Hochschule Neubrandenburg
University of Applied Sciences

Hochschule Neubrandenburg
Master – Geoinformatik und Geodäsie
Modul VMGG07 – Anwenderprojekte
1. Semester

Entwicklerdokumentation

1-Wire-Sensoren für Temperatur, Luftdruck und Luftfeuchte unter Linux mit Python auslesen

Autor:	Tino Schuldt Mathias Krüger
Betreuer:	M.Eng. Philipp Engel Prof. Dr.-Ing. Karl Foppe
Tag der Einreichung:	22.01.2016

Inhaltsverzeichnis

1 Einführung

1.1 Aufgabenstellung

Dieses Projekt beschäftigt sich mit dem Auslesen von unterschiedlichen 1-Wire-Sensoren. Diese 1-Wire-Sensoren sollen unter einem Linux-Betriebssystem mithilfe von 1-Wire-USB-Adaptern und der Programmiersprache Python ausgelesen werden.

1.2 Ziel

Das Ziel dieses Projektes ist das erfolgreiche Auslesen der vorgegebenen Sensoren mittels Python-Skripts. Weiterhin sollen die Messwerte für Temperatur, Luftfeuchte und Luftdruck in einer CSV-Datei gespeichert und anschließend visualisiert werden.

2 1-Wire-Sensoren

2.1 Vorwort

Die 1-Wire-Sensoren ermöglichen die digitale Erfassung von Umweltparametern (z.B. Temperatur, Luftfeuchtigkeit usw.). Zur Identifikation verfügt jeder der Sesoren über eine einmalige Seriennummer. Eingesetzt werden die Sensoren im Bereich der Mess- und Steuertechnik und dienen der Überwachung wissenschaftlicher Untersuchungen oder Automatisierungsaufgaben (vgl. [Fuc151]).

Um die 1-Wire-Sensoren mit einer RJ45-Steckverbindung an dem PC auslesen zu können, werden wie in der Abb. 2.1 dargestellt LinkUSB-Adapter bereitgestellt.

Abb. 2.1: LinkUSB-Adapter

2.2 T-Probe

Der erste Sensor ist der Temperatursensor T-Probe und ist in der Abb. 2.2 dargestellt. Dieser Sensor ist äußerst robust mit 3 Meter Kabel, einer RJ45-Steckverbindung.und basiert auf dem kostengünstigen DS18B20 Sensor. Außerdem ist der Sensor in einem widerstandsfähigen Kunststoff gegossen. Der messbare Temperaturbereich beträgt -55°C bis 60°C mit einer Abweichung von -0,5°C bis +0,5°C (vgl. [Fuc152]).

Abb. 2.2: Sensor T-Probe

2.3 OW-ENV-THPL

Der zweite Sensor ist der Temperatur-, Luftfeuchte-, Luftdruck- und Lichtsensor OW-ENV-THPL von EDS. Wie in der Abb. 2.3 dargestellt, stecken die einzelnen Sensoren alle unauffällig in einem grauen Gehäuse und hat eine RJ45-Steckverbindung. Der messbare Temperaturbereich beträgt -40°C bis 125°C mit einer Abweichung von -0,25°C bis +0,25°C. Der Messbereich für den Luftdruck beträgt 300 bis 1100 hPa/Millibar und der Messbereich für das Licht beträgt 0 bis 65535 Lux (vgl. [Fuc153]).

Abb. 2.3: Sensor OW-ENV-THPL

2.4 OW-Temp-S3-12R

Der dritte Sensor ist der Temperatursensor OW-Temp-S3-12R und ist in der Abb. 2.4 dargestellt. Der Sensor verfügt über einen RJ12-Stecker, der in diesem Projekt auf einem RJ45-Stecker gecrimpt wurde. Dieser Sensor ist äußerst robust mit 3,5 Meter Kabel und basiert auf dem kostengünstigen DS18S20 Sensor. Außerdem ist der Sensor umschlossen von einer Edelstahlhülse, mit der dieser Sensor wasserfest und resistent gegen viele Chemikalien ist. Der messbare Temperaturbereich beträgt -55°C bis 125°C mit einer Abweichung von -0,5°C bis +0,5°C (vgl. [Fuc154]).

