



The PiLogger One Manual



Table of Contents

1 Introduction.....	5
2 Safety instructions.....	6
3 Selecting the host computer.....	7
3.1 Raspberry Pi Family.....	7
3.2 ESP32 Microcontroller.....	9
4 Installing WebMonitor on Raspberry Pi.....	10
4.1 Installing the Raspberry Pi OS.....	10
4.1.1 Operating system selection.....	10
4.1.2 Prerequisites for SD card creation.....	10
4.1.3 Download Raspberry Pi Imager.....	11
4.1.4 Using the Raspberry Pi Imager.....	12
4.1.5 First Boot, SSH Connection.....	18
4.1.6 Setting up the system.....	20
4.1.6.1 Updating Raspi-Config.....	21
4.1.6.2 System Options.....	22
4.1.6.3 Display Options.....	23
4.1.6.4 Interface Options.....	23
4.1.6.4.1 I4 I2C.....	24
4.1.6.5 Performance Options.....	25
4.1.6.6 Localisation Options.....	25
4.1.6.6.1 L1 Locale.....	26
4.1.6.6.2 L3 Keyboard.....	28
4.1.6.7 Advanced Options.....	29
4.1.6.8 Finishing Raspi-Config.....	29
4.1.7 General Update.....	30
4.1.7.1 Firmware Update.....	30
4.1.7.2 Distribution Update.....	30
4.2 Assembly and Commissioning.....	32
4.2.1 Assembly of the PiLogger One Type A.....	32
4.2.1.1 Preparation.....	32
4.2.1.2 Attaching the Module.....	35
4.2.2 Connecting the PiLogger One Type B.....	38
4.2.3 Switching on for the first time.....	39
4.3 Installing the PiLogger WebMonitor.....	41
4.3.1 Setting up the router as an NTP server.....	41
4.3.2 Running the Installer.....	42
5 Installing WebMonitor on ESP32 Module.....	46
5.1 Soldering adapter and module.....	47
5.1.1 Adapter with NodeMCU ESP32 module.....	49
5.1.2 Assignment of optional pin fields.....	50
5.2 Installing Thonny.....	51
5.3 Checking/updating 'esptool'.....	52
5.4 Preparing the module with adapter.....	53
5.5 Flashing MicroPython.....	53
5.6 Copying the archive files.....	56

5.7 Customizing the WiFi access data.....	58
5.8 Changing the network name.....	59
5.9 Mounting the PiLogger One.....	60
5.10 Switching on for the first time.....	62
6 Operating instructions WebMonitor Software.....	63
6.1 Calling up the web page with the browser.....	63
6.2 The Homepage - <i>Live Values</i>	64
6.3 The page <i>Live Curve</i>	66
6.3.1 Setting the update interval.....	67
6.4 The page <i>Diagrams</i>	68
6.4.1 Filling the curve.....	68
6.4.2 Using a moving average.....	69
6.4.3 Selecting the series of measured values to be displayed.....	70
6.4.4 Showing a single measured value.....	71
6.4.5 Panning and zooming.....	72
6.4.5.1 Mouse operation.....	72
6.4.5.2 Touch operation.....	74
6.5 The page <i>Settings 1</i>	75
6.6 The page <i>Settings 2</i>	78
6.7 The page <i>Calibration</i>	82
6.8 The page <i>Download</i>	84
7 Connection terminals.....	86
7.1 DC voltage and current measurement.....	86
7.2 Pulse counting input.....	87
7.3 Temperature sensor input.....	88
8 Pin assignment GPIO strip.....	90
9 Application programming interface.....	91
9.1 Configuration registers.....	91
9.2 Measured value registers.....	93
9.3 Value ranges and scaling.....	94
10 Temperature sensor tables.....	96
11 Technical data.....	99
12 Calibration.....	100
12.1 Temperature measurement calibration.....	100
12.2 Pulse counting calibration.....	102
12.3 Voltage measurement calibration.....	102
12.4 Current measurement calibration.....	104
13 Integration into smart home systems.....	106
13.1 Home Assistant.....	106
14 Appendix.....	110
14.1 The I ² C Tools.....	110
14.2 The concept of the PiLogger WebMonitor.....	111

PiLogger ® and the PiLogger Logo are registered trademarks of G. Weiß-Engel.

Raspberry Pi ® is a trademark of the Raspberry Pi Foundation.

Linux ® is a registered trademark of Linus Torvalds.

Windows ® is a registered trademark of Microsoft Corporation.

All other trademarks are the property of their respective owners.

1 Introduction

The PiLogger One is an extension for the Raspberry Pi. With the PiLogger One, an independent 4-channel measuring system is connected to the Raspberry Pi via the I²C bus.

With the 'Type B' variant, connection to the host computer is possible via a short 5-core cable - this means that the GPIO strip remains largely available.

There is also an adapter for an ESP32 microcontroller and a specially adapted software version.

The focus is on direct current power measurement (DC power meter), supplemented by a pulse counter with which rotation rates can be measured, such as for wind speed or flow rate, as well as a measurement input for temperature sensors. More details can be found in the 'Technical data' chapter.

This combination is particularly suitable for monitoring small alternative power generators, such as wind turbines or solar systems and especially their batteries. The highlight is the concept of an independently operating peripheral unit. The PiLogger has its own time base with which it can maintain precise measurement intervals, independent of the software activities on the host computer.

The PiLogger takes over the periodic measurements, the power calculation and the moving averaging. The Raspberry Pi only has to perform a query at the desired log times and can then continue to evaluate and save as required. The programming interface is described in a separate chapter.

With the versatility of the Raspberry Pi and especially with the possibility of networking via LAN and WLAN, great possibilities open up - for example a small server with graphical evaluations, the free software 'WebMonitor'.

This web server makes the measured values of the PiLogger One available in the home network. These can then be used in smart home systems such as 'Home Assistant' or displayed directly via the web interface with a browser on any end device.

This manual attempts to provide step-by-step instructions for beginners. Advanced users can skip the first chapters. The last chapters serve as a technical reference and documentation for the successful integration of the PiLogger into your own projects.

Have fun and success with the PiLogger !

2 Safety instructions

The following safety instructions and hazard warnings are intended not only to protect your health, but also to protect the appliance. Please read the following points carefully:

- Before touching and when connecting the circuit board, suitable protective measures against static charge must be taken (e.g. earthing strap, non-conductive carpet pad, etc.).
- Installation and connection may only be carried out in a de-energized state. All power supplies must be switched off.
- The product is only suitable for dry, enclosed indoor areas. It must not become damp or wet, otherwise it may be damaged.
- Protect the product from cold, heat, direct sunlight, dust and dirt.
- Handle the product with care as it can be damaged by knocks, blows or falling from even a small height.
- The product is not a toy and does not belong in the hands of children. Place the product so that it cannot be reached by children.
- Do not leave the appliance or the packaging material lying around carelessly, as this could become a dangerous toy for children.



When attaching the measuring cables, ensure good insulation and protection against accidental contact. Risk of electric shock!



When attaching the measuring cables, ensure strain relief and a stable structure to avoid short circuits!

3 Selecting the host computer

The PiLogger One is basically a multi-sensor on the I²C bus. This means that all computers that provide an I²C connection can be considered.

Traditionally, these are the Raspberry Pi's with their freely accessible GPIO strips (General Purpose Input Output - generally usable input/output pins).

The PiLogger One itself does not store any measured values, which is why the availability of a fast permanent storage medium - i.e. an SD card - is an important criterion for a logger (data recorder). All previous Raspberry Pi models offer this. However, they are not standard for ESP32-based computer modules. With our adapter plate, an ESP32 WROOM module can be upgraded with a micro SD card holder.

Another criterion is the trade-off between power consumption and computing power. If the PiLogger One is merely an additional extension for a computer that performs other tasks, power consumption is of secondary importance. However, the sensor cables must be routed to the location of the computer.

If the PiLogger One is the main task, for example in a stand-alone system for battery monitoring, a small, energy-saving computer with a network connection is the better choice. In particular because the 'PiLogger WebMonitor' software is based on display and operation via a browser - i.e. not necessarily locally on the host computer - the host computer for the PiLogger One can dispense with the display and keyboard and instead be mounted close to the measurement location.

3.1 Raspberry Pi Family

The basic version of the PiLogger One is equipped with a 26-pin female connector that fits directly onto the GPIO strip of a Raspberry Pi 1. It also fits on the 40-pin pin headers of subsequent generations of the Raspberry Pi - only the backwards-compatible part of the header is used.

This means that all versions, including the Zero's, can be considered.

The computing power of even the smallest versions equipped with a single-core processor, such as the Zero 1 and the four 1s (A, B and A+, B+), is more than sufficient for operating the PiLogger One with the WebMonitor.

As all Raspberry Pi's are operated with a complete Linux operating system, connecting standard peripherals is generally not a problem.

This means that even Pi's that do not have a LAN or WLAN on board can be retrofitted as required.

For example, there are LAN adapters with micro USB plugs so that a Raspberry Pi Zero (without W, like WLAN) can also be integrated into the home network.

These are even available with POE (Power Over Ethernet) so that the Raspi can

also be supplied with power directly.

Or a WLAN adapter with a rod antenna, or ... or

In the end, this means that the decision for a particular Raspberry Pi depends on the planned application and the local conditions:

A small, economical Raspi directly at the measurement location and perhaps a Raspi 3 or 4 in the living room on the monitor.

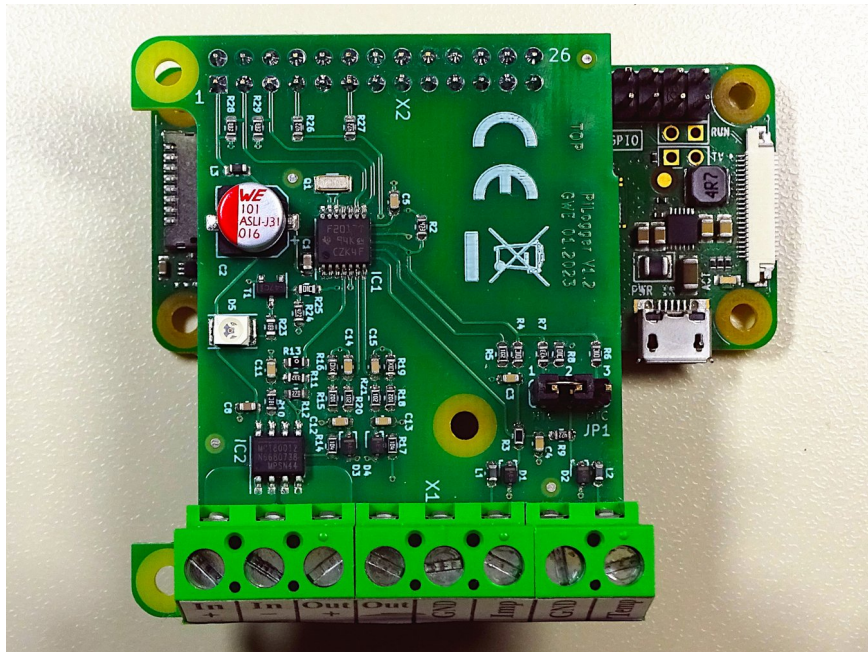


Bild 1: PiLogger One on top of Raspberry Pi Zero W

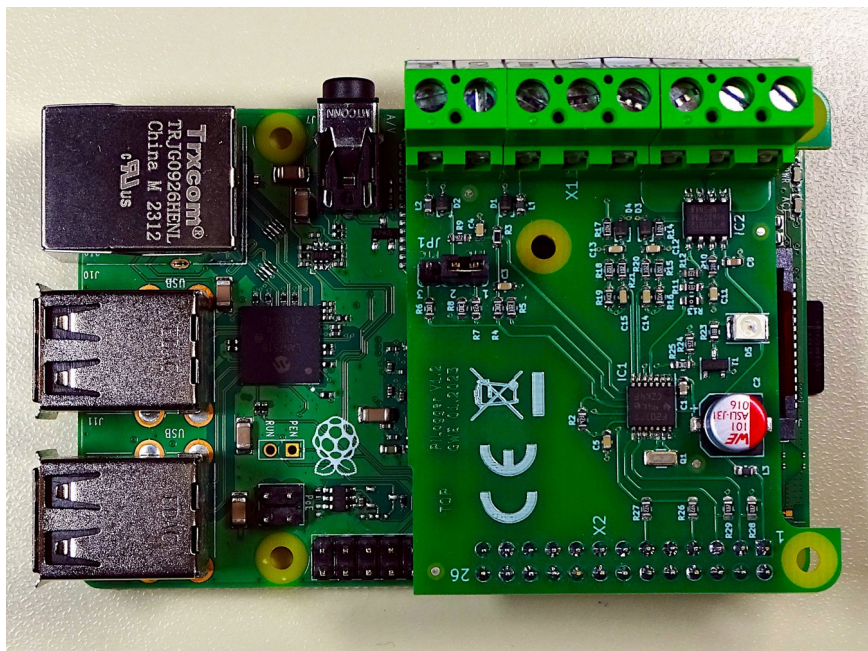


Bild 2: PiLogger One on top of Raspberry 4

The setup under Raspberry Pi OS is described in chapter 4.

3.2 ESP32 Microcontroller

MicroPython is a special implementation of Python for microcontrollers with low resource requirements. It is a free and open software project by Damien George under MIT license (<https://micropython.org/>).

Since the 'PiLogger WebMonitor' software is written in Python and does not require a keyboard or display, MicroPython is the main criterion for selecting an even more economical host computer. The second criterion is network capability. This means that all computer modules that benefit from a MicroPython port and are equipped with a WLAN module, for example, are initially shortlisted. Most of these modules do not have an SD card holder and if they do, they typically only have an SPI connection with just one data line. This means a further reduction in speed. We therefore decided to design an adapter plate with a microSD card holder for a 36-pin ESP32 WROOM module, whereby the SD card is controlled via the SDIO interface with 4 data lines.

We have successfully ported the 'PiLogger WebMonitor' for this combination.

The power consumption of the WebMonitor is thus almost 4 times as economical as with a Raspberry Pi Zero W: 230 mW to 900 mW at 5V.

The downside is a longer response time for web requests. This means a somewhat slower response when calling up pages in the browser.

Otherwise, everything works as in the Raspberry Pi version.

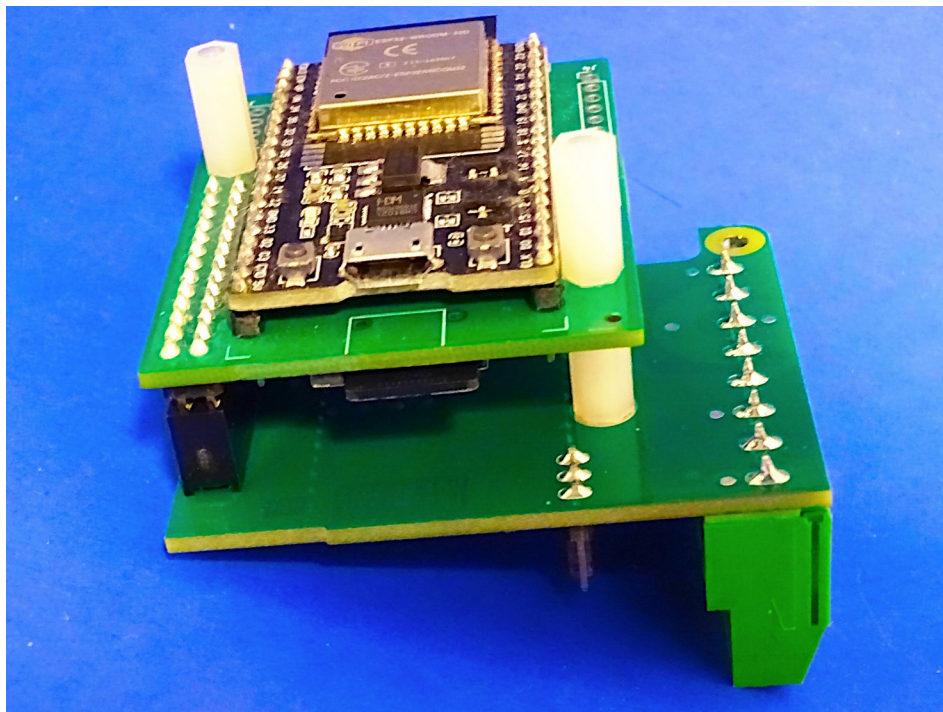


Bild 3: PiLogger One on bottom of ESP32 WROOM

The setup of the ESP32 variant is described in chapter 5.

4 Installing WebMonitor on Raspberry Pi

4.1 Installing the Raspberry Pi OS

This chapter describes step by step how to prepare the Raspberry Pi so that the PiLogger WebMonitor can then be installed.

4.1.1 Operating system selection

The PiLogger WebMonitor is based entirely on Python-based software components on the Raspberry Pi side. This means that, in principle, it can be used on any operating system (OS) with Python support. Only the special hardware support for the Raspberry Pi GPIOs under Python is still required. The aim of the following instructions is the simple and direct description of a so-called 'headless' installation - i.e. the operation of the Raspberry Pi without a connected display and input device. This means that the Raspberry runs in normal operation all by itself as a web-controlled logger with minimal hardware and therefore the lowest possible power consumption.

Therefore, the recommendation here is: Raspberry Pi OS in the Lite version. However, one of the desktop versions can of course also be used, whereby the installation is basically the same, except that the operation runs directly on the screen and keyboard of the Raspberry Pi.

Raspberry Pi OS is the official operating system provided by the Raspberry Pi Foundation, which is based on Debian Linux and is widely supported.

4.1.2 Prerequisites for SD card creation

To prepare an SD card for a Raspberry Pi, a computer with an SD card drive is of course required - whether internal or external.

In addition, an image file of the operating system must be downloaded via the Internet. Of course, the computer must have an Internet connection for this.

(An image is a large file that contains a 1:1 copy of the complete contents of a data carrier).

Since the version of the Raspberry Pi OS from 4.4.2022 (Debian 11, bullseye) there is no longer a pre-installed default user 'Pi'.

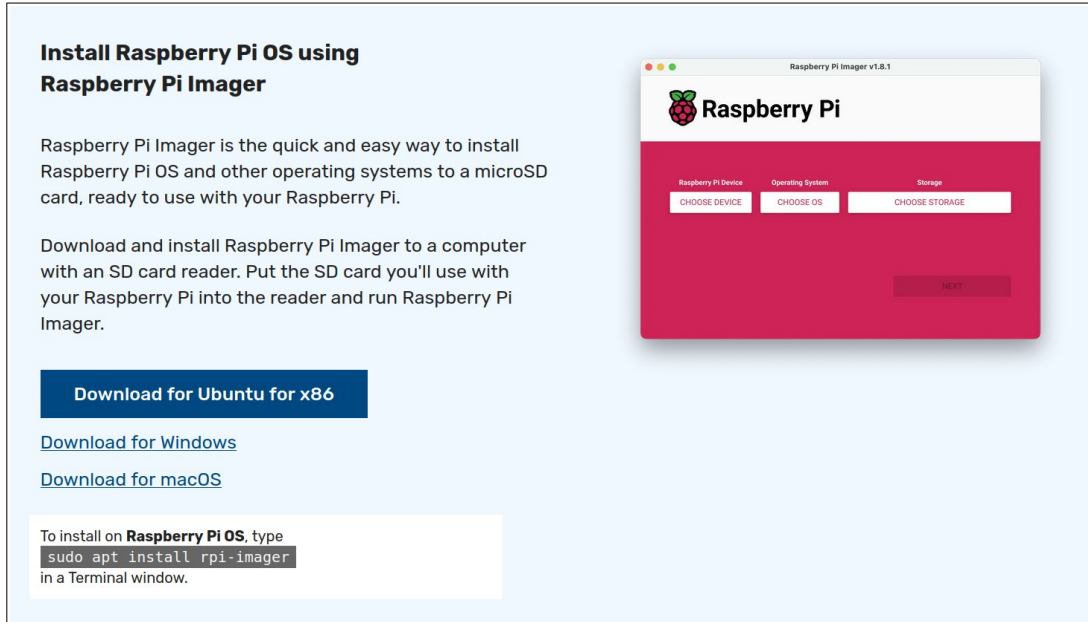
Especially for a 'headless' installation, a user must therefore be created for the first system start so that access via the network is possible at all.

A manual installation is still possible, but requires, among other things, another Linux computer (e.g. an already running Raspberry) to generate the password hash (fingerprint of the encrypted password) - in other words: it is no longer really easy.

For simple preparation, we therefore recommend using the 'Raspberry Pi Imager' utility program, which is provided by the Raspberry Pi Foundation.

4.1.3 Download Raspberry Pi Imager

On the software page of www.raspberrypi.org there is the possibility to download a comfortable program (tool) called 'Raspberry Pi Imager' for 3 different operating systems of an auxiliary computer or to install it via command line on an already running Raspberry Pi:

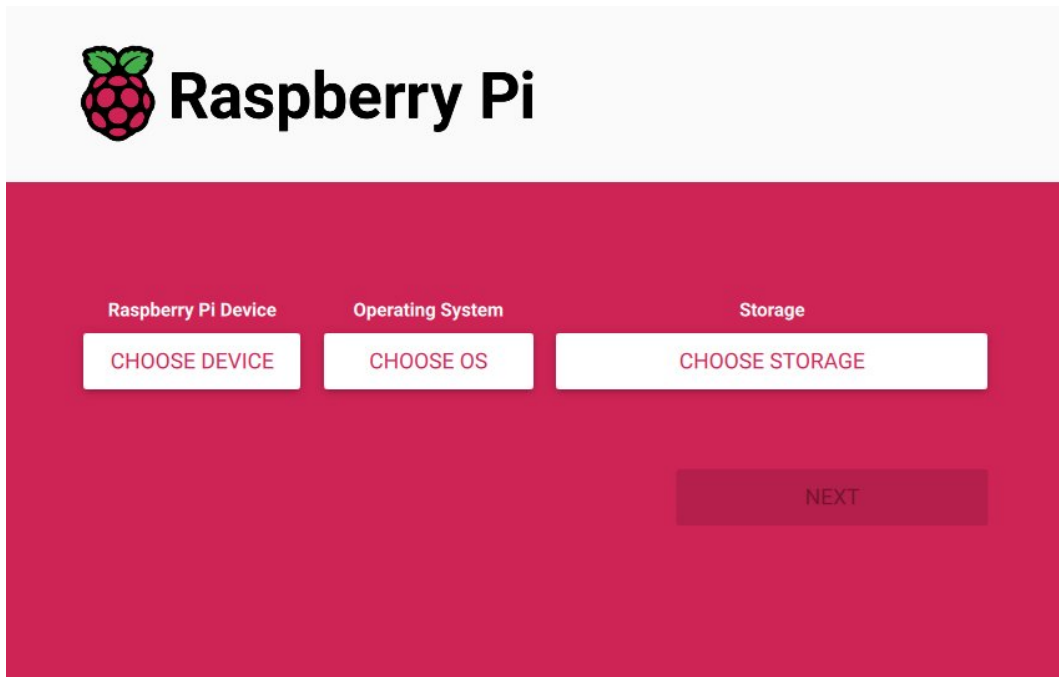


After the download for the relevant system, we install the software on our auxiliary computer.

This depends somewhat on the existing operating system, but is usually done by double-clicking on the file that has just been downloaded.

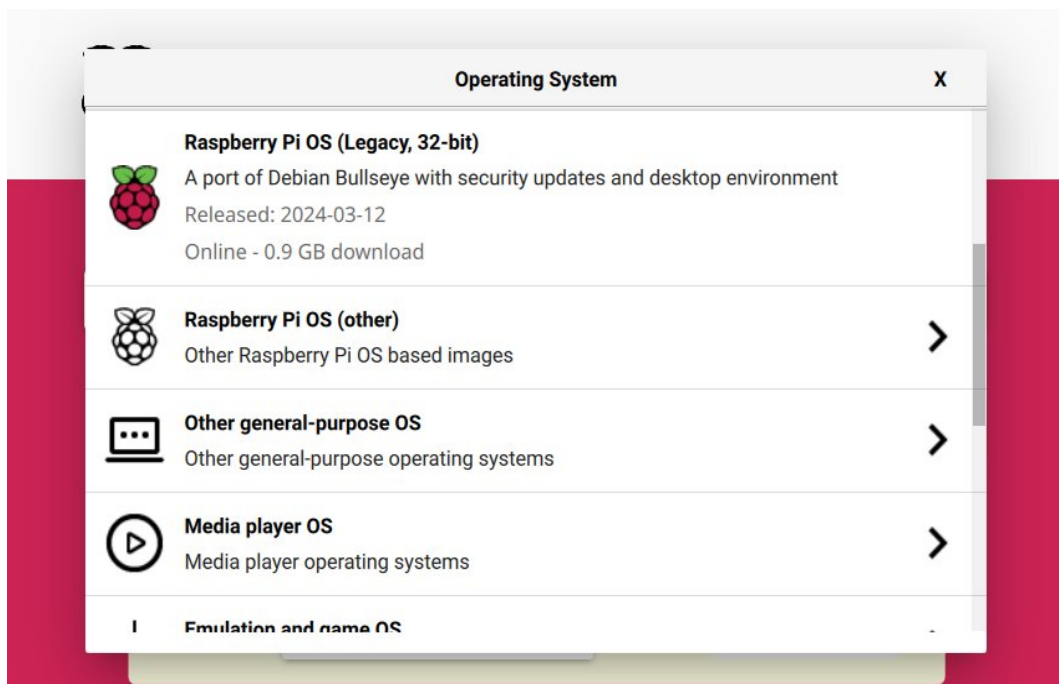
4.1.4 Using the Raspberry Pi Imager

We start the Raspberry Pi Imager and see this start window:

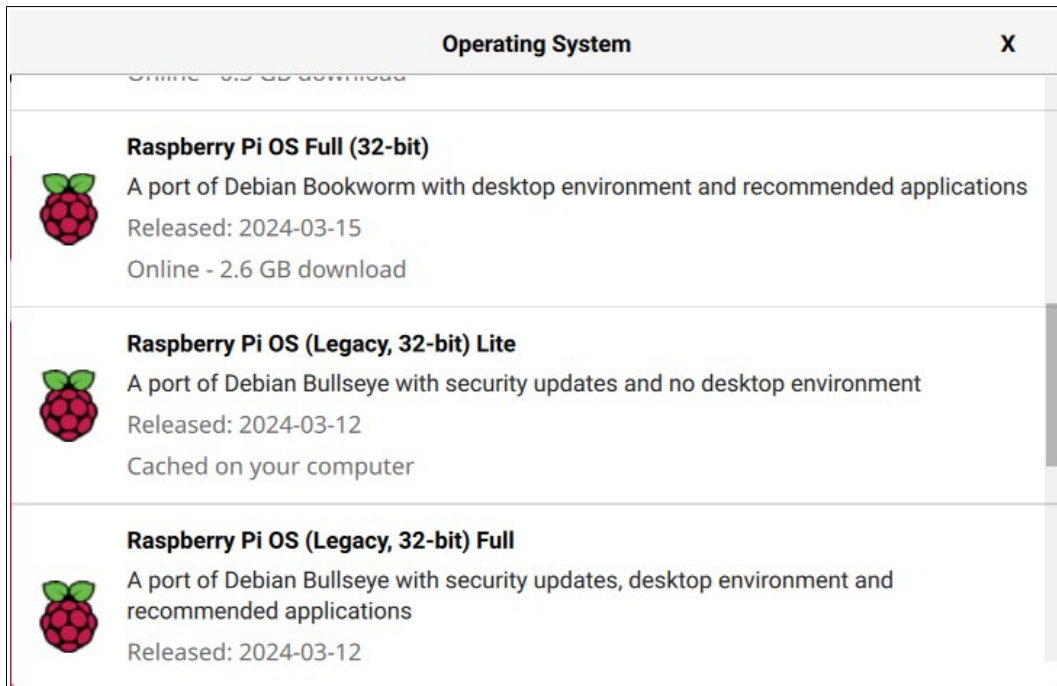


First, we need to select our existing Raspberry Pi under 'CHOOSE DEVICE'. This filters out options that are not suitable for the next selection and influences the recommendation.

In the next selection window, we now select one of the operating system variants. This selection is made in stages:



The recommendation for our model (here a Raspi 3) appears first.
Below that, under 'Raspberry Pi OS (other)', we can find the alternatives:



Since a system without the graphical user interface is to be used here as an example, the choice here is not the recommended first entry, but the entry 'Raspberry Pi OS Lite (32-bit)' from the submenu below.

The 64-bit version would also be possible, but would require more RAM and a little more power with little speed advantage - a matter of taste...

