ftrobopy - Ansteuerung des fischertechnik TXT Controllers in Python

Version 0.5 (prototype) - 2015 by Torsten Stuehn

class `ftrobopy.`**`ftTXT`**(*host*, *port*)

Basisklasse zum fischertechnik TXT Computer. Implementiert das Protokoll zum Datenaustausch ueber Unix Sockets. Die Methoden dieser Klasse werden typischerweise vom End-User nicht direkt aufgerufen, sondern nur indirekt ueber die Methoden der Klasse ftrobopy.ftrobopy, die eine Erweiterung der Klasse ftrobopy.ftTXT darstellt.

Initialisierung der ftTXT Klasse:

- Alle Ausgaenge werden per default auf 1 (=Motor) gesetzt
- Alle Eingaenge werden per default auf 1, 0 (=Taster, digital) gesetzt
- Alle Zaehler werden auf 0 gesetzt

Parameter: **host** (*string*) – Hostname oder IP-Nummer des TXT Moduls

- '127.0.0.1' im Downloadbetrieb
- '192.168.7.2' im USB Offline-Betrieb
- '192.168.8.2' im WLAN Offline-Betrieb
- '192.168.9.2' im Bluetooth Offline-Betreib

Parameter: **port** (*integer*) – Portnummer (normalerweise 65000)
Rückgabe: Leer

Anwedungsbeispiel:

```
>>> import ftrobopy
>>> txt = ftrobopy.ftTXT('192.168.7.2', 65000)
```

`queryStatus`()

Abfrage des Geraetenamens und der Firmware Versionsnummer des TXT Nach dem Umwandeln der Versionsnummer in einen Hexadezimalwert, kann die Version direkt abgelesen werden.

Rückgabe: Geraetename (string), Versionsnummer (integer)

Anwendungsbeispiel:

```
>>> name, version = txt.queryStatus()
```

`getDevicename`()

Liefert den zuvor mit queryStatus() ausgelesenen Namen des TXT zurueck

Rückgabe: Geraetename (string)

Anwendungsbeispiel:

```
>>> print 'Name des TXT: ', txt.getDevicename()
```

`getVersion`()

Liefert die zuvor mit queryStatus() ausgelesene Versionsnummer zurueck. Um die Firmwareversion direkt ablesen zu koennen, muss diese Nummer noch in einen Hexadezimalwert umgewandelt werden

Rückgabe: Versionsnummer (integer)

Anwendungsbeispiel:

```
>>> print hex(txt.getVersion())
```

`startOnline`()

Startet den Onlinebetrieb des TXT

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> txt.startOnline()
```

`stopOnline`()

Beendet den Onlinebetrieb des TXT

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> txt.stopOnline()
```

`setConfig`(M, I)

Einstellung der Konfiguration der Ein- und Ausgaenge des TXT. Diese Funktion setzt nur die entsprechenden Werte in der ftTXT-Klasse. Zur Uebermittlung der Werte an den TXT wird die updateConfig-Methode verwendet.

Parameter: **M** (*int[4]*) – Konfiguration der 4 Motorausgaenge (0=einfacher Ausgang, 1=Motorausgang)

- Wert=0: Nutzung der beiden Ausgaenge als einfache Outputs
- Wert=1: Nutzung der beiden Ausgaenge als Motorausgang (links-rechts-Lauf)

Parameter: **I** (*int[8][2]*) – Konfiguration der 8 Eingaenge
Rückgabe: Leer

Anwendungsbeispiel:

- Konfiguration der Ausgaenge M1 und M2 als Motorausgaenge
- Konfiguration der Ausgaenge O5/O6 und O7/O8 als einfache Ausgaenge
- Konfiguration der Eingaenge I1, I2, I6, I7, I8 als Taster
- Konfiguration des Eingangs I3 als Ultraschall Entfernungsmesser
- Konfiguration des Eingangs I4 als analoger Spannungsmesser
- Konfiguration des Eingangs I5 als analoger Widerstandsmesser

```
>>> M = [txt.C_MOTOR, txt.C_MOTOR, txt.C_OUTPUT, txt.C_OUTPUT]
>>> I = [(txt.C_SWITCH,     txt.C_DIGITAL),
         (txt.C_SWITCH,     txt.C_DIGITAL),
         (txt.C_ULTRASONIC, txt.C_ANALOG),
         (txt.C_VOLTAGE,    txt.C_ANALOG),
         (txt.C_RESISTOR,   txt.C_ANALOG),
```

```
          (txt.C_SWITCH,     txt.C_DIGITAL),
          (txt.C_SWITCH,     txt.C_DIGITAL),
          (txt.C_SWITCH,     txt.C_DIGITAL)]
>>> txt.setConfig(M, I)
>>> txt.updateConfig()
```

getConfig()

