Python-Ecke

ftComputing : Programme für die fischertechnik-Interfaces und -konstruktionskästen

ftComputing.de

Home
Back
FishStep

Sitemap
Index
Links
Impressum

Mail

Allgemeines

Python ist eine Sprache, die von Praktikern für Praktikern entwickelt wurde. Sie läßt alldas linksliegen über das man bei Java so schön und weltanschaulich diskutieren kann. Variablen-Deklarationen : wozu?, explizite Typkonversionen warum, Klammernzählen nee : Man muß halt ein wenig aufpassen, wenn mans bequem haben will. Trotzdem gibts OO-satt, sogar mehrfach. Und weil Python sich nun mal so einfach und kostenlos anläßt sieht man auch Möglichkeiten für den Einsatz im Informatik-Unterricht an Schulen.

Diese Seite gibt einen Überblick auf die Zugriffsmöglichkeiten aus Python auf die fischertechnik Interfaces. Sie bietet keine Einführung in Python.

Dazu werden zwei Alternativen angeboten :

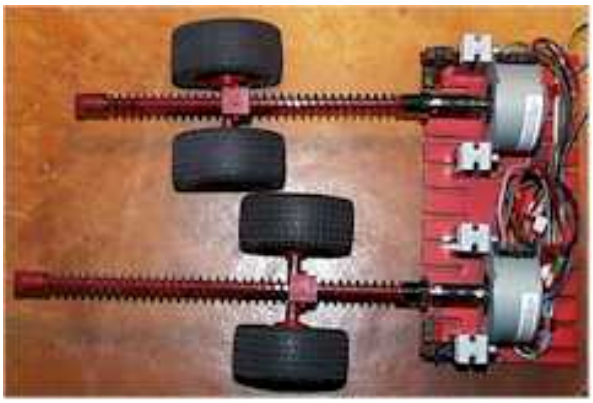
1. Die ActiveX.DLL **FishFa30.DLL**
2. Der Python Modul **FishFa30.py**

Beide Alternativen basieren auf der **umFish30.DLL**

Die **FishFa30.DLL-Alternative** ist insgesamt die komfortablere, da sie auch von sich aus für die Unterbrechbarkeit der Anwendungsprogramme sorgt. Es muß aber allerhand installiert werden.

Die **FishFa30.py-Alternative** erwartet außer einem installierten Python-Sytem nur noch die Installation von T.Hellers ctypes. Sie ist auch technisch interessanter, da hier der Zugriff auf umFish30.DLL

offen üer die Python-Klasse FishFace geschieht. Hier kommen noch die Klassen **FishRobot** zur Programmierung von Industry Robots und die Klasse **FishStep** zur Programmierung von Schrittmotoren hinzu.



Der Funktionsumfang der Basis-Klasse FishFace ist in beiden Fällen gleich.

Downloads

Außer einem installierten Python-System ab v2.2 werden weitere Komponenten benötigt. Das ist zunächst das Download-Päckchen PythonFish30.ZIP mit dem Modul FishFa30.py, einem Handbuch und einigen Beispielen.

Im beiliegenden Handbuch wird dann die Installation der zusätzlich erforderlichen Komponenten (win32all / FishFa30.DLL bzw. ctypes / umFish30.DLL) beschrieben. Die Download-Adressen sind dort ebenfalls zu finden.

Programmierung

Hier ein paar kleine Beispiele, die entweder an der "Python Shell" (von IDLE) direkt ausgeführt wurden oder über ein mit IDLE erstelltes Programm (wenns dann doch zu unübersichtlich wurde). Es wurde die Alternative 2 genutzt. Ausführliche Programmier-Beispiele finden sich im Handbuch von PythonFish30.ZIP. Bis auf die import und die Instanzierungszeile besteht zwischen den Alternativen keine gravierenden Unterschiede :

```
import win32com.client
ft =
win32com.client.Dispatch("FishFa30.FishFace")
```

Bei Alternative 2 hieße es dann :

```
from FishFa30 import *
ft = FishFace()
```

