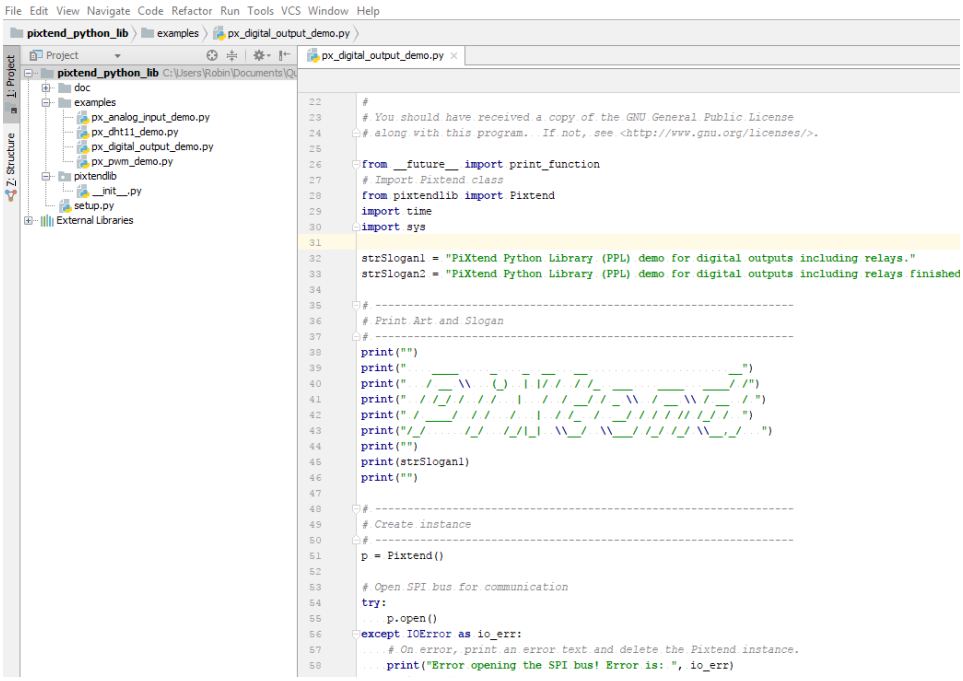
Application-Note
PiXtend Python Library

Installation, Einrichtung, Programmierung



APP-PX-401

Stand 05.03.2018, V1.02

Qube Solutions UG (haftungsbeschränkt)
Arbachtalstr. 6, 72800 Eningen, Germany
http://www.qube-solutions.de/
http://www.pixtend.de

Versionshistorie

Version	Beschreibung	Bearbeiter
1.00	Dokument erstellt	RT
1.01	Korrektur Python Installationspakete und Beispielprogramm	RT
1.02	Link Anpassung zur Verwendung der PPL Version 0.1.1	RT

Inhaltsverzeichnis

1. Einleitung

Wir freuen uns, Ihnen eine weitere Möglichkeit präsentieren zu können, mit der sich PiXtend programmieren lässt und die gleichzeitig das Tor zur *Python* Welt öffnet.

(Bildquelle: https://www.python.org/, The Python Brochure)

Die Programmiersprache *Python* wird als eine universelle und interpretierbare Sprache beschrieben. Sie soll eine knappe und lesbare Form haben und auf diesem Weg das Programmieren erleichtern. Ein Hauptmerkmal von *Python* ist beispielsweise, dass die Strukturierung des Programmcodes durch Einrückung des Codes erfolgt[1].

Die Begeisterung und die Verbreitung der Sprache *Python* gehen nicht zuletzt auf eine sehr große Bibliotheksbasis zurück, nahezu für jeden Anwendungsfall gibt es etwas. Auch die sehr große Gemeinschaft, die um *Python* herum entstanden ist, macht diese Sprache sehr beliebt.

Ferner steht *Python* jedem frei zur Verfügung und kann somit bei der Berufsausbildung, im Studium, zu Hause, aber auch kommerziell eingesetzt werden. Die Programmiersprache kann von Anfängern wie auch von fortgeschrittenen Programmierern verwendet werden, um Computerprogramme für verschiedene Systeme zu erstellen.

Daher ist es nicht verwunderlich, dass sich die *Raspberry Pi Foundation* entschieden hat, *Python* gleich mit in ihr Raspbian Image zu integrieren[2].

Gleich nach dem ersten Start kann sofort mit der *Python* Programmierung auf dem Raspberry Pi begonnen werden, alles ist bereits vorinstalliert. Der Zugriff beispielsweise auf die on-board GPIOs oder den SPI-Bus funktionieren „O*ut of the Box*“. Der Raspberry Pi, PiXtend und *Python* passen perfekt zusammen.

1 Quelle Wikipedia April 2017: https://de.wikipedia.org/wiki/Python_(Programmiersprache)
2 https://www.raspberrypi.org/documentation/usage/python/ und https://www.raspberrypi.org/downloads/raspbian/

Deshalb haben wir uns entschieden, PiXtend in Verbindung mit dem Raspberry Pi der *Python Community* zugänglich zu machen und haben ein eigenes *Python* Modul, die *PiXtend Python Library (PPL)*, ins Leben gerufen.

Die *PPL* ist Open Source und kann somit von jedem verwendet, geändert und nach Belieben erweitert werden. Wir würden uns freuen, wenn Sie Ihre *PPL*-Version im Forum posten.

In dieser App-Note möchten wir Ihnen Schritt für Schritt aufzeigen wie schnell und einfach Sie die *PiXtend Python Library (PPL)* auf Ihrem PiXtend-System einrichten und ein erstes *Python* Programm erstellen können.

Wir wünschen viel Spaß beim Testen, Programmieren und Experimentieren!

Viele weitere Informationen, Tipps und Tricks finden Sie auch in unserem Support-Forum unter: http://www.pixtend.de/forum/

Die jeweils neusten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: http://www.pixtend.de/downloads/

1.1 Voraussetzungen

Die Treiber-Unterstützung von PiXtend für Python 2.7.9 und später (nicht Python 3) ist gleichermaßen für **PiXtend V1.2 und V1.3** verwendbar. Sie können jedes PiXtend-System verwenden.

Auch gibt es keine spezielle Festlegung auf ein Raspberry Pi Modell.
Wir empfehlen jedoch eines der folgenden Modelle: **B, B+, 2 B, 3 B**.

Laden Sie das ***PiXtend Image - PiXtend Python Library*** SD-Karten Image aus unserem Download-Bereich herunter und verwenden Sie dieses als Ausgangspunkt für Ihre Projekte. Alternativ können Sie auch ein original Raspbian Jessie Image verwenden, entsprechende Installationsschritte finden Sie im Kapitel 3.

Auf den Raspberry Pi können Sie entweder mit direkt angeschlossener Tastatur und Monitor oder per SSH (TeraTerm / Putty) von einem PC zugreifen.
Verwenden Sie direkt ein Original Raspbian Image, dann benötigt der Raspberry Pi eine aktive Internetverbindung, damit Sie die Schritte in diesem Dokument durchführen können.

Wir empfehlen ferner, die **App-Notes** *Control- und Status-Bytes* sowie *Prozessdaten von PiXtend* bereitzuhalten. Diese Dokumente enthalten Informationen zur Konfiguration der PWM-Ausgänge und der analogen Eingänge, sowie Wissenswertes zu den GPIOs, den digitalen Ein- und Ausgängen und den Relais auf PiXtend.

1.2 Haftungsausschluss

Qube Solutions UG kann nicht für etwaige Schäden verantwortlich gemacht werden die unter Umständen durch die Verwendung der zur Verfügung gestellten Software, Hardware, Treiber oder der hier beschriebenen Schritte oder Software von Drittherstellern entstehen können.

1.3 Sicherheitshinweise

PiXtend darf nicht in sicherheitskritischen Systemen eingesetzt werden.

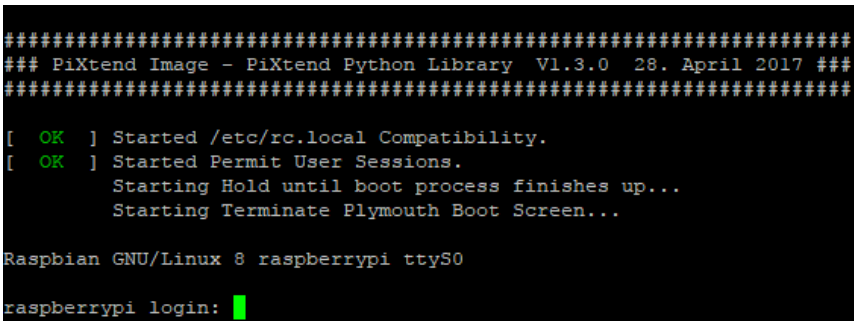
Prüfen Sie vor der Verwendung die Eignung von Raspberry Pi und PiXtend für Ihre Anwendung.

