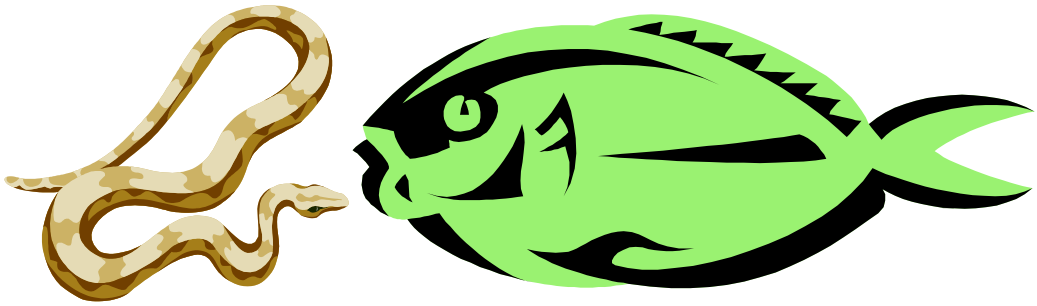
ftComputing

FishFaTX für Python

FishFaTX.py, umFish50.DLL und Python 3.1.1

Ulrich Müller



Inhaltsverzeichnis

Ulrich Müller, D-33100 Paderborn, Lange Wenne 18. Fon 05251/56873
eMail : um@ftcomputing.de
HomePage : www.ftComputing.de

Dokument : FishFaTXPy311.Doc, Druckdatum : 19.01.2010
Titelbild : Einfügen | Grafik | AusDatei | Office | Fish5.WMF

Übersichten

Allgemeines

Mit umFish50.DLL und der darauf aufbauenden Klasse FishFace wird die Möglichkeit geboten, fischertechnik Interfaces unter Python zu programmieren. FishFace erlaubt die Ansteuerung der ROBO Interfaces im "Online-Modus", also mit Verbindung zum PC.

Angeboten werden Befehle zur Schaltung der M-Ausgänge und zur Abfrage der Eingänge eines Interfaces. Dazu wird das Interface in von umFish50.DLL in regelmäßigen Abständen abgefragt. Zusätzlich werden die Veränderungen (Ein/Aus) an den I-Eingängen gezählt, sie werden außerdem zur Bestimmung der Schaltdauer der M-Ausgänge herangezogen. Dazu ist eine feste Zuordnung des an einen M-Ausgang angeschlossenen Motors und der an die I-Eingänge angeschlossenen Ende- und Impulstaster notwendig(RobMotoren). Die M-Ausgänge können außerdem mit verschiedener "Geschwindigkeit" betrieben werden, dazu werden sie in Intervallen ein- und ausgeschaltet (PWM).

Die Analog-Eingänge liefern Raw-Werte im Bereich von 0 – 1023.

fischertechnik Interfaces können über das
Python Modul FishFa50.py auf Basis von umFish50.DLL betrieben werden
Dies ist in Verbindung mit windll.umFish50… eine eher minimalistische Lösung. Sie hat den Vorteil transparant zu sein und kann den eigenen Wünschen angepaßt werden, da sie als Source vorliegt. Als Editor empfiehlt sich IDLE, es sind genauso auch andere möglich. Nachteil : FishFa50.py eignet sich weniger für GUI-Programme, da mit FishFa50.py geschriebene Programme von Haus aus nicht unterbrechbar (aber abbrechbar) sind, hier ist dann der Ablauf in einem Thread gefragt bzw. müssen Tk.update() eingestreut werden.

Hier wird der Einsatz des Moduls FishFa40.PY mit der Klasse FishFace und FishFaceException, beschrieben.

Installation Python 3.1.1 und FishFaTX

Python- 3.1.1
Enthält das Python-System mit Dokumentation einschließlich der Entwicklungsumgebung IDLE und der Python Command Line
www.python.org/download

PythonFish311.ZIP
Enthält dieses Handbuch, FishFaTX.py und Beispiele sowie umFish50.DLL.
umFish50.DLL sollte in ein zentrales Verzeichnis (\Windows\System32) kopiert werden
FishFaTX.py sollte nach \Python31\Lib kopiert werden.

www.ftcomputing.de/zip/pythonfish311.zip

Zusätzlich ist der ROBO TX Treiber, ftMscLib.DLL und das RoboTXTest-Tool von fischertechnik.de | Computing | Downloads | PC Programming ROBO-TXC 1.2 erforderlich.
ftMscLib.DLL sollte in das gleiche Verzeichnis wie umFish50.DLL kopiert werden.