Abb. 2.4: Sensor OW-Temp-S3-12R

3 Ansteuerung

3.1 Linux

Für die Ansteuerung der Sensoren wird ein Linux-Betriebssystem vorausgesetzt. Da keine bestimmte Linux-Distribution für dieses Projekt vorgegeben war, wurde sich für den Einsatz von Debian 8 entschieden.

3.2 USB-Schnittstelle

Nach der Installation des Betriebssystems werden die Sensoren mit dem LinkUSB-Adapter an dem PC angeschlossen.

3.3 DigiTemp

DigiTemp ist ein einfaches Programm um 1-Wire-Sensoren auszulesen und unterstützt eine Vielzahl solcher einfachen Sensoren. Zunächst wurde die Software heruntergeladen, entpackt und versucht die Sensoren mit DigiTemp auszulesen.[1]

Dazu werden die Sensoren angeschlossen und in der Kommandozeile folgender Befehl als root eingegeben, danach werden die angeschlossenen und erkannten Sensoren angezeigt.

- `./digitemp_DS9097U -w -s /dev/ttyUSB0`

```
DigiTemp v3.5.0 Copyright 1996-2007 by Brian C. Lane
GNU Public License v2.0 - http://www.digitemp.com
Turning off all DS2409 Couplers
.
Devices on the Main LAN
28DBF5BD030000BE : DS18B20 Temperature Sensor
```

Anschließend muss eine leere Konfigurationsdatei erstellt werden und in diese wird dann die Konfiguration automatisch geschrieben.

- `touch /etc/digitemp.conf`
- `./digitemp_DS9097U -i -c /etc/digitemp.conf -s /dev/ttyUSB0`

```
DigiTemp v3.5.0 Copyright 1996-2007 by Brian C. Lane
GNU Public License v2.0 - http://www.digitemp.com
Turning off all DS2409 Couplers
.
Searching the 1-Wire LAN
28DBF5BD030000BE : DS18B20 Temperature Sensor
ROM #0 : 28DBF5BD030000BE
Wrote /etc/digitemp.conf
```

[1] Download der Software digitemp-3.6.0: https://www.digitemp.com/software/linux/index.html

In die geschriebene Konfigurationsdatei können nun die Sensoren und das Log-Format eingesehen werden.

- `cat /etc/digitemp.conf`

```
READ_TIME 1000
LOG_TYPE 1
LOG_FORMAT "%b %d %H:%M:%S Sensor %s C: %.2C F: %.2F"
CNT_FORMAT "%b %d %H:%M:%S Sensor %s #%n %C"
HUM_FORMAT "%b %d %H:%M:%S Sensor %s C: %.2C F: %.2F H: %h%%"
SENSORS 1
ROM 0 0x28 0xDB 0xF5 0xBD 0x03 0x00 0x00 0xBE
```

Nach der Vorbereitung kann der Sensor T-Probe ausgelesen werden. Dies geschieht mit dem folgenden Befehl:

- `./digitemp_DS9097U -q -c /etc/digitemp.conf -a`

```
Oct 16 17:55:40 Sensor 0 C: 20.69 F: 69.24
```

Das Programm DigiTemp verfügt weiterhin über eine Vielzahl von Schaltern zum Auslesen der Sensoren. Wird der Schalter -o 2 hinzugefügt, werden nur die ID und der Temperaturwert in Grad Celsius ausgegeben. Dadurch könnten die Werte unkompliziert abgespeichert werden (vgl. [Fin15]).

Das Programm DigiTemp eignet sich gut, um unkompliziert einige Sensoren auslesen zu können. In den Versuchen wurde aber festgestellt, dass der Sensor OW-ENV-THPL mit diesem Programm nicht ausgelesen werden konnte.

3.4 OWFS

OWFS ist ein „One Wire File System“ und darauf ausgelegt mit 1-Wire-Sensoren zu kommunizieren und diese sehr einfach in das Dateisystem einzubinden.

