Norwegian Meteorological Institute

METreport Technical

Low Cost Temperature Sensor Calibration

Calibration Box User Manual

Adam Kristiansen

[Classification: open]

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

Title Low Cost Temperature Sensor Calibration	**Date** 15.08.2024
Section Obsklim OKD	**Report no.** No. [x]/[xxxx]
Author(s) Adam Kristiansen	**Classification** ● Free ○ Restricted
Client(s)	**Client's reference**

Abstract

To calibrate low cost temperature sensors. I had to design a box that could find the calibration values of several DS18B20 temperature sensors fast. It has to read several sensors at once, and save the data as a readable file. It also had to work with the reference temperatures from the climate control cabinet at the meteorological institute of Oslo.

Keywords
Low cost temperature sensors, Calibration, Open source

Disiplinary signature

Responsible signature

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

Abstract

To calibrate low cost temperature sensors. I had to design a box that could find the calibration values of several DS18B20 temperature sensors fast. It has to read several sensors at once, and save the data as a readable file. It also had to work with the reference temperatures from the climate control cabinet at the meteorological institute of Oslo.

Table of contents

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

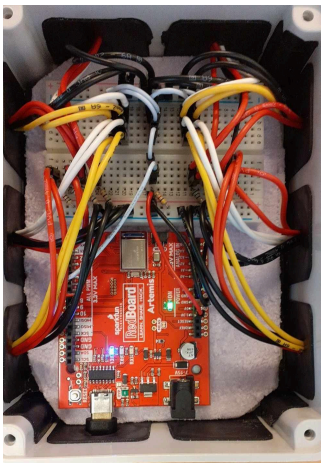
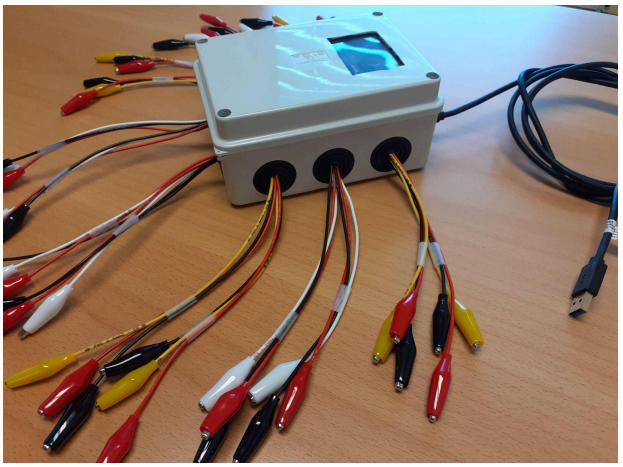
Low cost temperature sensor calibration

1.1. Properties

The DS18B20 calibration box is a system designed to read data from DS18B20 temperature sensors, and compare the values to a CSV file created by the milliK high precision thermometer with exact reference temperatures. To see how to use a milliK thermometer see https://docs.google.com/document/d/1RymtBrh2nqusMveZacmvP_HJ_Yaa6uwUz4di5o6gRRs/edit?usp=sharing

The system:

- Can calibrate up to 16 sensors at the same time
- Has crocodile clips for easy connection to sensors
- Has USB communication between calibration box and computer
- Has optional runtime and measurement frequency
- Saves all data in CSV format
- Has easy plotting of all data

1.2. Design

The DS18B20 has 3 main parts. The hardware, an arduino script and a python script for reading data from the sensors, and a python script for comparing the data to the data from the milliK high precision thermometer.

1.2.1. Hardware

The hardware box is a 105 x 148 x 74 mm box with 48 different colored crocodile clips and a 2 meter long USB cable. Inside there is an arduino artemis redboard that is responsible for reading and sorting the different data values from the sensors and sending them over the USB connection. The hardware box has a plastic window where you can look at two led lights on the arduino. The blue led will light up whenever the arduino is active and not waiting for the next measurement. This can be used to see if the program has stopped at an unexpected place in the code. The green led is an indication of communication over the USB cable. It lights up whenever the program sends data over the USB cable.

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

1.2.2. DS18B20_reading

There are two scripts dedicated to reading and saving the data from the sensors into a CSV file, and one script to calculate the calibration values.

DS18B20_reading.ino

This script is already downloaded to the arduino artemis redboard, and is not needed unless it somehow is deleted from the arduino, or if you want to change the code. If this happens you only need to download it again using the Arduino IDE software.

