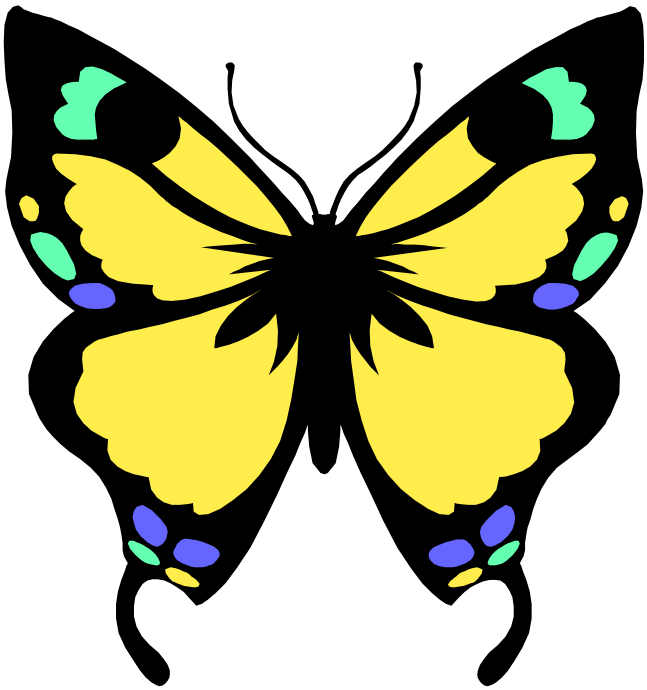
Robo Pro Light -> Java 6 / BlueJ 3.0

ROBO LT Beginner

Ulrich Müller



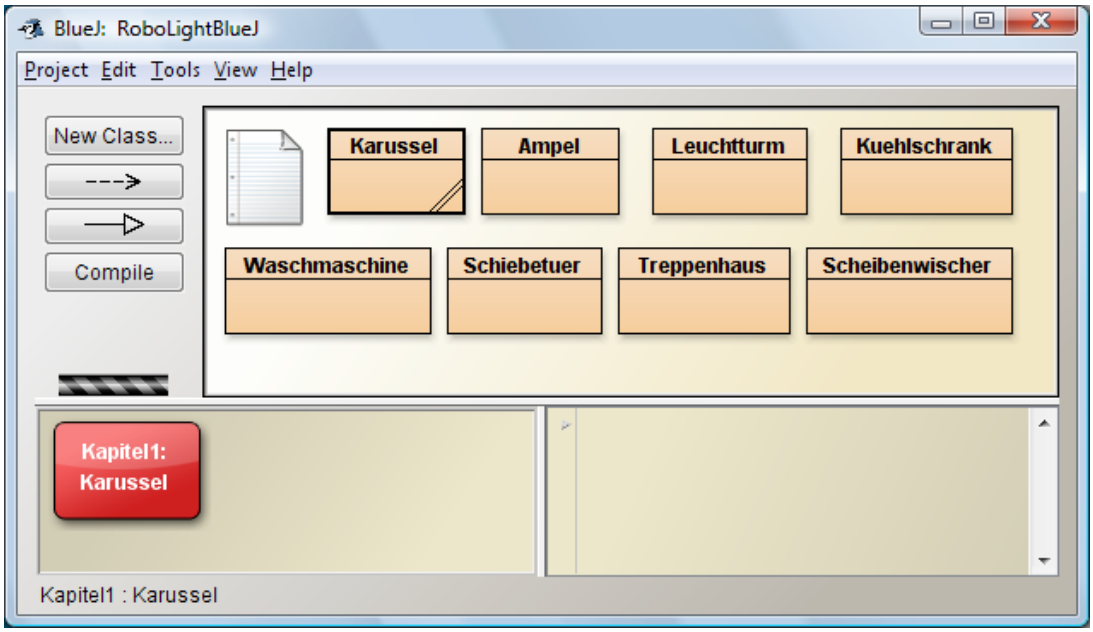
Inhaltsverzeichnis

Übersichten

Allgemeines

Die Programme können unter Vista 32bit und Windows 7 32/64bit und älteren Systemen betrieben werden.

