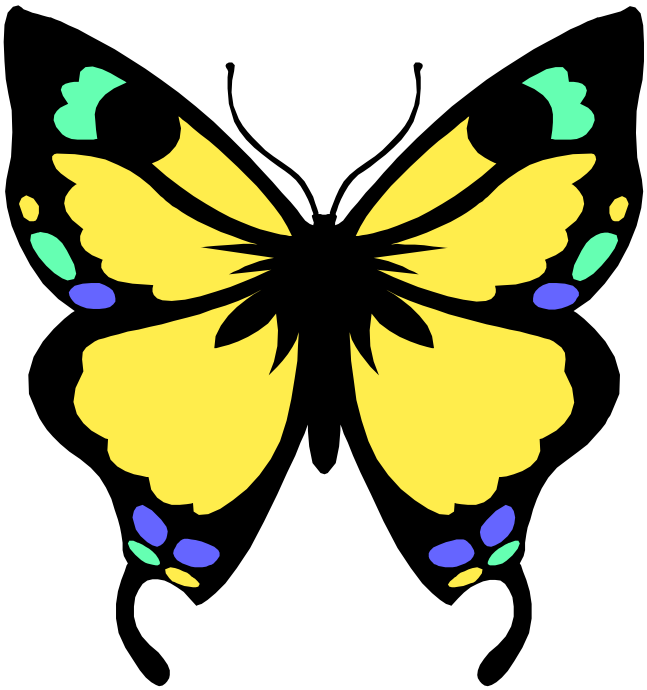
Robo Pro Light -> Python 3.1.1

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

Python 3.1.1

enthält das Python-System mit Dokumentation einschließlich der Entwicklungsumgebung IDLE und der Python Command Line.

www.python.org/download

roboLightPython.ZIP

Enthält die Sources der Beispielprogramme und die erforderlichen DLLs

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 roboLightPython.ZIP. In ..\System32 bzw. ..\SystemWOW64 (Windows 7/64) unterbringen.

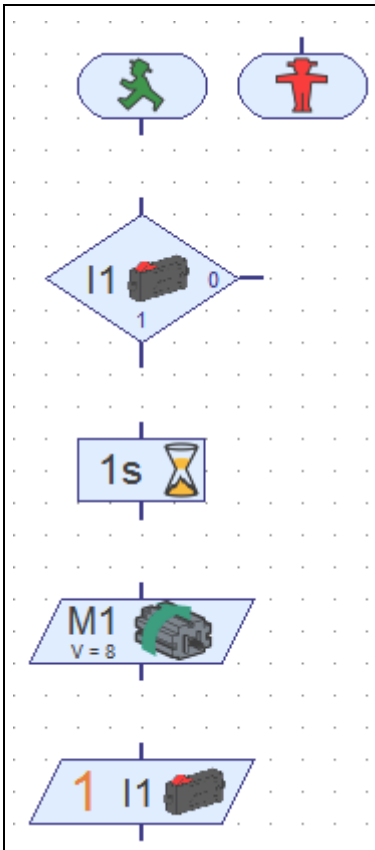
Klasse FishFace

Die Source FishFa40.PY aus roboLightPython.ZIP im Python-Pfad Verzeichnis ..\Lib untebringen.

In der nutzenden Source ist ein `from FishFa40 import *` erforderlich.

Zusätzlich sinnvoll www.ftcomputing.de/pythonecke.htm

Gegenüberstellung ROBO Pro Light - Python-Befehle

	Python-Befehl in Spalte 1
	
	`if fl.GetInput(1):`
	# --- möglich : Inp.I1, I2, I3 ---
	
	`fl.Pause(1000)` #' --- Zeitangabe in Millisekunden
	
	`fl.SetMotor(1, fl.Rechts[, speed]);`
	# --- möglich : 1, 2, fl.Ein (Lampen)
	# fl.Links, Rechts, Aus. speed 1 … 7
	# Bei speed = 7 kann es auch entfallen
	`fl.WaitForInput(1, True);`
	# --- True kann auch entfallen, alternativ False
	Warten auf einen Sensoreingang.