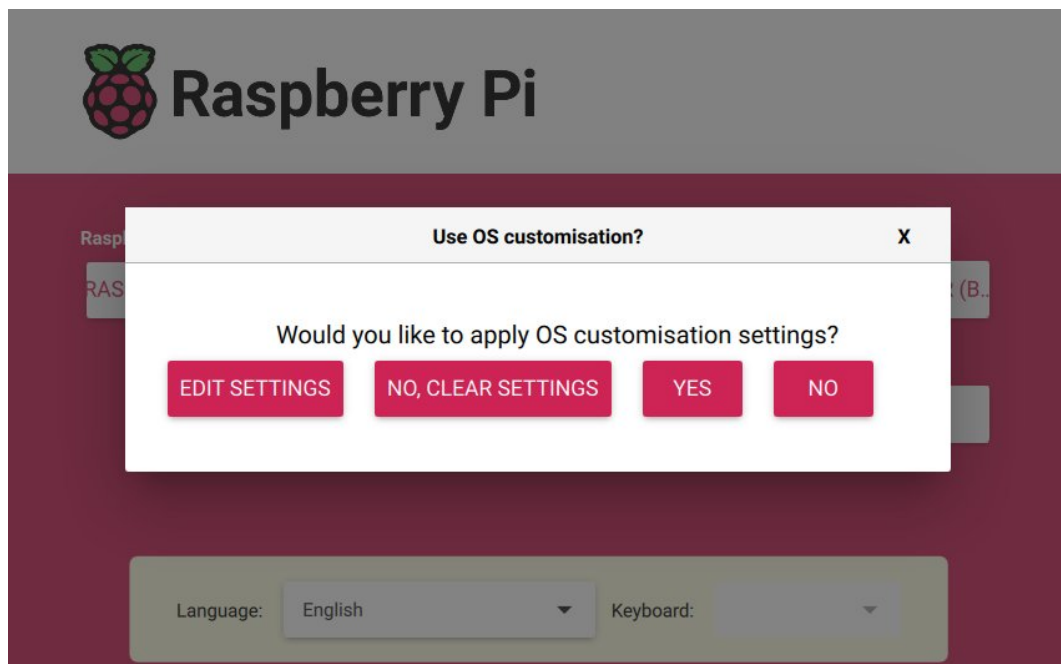
Of course, the PiLogger WebMonitor can also be used on a desktop system - i.e. with a graphical user interface - but that is not the aim of these instructions.

The Raspberry Pi Imager downloads the latest version live from raspberrypi.org - so an internet connection is required.

The downloaded image is saved on the computer for further SD cards to be flashed.

Now we need to select the correct drive with the micro SD card for the Raspi under 'CHOOSE STORAGE'.

If we now press 'NEXT', this prompt appears:



*These settings should always be edited / used to create a user and set up WLAN access!
Especially with a 'headless' installation otherwise no access is possible !*

So we go to 'EDIT SETTINGS' and get to this screen:

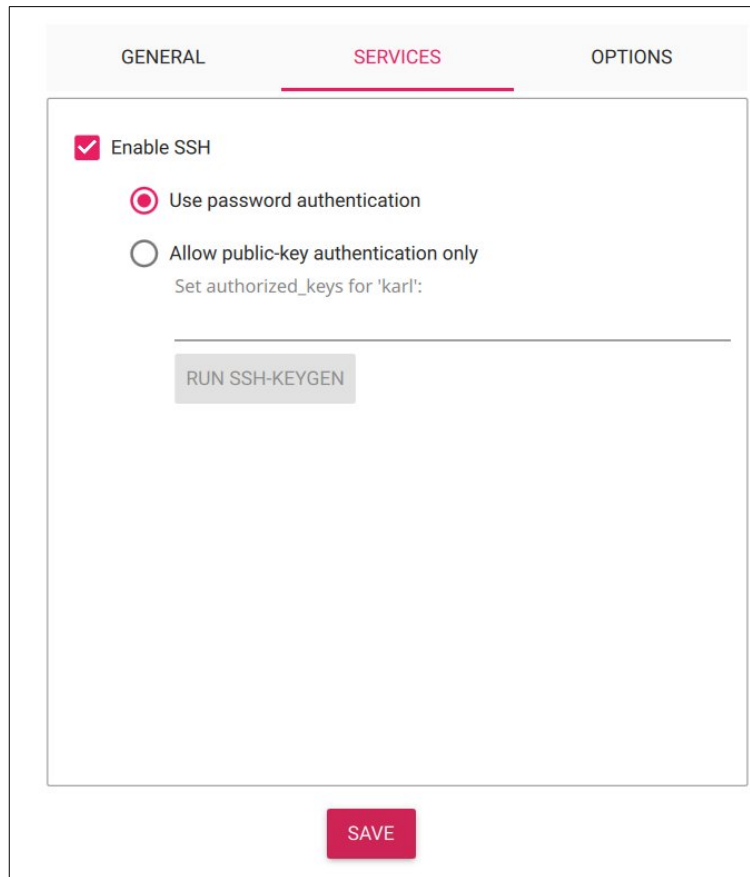
A screenshot of the 'GENERAL' settings screen in the PiLogger One configuration tool. The screen has three tabs: 'GENERAL' (selected), 'SERVICES', and 'OPTIONS'. The settings are as follows:

- ☒ Set hostname: pilogger.local
- ☒ Set username and password
 - Username: karl
 - Password:
- ☒ Configure wireless LAN
 - SSID: Homenet
 - Password:
 - ☐ Show password ☐ Hidden SSID
 - Wireless LAN country: GB
- ☒ Set locale settings
 - Time zone: Europe/London
 - Keyboard layout: gb

At the bottom center is a red 'SAVE' button.

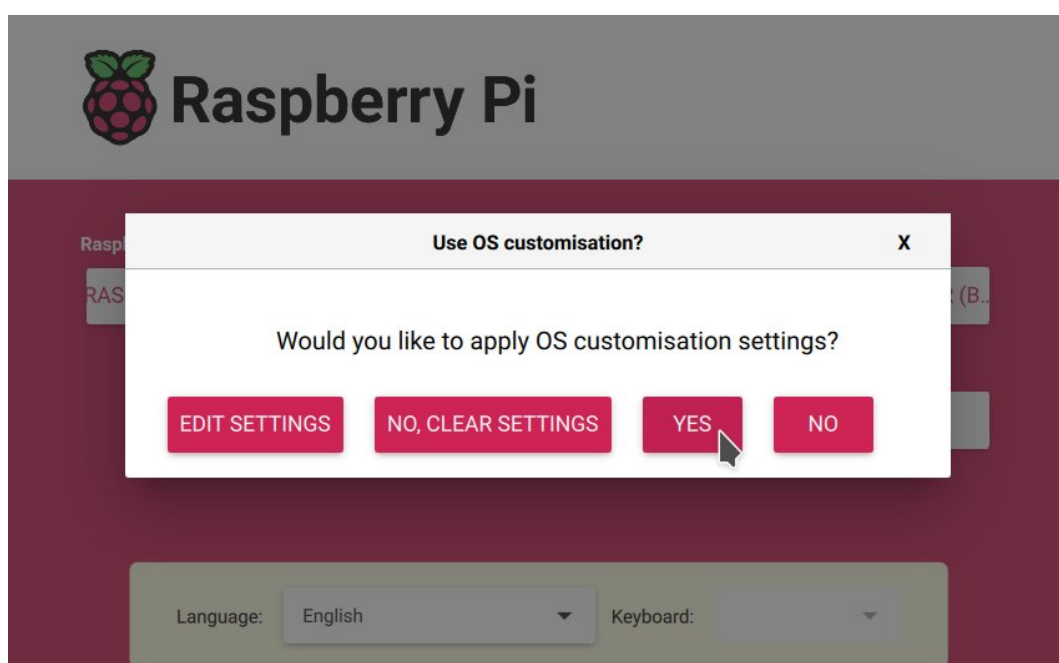
Under the 'GENERAL' tab, you can specify the network name (host name), user, Wifi (WLAN) access and language settings. Please fill in everything with your own values :)

SSH (Secure Shell) must then be activated under the 'SERVICES' tab so that remote access is possible, especially with a 'headless' system:

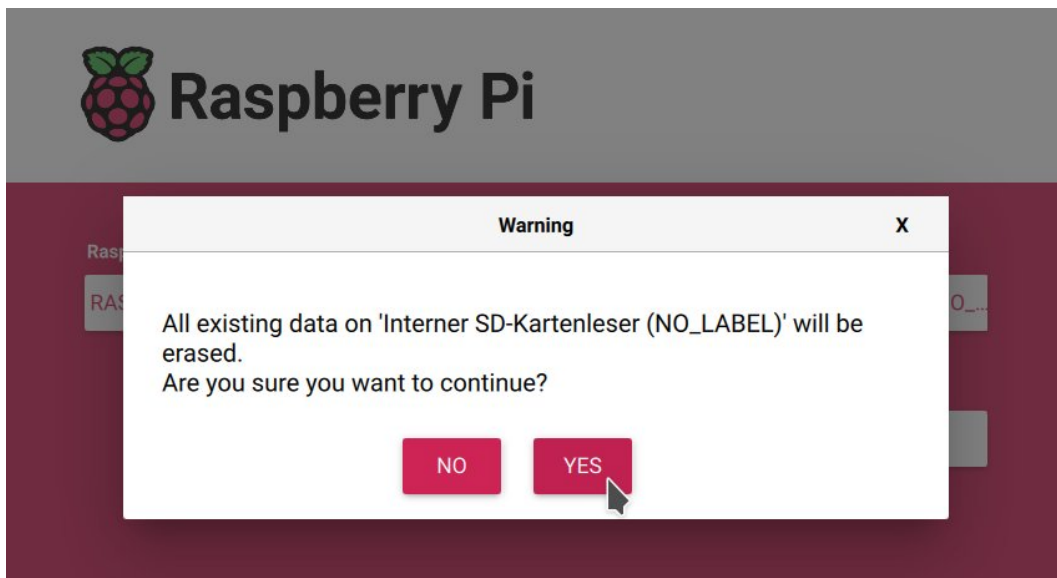


The screenshot shows the 'SERVICES' tab in the PiLogger One interface. At the top, there are three tabs: 'GENERAL', 'SERVICES' (which is active and highlighted with a red underline), and 'OPTIONS'. Below the tabs, there is a section for SSH settings. It starts with a checked checkbox labeled 'Enable SSH'. Below this, there are two radio button options: 'Use password authentication' (which is selected) and 'Allow public-key authentication only'. Under the second option, there is a text input field labeled 'Set authorized_keys for 'karl':'. Below the input field is a button labeled 'RUN SSH-KEYGEN'. At the bottom of the settings section is a red 'SAVE' button.

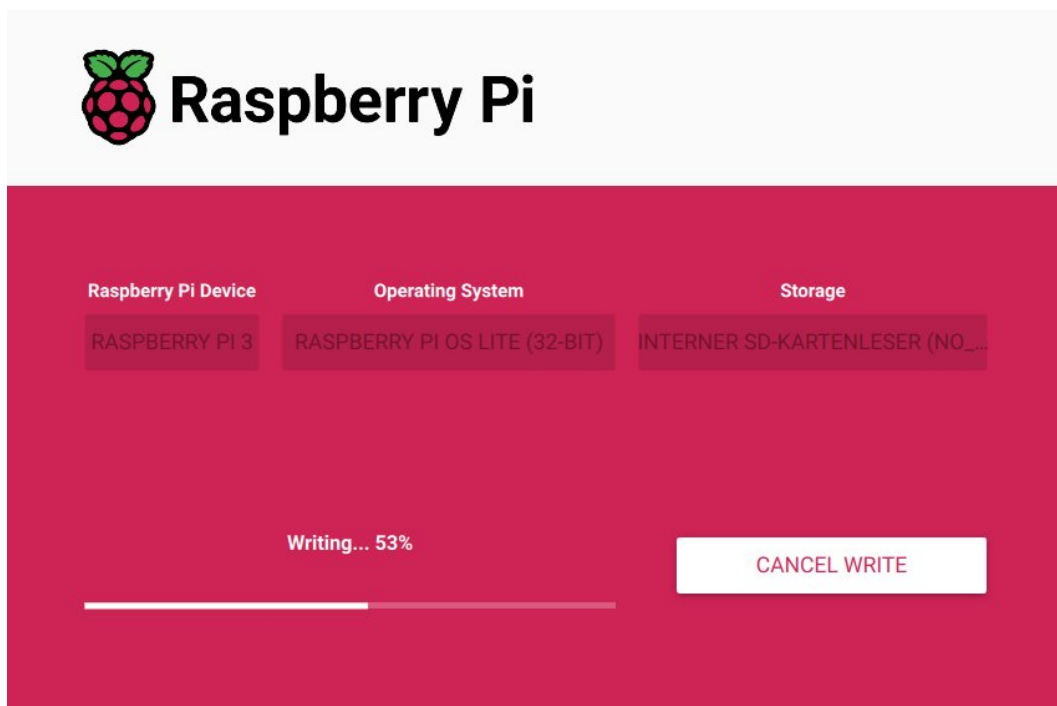
We then press 'SAVE' and return to the question:



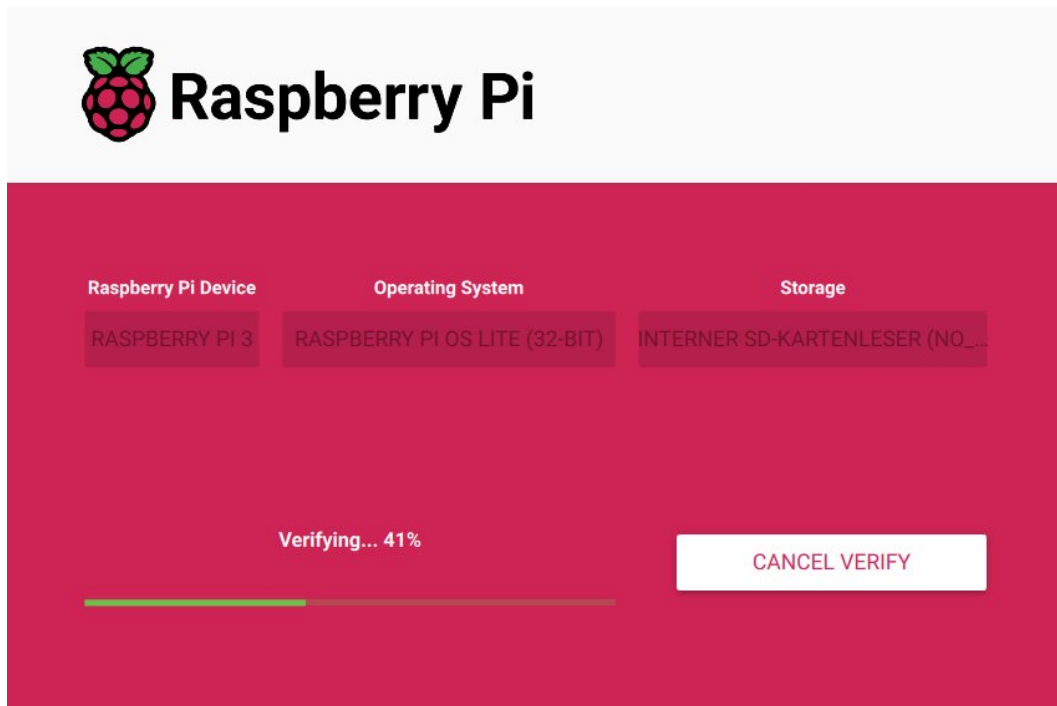
Which we now answer with 'YES'. Now we are warned that all data on the SD card will be erased - if this is the correct storage, we answer 'YES' again:



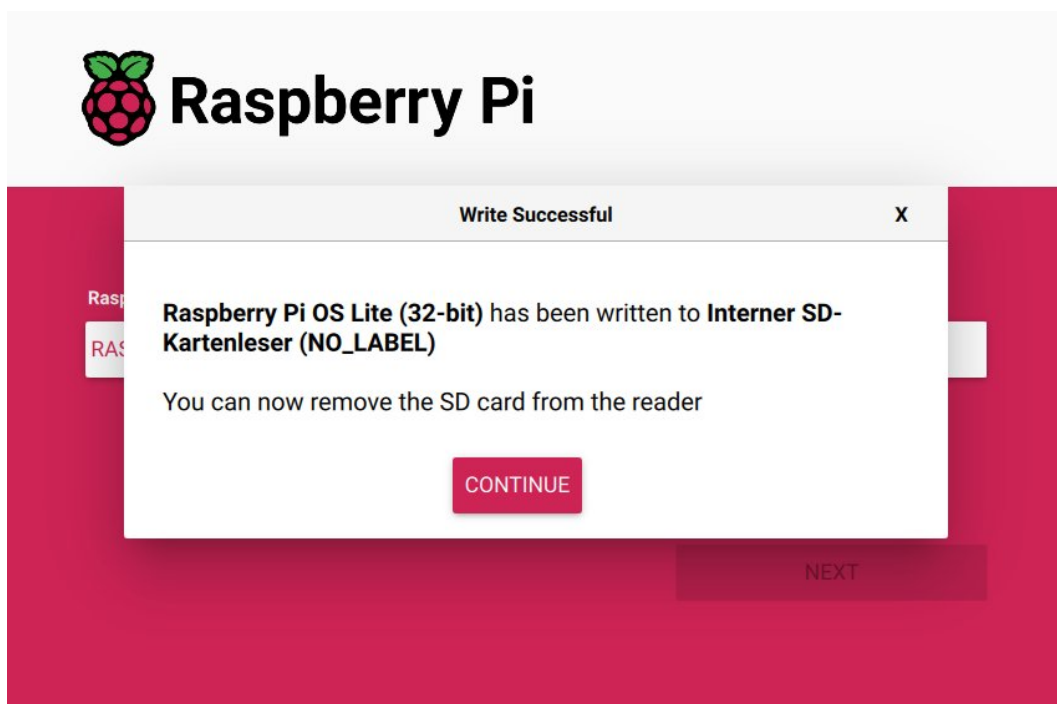
The write process will then finally start:



After writing the data, the verification is started automatically:



Successful completion is reported in this window :



The SD card can now be inserted into the Raspberry Pi.

4.1.5 First Boot, SSH Connection

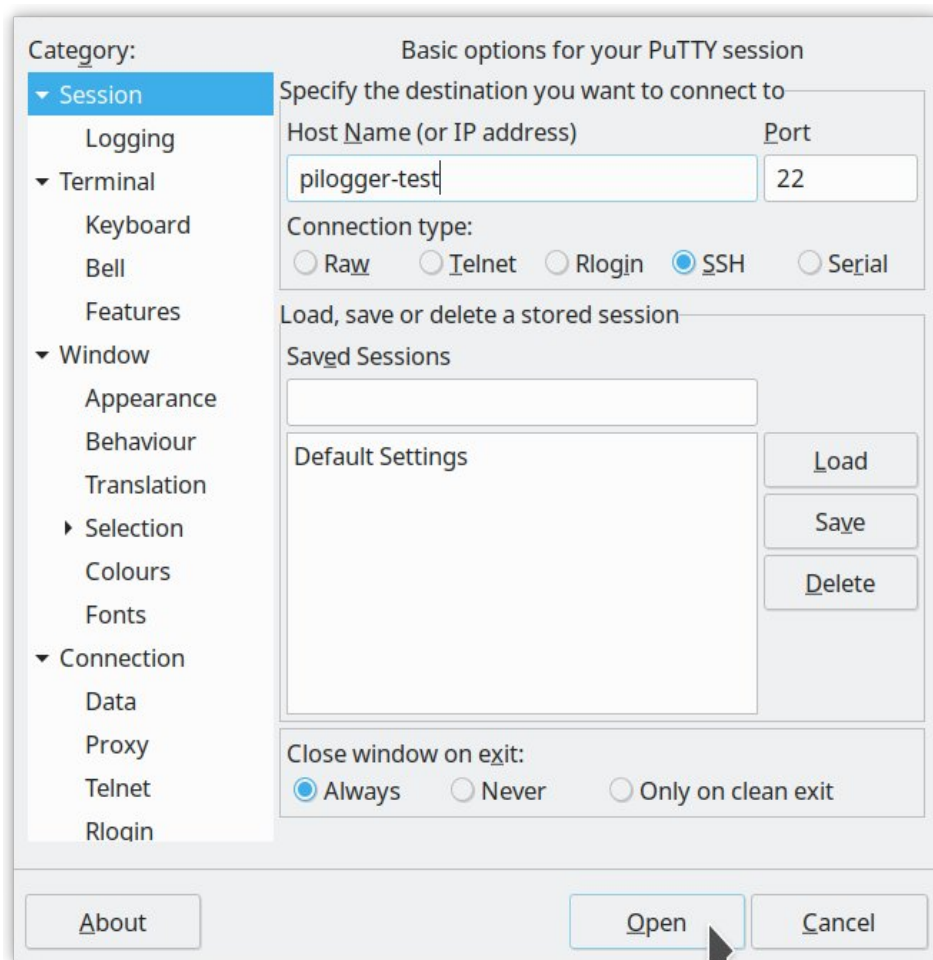
Now that the SD card has been inserted into the Raspberry Pi, it is time to switch on the power to watch the Raspberry Pi boot up for a while - the SD card LED must flash rapidly for some time.

The Raspberry now logs on to the WLAN router (or access point) the first time it boots up. It uses the network name that we specified in the special settings in the imager.

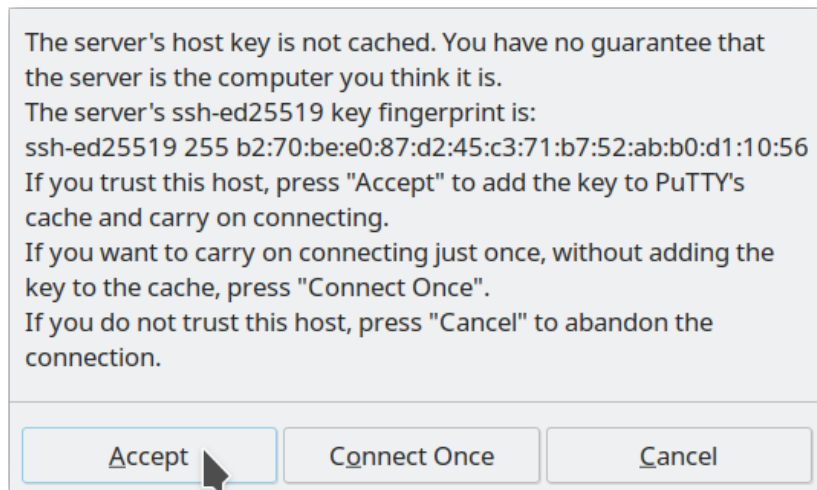
We now start an SSH-capable terminal program on our other computer - e.g. the well-known 'Putty'.

There we enter the network name and port 22 as the address for the Raspberry Pi to be contacted (now seen from here, the host computer).

The IP address that the router has assigned to the Raspberry is determined automatically. However, it makes sense to set in the router that the Raspberry should always receive exactly this address.



The Raspberry Pi now sends a certificate, which we now have to accept in Putty to make sure that the remote computer (in our case the Raspberry) is really the desired connection partner:



This request is like checking an ID card, a so-called authentication. It always appears when a new or changed SSH server key appears under a specific MAC address (network device interface ID - not to be confused with the IP address). This happens on first contact with a specific device or when the system has been reinstalled on this device - as in our case.

So we accept with 'Accept' and the connection is established. We now get a terminal window with command line input (CLI - command line interface).

Now we have to log in:

A screenshot of a terminal window with a black background and white text. The text shows the login process for a user named 'karl' on a system named 'pillogger-test'. Two red arrows point to the 'login as: karl' prompt and the password prompt. The terminal output includes the Linux version, system information, a disclaimer about Debian GNU/Linux software, and the last login time. The prompt 'karl@pillogger-test:~ \$' is shown at the bottom with a green cursor.

```
login as: karl
karl@pillogger-test's password:
Linux pillogger-test 5.15.61+ #1579 Fri Aug 26 11:08:59 BST 2022 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Sep 16 18:42:17 2022 from 192.168.178.29
karl@pillogger-test:~ $
```

Here, of course, we now use the user name and password that we entered during the SD card configuration.

4.1.6 Setting up the system

Normally, a fresh Raspberry Pi OS has to be configured once. However, because we have already done some of the configuration in the 'Raspberry Pi Imager', there is not so much left to do.

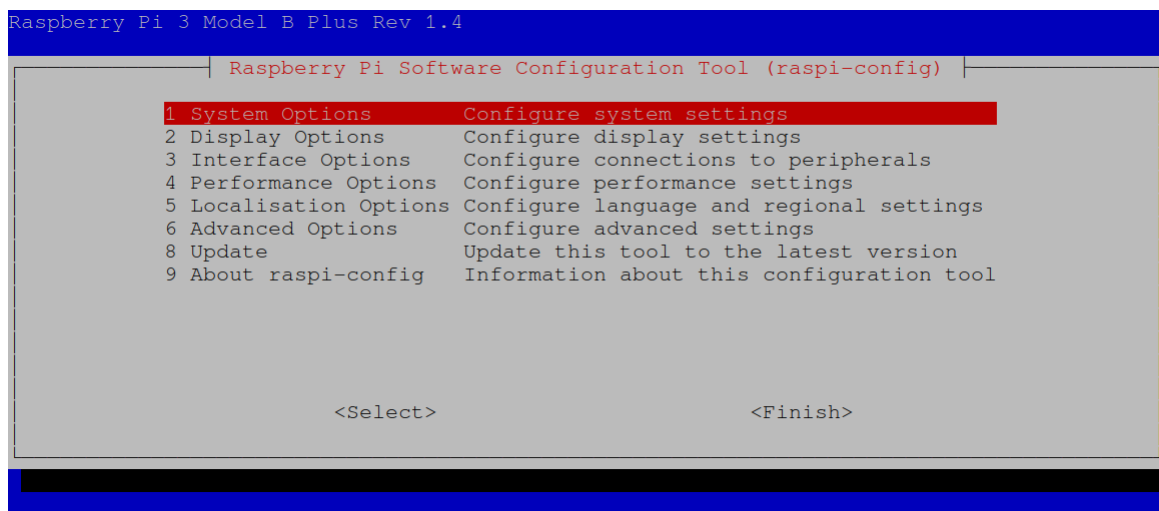
For example, the country setting has already been set in the special settings - but the Linux 'locale' has not been set completely correctly.

For the last necessary actions and to show the possibilities, we run the tool 'raspi-config'.

To do this, we enter the following in the command line:

```
sudo raspi-config
```

The program will start with this screen:



This tool is initially operated using the 'Down arrow' and 'Up arrow' buttons to navigate to the list items. The selected item can then be activated with the 'Enter' (or 'Return') key.

The option points <Select> and <Finish> can be selected with the 'Tab' key and then activated again with 'Enter'.

4.1.6.1 Updating Raspi-Config

First, let's take a detour to option 8 and try to update the tool itself.
To do this, we use the 'Down arrow' button to navigate to line 8:

```
Raspberry Pi 3 Model B Plus Rev 1.4

Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options      Configure system settings
2 Display Options     Configure display settings
3 Interface Options   Configure connections to peripherals
4 Performance Options Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options    Configure advanced settings
8 Update              Update this tool to the latest version
9 About raspi-config  Information about this configuration tool

<Select>                                <Finish>
```

A functioning Internet connection is required for this.

After pressing 'Enter', the display jumps back to the command line and the update is carried out:

```
Linux pilogger-test 6.6.20+rpt-rpi-v7 #1 SMP Raspbian 1:6.6.20-1+rpt1 (2024-03-07) armv7l

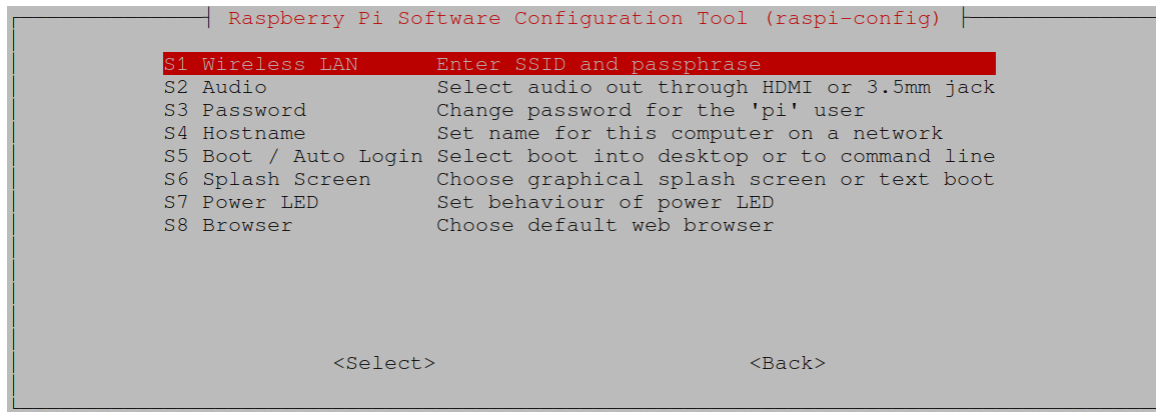
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
karl@pilogger-test:~$ sudo raspi-config
Get:1 http://archive.raspberrypi.com/debian bookworm InRelease [23.6 kB]
Get:2 http://raspbian.raspberrypi.com/raspbian bookworm InRelease [15.0 kB]
Get:3 http://archive.raspberrypi.com/debian bookworm/main arm64 Packages [384 kB]
Get:4 http://archive.raspberrypi.com/debian bookworm/main armhf Packages [394 kB]
Get:5 http://raspbian.raspberrypi.com/raspbian bookworm/main armhf Packages [14.5 MB]
Fetched 15.3 MB in 13s (1,217 kB/s)
Reading package lists... Done
W: http://raspbian.raspberrypi.com/raspbian/dists/bookworm/InRelease: Key is stored in legacy truste
d.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages will be upgraded:
  raspi-config
1 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
Need to get 30.8 kB of archives.
After this operation, 2,048 B of additional disk space will be used.
Get:1 http://archive.raspberrypi.com/debian bookworm/main armhf raspi-config all 20240424 [30.8 kB]
Fetched 30.8 kB in 0s (93.7 kB/s)
Reading changelogs... Done
(Reading database ... 62838 files and directories currently installed.)
Preparing to unpack .../raspi-config_20240424_all.deb ...
Unpacking raspi-config (20240424) over (20240313) ...
Setting up raspi-config (20240424) ...
Sleeping 5 seconds before reloading raspi-config
```

Once the update is complete, the script waits 5 seconds and restarts 'raspi-config' so that we are back in the main menu at point 1.