Zunächst wird das OWFS installiert.

- `apt-get install owfs`

Die Sensoren können sehr einfach in das Dateisystem eingebunden werden. Mit dem Schalter -d werden die USB-Geräte mit den Sensoren übergeben. Da in diesem Fall zwei Sensoren an den USB-Schnittstellen angeschlossen sind, wird die Angabe von zwei Pfaden benötigt. Mit dem Schalter -m wird der Pfad übergeben, an dem die Sensoren in das Dateisystem eingebunden werden.

- `mkdir /mnt/1wire`
- `owfs -d /dev/ttyUSB0 /dev/ttyUSB1 -m /mnt/1wire`

Die Sensoren werden in dem übergebenen Pfad angezeigt. Die Darstellung erfolgt mit einer hexadezimalen ID. Des Weiteren existieren weitere Dateien. In der Standardkonfiguation befinden sich zwei sogenannte „Fake-Sensoren“. Diese beiden Fake-Sensoren `05.4AEC29CDBAAB` und `10.67C6697351FF` können in der Konfigurationsdatei ausgeblendet bzw. einfach ignoriert werden, da diese lediglich zum Testen ohne Sensoren geeignet sind.

- `ls /mnt/1wire`

```
05.4AEC29CDBAAB     7E.782600001000     simultaneous
10.67C6697351FF     alarm               statistics
10.DCA98C020800     bus.0               system
28.DBF5BD030000     settings            uncached
```

Um nicht mit der hexadezimalen ID arbeiten zu müssen, kann die ID einen Namen zum Zugriff erhalten. Dazu wird eine Text-Datei wie z. B. „alias.txt“ benötigt. In dieser Textdatei kann pro Zeile eine ID einen Namen bekommen (bspw. ID=NAME). Anschließend muss diese Textdatei mit dem Parameter -a übergeben werden.

- `owfs -a alias.txt -d /dev/ttyUSB0 /dev/ttyUSB1 -m /mnt/1wire`

```
28.DBF5BD030000=t-probe
7E.782600001000=ow-env-thpl
10.DCA98C020800=ow-temp-s3-12r
```

Das OWFS eignet sich sehr gut für das weitere Vorgehen, da dieses sehr einfach einsetzbar ist und für viele Sensoren geeignet ist. Außerdem können verschiedene Programmiersprachen (besonders mit Python) auf diese Schnittstelle zugreifen und die Daten der Sensoren abfragen.

3.5 Fazit

In der Tabelle 3.1 werden beide Möglichkeiten zur Ansteuerung miteinander verglichen. Da es möglich ist, mehr Sensoren mit dem OWFS auszulesen als mit DigiTemp, fällt die Wahl für die Ansteuerung auf das OWFS. Da zum Auslesen auf Python gesetzt wird, ist auch hier das OWFS mit dem direkten Zugriff auf das Filesystem im Vorteil. Für das Projekt und die späteren Python-Skripte wird daher auf das OWFS gesetzt.

	Vorteile	Nachteile
Digi-Temp	+ Kompilierte Datei herunterladen und sofort die Daten auslesen	– Nicht alle 1-Wire-Sensoren werden unterstützt – Auslesen der Werte mit der kompilierten Datei
OWFS	+ Mehr Funktionalitäten als DigiTemp + So gut wie alle 1-Wire-Sensoren können ausgelesen werden + Zugriff über das Filesystem	– OWFS muss für das Betriebssystem zur Verfügung stehen

Tabelle 3.1: Vergleich der Möglichkeiten zur Ansteuerung der 1-Wire-Sensoren

4 Ablauf

4.1 Ordnerstruktur

Für das Projekt wird eine Ordnerstruktur angelegt. Für die Konfiguration vor der Installation wird im Ordner „config“ die Datei „config.sh“ genutzt. Um alle benötigten Pakete zu installieren, wird die Datei „install.sh“ verwendet. Die Datei „mount.sh“ stellt den einfachen Zugriff mit den Sensoren über das Dateisystem her. Im Ordner „logfiles“ werden alle Schritte beim Ausführen der Skripte protokolliert.