Unterstützt wird der ROBO TX Controller mit bis zu 8 angeschlossenen Extensions

Referenz

Importe und Instanzierung

FishFaTX.PY

Bei Einsatz von FishFaTX.py ist der folgende import erforderlich :

```
from FishFaTX import *
```

Es wird dabei vorausgesetzt, das FishFaTX.py in einem zugreifbaren Pfad liegt (siehe Installierung).

Die Instanzierung erfolgt über :

```
tx = FishFace()
```

Anstelle von tx kann natürlich ein beliebiger anderer Name gewählt werden.

Fehlerbehandlung

Es wird empfohlen FishFace-Fehler über ein try ... except-Konstrukt abzufangen :

```
try:
  tx.OpenController()
except FishFaceException:
  print ("------- Controller Problem ------")
  sys.exit(0)
```

Für sys ist ein zusätzliches `import sys` erforderlich.

FishFaTX.PY

Allgemeines

bool	0 = false, 1 = true
int	32bit Integer mit Vorzeichen, Python Standard Wert

Verwendete Variablenbezeichnungen

Die Variablen sind durchweg vom Typ integer (32bit-Wert)

CounterNr	Impulszähler (Impulse der C-Eingänge (1 - 4) zwischen true und wieder true werden automatisch gezählt).
ctrID	Nummer des zutreffenden Controllers (0 = Main, 1 = Ext1 ….)
Direction	Drehrichtung eines Motors (Ein, Aus, Links, Rechts)
InputNr	Nummer eines Universal-Einganges (1 – 8, bei GetInput auch 9 - 12 für die Zählereingänge)
LampNr	Nummer eines O-Ausganges (1-8)
MotorNr, MotNr	Nummer eines M-Ausganges (1-4)
MotorNrs	Liste von zu überwachenden MotorNummern
mSek	Zeitangabe in MilliSekunden
OnOff	Schalten eines O-Ausganges (1 - 512 = ein, 0 = aus) unter Angabe der Power
Speed	Geschwindigkeit mit der ein M-Ausgang betrieben werden soll (1 - 512)
Wert	Beliebiger 32bit Integerwert

Symbolische Konstanten

Fehler	Allgemeiner FishFace-Fehler
Ein	Einschalten M-Ausgang
Aus	Ausschalten M-Ausgang
Links	Einschalten M-Ausgang linksdrehend
Rechts	Einschalten M-Ausgang rechtsdrehend

FishFace Methoden

- **ClearCounter**(ctrlD, InputNr)
  Löschen(0) des angegebenen Counters

- **CloseController**()
  Schließen der Verbindung zum Interface

bool **Finish**(ctrlD=0, InputNr=0)
Feststellen eines Endewunsches (ESC-Taste, I-Eingang (optional))
Beispiel :

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

Die while-Schleife (eine blinkende Lampe) läuft solange bis die ESC-Taste gedrückt wird

int **GetAnalog**(ctrlD, InputNr)
Lesen Analog-Wert (A5K-Eingang : NTC, Photowiderstand, Potentiometer)

int **GetDistance**(ctrlD, InputNr)
Auslesen des aktuellen Wertes in cm des Ultraschallsensors

int **GetCounter**(ctrlD, CounterNr)
Auslesen des Wertes des angebenen Counters
Beispiel

```
print ("Turm Position : ", tx.GetCounter(2))
```

bool **GetInput**(InputNr)
Auslesen des Wertes des angegebene I-Einganges (D5K-Eingang : Taster, Reedkontakt, Phototransistor)
Beispiel

```
if tx.GetInput(1):
  .....
else:
  ....
```

Wenn der I-Eingang 1 true ist (!= 0), wird der
"else"-Zweig durchlaufen

bool **GetTrack**(ctrlD, InputNr)
Auslesen des aktuellen Wertes eines Einganges des Spursensors (0 von Spur, sonst auf Spur (schwarz))

int **GetVoltage**(ctrlD, VoltNr)
Auslesen des Spannungwertes des angegebenen A10V-Einganges (Farbsensor, Spannung allgemein).