4.1.6.2 System Options

As we are now back at item 1 'System Options' in the main menu, we go to the submenu by pressing the 'Enter' key:

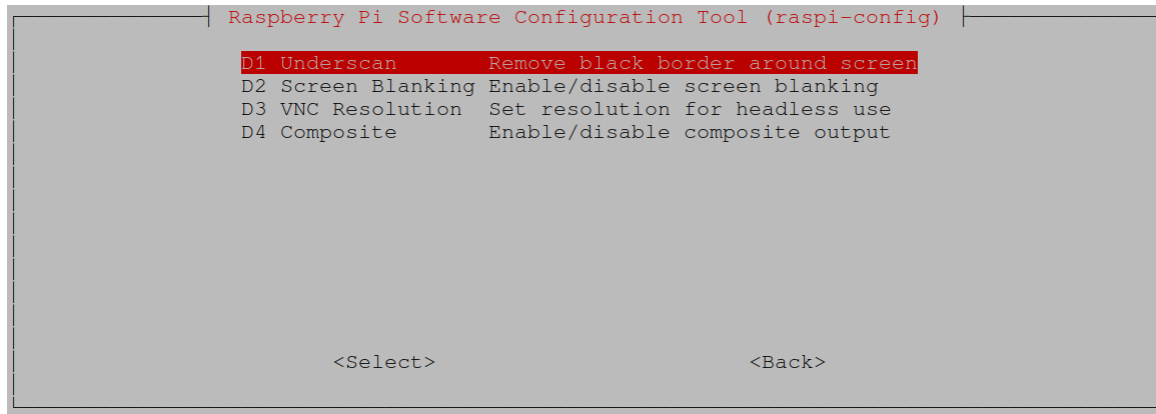


- S1 'Wireless LAN'
gives us the option to change the login data for the WLAN. As we have obviously logged in successfully, this is not necessary now.
- S2 'Audio'
there is the option to select the audio output for Raspberry Pi models with an analog audio socket (or via HDMI). This is not available on the Zero model.
- S3 'Password'
gives the option to change the password for the current user. Should not be necessary at the moment.
- S4 'Hostname'
is the network name of the Raspi. We had also just defined it.
- S5 'Boot / Auto Login'
is actually only interesting for systems with display and keyboard.
- S6 'Splash Screen'
is a setting option for systems with a display.
- S7 'Power LED'
allows you to change the behavior of the power LED on some Raspberry Pi models.
- S8 'Browser'
lets us choose whether we would prefer to set 'Firefox' as the default browser instead of 'Chromium'.

So a few interesting options that can be changed as desired, but do not need to be changed for the PiLogger WebMonitor now.

4.1.6.3 Display Options

In the main menu, we now select line 2 'Display Options' and get this submenu:



These options all relate to possible display settings.

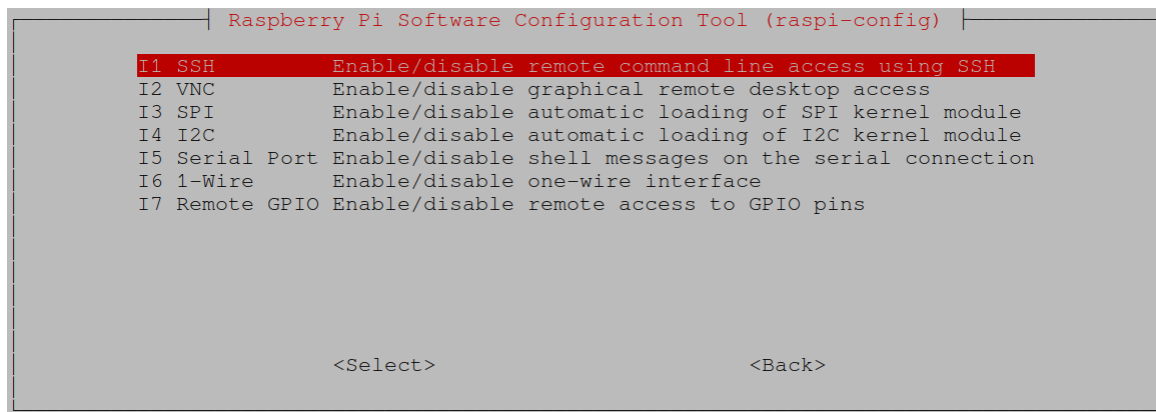
As we have set up a 'Lite' version (without desktop), these options are not relevant here.

With the 'Tab' key we go to <Back> and with 'Enter' we return to the main menu.

4.1.6.4 Interface Options

In the main menu, select item 3 'Interface Options'.

After confirming with the 'Enter' key, the submenu opens:

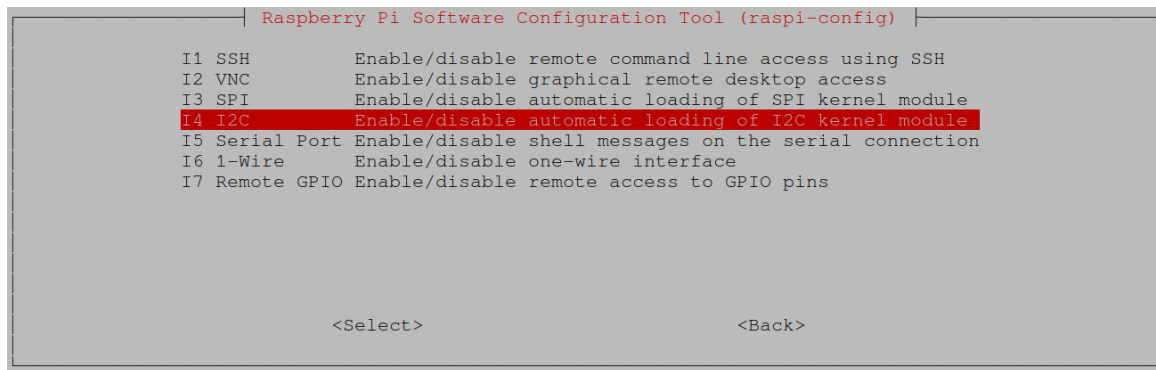


Of the many different options here, we are initially interested in two in the context of these instructions. The SSH server can be activated under I1 'SSH'. However, we have already done this successfully with the SD card preparation.

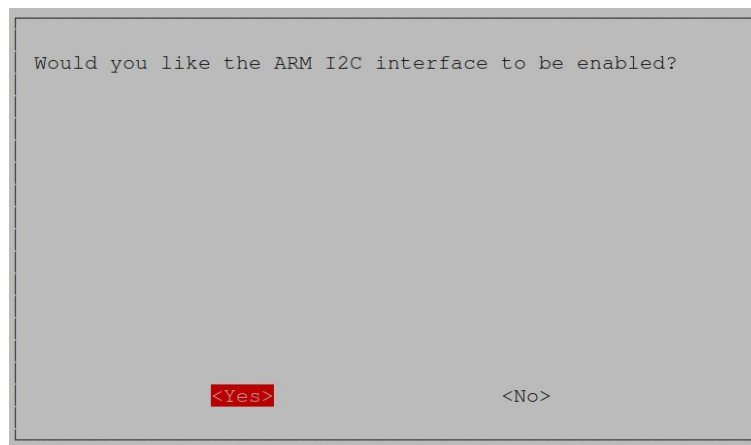
But we still have to activate the I²C interface with I4 'I2C'. This is necessary for the PiLogger.

4.1.6.4.1 I4 I2C

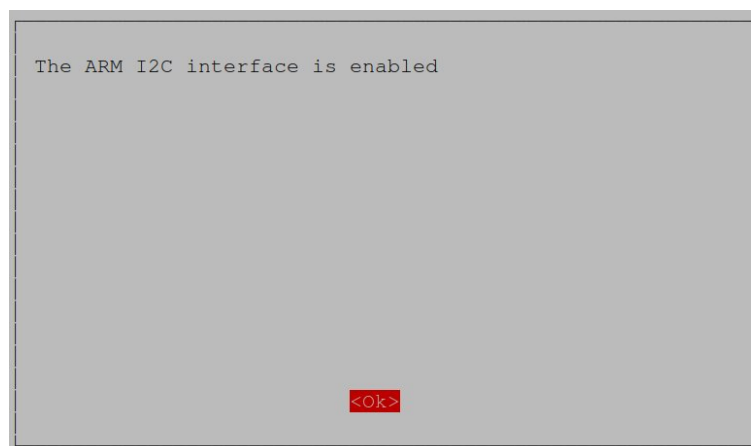
To activate the I²C interface, navigate to line I4:



After pressing 'Enter' we are asked whether we want to activate the 'ARM I2C interface':



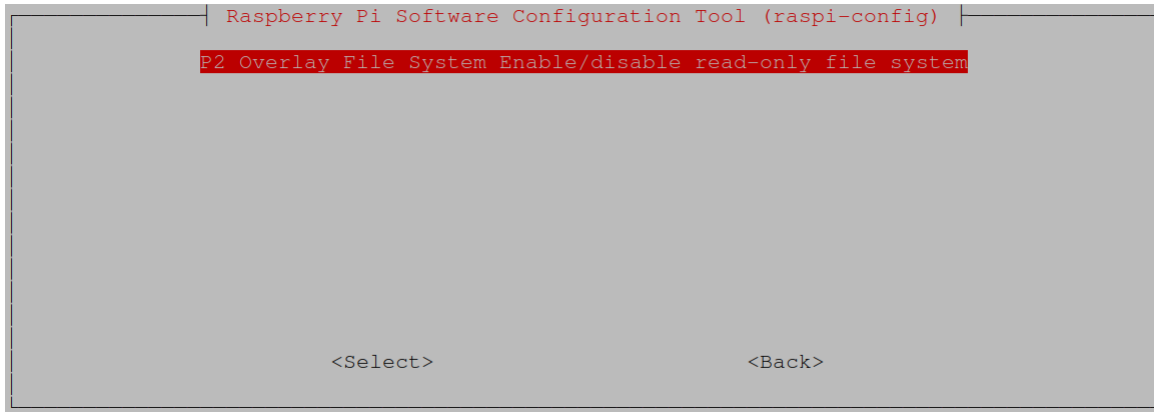
The default setting is <No> because this interface is deactivated by default. So we change to <Yes> with the 'Left arrow' key and press 'Enter'. We then get the confirmation message:



After pressing 'Enter', we are taken back to the main menu.

4.1.6.5 Performance Options

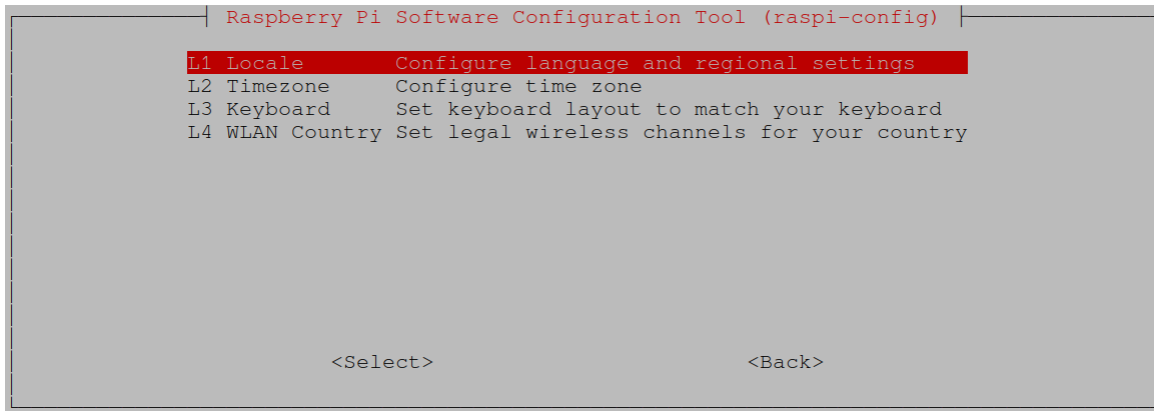
Settings for the performance of the Raspberry can be made under menu item 4. So we navigate to line 4 and press the 'Enter' key to get this submenu:



Depending on the model, this menu enables various settings for the Raspberry Pi relating to 'Performance'. In fact, we have nothing to do here at the moment. Press 'Tab' twice to jump to <Back> and press 'Enter' to return to the main menu.

4.1.6.6 Localisation Options

We now go to line 5 'Localization Options'. After selecting with the 'Enter' key, we get this submenu:



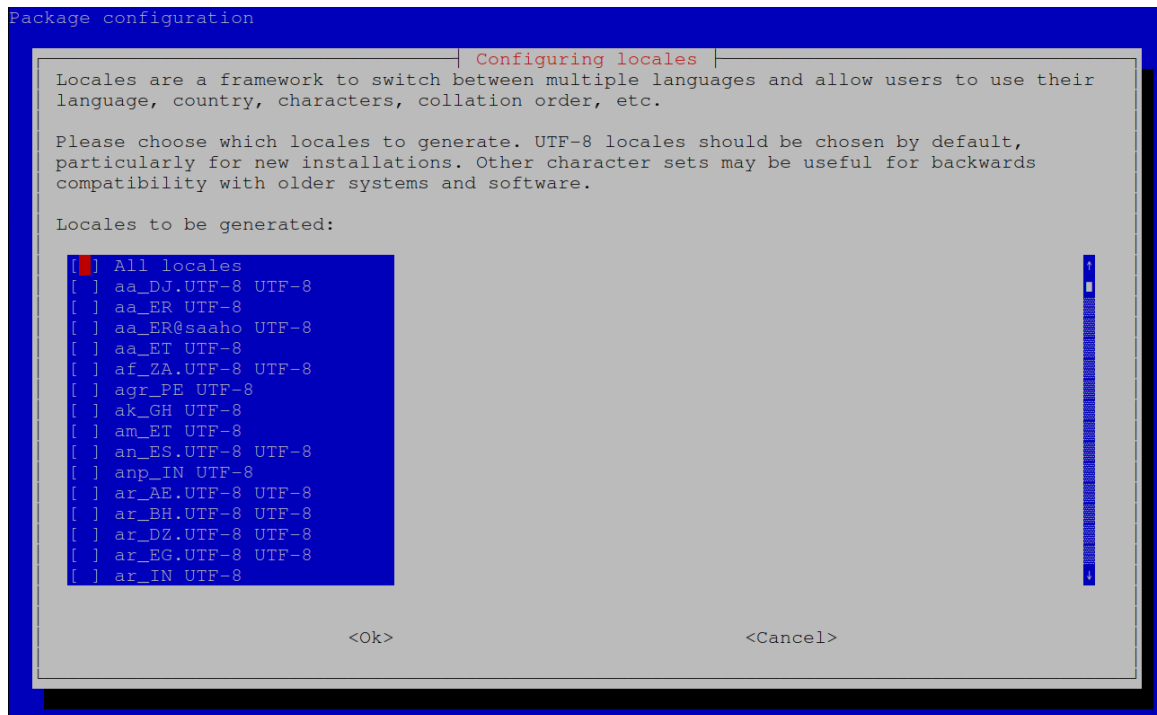
As mentioned above, the country selection in the 'Imager' does not work completely. However, the items 'Timezone' and 'WLAN Country' are completed. So we have to take care of L1 and L3 here.

4.1.6.6.1 L1 Locale

Option L1 allows you to set up the system's country-specific settings, i.e. select the character set, language and display formats such as date and time. These country settings are called 'Locales'.

By default, only English and a basic set are set up.

After pressing 'Enter', we are offered a long list of possible country settings to choose from:

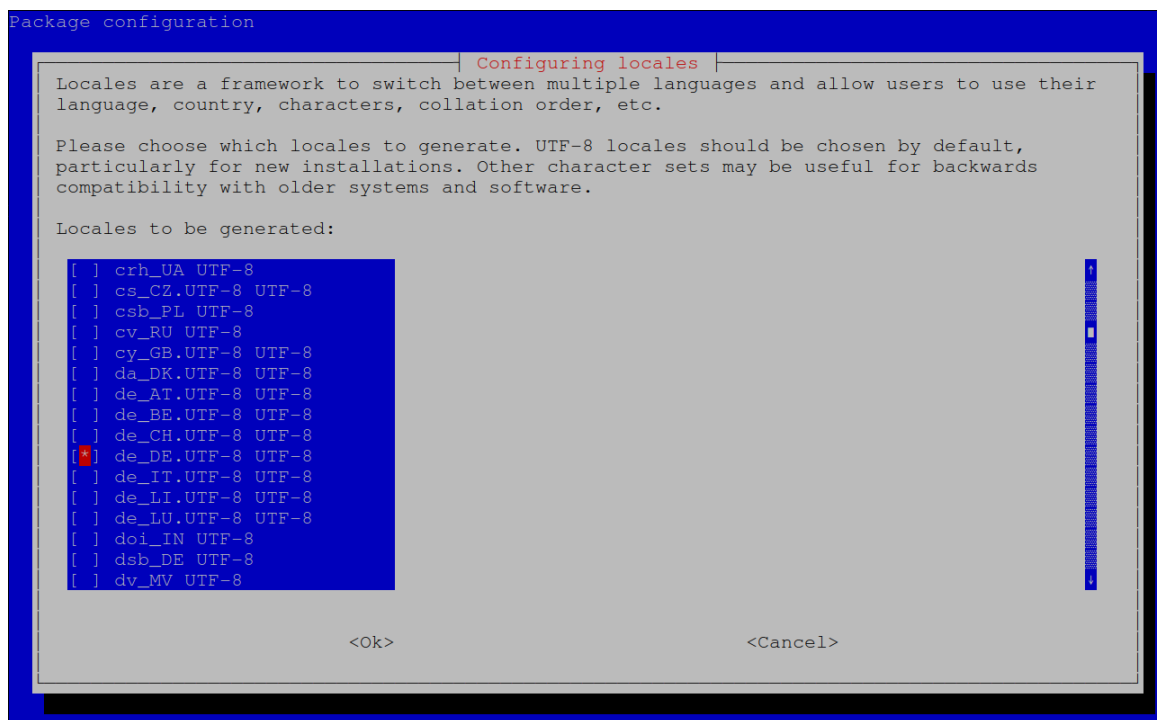


In principle, either all (All locales) or individual locales can be selected for further setup. Selecting all would not only take a long time to set up, it would also tie up resources unnecessarily during operation and slow down some processes.

We recommend adding exactly one additional country setting here - in our case 'de_DE.UTF-8 UTF-8'.

This is the language 'German' for the country 'Germany' with the character set encoding UTF-8.

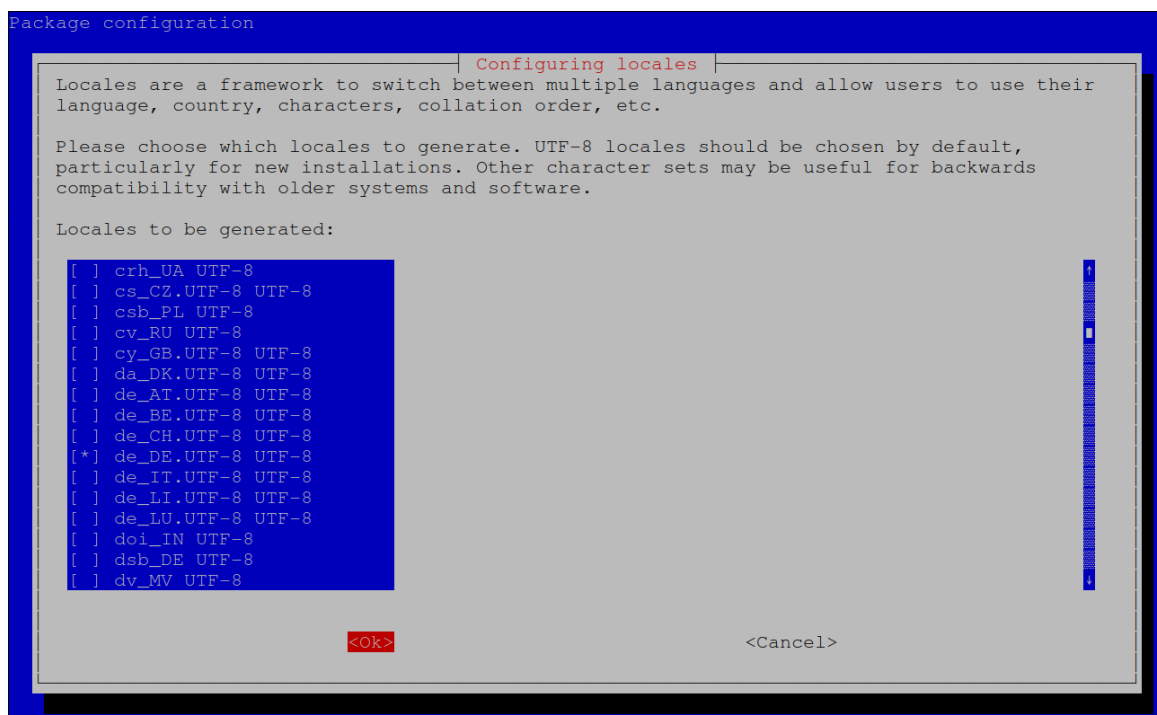
We navigate to this entry with the 'down arrow' button:



The selection is made here with the space bar, whereupon the entry receives an asterisk for the status 'selected':

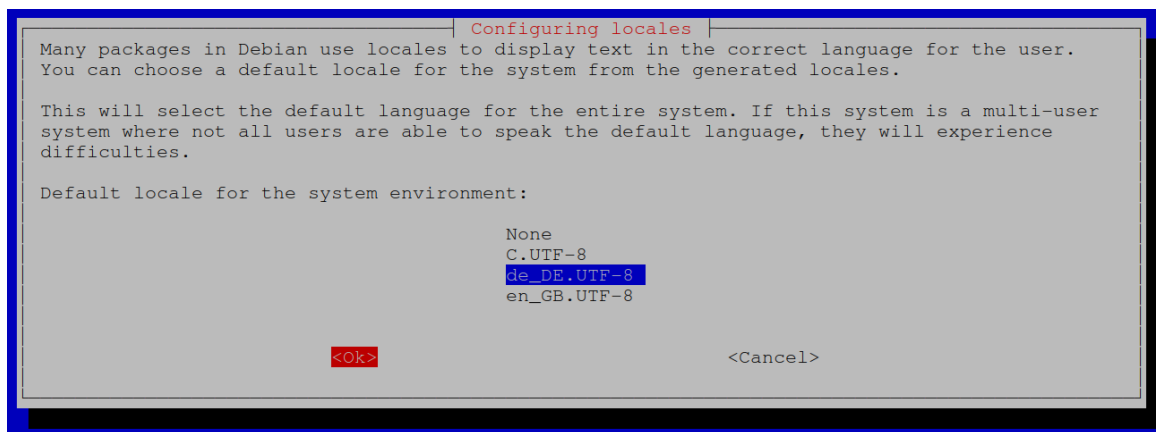
In the same way, further country records can now be selected if required. As already mentioned, this is not necessary in our case.

We now use the **'Tab'** key to go to the <Ok> action field, which is highlighted in red:



This entry is now completed by pressing 'Enter'.

The next window now prompts us to select the preferred country setting from those activated:



So we select 'de_DE.UTF-8' with the 'down arrow' key, go to <Ok> with the 'right arrow' key and press 'Enter'.

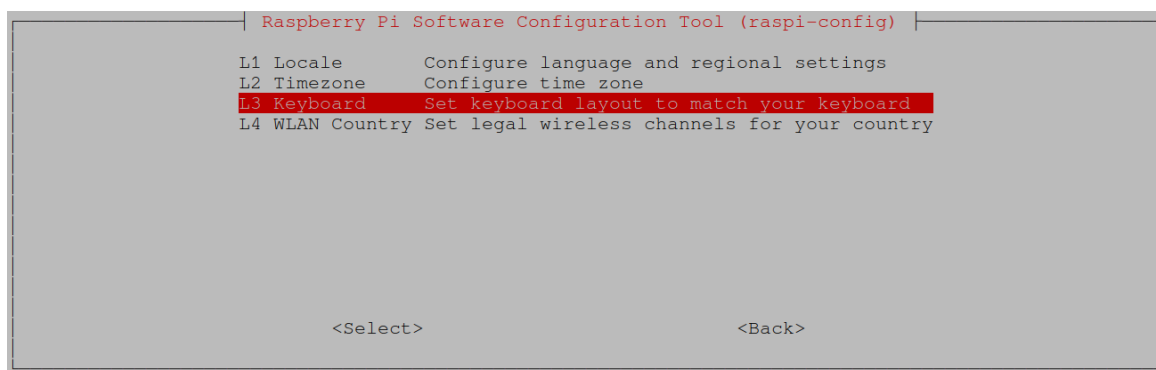
Now the display switches back to the console window and processing, i.e. setting up, takes a little while.

After that, the display jumps back to the main menu of 'Raspi-Config'.

4.1.6.6.2 L3 Keyboard

We therefore go back to menu 5 'Localization Options'.

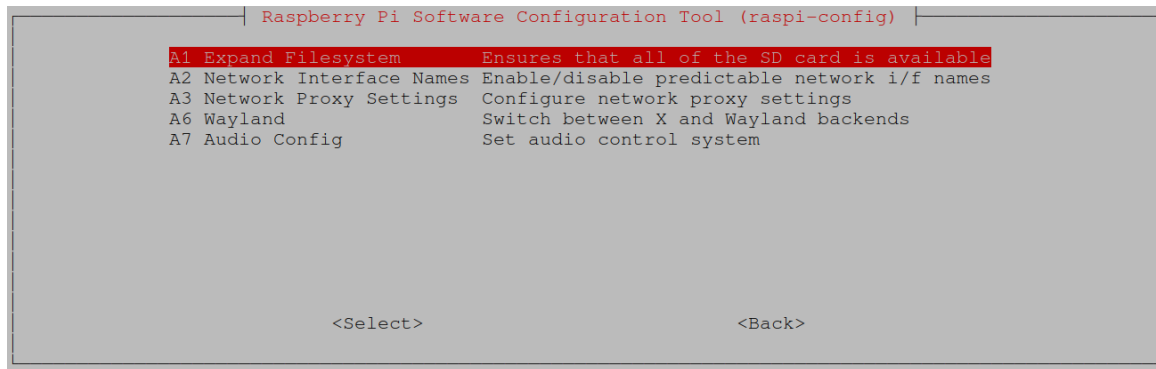
With the menu item L3, the keyboard layout is adapted to the country settings made so far. To do this, we need to execute this option once. We navigate to the L3 item as usual:



We start this action by pressing the 'Enter' key and are switched to the console display, where the message 'Reloading keymap. This may take a short while' asks for a little patience. We are then returned to the main menu.

4.1.6.7 Advanced Options

Even if we currently have no need to do anything in this submenu, let's take a look at the options - for information:



The item A1 'Expand Filesystem' was useful with older Raspi OS - but should now have been done automatically during the first boot process.

The other options deal with desktop graphics and network settings, which can be important in special cases.

We are finished here and switch to <Back> with 'Tab' and return to the main menu with 'Enter'.

4.1.6.8 Finishing Raspi-Config

Now that we have made all the necessary settings, we can exit the configuration program. To do this, we use the 'Right arrow' button to switch to the <Finish> action field and press 'Enter'.

For further actions, it makes sense to perform a restart.

If a restart is not offered by Raspi-Config itself due to the changes made, we now do this directly ourselves.

So we enter in the console window:

```
sudo reboot now
```

We conclude the entry with 'Enter'. As the Raspberry is now shutting down to restart, the connection to Putty is disconnected.

We now have to wait a while until the Raspberry has restarted and can then reconnect using Putty.

4.1.7 General Update

To complete the new Raspberry Pi OS installation, a normal update should be carried out. Depending on how old the image is, further improvements have been made in the meantime, some of which may also be security-relevant. These will be distributed as quickly as possible, i.e. also outside the series of complete installation images.

The following actions require a functioning Internet connection.

4.1.7.1 Firmware Update

A firmware update is **no longer recommended** for the newer versions of Raspberry Pi OS.

The firmware update should only be carried out in exceptional cases:

```
karl@pilogger-test:~ $ sudo rpi-update
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
*** Performing self-update
*** Relaunching after update
*** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
FW_REV:b57c32edddd41eb24a903eefc1c47e7e45fd9c5d
BOOTLOADER_REV:2bfd7cb74e6bc16559e040d0f5d788a4411819e4
*** We're running for the first time
*** Backing up files (this will take a few minutes)
*** Backing up firmware
*** Backing up modules 6.6.20+rpt-rpi-v7
WANT_32BIT:1 WANT_64BIT:1 WANT_PI4:1 WANT_PI5:0
#####
WARNING: This update bumps to rpi-6.6.y linux tree
See: https://forums.raspberrypi.com/viewtopic.php?p=2191175

'rpi-update' should only be used if there is a specific
reason to do so - for example, a request by a Raspberry Pi
engineer or if you want to help the testing effort
and are comfortable with restoring if there are regressions.

DO NOT use 'rpi-update' as part of a regular update process.
#####
Would you like to proceed? (y/N) █
```

In fact, the firmware is now also updated during a distribution update, if necessary. We will therefore skip this step and continue with the distribution update.

4.1.7.2 Distribution Update

A general upgrade of the Raspberry Pi OS distribution can now be carried out. To do this, the local database of installed and available software packages must be refreshed.

So we enter the following line:

```
sudo apt-get update
```

Here the tool 'apt-get' is called up with 'super user' rights and instructed to update the package database. This process does not take long.