Installation



BlueJ

Erforderlich ist eine Java Entwicklungsumgebung. Hier wird von BlueJ ausgegangen. BlueJ ist für Programmieranfänger besonders geeignet. BlueJ kann von http://www.bluej.org kostenlos bezogen werden. Bei den Beispielen wird die Fähigkeit von BlueJ genutzt, einzelne Methoden einer ausgewählten Klasse (hier Karussel) in der Entwicklungsumgebung ausführen zu können. Zusätzlich zu BlueJ sollte man das Tutorial downloaden.

BlueJ wird standardmäßig in englisch installiert. Eine Änderung auf deutsch ist im File \<BlueJ>\lib\bluej.defs möglich.

JDK Java Development Kit

ist bei den moderneren Rechnern bereits installiert, sonst JSE SDK von http://java.sun.com downladen.

USB Treiber

Das mit dem Kasten ROBO LT Beginner Lab gelieferte Robo Pro Light einschl. Treiber installieren.

Hinweis : Es kann genauso mit einem bereits vorhandenen Robo Interface und dem Robo Pro gearbeitet werden (auch im Wechsel mit dem LT Controller).

Hinweis 2 : Will man parallel dazu mit Robo Pro (standard) arbeiten, so ist dessen Version 3.0 erforderlich.

DLLs

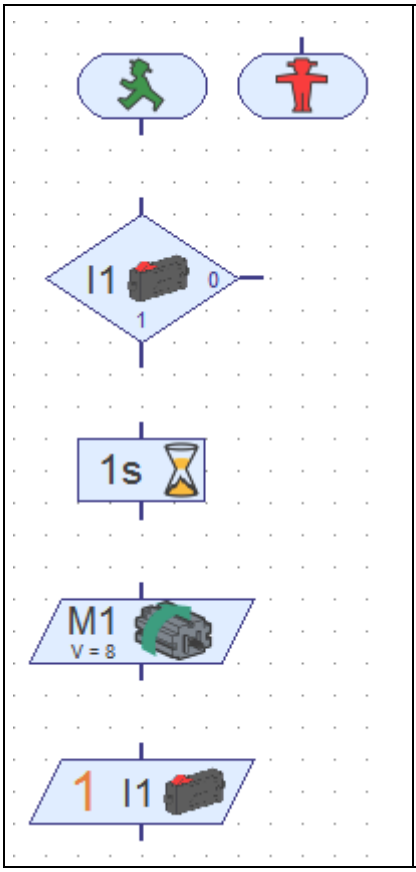
umFish40.DLL v4.3.77 und javaFish40.DLL aus roboLightBlueJ.ZIP. In ..\System32 bzw. ..\SystemWOW64 (Windows 7/64) unterbringen.

package

package ftcomputing.robo.jar aus roboLightBlueJ.ZIP

In der nutzenden Source ist ein import ftcomputing.robo.*; erforderlich. Zusätzlich muß das package bei BlueJ angemeldet werden : Menü | Tools| Preferences | Libraries

Gegenüberstellung ROBO Pro Light - Java-Befehle




```
public void name() {
}
```

```
if (fl.getInput(Inp.I1)){
' --- möglich : Inp.I1, I2, I3 ---
-----
}    Abfrage Eingang und Verzweigung
```

```
fl.pause(1000);    ' --- Zeitangabe in Millisekunden
```

```
fl.setMotor(Out.M1, Dir.Rechts, Speed.Full);
' --- möglich : Out.M1, M2, Dir.Ein (Lampen)
'     Dir.Links, Recht, Aus. Speed.L1 … Full, Half
'     Bei Speed.Full kann es auch entfallen
fl.waitForInput(Inp.I1, true);
' --- true kann auch entfallen, alternativ false
Warten auf einen Sensoreingang.
```

Die mit fl. beginnenden Methoden sind Teil der Klasse FishFace aus ftcomputing.robo.jar
Weitere Dokumentation dazu findet man in ftComputing für Java
http://ftcomputing.de/pdf/Eclipse34Fish.pdf.

Hinweise zur Swing-Programmierung auf http://www.ftcomputing.de/bluejSwing.htm

Programmrahmen

Hinweis : Der LT Controller muß vor dem Start eines Programmes am Rechner angeschlossen sein. Nach Programmänderungen muß die BlueJ IDE durch Rechtsclick auf den Laufbalken zurückgesetzt werden.