Die mit fl. beginnenden Methoden sind Teil der Klasse FishFace aus FishFa40.py. Weitere Dokumentation dazu findet man in ftComputing für Python http://ftcomputing.de/pdf/FishFa40Py311.pdf.

Programmrahmen

Hinweis : Der LT Controller muß vor dem Start eines Programmes am Rechner angeschlossen sein.

```
from FishFa40 import *

mFahrMotor   = 1
iStartTaster = 1

def Karussel1():
  "Einfache Runde"
  print("Zum Start I1-Taster drücken")
  fl.WaitForInput(iStartTaster)
----------

fl = FishFace()

print ("Karussel wird gestartet")
fl.OpenInterface()
Karussel3()
print("Karussel wird beendet")
fl.CloseInterface()
```

`from FishFa40 import`: Zugriff auf die Source FishFa40.py in <python>\Lib.

`mFahrMotor`: Verwendungszweck der Eingänge des LT Controllers (m : M-Ausgang, i : I-Eingang, d : Drehrichtung)

`def Karussel :` Verwendete Unterprogramme. Müssen vor dem "Hauptprogramm" liegen

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

`print("….")` : Beginn des Hauptprogramms

`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 :

```
def Karussel1():
  "Einfache Runde"
  print("Zum Start I1-Taster drücken")
  fl.WaitForInput(iStartTaster)
  fl.SetMotor(mFahrMotor, fl.Rechts)
  fl.Pause(10000)
  fl.SetMotor(mFahrMotor, fl.Aus)
```

Warten auf Starttaster, Einschalten des Motors, 10 Sekunden warten, Ausschalten des Motors. Diese Befehle ersetzen def Karussel1() des Programmrahmens.

Version 2 :

```
def Karussel2():
  "Endlos Runde"
  while not fl.Finish():
    print("Neue Runde : I1-Taster drücken")
    fl.WaitForInput(iStartTaster)
    fl.SetMotor(mFahrMotor, fl.Rechts)
    fl.Pause(10000)
    fl.SetMotor(mFahrMotor, fl.Aus)
```

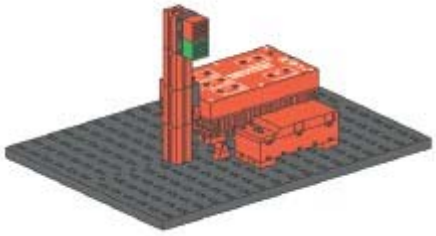
Wie 1, jetzt aber in einer "endlosen" `while` Schleife, die über die ESC-Taste der Tastatur abgebrochen werden kann (Führe die Befehle der Einrückung nach while aus bis feststellt, wird, dass eine ESC-Taste gedrückt wurde).

Version 3 :

```
def Karussel3():
  "Endlos Runde und zurück"
  while not fl.Finish():
    print("Neue Runde : I1-Taster drücken")
    fl.WaitForInput(iStartTaster)
    fl.SetMotor(mFahrMotor, fl.Rechts)
    fl.Pause(10000)
    fl.SetMotor(mFahrMotor, fl.Aus)
    fl.Pause(1000)
    fl.SetMotor(mFahrMotor, fl.Links)
    fl.Pause(10000)
    fl.SetMotor(mFahrMotor, fl.Aus)
```

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 :

```
def Ampel1():
    "Anforderung Fußphase"
    fl.SetMotor(mRot, fl.Ein)
    while not fl.Finish():
        if fl.GetInput(iAnforderung):
            print("Signal kommt")
            fl.Pause(5000)
            fl.SetMotor(mRot, fl.Aus)
            fl.SetMotor(mGruen, fl.Ein)
            fl.Pause(10000)
            fl.SetMotor(mRot, fl.Ein)
            fl.SetMotor(mGruen, fl.Aus)
    fl.SetMotor(mRot, fl.Aus)
```