The output looks something like this:

```
login as: karl
karl@192.168.178.60's password:
Linux pilogger-test 6.6.20+rpt-rpi-v7 #1 SMP Raspbian 1:6.6.20-1+rpt1 (2024-03-07) armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 28 23:42:39 2024 from 192.168.178.29
karl@pilogger-test:~$ sudo apt-get update
OK:1 http://raspbrian.raspberrypi.com/raspbian bookworm InRelease
OK:2 http://archive.raspberrypi.com/debian bookworm InRelease
Paketlisten werden gelesen... Fertig
W: http://raspbrian.raspberrypi.com/raspbian/dists/bookworm/InRelease: Schlüssel ist im ver-
alteten Schlüsselbund trusted.gpg gespeichert (/etc/apt/trusted.gpg), siehe den Abschnitt
MISSBILLIGUNG in apt-key(8) für Details.
karl@pilogger-test:~$
```

Now we enter the following line of text:

```
sudo apt-get dist-upgrade
```

This will now start the actual update of the Raspberry Pi OS installation. First, it is determined which packages need to be changed. Then it asks whether these changes should actually be installed:

```
karl@pilogger-test:~$ sudo apt-get dist-upgrade
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Paketaktualisierung (Upgrade) wird berechnet... Fertig
Die folgenden NEUEN Pakete werden installiert:
  linux-headers-6.6.28+rpt-common-rpi linux-headers-6.6.28+rpt-rpi-v6
  linux-headers-6.6.28+rpt-rpi-v7 linux-headers-6.6.28+rpt-rpi-v7l
  linux-image-6.6.28+rpt-rpi-v6 linux-image-6.6.28+rpt-rpi-v7
  linux-image-6.6.28+rpt-rpi-v7l linux-image-6.6.28+rpt-rpi-v8:arm64
  linux-kbuild-6.6.28+rpt python3-pycryptodome
Die folgenden Pakete werden aktualisiert (Upgrade):
  bsdxtrutils bsdutils eject fdisk libblkid1 libcamera-ipa libcamera0.2 libfdisk1
  libmount1 libpisp-common libpisp1 libsmartcols1 libuuid1 linux-headers-rpi-v6
  linux-headers-rpi-v7 linux-headers-rpi-v7l linux-image-rpi-v6 linux-image-rpi-v7
  linux-image-rpi-v7l linux-image-rpi-v8:arm64 linux-libc-dev mount pi-bluetooth
  raspi-firmware raspi-utils rfidkill rpi-eeeprom rpicas-apps-lite util-linux
  util-linux-extra
30 aktualisiert, 10 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen 139 MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 201 MB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n] J
```

After entering 'J' and ending with 'return', the main part of the update begins. Depending on how fresh the Raspberry Pi OS image used is, several packages may have to be updated. This can take up to an hour. The example shown is made with a 1 month old image, with a relatively manageable scope (30 packages to be updated).

The update is complete when the command line prompt appears again. It should now be restarted once:

```
sudo reboot
```

The Raspberry Pi is now basically set up.

4.2 Assembly and Commissioning

The PiLogger One was originally developed to fit a Raspberry Pi 1. It therefore has a 26-pin socket connector and fits next to the Raspberry Pi 1's RCA socket for audio. The newer Raspberry Pi then received a 40-pin pin header for the GPIOs. As the pins required by the PiLogger One are still available in a compatible configuration, the PiLogger One can also be used with these newer Raspberry Pi's.

A variant of the PiLogger One has recently become available that does not use the 26-pin female connector and instead has a 5-pin male connector. The connection to the Raspberry Pi is then made via a short 5-pin Dupont cable (female-female). This variant is called 'PiLogger One Type B'.

We therefore refer to the original variant as 'PiLogger One Type A' below - for better differentiation.

4.2.1 Assembly of the PiLogger One Type A

4.2.1.1 Preparation

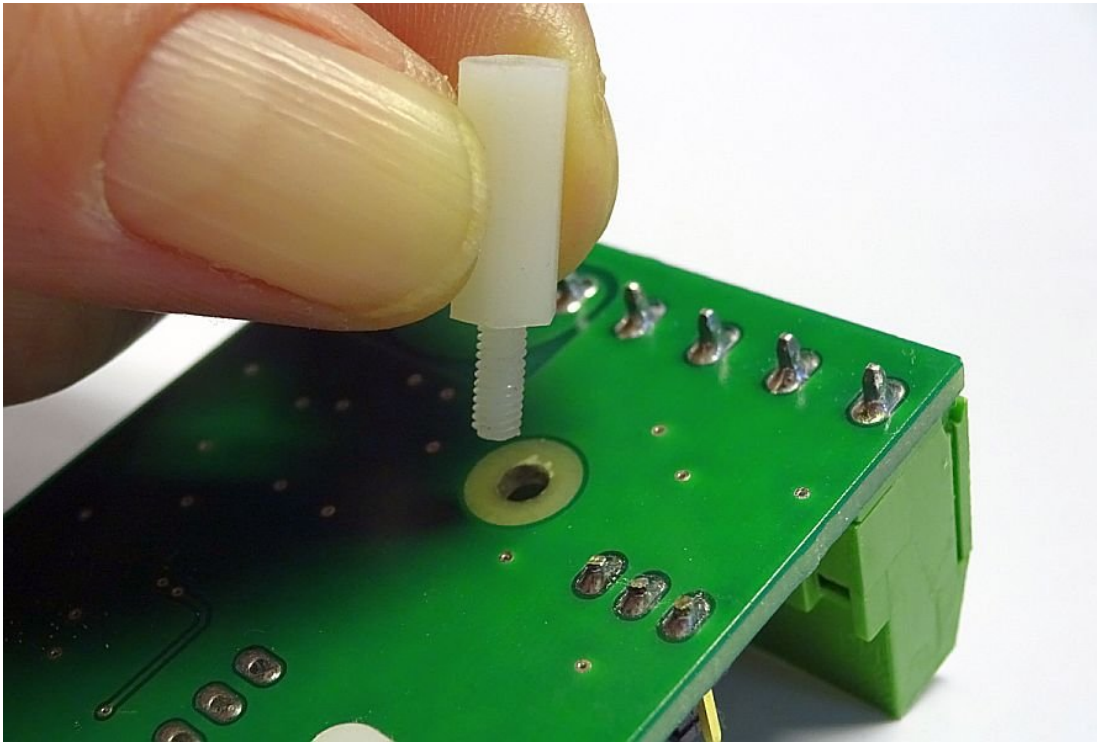
The PiLogger One Type A module can be used for the Raspberry Pi 1 with a 26-pin connector as well as for the newer Raspberry Pi's with the 40-pin connector.

When used with a Raspberry Pi with a 26-pin pin header, only the screw-on point near the screw terminal strip can be used; the two protruding corners with screw-on holes are not used here.

For Raspberrys with a 40-pin header, all 3 mounting holes should be used.

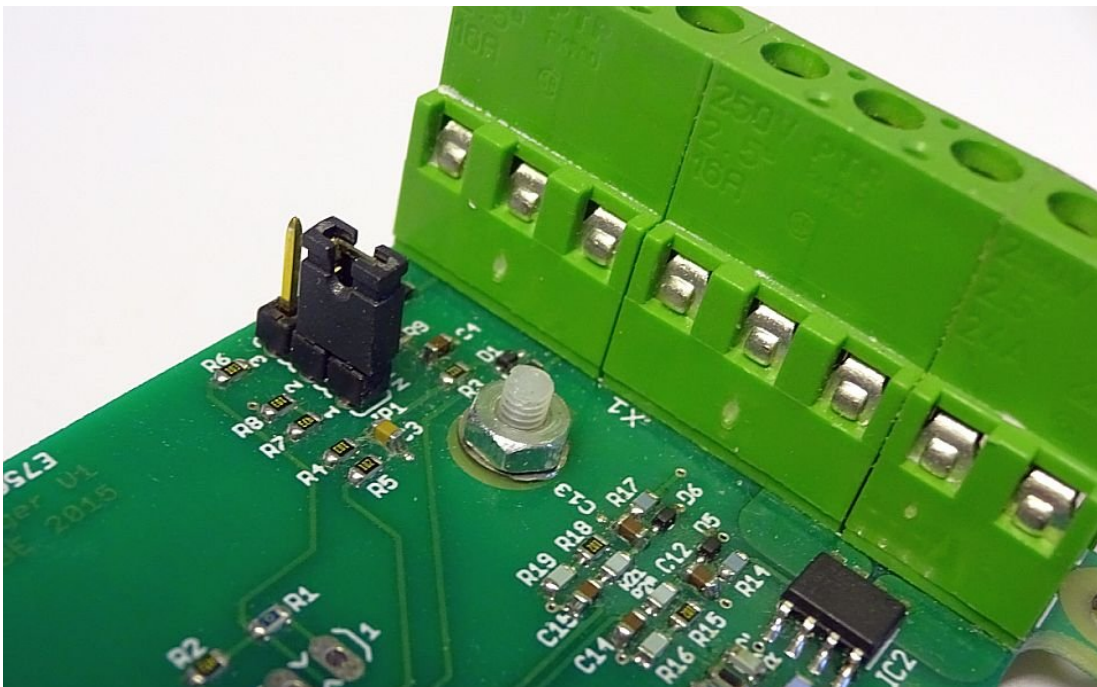
Although there is no suitable hole for the middle spacer on the Raspberry Pi, there is no component on the top that could be damaged at this point. This means that the third nylon bolt can be well supported there. *(To be on the safe side, please check the actual Pi model again!)*

The enclosed spacer bolts are now inserted into these holes in the PiLogger One. The threaded end is inserted directly into the hole in the PiLogger module from below:

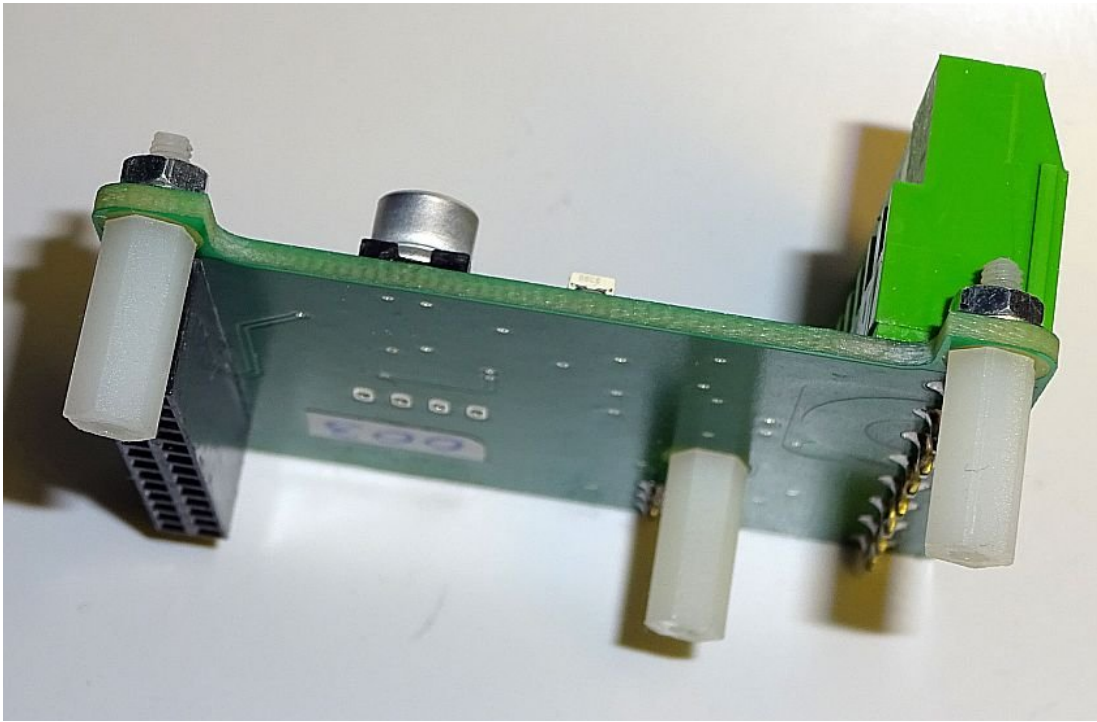


The bolt is then held in the hole and turned over with the module. Now first place a washer on the threaded pin, then a hexagon nut, which is initially screwed on loosely until the bolt no longer wobbles but can still be moved.

For the middle hole, it looks like this from above:



And for all 3 from the side like this:



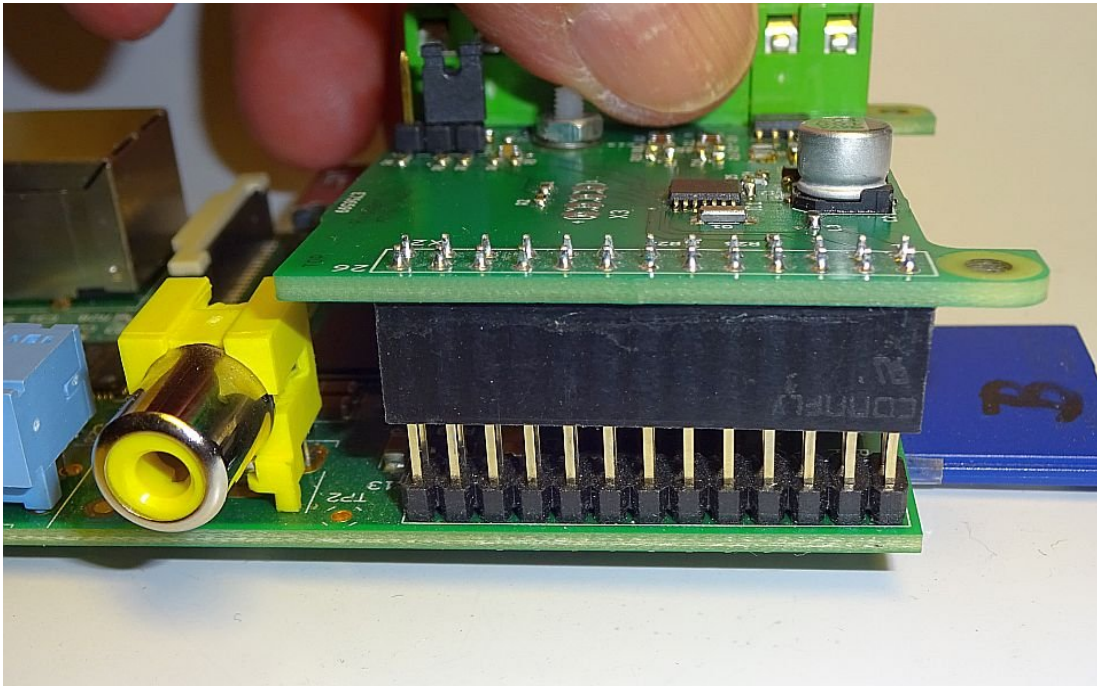
The module can now be attached.

4.2.1.2 Attaching the Module

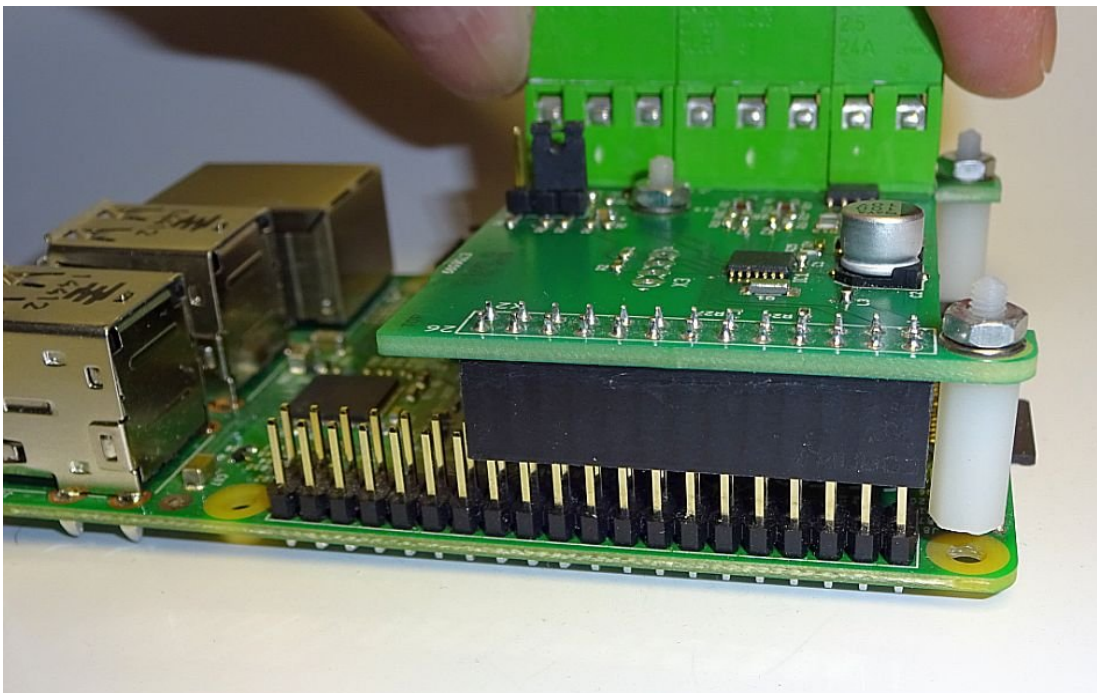
The PiLogger module is now placed on the Raspberry Pi. To do this, at least the power supply must be disconnected - it is better to remove all cables and connected peripherals for this purpose.

The 26-pin socket strip must be placed exactly on the pins of the Raspberry Pi so that the first pins on the outer corner of the Raspberry Pi are aligned with the first sockets on the corner with the mounting hole of the PiLogger (1 on 1).

Here first for a Raspberry Pi 1:

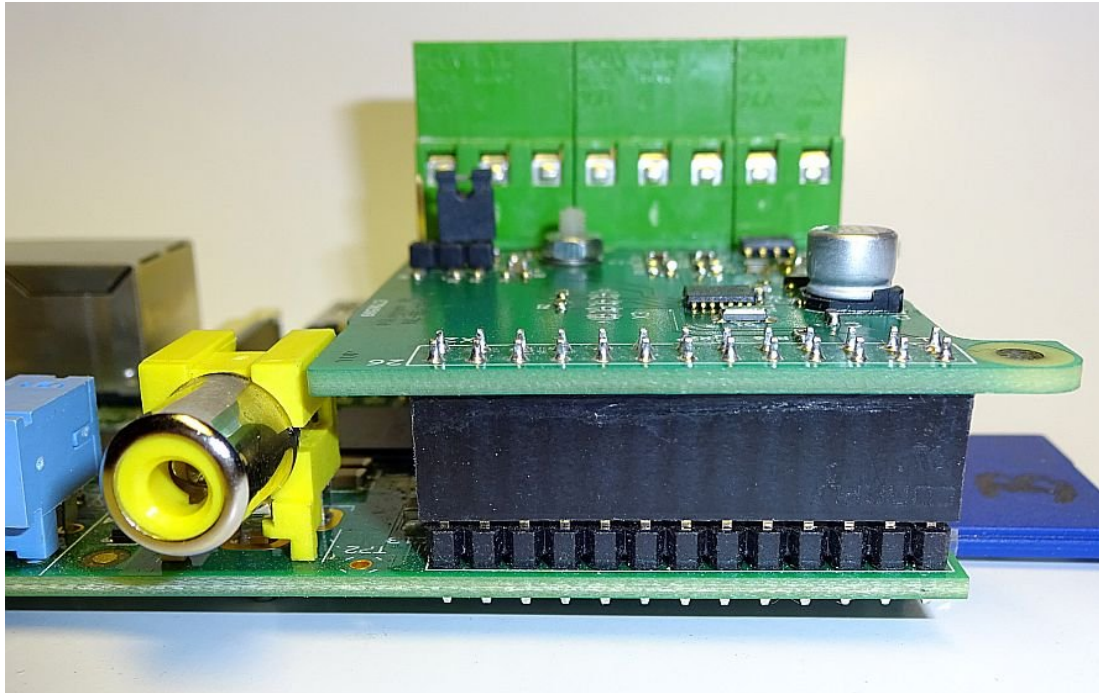


And here for a Raspberry Pi 2 B:

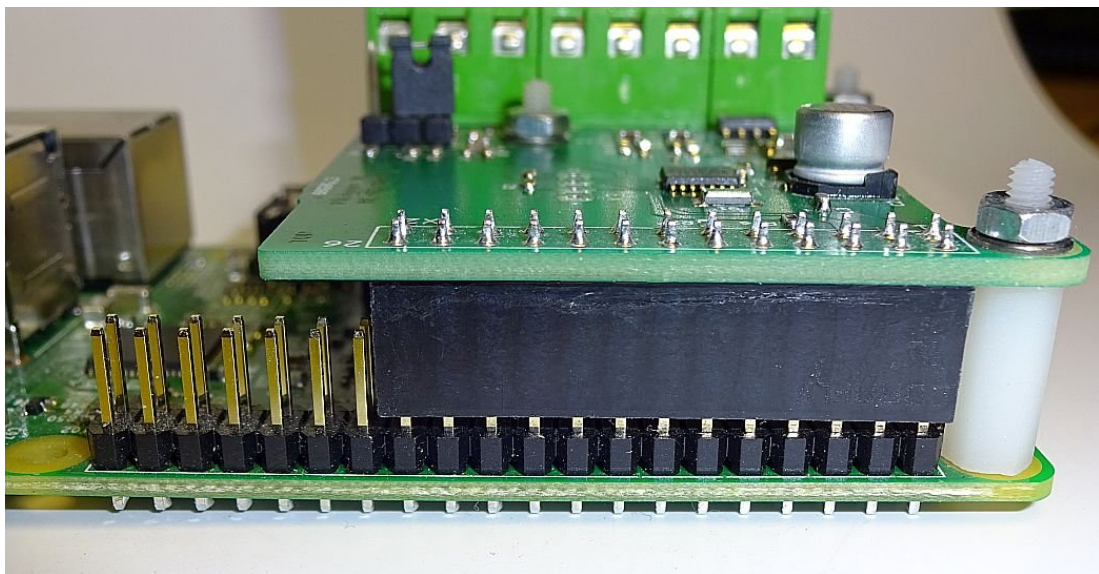


Now the PiLogger circuit board must be carefully pressed down evenly until the spacer bolts rest on the Raspberry Pi.

This is what it looks like on the Raspberry Pi 1:



And with the Raspberry Pi 2 B it looks like this:

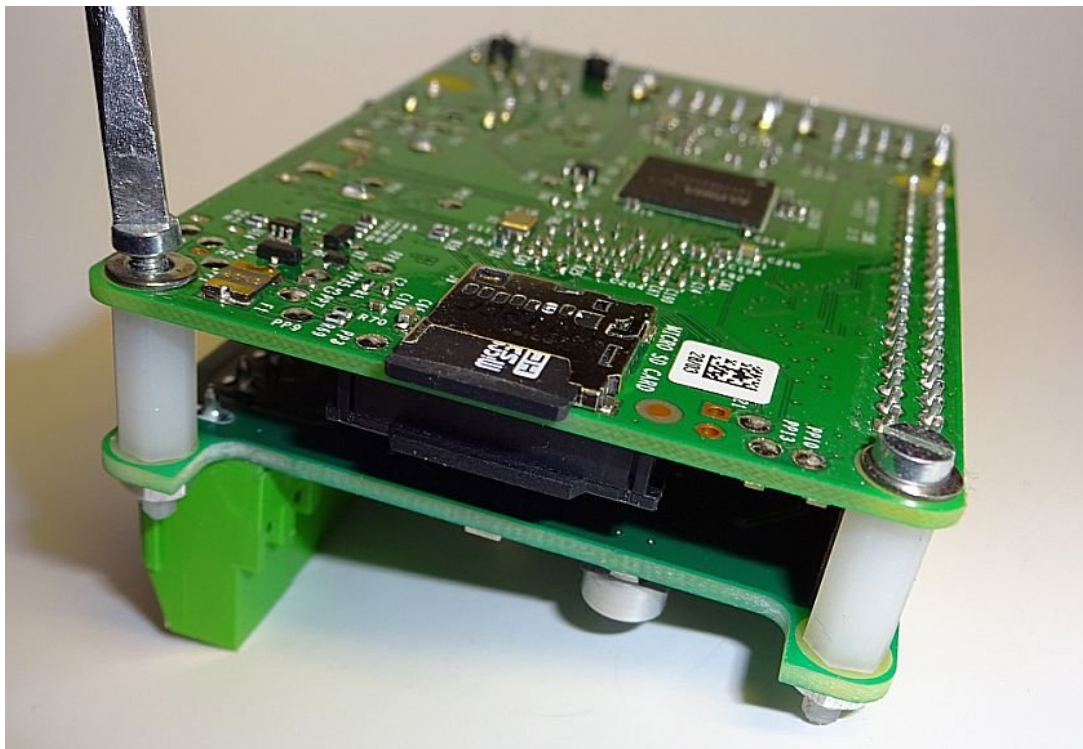


This means that the female connector does not sit completely on the base of the male connector. However, the contacts are well engaged.

Now we turn the double pack upside down. On the underside of the Raspberry Pi, place a washer on each hole with a spacer bolt behind it.



We then loosely screw in one cylinder head screw at a time so that both of the two spacer bolts are properly aligned - without bending. We then tighten them one after the other with a screwdriver at hand temperature:

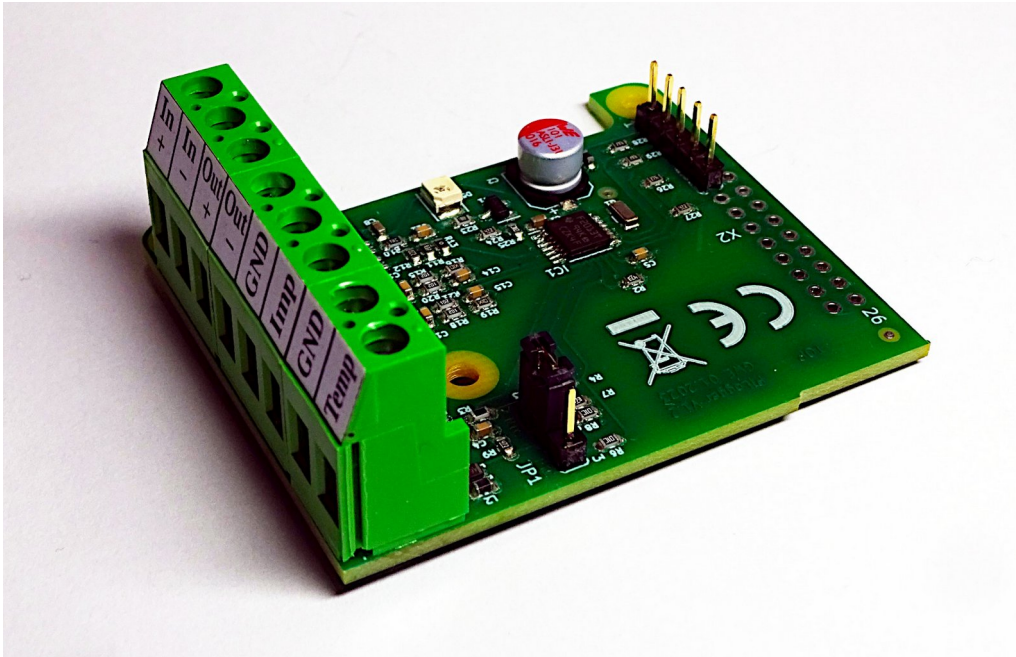


The PiLogger module is now installed.

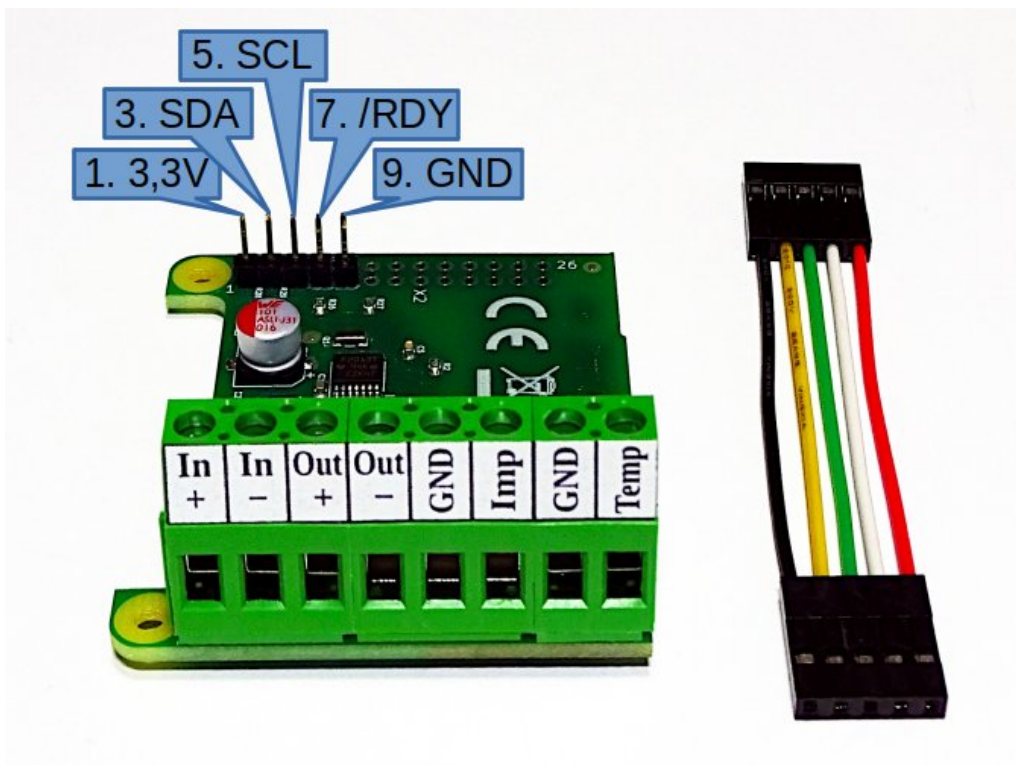
The double pack can now be placed back on the table as normal.