2. Installation mit PiXtend Image

Im *PiXtend Image - PiXtend Python Library* SD-Karten Image ist die *PiXtend Python Library* bereits vorinstalliert und kann sofort verwendet werden.

Auf dem Raspberry Pi finden Sie alle Dateien im Ordner */home/pi/ppl/*.

Sie können daher den Anfang von Kapitel 3 überspringen und direkt mit dem *Kapitel 3.3 Programmiertool* fortfahren. Haben Sie bereits einen SSH-Client installiert und ein Programm zur Übertragung von Dateien zum Raspberry Pi installiert bzw. arbeiten Sie direkt am Raspberry Pi mit Bildschirm und Tastatur, können Sie gleich mit dem *Kapitel 4. Programmierung* weitermachen.



3. Installation auf original Raspbian

Die Installation der benötigten Software wird nachfolgend erläutert:

Wir starten mit einem originalen **Raspbian Jessie Image** (Release: 10.04.17). Bei diesem Image startet nach dem ersten Booten automatisch die neue PIXEL-Oberfläche. Da wir diese hier nicht benötigen, deaktivieren wir sie. Für die nachfolgenden Schritte werden ***Bildschirm, Tastatur und Maus benötigt***. Ein Anmelden über Ethernet per SSH funktioniert nicht, da standardmäßig der SSH Server deaktiviert ist.

In der Menüleiste der PIXEL-Oberfläche auf das Terminal-Symbol klicken und ein Konsolenfenster öffnen. Das Fenster am besten maximieren.

Jetzt können wir auf dem Raspberry Pi die Konfiguration öffnen:

sudo raspi-config

Das Konfigurationsprogramm für den RPi wird ausgeführt. Im Menüpunkt „*3 Boot Options*“ wählen wir „*B1 Desktop / CLI*“ und dann im Untermenü den Punkt „*B2 Console Autologin*“. Die folgende Frage mit <*Yes*> beantworten und danach geht‘s mit <*Ok*> zurück ins Hauptmenü.

„*3 Boot Options*“ → „*B1 Desktop / CLI*“ → „*B2 Console Autologin*“ → <*Yes*> → <*Ok*>

Bei dieser Gelegenheit können wir auch gleich den **SPI-Bus aktivieren**. Dies geschieht ebenfalls im Konfigurationsprogramm **raspi-config**:

„*5 Interfacing Options*“ → „*P4 SPI*“ → <*Yes*> → <*Ok*>

Damit wir später über das Netzwerk auf den Raspberry Pi per **SSH** zugreifen können, um z.B. eigene Python Programm auszuführen, muss an dieser Stelle noch der SSH Server aktiviert werden:

„*5 Interfacing Options*“ → „*P2 SSH*“ → <*Yes*> → <*Ok*>

Jetzt können wir das Konfigurationsprogramm verlassen, einfach zwei Mal auf die Tabulatortaste drücken bis das Wort <*Finish*> rot hervorgehoben wird und dann die Eingabetaste drücken.

Nach diesen Änderungen muss ein Reboot durchgeführt werden, sollte raspi-config dies beim Verlassen des Programms nicht anbieten, folgenden Befehl eingeben:

```
sudo reboot
```

Nun können wir in den nächsten Schritten die benötigten Komponenten herunterladen und installieren. Für das Herunterladen der benötigten Dateien aus dem Internet muss der Raspberry Pi über eine aktive Internetverbindung verfügen.

3.1 Python Version

Für die *PiXtend Python Library (PPL)* wird Python in der Version 2.7.9 oder später benötigt, jedoch nicht Python 3.

Im Raspbian ist Python bereits vorinstalliert, die Version lässt sich leicht feststellen:

```
python --version
```

Python sollte sich mit Version 2.7.9 melden.

Damit die *PPL* Zugriff auf die GPIOs und den SPI-Bus des Raspberry Pi erhält, müssen zwei weitere Python Packages installiert sein. Die Packages heißen *RPi.GPIO*[3] und *spidev*[4] und können mit dem Programm *pip* leicht ausfindig gemacht werden:

```
pip freeze
```

Es folgt eine Liste mit allen installierten Python Packages und in dieser Liste sollte im oberen Teil RPi.GPIO und im unteren Teil spidev zu finden sein.

Diejenigen, die ein Raspbian Lite Image verwenden, müssen Python zu erst installieren, bevor die obigen Schritte ausgeführt und zum nächsten Schritt gegangen werden kann. Im Raspbian Lite Image fehlt die Python Entwicklungsumgebung:

```
sudo apt-get install python-dev
```

3 RPi.GPIO ist in Version 0.6.3 installiert
4 spidev liegt in Version 3.0 vor

3.2 PiXtend Python Library

Da alle Vorrausetzungen für die Installation der *PPL* erfüllt sind, legen wir im ersten Schritt ein Verzeichnis für das *PPL* Package an, laden dann das Package herunter und entpacken dieses in das erstellte Verzeichnis. Im letzten Schritt wird das *PPL* Package installiert. Ist dies alles geschehen, kann die *PiXtend Python Library* global in allen Python 2.7.9 Programmen verwendet werden.

Die Installation eines Python Packages kann viele Ausgaben auf der Konsole ausgeben, dies ist völlig normal.

Folgende Schritte der Reihe nach ausführen:

```
mkdir ppl
wget http://www.pixtend.de/files/downloads/ppl_v0.1.1.zip
unzip ppl_v0.1.1.zip -d ./ppl/
cd ppl
sudo python setup.py install
```

Wurde der letzte Befehl erfolgreich ausgeführt, so ist die *PiXtend Python Library* jetzt installiert und kann in eigenen Python Programmen verwendet werden.

Die Installation des PPL Packages kann, wie im vorherigen Kapitel gezeigt, mit *pip freeze* überprüft werden. In der Liste der installierten Packages befindet sich jetzt der Eintrag pixtendlib==0.1.1.

Alles, was wir jetzt noch brauchen, ist ein Editor, um Python Programme zu schreiben und ein Programm um die selbst erstellten Programme auf den Raspberry Pi zu übertragen und dann auszuführen.

HINWEIS:

*Die Funktionen und Eigenschaften der PPL dürfen nicht schneller als alle **100 ms** aufgerufen werden!*

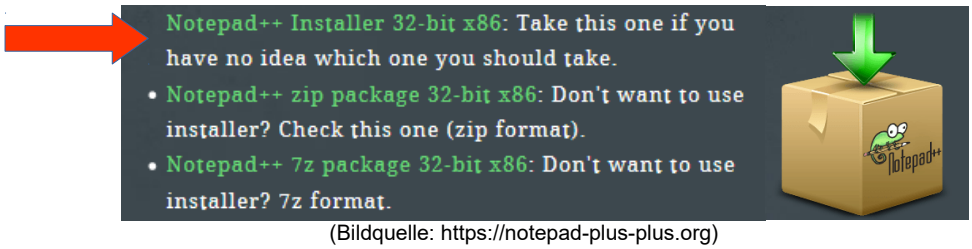
3.3 Programmiertool

Zur Erstellung von Python Programmen gibt es eine sehr große Anzahl an Programm bzw. Programmierumgebungen. Wir haben uns ein Programm aus dieser Menge herausgegriffen und zeigen alle Beispiele anhand dieses Programms.

In dieser Anleitung verwenden wir zur Erstellung von Python Programmen den Editor Notepad++ für Windows. Die 32Bit Version des Programms ist völlig ausreichend und bietet alles Nötige, um erste Python Programme zu schreiben.

Das Programm kann von folgender Adresse bezogen werden:

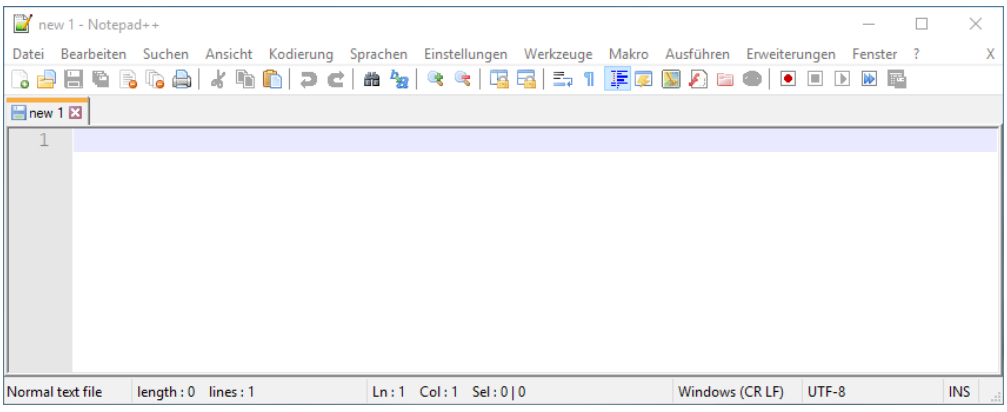
https://notepad-plus-plus.org



(Bildquelle: https://notepad-plus-plus.org)

Das Installationsprogramm von Notepad++ ausführen und den Schritten des Wizards folgen. Üblicherweise muss die Installation nicht angepasst werden und alle Schritte können einfach mit *Weiter* bzw. *Next* bestätigt werden.

