Temperature, Humidity and Pressure Monitoring with Raspberry Pi

On this page is a tutorial to set up different sensors, which are read out by a Raspberry Pi, send via an ethernetcable to an InfluxDB database located on the Magnesium Server. There the measurements are stored and can be visualized in the browser using Grafana, also running on the Magnesium Server.

The monitoring setup was developed by Etiénne Wodey. It is explained in

this talk

by him. The main aspects on how to set it up are documented in

this email

.

Below there's an overview of the functionality of an used Raspberry Pi (including it's firmware), which monitors temperatures with multiple 1-wire sensors. The data can be accessed here.

Magnesium Server

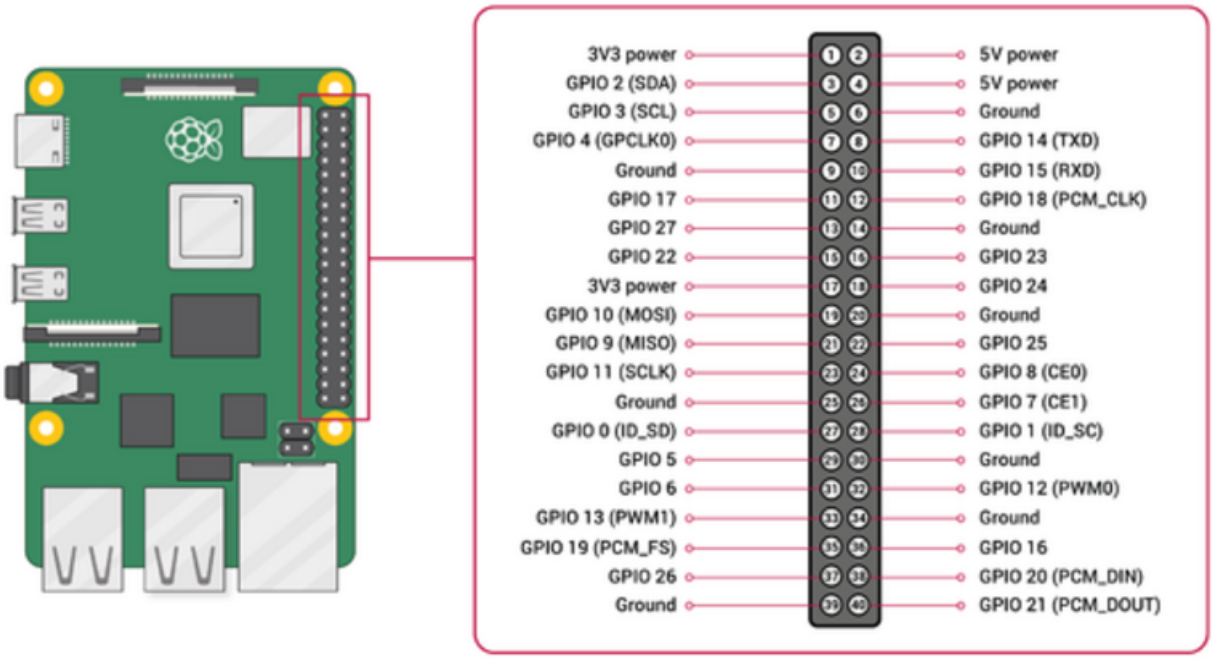
- Hostname: `thingol`, runs Debian 11
- Login: magnesium
- Password: *the standard mg password*
- Root password: *the standard reslab password*

Just use the institutes server for the databases. Someone's always taking care of it.

Hardware Setup

Besides a Raspberry Pi - in use is the Raspberry Pi 2 Model B V1.1, the datatransfer takes place over a ethernetcable, the Raspberry Pi 3 is the first one which has a build-in wireless chip - two sensors are needed. To set up your Raspberry Pi check this tutorial or use the firmware if you want to copy the temperature monitoring from Reslab. In use are the sensors AM2302 for humidity and temperature and BMP280 for pressure and temperature.

The only difference between AM2302 and DHT22 is that one has an integrated 5,1 K Pullup-Resistor and attached cables and the other one has pins and needs an external Pullup-Resistor (around 10 K) between the dataconnection and the VCC-Pin, both cases are covered below.



