Calibration of FOTEMP Devices

OEM

Weidmann Technologies Deutschland GmbH

V2 2019-11-18

Tino Schurzmann

Content

1 Calibration

1.1 General

Before a device leaves the rooms of Weidmann, a customer depending calibration is performed. This calibration results in a Calibration Table, unique for each device + sensor combination.

Depending on the calibration range several block calibrators are used.
We are equiped with:

- Dry-Block-Calibrator Ametek RTC-159-C
- Dry-Block-Calibrator Ametek ATC-250B
- Dry-Block-Calibrator Ametek RTC-700C
- PT100 measurement device Ametek DTI050
- heating source Omega 760
- calibration control software “Kalibrator2”

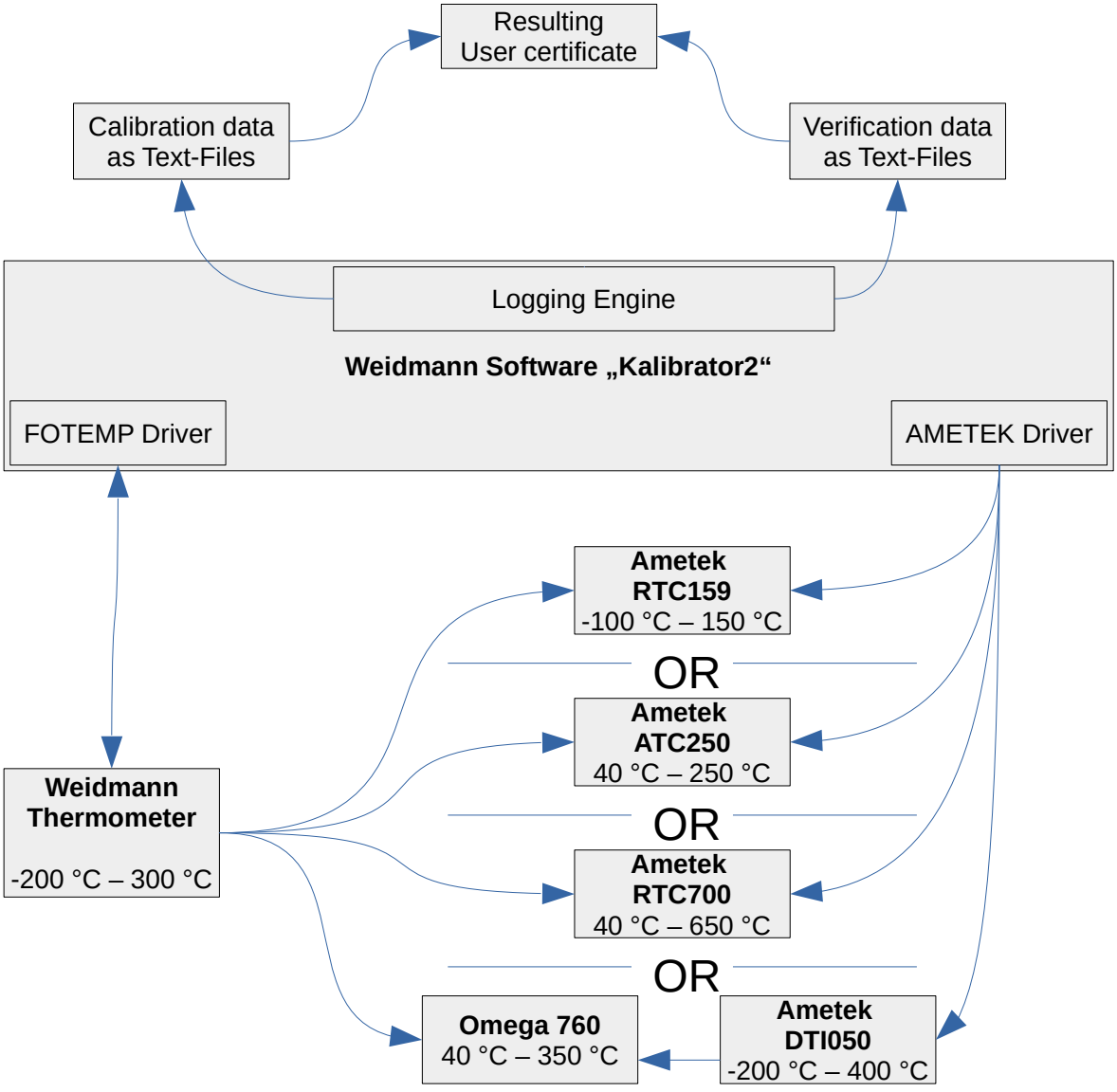

Figure 3 overview of calibration system

Figure 3 shows the calibration system. Liquid nitrogen is not shown there. This is one point where the sensor is let it and the device's response is included in the calibration table by hand.
The software "Kalibrator2" expecting a WEIDMANN thermometer and a calibrator connected. Now it is set up to step through a list of calibration temperatures (maximum is 16), given by the operator and before that from a customer.
The software executes at least two tasks: "calibration" and "adjustment" of a WEIDMANN thermometer. During the "calibration" step the spectrometer's response at the 16 calibration points is found and stored in the device. After running through all points, the software restarts at the former first step and measures the 16 temperatures again to see the device's differences. It stores them and the calibration table in text files for backup reasons and the operator to check and calculate the standard deviation of the device and to create the device's certificate.
This certificate is created by reading the software's text files (unchanged), checking the accuracy and generating the certificate (PDF) automatically.

An example of the calibration table is shown in Figure 4. It can be read and written even after the device left Weidmann. For not dealing with floating numbers the temperature is multiplied by ten. The EDGE value is one special value out of the gallium arsenide reflection spectrum, showing the temperature depending moved reflection spectrum. This value corresponds to the spectrometers output and the optical setup. Due to the manufacturing inaccuracies there is no way to neglect calibration at desired accuracy.