Nach dem ersten Start zeigt Notepad++ die am Programm durchgeführten Änderungen gegenüber der Vor-Version an. Dieses Editor-Fenster kann einfach geschlossen werden, indem man auf das weiße X im roten Viereck klickt. Der Text im Fenster verschwindet und man erhält ein leeres Editor-Fenster.



Möchten Sie nicht unter Windows arbeiten, sondern direkt auf dem Raspberry Pi mit Bildschirm und Tastatur, können Sie den vorinstallierten Texteditor *nano* verwenden. Weitere Informationen zu diesem und anderen Editoren finden Sie auf Raspberry Pi Org https://www.raspberrypi.org/documentation/linux/usage/text-editors.md oder auf der *nano* Homepage unter https://www.nano-editor.org.

Die Kapitel 3.4 und 3.5 können in diesem Fall übersprungen werden, da kein SSH-Client benötigt wird, und das Programm WinSCP entfällt ebenfalls.

Alle fortgeschrittenen Anwender und Programmierer, die bereits über viel Erfahrung verfügen und nach einer umfassenderen Lösung suchen, können sich von der tschechischen Firma JetBrains das Programm *PyCharm* anschauen. Hierbei handelt es sich um eine vollständige Python Entwicklungsumgebung mit vielen Zusatzfunktionen.

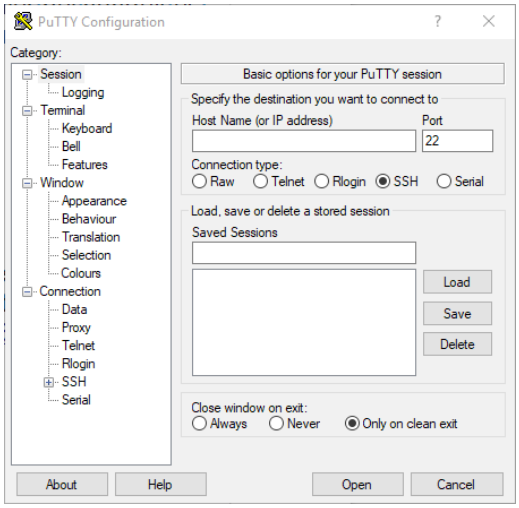
Unter folgender Adresse gibt es mehr Informationen zu diesem Tool:

https://www.jetbrains.com/pycharm/

3.4 SSH-Client

Damit auf dem Raspberry Pi Befehle über das Netzwerk ausgeführt werden können, ist ein sog. SSH-Client (Secure Shell Client) notwendig. Ausführliche Informationen zu SSH können Sie z.B. auf Wikipedia finden: https://de.wikipedia.org/wiki/Secure_Shell.

Ein solcher SSH-Client ist beispielsweise das Programm PuTTY.exe – www.putty.org. Im Folgenden werden wir dieses Programm nutzen, um uns mit dem Raspberry Pi zu verbinden und Befehle auszuführen. Auf der Programm Homepage gibt es ein Installationsprogramm, das unter Windows eine bequeme Einrichtung erlaubt.

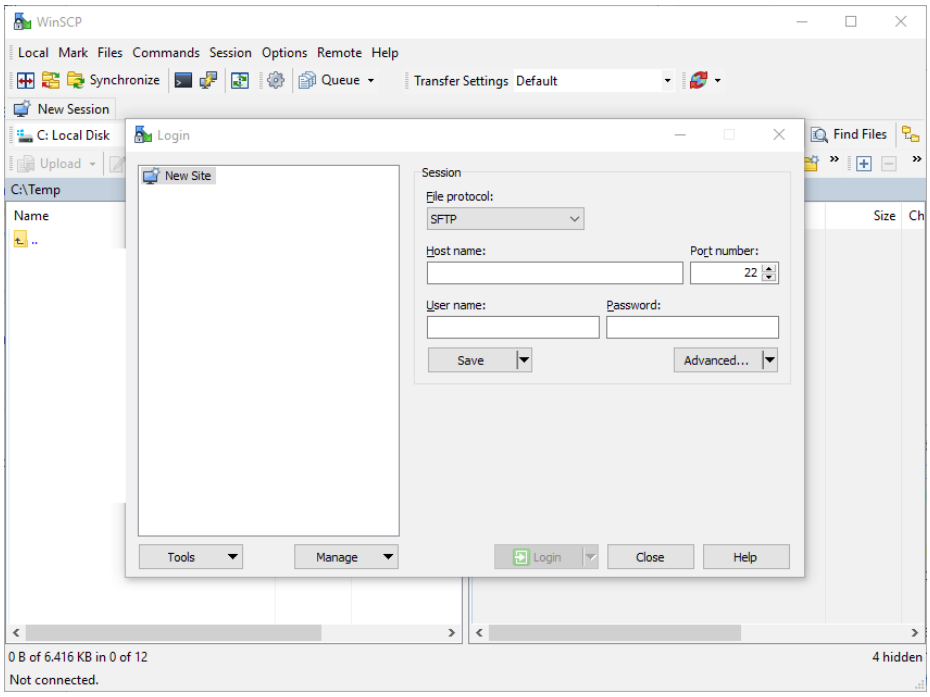


3.5 WinSCP (Windows Secure Copy)

WinSCP ermöglicht das Übertragen von Dateien von einem Computer zum nächsten unter Vewendung verschiedener Datenübertragungsprotokolle, darunter befindet sich auch das *SSH* Protokoll, welches auch von PuTTY eingesetzt wird.

WinSCP bietet uns gegenüber PuTTY jedoch die Möglichkeit, über eine grafische Oberfläche Dateien auf den Raspberry Pi zu laden oder von dort zu holen.
Der Grund, warum wir zwei Programme einsetzen: Bei *WinSCP* ist die eingebaute Konsole bzw. der SSH-Client um dem Raspberry Pi Befehle zu schicken, nicht so praktisch wie dies mit PuTTY der Fall ist.

Die Oberfläche von *WinSCP* ist auch in deutscher Sprache verfügbar, nachdem man von der *WinSCP* Homepage die deutsche Sprachdatei geladen und in das *WinSCP* Programmverzeichnis entpackt hat. In *Preferences (Einstellungen)* lässt sich unter *Languages (Sprachen)* Deutsch bzw. German auswählen.



Damit haben wir nun alle Software-Komponenten vorbereitet. Im nächsten Kapitel geht es ans Testen und Programmieren!

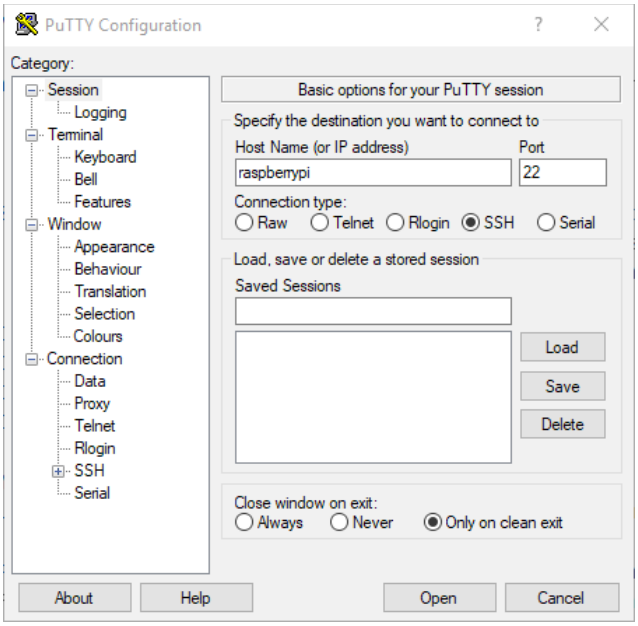
4. Programmierung

Im ersten Schritt bedienen wir uns eines Beispielprogramms, das sich bereits auf dem Raspberry Pi befindet. Es ist Teil der *PiXtend Python Library* und befindet sich im Ordner */home/pi/ppl/examples*.

Diejenigen, die direkt am Raspberry Pi arbeiten, können den ersten Schritt überspringen und direkt mit dem zweiten beginnen.
Alle, die sich per Netzwerk, über SSH mit PuTTY, mit dem Raspberry Pi verbinden möchten, beginnen mit Schritt 1.

1. Einloggen

PuTTY starten und unter *Host Name (or IP address)* den Namen oder die IP-Adresse des Raspberry Pi eingeben. Sicherstellen, dass unter *Port* 22 eingestellt und unter *Connection type* SSH angewählt ist. Mit einem Klick auf *Open* wird die Verbindung zum Raspberry Pi aufgebaut.
Benutzername ist „pi“ und das Passwort ist „raspberry“ ohne die Anführungsstriche.