Die Beispielprogramme sind "Konsolenanwendungen" und werden über Menü | Projects | New Projects.. erstellt. Innerhalb eines Projektes kann man über den Button New Class neue Klassen anlegen. Die erste Methode wird durch Modifikation der vorgegebenen Beispiel-Methode erstellt, dann freiHand.

```
import ftcomputing.robo.*;
/**
 * Betrieb des Karussels von Robo LT Beginner Lab
 *
 * @author Ulrich Müller
 * @version 12.08.10
 */
public class Karussel {
    private FishFace fl;

    public Karussel() {
        fl = new FishFace();
        fl.openInterface(0, 0);
    }
    public void Beenden() {
        fl.closeInterface();
    }
    public void Karussel1() {
        System.out.println("Karussel starten : I1-Taster");
        fl.waitForInput(Inp.I1);
-----
    }
}
```

`import`: Der Namespace ftcomputing.robo soll verwendet werden. Dazu muß vorher das packages ftcomputing.robo.jar über das Menü der BlueJ IDE nach Tools | Preferences | Libraries aufgenommen werden.

`class`: Rahmen für das Gesamtprogramm (Ausgaben erfolgen im "Terminal Window" von BlueJ.

`private FishFace fl`: Instanz der Klasse FishFace für den Zugriff auf den LT Controller (über die Objekt-Variable `fl`)

`public void Karussel1()` { -- }: Das Methode mit dem Betriebsprogramm. Start : RechtsClick auf Kasten Karussel in der IDE, dann Instanz, ergibt einen roten Kasten mit dem Objekt Karussel, wieder RechtsClick auf die gewünschte Methode.

```
System.out.println("Karussel starten : I1-Taster");
```

Kommentierender Text, der im Terminal Window angezeigt wird.

`openInterface`(..) und `closeInterface` : Herstellen und Beenden einer Verbindung zum LT Controller an USB.

Modell-Programme

Karussel

Betrieb eines Karussels über einen Motor an M1 und einen Starttaster an I1.

1 : Karusselfahren für 10 Sekunden, Start über I1

2 : Wiederholmöglichkeit durch Endlosschleife

3 : Zusätzlich Drehen in der Gegenrichtung

Version 1 :

```
    public void Karussel1() {
        System.out.println("Karussel starten : I1-Taster");
        fl.waitForInput(Inp.I1);
        fl.setMotor(Mot.M1, Dir.Left);
        fl.pause(10000);
        fl.setMotor(Mot.M1, Dir.Off);
        System.out.println("--- FINIS ---");
    }
```

Warten auf Starttaster, Einschalten des Motors, 10 Sekunden warten, Ausschalten des Motors. Diese Befehle werden anstelle des Textes -Nutzfunktionen- in den Programmrahmen eingefügt.

Version 2 :

```
    public void Karussel2() {
        do {
            System.out.println("Neue Runde : I1-Taster");
            fl.waitForInput(Inp.I1);
            fl.setMotor(Mot.M1, Dir.Left);
            fl.pause(10000);
            fl.setMotor(Mot.M1, Dir.Off);
        } while (!fl.finish());
        System.out.println("--- FINIS ---");
    }
```

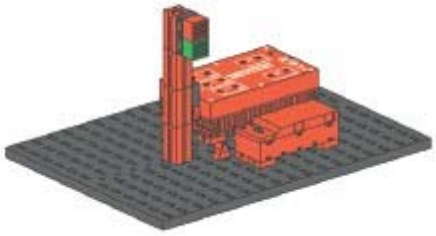
Wie 1, jetzt aber in einer "endlosen" `do {} while` Schleife, die über die ESC-Taste der Tastatur abgebrochen werden kann (Führe die Befehle zwischen Do und Loop aus bis Finish() feststellt, dass eine ESC-Taste gedrückt wurde).

Version 3 :

```
    public void Karussel3() {
        do {
            System.out.println("Neue Runde : I1-Taster");
            fl.waitForInput(Inp.I1);
            fl.setMotor(Mot.M1, Dir.Left);
            fl.pause(10000);
            fl.setMotor(Mot.M1, Dir.Off);
            fl.pause(1000);
            fl.setMotor(Mot.M1, Dir.Right);
            fl.pause(10000);
            fl.setMotor(Mot.M1, Dir.Off);
        } while (!fl.finish());
        System.out.println("--- FINIS ---");
    }
```

Wie 2, hinzugekommen sind 10 Sekunden in Gegenrichtung.