DS18B20_reading.py

This is a python script for sorting all the data from the arduino artemis redboard into a readable CSV file with timestamps for each measurement. It also reads the wanted runtime and measurement frequency from the user.

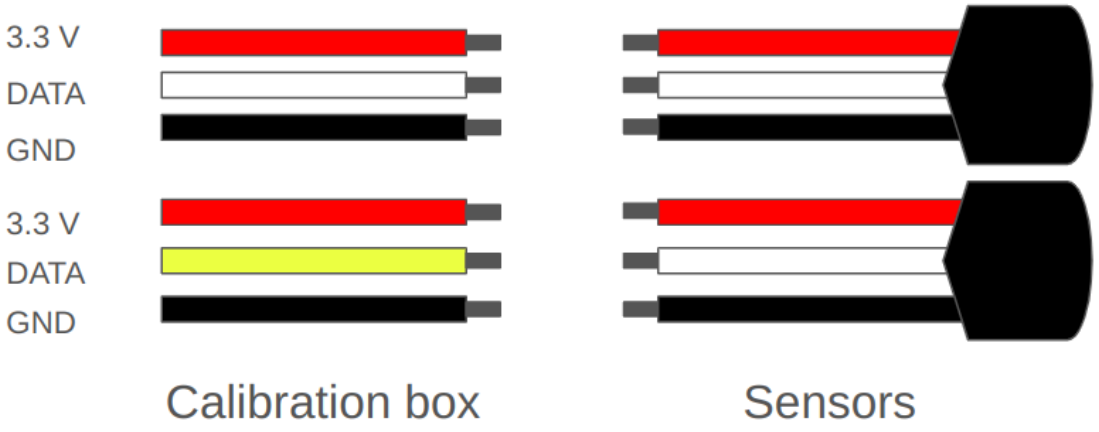
Calibration.py

The calibration script is as you can probably guess, the script used for finding calibration values for each sensor. It reads the file directories from the user for the datafile that is created in the DS18B20_reading program, and the CSV file from the milliK thermometer. It compares the datasets and calculates the average offset between sensor values and reference temperatures.

1.3. Using the system

1.3.1. Connecting hardware

The hardware is relatively simple to connect. The only connection points are the USB cable and the crocodile clips. The USB cable connects to a computer with the necessary software programs. The crocodile clips have 3 different colors. Red, Black and white/yellow. The red clip is the 3.3V source. The black clip is ground and the white and yellow clip are the same, and is used for data communication. The sensors are designed with a black, red and white cable, so the connection should be intuitive.



For more information on the sensors see datasheet for DS18B20 sensors.
https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

1.3.2. Running the reading program

To read the data files with the data from the sensors, you need to run the DS18B20_reading.py python script. This can be run in the terminal window.

The program needs a couple of specifications from the user before it can run. These specifications can be given in one of the two following ways.

If you run the program without anything in the command line it will ask for the necessary information in the terminal window.

```
adkri0496@pc5997:~$ ipython ~/DS18B20_reading.py
Enter number of measurements per minute: (x)
Enter runtime for the program in minutes: (y)
Select port: (port)
```

You can also give the information in the command line. The order is the same as if you let the program ask for it.

```
adkri0496@pc5997:~$ ipython ~/DS18B20_reading.py x y port
```

In both cases the X represents the number of measurements per minute. If you for example wanted one measurement every tenth second you would have to write 6 for the first value. Note that the sensores use one second to download the temperatures to the scratchpad, so if you write more than 60 measurements per second, the program still uses the value 60.

The Y represents the total time you want the program to run for in minutes. This has no maximum value, but it has to be in full minutes.

The port value represents the name of the USB port. You have to find this on your computer beforehand, because it is specific for your computer. It can for example be ‘USB0’ or ‘/dev/ttyUSB0’. This name has to be spelled correctly, or else the program will not connect correctly to the arduino artemis redboard.

After these values have been given the program will print out the values on the terminal and save them as a CSV file. You can find the file either in your home folder or in the same directory as the python script. You can also end the program at any time safely by pressing ‘Ctrl + c’ in the terminal. The file will then be saved as it is

1.3.3. Running the calibration program

To find the calibration values you have to run the Calibration.py script. This can be run in the terminal window.

Like the former program, this program also needs some initial information. In this script it can also be given in two ways.