4.2.2 Connecting the PiLogger One Type B

The PiLogger One Type B looks like this:



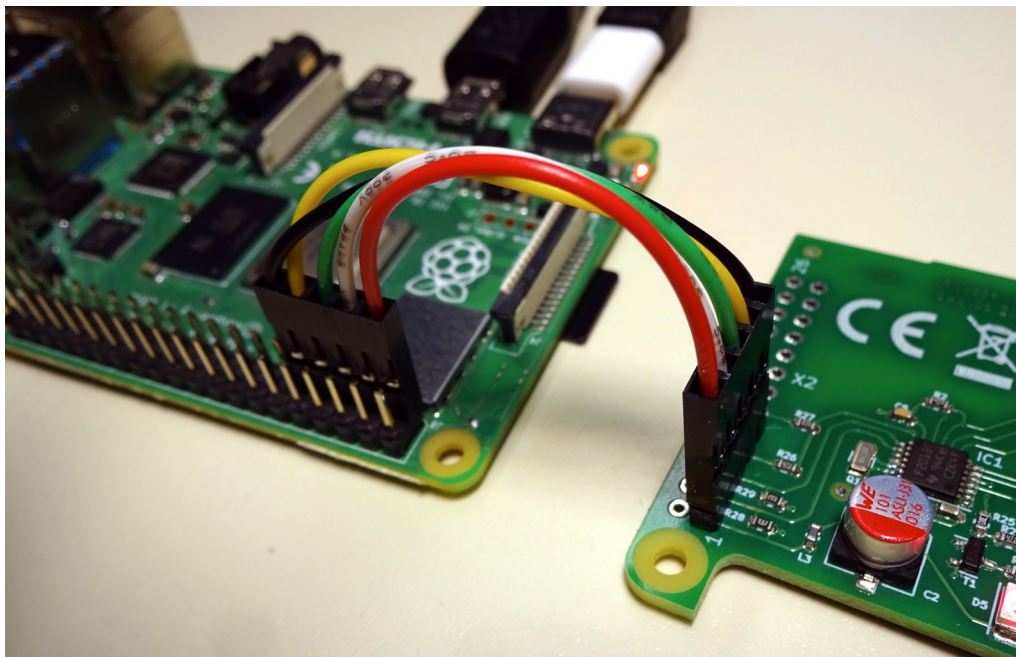
It is supplied without spacer bolts, but with a short connection cable.
The 5-pin pin header is assigned as follows:



We recommend using the cable with the red wire for the 3.3 V supply voltage and the black wire for GND (ground).

We therefore plug one female connector of the cable into the pin header of the PiLogger and the other into pins 1, 3, 5, 7 and 9 of the Raspberry Pi pin header.

With a Raspberry Pi 4, the whole thing looks like this:



Attention : Pay attention to the correct pin assignment !
So pin 1 to pin 1 (red cable, inner row of pins).

4.2.3 Switching on for the first time

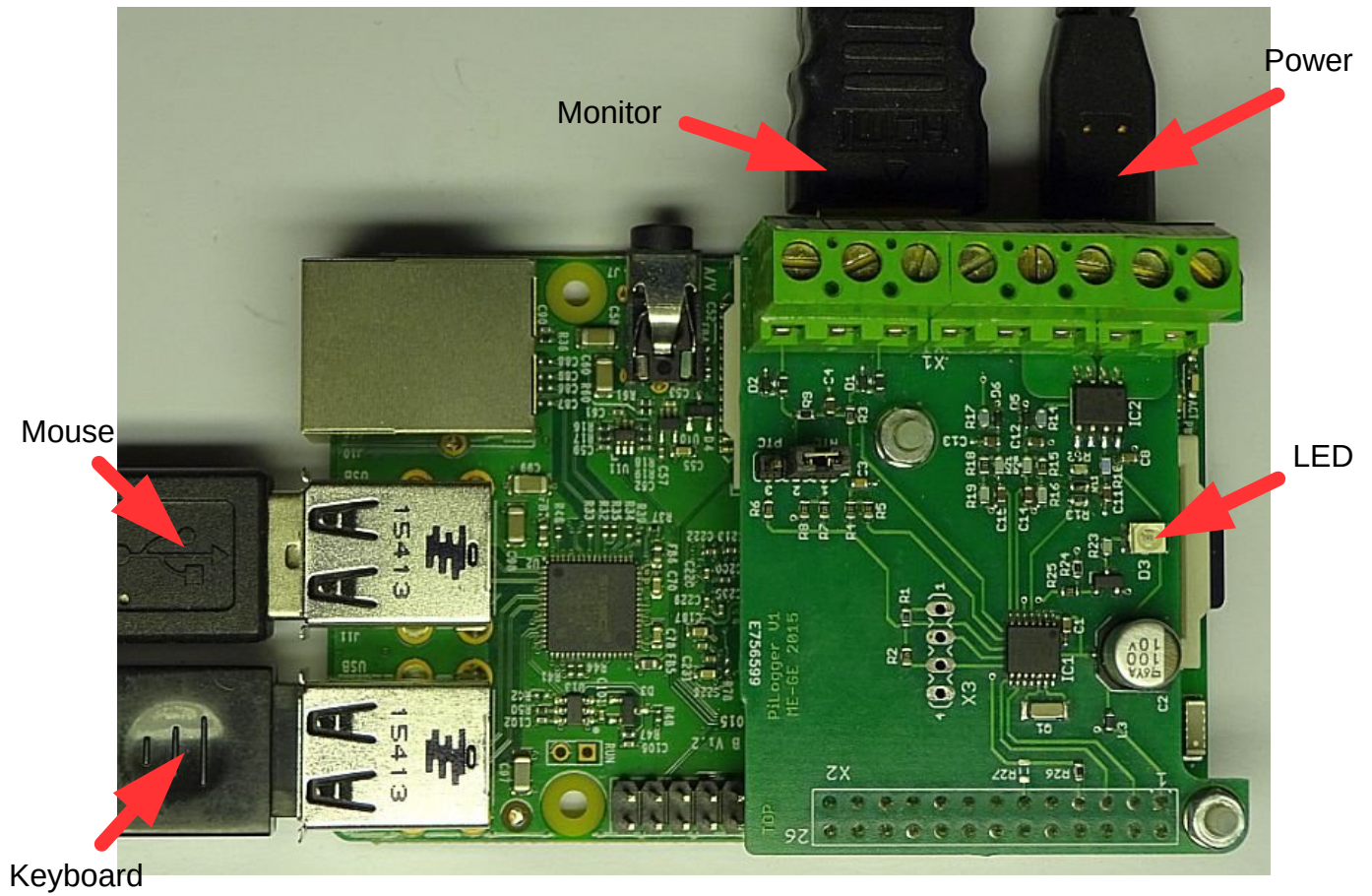
Before switching on the Raspberry Pi with PiLogger for the first time, all required cables and peripherals such as monitor and keyboard must of course be connected.

Finally, we plug the power supply unit into the socket - and in addition to the normal booting of the Raspberry Pi, the PiLogger starts immediately with a short flash of its LED once per second.

This is the factory default setting and shows that it is working correctly!

Whenever the Raspberry Pi is supplied with power, the PiLogger works autonomously with its stored parameters.

Here is an example of the PiLogger One Type A on a Raspberry Pi 3 :



4.3 Installing the PiLogger WebMonitor

The installation of the PiLogger WebMonitor is carried out by an interactive installation program (shell script) - the 'Installer'.

The installer can simply be started with a line in the SSH console as described below. As this is an executable file that is downloaded from the Internet, you can also download the installer separately from our server, inspect it and start it manually:

<https://www.pillogger.de/index.php/de/download-de/category/2-software>

→ Item : Installer PiLo-WebMon

The installer carries out all the necessary steps and offers optional settings.

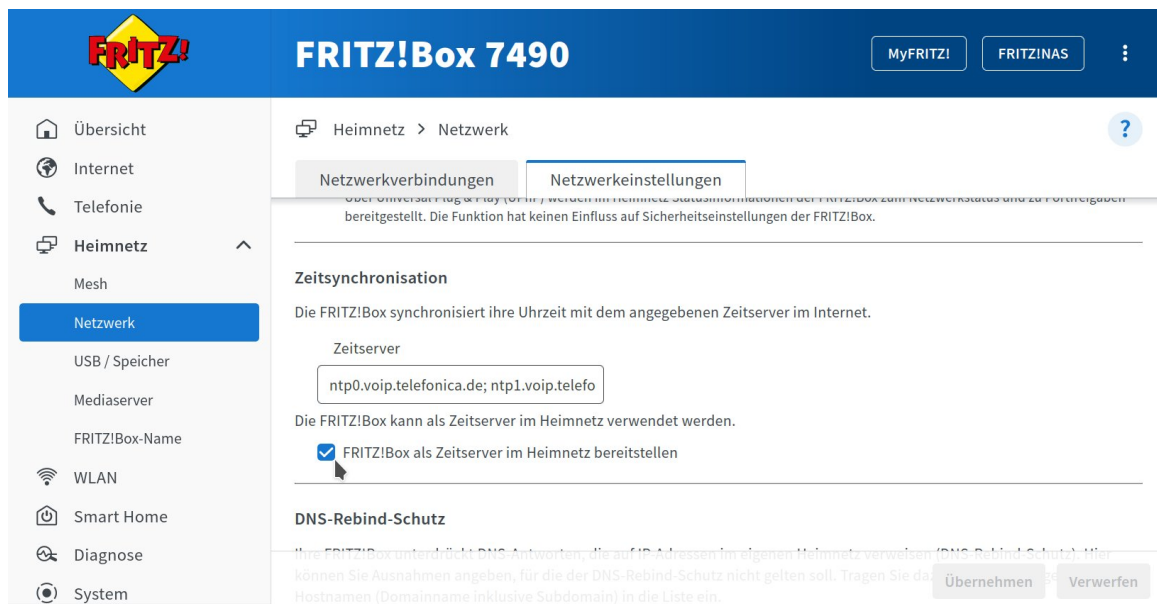
This also includes the option of redirecting the time synchronization with an Internet time server to your own home network router. This time synchronization is necessary because the Raspberry Pi does not have its own real-time clock (RTC). However, this is the only external Internet connection that is required and thus prevents the Raspberry Pi from being blocked from accessing the public Internet in the router. If your own router offers the option of providing an NTP server in the home network, this connection can also be kept internal and Internet access for the Raspberry can be blocked. This optional preparation is therefore described next.

4.3.1 Setting up the router as an NTP server

Internet routers synchronize their own time with NTP servers on the Internet. With VOIP telephony, the provider usually sets its own time server in the router. With most WLAN routers, you can also set the router itself to be available as a time server in your own network.

As an example, we show the settings page of a Fritz box. Log in to the router's configuration page and navigate to the 'Network' submenu via the 'Home network' menu item. There you switch to the second 'tab' (index card) 'Network settings'. Here, quite far down, there is the option to expand 'Further settings'. There you will find the section 'Time synchronization' and the setting option "Provide Fritz! Box as time server in the home network".

We activate this option and then press the 'Apply' button at the bottom:



That's it.

4.3.2 Running the Installer

To start the installation, we connect to the Raspberry using the SSH terminal program (here 'Putty').

The Raspberry needs a working internet connection for the installation.

We then enter exactly this line in the command line (best copied here):

```
curl -sSL https://www.pilogger.de/get/installer-webmon | bash
```

We conclude the input with 'Enter'.

```
login as: karl
karl@192.168.178.60's password:
Linux pilogger-test 6.6.28+rpt-rpi-v7 #1 SMP Raspbian 1:6.6.28-1+rpt1 (2024-04-22) armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 29 03:16:28 2024 from 192.168.178.29
karl@pilogger-test:~$ curl -sSL https://www.pilogger.de/get/installer-webmon | bash

Dieses Script installiert die PiLogger WebMonitor Software
im Verzeichnis /home/karl/pilogger

Wollen Sie fortfahren ? [j/N] j
```

The 'curl' utility program loads the script from the PiLogger.de website and starts its execution. In order to allow an abort in the event of an accidental call, the installation start must be confirmed - with the 'j' key.

We are then asked whether we want to use the home network router as a time server:


```
login as: karl
karl@192.168.178.60's password:
Linux pilogger-test 6.6.28+rpt-rpi-v7 #1 SMP Raspbian 1:6.6.28-1+rpt1 (2024-04-22) armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr 29 03:16:28 2024 from 192.168.178.29
karl@pilogger-test:~$ curl -sSL https://www.pilogger.de/get/installer-webmon | bash

Dieses Script installiert die PiLogger WebMonitor Software
im Verzeichnis /home/karl/pilogger

Wollen Sie fortfahren ? [j/N] j

Wenn Ihr Heimnetz-Router einen NTP-Zeitserver zur Verfügung stellt,
können Sie die Zeit des Raspberry Pi damit synchronisieren.
Dadurch wird hierfür kein externer Internetzugang mehr benötigt.

Heimnetz-Router als Zeitserver verwenden ? [j/N] j
```

If we have set up the router accordingly as in 4.3.1, we can answer 'j' here.

Next, an alternative I²C driver is activated.

```
Wenn Ihr Heimnetz-Router einen NTP-Zeitserver zur Verfügung stellt,
können Sie die Zeit des Raspberry Pi damit synchronisieren.
Dadurch wird hierfür kein externer Internetzugang mehr benötigt.

Heimnetz-Router als Zeitserver verwenden ? [j/N] j

192.168.178.1 als Zeitserver gesetzt

i2c_arm=on Parameter bereits aktiv

Overlay i2c-gpio,i2c_gpio_sda=2,i2c_gpio_scl=3,i2c_gpio_delay_us=2,bus=1 zu /boot/firmware/c
onfig.txt hinzugefügt
```

The I²C hardware modules of the Broadcom processors have errors, which is why the I²C bus behavior 'Clock Stretching' (i.e. the bus master Raspi waits for the peripheral device that asks for a short pause) does not work with the standard drivers. However, the PiLogger and other I²C modules need this feature.

For this reason, an I²C module is installed here that maps the I²C port to the correct pins in software. This works very well for all Raspberry Pi's from Zero to Model 5.

These actions are then carried out automatically:

- Installing 'python3-smbus'
(Python extension for the I²C interface)
- Installing 'python3-rpi.gpio'
(Python extension for direct GPIO access)
- Installing python3-gpiozero
(new Python extension for direct GPIO access)
- Installing 'python3-bottle'
(WSGI Micro Web Framework for Python)

- Downloading and unpacking the PiLogger WebMonitor software components
- Setting up the autostart of the WebMonitor software

An option is then offered with which the daily contacts to the Raspberry Pi OS update servers can be switched off:

```

inflating: static/diagramme.html
inflating: static/dygraph_p.css
inflating: static/w3_pilo.css
inflating: static/kalibration.html
inflating: static/dygraph_p.min.js
inflating: static/download.html
inflating: static/einstell1.html
extracting: Version.txt
inflating: PiLogger-bottle.py
inflating: pilo-webmon.service
inflating: PiLo_Thermistor.py

Created symlink /etc/systemd/system/multi-user.target.wants/pilo-webmon.service → /lib/systemd/system/pilo-webmon.service.

Autostart eingerichtet

Die täglichen Kontakte zum Raspbian-Update-Server (apt-daily)
sind externe Internetkontakte und können deaktiviert werden.

Möchten Sie Apt-daily deaktivieren ? [j/N] █

```

Switching off is not absolutely necessary if the Raspberry's Internet access is blocked, because the requests will then simply go nowhere.

If the Internet access is not blocked, these requests generate initially harmless user statistics, but on the other hand only really make sense on a Raspberry that has other Internet tasks - for example on a desktop system (on which automatic updates are then carried out with the newer Raspberry Pi OS).

Otherwise, the Raspberry can work completely privately in your own home network as a logger and, without these breadcrumbs, offers no particular attack surface and therefore no urgent need for updates - provided you have a good and secure router.

If this last choice is answered with 'j', the result looks something like this:


```

Möchten Sie Apt-daily deaktivieren ? [j/N] j
Removed "/etc/systemd/system/timers.target.wants/apt-daily.timer".
O apt-daily.timer - Daily apt download activities
  Loaded: loaded (/lib/systemd/system/apt-daily.timer; disabled; preset: enabled)
  Active: inactive (dead) since Tue 2024-04-30 03:14:58 CEST; 2s ago
  Duration: 23h 58min 6.643s
  Trigger: n/a
  Triggers: ● apt-daily.service

Apr 29 03:16:51 pilogger-test systemd[1]: Started apt-daily.timer - Daily apt download...ties.
Apr 30 03:14:58 pilogger-test systemd[1]: apt-daily.timer: Deactivated successfully.
Apr 30 03:14:58 pilogger-test systemd[1]: Stopped apt-daily.timer - Daily apt download...ties.
Hint: Some lines were ellipsized, use -l to show in full.
Created symlink /etc/systemd/system/apt-daily.service → /dev/null.
O apt-daily.service
  Loaded: masked (Reason: Unit apt-daily.service is masked.)
  Active: inactive (dead)
  TriggeredBy: O apt-daily.timer
Removed "/etc/systemd/system/timers.target.wants/apt-daily-upgrade.timer".
O apt-daily-upgrade.timer - Daily apt upgrade and clean activities
  Loaded: loaded (/lib/systemd/system/apt-daily-upgrade.timer; disabled; preset: enabled)
  Active: inactive (dead)
  Trigger: n/a
  Triggers: ● apt-daily-upgrade.service

Apr 29 03:16:51 pilogger-test systemd[1]: Started apt-daily-upgrade.timer - Daily apt ...ties.
Apr 30 03:15:05 pilogger-test systemd[1]: apt-daily-upgrade.timer: Deactivated successfully.
Apr 30 03:15:05 pilogger-test systemd[1]: Stopped apt-daily-upgrade.timer - Daily apt ...ties.
Hint: Some lines were ellipsized, use -l to show in full.

Die Installation ist jetzt durchgeführt.
Einige Änderungen am System erfordern einen Neustart.
Nach dem Neustart wird automatisch PiLogger WebMon ausgeführt.

Bereit zum Neustart ? [j/N] █

```

The installation is now complete and the PiLogger WebMonitor can be started by restarting the Raspberry.

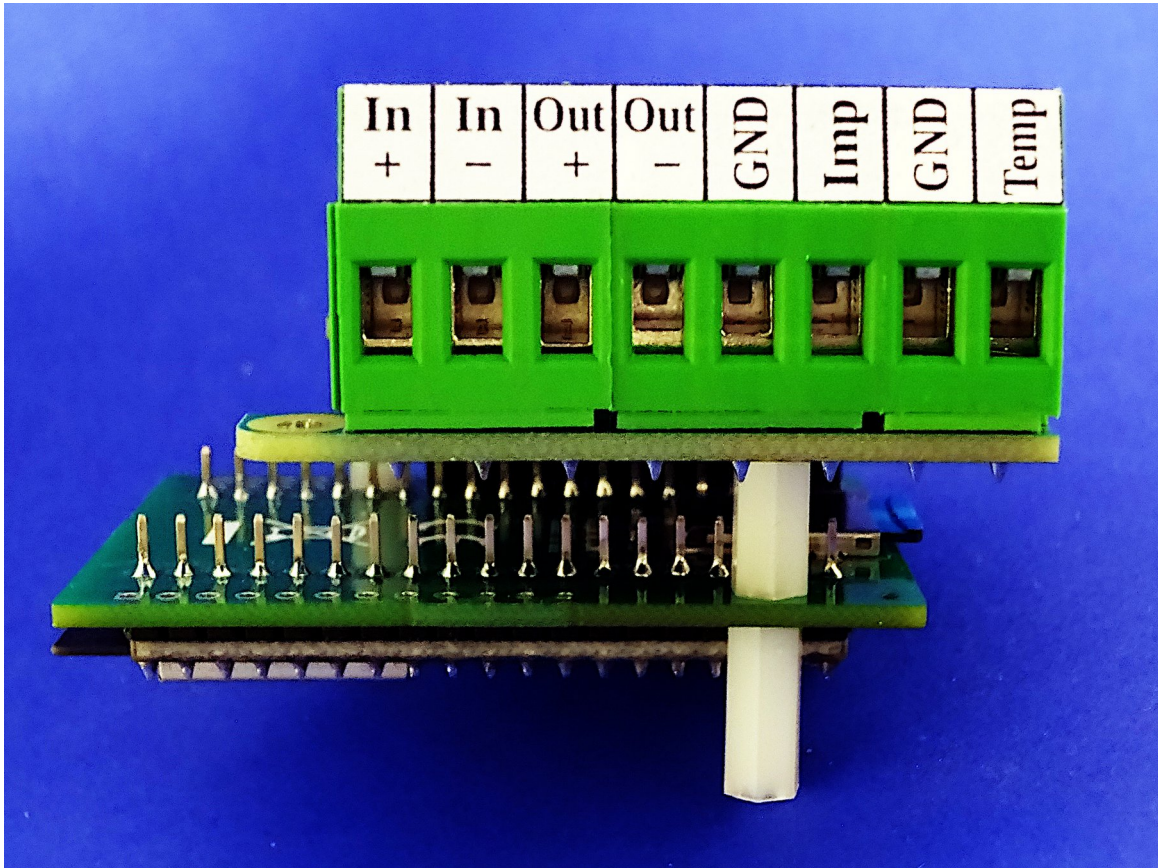
So we confirm with a 'j' - and the Raspberry terminates the connection and restarts. After this restart, we can establish the first connection to the WebMonitor with a web browser.

The operation of the WebMonitor software is described in chapter 6.

5 Installing WebMonitor on ESP32 Module

The PiLogger One can now also be operated with an ESP32 module from Espressif Systems.

We have adapted (ported) the PiLogger Web-Monitor accordingly. You can find the archive with all required files here : [PiLogger Web-Monitor ESP32](#)



A prerequisite for successful logger operation is a fast SD card for data storage. Since the ESP32 usually comes without an SD card holder and available SD card shields then use the SPI interface with only 1 data line, we have designed a special adapter plate for the PiLogger One that uses the SDIO interface of the ESP32 with 4 data lines.

This adapter is available in our store : [Buy adapter plate](#)

The PiLogger Web-Monitor ESP32 is based on [MicroPython](#) by Damien George. This means that the firmware of the ESP32 module must first be overwritten with the ESP32 port of MicroPython. The necessary .bin file is included in the download archive.

The ESP32 can then be addressed directly on the command line of the Python interpreter (REPL).

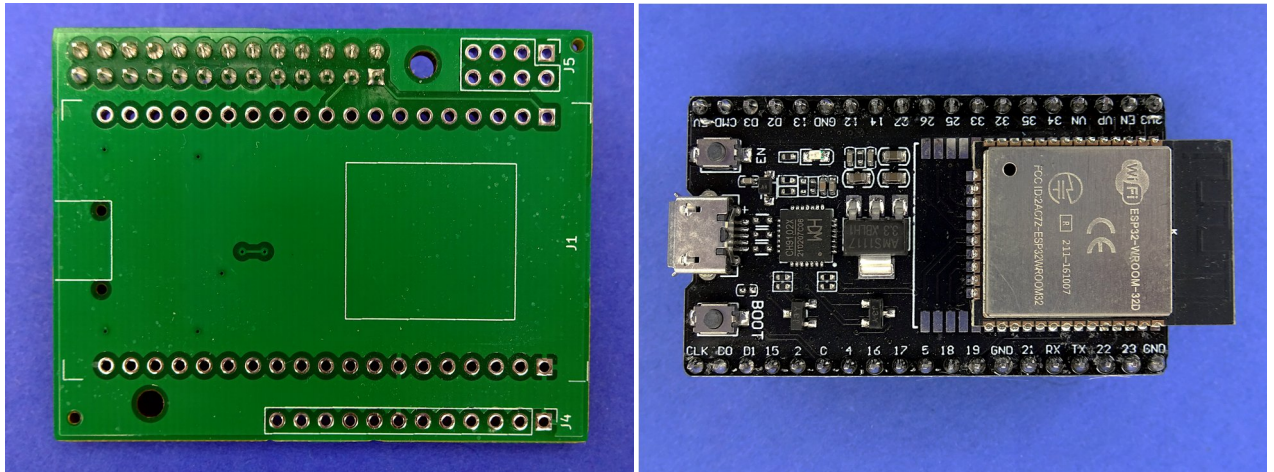
We therefore recommend [Thonny](#) as a convenient development environment for the next steps.

Thonny can also be used as a GUI for the Espressif Flash tool [esptool](#).

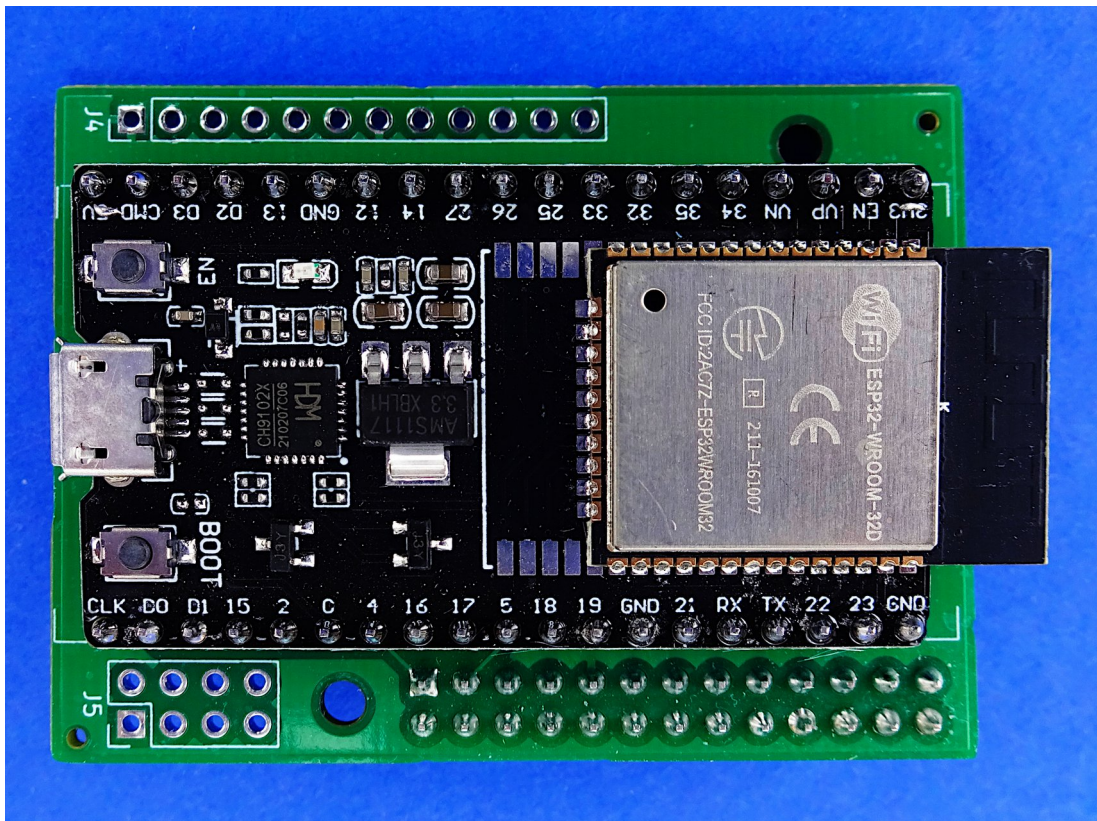
The following chapters describe the complete installation step by step.