Fußgängerampel



Betrieb einer Fußgängerampel mit Lampen Rot an M1 und Grün an M2 sowie einem Anforderungstaster an I1 in einer Endlosschleife

1 : Rot einschalten, Fußphase anfordern,
5 Sekunden warten. 10 Sekunden grün

2 : Zusätzlich Grünblinken am Ende der Fußphase

Version 1 :

```
    public void Ampel1() {
        fl.setMotor(Mot.M1, Dir.On);
        do {
            if (fl.getInput(Inp.I1)) {
                System.out.println("Signal kommt");
                fl.pause(5000);
                fl.setMotor(Mot.M1, Dir.Off);
                fl.setMotor(Mot.M2, Dir.On);
                fl.pause(10000);
                fl.setMotor(Mot.M1, Dir.On);
                fl.setMotor(Mot.M2, Dir.Off);
            }
        } while (!fl.finish());
        fl.setMotor(Mot.M1, Dir.Off);
        System.out.println("--- FINIS ---");
    }
```

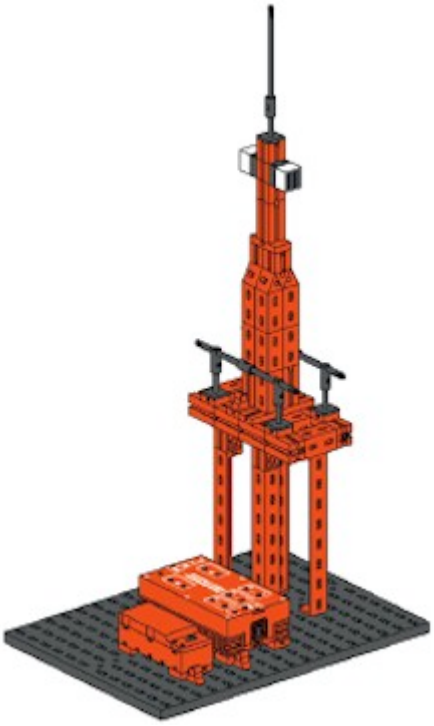
Einschalten Rot, Endlosschleife mit Warten auf Anforderung Fußphase (`If (fl.getInput`..), bei Erkennen : Nachricht, Warten 5 Sekunden, Rot Aus, Grün An für 10 Sekunden, dann wieder Rot an, Grün aus.

Version 2 :

```
public class Ampel {
    private FishFace fl;
    final int Rot         = Mot.M1;
    final int Gruen       = Mot.M2;
    final int Anforderung = Inp.I1;

-----
    public void Ampel2() {
        int i;
        fl.setMotor(Rot, Dir.On);
        do {
            if (fl.getInput(Anforderung)) {
                System.out.println("Signal kommt");
                fl.pause(5000);
                fl.setMotor(Rot, Dir.Off);
                fl.setMotor(Gruen, Dir.On);
                fl.pause(10000);

                for (i = 1; i <= 3; i++) {
                    fl.setMotor(Gruen, Dir.Off);
                    fl.pause(1000);
                    fl.setMotor(Gruen, Dir.On);
                    fl.pause(1000);
                }
                fl.setMotor(Rot, Dir.On);
                fl.setMotor(Gruen, Dir.Off);
            }
        } while (!fl.finish());
        fl.setMotor(Rot, Dir.Off);
        System.out.println("--- FINIS ---");
    }
}
```

`final`: Um die Lesbarkeit des Programmes zu erhöhen werden hier anstelle der bisherigen `enums` (von FishFace) Konstanten mit "sprechenden" Namen verwendet. Hinzugekommen ist eine `for` -Schleife in der Gruen 3mal blinkt.

Leuchtturm mit Blinklicht



Betrieb eines Leuchtturms mit zwei weißen Lampen (M1, M2) an der Turmspitze.

1 : Gleichtaktfeuer - Hell und Dunkel sind gleich lang (2 Sekunden)

2 : Blitzprinzip - Die Lichtphasen sind kürzer als die Dunkelphasen (0,5 / 1,5 Sekunden)

3. Blinkprinzip - Lichtphasen sind kürzer als die Dunkelphasen. Die Lampen leuchten unabhängig voneinander.

Version 1 :

```
    public void Leuchtturm1() {
        do {
            fl.setMotor(Mot.M1, Dir.On);
            fl.setMotor(Mot.M2, Dir.On);
            fl.pause(2000);
            fl.setMotor(Mot.M1, Dir.Off);
            fl.setMotor(Mot.M2, Dir.Off);
            fl.pause(2000);
        } while (!fl.finish());
    }
```

Version 2 :

```
    public void Leuchtturm2() {
        do {
            fl.setMotor(Mot.M1, Dir.On);
            fl.setMotor(Mot.M2, Dir.On);
            fl.pause(300);
            fl.setMotor(Mot.M1, Dir.Off);
            fl.setMotor(Mot.M2, Dir.Off);
            fl.pause(1500);
        } while (!fl.finish());
     }
```

Wie gehabt, aber mit anderen Zeiten.