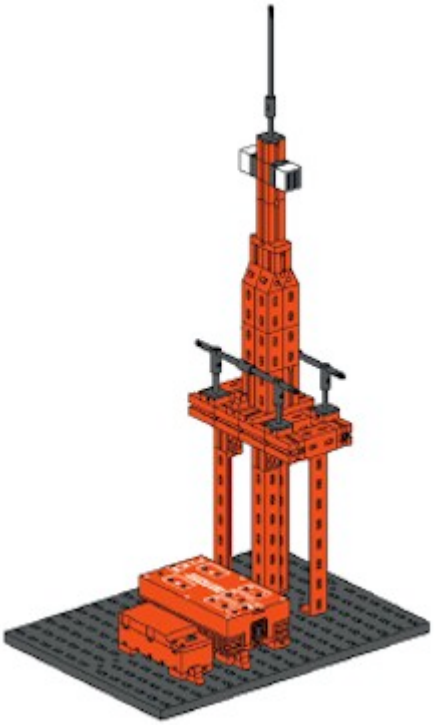
Einschalten Rot, Endlosschleife mit Warten auf Anforderung Fußphase (`If fl.GetInput`..), bei Erkennen : Nachricht, Warten 5 Sekunden, Rot Aus, Grün An für 10 Sekunden, dann wieder Rot an, Grün aus.

Version 2 :

```
def Ampel2():
    "Fußphase mit Grünblinken"
    fl.SetMotor(mRot, fl.Ein)
    while not fl.Finish():
        if fl.GetInput(iAnforderung):
            print("Signal kommt")
            fl.Pause(5000)
            fl.SetMotor(mRot, fl.Aus)
            fl.SetMotor(mGruen, fl.Ein)
            fl.Pause(10000)
            for i in range(1,3):
                fl.SetMotor(mGruen, fl.Aus)
                fl.Pause(1000)
                fl.SetMotor(mGruen,  fl.Ein)
                fl.Pause(1000)
            fl.SetMotor(mRot, fl.Ein)
            fl.SetMotor(mGruen, fl.Aus)
    fl.SetMotor(mRot, fl.Aus)
```

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 :

```
def Leuchtturm1():
  "Gleichtaktfeuer"
  while not fl.Finish():
    fl.SetMotor(mLicht1, fl.Ein)
    fl.SetMotor(mLicht2, fl.Ein)
    fl.Pause(2000)
    fl.SetMotor(mLicht1, fl.Aus)
    fl.SetMotor(mLicht2, fl.Aus)
    fl.Pause(2000)
```

Version 2 :

```
def Leuchtturm2():
  "Blitzprinzip"
  while not fl.Finish():
    fl.SetMotor(mLicht1, fl.Ein)
    fl.SetMotor(mLicht2, fl.Ein)
    fl.Pause(300)
    fl.SetMotor(mLicht1, fl.Aus)
    fl.SetMotor(mLicht2, fl.Aus)
    fl.Pause(1500)
```

Wie gehabt, aber mit anderen Zeiten.

Version 3 - ist haarig :

```
def Lampe2Blinken(n):
  while not fl.Finish():
    fl.SetMotor(mLicht2, fl.Ein)
    fl.Pause(3000)
    fl.SetMotor(mLicht2, fl.Aus)
    fl.Pause(5000)

def Leuchtturm3():
  "Blinkprinzip"
  _thread.start_new_thread(Lampe2Blinken, (0,))
  while not fl.Finish():
    fl.SetMotor(mLicht1, fl.Ein)
    fl.Pause(2000)
    fl.SetMotor(mLicht1, fl.Aus)
    fl.Pause(3000)
```

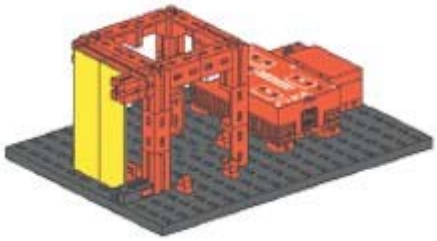
Mit Robo Pro ist das deutlich einfacher : Man benötigt dazu nur ein zweites grüne Männchen

Mit Python geschieht das über Threading (mehrere unabhängige Programmzweige), ist aber doch noch recht friedlich :

```
import _thread
  _thread.start_new_thread(Lampe2Blinken, (0,))
```

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) :

```
def Kuehlschrank1():
    "Weiße Lampe"
    while not fl.Finish():
        if fl.GetInput(iOffen):
            fl.SetMotor(mWeiss, fl.Aus)
        else:
            fl.SetMotor(mWeiss, fl.Ein)
    fl.SetMotor(mWeiss, fl.Aus)
```

`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 `not` tun. Man kann aber auch einfach `fl.GetInput(iOffen) == False` schreiben.