- projekt
  - 1wire
  - config
    - config.sh
    - config_python.py
  - data
  - logfiles
  - scripts
    - install.sh
    - mount.sh
    - observable.py
    - plot_data.py
    - read_all.py
    - save_plot.py
    - synchronized_all.py
  - install.py
  - read.py
  - README

4.2 Konfiguration

Zuerst erfolgt der Anschluss der Sensoren an dem PC mit den LinkUSB-Adaptern. Je nach Einsatz der Sensoren wird die „config.sh“-Datei angepasst. In dieser Konfigurationsdatei können den Sensoren einen Namen zugeordnet und der Mountpfad verändert werden.

4.3 Installation

Alle Skripte benötigen Ausführungsrechte, bevor die Installation anläuft. Dazu muss in der Kommandozeile folgender Befehl im Projektverzeichnis ausgeführt werden:

- `chmod +x . -R`

Anschließend wird die Installation durch das Aufrufen des Skripts „install.py“ mit Root-Rechten ausgeführt. Dieses Skript wird zunächst Python, OWFS und anschließend Matplotlib installieren.

- `sudo ./install.py`

```
LOG: 17:18:34 Package <python> is installed. Python Version: 2.7.9
LOG: 17:18:34 Package <owfs> is installed.
LOG: 17:18:34 Package <python-matplotlib> is installed.
```

4.4 OWFS-Mountpfad

Vor dem Auslesen der Sensoren muss das OWFS die Sensoren in das Dateisystem einbinden. Dies geschieht automatisch beim Aufruf der „read.py“ mit Root-Rechten. Im Terminal werden alle Schritte protokolliert und die angeschlossenen Sensoren aufgelistet:

```
LOG: 17:18:34 Write Sensorlist in File <config/sensoren_name.txt>.
LOG: 17:18:34 Mount paths in </home/student/projekt/1wire>.
LOG: 17:18:34 Found Sensor in USB-Device <ttyUSB0>.
LOG: 17:18:34 Found Sensor in USB-Device <ttyUSB1>.
LOG: 17:18:34 Found Sensor in USB-Device <ttyUSB2>.
LOG: 17:18:34 Mount USB-Devices </dev/ttyUSB0 /dev/ttyUSB1
/dev/ttyUSB2 >.
LOG: 17:18:34 Sensor available <ow-temp-s3-12r>.
LOG: 17:18:34 Sensor available <t-probe>.
LOG: 17:18:34 Sensor available <ow-env-thpl>.
```

4.5 Sensoren auslesen

Das Auslesen der Sensoren erfolgt über die „synchronized_all.py“. Diese wird beim Aufrufen der „read.py“ mit gestartet, sodass die „config_python.py“ vorher angepasst werden muss. In dieser befindet sich ein Python-Dictionary „configuration“ und eine String-Variable „plotting“. Der erste Eintrag im Dictionary ist eine Liste der angeschlossenen Sensoren. Hier müssen die Namen verwendet werden, die den Sensoren in der „config.sh“ gegeben wurden. Der zweite Eintrag ist der Mountpfad, an dem die Sensoren eingebunden sind. Standardmäßig zeigt der Eintrag auf den Mountpfad in unserer Ordnerstruktur. Der dritte Eintrag ist der Speicherort, an dem die CSV-Daten gespeichert werden. Auch hier ist der Standardeintrag der dafür vorgesehene Ordner in unserer Ordnerstruktur. Der vierte Eintrag gibt die Anzahl der Messungen an, die vorgenommen werden soll. Dieser Eintrag muss ausgefüllt sein, da dieser die Abbruchbedingung für das Auslesen darstellt. Die String-Variable „plotting“ wird für das Plotten der CSV-Daten im Postprocessing benötigt. Hier muss der Pfad und Name der Datei angegeben werden, die mittels der „plot_data.py“ geplottet werden soll.