OpenInterface(ComNr)
Verbindung zu einem ROBO TX Controller herstellen. OpenController muß als erste Methode aufgerufen werden.
ComNr : Nummer der USB-COM-Verbindung (aktuellen Wert mit RoboTxTest feststellen
Exception FishFaceException bei Openfehler.

- **Pause**(mSek)
  Anhalten des Programmablaufs für mSek MilliSekunden
  Beispiel

```
tx.SetMotor(1, tx.Links)
tx.Pause(1000)
tx.SetMotor(1, tx.Aus)
```

Der Motor am M-Ausgang M1 wird für eine Sekunde (1000 MilliSekunden) eingeschaltet.

- **SetLamp**(ctrlD, LampNr, OnOff)
  Setzen eines O-Ausganges auf OnOff = 0 : Abschalten, 1 - 512 : Einschalten.
  Beispiel:

```
tx.SetLamp(1, 512)
tx.Pause(2000)
tx.SetLamp(1, 0)
tx.SetLamp(2, 444)
```

Die Lampe an O1 vorn und Masse wird für 2 Sekunden eingeschaltet und anschließend die an O2, hier mit geringerer Helligkeit...

- **SetMotor**(ctrlD, MotorNr, Direction, Speed=512)
  setzen eines M-Ausganges, optional mit Geschwindigkeitsangabe (default = 512, full)

```
tx.SetMotor(0, 1, tx.Rechts, 333)
tx.Pause(1000)
tx.SetMotor(1, tx.Links)
tx.Pause(1000)
tx.SetMotor(1, tx.Aus)
```

Der Motor an M1 wird für 1000 MilliSekunden rechtsdrehend, geringe Geschwindigkeit eingeschaltet und anschließend für 1 Sekunde linksdrehend, volle Geschwindigkeit

StartMotor(ctrlD, MotorNr, Direction, Speed, ICount)
Starten eines Motors mit Impulsrad (C-Eingang wie MotorNr) für eine vorgegebene Anzahl Impulse. Asynchron, es wird nicht auf Erreichen der Zielposition gewartet.

```
tx.SetMotor(0, 2, tx.Links, 512, 123)
```

Der Motor an M2 wird für 123 Impulse an C2 mit der Geschwindigkeitsstufe 512 eingeschaltet. Das Abschalten erfolgt selbsttätig. Das Programm läuft währenddessen weiter. Siehe WaitForMotor, WaitForMotors

StartRobMotor(ctrlD, MotorNr, Direction, Speed, ICount)
Starten eines Motors mit Impulsrad (C-Eingang wie MototNr) und zusätzlich Endtaster an I-Eingang (wie MotorNr) zur alternativen Begrenzung des Weges bei Linkslauf.

```
tx.SetMotor(0, 1, tx.Links, 512, 9999)
tx.WaitForMotor(0, 1)
tx.SetMotor(0, 1, tx.Rechts, 333, 6)
tx.WaitForMotor(0, 1)
```

Motor an M1 fährt mit Fullspeed nach links zum Endtaster an I1. Der Wert 9999 dient dabei nur dazu um einen vorzeitigen Stop zu verhindern. Anschließend fährt er in Schleichfahrt 6 Impulse nach rechts.

- **WaitForHigh**(ctrlD, InputNr)
  Warten auf einen false/true Durchgang an InputNr
  Beispiel :

```
tx.SetMotor(0, 1, tx.Ein)
tx.SetMotor(0, 2, tx.Links)
tx.WaitForHigh(1)
tx.SetMotor(2, tx.Aus)
```

Eine Lichtschranke mit Lampe an M1 und Phototransistor an I1 wird eingeschaltet. Ein Förderband mit Motor M2 wird gestartet, es wird gewartet bis ein Teil auf dem Förderband aus der Lichtschranke ausgefahren ist (die Lichtschranke wird geschlossen), dann wird abgeschaltet. Die Lichtschranke muß vorher false sein(unterbrochen)

- **WaitForInput**(ctrlD, InputNr, OnOff=1)
  Warten auf InputNr = OnOff. OnOff ist optional (default=1)
  Beispiel :

```
tx.SetMotor(0, 1, tx.Links)
tx.WaitForInput(0, 1)
tx.SetMotor(0, 1, tx.Aus)
```

Der Motor an M1 wird gestartet, es wird auf I1 = true gewartet, dann wird der Motor wieder abgeschaltet : Anfahren einer Endposition.