Version 3 - ist haarig :

```
    public void Leuchtturm3() {
        Thread Lampe2 = new Thread(new Lampe2Blinken());
        Lampe2.start();

        do {
            fl.setMotor(Mot.M1, Dir.On);
            fl.pause(2000);
            fl.setMotor(Mot.M1, Dir.Off);
            fl.pause(3000);
        } while (!fl.finish());
    }
    class Lampe2Blinken implements Runnable {
        @Override public void run() {
            do {
                fl.setMotor(Mot.M2, Dir.On);
                fl.pause(3000);
                fl.setMotor(Mot.M2, Dir.Off);
                fl.pause(5000);
            } while (!fl.finish());
        }
    }
```

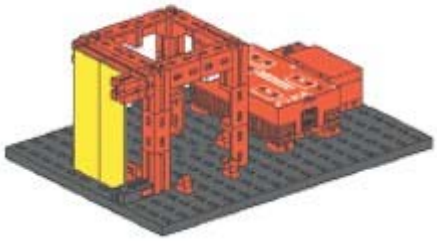
Mit Robo Pro ist das deutlich einfacher : Man benötigt dazu nur ein zweites grüne Männchen

Mit Java geschieht das über Threading (mehrere unabhängige Programmzweige) :

```
        Thread Lampe2 = new Thread(new Lampe2Blinken());
        Lampe2.start();
```

Definieren und Starten eines zusätzlichen Threads ("Programmfaden") die eigentliche Verarbeitung geschieht in den zugehörenden Threadroutine (Lampe2Blinken ..)
Geblinkt wird im Thread der aufgerufenen Methode und in dem zweiten Thread Lampe2Blinken. Ende durch ESC-Taste.

Kühlschrank



Simulation eines Kühlschranks.

1 : Tür offen (Taster I1 aus) - weiße Lampe an

2 : Zusätzlich - nach 3 Sekunden Rotblinken.

Version 1 (harmlos) :

```
public void Kuehlschrank1() {
    do {
        if (fl.getInput(Inp.I1)) fl.setMotor(Mot.M1, Dir.Off);
        else fl.setMotor(Mot.M1, Dir.On);
    } while (!fl.finish());
    fl.setMotor(Mot.M1, Dir.Off);
}
```

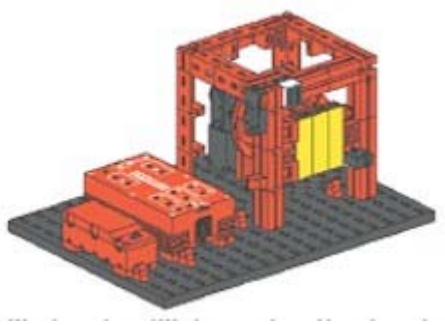
`fl.getInput` liefert als Return-Wert `true`, wenn der Sensor geschlossen ist und `false`, wenn nicht. Im `if` muß ein logischer Ausdruck stehen, der `true` oder `false` liefert, das tut getInput.
Wenn man nun aber lieber den then- und den else-Zweig vertauschen will, kann man das durch ein vorangestelltes `!` tun. Man kann aber auch einfach `fl.getInput(Inp.I1) == false` schreiben.