Abfrage der aktuellen Konfiguration des TXT

Rückgabe: M[4], I[8][2]
Rückgabetyp: M:int[4], I:int[8][2]

Anwendungsbeispiel: Aenderung des Eingangs I2 auf analoge Ultraschall Distanzmessung

- Hinweis: Feldelemente werden in Python typischerweise durch die Indizes 0 bis N-1 angesprochen
- Der Eingang I2 des TXT wird in diesem Beispiel ueber das Feldelement I[1] angesprochen

```
>>> M, I = txt.getConfig()
>>> I[1] = (txt.C_ULTRASONIC, txt.C_ANALOG)
>>> txt.setConfig(M, I)
>>> txt.updateConfig()
```

updateConfig()

Uebertragung der Konfigurationsdaten fuer die Ein- und Ausgaenge zum TXT

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> txt.setConfig(M, I)
>>> txt.updateConfig()
```

exchangeData()

Low-Level Uebermittlung von Motor-, bzw. Output-Daten (z.B. Motorgeschwindigkeit, Motordistanz) an den TXT und Abholen der aktuellen Eingangswerte vom TXT.

Rückgabe: Leer

Hinweis

- Diese Methode wird typischerweise nicht direkt aufgerufen, sondern nur indirekt ueber die update()-Methode (siehe unten)

startCameraOnline()

Die Camerafunktionen werden noch nicht unterstuetzt.

stopCameraOnline()

Die Camerafunktionen werden noch nicht unterstuetzt.

cameraOnlineFrame()

Die Camerafunktionen werden noch nicht unterstuetzt.

cameraFramesReady()

Die Camerafunktionen werden noch nicht untertuetzt.

update(*wait=False*)

Senden von Motor- und/oder Output Parametern an den TXT und unmittelbares Abholen der Werte der TXT Eingaenge.

- der TXT kommuniziert nur alle 20ms (oder 10ms ?) ueber den Port 65000 mit der Anwendung
- die update()-Methode stellt automatisch sicher, dass dieses 20ms Kommunikationsintervall eingehalten wird und die exchangeData()-Methode nicht haeufiger aufgerufen wird. Bei direktem (haeufigen) Aufruf der exchangeData()-Methode bricht sonst evtl. die Verbindung zum TXT ab.
- Diese Funktionsweise wurde “experimentell” ermittelt. Die Dokumentation von fischertechnik ist an dieser Stelle nicht ganz vollstaendig.

Parameter: **wait** (*boolean True oder False*) – Warten auf naechstes 20ms Intervall

- Wird dieser Parameter auf True gesetzt, wartet die update()-Methode bis zum naechsten 20ms-Zyklus und fuehrt dann den Datenaustausch durch.
- Steht dieser Parameter auf False (default) wird der Datenaustausch nur durchgefuehrt, falls seit dem letzten Datenabgleich mindestens 20ms vergangen sind, falls nicht, wird kein Datenaustausch vorgenommen.

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> txt.setPwm(2, 512)
>>> txt.setPwm(3, 0)
>>> txt.update(wait=True)
>>> Taster_I6_gedrueckt = txt.getCurrentInput(6)
```

sleep(*seconds*)

Wartet die angegebene Anzahl von Sekunden ab, bevor die Programmausfuehrung fortgesetzt wird. Waehrend der Wartezeit wird automatisch kontinuierlich die update()-Methode aufgerufen um die Verbindung zum TXT aufrecht zu erhalten.