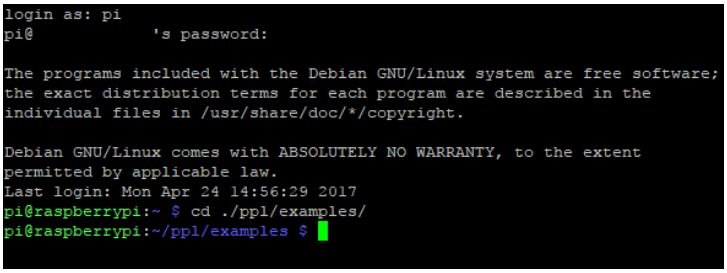


2. Beispiel-Verzeichnis

Nach dem Login sind wir gleich im *pi*-Benutzer Home-Verzeichnis (*/home/pi*). Hier befindet sich bereits der Ordner *ppl,* der eingangs erstellt wurde und in dem sich der Quellcode der *PiXtend Python Library* und der Unterordner *examples* mit den Beispielen befinden.

Wir wechseln in das Verzeichnis mit den Beispielen:

```
cd ./ppl/examples/
```



3. Beispiel ausführen

Damit sich gleich was am PiXtend Board tut, starten wir das Beispiel für die digitalen Ausgänge und die Relais. Das Schalten der Relais ist gut hörbar und die LEDs zeigen zusätzlich den Zustand an.

Mit folgendem Befehl geht es los:

```
sudo python px_digital_output_demo.py
```

```
pi@raspberrypi:~/ppl/examples $ sudo python px_digital_output_demo.py

PiXtend Python Library (PPL) demo for digital outputs including relays.

Running Main Program - Hit Ctrl + C to exit
Auto Mode - Communication is not yet up...Please wait...
One time configuration: Setting the relays and digital outputs in an alternating
 pattern
The value 0 = OFF and the value 1 = ON

Cycle No.: 2642
Digital Output 0: 0
Digital Output 1: 1
Digital Output 2: 0
Digital Output 3: 1
Digital Output 4: 0
Digital Output 5: 1
Relay 0:          0
Relay 1:          1
Relay 2:          0
Relay 3:          1
^C
PiXtend Python Library (PPL) demo for digital outputs including relays finished.
pi@raspberrypi:~/ppl/examples $
```

Nach dem Start des Programms ändern die digitalen Ausgänge und die Relais etwa alle 2,5 Sekunden ihre Zustände. In der Konsole wird der aktuelle Zustand in Textform angezeigt, wobei eine 0 Aus und eine 1 Ein bedeutet, ferner gibt es einen Zykluszähler, der sich alle 100 ms um eins erhört. Nicht, dass man denkt, das Programm würde nichts tun.

Mit den Tasten Strg + C (Ctrl + C) kann das Python Programm beendet werden.

4. Fertig

Wurde das Python Programm mit dem Tastendruck Strg + C (Ctrl + C) beendet, können wir den ersten Test mit Python und dem PiXtend Board erfolgreich abschließen und uns im Weiteren um die Programmierung kümmern und unser erstes Programm erstellen.

4.1 Die PiXtend Python Library (PPL)

Bisher haben wir die *PiXtend Python Library* zusammen mit einem Beispiel angewendet. Um ein Programm erstellen zu können, ist jedoch Wissen über die *PPL* selbst notwendig, und zwar welche Eigenschaften es gibt und welche Funktionen verwendet werden müssen, damit sich überhaupt etwas tut.

Im Folgenden werden kurz die wichtigsten Funktionen und Eigenschaften aufgeführt die notwendig sind, um mit den digitalen Ein- und Ausgängen zu arbeiten. Eine vollständige Übersicht aller Eigenschaften und Funktionen befindet sich im *Anhang A* dieses Dokuments. Ferner empfehlen wir, mit dem Programm *WinSCP* die Python Beispiele vom Raspberry Pi herunterzuladen und diese anzuschauen, wenn Sie z.B. die analogen Ein- und Ausgänge, GPIOs oder Modellbau-Servomotoren nutzen möchten. Im *PPL* Verzeichnis auf dem Raspberry Pi ist ein *doc* Verzeichnis vorhanden. Hier liegt eine HTML-Datei, die mit dem Python Programm *pydoc* erstellt wurde. Hierbei handelt es sich um eine rein technische (automatisch generierte) Dokumentation des gesamten *PPL Packages, die* und zeigt alle öffentlichen Funktionen und Eigenschaften zusammen mit Kommentaren aus dem Sourcecode zeigt.

Die *PPL* umfasst im Wesentlichen eine Python Klasse: die ***„Pixtend“*** Klasse die von einem Python Objekt abgeleitet ist. Diese Klasse bietet dem Anwender eine Reihe von Eigenschaften und Funktionen, um den Mikrocontroller auf dem PiXtend Board zu konfigurieren und die digitalen und analogen Ein- und Ausgänge sowie die GPIOs abzufragen bzw. anzusteuern.

Die folgende Tabelle zeigt die wichtigsten Funktionen und Eigenschaften[5]:

Name	Typ	Beschreibung
open	Funktion	Die *open* Funktion startet den Python SPI Treiber, und öffnet den SPI Master 0 mit Chip Select Leitung 0 und stellt die für den Mikrocontroller notwendigen 100 kHz Übertragungsgeschwindigkeit ein. Die *open* Funktion muss nach der Erstellung einer „Pixtend“ Instanz als nächstes aufgerufen werden, wird eine andere Funktion oder Eigenschaft verwendet, gibt es einen IOError. Beispiel: p = Pixtend() p.open()
auto_mode	Funktion	Die *auto_mode* Funktion kümmert sich um die Kommunikation mit dem Mikrocontroller und sollte nach der *open* Funktion als nächstes aufgerufen werden, am besten in einer Endlosschleife oder solange, bis die Funktion 0 zurückliefert und die Eigenschaft *uc_status* den Wert 1 hat. Ab diesem Zeitpunkt kann die

5 Bedeutung (r) und (rw) hinter einer Eigenschaft: r = read only / nur lesen und rw = read and write / lesen und schreiben

		auto_mode Funktion auch sporadisch aufgerufen werden, z.B. immer dann, wenn neue Werte benötigt werden oder ein Ausgang gesetzt werden soll. Ein zyklischer Aufruf wird empfohlen, ist aber nicht zwingend. *Es gibt eine Einschränkung: Die auto_mode Funktion darf nicht schneller als alle 100 ms aufgerufen werden.* Beispiel: if p.auto_mode() == 0: p.relay0 = p.ON
close	Funktion	Soll das Python Programm beendet werden, wird empfohlen, die *close* Funktion aufzurufen und erst im Anschluss die „Pixtend“ Instanz zu löschen. Die *close* Funktion setzt alle internen Variablen und Objekte zurück und schließt den SPI Treiber. Dieses Vorgehen soll helfen, Speicherlecks zu verhindern und das Python Programm "*sauber*" zu beenden. Somit kann man das eigene Python Programm auch gleich wieder starten, ohne dass es eine Fehlermeldung gibt. Beispiel: p.close() p = None
digital_input0 .. 7	Eigenschaft (r)	Mit diesen 8 *nur lesen* Eigenschaften (*digital_input0* bis *digital_input7)* können die Zustände der digitalen Eingänge vom PiXtend Board gelesen werden. Die Eigenschaften liefern entweder den Wert 0 für *aus* (OFF) oder den Wert 1 für *an* (ON).
digital_output0 .. 5	Eigenschaft (rw)	Die 6 digitalen Ausgänge können über die Eigenschaften *digital_output0* bis *digital_output5* sowohl gelesen als auch geschrieben werden. Wird der Eigenschaft der Wert 0 zugewiesen, so geht der entsprechende digitale Ausgang *aus* (OFF) oder bei der Zuweisung des Wertes 1 geht der Ausgang *an* (ON).
relay0 .. 3	Eigenschaft (rw)	Die 4 Relais auf dem PiXtend Board können über die Eigenschaften *relay0* bis *relay3* sowohl gelesen als auch geschrieben werden. Wird der Eigenschaft der Wert 0 zugewiesen, so geht das entsprechende Relais *aus* (OFF) oder bei der Zuweisung des Wertes 1 geht es *an* (ON).
uc_board_version	Eigenschaft (r)	Über diese *nur lesen* Eigenschaft kann die PiXtend Board Version ausgelesen werden. Der Wert 12 steht für Board Version 1.2.x, 13 für Board Version 1.3.x usw.
uc_fw_version	Eigenschaft (r)	Über diese *nur lesen* Eigenschaft kann die Firmware Version des Mikrocontrollers auf dem PiXtend Board ermittelt werden.
uc_status	Eigenschaft (r)	Die *uc_status* Eigenschaft liefert den aktuellen Status des Mikrocontrollers auf dem PiXtend Board. Eine 0 entspricht *aus* bzw. *Auto Mode* ist nicht aktiv. Eine 1 hingegen besagt, dass der MC im *Auto Mode* betrieben wird und sich im Status *Run* befindet.
ON	Konstante	Entspricht dem Dezimalwert 1.
OFF	Konstante	Entspricht dem Dezimalwert 0.