Temperature x 10	EDGE
1500	15801
1400	16115
1300	16417
...	...
-300	20307
-400	20509
-500	20701

Figure 4 Example of FOTEMP Calibration Table

1.2 Preinstallation of the PC

For up. And downloading calibration data the user can read the Device' interface documentetion
http://www.optocon.de/fileadmin/media/fotemp/FotempCommunicationCompact_OEM_Rev_14.pdf from www.optocon.de or follow the steps in the next chapters of this document.

The later used scripts are Python scripts. Therefore the programming language interpreter "Python" has to be installed in its version 2.7.10 or higher. Additional a communication via a COM port (RS232 or USB) is needed, so the package "pyserial-2.7" must be installed too. This document shows the usage on a Win10 PC. Linux works in the same way.

All scripts need a COM port, we are starting here.

1. Click with your mouse on the lower left desktop corner and choose "Device manager" and find the COM port under "Connections"
2. download "python-2.7.16" from here for your system: https://www.python.org/downloads/release/python-2716/, Linux has Python preinstalled
3. download "pyserial-2.7.win32" from here for your system: https://sourceforge.net/projects/pyserial/files/pyserial/2.7/, Linux: enter "pip install pyserial" in bash
4. install Python (from step 1)
5. install "PySerial" from step 2
6. reboot your Windows PC
7. copy the script files (find them on www.optocon.de)
8. only for Linux: change the lines "*ser = serial.Serial("COM"+str(comport), COMPORT_SPEED, timeout=COMPORT_TIMEOUT)*" to "*ser = serial.Serial("<your_registered_device_path>"+str(comport), COMPORT_SPEED, timeout=COMPORT_TIMEOUT)*" in all scripts
9. open a command line (Win+R, enter "cmd", press Enter key, Ctrl+Alt+T on Linux)
10. browse to the script folder

You are now ready to execute the scripts.

1.3 Download and Upload of Calibration Data

The target of this section is to describe the process of downloading a calibration file from the device to a PC. Given you have a running Python inclusive PySerial and you have the scripts

A 4-channel OEM-PLUS device was set up and is running. The device was connected to a Win10 PC. “Device Manager” shows a connected device “USB Serial Port (COM8)”, so our COM port is “8”.