- **WaitForLow**(ctrlD, InputNr)
  Warten auf einen true/false Durchgang an InputNr
  Beispiel :

```
tx.SetMotor(0, 1, tx.Ein)
tx.SetMotor(0, 2, tx.Links)
tx.WaitForLow(0, 1)
tx.SetMotor(0, 2, tx.Aus)
```

Eine Lichtschranke mit Lampe an M1 und Phototransistor an I1 wird eingeschaltet. Ein Förderband mit Motor an M2 wird gestartet, es wird gewartet, bis ein Teil auf dem Förderband in die Lichtschranke einfährt (sie unterbricht), dann wird abgeschaltet. Die Lichtschranke muß vorher true sein (nicht unterbrochen).

int **WaitForMotor**(ctrlD, MotorNr)
Warten auf das Ende des mit StartMotor/StartRobMotor gestarteten Motors. Der Rückgabewert enthält eine evtl. Differenz der tatsächlichen gefahrenen Impulse zur Vorgabe.

- **WaitForMotors**(ctrlD, MotorNrs)
  Warten auf ein MotorReady-Ereignis der mit StartMotor/StartRobMotor gestarten

```
tx.SetMotor(0, 4, tx.Links, 512, 50)
tx.SetMotor(0, 3, tx.Rechts, 444, 40)
tx.WaitForMotors(0, [4, 3])
```

Der Motor an M4 wird linksdrehend mit voller Geschindigkeit für 50 Impulse gestartet, der an M3 rechtdrehend mit halber Geschwindigkeit für 40 Impulse. Mit WaitForMotors wird auf das Erreichen der vorgegebenen Position beider Motoren gewartet.

Die Methoden erwarten ein vorhergehendes OpenController. Eine entsprechende Exception wird z.Zt. nicht ausgelöst. Sie enthalten kein **DoEvents** (Abarbeiten der Messagequeue) um das Programm unterbrechbar zu machen. Ggf. sind sie (besonders bei GUI-Progammen) in eigene Thread zu verpacken. Wird im Ablauf ein ControllerProblem festgestellt, wird keine entsprechende **Exception** ausgelöst. Die Wait-Methoden setzen bei Bedarf den zugehörenden **Counter** zurück. Das Auslösen von Exceptions in den Methoden wäre möglich, wurde aber mit Rücksicht auf Stil und Geist von Python unterlassen. Ein Einstieg in die Messagequeue wurde noch nicht gefunden.

Die SetMotor(s)-Methoden sind **asynchron** d.h. der oder die angesprochenen Motoren (Lampen) werden mit der Methode gestartet. Sie laufen dann unabhängig vom Programm weiter. Sie werden durch ein weiteres SetMotors mit Direction = 0 beendet. Ausnahme : SetMotor mit Count-Parameter. Diese Methode beendet sich nach Erreichen der vorgegebenen Position selber.

Die Wait-Methoden koordinieren – meist in Verbindung mit End- bzw. ImpulsTastern den asynchronen Motorlauf mit dem Ablauf des Programms.Sie halten den weiteren Progammablauf an, bis das Waitziel (Ablauf Zeit, erreichte Position, Tasterstellung ...) erreicht ist d.h. sie synchronisieren den Programmablauf wieder.

Tips & Tricks

Programmrahmen

Die im Kapitel Techniken angeführten Programmausschnitte benötigen einen Programmrahmen innerhalb dessen sie ablaufen können. Er wird im Kapitel Techniken dann nicht mehr extra angegeben.

Import

Die Programmausschnitte nutzen den Modul FishFa40.py, der in einem Python Zugriffspfad liegen muß. z.B. C:\Programme\Python\Modules (bei Standardinstallation von Python) :

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

Open/Close

Vor einem Zugriff auf ein Interface ist die Verbindung dazu herzustellen z.B. :

```
tx.OpenController(4)
```

Der Portname muß den eigenen Erfordernissen angepaßt werden. Hier Zugriff auf den ROBO TX Controller an USB, der über COM4 adressiert wird.

