player_2 = "Berta";
25     private final String myID = getRobotID();
26     private String colleague;
27
28     //Messages
29     private final String leaderMsg = "I AM THE LEADER";
30
31     @Override
32     public void run() {
33
34         // find out who i am, so i know who my colleague is
35         if(myID.equals(player_1)){
36             colleague = player_2;
37         }else{
38             colleague = player_1;
39         }
40
41         while(!isInterrupted()) {
42
43             if (isButtonClicked(Button.ENTER)) {
44                 sendLogMessage("I am the leader!");
45                 sendMessage(colleague, leaderMsg);
46                 driveAsLeader();
47             }
48             if (hasMessage()) {
49                 MindroidMessage msg = getNextMessage();
50                 sendLogMessage("I received a message: " + msg.getSource().getValue
                       ↪ () + ": \"" + msg.getContent() + "\"");
```

```
                if (msg.getContent().equals(leaderMsg)) {
                    //Colleague is the leader
                    sendLogMessage("I am NOT the leader!");
                    driveAsFollower();
                }
            }
        }
    }

    private void driveAsLeader(){
        setMotorSpeed(200);
        forward();
        while (!isInterrupted()) {
            delay(50);
        }
        stop();
    }
    private void driveAsFollower(){
        while(!isInterrupted()) {
            clearDisplay();
            float distance = getDistance();
            drawString("Dist: " + distance, Textsize.MEDIUM, 10,50);
            if (prevState != State.FAST && distance > 0.30f) {
                forward(300);
                prevState = State.FAST;
                setLED(EV3StatusLightColor.GREEN, EV3StatusLightInterval.ON);
            } else if (prevState != State.SLOW && distance < 0.20f) {
                forward(100);
                prevState = State.SLOW;
                setLED(EV3StatusLightColor.RED, EV3StatusLightInterval.ON);
            } else if (prevState != State.MED && distance > 0.20f && distance <
                ↪ 0.30f) {
                forward(200);
                prevState = State.MED;
                setLED(EV3StatusLightColor.YELLOW, EV3StatusLightInterval.ON);
            }
            delay(50);
        }
        stop();

    }
}
```

Lösung 7 Dancing Robots

Listing 12: Follow.java

```
package org.mindroid.android.app.programs.workshop.solutions;

import org.mindroid.api.ImperativeWorkshopAPI;
import org.mindroid.api.ev3.EV3StatusLightColor;
import org.mindroid.api.ev3.EV3StatusLightInterval;
import org.mindroid.common.messages.server.MindroidMessage;
import org.mindroid.impl.brick.Button;

```

```
9  import java.util.Random;
10
11 public class Follow extends ImperativeWorkshopAPI {
12
13     public Follow() {
14         super("Follower");
15     }
16     enum PlatoonState {
17         FAST,
18         MED,
19         SLOW
20     }
21     private PlatoonState prevState;
22
23     enum RoleState {
24         LEADER,
25         FOLLOWER
26     }
27     private RoleState role;
28
29     enum Direction {
30         LEFT,
31         RIGHT
32     }
33
34     private Random rnd = new Random();
35     private String colleague;
36
37     private String player_1 = "Robert";
38     private String player_2 = "Berta";
39
40
41     //Messages
42     private final String leaderMsg = "I AM THE LEADER";
43     private final String turnMsg = "TURN!";
44     private final String turnedMsg = " TURNED";
45     private String myTurnedMsg;
46     private String otherTurnedMsg;
47
48     @Override
49     public void run() {
50
51         String myID = getRobotID();
52         //String colleague;
53
54         // find out who i am, so i know who my colleague is
55         if(myID.equals(player_1)){
56             colleague = player_2;
57         }else{
58             colleague = player_1;
59         }
60
61         myTurnedMsg = myID + turnedMsg;
62         otherTurnedMsg = colleague + turnedMsg;
63         boolean initDone = false;
```

```
64          // get Roles
65          while(!isInterrupted() && !initDone ) {
66              if (isButtonClicked(Button.ENTER)) {
67                  sendLogMessage("I am the leader!");
68                  sendMessage(colleague, leaderMsg);
69                  role = RoleState.LEADER;
70                  initDone = true;
71              }
72              if (hasMessage()) {
73                  MindroidMessage msg = getNextMessage();
74                  sendLogMessage("I received a message: " + msg.getSource().getValue
                        ↪ () + ": \"" + msg.getContent() + "\"");
75                  if (msg.getContent().equals(leaderMsg)) {
76                      //Colleague is the leader
77                      sendLogMessage("I am NOT the leader!");
78                      role = RoleState.FOLLOWER;
79                  }
80                  initDone = true;
81              }
82          }
83
84          // drive with changing roles
85          while(!isInterrupted()){
86              switch(role){
87                  case LEADER:
88                      driveAsLeader(); break;
89                  case FOLLOWER:
90                      driveAsFollower(); break;
91              }
92          }
93      }
94
95      private void driveAsLeader(){
96          sendLogMessage("Leading the way!");
97          // drive at least 3000ms + random(0..3000ms)
98          int driveTime = (int) (300 + Math.floor(rnd.nextFloat() * 300));
99          int drivenTime = 0;
100         sendLogMessage("Driving for " + driveTime * 10 + "ms");
101         setMotorSpeed(200);
102         forward();
103         // drive until interrupted or time is up
104         while (!isInterrupted() && drivenTime < driveTime) {
105             delay(10);
106             drivenTime++;
107         }
108         stop();
109
110         sendMessage(colleague, turnMsg);
111         waitForTurn();
112         turn(Direction.RIGHT);
113         role = RoleState.FOLLOWER;
114     }
115
116     private void driveAsFollower(){
117         sendLogMessage("Following!");
```

```
118         while(!isInterrupted()) {
119             float distance = getDistance();
120             if (prevState != PlatoonState.FAST && distance > 0.35f) {
121                 forward(300);
122                 prevState = PlatoonState.FAST;
123                 setLED(EV3StatusLightColor.GREEN, EV3StatusLightInterval.ON);
124             } else if (prevState != PlatoonState.SLOW && distance < 0.25f) {
125                 forward(100);
126                 prevState = PlatoonState.SLOW;
127                 setLED(EV3StatusLightColor.RED, EV3StatusLightInterval.ON);
128             } else if (prevState != PlatoonState.MED && distance > 0.25f &&
                    ↪ distance < 0.35f) {
129                 forward(200);
130                 prevState = PlatoonState.MED;
131                 setLED(EV3StatusLightColor.YELLOW, EV3StatusLightInterval.ON);
132             }
133 
134             // check for Turn Message
135             if(hasMessage()){
136                 MindroidMessage msg = getNextMessage();
137                 sendLogMessage("I received: " + msg.getContent());
138                 if( msg.getContent().equals(turnMsg)){
139                     turn(Direction.LEFT);
140                     waitForTurn();
141                     role = RoleState.LEADER;
142                     return;
143                 }
144             }
145             delay(50);
146         }
147         stop();
148     }
149 
150     private void waitForTurn(){
151         while (!hasMessage()) delay(50);
152         if(hasMessage()){
153             MindroidMessage msg = getNextMessage();
154             if (msg.getContent().equals(otherTurnedMsg))
155                 sendLogMessage("I received: " + msg.getContent());
156         }
157     }
158 
159     private void turn(Direction dir){
160         sendLogMessage("turning...");
161         switch(dir) {
162             case LEFT:
163                 turnLeft(180, 100);
164                 break;
165             case RIGHT:
166                 turnRight(180, 100);
167                 break;
168         }
169         sendMessage(colleague, myTurnedMsg);
170     }
171 }
```