5.1 Soldering adapter and module

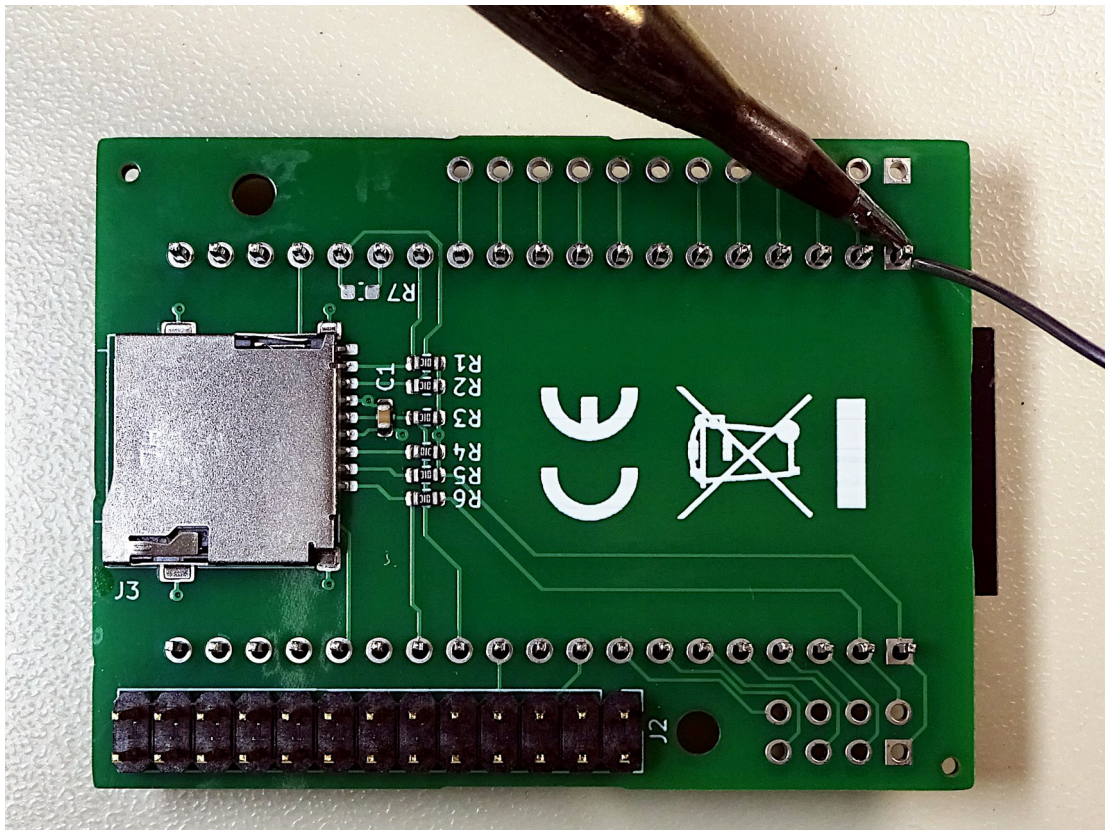
The adapter plate has white markings for the micro-USB socket and the ESP32 core module on the side where the ESP32-DevKit-C module must be plugged in. This indicates which way round the module must be plugged in:



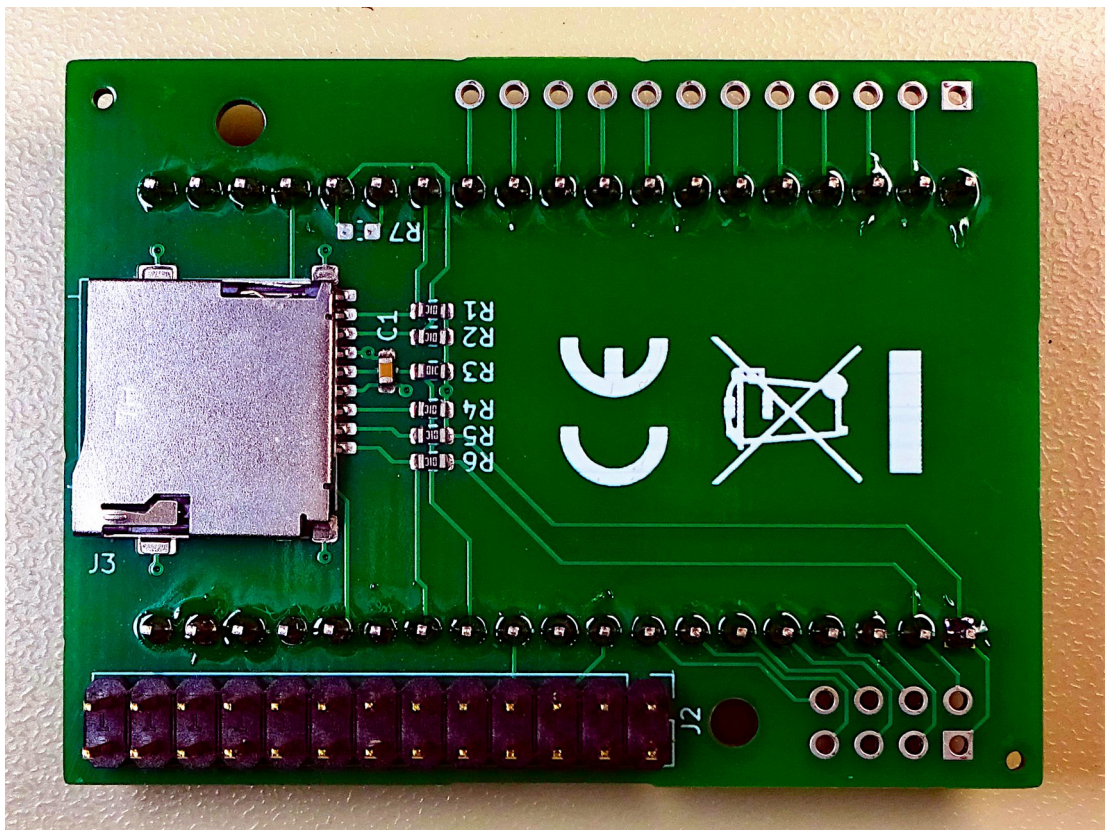
When assembled, it looks like this:



Now this combination must be turned around so that the pins that have been pushed through and are exposed on the other side can be soldered.

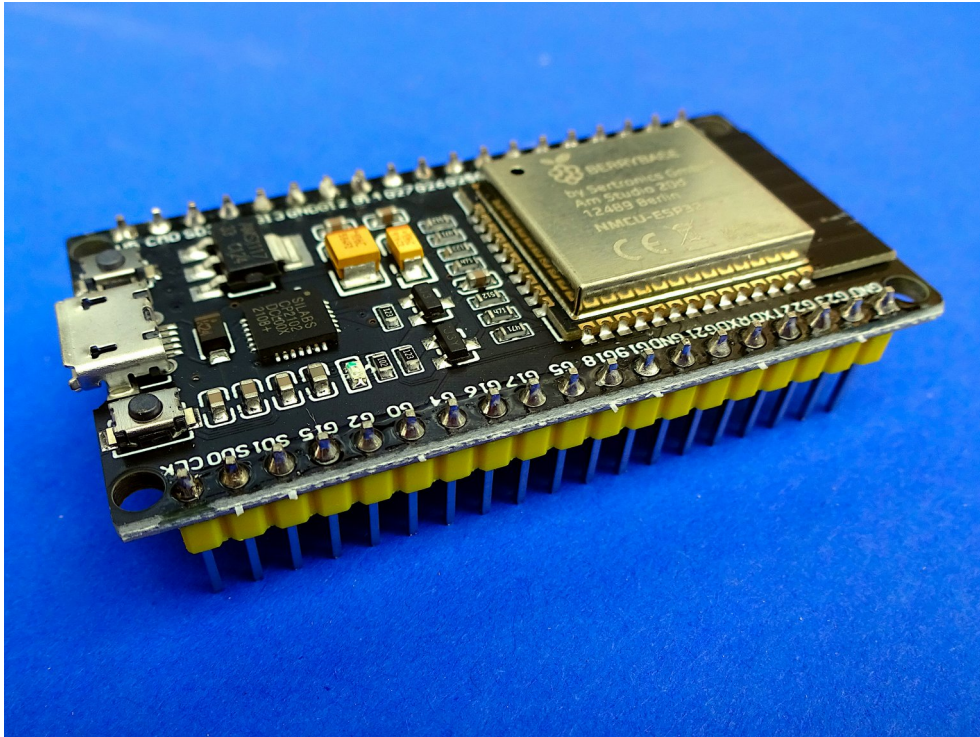


Once soldered, it looks like this :



5.1.1 Adapter with NodeMCU ESP32 module

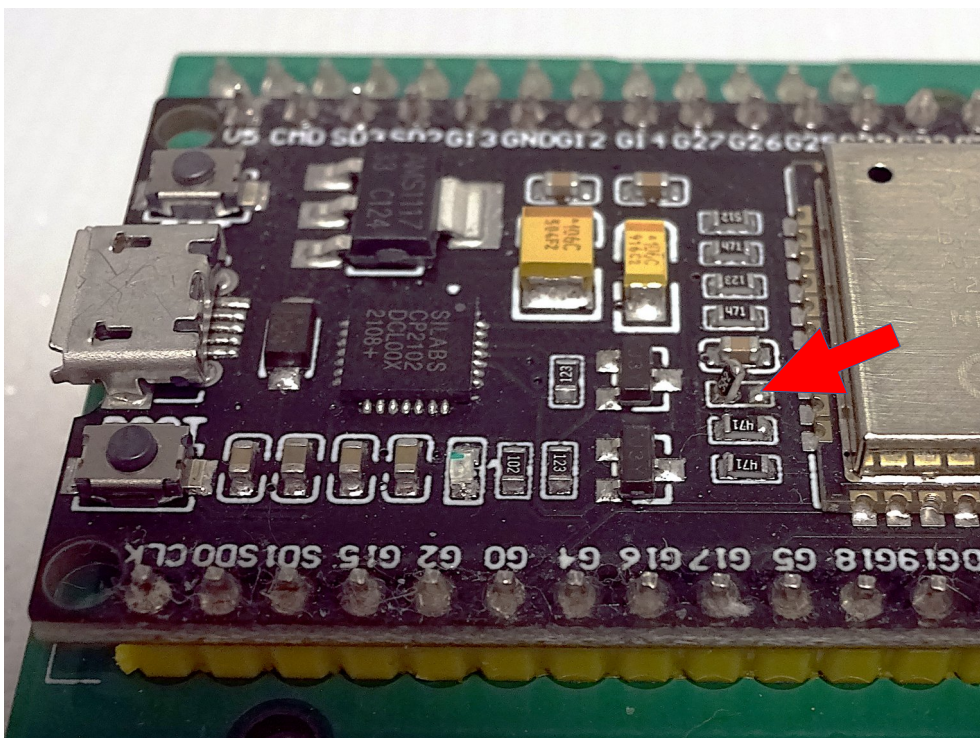
The PiLogger adapter for ESP32 modules can also be used with a so-called NodeMCU module. This module also has 38 pins, which have the same assignment. The module looks like this:



However, there is one detail that requires a small adjustment:

With this module, there is a pull-down (5.1 kΩ resistor to GND) at GPIO G2, which blocks the operation of the SD card.

This resistor must be removed or soldered 'up' as shown in the picture:

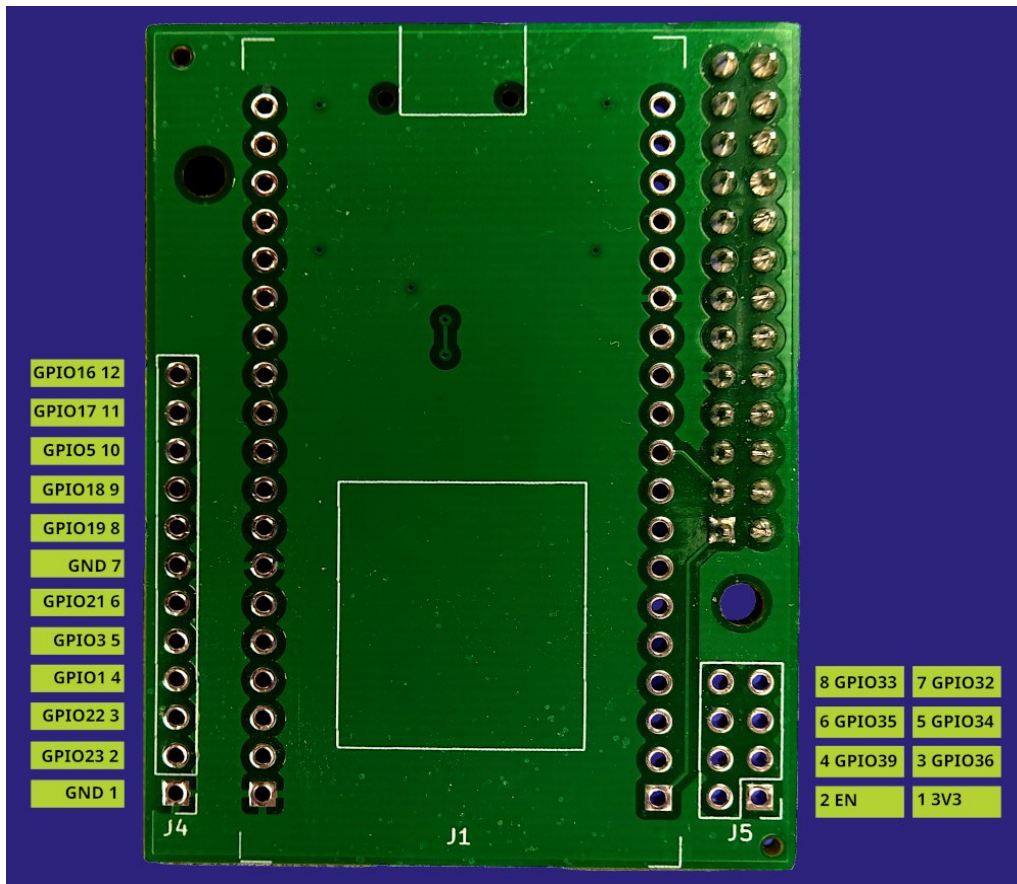


5.1.2 Assignment of optional pin fields

The unused pins of the ESP32 module are led out to 2 pin headers that can be populated. J4 is a 12-pin, single-row pin header, J5 is an 8-pin, double-row pin header.

Of course, individual cables can also be soldered into the solder pads of the unpopulated, optional pin headers.

The assignment of these pin headers is shown in this picture:



5.2 Installing Thonny

On [Thonny](#)'s homepage you will find the right installation variant for your PC. If required, Thonny can bring Python with it.

After successful installation - please start once - Thonny must now be set to the ESP32. To do this, please go to 'Options...' under 'Extras'. Switch to the 'Interpreter' tab. Now search for 'MicroPython (ESP32)' in the drop-down list and select it.

Select the USB setting 'automatic' under 'Port or WebREPL' and leave the rest as it is - 'OK'.

Now you can connect the ESP32 to a USB port on your PC using a MicroUSB cable.

The red LED indicates that the module has power - the current requirement is low enough (< 250 mA) to be supplied by the PC. Now press the red stop/restart button in Thonny once and then press the reset button on the module. A fresh module should then respond like this:

```
ets Jul 29 2019 12:21:46

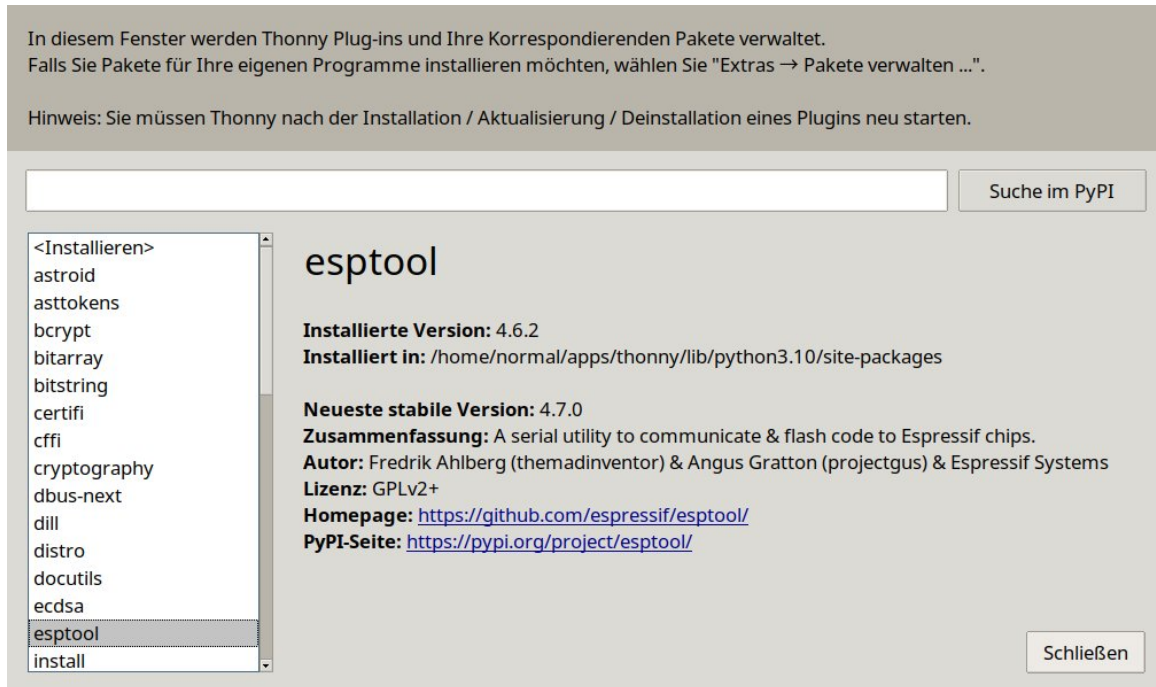
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:5564
load:0x40078000,len:0
load:0x40078000,len:13756
entry 0x40078fb4
I (29) boot: ESP-IDF v3.0.3 2nd stage bootloader
I (29) boot: compile time 08:53:32
I (29) boot: Enabling RNG early entropy source...
I (34) boot: SPI Speed      : 40MHz
I (38) boot: SPI Mode      : DIO
I (42) boot: SPI Flash Size : 4MB
I (46) boot: Partition Table:
I (49) boot: ## Label      Usage          Type ST Offset   Length
I (57) boot:  0 phy_init    RF data        01 01 0000f000 00001000
I (64) boot:  1 otadata     OTA data       01 00 00010000 00002000
I (72) boot:  2 nvs         WiFi data      01 02 00012000 0000e000
I (79) boot:  3 at_customize  unknown        40 00 00020000 000e0000
I (87) boot:  4 ota_0         OTA app        00 10 00100000 00180000
I (94) boot:  5 ota_1         OTA app        00 11 00280000 00180000
I (102) boot: End of partition table
I (106) boot: No factory image, trying OTA 0
```

This means that there is no application in Flash. All good so far.

5.3 Checking/updating 'esptool'

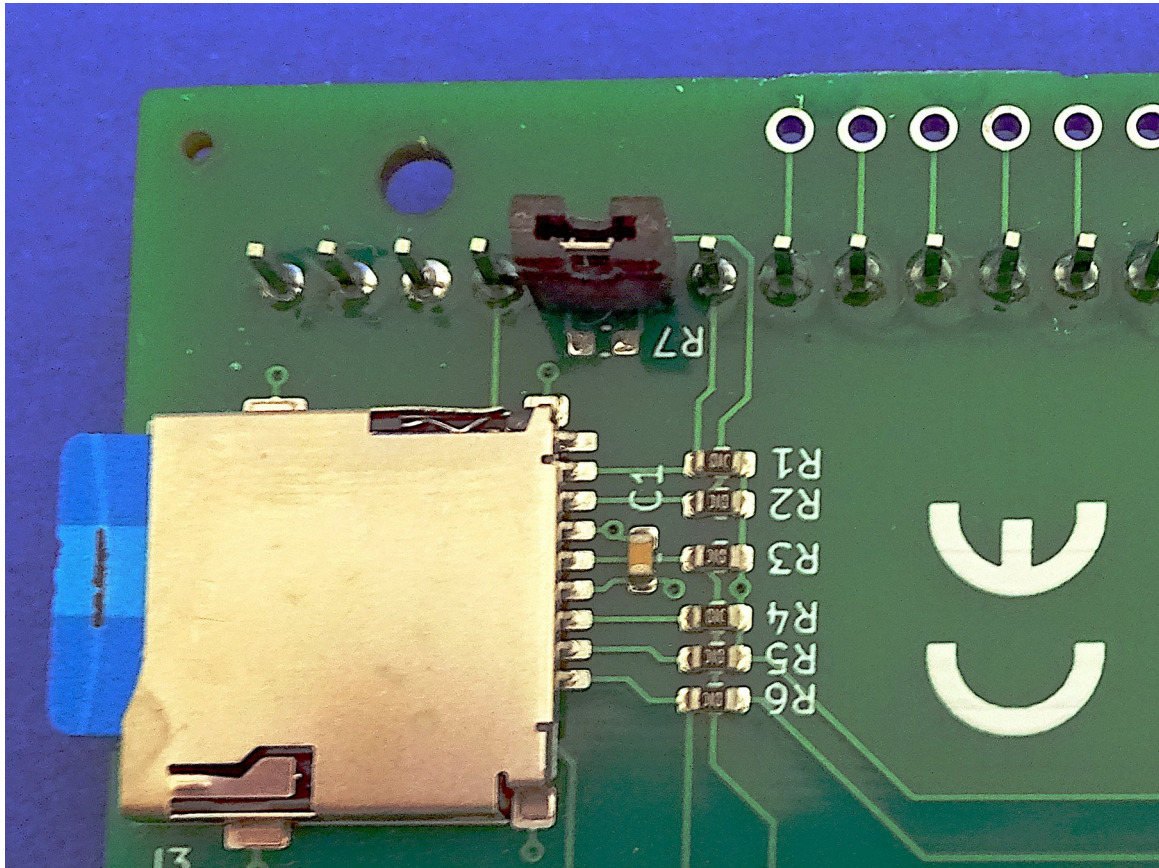
Thonny not only includes Python, but also the Flash tool 'esptool', which itself runs under Python (on the PC). Here you should make sure that you have the latest version (> 4.62). We check this in Thonny by going to 'Manage plug-ins...' under 'Tools'.

Now search for 'esptool' in the list of packages displayed on the left and select it with a mouse click. The main window now shows the installed version and the installation directory. If necessary, please install a newer version from [PyPi](https://pypi.org/project/esptool/).



5.4 *Preparing the module with adapter*

If the ESP32 module is already soldered to the adapter board, the pull-up for the SD_D0 signal on GPIO2 prevents the flash from being recognized and programmed. This pin must therefore be shorted to its neighbor GPIO0 before flashing. These two pins are marked on the adapter board by the unassembled resistor R7. The easiest method here is to borrow the jumper from the PiLogger One and plug it onto the two module pins (don't forget to plug it back onto the PiLogger afterwards):



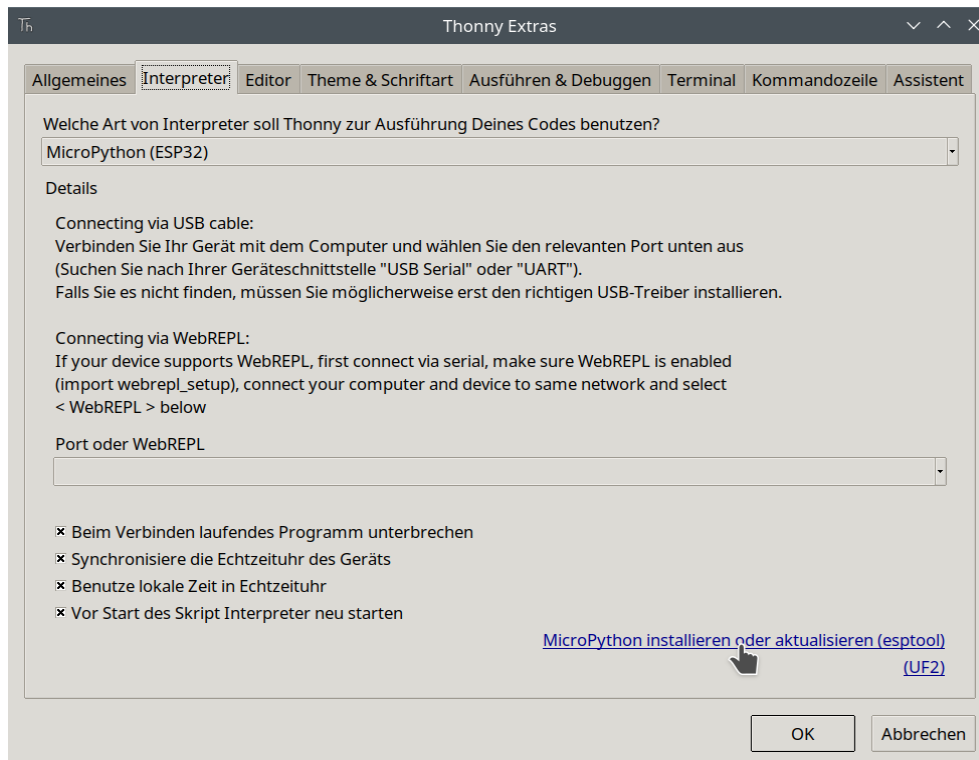
5.5 *Flashing MicroPython*

Thonny searches for the file to be flashed in the root directory by default. As the development and tests took place on the MicroPython version 'ESP32_GENERIC-20240222-v1.22.2.bin', this version of the .bin file is included in the download archive.

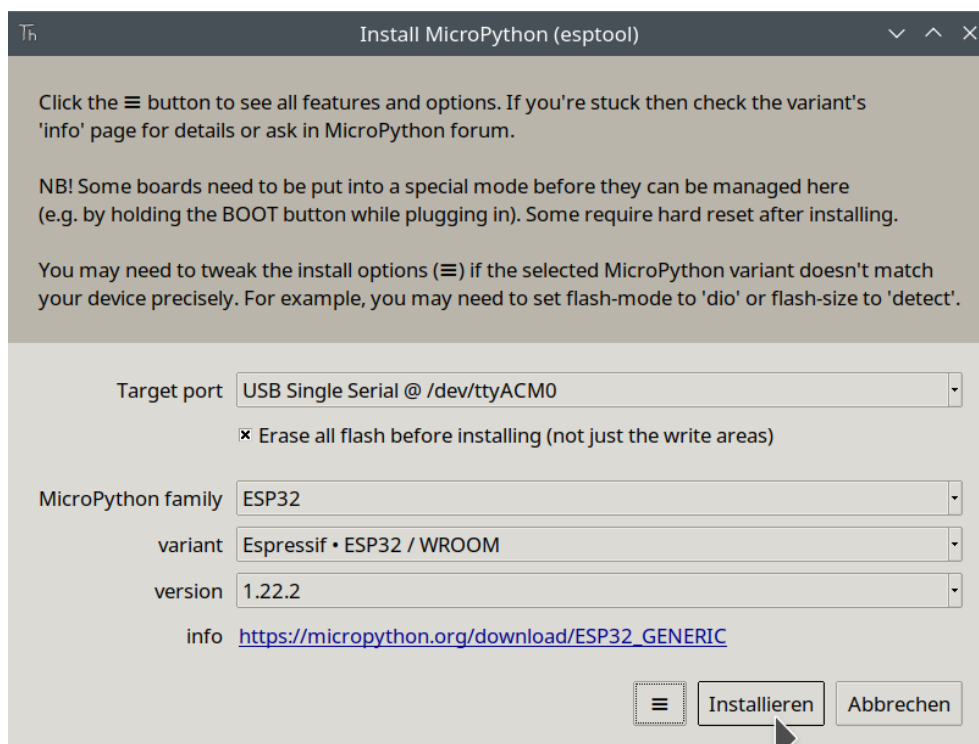
We unpack this file into the root directory on the PC.

In Thonny, we call up the 'Interpreter' page under 'Run', 'Configure the interpreter...':

PiLogger One Manual



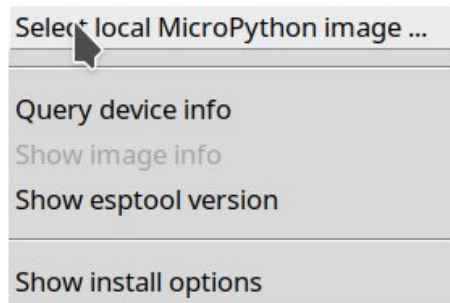
Now we click on 'Install MicroPython'.



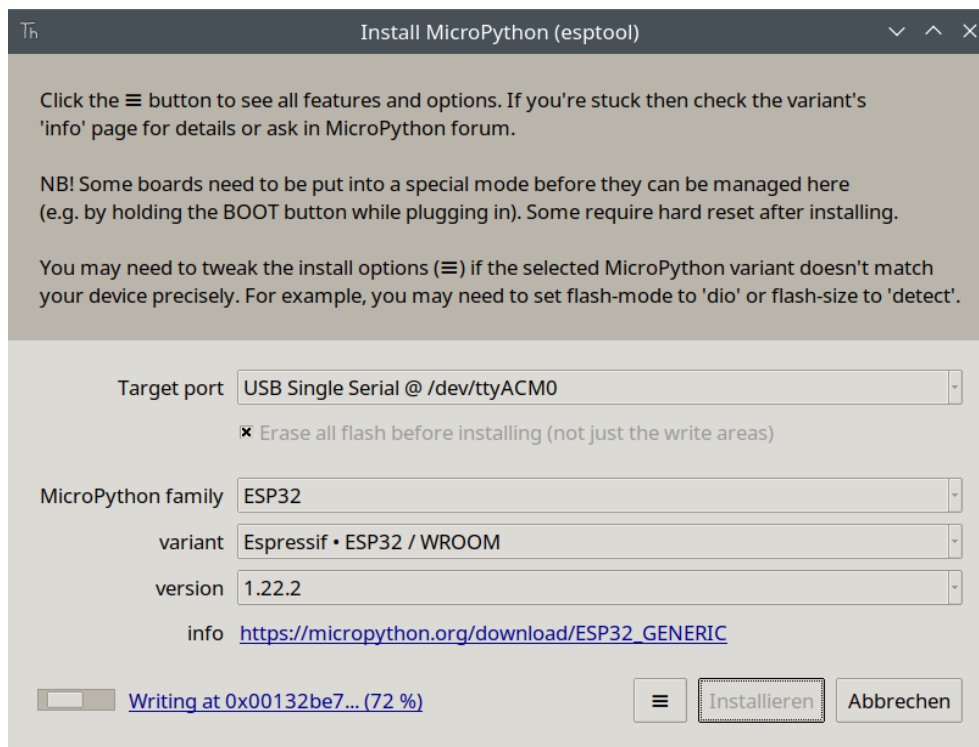
This screenshot is an example from a Linux computer, which is why the 'Target port' here is called 'ttyACM0'. The real USB port can be selected from the drop-down list using the automatic search function.

By selecting 'ESP32' for 'MicroPython family' and 'Espressif - ESP32 / WROOM' for 'variant', the current version is selected and displayed as a download via an online query.

If a newer version than '1.22.2' appears here and there are therefore compatibility problems, the local file contained in the archive can be used instead. To do this, we press the button with the 'hamburger' symbol and get this context menu:



We can then navigate to the unpacked file 'ESP32_GENERIC-20240222-v1.22.2.bin' with a file selector and select it with 'Open'. Click on 'Install' to start the Flash process.



After writing, the message 'Done !' appears at the bottom left.

We close the two windows and reset the ESP32 with the reset button on the module. After clicking on the stop symbol in the menu bar of Thonny, the ESP32 should respond with the MicroPython prompt.