Starting a command line on Win10 means pressing WindowsKey+R together, entering “cmd” and pressing enter key. Changing the partition and change the directory to the downloaded scripts is shown in Figure 1.

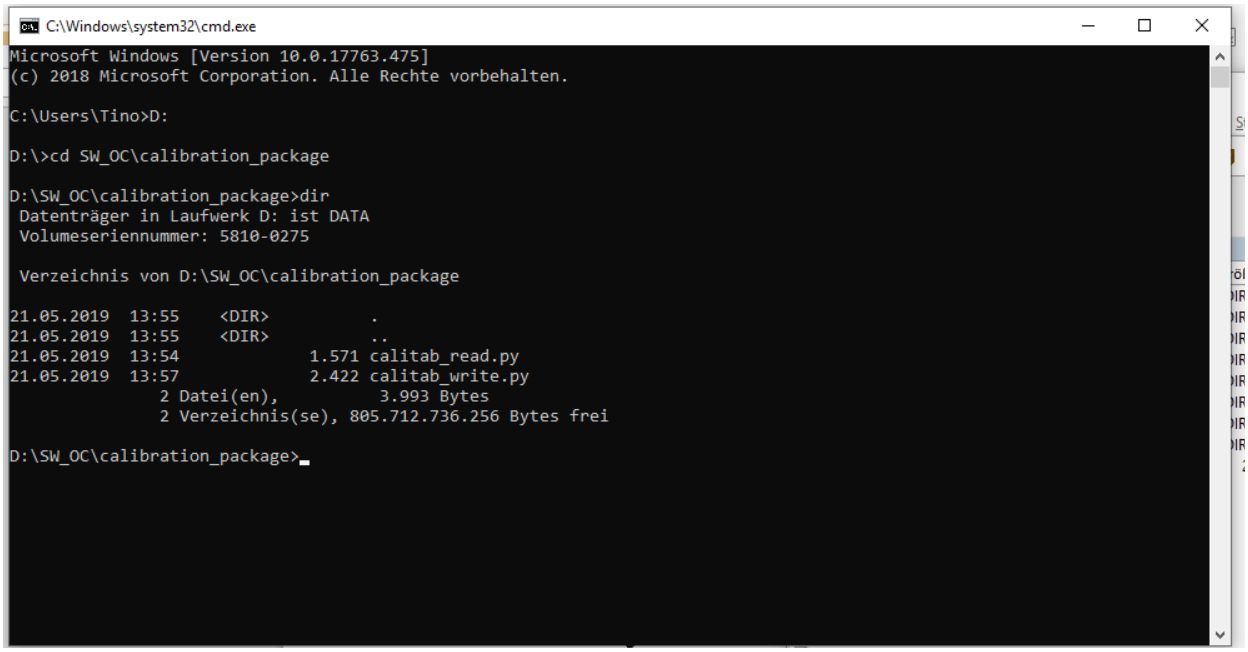

Figure 1 listing the script files

The reading script downloads or reads calibration data from a FOTEMP device (not ModuleSystem). It needs to know the COM port number and the channel to be read.
After reading one channel it creates a text file with the device’s calibration data in one file per channel for not mixing anything. The device’s data are written in the text files “data_read_ch<1/2/3/4>.cal”. Compare Figure 2 for reading channel “3”.

The writing script writes calibration data from a text file to the FOTEMP device (not ModuleSystem). It needs to know the COM port number and the channel to write to. Always read the device’s data before overwriting anything. When the COM port number and the channel is known, the script expects the data to be written in a file with a special file name as a safety action. For channel 1 the data must be in a text file called “data_write_ch1.cal”, for channel 3 the data must be in “data_write_ch3.cal”.
Use your favorite text editor to change these files for your needs.