4.2 Erstes Programm

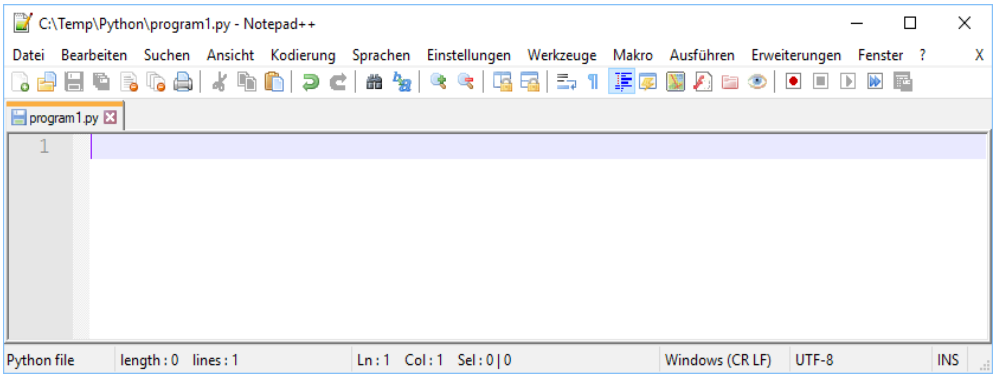
Die Grundlagen zur Programmierung von PiXtend mit der *PPL* in Python sind geschafft, jetzt können wir unser erstes Programm schreiben. Die nachfolgenden Schritte zeigen einen Weg auf, wie eigene Python Programme erstellt werden können.

1. Start

Als Erstes benötigen wir einen Texteditor, unter Windows z.B. das erwähnte *Notepad++*, unter Linux den Editor *nano*, in den wir unser Programm eingeben können. Die nun folgenden Schritte werden unter Windows durchgeführt und die eingangs genannten Programme *Notepad++*, *WinSCP* und *PuTTY* kommen zum Einsatz.

2. Editor öffnen

Den Texteditor *Notepad++* starten und eine neue leere Textdatei erstellen. Die Datei am besten gleich speichern, z.B. *program1.py*. Das Speichern führt dazu, dass *Notepad++* automatisch die Sprache *Python* einstellt und für diese Sprache das Syntax-Highlighting aktiviert, was uns beim Programmieren hilft.



3. Das Programm erstellen

Alles ist vorbereitet, jetzt können wir mit der *Python* Programmierung los legen. Als erste Aufgabe lassen wir das Relais 0 zyklisch jede Sekunde *an* und *aus* gehen.

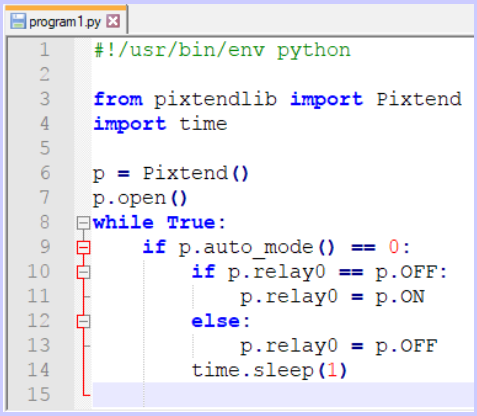
Das folgende Programm soll diese Aufgabe für uns erledigen. Einfach das Programm in *Notepad++* eingeben oder rüber kopieren und abspeichern. Bei manueller Eingabe bitte ganz genau auf die Einrückung achten und nicht *Leerzeichen* mit *Tabulatoren* mischen, sonst läuft das Programm später nicht und es gibt Fehlermeldungen, *Python* versteht hier keinen Spaß.

Das erste Programm:

```
#!/usr/bin/env python

from pixtendlib import Pixtend
import time

p = Pixtend()
p.open()
while True:
    if p.auto_mode() == 0:
        if p.relay0 == p.OFF:
            p.relay0 = p.ON
        else:
            p.relay0 = p.OFF
        time.sleep(1)
```



Programm Erläuterung:

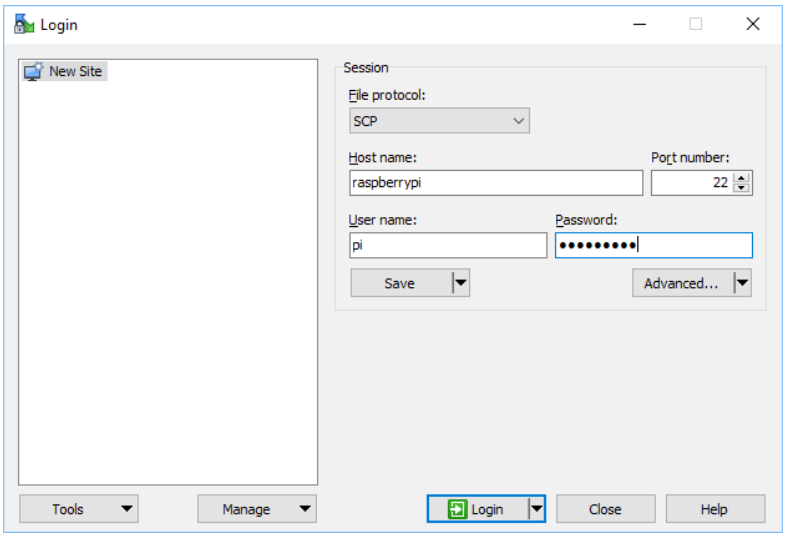
- In der ersten Zeile geben wir bekannt, dass es sich um ein *Python* Programm handelt, das Programm läuft ja später auf dem Raspberry Pi.
- Zum Warten am Ende des Programms importieren wir noch die *time* Klasse.
- Danach muss die „Pixtend“ Klasse importiert werden, so erhalten wir Zugriff auf die erwähnten Eigenschaften und Funktionen und eine Kommunikation mit dem Mikrocontroller und DAC auf dem PiXtend Board wird möglich.
- In einer *While* Schleife rufen wir die *auto_mode* Funktion auf, dies können wir in *Python* gleich mit einer *if*-Abfrage verbinden, was uns eine kurze und knappe Code-Zeile ermöglicht. Wir vergleichen die Rückantwort der *auto_mode* Funktion mit der Zahl 0 (Null), d.h. wir prüfen, ob alles in Ordnung ist. Liefert die Funktion den Wert

-1 zurück, dann gibt es ein Problem bei der Kommunikation mit dem Mikrocontroller.

- **ACHTUNG**: Die *auto_mode* Funktion nicht schneller als alle ***100 ms*** (0,1 Sekunde) aufrufen.
- War die Abfrage erfolgreich, dann können wir z.B. das Relais 0 zyklisch ein- und ausschalten.
- Die Konstanten ON und OFF können durch 1 und 0 ersetzt werden. Das Arbeiten mit Namen soll das eigentliche Vorgehen besser verdeutlichen.
- Ganz am Schluss warten wir eine Sekunde, um das Relais 0 nicht überzustrapazieren. Wie oben erwähnt, das Warten ist Pflicht!

4. Programm an Raspberry Pi übertragen

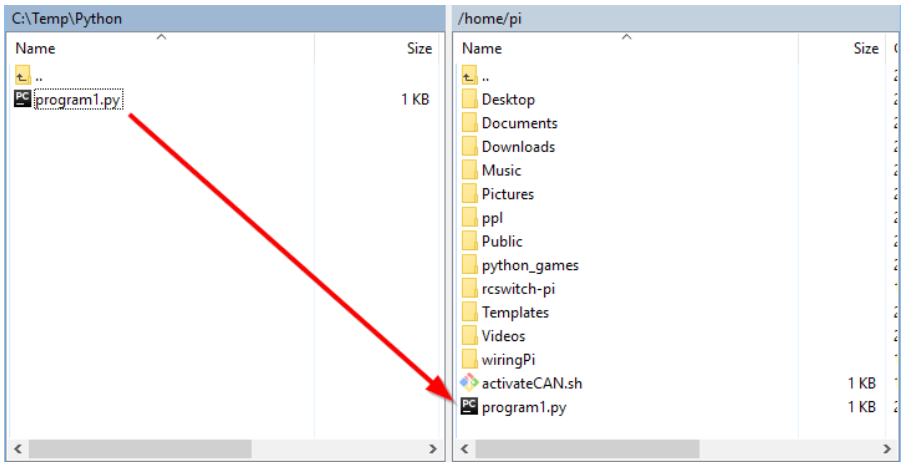
Mit dem Programm *WinSCP* übertragen bzw. kopieren wir das Programm / die Python-Datei auf den Raspberry Pi. Wer direkt am Rpi arbeitet, kann diesen Schritt auslassen und mit Schritt 5 fortfahren.
Nach dem Start des Programm muss eine neue Verbindung (Site) angelegt werden, als Protokoll wählen wir SCP und können auch gleich den Raspberry Pi Benutzer *pi* und das Passwort angeben. Mit einem Klick auf *Save (Speichern)* wird die neue Verbindung angelegt, dann wird mit einem Klick auf *Login* die Verbindung zum Raspberry Pi hergestellt.