Version 2 (wieder eine Thread-Lösung) - Funktioniert genauso wie die Leuchtturm-Lösung.

```
public void Kuehlschrank2() {
    Thread LampeRot = new Thread(new LampeRotBlinken());
    LampeRot.start();

    do {
        if (fl.getInput(Inp.I1) == false)
            fl.setMotor(Mot.M1, Dir.On);
        else fl.setMotor(Mot.M1, Dir.Off);
    } while (!fl.finish());
    fl.setMotor(Mot.M1, Dir.Off);
}
class LampeRotBlinken implements Runnable {
    @Override public void run() {
        do {
            fl.waitForInput(Inp.I1, false);
            fl.pause(3000);
            do {
              fl.setMotor(Mot.M2, Dir.On);
              fl.pause(500);
              fl.setMotor(Mot.M2, Dir.Off);
              fl.pause(500);
            } while (!(fl.finish() | fl.getInput(Inp.I1)));
        } while (!fl.finish());
    }
}
```

Es wird endlos auf das Öffnen der Kühlschranktür gewartet. Wird sie geöffnet, wird nach 3 Sekunden Rot geblinkt. Der `do {} while` fragt ständig den Türtaster ab und enthält ein zusätzlich ein `| fl.finish()` um die Abbrechbarkeit des Programms sicherzustellen.

Waschmaschine



Die Trommel der Waschmaschine wird nach allen Regeln der Kunst gescheucht. Motor an M1, Anzeige an M2, Starttaster an I1, TuerZu-Taster an I2.

1: Nach Starttaster langsam Drehen für 10 Sekunden, 1 Sekunde Pause

2 : Wie 1 plus Warten auf TürZu

3 : Plus Schleudergang

4 : Plus Trocknen und TextAnzeige

Version 1 :

```
    public void Waschmaschine1() {
        do {
            System.out.println("Start Waschgang : I1-Taster");
            fl.waitForInput(Starter);
            fl.setMotor(Anzeige, Dir.On);
            fl.setMotor(Trommel, Dir.Right, Speed.L2);
            fl.pause(10000);
            fl.setMotor(Trommel, Dir.Off);
            fl.setMotor(Anzeige, Dir.Off);
            fl.pause(1000);
        } while (!fl.finish());
    }
```

Version 2 :

```
    public void Waschmaschine2() {
        do {
            System.out.println("Start Waschgang : I1-Taster");
            do {
                fl.waitForInput(Starter);
            } while (!fl.getInput(TuerZu));
            fl.setMotor(Anzeige, Dir.On);
            fl.setMotor(Trommel, Dir.Right, Speed.L2);
            fl.pause(10000);
            fl.setMotor(Trommel, Dir.Off);
            fl.setMotor(Anzeige, Dir.Off);
            fl.pause(1000);
        } while (!fl.finish());
    }
```

In `do {} while` wird auf den Stattaster gewartet, das do wird aber erst verlassen, wenn TuerZu ist.

Version 3 :

Mit dem versprochenen Schleudergang :

```
    fl.setMotor(Trommel, Dir.Right, Speed.Full);
            fl.pause(15000);
```

Version 4 :

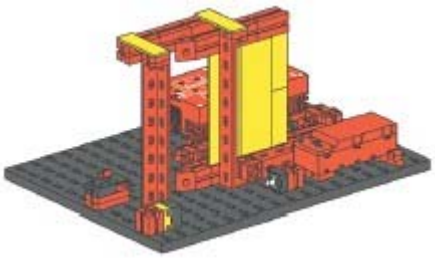
Mit Trockengang und Textausgabe zur Anzeige des aktuellen Waschganges.

```
    public void Waschmaschine4() {
        do {
            System.out.println("Start Waschgang : I1-Taster");
            do {
                fl.waitForInput(Starter);
            } while (!fl.getInput(TuerZu));

            System.out.println("Waschen");
            fl.setMotor(Anzeige, Dir.On);
            fl.setMotor(Trommel, Dir.Right, Speed.L2);
            fl.pause(10000);
            fl.setMotor(Trommel, Dir.Off);
            fl.pause(1000);

            System.out.println("Schleudern");
            fl.setMotor(Trommel, Dir.Right, Speed.Full);
            fl.pause(15000);
            fl.setMotor(Trommel, Dir.Off);
            fl.pause(1000);

            System.out.println("Trocknen");
            Trocknen(Dir.Right);
            Trocknen(Dir.Left);
            fl.setMotor(Anzeige, Dir.Off);
        } while (!fl.finish());
    }
```

Und der zugehörenden Sub Trocknen. Hier wird der Parameter Richtung zu Wechsel der Drehrichtung genutzt.

```
    private void Trocknen(int Richtung) {
        fl.setMotor(Trommel, Richtung, Speed.L1);
        fl.pause(10000);
        fl.setMotor(Trommel, Dir.Off);
        fl.pause(1000);
    }
```