Parameter: **seconds** (*float*) – Wartezeit in Sekunden

Anwendungsbeispiel:

Der Programmablauf wird fuer eine halbe Sekunde pausiert (z.B. um einem Pneumatik-Zylinder, der von einem Magnetventil angesteuert wird, Zeit zu geben, in seine Endstellung zu kommen)

```
>>> txt.sleep(0.5)
```

incrMotorCmdId(*idx*)

Erhoehung der sog. Motor Command ID um 1.

Diese Methode muss immer dann aufgerufen werden, wenn die Distanzeinstellung eines Motors (gemessen ueber die 4 schnellen Counter-Eingaenge) geaendert wurde oder wenn ein Motor mit einem anderen Motor synchronisiert werden soll. Falls nur die Motorgeschwindigkeit veraendert wurde, ist der Aufruf der incrMotorCmdId()-Methode nicht notwendig.

Parameter: **idx** (*integer*) – Nummer des Motorausgangs

Achtung:

- Die Zaehlung erfolgt hier von 0 bis 3, idx=0 entspricht dem Motorausgang M1 und idx=3 entspricht dem Motorausgang M4

Anwendungsbeispiel:

Der Motor, der am TXT-Anschluss M2 angeschlossen ist, soll eine Distanz von 200 (Counterzaehlungen) zuruecklegen.

```
>>> txt.setMotorDistance(1, 200)
>>> txt.incrMotorCmdId(1)
>>> txt.update(wait=True)
```

getMotorCmdId(*idx=None*)

Liefert die letzte Motor Command ID eines Motorausgangs (oder aller Motorausgaenge als array) zurueck.

Parameter:	**idx** (*integer*) – Nummer des Motorausgangs. Falls dieser Parameter nicht angeben wird, wird die Motor Command ID aller Motorausgaenge als Array[4] zurueckgeliefert.
Rückgabe:	Die Motor Command ID eines oder aller Motorausgaenge
Rückgabetyp:	integer oder integer[4] array

Anwendungsbeispiel:

```
>>> letzte_cmd_id = txt.getMotorCmdId(4)
```

cameraOnline()

Die Camerafunktionen werden noch nicht unterstuetzt

getSoundCmdId()

Liefert die letzte Sound Command ID zurueck.

Rückgabe:	Letzte Sound Command ID
Rückgabetyp:	integer

Anwendungsbeispiel:

```
>>> last_sound_cmd_id = txt.getSoundCmdId()
```

incrCounterCmdId(*idx*)

Erhoehung der Counter Command ID um eins. Falls die Counter Command ID eines Counters um eins erhoeht wird, wird der entsprechende Counter zurueck auf 0 gesetzt.

Parameter:	**idx** (*integer*) – Nummer des schnellen Countereingangs, dessen Command ID erhoeht werden soll. (Hinweis: die Zaehlung erfolgt hier von 0 bis 3 fuer die Counter C1 bis C4)
Rückgabe:	Leer

Anwendungsbeispiel:

Erhoehung der Counter Command ID des Counters C4 um eins.

```
>>> txt.incrCounterCmdId(3)
```

incrSoundCmdId()

Erhoehung der Sound Command ID um eins. Die Sound Command ID muss immer dann um eins erhoeht werden, falls ein neuer Sound gespielt werden soll oder wenn die Wiederholungsanzahl eines Sounds veraendert wurde. Falls kein neuer Sound index gewaehlt wurde und auch die Wiederholungsrate nicht veraendert wurde, wird der aktuelle Sound erneut abgespielt.

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> txt.incrSoundCmdId()
```

setSoundIndex(*idx*)

Einstellen eines neuen Sounds.

Parameter: **idx** (*integer*) – Nummer des neuen Sounds (0=Kein Sound, 1-29 Sounds des TXT)

Rückgabe: Leer

Anwendungsbeispiel:

Sound “Augenzwinkern” einstellen und 2 mal abspielen.

```
>>> txt.setSoundIndex(26)
>>> txt.setSoundRepeat(2)
>>> txt.incrSoundCmdId()
```

getSoundIndex()

Liefert die Nummer des aktuell eingestellten Sounds zurueck.

Rückgabe: Nummer des aktuell eingestellten Sounds
Rückgabetyp: integer

Anwendungsbeispiel:

```
>>> aktueller_sound = txt.getSoundIndex()
```

setSoundRepeat(*rep*)

Einstellen der Anzahl der Wiederholungen eines Sounds.

Parameter: **rep** (*integer*) – Anzahl der Wiederholungen (0=unendlich oft wiederholen)

Anwendungsbeispiel:

“Motor-Sound” unendlich oft (d.h. bis zum Ende des Programmes oder bis zur naechsten Aenderung der Anzahl der Wiederholungen) abspielen.

