Mindroid Pilotworkshop am xx.06.2018

TECHNISCHE UNIVERSITÄT DARMSTADT

FG ES / MAKI
TU Darmstadt

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("ImpSingleWallPingPong");
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("ImpCoordWallPingPong");
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
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()+": \""+
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

```
1  package org.mindroid.android.app.programs.workshop.solutions;
2
3  import org.mindroid.api.ImperativeWorkshopAPI;
4
```

```java
public class LawnMower extends ImperativeWorkshopAPI {

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

  @Override
    public void run() {

    }
}
```

Lösung 6 Platooning

Listing 11: Platooning_A.java

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

import org.mindroid.api.ImperativeWorkshopAPI;

public class Platooning_A extends ImperativeWorkshopAPI {

    public Platooning_A() {
        super("Platooning A");
    }

    @Override
    public void run() {
        setMotorSpeed(200);
        forward();
        while (!isInterrupted()) {
            delay(10);
        }
        stop();
    }
}
```

Listing 12: Platooning_B.java

```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.impl.brick.Textsize;

public class Platooning_B extends ImperativeWorkshopAPI {

    enum State {
        FAST,
        MED,
        SLOW
    }

    State prevState;
```

```
17
18
19      public Platooning_B() {
20          super("Platooning B");
21      }
22
23      @Override
24      public void run() {
25          while (!isInterrupted()) {
26              float distance = getDistance();
27              clearDisplay();
28              if (prevState != State.FAST && distance > 0.35) {
29                  forward(250);
30                  prevState = State.FAST;
31                  setLED(EV3StatusLightColor.GREEN, EV3StatusLightInterval.ON);
32              }
33              else if (prevState != State.SLOW && distance < 0.15f) {
34                  forward(150);
35                  prevState = State.SLOW;
36                  setLED(EV3StatusLightColor.RED, EV3StatusLightInterval.ON);
37              }
38              else if (prevState != State.MED) {
39                  forward(200);
40                  prevState = State.MED;
41                  setLED(EV3StatusLightColor.YELLOW, EV3StatusLightInterval.ON);
42              }
43              delay(500);
44          }
45          stop();
46      }
47  }
```

Lösung 7 Verfolgung

Listing 13: Follow.java

```
1   package org.mindroid.android.app.programs.workshop.solutions;
2
3   import org.mindroid.api.ImperativeWorkshopAPI;
4
5   public class Follow extends ImperativeWorkshopAPI {
6
7       public Follow() {
8           super("Follower");
9       }
10
11      @Override
12      public void run() {
13      }
14  }
```