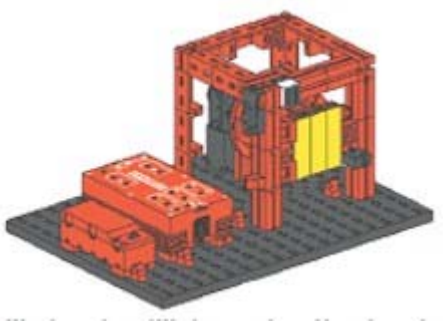
Version 2 (wieder eine Thread-Lösung) - Funktioniert genauso wie die Leuchtturm-Lösung.

```
def RotBlinken(n):
    "Rotblinken nach 3 sec"
    while not fl.Finish():
        fl.WaitForInput(iOffen, False)
        fl.Pause(3000)
        while (not(fl.Finish() or fl.GetInput(iOffen))):
            fl.SetMotor(mRot, fl.Ein)
            fl.Pause(500)
            fl.SetMotor(mRot, fl.Aus)
            fl.Pause(500)
        fl.SetMotor(mRot, fl.Aus)

def Kuehlschrank2():
    "Weiß und Rot"
    _thread.start_new_thread(RotBlinken, (0,))
    while not fl.Finish():
        if fl.GetInput(iOffen) == False:
            fl.SetMotor(mWeiss, fl.Ein)
        else :
            fl.SetMotor(mWeiss, fl.Aus)
    fl.SetMotor(mWeiss, fl.Aus)
```

Es wird endlos auf das Öffnen der Kühlschranktür gewartet. Wird sie geöffnet, wird nach 3 Sekunden Rot geblinkt. Der `while` Loop 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 :

```
def Waschmaschine1():
    "Starter : Trommel langsam 10 sec"
    while not fl.Finish():
        print("Start Waschgang : I1-Taster")
        fl.WaitForInput(iStarter)
        fl.SetMotor(mAnzeige, fl.Ein)
        fl.SetMotor(mTrommel, fl.Rechts, cLangsam)
        fl.Pause(10000)
        fl.SetMotor(mTrommel, fl.Aus)
        fl.SetMotor(mAnzeige, fl.Aus)
        fl.Pause(1000)
```

Version 2 :

```
def Waschmaschine2():
    "Starter : Trommel langsam 10 sec, Tür zu"
    while not fl.Finish():
        print("Start Waschgang : I1-Taster")
        fl.WaitForInput(iTuerZu)
        fl.WaitForInput(iStarter)
        fl.SetMotor(mAnzeige, fl.Ein)
        fl.SetMotor(mTrommel, fl.Rechts, cLangsam)
        fl.Pause(10000)
        fl.SetMotor(mTrommel, fl.Aus)
        fl.SetMotor(mAnzeige, fl.Aus)
        fl.Pause(1000)
```

In `while` wird auf den Stattaster und TuerZu gewartet.

Version 3 :

Mit dem versprochenen Schleudergang :

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

Version 4 :