WinSCP Login Bildschirm

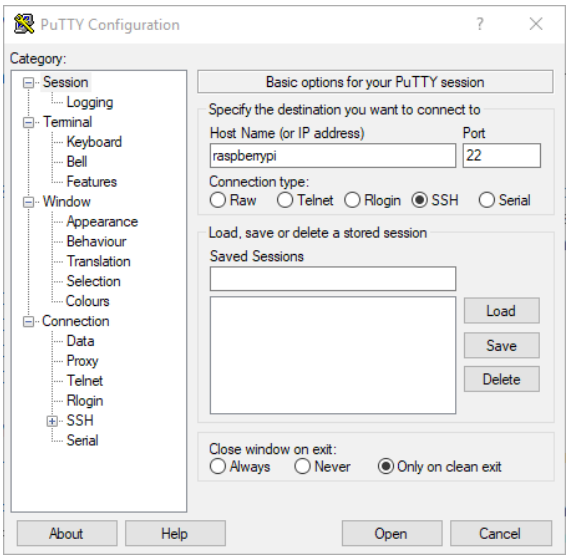
Ist dies die erste Verbindung mit *WinSCP* zum Raspberry Pi, dann erscheint noch eine Sicherheitsabfrage, die bestätigt werden muss.

Wurde die Verbindung aufgebaut, so kann die Python-Datei *program1.py* auf den Raspberry Pi übertragen/kopiert werden. Dies lässt sich in *WinSCP* praktischerweise einfach per Drag & Drop erledigen und bei der geringen Dateigröße dauert die Übertragung nur einen Moment.



5. Programm ausführen

Mit dem Programm PuTTY können wir eine *SSH* Verbindung zum Raspberry Pi aufbauen und über eine textbasierte Konsole Befehle ausführen lassen, wie z.B unser *Python* Programm.
Für eine Verbindung zum RPi einfach den Namen oder die IP-Adresse eingeben, unter Port 22 einstellen und als Verbindungstyp (Connection type) SSH wählen.



Mit einem Klick auf *Open* wird eine Verbindung zum Minicomputer hergestellt.

Ist dies die erste Verbindung, die mit dem Raspberry Pi hergestellt wird, dann erscheint eine Warnung bzw. ein Hinweis, der bestätigt werden muss.

Danach erhalten wir eine Eingabeaufforderung, bei der wir zuerst den Benutzer *pi* eingeben und dann das dazugehörige Passwort. Jede

Eingabe muss mit der Eingabetaste (Enter) bestätigt werden.

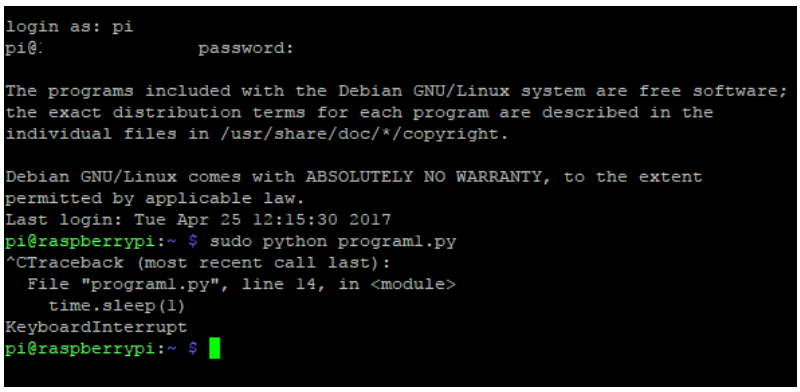
Hat der Login funktioniert, dann bekommen wir ein paar Texte, Hinweise und die Eingabeaufforderung vom Linux Betriebssystem auf dem Raspberry Pi wird angezeigt.

Mit dem folgenden Befehl starten wir unser Programm:

```
sudo python program1.py
```

Nach etwa 1 Sekunde fängt das Relais 0 an, sich ein- und auszuschalten. Wie von uns programmiert, geschieht dies jetzt jede Sekunde. So lange wir das Programm laufen lassen bzw. der Raspberry Pi Strom hat, wird auch das Relais 0 jede Sekunde klicken und klacken.

Mit der Tastenkombination Strg + C (Ctrl + C) lässt sich das *Python* Programm beenden. Es erscheint eine Fehlermeldung, dass das Programm unterbrochen wurde und das ist in Ordnung so. Wir haben keine Fehlerbehandlung in unser Programm eingebaut, die sich um diese spezielle Tastatureingabe kümmert. In den Beispielen zur *PPL* wird gezeigt, wie man eine solche Fehlerbehandlung einbauen kann.



Herzlichen Glückwunsch!
Sie haben Ihr erstes Python-Programm auf PiXtend in Betrieb genommen.

Das erste *Python* Programm wäre geschafft, jetzt können Sie es nach Belieben verändern, erweitern oder ganz neu beginnen. Den Zugang zum PiXtend Board mit *Python* und der *PPL* haben Sie jetzt.

Wir wünschen viel Spaß und Erfolg beim Programmieren!

5. Weitere Informationen

In diesem Kapitel möchten wir Ihnen weitere Informationen mitgeben, damit Sie mehr Funktionen von PiXtend in Ihren Python-Programmen verwenden können.

5.1 Verfügbare PiXtend I/Os

In der aktuellen Version der *PPL* (Stand: April 2017) werden alle digitalen und analogen Ein- und Ausgänge des PiXtend Boards unterstützt.

Da die analogen Ausgänge über einen DAC laufen und dieser am SPI Bus am Chip Select 1 hängt, muss dies im Python-Programm berücksichtigt werden. Es gibt hierzu eine weitere Funktion mit Namen *open_dac,* um die Kommunikation mit dem Chip zu starten.

Die 4 GPIOs auf dem PiXtend Board können wahlweise direkt genutzt oder jeder einzelne als DHT11/22 Eingang verwendet werden.

Die zwei PWMs stehen ebenfalls zur Verfügung.

Informationen zu den verschiedenen Modi der PWMs, analogen Eingänge und GPIOs finden Sie im Download-Bereich unter http://www.pixtend.de/downloads/, nachdem Sie Ihre PiXtend Version ausgewählt haben.

Schauen Sie dazu auch in die App-Notes Control- und Status-Bytes und Prozessdaten von PiXtend.

5.2 Umgang mit dem Auto(matic) Mode

Der Auto Mode ist die effizienteste Methode, die Funktionen von PiXtend in *Python* zu verwenden. Hier kann man alle 100 ms neue Werte bekommen, z.B. von den analogen Eingängen, oder man kann die digitalen Ein- und Ausgänge schalten. Diese Funktion muss spätestens immer dann aufgerufen werden, wenn der Zustand des PiXtend Boards geändert werden soll. Ein zyklischer Aufruf ist nicht zwingend erforderlich.

Möchte man jedoch diesen Teil nicht nutzen bzw. genügt auch eine langsame Reaktion des PiXtend Boards, so kann man auf den Auto Mode verzichten und alle Eigenschaften im sog. manuellen Modus verwenden. Der benötigte Aufruf der *auto_mode* Funktion kann somit entfallen. Der manuelle Modus ist jedoch sehr langsam, da bei jeder Veränderung einer Eigenschaft eine vollständige SPI Datenübertragung stattfindet und die Daten vom Mikrocontroller erst verarbeitet werden müssen.

5.3 Python Programm automatisch starten

Nach einem Reboot / Power-Up des Raspberry Pi startet ein Python-Programm nicht automatisch. Wie bereits erwähnt kann ein Python-Programm mit folgendem Befehl gestartet werden, das *myprogram.py* bitte durch den eigenen Programmnamen ersetzen:

```
sudo python myprogram.py
```

Wir können diesen Befehl aber auch automatisch beim Hochfahren des Linux-Systems ausführen lassen. Wir sind ja schließlich Automatisierer!

Die Änderung ist in der Linux-Console schnell erledigt:

```
sudo nano /etc/rc.local
```

Es öffnet sich die Datei „rc.local“ mit etwas Inhalt. Wir tragen vor der Zeile „exit 0“ zwei neue Zeilen ein mit der Annahme, das zu startende Programm liegt im Home-Verzeichnis:

```
cd /home/pi/
sudo python myprogram.py &
```

Das „&“ ist kein Tippfehler. Damit startet das Python-Programm als Prozess im Hintergund.

Abspeichern und mit STRL+X verlassen → Rebooten

ACHTUNG: Das Programm bitte vorher gut testen! Hat das Programm einen Fehler, z.B. dass die digitalen Ausgänge unkontrolliert oder sehr schnell angesteuert werden, so kann dies ggf. die angeschlossene Peripherie beschädigen und da das fehlerhafte Programm als "Autostartprogramm" eingerichtet ist, hilft also auch kein Neustart, um das Problem zu beheben.