Raspberry Pi	**BMP280**
3,3 V	1 VCC
Ground	2 GND
SCL (GPIO 3)	3 SCL
SDA (GPIO 2)	4 SDA
NC	5 CSB
NC	6 SD0

Raspberry Pi	**AM2302**	**DHT22**
3,3 V	Red	1 VCC
Ground	Black	4 Ground
some GPIO Pin	Yellow	2 Data

This sensor works with 5 V supply voltage, too, but the sensor uses the same voltage on it's dataconnection and the Raspberry Pi can only take 3,3 V on it's GPIO pins, which is why we run the sensor with 3,3 V supply voltage. If you work with an Arduino, consider using 5 V supply voltage instead.

Read sensors manually

The communication with the sensors is written in C, but there are Python-packages from Adafruit to easily integrate the communication in personal codes and applications. Here's a short overview on how to read out the sensors:

After setting up and updating the Raspberry Pi with `sudo apt-get update` and `sudo apt-get upgrade`you need to enable I2C and 1-Wire connection. Enter the command `raspi-config` and go to Interfacing Options or Advanced Options (depends on your model) and enable I2C and 1-Wire.

Reboot afterwards.
You can check if the I2C kernel module is currently loaded by entering the command `lsmod | grep i2c_`. It should print the following modules:

```
i2c_bcm2835 16384 0
i2c_dev 20480 0
```

If you don't see anything like this, there are several things to do, to force your Raspberry Pi to load the module. Here are just some of them:

- Edit the configfile - which decides which modules should be loaded while booting - with `sudo nano /boot/config.txt`. Edit or add the line `dtparam=i2c_arm=on`, then recompile the rpi-firmware package with `make rpi-firmware-rebuild`, regenerate the SD-Card image with `make` and reboot.
- If you can't find the i2c-1 bus in /dev, start the i2c module manually by entering `modprobe i2c-bcm2835` and `modprobe i2c-bcm2835`. Go to /dev and check with `ls -la /dev/i2c-1`if the i2c directory does exist now, which means it's usuable now. This does the same as just using the GUI as described previously, but maybe this way will solve your problem.

Next you need to install some packages using the following commands:

```
sudo apt install -y python3-smbus i2c-tools
sudo apt-get install build-essential python-dev python-openssl git-core
sudo apt install python3-pip -y
pip3 install adafruit-circuitpython-bmp280
```

To check if your Raspberry Pi found the sensor and to get it's address, enter `i2cdetect -y 1`. There should be at least one entry, which is your address. If e.g. the last line of the output looks like this

```
70: – – – – – – 76 –
```

the sensor has the address 0x76. If you can't see any entry and i2c is enabled and running, check your wires and connections. Now you can write your python script! First create a file with e.g. `touch bmp280.py` and edit it with `nano bmp280.py`. All it takes to read out temperature and pressure are the following lines. Be aware to enter the correct address!

```
<fc #008000>#!/usr/bin/python3</fc>

<fc #008000>import</fc> <fc #6495ed>board</fc>
<fc #008000>import</fc> <fc #6495ed>busio</fc>
<fc #008000>import</fc><fc #6495ed> adafruit_bmp280</fc>

i2c = busio.I2C(board.SCL, board.SDA)
bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c, address=0x76)

<fc #008000>print</fc>(<fc #ff0000>“Temperature: %0.1f C”</fc> % bmp280.temperature)
<fc #008000>print</fc>(<fc #ff0000>“Pressure: %0.1f hPa”</fc> % bmp280.pressure)
```

You can execute the program with `python bmp280.py`. If you create this file in the /usr/local/sbin directory and allow everybody to execute the program with `chmod +x`

`/usr/local/sbin/bmp280.py` you can simply execute it everywhere by entering `bmp280.py`. Now we take care of the software for the AM2302 sensor. Therefor you don't even need to write a script. Go to your home directory with `cd ~` and clone the branch of the AM2302 python project with the following command.