Wurde die Konfiguration entsprechend angepasst, kann die „read.py“ wie oben beschrieben ausgeführt werden. Diese ruft die „synchronized_all.py“ auf und startet damit das Auslesen der Sensoren. Die „synchronized_all.py“ liest als Erstes die Konfiguration ein. Danach legt diese für die dort eingetragenen Sensoren Objekte an. Die Klassen dieser Objekte befinden sich in der „read_all.py“. Dort gibt es für jeden Sensor eine Klasse zum Auslesen der Daten.

Als Nächstes wird für jeden Sensor ein Objekt der Speicherklasse angelegt. Diese nennt sich „csv_dumper“ und kommt aus der „save_plot.py“. Diese Objekte werden an die Sensor-Objekte mit Hilfe eines Observer-Pattern[2] gekoppelt. Daraufhin wird für jeden Sensor ein Thread angelegt, der das dazugehörige Objekt erhält und startet.

Nach dem Start liest das Skript ungefähr alle 16 Sekunden einen Wert aus und übergibt diesen an das Objekt „csv_dumper“. Dieses Objekt schreibt wie in der Abb. 4.1 dargestellt

[2] http://www.oodesign.com/observer-pattern.html

den Messwert bzw. die Messwerte in eine CSV-Datei. Die Dateinamen folgen dabei dem Muster „Sensorname + heutiges Datum“. Es wird jeden Tag für jeden Sensor eine neue CSV-Datei angelegt. Nach der festgelegten Anzahl an Messungen in der „config_python.py“ werden die Threads und das Programm beendet.

2016-01-20 14:08:06.757989	20.3125	42.0625	7.0	1003.58	0.0
2016-01-20 14:08:22.216052	20375	42125	7.0625	1003.58	10.0
2016-01-20 14:08:37.916902	20375	43.0	7375	1003.58	68.0
2016-01-20 14:08:53.621212	20375	42375	7125	1003.59	70.0
2016-01-20 14:09:09.199000	20375	42125	7.0625	1003.6	74.0
2016-01-20 14:09:24.860703	20.4375	41.9375	7.0	1003.59	74.0
2016-01-20 14:09:40.322998	20375	41.9375	7.0	1003.59	78.0
2016-01-20 14:09:56.006928	20.4375	41.9375	7.0	1003.59	66.0
2016-01-20 14:10:11.651287	20.4375	41.8125	7.0	1003.6	54.0
2016-01-20 14:10:27.292240	20.4375	41625	6.9375	1003.6	53.0
2016-01-20 14:10:42.822529	20.4375	41625	6.9375	1003.57	51.0
2016-01-20 14:10:58.493734	20.5	41625	7.0	1003.57	46.0
2016-01-20 14:11:14.248774	20.4375	41625	6.9375	1003.57	47.0
2016-01-20 14:11:30.025857	20.5	41.5	6.9375	1003.57	48.0
2016-01-20 14:11:45.649783	20.4375	41.5	6875	1003.57	48.0
2016-01-20 14:12:01.353806	20.5	41.1875	6.8125	1003.56	48.0
2016-01-20 14:12:17.015353	20.5	41.5	6875	1003.56	46.0
2016-01-20 14:12:32.697155	20.5	41.5	6.9375	1003.56	46.0
2016-01-20 14:12:48.250261	20.5	41375	6875	1003.58	45.0
2016-01-20 14:13:03.934852	20.5	41625	7.0	1003.59	45.0
2016-01-20 14:13:19.756693	20.5625	41375	6.9375	1003.58	45.0
2016-01-20 14:13:35.384215	20.5	41375	6875	1003.57	45.0
2016-01-20 14:13:51.039135	20.5625	41375	6.9375	1003.57	45.0
2016-01-20 14:14:06.580025	20.5625	41.5	7.0	1003.57	44.0
2016-01-20 14:14:22.213041	20.5	41.5	6.9375	1003.57	43.0
2016-01-20 14:14:37.873495	20.5625	41375	6.9375	1003.56	43.0
2016-01-20 14:14:53.556136	20.5625	41.5	7.0	1003.56	43.0
2016-01-20 14:15:09.196134	20.5625	41375	6.9375	1003.57	43.0
2016-01-20 14:15:24.928138	20.5	41625	7.0	1003.57	43.0
2016-01-20 14:15:40.587505	20.5625	41375	6.9375	1003.58	44.0