```
>>> txt.setSound(19) # 19=Motor-Sound
>>> txt.setSoundRepeat(0)
```

getSoundRepeat()

Liefert die aktuell eingestellte Wiederholungs-Anzahl des Sounds zurueck.

Rückgabe: Aktuell eingestellte Wiederholungs-Anzahl des Sounds.
Rückgabetyp: integer

Anwendungsbeispiel:

```
>>> repeat_rate = txt.getSoundRepeat()
```

getCounterCmdId(*idx=None*)

Liefert die letzte Counter Command ID eines (schnellen) Counters zurueck

Parameter: **idx** – Nummer des Counters. (Hinweis: die Zaehlung erfolgt hier von 0 bis 3 fuer die Counter C1 bis C4)

Anwendungsbeispiel:

Counter Command ID des schnellen Counters C3 in Variable num einlesen.

```
>>> num = txt.getCounterCmdId(2)
```

setPwm(*idx*, *value*)

Einstellen des Ausgangswertes fuer einen Motor- oder Output-Ausgang.

Parameter:
- **idx** (*integer (0-7)*) – Nummer des Ausgangs. (Hinweis: die Zaehlung erfolgt hier von 0 bis 7 fuer die Ausgaenge O1-O8)
- **value** (*integer (0-512)*) – Wert, auf den der Ausgang gesetzt werden soll (0:Ausgang ausgeschaltet, 512: Ausgang auf maximum)

Rückgabe: Leer

Anwendungsbeispiel:

- Motor am Anschluss M1 soll mit voller Geschwindigkeit Rueckwaerts laufen.
- Lampe am Anschluss O3 soll mit halber Leuchtkraft leuchten.

```
>>> txt.setPwm(0,0)
>>> txt.setPwm(1,512)
>>> txt.setPwm(2,256)
>>> txt.update(wait=True) # uebermitteln der Daten an den TXT
```

getPwm(*idx=None*)

Liefert die zuletzt eingestellten Werte der Ausgaenge O1-O8 (als array[8]) oder den Wert eines Ausgangs.

Parameter: **idx** (*integer oder None, bzw. leer*) –

- Wenn kein idx-Parameter angegeben wurde, werden alle Pwm-Einstellungen als array[8] zurueckgeliefert.
- Ansonsten wird nur der Pwm-Wert des mit idx spezifizierten Ausgangs zurueckgeliefert.

Hinweis: der idx-Parameter wird angeben von 0 bis 7 fuer die Ausgaenge O1-O8

Rückgabe: der durch (idx+1) spezifizierte Ausgang O1 bis O8 oder das gesamte Pwm-Array
Rückgabetyp: integer oder integer array[8]

Anwendungsbeispiel:

Liefert die

```
>>> M1_a = txt.getPwm(0)
>>> M1_b = txt.getPwm(1)
>>> if M1_a > 0 and M1_b == 0:
      print "Der an M1 angeschlossene Motor laeuft mit der Geschwindigkeit ", M1
    else:
      if M1_a == and M1_b > 0:
        print "Der an M1 angeschlossene Motor laeuft mit der Geschwindigkeit ",
```

setMotorSyncMaster(*idx*, *value*)

Hiermit koennen zwei Motoren miteinander synchronisiert werden, z.B. fuer perfekten Geradeauslauf.

Parameter:
- **idx** (*integer*) – Der Motorausgang, der synchronisiert werden soll
- **value** (*integer*) – Die Numer des Motorausgangs, mit dem synchronisiert werden soll.

Rückgabe: Leer

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Motor-Ausgaenge M1 bis M4.
- der value-PArameter wird angeben von 1 bis 4 fuer die Motor-Ausgaenge M1 bis M4.

Anwendungsbeispiel:

Die Motorausgaenge M1 und M2 werden synchronisiert. Um die Synchronisations-Befehle abzuschliessen, muss ausserdem die MotorCmdId der Motoren erhoeht werden.