5.4 Verwendung der seriellen Schnittstelle

Die Verwendung der seriellen Schnittstelle ist nicht im *PPL* Package enthalten, hier kann z.B. auf *pySerial* zurückgegriffen werden, da es sich hier um ein Raspberry Pi und Python Thema handelt und nicht direkt etwas mit dem PiXtend Board zu tun hat. Bei der in dieser App-Note verwendeten Raspbian Version läuft die serielle Schnittstelle beispielsweise über das Linux Gerät */dev/ttyS0*.
Ferner gilt zu beachten, dass hier von Haus aus die Ausgaben der Linux Konsole ausgegeben werden, man muss also vor Verwendung die */boot/cmdline.txt* bearbeiten und den Teil *console=serial0,115200* aus der ersten Zeile entfernen.

Eine Anmerkung: Das PiXtend Board führt die serielle Schnittstelle des RPi nach außen und standardmäßig wird die RS232 Schnittstelle angesprochen. Da es aber auch eine RS485 Schnittstelle gibt, muss eine Umschaltung erfolgen. Diese Umschaltung kann mit der *PPL* vorgenommen werden.

Die Umschaltung geht beispielsweise so:

```
# RS232 Mode (default)
    p.serial_mode = p.RS232
# RS485 Mode einschalten
    p.serial_mode = p.RS485
```

5.5 Verwendung des CAN-Bus

Die Verwendung des CAN-Busses wird nicht von der *PPL* abgedeckt, da dies Teil des Raspberry Pi im Zusammenspiel mit PiXtend ist und zusätzliche vorbereitende Schritte erfordert.

In unserem Downloadbereich finden Sie eine App-Note zur Inbetriebnahme des CAN-Busses auf dem Raspberry Pi, damit der MCP2515 CAN Chip auf PiXtend genutzt werden kann. Die App-Note wurde zwar für die CODESYS V3 Runtime erstellt, die einleitenden Schritte bis zum Ende von *Kapitel 3* sind jedoch allgemeingültig, der Teil, der sich auf CODESYS V3 bezieht, kann einfach ignoriert werden.

Direkter Link zur App-Note:

http://www.pixtend.de/files/manuals/APP-PX-530_Codesys_CAN_DE.pdf

6. Frequently Asked Questions (FAQ)

Ich kann mein Programm nicht starten, Python gibt eine Fehlermeldung aus über eine ungültige *Identation*?

Hier liegt vermutlich ein Problem bei der Einrückung der verschiedenen Code-Zeilen im Programm vor. Noch mal genau schauen, dass bei jeder Zeile die Einrückung stimmt. Wir verwenden z.B. immer 4 Leerzeichen pro Einrückungsebene, fehlt nur ein Leerzeichen oder man hat Leerzeichen und Tabulatoren gemischt, gibt Python eine Fehlermeldung aus. In Notepad++ z.B. kann man sich die „nicht druckbaren“ Zeichen anzeigen lassen und sieht so sehr schnell wo etwas nicht stimmt, zudem sagt Python selbst, in welcher Zeile es den Fehler festgestellt hat.

Kann ich mein Programm ohne *sudo* laufen lassen?

Dies scheint mit Python in dem hier verwendeten Release von Raspbian tatsächlich möglich zu sein, wir empfehlen dennoch, *sudo* zu verwenden, damit Python auch Systemfunktionen ausführen kann. Hier kann es sonst schnell zu Problemen kommen deren Ursache ggf. im ersten Schritt nicht zu erklären sind.

Die angegebenen 100 ms Zykluszeit sind mir zu langsam, kann PiXtend nicht schneller angesprochen werden?

Ja! Dies geht. Die 100 ms sind ein Vorschlag von uns, damit alle Funktionen von PiXtend genutzt werden können. Verwenden Sie keine DHT11/22 Sensoren an den GPIO0 – GPIO3, können Sie PiXtend auch mit ***25 ms*** ansprechen, also viermal schneller. Noch schneller geht es dann aber nicht.

Gerne möchten wir Sie für den Informationsaustausch in die Foren von Qube Solutions einladen:

http://www.pixtend.de/forum/

7. Anhang A

Nachstehend folgt eine Tabelle mit allen öffentlichen Eigenschaften und Funktionen der *PiXtend Python Library,* die ein Endanwender benutzen kann.

Name	Typ	Datentyp	Zugriff[6]	Werte	Beschreibung
auto_mode()	Funktion	int	-	0 = OK, -1 = Fehler	Aktivierung des Auto Mode und Kommunikation mit dem Mikrocontroller auf dem PiXtend Board, sollte zyklisch aufgerufen werden, Minimum ist 100ms.
close()	Funktion	-	-		Schließen des SPI Treibers und zurücksetzen aller internen Variablen und Objekte. Aufrufen vor Ende des eigenen Programms.
open()	Funktion	-	-		Öffnen des SPI Masters 0 mit Chip Select 0 zur Kommunikation mit dem Mikrocontroller auf dem PiXtend Board. Darf nur 1x aufgerufen werden.
open_dac()	Funktion	-	-		Öffnen des SPI Masters 0 mit Chip Select 1 zur Kommunikation mit dem DAC.
pwm_ctrl_configure()	Funktion	-	-		Konfiguration der PWM Kanäle festlegen, muss nach jeder Änderung der PWM Einstellungen aufgerufen werden.
set_dac_output()	Funktion	-	-		Übertragung der Konfiguration und Analogwerte für den DAC A und DAC B. Vor Aufruf mit *dac_selection* Ziel DAC festlegen.
update_rtc()	Funktion	-	-		Linux Systemzeit in die RTC auf dem PiXtend Board schreiben.
analog_input0	Eigenschaft	float	L	0 V .. 5 V/10 V	Analog Eingang auf dem PiXtend Board, liefert Werte immer in der Einheit *Volt*. Wertebereich abhängig von *analog_input0_10volts_jumper*.
analog_input0_10volts_jumper	Eigenschaft	int	L / S	0 = Aus, 1 = An	Angabe, ob der 5V/10V Jumper auf dem PiXtend Board physikalisch gesteckt ist.
analog_input0_nos	Eigenschaft	int	L / S	1, 5, 10, 50	Anzahl der Samples, die der MC vornehmen soll. Der Defaultwert ist 10.
analog_input0_raw	Eigenschaft	int	L	16 Bit Wert	Rohwert vom Mikrocontroller ohne Umrechnung.
analog_input1	Eigenschaft	float	L	0 V .. 5 V/10 V	Analog Eingang auf dem PiXtend, liefert Werte immer in der Einheit *Volt*. Wertebereich abhängig von *analog_input1_10volts_jumper*.
analog_input1_10volts_	Eigenschaft	int	L / S	0 = Aus, 1 = An	Angabe, ob der 5 V/10 V Jumper auf dem

6 L = Lesen , S = Schreiben, L / S = Lesen und Schreiben

Name	Typ	Datentyp	Zugriff	Werte	Beschreibung
jumper					PiXtend Board physikalisch gesteckt ist.
analog_input1_nos	Eigenschaft	int	L / S	1, 5, 10, 50	Anzahl der Samples die der MC vornehmen soll. Der Defaultwert ist 10.
analog_input1_raw	Eigenschaft	int	L	16 Bit Wert	Rohwert vom Mikrocontroller ohne Umrechnung.
analog_input2	Eigenschaft	float	L	0 mA .. 20 mA	Umgerechneter Analogwert vom Mikrocontroller in 0 bis 20 Milliampere.
analog_input2_nos	Eigenschaft	int	L / S	1, 5, 10, 50	Anzahl der Samples, die der MC vornehmen soll. Der Defaultwert ist 10.
analog_input2_raw	Eigenschaft	int	L	16 Bit Wert	Rohwert vom Mikrocontroller ohne Umrechnung.
analog_input3	Eigenschaft	float	L	0 mA .. 20 mA	Umgerechneter Analogwert vom Mikrocontroller in 0 bis 20 Milliampere.
analog_input3_nos	Eigenschaft	int	L / S	1, 5, 10, 50	Anzahl der Samples, die der MC vornehmen soll. Der Defaultwert ist 10.
analog_input3_raw	Eigenschaft	int	L	16 Bit Wert	Rohwert vom Mikrocontroller ohne Umrechnung.
analog_input_nos_freq	Eigenschaft	float	L / S	0.125, 0.250, 0.500, 1.0, 2.0, 4.0, 8.0 MHz	Abtastfrequenz der analogen Eingänge.
dac_selection	Eigenschaft	int	L / S	0 = DAC A, 1 = DAC B	Auswahl des DAC, der geändert werden soll. DAC A = AO 0, DAC B = AO 1.
dht0	Eigenschaft	int	L / S	0 = Aus, 1 = An	Konfiguration des GPIO 0 als DHT11/22 Eingang. Bei 1 (An) werden alle anderen GPIO Einstellungen ignoriert.
dht1	Eigenschaft	int	L / S	0 = Aus, 1 = An	Konfiguration des GPIO 1 als DHT11/22 Eingang. Bei 1 (An) werden alle anderen GPIO Einstellungen ignoriert.
dht2	Eigenschaft	int	L / S	0 = Aus, 1 = An	Konfiguration des GPIO 2 als DHT11/22 Eingang. Bei 1 (An) werden alle anderen GPIO Einstellungen ignoriert.
dht3	Eigenschaft	int	L / S	0 = Aus, 1 = An	Konfiguration des GPIO 3 als DHT11/22 Eingang. Bei 1 (An) werden alle anderen GPIO Einstellungen ignoriert.
di0	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
di1	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
di2	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.