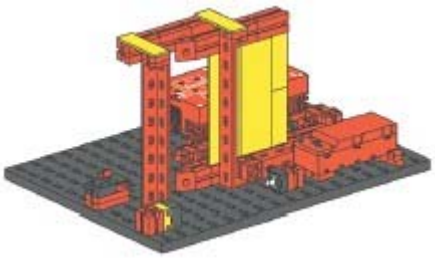
Mit Trockengang und Textausgabe zur Anzeige des aktuellen Waschganges.

```
def Trocknen(richtung):
    "Trockengang in Richtung"
    fl.SetMotor(mTrommel, richtung, cLangsam)
    fl.Pause(10000)
    fl.SetMotor(mTrommel, fl.Aus)
    fl.Pause(1000)

def Waschmaschine4():
    "Starter : Trommel 10 sec, Tür zu, Schleudergang, Trocknen"
    while not fl.Finish():
        print("Start Waschgang : I1-Taster")
        if not fl.GetInput(iTuerZu):
            print("Tür ZU")
            fl.Pause(555)
            continue
        fl.WaitForInput(iStarter)
        print("Waschen")
        fl.SetMotor(mAnzeige, fl.Ein)
        fl.SetMotor(mTrommel, fl.Rechts, cLangsam)
        fl.Pause(10000)
        fl.SetMotor(mTrommel, fl.Aus)
        fl.Pause(1000)
        print("Schleudern")
        fl.SetMotor(mTrommel, fl.Rechts, cVoll)
        fl.Pause(15000)
        fl.SetMotor(mTrommel, fl.Aus)
        fl.Pause(1000)
        print("Trocknen")
        Trocknen(cRechts)
        Trocknen(cLinks)
        fl.SetMotor(mAnzeige, fl.Aus)
```

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

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 :

```
def Schiebetuer1():
    "Tür unbedingt schliessen"
    print("Tür wird geschlossen")
    fl.SetMotor(mTuer, dSchliessen, cMittel)
    fl.WaitForInput(iTuerZu)
    fl.SetMotor(mTuer, fl.Aus)
```

eher einfach

Version 2 :

```
def Schiebetuer2():
    "Öffnen durch Lichtschranke, Schließen nach 10 sec"
    fl.SetMotor(mLLampe, fl.Ein)
    fl.Pause(500)
    fl.SetMotor(mTuer, dSchliessen)
    fl.WaitForInput(iTuerZu)
    fl.SetMotor(mTuer, fl.Aus)
    while not fl.Finish():
        fl.WaitForInput(iLichtschranke, cUnterbrochen)
        fl.SetMotor(mTuer, dOeffnen)
        fl.WaitForInput(iTuerOffen)
        fl.SetMotor(mTuer, fl.Aus)
        fl.Pause(10000)
        fl.SetMotor(mTuer, dSchliessen)
        fl.WaitForInput(iTuerZu)
        fl.SetMotor(mTuer, fl.Aus)
```

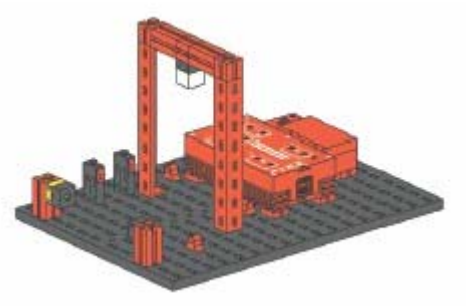
Erstmal Tür schließen und dann im Endlos-Loop auf `iLichtschranke cUnterbrochen` warten. Tüer öffnen und nach 10 Sekunden wieder schließen.

Version 3 :

```
def Schiebetuer3():
    "Türbetrieb : Beim Schließen Lichtschranke beachten"
    fl.SetMotor(mLLampe, fl.Ein)
    fl.Pause(500)
    fl.SetMotor(mTuer, dSchliessen)
    fl.WaitForInput(iTuerZu)
    fl.SetMotor(mTuer, fl.Aus)
    while not fl.Finish():
        fl.WaitForInput(iLichtschranke, cUnterbrochen)
        fl.SetMotor(mTuer, dOeffnen)
        fl.WaitForInput(iTuerOffen)
        fl.SetMotor(mTuer, fl.Aus)
        fl.Pause(10000)
        fl.SetMotor(mTuer, dSchliessen)
        while (not fl.GetInput(iTuerZu) and not fl.Finish()):
            if not fl.GetInput(iLichtschranke):
                fl.SetMotor(mTuer, dOeffnen)
                fl.WaitForInput(iTuerOffen)
                fl.SetMotor(mTuer, fl.Aus)
                fl.Pause(5000)
                fl.SetMotor(mTuer, dSchliessen)
        fl.SetMotor(mTuer, fl.Aus)
```