```
>>> txt.setMotorSyncMaster(0, 2)
>>> txt.setMotorSyncMaster(1, 1)
>>> txt.incrMotorCmdId(0)
>>> txt.incrMotorCmdId(1)
>>> txt.update(Wait=True)
```

getMotorSyncMaster(*idx=None*)

Liefert die zuletzt eingestellte Konfiguration der Motorsynchronisation fuer einen oder alle Motoren zurueck.

Parameter: **idx** (*integer*) – Die Nummer des Motors, dessen Synchronisation geliefert werden soll oder None oder <leer> fuer alle Ausgaenge.

Rückgabe: Leer

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Motor-Ausgaenge M1 bis M4.
- oder None oder <leer> fuer alle Motor-Ausgaenge.

Anwendungsbeispiel:

```
>>> print "Aktuelle Konfiguration aller Motorsynchronisationen: ", txt.getMotorS
```

setMotorDistance(*idx*, *value*)

Hiermit kann die Distanz (als Anzahl von schnellen Counter-Zaehlungen) fuer einen Motor eingestellt werden.

Parameter: **idx** (*integer*) – Nummer des Motorausgangs

Rückgabe: Leer

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Motor-Ausgaenge M1 bis M4.

Anwendungsbeispiel:

Der Motor an Ausgang M3 soll 100 Counter-Zaehlungen lang drehen. Um den Distanz-Befehl abzuschliessen, muss ausserdem die MotorCmdId des Motors erhoeht werden.

```
>>> txt.setMotorDistance(2, 100)
>>> txt.incrMotorCmdId(2)
>>> txt.update(wait=True)
```

getMotorDistance(*idx=None*)

Liefert die zuletzt eingestellte Motordistanz fuer einen oder alle Motorausgaenge zurueck.

Parameter:	**idx** (*integer*) – Nummer des Motorausgangs
Rückgabe:	Letzte eingestellte Distanz eines Motors (idx=0-3) oder alle zuletzt eingestellten Distanzen (idx=None oder kein idx-Parameter angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Motor-Ausgaenge M1 bis M4.

Anwendungsbeispiel:

```
>>> print "Die zuletzt mit setMotorDistance() eingestellte Distanz fuer den Motc
```

getCurrentInput(*idx=None*)

Liefert den aktuellen vom TXT zurueckgelieferten Wert eines Eingangs oder aller Eingaenge als Array

Parameter:	**idx** (*integer*) – Nummer des Eingangs
Rückgabe:	Aktueller Wert eines Eingangs (idx=0-7) oder alle aktuellen Eingangswerte des TXT-Controllers als Array[8] (idx=None oder kein idx angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 7 fuer die Eingaenge I1 bis I8.

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> print "Der aktuelle Wert des Eingangs I4 ist: ", txt.getCurrentInput(3)
```

getCurrentCounterInput(*idx=None*)

Zeigt an, ob ein Counter oder alle Counter (als Array[4]) sich seit der letzten Abfrage veraendert haben.

Parameter:	**idx** (*integer*) – Nummer des Counters
Rückgabe:	Aktueller Status-Wert eines Counters (idx=0-3) oder aller schnellen Counter des TXT-Controllers als Array[4] (idx=None oder kein idx angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Counter C1 bis C4.

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> c = txt.getCurrentCounterInput(0)
>>> if c==0:
>>>   print "Der Counter von Motor M1 hat sich seit der letzten Abfrage nicht ve
>>> else:
>>>   print "Der Counter von Motor M1 hat sich seit der letzten Abfrage veraende
```

getCurrentCounterValue(*idx=None*)

Liefert den aktuellen Wert eines oder aller schnellen Counter Eingaenge zurueck. Damit kann z.B. nachgeschaut werden, wie weit ein Motor schon gefahren ist.

Parameter:	**idx** (*integer*) – Nummer des Counters
Rückgabe:	Aktueller Wert eines Counters (idx=0-3) oder aller schnellen Counter des TXT-Controllers als Array[4] (idx=None oder kein idx angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Counter C1 bis C4.

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> print "Der aktuelle Wert des Counters von Motor M1 ist: ", txt.getCurrentCou
```

getCurrentCounterCmdId(*idx=None*)

Liefert die aktuelle Counter Command ID eines oder aller Counter zurueck.