Abb. 4.1: CSV-Datei der Messung mit dem Sensor OW-ENV-THPL

4.6 Postprocessing

Für das Postprocessing gibt es das Script „plot_data.py“. Dieses muss eigenständig aufgerufen werden und dient zum Plotten der ausgelesenen Daten. Dazu liest es den Dateinamen aus der Konfiguration ein und öffnet das entsprechende File. Danach liest es die Werte aus und plottet sie wie in der Abb. 4.2 und Abb. 4.3 dargestellt.

Bei Sensoren, die nur einen Wert liefern (z.B. T-Probe), wird nur ein Plot erstellt.

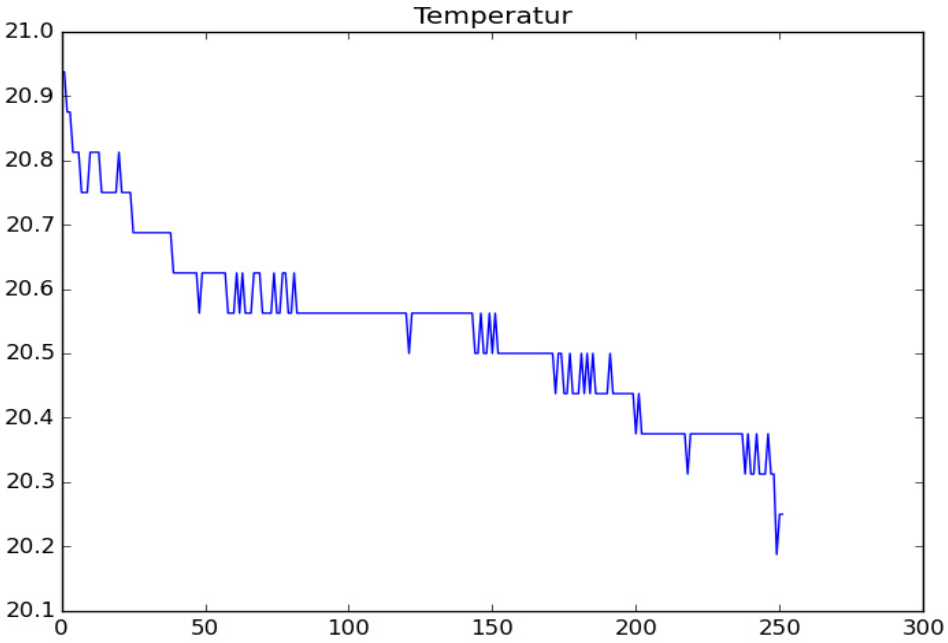

Abb. 4.2: Temperatur des Sensors T-Probe plotten

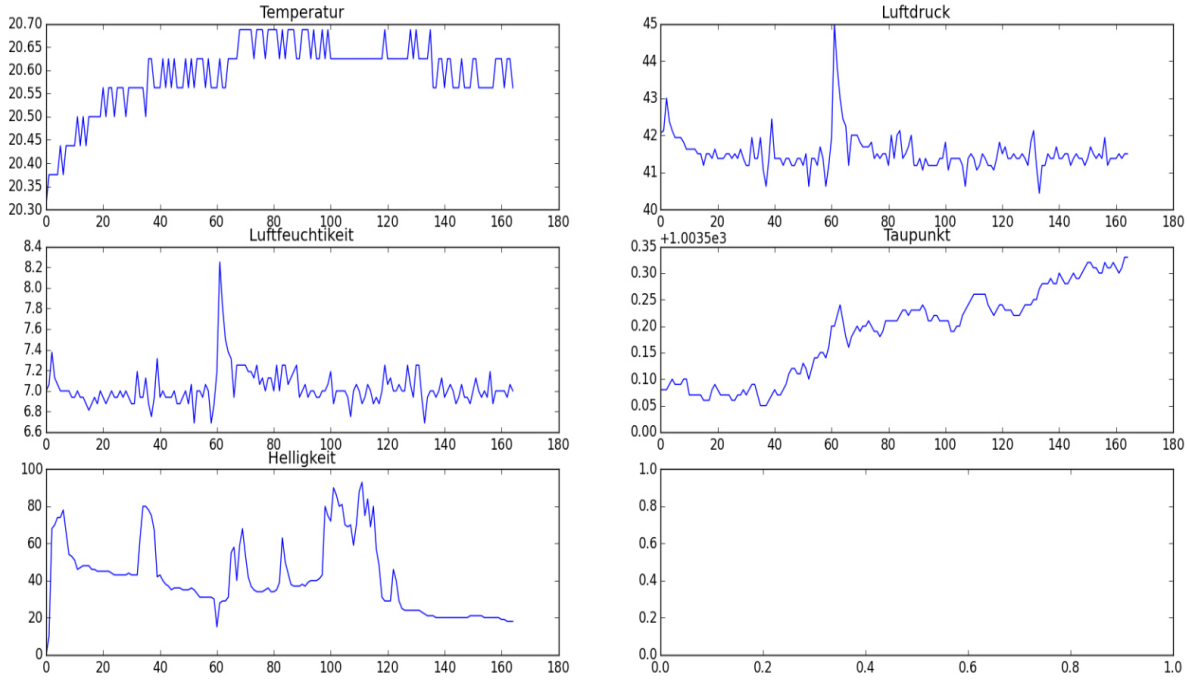

Abb. 4.3: Mehrere Subplots bei mehreren Messarten des Sensors OW-ENV-THPL

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Für das Projekt wurden 3 Sensoren und 2 USB-Adapter zur Verfügung gestellt. Diese sollen in einem Linux-Betriebssystem mithilfe der Programmiersprache Python ausgelesen werden. Zum Auslesen der Sensoren wird das Paket OWFS (One Wire File System) genutzt, welches im Betriebssystem zunächst installiert wird. Anschließend bindet das Skript mittels OWFS alle verfügbaren USB-Geräte in das System ein.

Das Auslesen der Sensoren erfolgt über Python-Skripte. Zuerst erfolgt die einmalige Grundinstallation aller benötigen Komponenten durch das Skript „install.py“. Anschließend kann das Auslesen aller angeschlossenen Sensoren durch das Skript „read.py“ erfolgen. Die Messdaten speichert das Skript als CSV-Datei, damit das Skript „plot_data.py“ diese im Postprocessing anzeigen kann.