In beiden Fällen ist dann ein Open und bei Ende der Experimente ein Close erforderlich :

```
ft.OpenInterface("COM1")
```

```
ft.CloseInterface()
```

Der Portname ist auf die eigenen Gegebenheiten zu ändern ("LPT", "COM1" - "COM8", "LPT1" - "LPT3")

Blinker

Lampe an M1 blinkt im Sekundentakt :

```
Ein, Aus = 1, 0

while not ft.Finish():
  ft.SetMotor(1, Ein)
  ft.Pause(555)
  ft.SetMotor(1, Aus)
  ft.Pause(333)
```

Beenden duch ESC-Taste

WechselBlinker

Lampen an M1 und M2 blinken im Wechsel :

```
while not ft.Finish():
  ft.SetMotors(0x1)
  ft.Pause(444)
  ft.SetMotors(0x4)
  ft.Pause(444)
```

Hier werden alle M-Ausgänge gleichzeitig geschaltet. Jeweils zwei bit pro M-Ausgang. Also 00000001 für M1 Ein und 00000100 für M2 ein. Alle anderen

Ausgänge sind Aus.

StatusAnzeige

Der Schaltzustand eines ausgewählten E-Eingangs (hier E3) wird angezeigt (Abbruch durch ESC-Taste):

```
while not ft.Finish():
  print "Status E3 : ", ft.GetInput(3)
  ft.Pause(1000)
```

Anzeige 1 steht für ein, Anzeige im Sekundentakt.

Der Schaltzustand aller E-Eingänge wird permanent angezeigt :

```
while not ft.Finish():
  print "Status der E-Eingaenge : ",
hex(ft.GetInputs())
  ft.Pause(1234)
```

Die Anzeige erfolgt Hexa im Sekundentakt.

Analoganzeige

Der Wert von EX wird laufend angezeigt :

```
while not ft.Finish():
  print "Wert EX : ", ft.GetAnalog(0)
  ft.Pause(1234)
```

Dabei ist zu beachten, dass das Open mit ft.OpenInterface("COM1", 1) für "mit AnalogScan" erfolgen muß.
Achtung vorher mit ft.CloseInterface() schließen, wenn bereits offen.

Nutzen der FishRobot-Funktionen

Bei Verwendung der Klasse FishRobot anstelle von FishFace stehen zusätzich noch einige Robot-Methoden zur Verfügung :

```
from FishFa30 import *
ft = FishRobot([[3,222],[4,88]])

ft.OpenInterface("COM2")

ft.Home()
```

```
ft.MoveTo(ft.PosPrint, [23,34])
ft.MoveTo(None, [23,34])
ft.MoveTo(ft.PosPrint, [23,23])


ft.CloseInterface()
```

Bei der Instanzierung wird die Roboter-Konfiguration mitgeteilt Motore an M3 und M4 mit maximalem Fahrweg ab Endtaster von 222 bzw. 88 Impulsen.
Nach dem Open wird die Home-Position (die Position an den Endtastern) angefahren und der interne Positionszähler in ft.MotCntl auf 0 gesetzt.
Anschließend wird auf die Position M3 = 23 und M4 = 34 gefahren. Die laufende Position wird über PosPrint auf der Konsole ausgegeben. Dann wird nochmal die gleiche Position angefahren – um zu zeigen, daß da nichts ruckelt – und dann die Position M4 = 23, also wieder in Richtung Endtaster. Siehe auch "Anmerkungen zu den Rob-Funktionen".

Nutzen der FishStep-Funktionen

Beispiel siehe Elefanten-Runde

Stand : 05.09.2003

FishStep

ftComputing : Programme für die fischertechnik-Interfaces und -konstruktionskästen