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

Then change into the directory you just downloaded:

```
cd Adafruit_Python_DHT
```

And compile the code you downloaded, so that it fits your board.

```
sudo python3 setup.py install
```

From now on you can manually read out humidity and temperatur by entering the following command in any directory. The first number is the argument for the sensormodel you use (11,22, or 2302), the second argument is the GPIO pin number (not the physical pin number!) you attached your dataconnection to. For example, if your AM2302 sensor is connected to Pin 12, you enter the following command

```
./Adafruit_Python_DHT/examples/AdafruitDHT.py 2302 18
```

If your Raspberry Pi can't find installed packages and your using python3, change the top line in AdafruitDHT.py to “#!/usr/bin/python3” e.g. with `nano` When you implement a continuous monitoring be aware that the AM2302 can only read a value every 3 seconds, otherwise it'll hang itself up - due to the slow 1-wire connection - and needs to be restartet (unplugged).

Read sensors automatically with collectd

Here's an overview of what collectd is and does.

`collectd` is run on the Raspberry Pi to read out the sensors and send the messages to the influxdb database located on another server.

install collectd:

```
sudo apt-get update
sudo apt-get install collectd
```

check the status of the daemon:

```
sudo systemctl status collectd
```

if the status is running, stop the daemon:

```
sudo systemctl stop collectd
```

to view all systemlog messages by collectd:

```
journalctl -u collectd -b
```

The collectd configuration file is located at /etc/collectd/collectd.conf. Edit it:

```
sudo nano /etc/collectd/collectd.conf
```

Copy the contents of this file into there.

Make sure to adjust the name of the server in the network plugin section to your needs.

Starting collectd

Start the collectd service using:

```
sudo systemctl start collectd
```

Check if it runs without errors by reading

```
sudo systemctl status collectd
```

or checking

```
journalctl -u collectd -b
```

The system automatically gets new slaves but doesn't support removing slaves while running

InfluxDB

InfluxDB is an open-source time series database, which stores the by the daemon collected data and sends it to Grafana.

InfluxDB has only binary files for 64-Bit architectures. To check your architecture enter `uname -a`. If the output contains `armv7` or lower versions, you need to change your architecture from 32-Bit to 64-Bit. Note that you will need a CPU that is capable of ARMv8 for the 64-bit kernel to boot. Currently this is only available on *Raspberry Pi 3* or higher - with one exception: *Raspberry Pi 2 Model B v1.2*.

If you see something like `armv8` you already have arm64 and you can skip the following step.

Switch to 64-Bit Kernel

Make sure, that your Raspberry Pi is up to date with

```
sudo apt update
sudo apt upgrade
```

Make sure, that you are running the newest version of Raspberry Pi OS called Buster. Enter `lsb_release -a` and check the description. If you're not running Buster, you can't switch your OS to 64-Bit.

Now to verify that the 64-bit kernel exists:

```
ls /boot/kernel8.img
```

If it exists, it will simply print out the path to it. Otherwise, it will tell you: `No such file or directory`. Edit the config file with

```
sudo nano /boot/config.txt
```

And add a new line at the very end containing `arm_64bit=1`. Now reboot with `sudo systemctl reboot`. Check again your architecture with `uname -a`, which will output something like this:

```
Linux raspberrypi 4.19.97-v8+ #1294 SMP PREEMPT Thu Jan 30 13:27:08 GMT 2020 aarch64 GNU/Linux
```

or

```
Linux pi400 5.10.17-v8+ #1403 SMP PREEMPT Mon Feb 22 11:37:54 GMT 2021 aarch64 GNU/Linux
```

Install InfluxDB

First you need to manually download and install the influxd arm64 binary by entering the following command if you haven't changed your architecture to amd64.

```
wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.1.1-linux-arm64.tar.gz
```

Then you need to extract the downloaded binary

```
tar xvzf influxdb2-2.1.1-linux-arm64.tar.gz
```

Place the extracted influxd executable binary in your system $PATH to execute it without the prefix `./`.

```
sudo cp influxdb2-2.1.1-linux-arm64/influxd /usr/local/bin/
```

Mg-Server

Creating the database

Create a database in which you later wish to store the measurements, for now we call it `collectd_test`.
First, start influxDB by entering `influxd`, then execute

```
CREATE DATABASE collectd_test
```

Check if it was created properly using:

```
USE collectd_test
```

Exit using

```
EXIT
```

Creating a user

In order to secure InfluxDB against external access, we set up an user account.

```
CREATE USER magnesium WITH PASSWORD 'influxMG' WITH ALL PRIVILEGES
```

Now we need to configure `/etc/influxdb/influxdb.conf` to only allow access with a valid username and password. For this to work, edit the file and move down to the [http] section. Enable

```
auth-enabled = true
```

(See also:
https://docs.influxdata.com/influxdb/v1.6/administration/authentication_and_authorization/#set-up-authentication)

Configuring for collectd

Stop the influxdb server by running:

```
sudo systemctl stop influxd
```

The configuration file is `/etc/influxdb/influxdb.conf`. The documentation can be found here:
https://docs.influxdata.com/influxdb/v1.6/administration/config/

The config file we use is

this

.