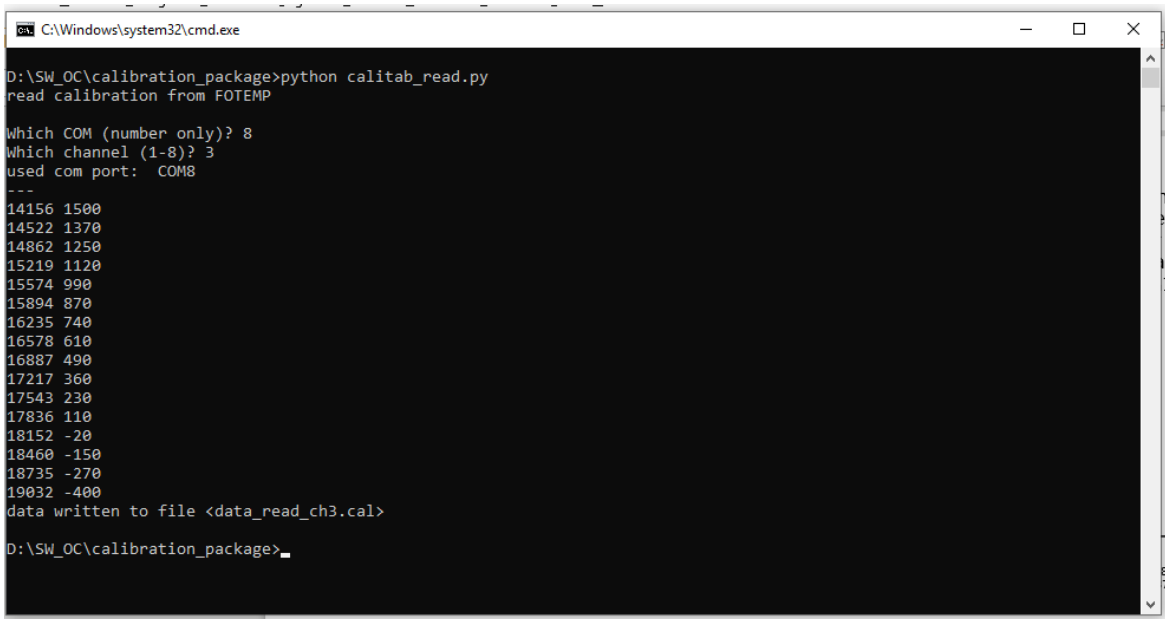

Figure 2 reading channel “3” of the device at COM8

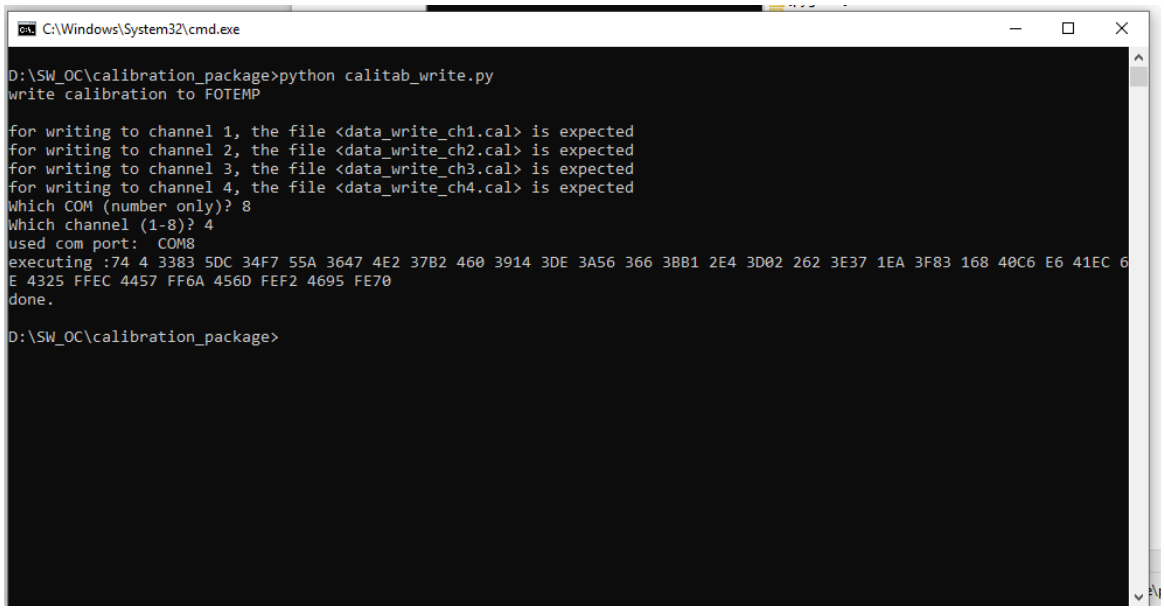

Figure 3 writing “data_write_ch4.cal” to the device at COM8

2 Measure EDGE values

At this point the question could arise, where to get the EDGE values.
For an own calibration on site the EDGE value can be measured using the following script “ftoem_read_edge.py”. This script shall be donwloadable as “Read Calibration Value Scripts” from www.optocon.de.

```
#!/usr/bin/env python
# --------------------------------------------------------------------------------
# fw setup
# version 1
# used python 2.7.10 and pyserial2.7
# --------------------------------------------------------------------------------
# TEST SETTINGS ...
COMPORT = 'COM12'
COMPORT_TIMEOUT = 1                     # in seconds
COMPORT_SPEED = 57600                   # in baud
# --------------------------------------------------------------------------------
# should print "2019-11-04_12:02:55: Ch1: 17491, Ch2: 16496, Ch3: 17509, Ch4: 16558"
# --------------------------------------------------------------------------------

import time
import serial
from time import gmtime, strftime

def get_last_edge(ser):
        rc=""
        ser.write("?73\r\n")
        rc =ser.readline()
        rc+=ser.readline()
        rc1=rc.split('\n')
        rc2=rc1[0].split(' ')
        rc3=rc2[1].replace('\r','')
        return rc3

def enable_all_channels(ser):
        rc=""
        try:
                s=":10 FF\r\n"
                ser.write(s)
                rc =ser.readline()
                rc+=ser.readline()
                return "*activated channel 0x"+ch+": "+rc
        except:
                return "no device. ",rc

def disable_all_channels(ser):
        rc=""