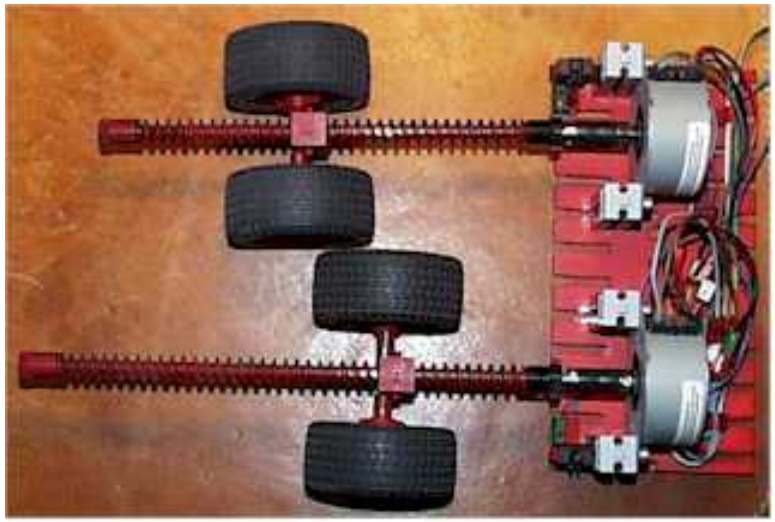
ftComputing.de

Home
Back

Sitemap
Index
Links
Impressum

Mail

FishStep : Schrittmotoren



Die Klasse FishStep des Python Moduls FishFa30 ermöglicht die Programmierung von Schrittmotoren. Einzeln oder im XY-Verbund.
Dazu gibt es die Funktionen :
Einzelner Schrittmotor (an zwei nebeneinanderliegenden M-Ausgängen) :
- StepHome(MotNr)
- StepTo(PosHook, MotNr, Xabs)
- StepDelta(PosHook, MotNr, Xrel)

Zwei Schrittmotoren im XY-Verbund (an drei nebeneinanderliegenden M-Ausgängen):
- PlotHome(MotNr)
- PlotTo(PosHook, MotNr, Xabs, Yabs)
- PlotDelta(PosHook, MotNr, Xrel, Yrel)

Die Positionierungsangaben erfolgen in Zyklen (a 4 Steps von 7,5°) gerechtet ab zugehörenden Endtaster (StepTo, PlotTo) oder bezogen auf die aktuelle Position (StepDelta, PlotDelta). Mit

StepHome/PlotHome wird die Nullposition angefahren.

Beispiel Zeichnen eines Quadrates :

```
from FishFa30 import *


def Vieleck(RelList):
    for xy in RelList: ft.PlotDelta(None, 1, xy[0], xy[1])

Plotter = 1
ft = FishStep(((1,333),(3,222)))
ft.OpenInterface("COM2")


ft.PlotHome(Plotter)
ft.PlotTo(None, Plotter, 50, 50)
Vieleck(((50,0),(0,50),(-50,0),(0,-50)))


ft.CloseInterface()
```

Die Schrittmotoren sind an M1-M3 angeschlossen, der maximale Fahrweg beträgt 333/222 Zyklen in X/Y-Richtung (ft = FishStep)
Mit PlotHome wird die Null-Stellung angefahren. PlotTo fährt zur linken unteren Ecke des zu zeichnenden Quadrats (Koordinaten 50/50 bezogen auf den Nullpunkt). Anschließen wird mit PlotDelta über die Funktion Vieleck ein Quadrat mit 50 Zyklen Kantenlänge gezeichnet.

Wenn man denn gerade keinen Plotter auf dem Schreibtisch stehen hat und auch die Mühe scheut, schnell einen zu bauen, startet man ein Elefantenrennen :
An die ohnehin schon zu Testzwecken aufgebauten Schrittmotoren steckt man möglichst viel Schnecke und an die Schneckenmutter steckt man je zwei Jumbo-Räder. Das Quadratzeichnen kann beginnen :

1. Die Räder rollen einträglich, nebeneinander und gleichzeitig in Richtung Endtaster (vor den grauen Motorträgern). Ist einer eher da wartet er auf den anderen (PlotHome)

2. Es geht wieder 50 Zyklen zurück in Richtung Schneckenende, schön im Gleichschritt (PlotTo).
3. Und nun abwechselnd, mal der eine, mal der anderer jedoch immer 50 Zyklen, hin oder her.

Das wars (man könnte sie natürlich auch noch überkreuz stellen).

Beispiele und Handbuch sind in PythonFish30.ZIP enthalten.

Stand : 05.09.2003