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 :

```
def Treppenhaus1():
    "Taster : Einschalten für 10 sec"
    while not fl.Finish():
        if (fl.GetInput(iTaster1) or fl.GetInput(iTaster2)):
            fl.SetMotor(mBeleuchtung, fl.Ein)
            fl.Pause(10000)
        else:
            fl.SetMotor(mBeleuchtung, fl.Aus)
```

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

Version 2 :

```
def Treppenhaus2():
    "Taster, Lichtschranke : Einschalten für 10 sec"
    fl.SetMotor(mLLampe, fl.Ein)
    fl.Pause(500)
    while not fl.Finish():
        if (fl.GetInput(iTaster1) or fl.GetInput(iTaster2) \
            or not fl.GetInput(iLichtschranke)):
           fl.SetMotor(mBeleuchtung, fl.Ein)
           fl.Pause(10000)
        else:
            fl.SetMotor(mBeleuchtung, fl.Aus)
```

Wie 1 : Die Abfrage wurde um `!fl.GetInput(iLichtschranke)` erweitert.

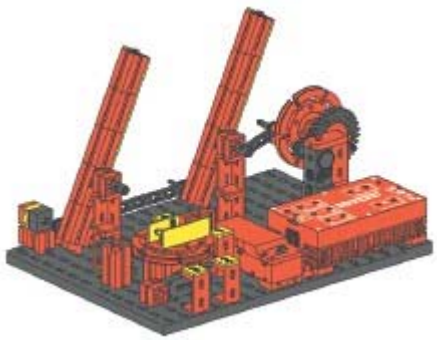
Version 3 :

```
def WaitForOn():
    while not fl.Finish():
        if (fl.GetInput(iTaster1) or fl.GetInput(iTaster2)):
            break
        fl.Pause(50)
def WaitForOff():
    fl.WaitForInput(iTaster1, False)
    fl.WaitForInput(iTaster2, False)

def Treppenhaus3():
    "Wechselschalter"
    while not fl.Finish():
        WaitForOff()
        WaitForOn()
        fl.SetMotor(mBeleuchtung, fl.Ein)
        WaitForOff()
        WaitForOn()
        fl.SetMotor(mBeleuchtung, fl.Aus)
```

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

```
def Scheibenwischer1():
    "Schnellgang : Drehschalter Pos 1"
    while not fl.Finish():
        if fl.GetInput(iTaster1):
            fl.SetMotor(mWischer, fl.Rechts);
        else: fl.SetMotor(mWischer, fl.Aus);
```

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

```
def Scheibenwischer2():
    "Pos 1 : Langsam, Pos 2 : Schnell"
    while not fl.Finish():
        if fl.GetInput(iTaster1)and fl.GetInput(iTaster2):
            fl.SetMotor(mWischer, fl.Rechts);
        elif fl.GetInput(iTaster1):
            fl.SetMotor(mWischer, fl.Rechts, cLangsam)
        else: fl.SetMotor(mWischer, fl.Aus);
```

Version 3 :

```
def Scheibenwischer3():
    "Pos 1 : Interval, 2 : Langsam, 3 : Schnell"
    fl.SetMotor(mLLampe, fl.Ein)
    fl.Pause(500)
    while not fl.Finish():
        if fl.GetInput(iTaster1) and fl.GetInput(iTaster2):
            fl.SetMotor(mWischer, fl.Rechts, cLangsam);
        elif fl.GetInput(iTaster1) and not fl.GetInput(iTaster2):
            fl.SetMotor(mWischer, fl.Rechts, cLangsam)
            fl.WaitForChange(iLichtS, 4)
            fl.SetMotor(mWischer, fl.Aus)
            fl.Pause(2000)
        elif not fl.GetInput(iTaster1) and fl.GetInput(iTaster2):
            fl.SetMotor(mWischer, fl.Rechts)
        else: fl.SetMotor(mWischer, fl.Aus);
```

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