5.2 Ausblick und weiterführende Arbeiten

Der Prototyp kann genutzt werden, um schnell und einfach die Sensoren unter einem Linux-Betriebssystem auszulesen. In weiterführenden Arbeiten können den bisher genutzten Sensoren, weitere folgen. Für jeden neuen Sensor erfolgt ein Eintrag der ID und des Namens in der „config.sh“. Als Letztes muss eine Anpassung der Skripte „read_all.py“, „synchronized_all.py“ und „plot_data.py“ erfolgen.

Eine weiterführende Arbeit könnte sich mit den Rechten beschäftigen. Für das Einbinden der Sensoren mithilfe von OWFS benötigen die Skripte immer Root-Rechte, ohne diese kann das System die angeschlossenen USB-Geräte nicht erkennen.

Sinnvoll wäre in weiterführenden Arbeiten die Unterstützung von dynamischen Plots, so dass der Plot bei neuen Messdaten automatisch aktualisiert wird.

Quellenverzeichnis

[Fin15] http://www.finnie.org/2010/03/07/external-temperature-monitoring-with-linux/ (04.11.2015).
[Fuc151] http://www.fuchs-shop.com/de/faq/ (16.11.2015).
[Fuc152] http://www.fuchs-shop.com/de/shop/6/1/13372141/ (16.11.2015).
[Fuc153] http://www.fuchs-shop.com/de/shop/6/1/13372331/ (16.11.2015).
[Fuc154] http://www.fuchs-shop.com/de/shop/6/1/13372460/ (16.11.2015).

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Anhang A – Projektdateien

- Projektdateien zum Auslesen der Sensoren