Nach der Nutzung ist er Port wieder freizugeben :

```
tx.CloseController
```

Will man mögliche Fehler beim Open abfangen, ist das OpenController in einen try / except Block zustellen. Der Standard Programmrumpf sieht dann so aus :

```
# ---------- Programmname.py : Kurztext ---------------

from FishFaTX import *
import sys
tx = FishFace()

try:
  tx.OpenController(4)
except FishFaceException:
  print ("--- Interface Problem ---")
  sys.exit(0)

# --- Beginn des eigentlichen Programms ---


.....


tx.CloseController()
```

Techniken

Blinker/Schleife

Lampe an M1 blinkt im Sekundentakt :

```
mGelb = 1

while not tx.Finish():
  tx.SetMotor(0, mGelb, tx.Ein)
  tx.Pause(555)
  tx.SetMotor(0, mGelb, tx.Aus)
  tx.Pause(333)
```

Die Parameter für die FishFace Methoden sollten benannt werden. Für einige Standardwerte gibt es bereits Name : tx.Ein, tx.Aus, tx.Links, tx.Rechts und tx.Ende, tx.Time, tx.ESC für die Methode WaitForMotors. Weitere sollte man selber erfinden.

Meistens enthält das Programm ein große Schleife in der alle Befehle wiederholt durchlaufen werden. Hier ist das `while not tx.Finish():` Die Methode Finish prüft, ob ein Abbruchwunsch vorliegt und meldet dann true (1) zurück, deswegen in der while-Schleife die Verneinung durch `not`
Beenden der Schleifte duch ESC-Taste. Es kann auch zusätzlich ein I-Eingang angegeben werden : `tx.Finish(8)`. Beendigung durch ESC-Taste oder I8.

WechselBlinker

Lampen an M1 und M2 blinken im Wechsel

```
while not tx.Finish():
  tx.SetMotor(0, 2, tx.Aus)
  tx.SetMotor(0, 1, tx.Ein)
  tx.Pause(444)
  tx.SetMotor(0, 1, tx.Aus)
  tx.SetMotor(0, 2, tx.Ein)
  tx.Pause(444)
```

Abfrage eines I-Einganges

Wenn I1 geschaltet ist print ("---EIN---") sonst "---AUS---" :

```
if tx.GetInput(0, 1):
  print ("--- EIN ---")
else:
  print ("--- Aus ---")
```

Warten auf einen I-Eingang

Wenn I1 geschlossen ist, wird print ("--- Es geht los ---") ausgegeben :

```
print ("--- Zum Programmstart I1 drücken ---")
tx.WaitForInput(0, 1)
print ("--- Es geht los ---")
```

Anzeige des Status der I-Eingänge

Status von I1 :

```
e1 = tx.GetInput(0, 1)
if e1: print ("I1 : ", e1)
```

Analog-Anzeige

Laufende Anzeige der beiden Analog-Eingänge I1 und I2 :

```
while not tx.Finish():
  print ("Werte I1 : ", tx.GetAnalog(0, 1), " I2 : ", \
        tx.GetAnalog(0, 2))
  tx.Pause(1111)
```

Fahren für eine bestimmte Zeit

Der Motor an M3 soll 3,5 Sekunden nach links laufen :

```
tx.SetMotor(0, 3, tx.Links)
tx.Pause(3500)
tx.SetMotor(0, 3, tx.Aus)
```

Fahren zum Endtaster

Der Motor an M3 soll den Endtaster I5 anfahren und dann abschalten :

```
tx.SetMotor(0, 3, tx.Links)
while not tx.GetInput(0, 5): pass
tx.SetMotor(0, 3, tx.Aus)
```

Das Beispiel ist umständlich und nicht durch ESC-Taste abbrechbar.
besser :

```
tx.SetMotor(0, 3, tx.Links)
tx.WaitForInput(0, 5)
tx.SetMotor(0, 3, tx.Aus)
```

WaitForMotors

Der Motor an M3 fährt für 12 Impulse an I6 mit verminderter Geschwindigkeit nach Links.

```
tx.SetMotor(0, 3, tx.Links, 444, 12)
tx.WaitForMotors(0, [3,])
```

Es wird gewartet, bis das Ziel erreicht wurde. Es geht auch ohne WaitForMotors, wenn das Programm anderweitig beschäftigt ist (Die Motoren schalten bei Erreichen der Zielposition selbsttätig ab). Siehe auch Anmerkungen : Rob-Funktionen.