The interesting part is:

```
[[collectd]]
  enabled = true
  bind-address = ":25826"
  database = "collectd_test"
  security-level = "encrypt"
  auth-file = "/etc/influxdb/auth_file_collectd"
```

This enables the collectd plugin for influx and writes to the database we specify with credentials required. Communication is done encrypted.

Now create a new file name `/etc/influxdb/auth_file_collectd`

```
magnesium: influxMG
```

This is the username and password for any external collectd services to authenticate with the influxdb database.

types.db

The collectd plugin for InfluxDB requires a types.db document. Copy this from `/usr/share/collectd/` on the raspberry pi to `/usr/local/share/collectd/` on the server running influxDB.

Restarting the server

Start the server by typing

```
influxd
```

and leave the terminal open. If there are any errors, these should be displayed now.

To check if there are values being written into the database, use another terminal or ssh to log into influx:

```
influx
```

and check for values in the database:

```
USE collectd_test
SHOW SERIES
```

If something is displayed, try listing those values by using

```
SELECT * FROM <name of the time series>
```

IQ-Server

Only Admins are allowed to create a database and users on the institute's server.

Configure CollectD

You have to write the login data you receive from your admin and the hostname (probably "https://log.iqo.uni-hannover.de" in the configfile for collectd, e.g. with
`sudo nano /etc/collectd/collectd.conf`
The given security level "Sign" allows data processing of signed data over the default port "25826" if the correct login data is given. Some examples are given below.

The rest of collectd.conf is identical to the one for working with the Mg-Server (only Plugins used are syslog, network and python). You can find more information here.

Configure InfluxDB

Enter your security level, database name, port, directory of saved data and additional configurations in `sudo nano /etc/influxdb/influxdb.conf` in the paragraph collectd. An example for InfluxDB v1.6.4 can be found below.

Automated Backups

In order to have the whole database constantly backed up, a daily cronjob is ran, which saves the whole InfluxDB to /home/magnesium/backups.

First, create the backup folder, while being logged in as "magnesium"

```
mkdir /home/magnesium/backups
```

The backup-script itself is rather simple. Open a new backup script via

```
sudo nano /etc/cron.daily/influx_backup
```

and copy in the following code

```
#!/bin/bash

#Stop influx db
systemctl stop influxdb

#Backup the influx DB
```

```
cd /home/magnesium/backups
date=$(date +%Y%m%d_%H%M%S)
command="tar -czvf influxdb_backup_$date.tar.gz /var/lib/influxdb"

eval $command
chown magnesium influxdb_backup_$date.tar.gz
chgrp magnesium influxdb_backup_$date.tar.gz

#Start influxdb
systemctl start influxdb
```

The backups are tagged with the date and time of creation.