You can give the information in the command line like the picture underneath.

```
adkri0496@pc5997:~$ ipython ~/Calibration.py x y
```

You can also wait for the program to ask you for the information.

```
adkri0496@pc5997:~$ ipython ~/Calibration.py
Enter full file directory: (x)
Enter file directory for referance file: (y)
```


Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

In both cases the x represents the full file directory for the datafile from the sensors you have already measured, this can for example be ~/Desktop/Datafile_2024-08-07T13:14:31.csv.

The y represents the full file directory of the logfile from the milliK temperature thermometer. This can for example be if you use the default filename on the milliK thermometer, ~/Desktop/logfile_20240807_1318.csv.

These directories must be correctly written to get the wanted readings.

After the program has been run with the correct files, the program calculates the offset values for every measurement you have made, and an average offset value for each sensor. The values are printed out on the terminal, and as a CSV file that will be explained later in this document. The program will also plot all the plots that are added to the script. The program should end after only a few seconds.

1.4. Files

1.4.1. Datafile

The datafile is the CSV file that is created in the DS18B20_reading programs. I call it the Datafile because it automatically gets the name Datafile followed by the time it was created down to seconds.

The file has 5 values per measurement, and one line is dedicated to one measurement of one specific sensor. So if you for example wanted to measure 2 sensors 5 times you would get 10 lines.

Time:

The time column represents the specific UTC time the measurement was done. It is written in iso format.

Sensor ID:

The sensor ID column contains the 8 byte address of the specific sensor written in hexadecimal format. This is the number used for sorting the temperatures later.

ID 6 bit:

The ID 6 bit column represents a part of the sensor ID. It is the 6 least significant bits of the second byte in the sensor ID. This value is not written in hexadecimal format, but rather normal numbers. Meaning the number 56 represents 111000. This value is only for a reader to easily find a specific sensor without using the full 8 byte ID.

Sensor data:

This is all the 9 byte data read from the sensor written in hexadecimal format. This data is not used later in the program, and will most likely not be required at all. I only added it in case someone wanted it.

Celsius:

The Celsius column is the actual temperature in celsius. This is the data that is actually used to calibrate the sensors.

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

	Time	Sensor ID	ID 6 bit	Sensor data	Celsius
0	2024-08-12T11:54:22.618619+00:00	28 08 42 8D 0C 00 00 2A	8	40 01 4B 46 7F FF 10 10 1D	20.0000
1	2024-08-12T11:54:22.618619+00:00	28 78 12 18 0D 00 00 EC	56	3F 01 4B 46 7F FF 01 10 7A	19.9375
2	2024-08-12T11:54:25.947965+00:00	28 08 42 8D 0C 00 00 2A	8	3D 01 4B 46 7F FF 03 10 6D	19.8125
3	2024-08-12T11:54:25.947965+00:00	28 78 12 18 0D 00 00 EC	56	3D 01 4B 46 7F FF 03 10 6D	19.8125
4	2024-08-12T11:54:30.959291+00:00	28 08 42 8D 0C 00 00 2A	8	3D 01 4B 46 7F FF 03 10 6D	19.8125
5	2024-08-12T11:54:30.959291+00:00	28 78 12 18 0D 00 00 EC	56	3D 01 4B 46 7F FF 03 10 6D	19.8125
6	2024-08-12T11:54:35.970673+00:00	28 08 42 8D 0C 00 00 2A	8	3D 01 4B 46 7F FF 03 10 6D	19.8125
7	2024-08-12T11:54:35.970673+00:00	28 78 12 18 0D 00 00 EC	56	3D 01 4B 46 7F FF 03 10 6D	19.8125
8	2024-08-12T11:54:40.981998+00:00	28 08 42 8D 0C 00 00 2A	8	3D 01 4B 46 7F FF 03 10 6D	19.8125
9	2024-08-12T11:54:40.981998+00:00	28 78 12 18 0D 00 00 EC	56	3D 01 4B 46 7F FF 03 10 6D	19.8125
10	2024-08-12T11:54:45.993176+00:00	28 08 42 8D 0C 00 00 2A	8	3D 01 4B 46 7F FF 03 10 6D	19.8125

1.4.2. Offset file

The offset file is the CSV file created from the Calibration.py code. The file is automatically saved as a CSV file with the name offsetfile, followed by each 6 bit ID in the file. It can for example be called “Offestfile_8_56.csv” where 8 and 56 represent two 6 bit ID’s.

This file has one column for each sensor in the measurements, as well as one column with timestamps for each measurement that is used for calculating offset. There is one line for each measurement used for calculation.

Time:

The time column contains a UTC time in iso format on every line. The timestamp is taken when the measurement was done on the milliK thermometer.