Lampen

Lampen werden meistens genauso behandelt wie Motoren (mit zwei Polen an einem M-Ausgang, z.B. `tx.SetMotor(0, 1, tx.Ein))`, da sie aber nur ein- oder ausgeschaltet werden können, ist auch die Schaltung an einem Pol eines M-Ausganges und Masse möglich man kann so bis zu acht Lampen an ein Interface anschließen :

```
tx.SetLamp(0, 1, 512)
tx.SetLamp(0, 4, 512)
tx.Pause(1000)
tx.SetLamp(0, 1, 0)
tx.SetLamp(0, 4, 0)
```

Die Lampen an Pin O1 und O4 werden für 1 Sekunde eingeschaltet.

Lichtschranken

Warten auf Lichtschranke

Lampe an M1, Phototransistor an I1. Es wird auf eine Unterbrechung der Lichtschranke gewartet :

```
mLicht, ePhoto, false = 1,1,0

tx.SetMotor(0, mLicht, tx.Ein)
tx.Pause(555)
tx.WaitForInput(0, ePhoto, false)
```

Lampe wird eingeschaltet, danach 0,5 Sekunden Pause um den Phototransistor "anzuwärmen", dann wird auf eine Unterbrechung der Lichtschranke gewartet.

Warten auf Einfahrt in eine Lichtschranke

Lampe an M1, Förderbandmotor an M3, Phototransistor an I1 :

```
mBand, ePhoto = 2,1

tx.SetMotor(0, mBand, tx.Links)
tx.WaitForLow(0, ePhoto)
tx.SetMotor(0, mBand, tx.Aus)
```

Der Motor M1 läuft solange bis ein Teil auf dem Band in die vorher nicht unterbrochene Lichtschranke einfährt. Die Lichtschranke wurde bereits vorher eingeschaltet.

Warten auf Ausfahrt aus einer Lichtschranke

Lampe an M1, Förderbandmotor an M3, Phototransistor an I1 :

```
mBand, ePhoto = 2,1

tx.SetMotor(0, mBand, tx.Links)
tx.WaitForHigh(0, ePhoto)
tx.SetMotor(0, mBand, tx.Aus)
```

Der Motor M1 läuft solange bis ein Teil auf dem Band,das die Lichtschranke unterbricht, aus der Lichtschranke herausgefahren ist.

Anmerkungen zum Verständnis

Zugriff auf den Controller

Der Zugriff auf das Interface erfolgt indirekt über eine Poll-Routine, die in regelmäßigen Abständen (ca. 10 msec) die Werte des Interface ausliest und gleichzeitig den Status der M-Ausgänge setzt (schaltet).

Die ausgelesenen Werte werden in einem internen Kontrollblock abgestellt bzw.die Werte für die M-Ausgänge werden dort entnommen. Der Kontrollblock enthält darüberhinaus alle Werte die für den Betrieb eines Controllers (ggf. mit Extensions) erforderlich sind. Ein Parallel-Betrieb mehrerer Controller die direkt über USB an den Rechner angeschlossen sind, ist nicht möglich.

Die Poll-Routine erledigt über den reinen Verkehr mit dem Interface hinaus noch weitere Aufgaben. Das sind die Zählung der Impulse an den C-Eingängen (Veränderung am true/false-Status eines Einganges) und im RobMode das Abschalten eines M-Ausganges, wenn der zugehörende Impuls-Counter den Wert null erreicht hat.

Die angebotenen Zugriffsfunktionen sind ein Mix aus Notwendigkeit und Komfort. Open/CloseController stellen die Verbindung zum Controller her, setzen default-Parameter und beenden die Verbindung wieder.

RobMotoren und EncoderMotoren sind Motoren mit festvorgegebener Zuordnung von Impulseingang und beim RobMotor auch Endtastereingang. Zugeordnet werden immer "gleiche" Nummern : M1 - Impulseingang C1, Endtastereingang I1 und weiter bis M4 - C4 - I4