Schiebetür



Schiebetür mit Lichtschranke

1 : Unbedingtes Schließen der Tür

2 : Öffnen der Tür, wenn Lichtschranke unterbrochen. Schließen nach 10 Sekunden

3 : Zusätzlich beim Schließen der Tür : Abbruch, wenn Lichtschranke unterbrochen, wieder öffnen, nach 5 Sekunden erneuter Versuch

Version 1 :

```
    public void Schiebetuer1() {
        System.out.println("Tür wird geschlossen");
        fl.setMotor(Tuer, Schliessen, Speed.Half);
        fl.waitForInput(TuerZu);
        fl.setMotor(Tuer, Dir.Off);
        fl.pause(1234);
    }
```

eher einfach

Version 2 :

```
    public void Schiebetuer2() {
        fl.setMotor(LLampe, Dir.On);
        fl.pause(500);
        fl.setMotor(Tuer, Schliessen);
        fl.waitForInput(TuerZu);
        fl.setMotor(Tuer, Dir.Off);

        do {
            fl.waitForInput(Lichtschranke, Unterbrochen);
            fl.setMotor(Tuer, Oeffnen);
            fl.waitForInput(TuerOffen);
            fl.setMotor(Tuer, Dir.Off);

            fl.setMotor(Tuer, Schliessen);
            fl.waitForInput(TuerZu);
            fl.setMotor(Tuer, Dir.Off);
        } while (!fl.finish());
    }
```

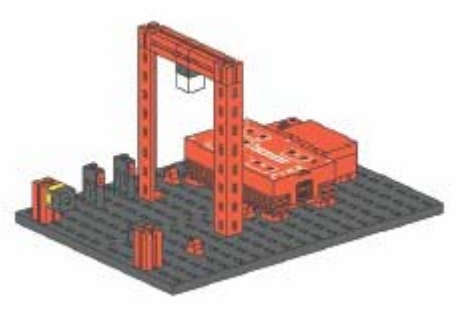
Erstmal Tür schließen und dann im Endlos-Loop auf `Lichtschranke Unterbrochen` warten. Tüer öffnen und nach 10 Sekunden wieder schließen.

Version 3 :

```
    public void Schiebetuer3() {
        fl.setMotor(LLampe, Dir.On);
        fl.pause(500);
        fl.setMotor(Tuer, Schliessen);
        fl.waitForInput(TuerZu);
        fl.setMotor(Tuer, Dir.Off);

        do {
            fl.waitForInput(Lichtschranke, Unterbrochen);
            fl.setMotor(Tuer, Oeffnen);
            fl.waitForInput(TuerOffen);
            fl.setMotor(Tuer, Dir.Off);
            fl.pause(10000);

            fl.setMotor(Tuer, Schliessen);
            while (!fl.getInput(TuerZu) & !fl.finish()) {
                if (fl.getInput(Lichtschranke) == Unterbrochen) {
                    fl.setMotor(Tuer, Oeffnen);
                    fl.waitForInput(TuerOffen);
                    fl.setMotor(Tuer, Dir.Off);
                    fl.pause(5000);
                    fl.setMotor(Tuer, TuerZu);
                }
            }
            fl.setMotor(Tuer, Dir.Off);
        } while (!fl.finish());
    }
```

Hier wird die Tür nicht mehr einfach nach 10 Sekunden geschlossen. Bei Schliessen wird ständig nach Lichtschranke = Unterbrochen gefragt und dann ggf. die Tüer wieder geöffnet.

Treppenhausbeleuchtung



Treppenhausbeleuchtung mit Tastern und Lichtschranke.

1: Wenn Taster I1 oder I2 gedrückt wird, Beleuchtung für 10 Sekunden anschalten.

2 : Einschalten zusätzlich durch Unterbrechung der Lichtschranke

3 : Wechselschalter : Einschalten und Ausschalten über Taster.

Version 1 :

```
    public void Treppenhaus1() {
        do {
            if (fl.getInput(Taster1) | fl.getInput(Taster2)) {
                fl.setMotor(Mot.M1, Dir.On);
                fl.pause(10000);
            }
            else fl.setMotor(Mot.M1, Dir.Off);
        } while (!fl.finish());
    }
```

Wenn einer der Taster an I1 oder I2 betätigt wird, für 10 sekunden Einschalten

Version 2 :

```
    public void Treppenhaus2() {
        fl.setMotor(Mot.M2, Dir.On);
        fl.pause(500);
        do {
            if (fl.getInput(Inp.I1) | fl.getInput(Inp.I2)
                                    | !fl.getInput(Inp.I3) |
!fl.finish()) {
                fl.setMotor(Mot.M1, Dir.On);
                fl.pause(10000);
            }
            else fl.setMotor(Mot.M1, Dir.Off);
        } while (!fl.finish());
    }
```

Wie 1 : Die Abfrage wurde um `!fl.getInput(Inp.I3)` (Lichtschranke) erweitert.