Parameter:	**idx** (*integer*) – Nummer des Counters
Rückgabe:	Aktuelle Commmand ID eines Counters (idx=0-3) oder aller Counter des TXT-Controllers als Array[4] (idx=None oder kein idx angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Counter C1 bis C4.

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> print "Die aktuelle Counter Command ID von Counter C4 lautet: ", txt.getCurr
```

getCurrentMotorCmdId(*idx=None*)

Liefert die aktuelle Motor Command ID eines oder aller Motoren zurueck.

Parameter:	**idx** (*integer*) – Nummer des Motors
Rückgabe:	Aktuelle Commmand ID eines Motors (idx=0-3) oder aller Motoren des TXT-Controllers als Array[4] (idx=None oder kein idx angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 3 fuer die Motoren M1 bis M4.

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> print "Die aktuelle Motor Command ID von Motor M4 lautet: ", txt.getCurrentM
```

`getCurrentSoundCmdId`()

Liefert die aktuelle Sound Command ID zurueck.

Rückgabe:	Die aktuelle Sound Command ID
Rückgabetyp:	integer

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> print "Die aktuelle Sound Command ID ist: ", txt.getCurrentSoundCmdId()
```

`getCurrentIr`()

Liefert die aktuellen Werte der Infrarot Fernsteuerung zurueck (als Array oder als einzelnen Wert)

Die Werte werden fuer jede einzelne Einstellung der DIP-Switche auf der IR-Fernsteuerung zurueckgeliefert, so dass insgesamt 4 Fernsteuerungen abgefragt werden koennen (als Python-Liste von Python-Listen). Die 5. Liste innerhalb der Python-Liste gibt den Zustand einer Fernsteuerung mit beliebiger DIP-Switch-Einstellung zurueck. Die Einstellung der Switche selbst wird auch zurueckgelifert, aber nicht, wenn wenn die Switche sich aendern, sondern nur, wenn einer der anderen Werte sich aendert.

Parameter:	**idx** (*integer*) – Nummer des IR Eingangs
Rückgabe:	Aktueller Wert eines IR Eingangs (idx=0-5) oder aller IR Eingaenge des TXT-Controllers als Array[5] (idx=None oder kein idx angegeben)

Hinweis:

- der idx-Parameter wird angeben von 0 bis 5 fuer die einzelnen Listen innerhalb der Liste.

Anwendungsbeispiel:

Vor der Abfrage werden die aktuellen Werte beim TXT “abgeholt”.

```
>>> txt.update(wait=True)
>>> print "Die aktuellen Werte der IR Fernsteuerung lauten: ", txt.getCurrentIr(
```

class `ftrobopy.`**`ftrobopy`**(*host*, *port*)

Basisklassen: **`ftrobopy.ftTXT`**

Erweiterung der Klasse ftrobopy.ftTXT. In dieser Klasse werden verschiedene fischertechnik Elemente auf einer hoeheren Abstraktionsstufe (aehnlich den Programmelementen aus der ROBOPro Software) fuer den End-User zur Verfuegung gestellt. Derzeit sind die folgenden Programmelemente implementiert:

- **motor**, zur Ansteuerung der Motorausgaenge M1-M4
- **output**, zur Ansteuerung der universellen Ausgaenge O1-O8
- **input**, zum Einlesen von Werten der Eingaenge I1-I8
- **ultrasonic**, zur Bestimmung von Distanzen mit Hilfe des Ultraschall Moduls

Ausserdem werden die folgenden Sound-Routinen zur Verfuegung gestellt:

- **play_sound**
- **stop_sound**
- **sound_finished**

Initialisierung der ftrobopy Klasse:

- Herstellen der Socket-Verbindung zum TXT Controller und Abfrage des Geraetenamens und der Firmwareversionsnummer
- Starten des Online-Modus des TXT
- Initialisierung aller Datenfelder der ftTXT Klasse mit Defaultwerten

Parameter: **host** (*string*) – Hostname oder IP-Nummer des TXT Moduls

- '127.0.0.1' im Downloadbetrieb
- '192.168.7.2' im USB Offline-Betrieb
- '192.168.8.2' im WLAN Offline-Betrieb
- '192.168.9.2' im Bluetooth Offline-Betreib

Parameter: **port** (*integer*) – Portnummer (normalerweise 65000)
Rückgabe: Leer

Anwedungsbeispiel:

```
>>> import ftrobopy
>>> ftrob = ftrobopy.ftrobopy('192.168.7.2', 65000)
```

motor(*output*)

Diese Funktion erzeugt ein Motor-Objekt, das zur Ansteuerung eines Motors verwendet werden kann, der an einem der Motorausgaenge M1-M4 des TXT angeschlossen ist.

Anwendungsbeispiel:

```
>>> Motor1 = ftrob.motor(1)
```

Das so erzeugte Motor-Objekt hat folgende Methoden:

setSpeed (speed)

Einstellung der Motorgeschwindigkeit

Parameter: **speed** (*integer*) –
Rückgabe: Leer

Gibt an, mit welcher Geschwindigkeit der Motor laufen soll:

- der Wertebereich der Geschwindigkeit liegt zwischen 0 (Motor anhalten) und 512 (maximale Geschwindigkeit)
- Falls die Geschwindigkeit negativ ist, laeuft der Motor Rueckwaerts

Hinweis: Der eingegebene Wert fuer die Geschwindigkeit haengt nicht linear mit der tatsaechlichen

Drehgeschwindig des Motors zusammen, d.h. die Geschwindigkeit bei 400 ist nicht doppelt so gross, wie die Geschwindigkeit 200. Bei Hoeheren Werten von speed kann dadurch die Geschwindigkeit in feineren Stufen reguliert werden.

Anwendungsbeispiel:

Dieser Befehl wird erst dann ausgefuehrt, wenn das “update(wait=True)” Kommando ausgefuehrt wurde. Dadurch ist es moeglich die Geschwindigkeit von mehreren Motoren anzugeben und erst dann diese Werte gemeinsam an den TXT zu uebermitteln.

```
>>> Motor1.setSpeed(512)
>>> ftrob.update(wait=True)
```

setDistance (distance, syncto=None)

Einstellung der Mototdistanz, die ueber die schnellen Counter gemessen wird.

Parameter:
- **distance** (*integer*) – Gibt an, um wieviele Counter-Zaehlungen sich der Motor drehen soll.
- **syncto** (*ftrobopy.motor Objekt*) – Hiermit koennen zwei Motoren synchronisiert werden um z.B. perfekten Geradeauslauf zu ermoeglichen. Als Parameter wird hier ein anderes Motorobjekt uebergeben.

Rückgabe: Leer

Anwendungsbeispiel:

Der Motor wir mit dem Motor synchronisiert, der am Motorausgang M2 angeschlossen ist. Grundsaetzlich wird dieser Befehl erst dann ausgefuehrt, wenn das “update(wait=True)” Kommando ausgefuehrt wurde. Nur so ist moeglich, zwei Motoren exakt zu synchronisieren. (das incrMotorCmdId()-Kommando wird innerhalb der setDistance-Routine automatisch ausgefuehrt).

```
>>> Motor_links=ftrob.motor(1)
>>> Motor_rechts=ftrob.motor(2)
>>> Motor_links.setDistance(100, syncto=Motor_rechts)
>>> Motor_rechts.setDistance(100, syncto=Motor_rechts)
>>> ftrob.update(wait=True)
```

finished ()

Abfrage, ob die eingestellte Distanz bereits erreicht wurde.

Rückgabe: False: Motor laeuft noch, True: Distanz erreicht
Rückgabetyp: boolean

Anwendungsbeispiel:

Hinweis: Der finished() Befehl holt sich autmatisch zuerst die aktuellen Werte vom TXT per ftrob.update(wait=True)

```
>>> while not Motor1.finished():
      print "Motor laeuft noch"
```

stop ()

Anhalten des Motors durch setzen der Geschwindigkeit auf 0.

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> Motor1.stop()
>>> ftrob.update(wait=True)
```

output(*num*, *level=0*)

Diese Funktion erzeugt ein allgemeines Output-Objekt, das zur Ansteuerung von Elementen verwendet wird, die an den Ausgaengen O1-O8 angeschlossen sind.