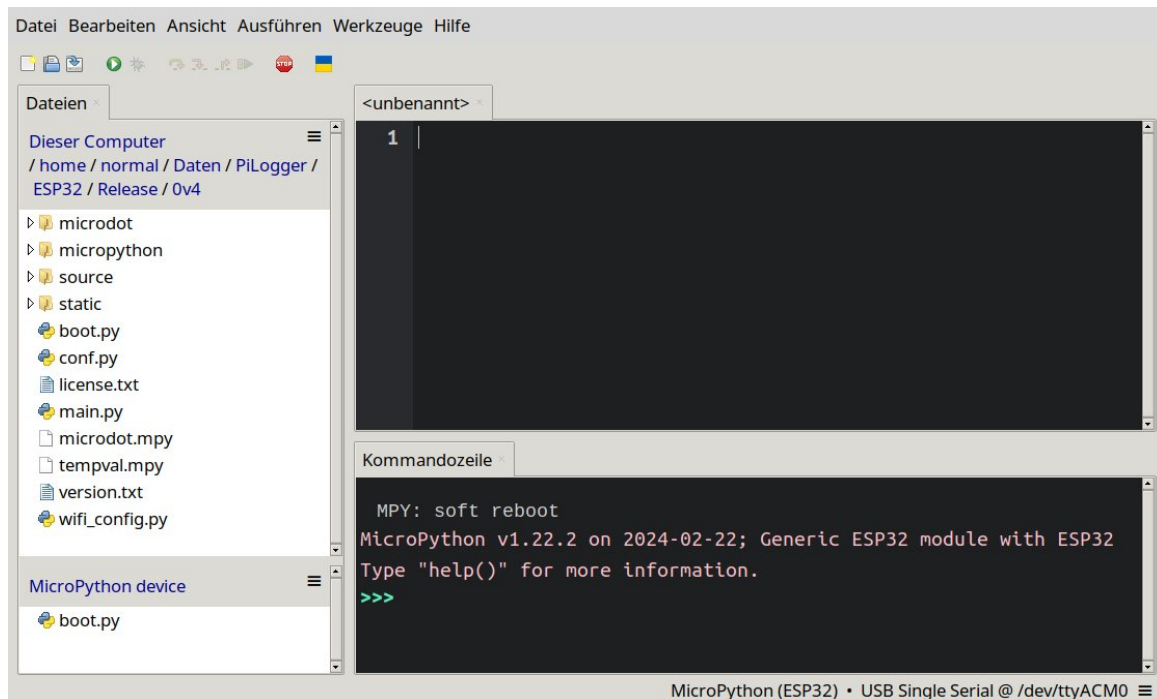
We can then remove the jumper again and plug it back into the correct position on the PiLogger.

5.6 Copying the archive files

Now we unpack the downloaded archive into an appropriately named working folder for Thonny.

If we now open Thonny, plug in the ESP32 module, press 'Stop/Restart' in Thonny and reset the module 😊

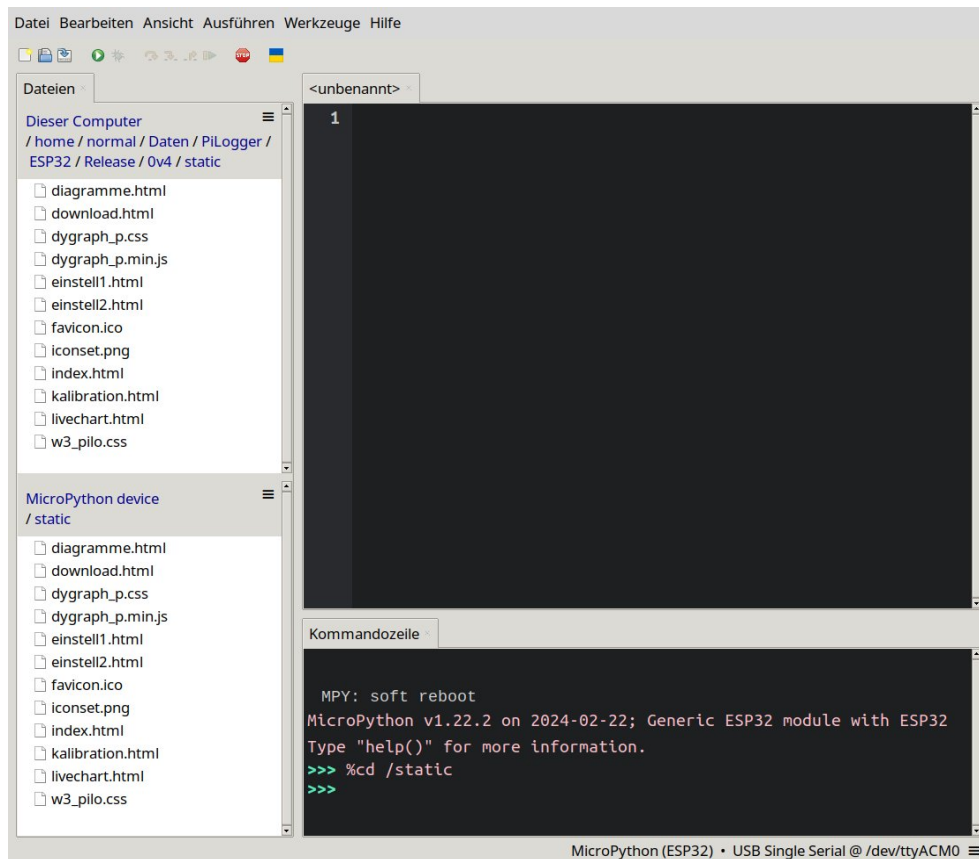
then it should look something like this on the screen:



This means that there is now exactly one file on the 'target' - the 'MicroPython device': 'boot.py'. It doesn't do anything - it doesn't even play...

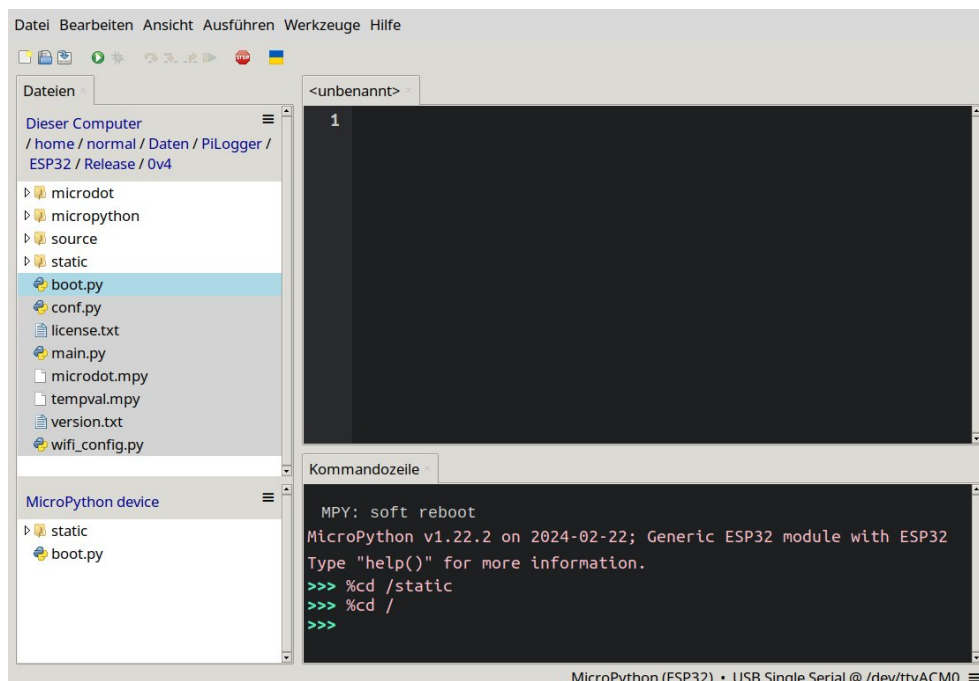
By right-clicking on the 'static' folder in the upper file tree, we call up the context menu in which we can copy the entire folder to the target with 'Upload to /'.

Then we switch to the 'static' folders in both file trees and briefly check that they have the same content:



Now we can go back one level in both trees.

In 'This computer' we select the following 8 files with 'Ctrl'+left-click:



By right-clicking on one of the selected files, we can then upload all 8 at the same time with 'Upload to /'.

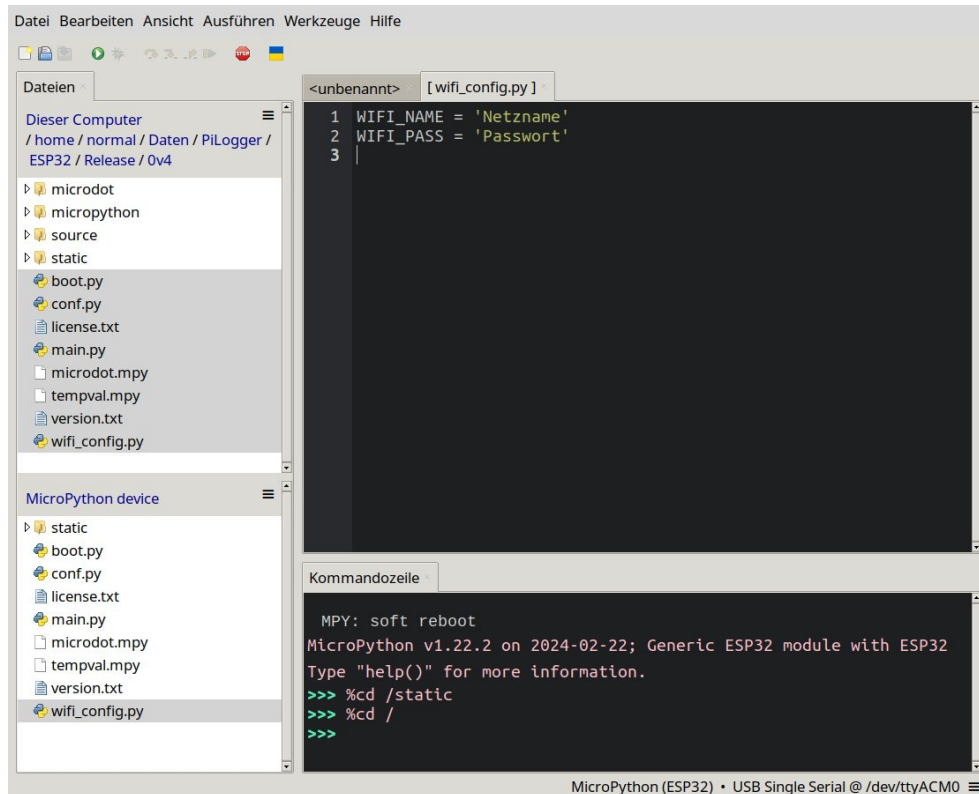
For the 'boot.py' file, we are asked whether it should be overwritten, which we confirm.

The required files are now on the ESP32.

5.7 Customizing the WiFi access data

In the file 'wifi_config.py' on the ESP32, the data actually required for the ESP32 to log into the home network must now be entered.

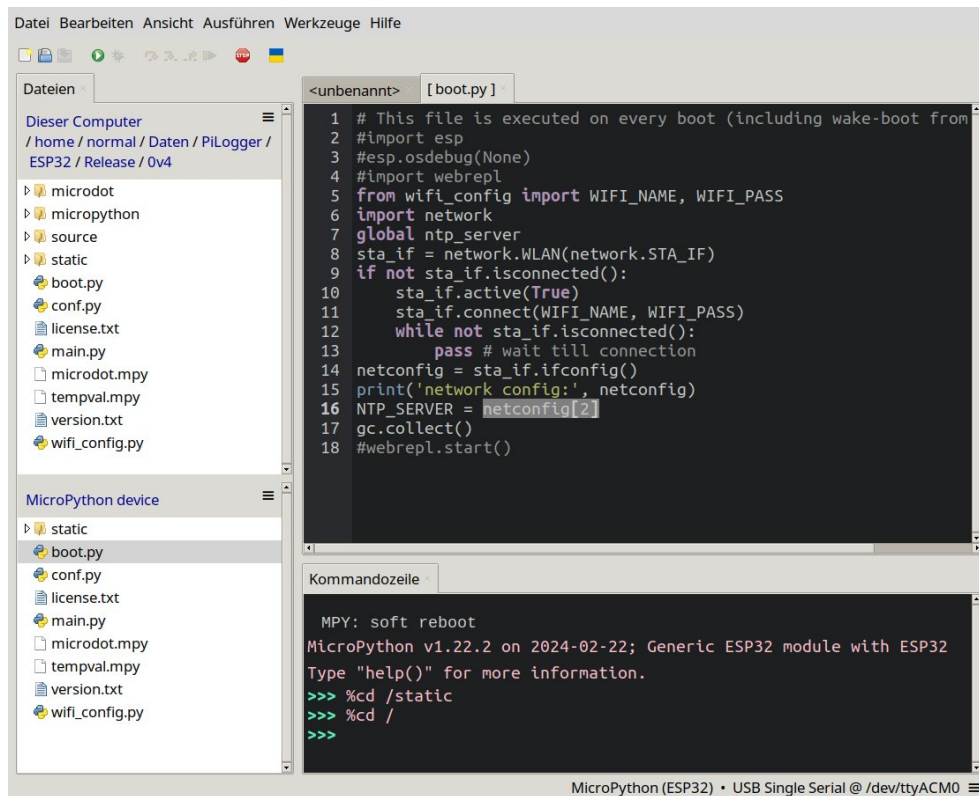
To do this, we double-click on this file, which downloads it from the ESP32 and displays it in the upper window of Thonny - the editor:



After entering the real individual values for the placeholders between the quotation marks, we save by clicking on the 'Save (Ctrl+S)' icon or via 'File', 'Save'

Important note : The software normally assumes that the home network router acts as an internal NTP server (can be set on many routers).

If this is not possible, the 'boot.py' file on the ESP32 must be changed. To do this, the value for 'NTP_SERVER' in line 16 must be changed to 'pool.ntp.org', for example:



Please note !

This means that Internet access is required for time synchronization and the ESP32 can no longer be blocked for Internet access!

Local time synchronization (router as internal NTP server) is the better solution for security reasons.

5.8 Changing the network name

By default, the ESP32 is given the network name 'espressif'.

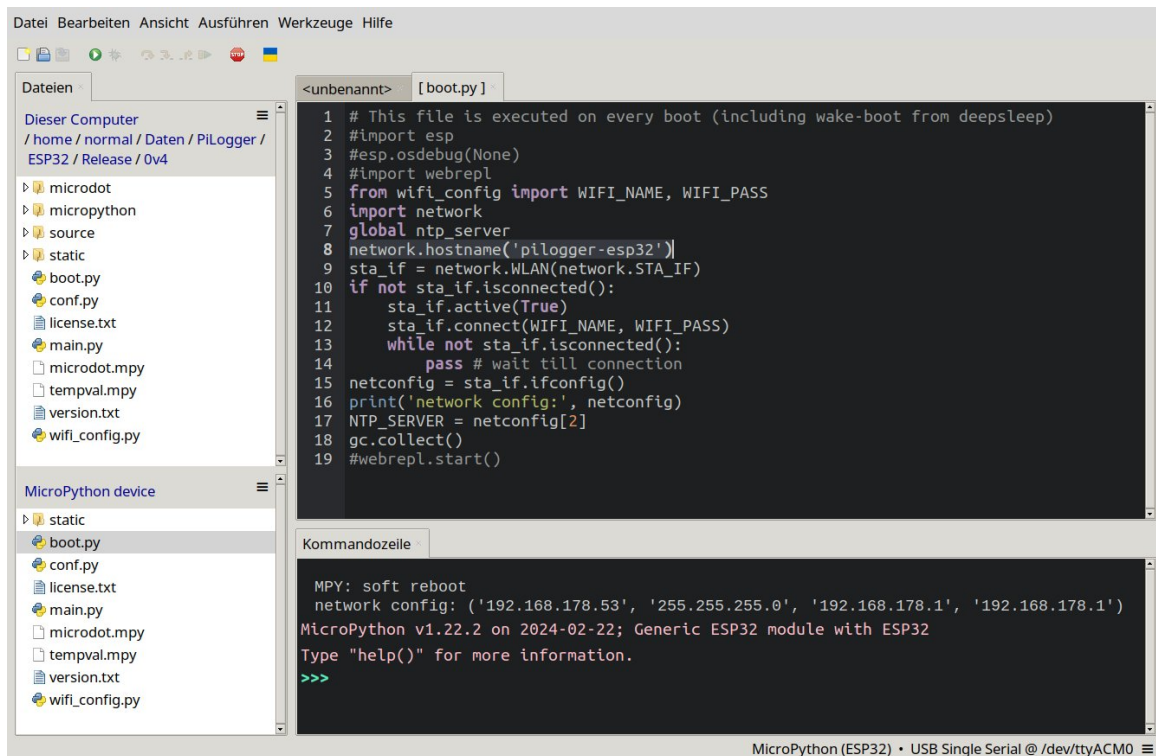
This can be changed by inserting an additional line in the 'boot.py' file on the ESP32. The line reads:

```
network.hostname('pillogger-esp32')
```

This configures the MicroPython module 'network', so the line must appear after the import of this module. And this configuration should take place before the WLAN interface is initialized.

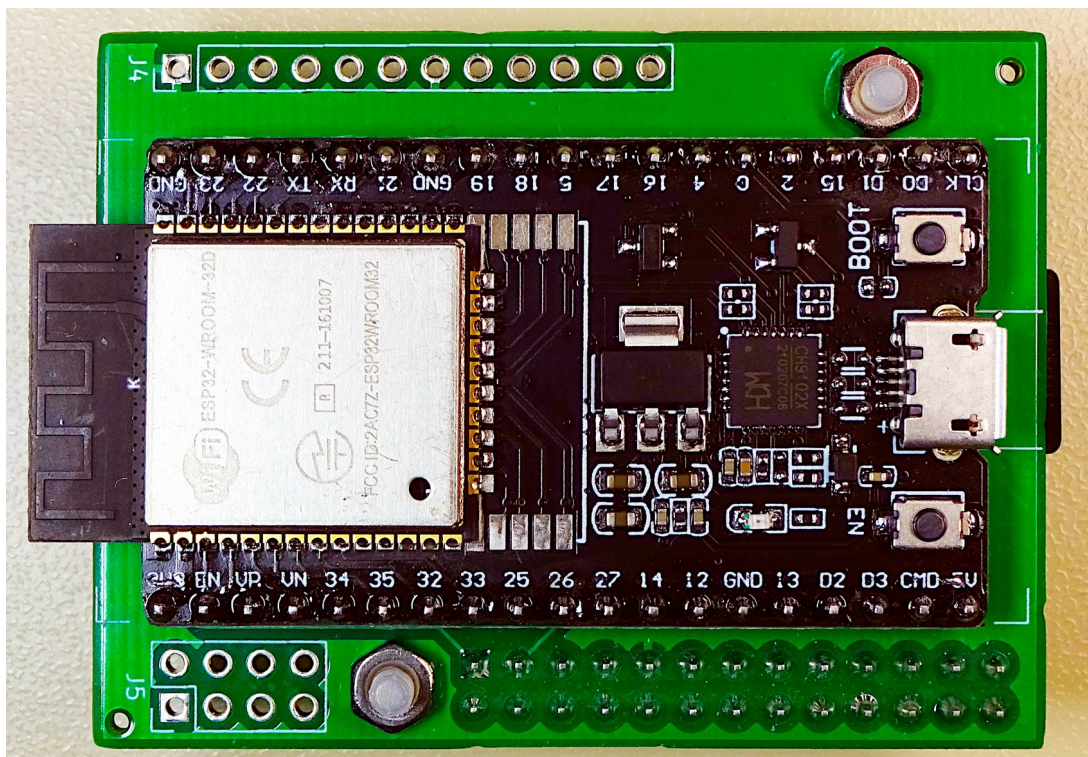
The name selected here is of course only an example and can be customized as required. In accordance with RFC specifications, it is recommended to use only lower case letters, numbers and the minus sign - for maximum compatibility.

The whole thing then looks something like this :

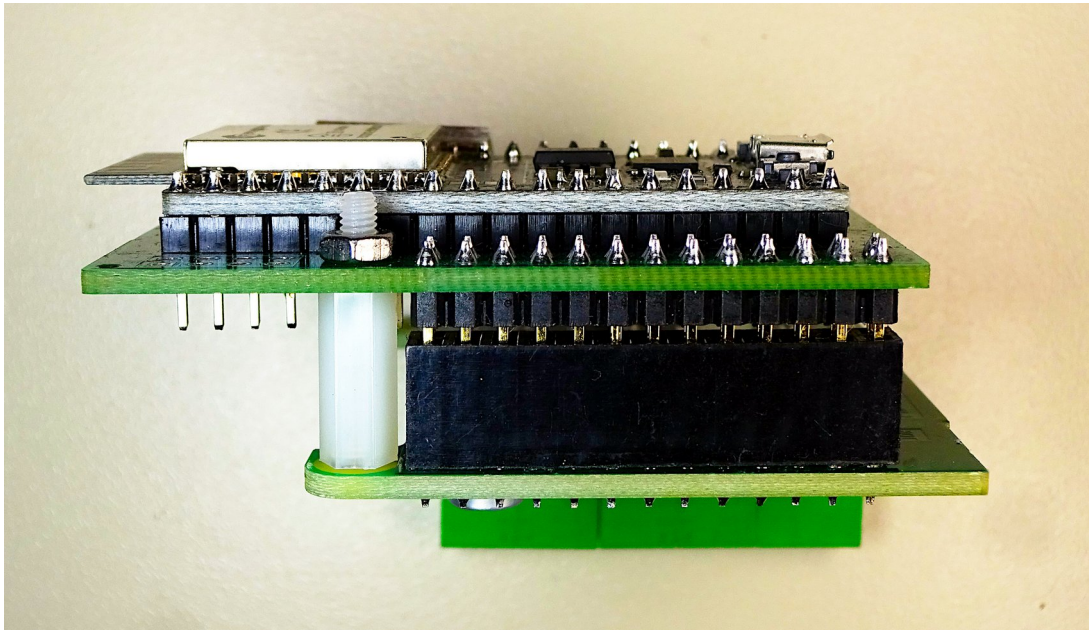


5.9 Mounting the PiLogger One

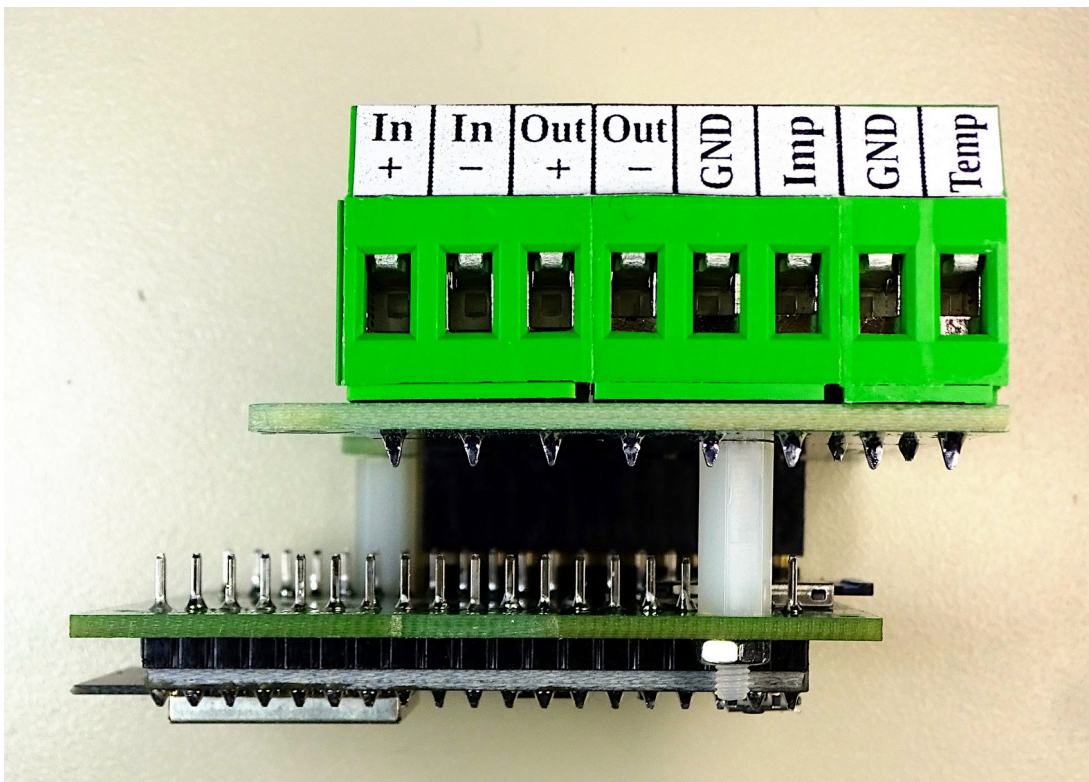
To assemble the ESP32 with adapter plate and the PiLogger One, we first insert 2 spacer bolts into the mounting holes of the adapter plate and screw them together with a nut without a washer:



Then insert the pin header of the adapter plate exactly into the socket connector of the PiLogger and press them together so that an even gap remains in accordance with the spacer bolt:

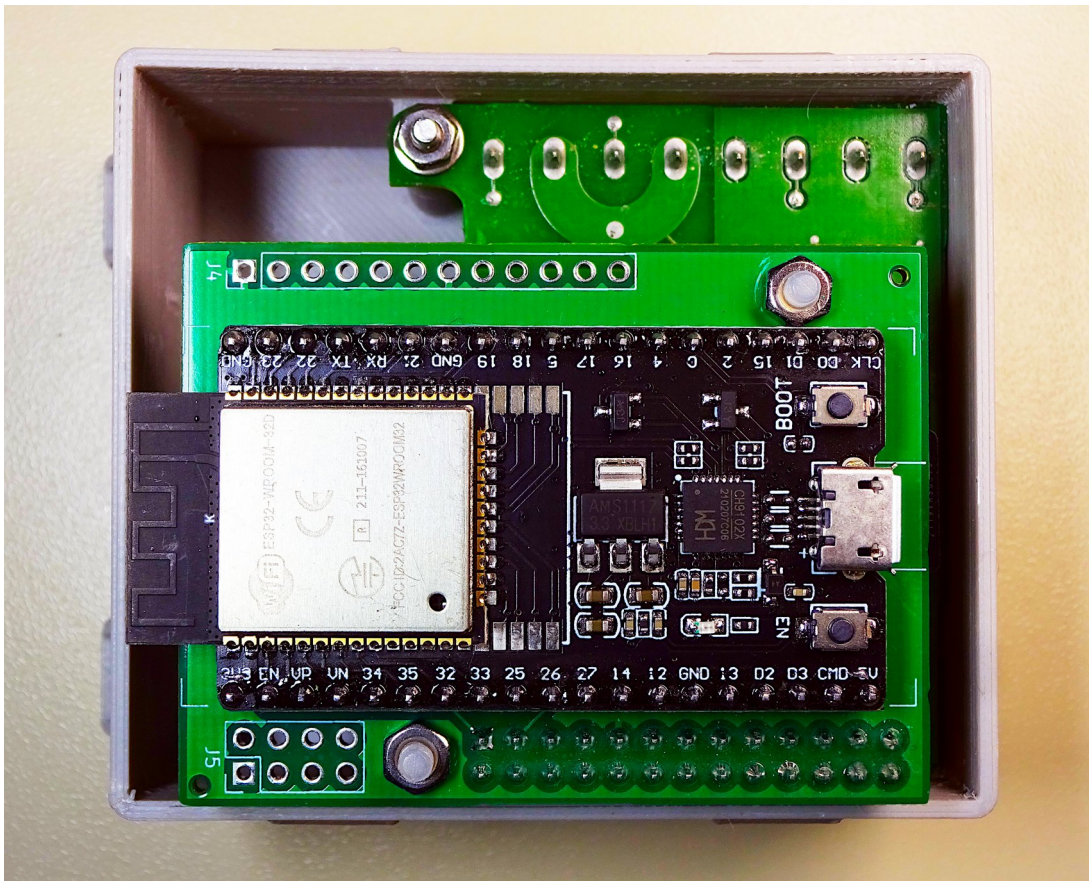


From the other side, the whole thing looks like this :



If the 3D-printed ESP32 housing from our store is to be used, the double pack can now be inserted upside down into the housing.

The 2 screws, which are screwed into spacer bolts, can then be inserted directly. Then insert the third screw with washer and nut:



If no housing is used, the two spacer bolts are each screwed to the PiLogger with a washer and screw.

5.10 *Switching on for the first time*

Now just reconnect the module with the USB cable and the ESP32 will boot as a PiLogger 😊 with the network name 'espressif' or the specially selected network name.

When starting with Thonny, the ESP32 reports the IP address under which it can be reached in the debug window.

On any device in the home network, you can now enter the IP with the suffix ':8080' as the address in a browser.

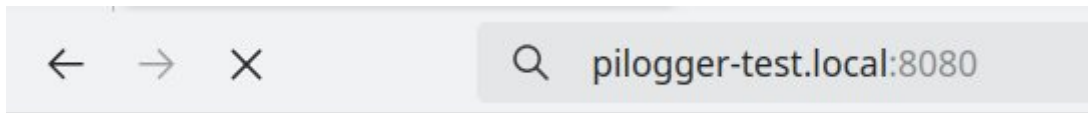
For example, '192.168.178.53:8080'.