Name	Typ	Datentyp	Zugriff	Werte	Beschreibung
di3	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
di4	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
di5	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
di6	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
di7	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs in Kurzschreibweise.
digital_input0	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input1	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input2	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input3	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input4	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input5	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input6	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_input7	Eigenschaft	int	L	0 = Aus, 1 = An	Zustand des digitalen Eingangs.
digital_output0	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich.
digital_output1	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich.
digital_output2	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich.
digital_output3	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich.
digital_output4	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich.
digital_output5	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich.
do0	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich. Hier handelt es sich um eine Kurzschreibweise des entsprechenden digital Ausgangs und nicht um einen weiteren Ausgang.
do1	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich. Hier handelt

Name	Typ	Datentyp	Zugriff	Werte	Beschreibung
					es sich um eine Kurzschreibweise des entsprechenden digital Ausgangs und nicht um einen weiteren Ausgang.
do2	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich. Hier handelt es sich um eine Kurzschreibweise des entsprechenden digital Ausgangs und nicht um einen weiteren Ausgang.
do3	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich. Hier handelt es sich um eine Kurzschreibweise des entsprechenden digital Ausgangs und nicht um einen weiteren Ausgang.
do4	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich. Hier handelt es sich um eine Kurzschreibweise des entsprechenden digital Ausgangs und nicht um einen weiteren Ausgang.
do5	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des digitalen Ausgangs. Lesen und Schreiben sind immer möglich. Hier handelt es sich um eine Kurzschreibweise des entsprechenden digital Ausgangs und nicht um einen weiteren Ausgang.
gpio0	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des GPIO, Achtung! Das "*Setzen*" dieser Eigenschaft geht nur wenn der GPIO als "*Ausgang*" konfiguriert wurde, sonst gibt es ein *IOError.*
gpio0_direction	Eigenschaft	int	L / S	0 = Eingang, 1 = Ausgang	Konfiguration des GPIO's entweder als Eingang (default) oder als Ausgang.
gpio1	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des GPIO, Achtung! Das "*Setzen*" dieser Eigenschaft geht nur, wenn der GPIO als "*Ausgang*" konfiguriert wurde, sonst gibt es ein *IOError.*
gpio1_direction	Eigenschaft	int	L / S	0 = Eingang, 1 = Ausgang	Konfiguration des GPIO's entweder als Eingang (default) oder als Ausgang.
gpio2	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des GPIO, Achtung! Das "*Setzen*" dieser Eigenschaft geht nur, wenn der GPIO als "*Ausgang*" konfiguriert wurde, sonst gibt es ein *IOError.*
gpio2_direction	Eigenschaft	int	L / S	0 = Eingang, 1 = Ausgang	Konfiguration des GPIO's entweder als Eingang (default) oder als Ausgang.
gpio3	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des GPIO, Achtung! Das "*Setzen*" dieser Eigenschaft geht nur, wenn der GPIO als "*Ausgang*" konfiguriert wurde, sonst gibt es ein *IOError.*
gpio3_direction	Eigenschaft	int	L / S	0 = Eingang, 1 =	Konfiguration des GPIO's entweder als

Name	Typ	Datentyp	Zugriff	Werte	Beschreibung
				Ausgang	Eingang (default) oder als Ausgang.
h0_dht11	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h0_dht22	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h1_dht11	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h1_dht22	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h2_dht11	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h2_dht22	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h3_dht11	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
h3_dht22	Eigenschaft	float	L	Rel. Luftfeuchte in %	Umgerechneter Wert vom Sensor entsprechend den Vorgaben des Datenblatts des Sensors.
hum_input0_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Diese Eigenschaft liefert nur den Rohwert vom Sensor ohne Umrechnung.
hum_input1_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Diese Eigenschaft liefert nur den Rohwert vom Sensor ohne Umrechnung.
hum_input2_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Diese Eigenschaft liefert nur den Rohwert vom Sensor ohne Umrechnung.
hum_input3_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Diese Eigenschaft liefert nur den Rohwert vom Sensor ohne Umrechnung.
pwm0	Eigenschaft	int	L / S	0 .. 65000	PWM 0 Duty Cycle.
pwm1	Eigenschaft	int	L / S	0 .. 65000	PWM 1 Duty Cycle.
pwm_ctrl_cs0	Eigenschaft	int	L / S	0 = Aus, 1 = An	PWM Clock Select Bit 0.
pwm_ctrl_cs1	Eigenschaft	int	L / S	0 = Aus, 1 = An	PWM Clock Select Bit 1.
pwm_ctrl_cs2	Eigenschaft	int	L / S	0 = Aus, 1 = An	PWM Clock Select Bit 2.
pwm_ctrl_mode	Eigenschaft	int	L / S	0 = Servo Mode, 1 =	Betriebsart der PWM Ausgänge.

Name	Typ	Datentyp	Zugriff	Werte	Beschreibung
				PWM Mode	
pwm_ctrl_od0	Eigenschaft	int	L / S	0 = Aus, 1 = An	Overdrive Einstellung für PWM 0.
pwm_ctrl_od1	Eigenschaft	int	L / S	0 = Aus, 1 = An	Overdrive Einstellung für PWM 1.
pwm_ctrl_period	Eigenschaft	int	L / S	0 .. 65000	Frequenz der PWM's.
relay0	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des Relais 0.
relay1	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des Relais 1.
relay2	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des Relais 2.
relay3	Eigenschaft	int	L / S	0 = Aus, 1 = An	Zustand des Relais 3.
serial_mode	Eigenschaft	bool	L / S	False = RS232, True = RS485	Modus der seriellen Schnittstelle auf dem PiXtend Board festlegen.
servo0	Eigenschaft	int	L / S	0 .. 250	Einheit zur Festlegung der Motorstellung vom Modellbau-Servomotor.
servo1	Eigenschaft	int	L / S	0 .. 250	Einheit zur Festlegung der Motorstellung vom Modellbau-Servomotor.
t0_dht11	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.0	Werte von einem DHT11 Sensor angeschlossen an GPIO0.
t0_dht22	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.5	Werte von einem DHT22 Sensor angeschlossen an GPIO0.
t1_dht11	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.0	Werte von einem DHT11 Sensor angeschlossen an GPIO1.
t1_dht22	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.5	Werte von einem DHT22 Sensor angeschlossen an GPIO1.
t2_dht11	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.0	Werte von einem DHT11 Sensor angeschlossen an GPIO2.
t2_dht22	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.5	Werte von einem DHT22 Sensor angeschlossen an GPIO2.
t3_dht11	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.0	Werte von einem DHT11 Sensor angeschlossen an GPIO3.
t3_dht22	Eigenschaft	float	L	Temperatur in Grad Celsius, z.B. 9.5	Werte von einem DHT22 Sensor angeschlossen an GPIO3.
temp_input0_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Direkter Rohwert vom Sensor ohne Umrechnung.
temp_input1_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Direkter Rohwert vom Sensor ohne Umrechnung.
temp_input2_raw	Eigenschaft	int	L	16 Bit Integer-Wert	Direkter Rohwert vom Sensor ohne

Name	Typ	Datentyp	Zugriff	Werte	Beschreibung
				vom Sensor	Umrechnung.
temp_input3_raw	Eigenschaft	int	L	16 Bit Integer-Wert vom Sensor	Direkter Rohwert vom Sensor ohne Umrechnung.
uc_board_version	Eigenschaft	int	L	12 = 1.2.x, 13 = 1.3.x	Hardware Version des PiXtend Boards, z.B. kann Ver. 1.2.x die digitalen Ausgänge und Relais nicht rücklesen.
uc_control	Eigenschaft	int	L	0 = Man. Mode, 16 = Auto Mode	Einstellung des Betriebsmode des Mikrocontrollers auf dem PiXtend Board.
uc_fw_version	Eigenschaft	int	L	1, 2, 3, 4, 5 entspr. Firmware Version	Firmware Version des Mikrocontrollers auf dem PiXtend Board.
uc_status	Eigenschaft	int	L	0 = Init, 1 = Run	Status des Mikrocontrollers.
use_fahrenheit	Eigenschaft	bool	L / S	False = °C, True = °F	Die Temperaturen von T0 bis T3 werden standardmäßing in °C ausgegeben, ist dieser Wert "*True*", erhält man °F.