Anwendungsbeispiel:

Am Ausgang O7 ist eine Lampe oder eine LED angeschlossen:

```
>>> Lampe = ftrob.output(7)
```

Das so erzeugte allg. Output-Objekt hat folgende Methoden:

setLevel (level)

Parameter: **level** (*integer, 1 - 512*) – Ausgangsleistung, die am Output anliegen soll (genauer fuer die Experten: die Gesamtlaenge des Arbeitsintervalls eines PWM-Taktes in Einheiten von 1/512, d.h. mit level=512 liegt das PWM-Signal waehrend des gesamten Taktes auf high).

Mit dieser Methode kann die Ausgangsleistung eingestellt werden, um z.B. die Helligkeit einer Lampe zu regeln.

Anwendungsbeispiel:

Die tatsaechlichen Werte werden erst nach einem update(wait=True) an den TXT uebertragen.

```
>>> Lampe.setLevel(512)
>>> ftrob.update(wait=TRUE)
```

input(*num*)

Diese Funktion erzeugt ein allgemeines Input-Objekt fuer Sensoren oder Taster, die an einem der Eingaenge I1-I8 des TXT angeschlossen sind.

Anwendungsbeispiel:

```
>>> Taster = ftrob.input(5)
```

Das so erzeugte allg. Input-Objekt hat folgende Methoden:

state ()

Mit dieser Methode wird der Status des Eingangs abgefragt.

Rückgabe: Zustand des Schalters (0: Kontakt geschlossen, 1: Kontakt geoeffnet)
Rückgabetyp: integer

Anwendungsbeispiel:

Hinweis: Die update(wait=True) Methode zum Holen der aktuellen Werte vom TXT wird nicht automatisch

innerhalb der state()-Routine ausgefuehrt. Sie muss bei Bedarf vorher von Hand ausgefuehrt werden.

```
>>> ftrob.update(wait=True)
>>> if Taster.state() == 1:
      print "Der Taster an Eingang I5 wurde gedrueckt."
```

ultrasonic(*num*)

Diese Funktion erzeugt ein Objekt zur Abfrage eines an einem der Eingaenge I1-I8 angeschlossenen TX-Ultraschall-Distanzmessers.

Anwendungsbeispiel:

```
>>> ultraschall = ftrob.ultrasonic(6)
```

Das so erzeugte Ultraschall-Objekt hat folgende Methoden:

distance ()

Mit dieser Methode wird der aktuelle Distanz-Wert abgefragt

Rückgabe:	Die aktuelle Distanz zwischen Ultraschallsensor und vorgelagertem Objekt in cm.
Rückgabetyp:	integer

Hinweis: Die update(wait=True) Methode zum Holen der aktuellen Werte vom TXT wird nicht automatisch innerhalb der state()-Routine ausgefuehrt. Sie muss bei Bedarf vorher von Hand ausgefuehrt werden.

Anwendungsbeispiel:

```
>>> ftrob.update(wait=True)
>>> print "Der Abstand zur Wand betraegt ", ultraschall.distance(), " cm."
```

sound_finished()

Ueberpruefen, ob die Zeit des zuletzt gespielten Sounds bereits abgelaufen ist

Rückgabe:	True (Zeit ist abgelaufen) oder False (Sound wird noch abgespielt)
Rückgabetyp:	boolean

Anwendungsbeispiel:

```
>>> while not ftrob.sound_finished():
      pass
```

play_sound(*idx*, *seconds=1*)

Einen Sound eine bestimmte zeitlang abspielen

Anwendungsbeispiel:

```
>>> ftrob.play_sound(27, 5) # 5 Sekunden lang Fahrgeraeusche abspielen
```

stop_sound()

Die Aktuelle Soundausgabe stoppen. Dabei wird der abzuspielende Sound-Index auf 0 (=Kein Sound) und der Wert fuer die Anzahl der Wiederholungen auf 1 gesetzt.

Hinweis: die Befehle incrSoundCmdId() und update(wait=True) werden innerhalb dieser Routine automatisch

aufgerufen.

Rückgabe: Leer

Anwendungsbeispiel:

```
>>> ftrob.stop_sound()
```