Grafana

Mg-Server

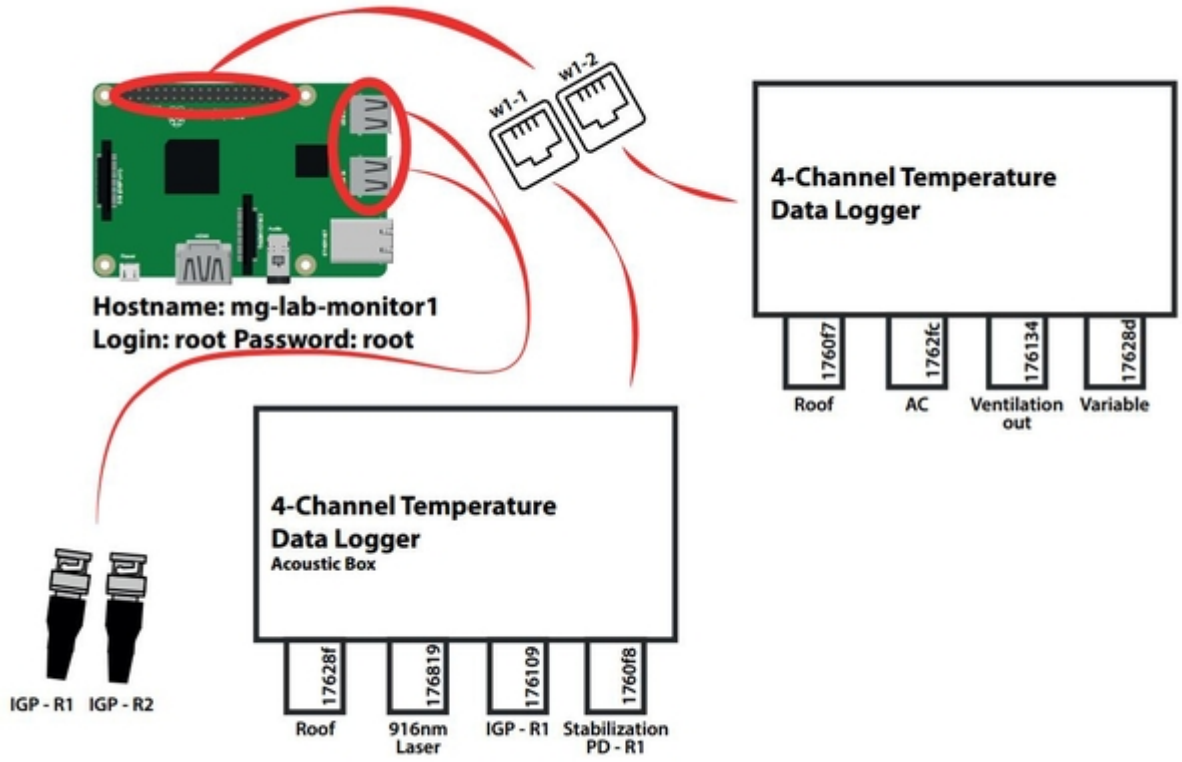
The Grafana Server was pretty much set up as described in the getting started page on their website.

IQ-Server

Here you can find all the dashboards.

Temperature Monitoring ResLab

In Reslab there is an active temperature monitoring with the following basic principle.



rasptemp1.pdf

To-Do

☐ Warn-Emails senden an Res-Lab Leute

- ☐ Emailserver ausfsetzen
- ☐ Akustikboxdach wärmer als 32.5°C
- ☐ Klimaanlage wärmer als 20°C
- ☐ Quelltextdatei erneuern (da Klaus Messbereich von 100°C auf 300°C geändert)

Firmware

A cloned image can be found at `\\AFS\iqo.uni-hannover.de\projects\magnesium\Software\Raspberry Temperature Control\rasp_temp_control.img`

This is the image for a basic setup. The operating system used is the minimal raspbian installation.

Python Plugin

The temperature sensors are read out by a python module using the python plugin for collectd. Place the script

w1_therm_monitor.py

in the folder `/home/pi/` or change the ModulePath option for the python plugin in `/etc/collectd/collectd.conf` if you want to place it in another folder.

The collectd module reads `w1_master_slaves`, takes all the serial numbers starting with 28 (DS18B20 family) or 3b (MAX31850 family) and reads the corresponding slave files (i.e. the temperatures).

Three w1-Busses

To have three one wire busses running at the same time use the modified device tree by Etiénne and copy it to `/boot`. The device tree can be found in `\\AFS\iqo.uni-hannover.de\projects\magnesium\Ehemalige Mitarbeiter\Rasmus Holst\Temperature Monitoring with RasPi`
In `/boot/config.txt` add the line

```
device_tree=name_of_the_device_tree
```

replacing `name_of_the_device_tree` with the actual name of the file located in `/boot`.