        try:
                s=":10 0\r\n"
                ser.write(s)
                rc =ser.readline()
                rc+=ser.readline()
                return "*activated channel 0x"+ch+": "+rc
        except:
                return "no device. ",rc

def enable_channel(ser,channel):
        rc=""
        try:
                c=1<<(channel-1)
                ch="{:02x}".format(c)
                s=":10 "+str(ch)+"\r\n"
                ser.write(s)
                rc =ser.readline()
                rc+=ser.readline()
                return "*activated channel 0x"+ch+": "+rc
        except:
                return "no device. ",rc
```

```
if __name__ == "__main__":
        ser = serial.Serial(COMPORT, COMPORT_SPEED, timeout=COMPORT_TIMEOUT)
        print "*used com port: ",ser.name
        while 1:
                enable_channel(ser,1)
                e1=int(get_last_edge(ser),16)
                enable_channel(ser,2)
                e2=int(get_last_edge(ser),16)
                enable_channel(ser,3)
                e3=int(get_last_edge(ser),16)
                enable_channel(ser,4)
                e4=int(get_last_edge(ser),16)
                mytime=strftime("%Y-%m-%d_%H:%M:%S", gmtime())
                print(mytime+": Ch1: "+str(e1)+", Ch2: "+str(e2)+", Ch3: "+str(e3)+", Ch4: "+str(e4))
                time.sleep(0.25)
        enable_all_channels(ser)
        ser.close()
```

Figure 4 script to measure EDGE value

So the general calibration setup should be like this:

1. performe EDGE measurement with the script from Figure 4 for all temperatures the user is interested in
2. write the EDGE values in a text file as an input for the calibration scripts
3. execute the calibration scripts from chapter 1.3
4. restart the physical device

- EoD -