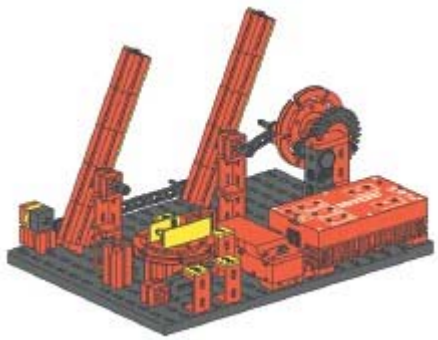
Version 3 :

```
    public void Treppenhaus3() {
        do {
            waitForOff();
            waitForOn();
            fl.setMotor(Beleuchtung, Dir.On);
            waitForOff();
            waitForOn();
            fl.setMotor(Beleuchtung, Dir.Off);
        } while (!fl.finish());
    }
```

```
private void waitForOn() {
    while (!(fl.getInput(Taster1) | fl.getInput(Taster2)
                                  | fl.finish())) {
       fl.pause(50);
    }
}
private void waitForOff() {
   fl.waitForInput(Taster1, false);
   fl.waitForInput(Taster2, false);
}
```

Wechselschalter (Ein- Ausschalten an beliebigen Schaltern) : Zum Erkennen werden die Subs WaitForOn und WaitForOff eingesetzt.

Scheibenwischer



Scheibenwischer, der über einen Drehschalter in den Betriebsarten Interval, Langsam und Schnell betrieben werden kann.

1 : Drehschalter auf Stellung 1 : Schnellgang

2 : Drehschalter 1 Langsam, 2 Schnell

3 : Drehschalter 1 Interval , 2 Langsam, 3 Schnell

Version 1 : Taster 1 an - Scheibenwischer im Schnellgang

```
public void Scheibenwischer1() {
    do {
        if (fl.getInput(Inp.I1)) fl.setMotor(Mot.M1, Dir.Right);
        else fl.setMotor(Mot.M1, Dir.Off);
    } while (!fl.finish());
}
```

Version 2 Taster 1 an - Langsamgang, Taster 1 und 2 an : Langsamgang

```
public void Scheibenwischer2() {
    do {
        if (fl.getInput(Inp.I1) & fl.getInput(Inp.I2)) {
            fl.setMotor(Mot.M1, Dir.Right);
        } else if (fl.getInput(Inp.I1)) {
            fl.setMotor(Mot.M1, Dir.Right, Speed.L2);
        } else fl.setMotor(Mot.M1, Dir.Off);
    } while (!fl.finish());
}
```

Version 3 :

```
public void Scheibenwischer3() {
    fl.setMotor(Mot.M2, Dir.On);
    fl.pause(500);
    do {
        if (fl.getInput(Inp.I1) & fl.getInput(Inp.I2)) {
            fl.setMotor(Mot.M1, Dir.Right, Speed.L2);
        } else if (fl.getInput(Inp.I1) & !fl.getInput(Inp.I2)) {
            fl.setMotor(Mot.M1, Dir.Right, Speed.L2);
            fl.waitForChange(Inp.I3, 4);
            fl.setMotor(Mot.M1, Dir.Off);
            fl.pause(2000);
        } else if (!fl.getInput(Inp.I1) & fl.getInput(Inp.I2)) {
            fl.setMotor(Mot.M1, Dir.Right);
        } else fl.setMotor(Mot.M1, Dir.Off);
    } while (!fl.finish());
}
```

Taster 1 an und Taster 2 aus (Pos 1) : Intervalschaltung
Taster 1 und 2 an (Pos 2) : Langsamgang
Hier wurde anstelle der drei WaitForInput von Robo Light die Methode WaitForChange eingesetzt, die die Intervalsteuerung besser löst. Hier mit vier Wechseln Lichtschranke True/False bzw. False/True.
Taster 1 aus und Taster 2 an (Pos 3) : Schnellgang