Another option is to do it by overloading the dtoverlay parameter like described here. This has proven to be easier. Edit '/boot/config.txt' and replace the old code to enable w1 with the following lines:

```
    # Enable w1-Bus
    dtoverlay=w1-gpio,gpiopin=15 # corresponds to header pin 10
    dtoverlay=w1-gpio,gpiopin=14 # corresponds to header pin 8
    dtoverlay=w1-gpio,gpiopin=4 # corresponds to header pin 7
```

The order in which you configure the busses is inverse to the numbering of them. Like this w1-bus-master1 is on pin 7, w1-bus-master2 is on pin 8 and w1-bus-master3 on pin 10. Depending on your Setup you might need to change the pin numbers.

Modify the python plugin section in `/etc/collectd/collectd.conf` to this:

```
<Plugin python>
        ModulePath "/home/pi/"

        Import w1_therm_monitor

        <Module w1_therm_monitor>
                Master "w1_bus_master1"
                Master "w1_bus_master2"
                Master "w1_bus_master3"
                MaxTemp 100
        </Module>

        LogTraces true
```

```
</Plugin>
```

Temperature, Pressure and Humidity of the Frequencycomb

To check if the temperature and humidity of the frequencycomb changes the refractive index of the fiber those values are supposed to be monitored by a Raspberry Pi. The same one may be used to check for pressurechanges due to movement in the room or by entering it.

General Data

- User: raslab, Password: mg24
- User: root, Passwort: magnesium24
- hostname: freqcombfiber
- IPv4-Address: 130.75.102.201
- MAC-Address: b8:27:eb:7b:17:a8
- InfluxDB:
  - Host: https://log.iqo.uni-hannover.de
  - Username: magnesium
  - Database: magnesium
  - Password: BttDFrUqEm
- Read sensors manually:
  - Pressure + Temperature: `pressure+temp.py`
  - Pressure + Humidity: `AdafruitDHT.py 2302 18`

Code for reading sensors automatically:

> <fc #008000>*#!/usr/bin/python3*</fc>
>
> **<fc #008000>import</fc> <fc #6495ed>board</fc>**
> **<fc #008000>import</fc> <fc #6495ed>busio</fc>**
> **<fc #008000>import</fc><fc #6495ed>adafruit_bmp280</fc>**
> **<fc #008000>import</fc><fc #6495ed>sys</fc>**
> **<fc #008000>import</fc><fc #6495ed>Adafruit_DHT</fc>**
> **i2c = busio.I2C(board.SCL, board.SDA)**
> **bmp280 = adafruit_bmp280.Adafruit_BMP280_I2C(i2c, address=0x76)**
> **sensor = Adafruit_DHT.AM2302**
> **pin = 18**
>
> **humidity, temperature = Adafruit_DHT.read_retry(sensor,pin)**
> **pressure = bmp280.pressure**
>
> **<fc #008000>print</fc>(<fc #ff0000>“Pressure: %0.1f hPa”</fc> % pressure)**

> **<fc #008000>if</fc> humidity <fc #008000>is not</fc> <fc #ff00ff>None</fc> <fc #008000>and</fc> temperature <fc #008000>is not</fc> <fc #ff00ff>None</fc>:**
>
> **temperature = (bmp280.temperature + temperature)/2**
> **<fc #008000>print</fc>(<fc #ff0000>“Temperature: %0.1f C”</fc> % temperature)**
> **<fc #008000>print</fc>(<fc #ff0000>“Humidity: %0.1f g/m3”</fc> % humidity)**
>
> **<fc #008000>else</fc>:**
>
> **print(<fc #ff0000>“Failed to read AM2302 sensor! Check connections!”</fc>)**
> **sys.exit(1)**

Plugin Network in configfile of collectd:

```
<Plugin network>
        <Server "https//log.iqo.uni-hannover.de" "25826">
                SecurityLevel "Sign"
                Username "magnesium"
                Password "BttDFrUqEm"
        </Server>
</Plugin>
```

CollectD paragraph in influxdb.conf:

```
[[collectd]]
  enabled = true
        bind-address = ":25826"
  database = "magnesium"
  retention-policy = ""
  typesdb = "/usr/local/share/main"
  security-level = "sign"
  username = "magnesium"
  password = "BttDFrUqEm"
```

To Do

- if bind address already in use find PID with `netstat -tulpn` and kill process e.g. with `sudo kill 1258`
- TSM files safed in /var/lib/influxdb/data
- types db file in /usr/local/share/collectd (example)