Other columns:

Every other column in the file contains only the data from one sensor. The first line contains the word offset, followed by the 8 byte ID of the sensor, followed by the 6 bit ID in parentheses at the end. All lanes directly below these lines contain data from the sensor with a matching ID.

The lines underneath are the offset values, these are calculated using the measurements from the milliK thermometer. For each measurement the program finds the measurement that is closest in time from the DS18B20 sensors and calculates the difference between the temperatures. The calculation is:

Offset = Reference temperature - Sensor temperature

This means that if the measurements are made with a lot of time between there may be some time between the temperatures that are being compared. This can result in a significant error in some cases. Therefore it is recommended to do the measurements frequently.

At the bottom of the file there are two lines. The first says “Average” and the second is the average offset value of the sensor with the ID it is below.

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

	Time	('Offset: 28 08 42 8D 0C 00 00 2A (8)',)	('Offset: 28 78 12 18 0D 00 00 EC (56)',)
0	2024-08-12 11:54:19+00:00	-0.17139999999999844	-0.10889999999999844
1	2024-08-12 11:54:21+00:00	-0.1711999999999989	-0.10869999999999891
2	2024-08-12 11:54:23+00:00	0.01630000000000109	0.01630000000000109
3	2024-08-12 11:54:25+00:00	0.016200000000001324	0.016200000000001324
4	2024-08-12 11:54:27+00:00	0.01630000000000109	0.01630000000000109
5	2024-08-12 11:54:29+00:00	0.016500000000000625	0.016500000000000625
6	2024-08-12 11:54:30+00:00	0.016200000000001324	0.016200000000001324
7	2024-08-12 11:54:32+00:00	0.016200000000001324	0.016200000000001324
8	2024-08-12 11:54:34+00:00	0.01630000000000109	0.01630000000000109
9	2024-08-12 11:54:36+00:00	0.016200000000001324	0.016200000000001324
10	2024-08-12 11:54:38+00:00	0.01630000000000109	0.01630000000000109
34		Average	Average
35		0.004267647058824555	0.007944117647059849

1.5. Different functionalities

There are several uses for this program. Here I will talk about 3 of them. Reading the ID’s for a sensor, calculating offsets, and plotting linearity.

1.5.1. Reading the ID’s for a sensor

When you want to find the 8 byte ID or the 6 bit ID of a specific sensor you can only do one at a time since there is no way of knowing which sensor is which in the files, without the ID.

To find the ID you connect one sensor to the calibration box and run the DS18B20_reading.py script. You can use whatever values you like because you can end the program by pressing “Ctrl + c” in the terminal after the first datatransfer is done. You can see if the first datatransfer is done by waiting for the blue led to turn on and off again in the hardwear box. This will indicate that one full dataset has been sent over the USB cable. You will then get a file with only one sensor ID.

1.5.2. Calculating offset

To calculate the offset on 1 to 16 sensors you connect all the sensors you want the offset for and run the DS18B20_reading.py code for as long as you deem necessary. At the same time you collect the reference temperatures from the milliK thermometer on a flash drive. This will result in a CSV file with all the temperatures from the sensors on your computer and a csv file with all the reference temperatures on the flash drive.

See datasheet for more information on the miliK CSV file.
https://isotech.co.uk/wp-content/uploads/2020/09/milliK-User-Manual-version-3_02.pdf

Use the Calibration.py program with both these files to calculate the offset values. How to do this is explained earlier in the document. After this you close all plots that are created and are left with a CSV file with all offset values.

Meteorologisk institutt
Meteorological Institute
Org.no 971274042
post@met.no

Oslo
P.O. Box 43 Blindern
0313 Oslo, Norway
T. +47 22 96 30 00

Bergen
Allégaten 70
5007 Bergen, Norway
T. +47 55 23 66 00

Tromsø
P.O. Box 6314,
Langnes
9293 Tromsø, Norway
T. +47 77 62 13 00

www.met.no

1.5.3. Plotting linearity of a sensor

Part of the Calibration.py code contains a script that plots 3 plots. The first plot is the temperature of each sensor over time. The second plot is a histogram of offset values for each sensor, and the last plot is offset values over time for each sensor. The sensors are sorted into different colours in the plots.

To access these plots you only have to calculate the offset like explained earlier for whatever time period you want. This can be used to see both how stable a sensor is or how fast it changes temperatures compared to other sensors.

To change the types of plots in the program you can simply go to the bottom of the Calibration.py code and edit the code using the matplot library.