6 Operating instructions WebMonitor Software

6.1 Calling up the web page with the browser

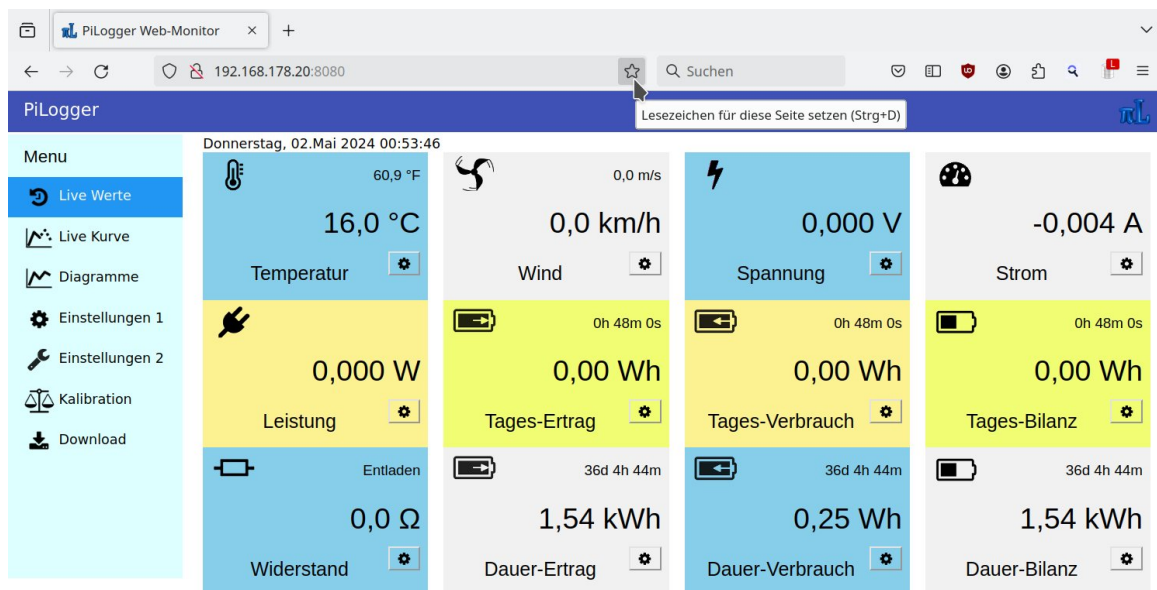
Once all the preparations from the previous chapters have been successfully completed, the PiLogger WebMonitor web page can be accessed on any device in the home network using a web browser.

To do this, we enter the network name of the Raspberry followed by '.local:8080' in the address bar of the browser:



For some browsers, '.local' is not absolutely necessary - it only indicates to the browser that the website is located in the local network. For particularly stubborn browsers / routers, the IP address with the addition ':8080' - the port - must be used instead. For example, '192.168.178.20:8080'.

The entry must be completed with 'Enter'.

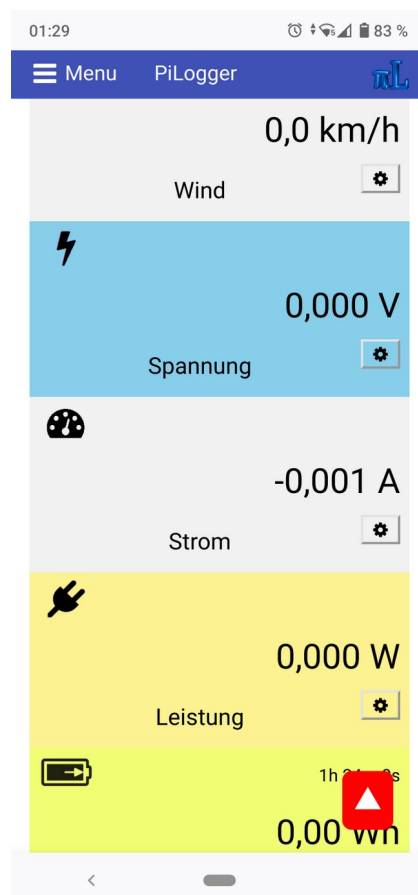
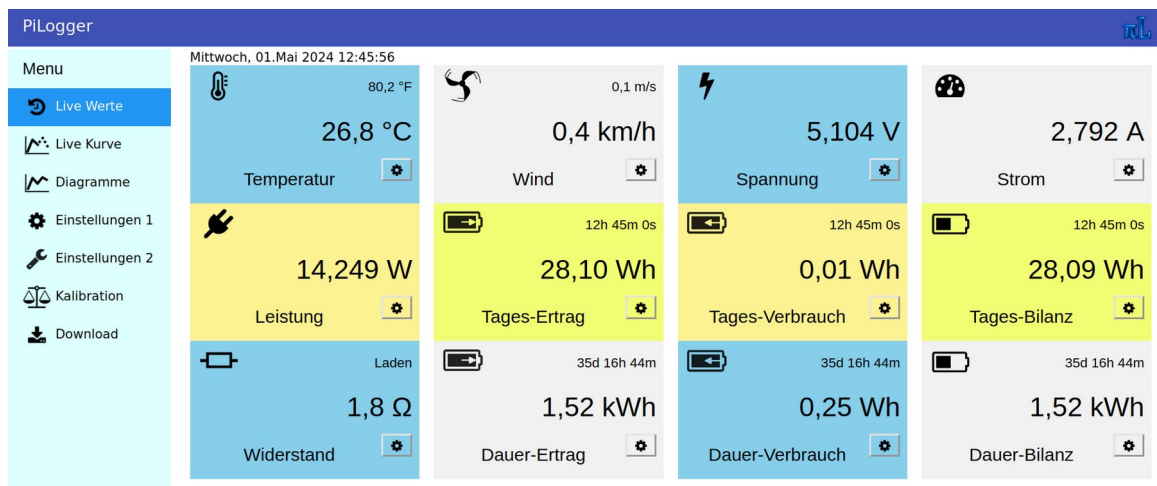


This is the start page of our newly set up server if everything went as planned. We can now create a favorite for the browser - in this example for 'Firefox' by clicking on the star on the far right in the address bar. This allows us to call up this page again with a single click.

6.2 The Homepage - *Live Values*

The start page displays the current measurement and counter values of the PiLogger One and automatically refreshes them every 2 seconds.

There is a navigation menu on the left-hand side that allows you to call up the other pages. The current page is highlighted in blue. This menu disappears on devices with a screen that is too small and must be called up separately by tapping on the menu icon that appears instead.



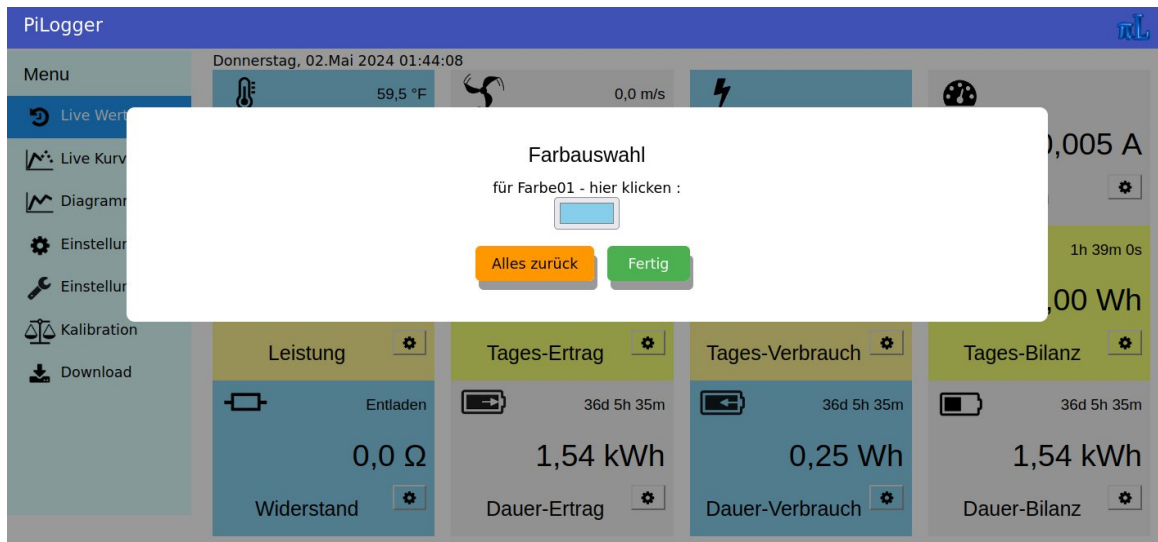
On smaller screens, such as smartphones, the measured value fields appear as a list. If this list is scrolled (swiped up), a red square appears, which allows you to quickly return to the beginning of the list.

Each display field for a measured value or counter reading also serves as a button for a link to the time diagram display of the respective value.

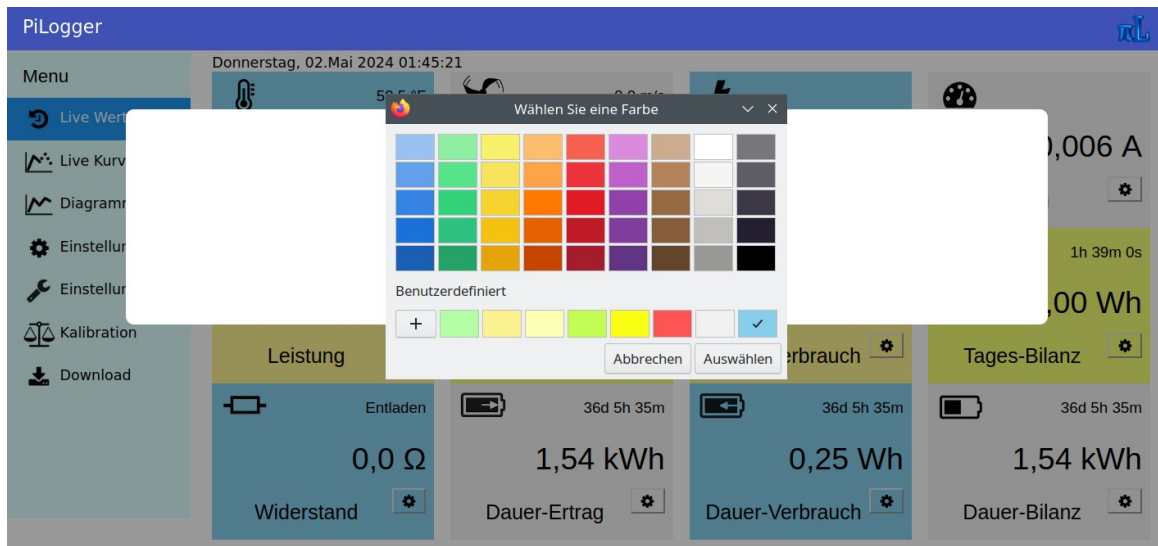
A click or tap on the middle of the temperature field therefore calls up the diagram page and activates the display of the mean values of the temperature measurements.

The same applies analogously to the other measured value fields.

A click (or tap) on the small gear icon to the right below the measured value opens a pop-up window for selecting the background color for the relevant measured value tile.



Another click on the color swatch field opens the actual color picker of the browser used. The color pickers of the various browsers are very different - here, for example, the desktop version of Firefox:



There is already a considerable pre-selection here - but you can mix any color you like by clicking on the '+' in the 'Custom' area.

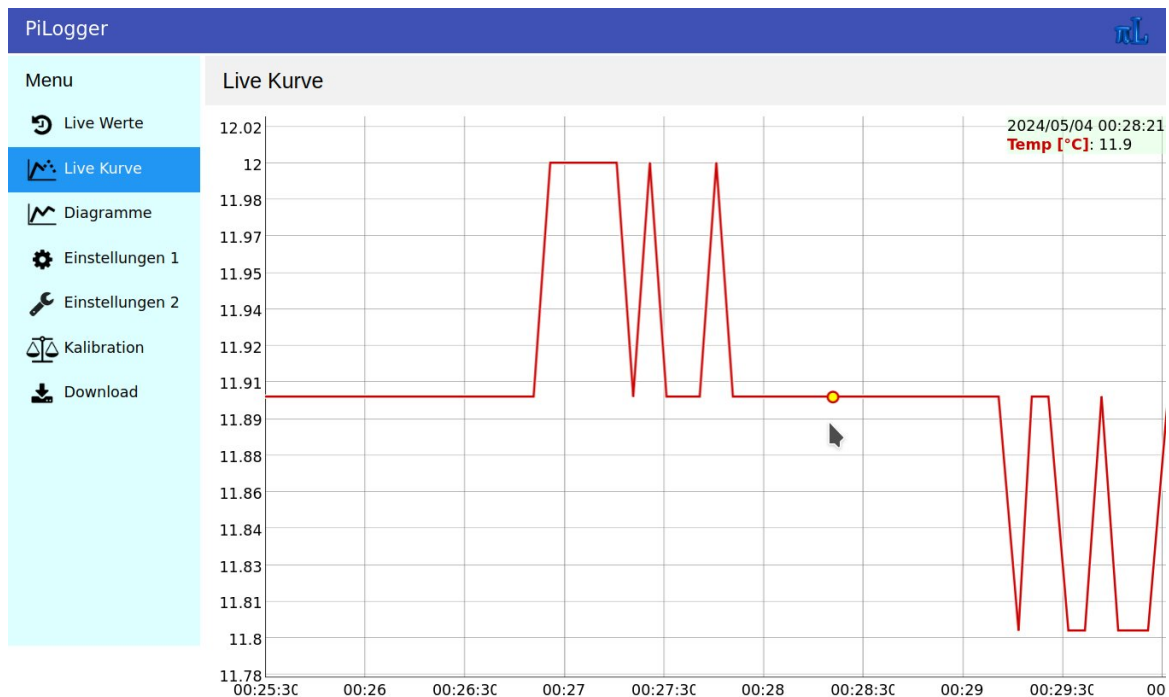
But this is totally different in the Android version of Firefox - there are exactly 10 colors to choose from ...

Because this can also lead to unwanted results, there is a 'Reset all' button in the pop-up window that can be used to restore the predefined colors. These are somewhat paler and less conspicuous because the bright colors are available everywhere 😊

Click on the 'Done' button to apply the previously selected color for this one field. All field colors are saved locally in this browser - this means that completely different colors can be selected individually on another device.

6.3 The page *Live Curve*

The 'Live curve' page can be called up via the navigation menu and starts with the display of the current measured temperature values over time the first time it is called up:



The diagram area automatically adapts to the available screen area. The axes are automatically adapted to the data ranges.

The data is retrieved freshly from the Raspberry Pi at the selected interval and thus obtained as the current value from the PiLogger. For this purpose, the measurement interval should be set to a correspondingly short time (page 'Settings 1') so that the displayed measurement points can actually change.

A legend of the active curves is displayed at the top right of the diagram area and, if the marker is active, the respective individual values at a selected time (see 6.4.2).

Below the diagram area is a line with setting options for filling the displayed curve, the display mean value factor and the update interval.

☒ Füllen
 Mittelwert: Werte
Intervall: ☐ kurz ☒ normal ☐ Benutzer sec

Below is a table of the available series of measured values:

Anzeige linke Achse:				Anzeige rechte Achse:			
Temp	<input checked="" type="checkbox"/> Akt			Temp	<input type="checkbox"/> Akt		
Wind	<input type="checkbox"/> Akt			Wind	<input type="checkbox"/> Akt		
Volt	<input type="checkbox"/> Akt			Volt	<input type="checkbox"/> Akt		
Amps	<input type="checkbox"/> Akt			Amps	<input type="checkbox"/> Akt		
Watt	<input type="checkbox"/> Akt			Watt	<input type="checkbox"/> Akt		
Ohm	<input type="checkbox"/> Akt			Ohm	<input type="checkbox"/> Akt		
Wh Dauer	<input type="checkbox"/> Bila	<input type="checkbox"/> Ertr	<input type="checkbox"/> Verb	Wh Dauer	<input type="checkbox"/> Bila	<input type="checkbox"/> Ertr	<input type="checkbox"/> Verb
Wh Tag	<input type="checkbox"/> BilTg	<input type="checkbox"/> ErtTg	<input type="checkbox"/> VerTg	Wh Tag	<input type="checkbox"/> BilTg	<input type="checkbox"/> ErtTg	<input type="checkbox"/> VerTg

When you first open this page, only the 'Temp' display for the left axis is active. The individual selection fields (checkboxes) can be activated or deactivated by clicking or tapping on them. If at least one measured value is selected for the right-hand axis, a second value axis is displayed on the right-hand side of the diagram area.

If an already activated measured value is activated on the other axis, it is automatically deactivated on the first axis.

All these settings are saved locally in the browser for this page.

This means that this page is started with the last selected settings the next time it is accessed.

The saved settings therefore apply to exactly this end device and the browser used. If the same page is called up on another device (or on the same device with a different browser), other individual settings are saved there.

This is done using the 'Web Storage' technology available in HTML 5, more precisely here using 'Local Storage', which has nothing to do with the 'cloud' and is actually more of a type of 'cookie'.



For this reason, either the 'Delete website data on exit' function must be deactivated in the browser or an exception must be added for the PiLogger WebMonitor address.

The operation of this page is basically the same as for the 'Diagrams' page - except for the setting of the update interval. For this reason, this setting is described here first and reference is made to the next chapter 6.4 'The diagrams page' for the other points.

6.3.1 Setting the update interval

The update interval is set to 'normal' by default:

Intervall:	<input type="radio"/> kurz	<input checked="" type="radio"/> normal	<input type="radio"/> Benutzer	<input type="text" value="5"/> sec
-------------------	----------------------------	---	--------------------------------	------------------------------------

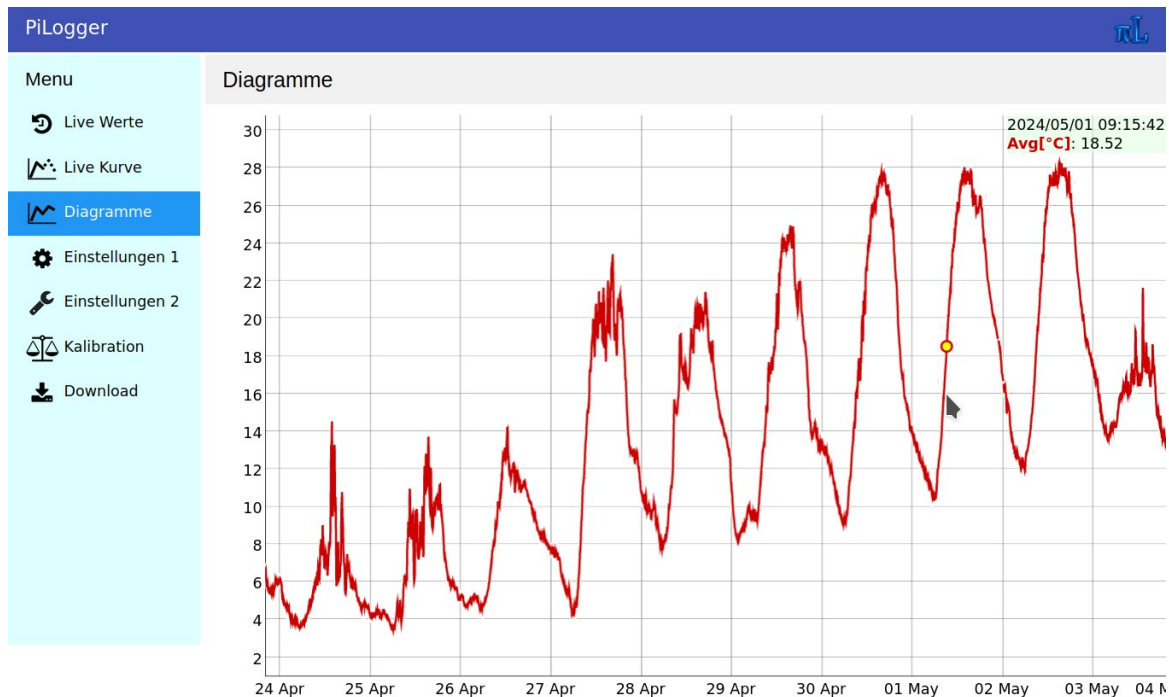
This corresponds to the 5 seconds shown on the right. With a click or tap on the 'short' button, you can quickly switch to 2 seconds.

If we click in the number field, we can directly enter a time in seconds, which must be in the range 1...120. We end the entry with an 'Enter' and the value is accepted and activated as the user value.

It makes sense that the update interval should not be shorter than the measurement interval of the PiLogger, which is set on the 'Settings 1' page, and should preferably be an integer multiple of this.

6.4 The page *Diagrams*

If the 'Diagrams' page is called up via the navigation menu, this page starts with the display of the average temperature values over time the first time it is called up:



The diagram area automatically adapts to the available screen area. The axes are automatically adjusted to the data ranges.

The data is taken from the current log file plus the previous log period. For this purpose, the 'showdata.csv' file is downloaded fresh from the Raspberry when the page is called up.

A legend of the active curves is displayed at the top right of the diagram area and, if the marker is active, the respective individual values at a selected time (see 6.4.4).

Below the diagram area, there are a number of settings.

We can access this area by scrolling, i.e. by moving the content of the page upwards.

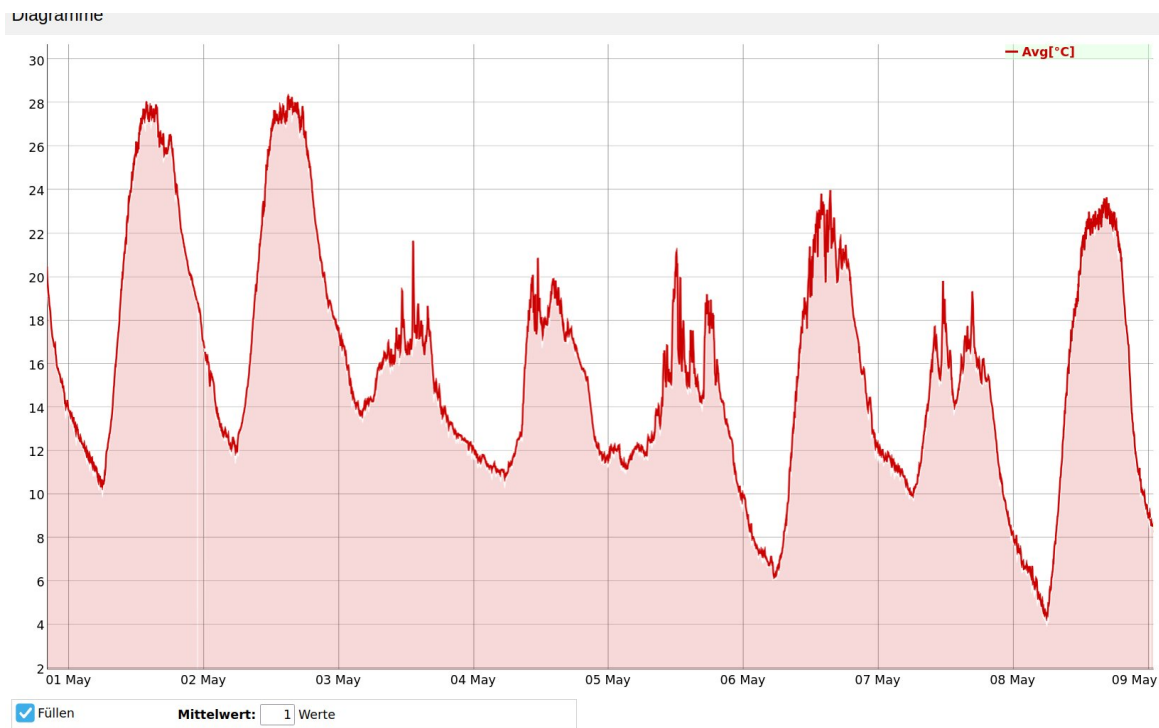
These setting options are described below.

6.4.1 Filling the curve

First, a line appears with a button 'Fill' and an input field 'Average value':

☐ Füllen **Mittelwert:** Werte

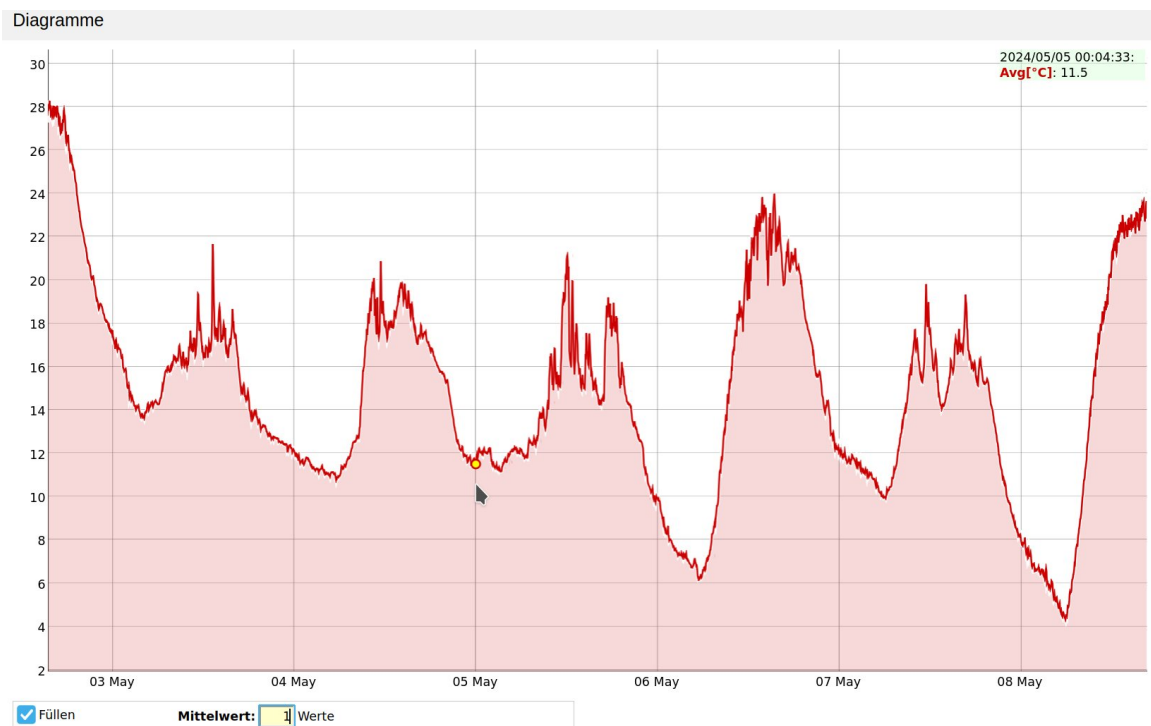
If we activate the 'Fill' option, the display of the measured value curve changes so that the area under the curve appears as a filled area:



The fill color is a lighter transparent variant of the curve color.
This setting is saved in 'local storage'.

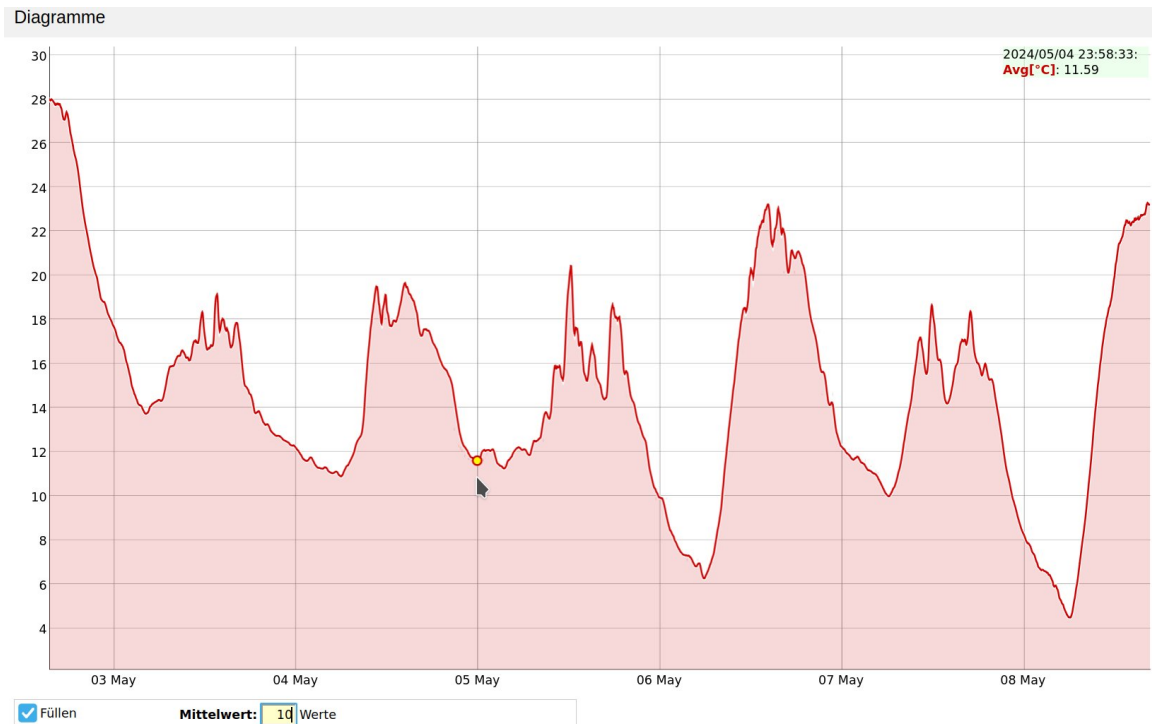
6.4.2 Using a moving average

The diagram curves can be temporarily smoothed with a moving average (the original data remains unchanged). To do this, we click in the small input field 'Average value':



We then enter a factor greater than '1' (no averaging), for example '10', and conclude the entry with 'Enter'.

This results in this representation:



Averaging is reset by entering '1'.

Integer values from 1 to 1000 can be entered.

This setting is saved in the 'local storage'.

6.4.3 Selecting the series of measured values to be displayed

If we scroll further down, i.e. to the lower, hidden page area, this table with buttons becomes visible:

Anzeige linke Achse:				Anzeige rechte Achse:			
Temp	<input checked="" type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max	Temp	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max
Wind	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max	Wind	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max
Volt	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max	Volt	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max
Amps	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max	Amps	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max
Watt	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max	Watt	<input type="checkbox"/> Avg	<input type="checkbox"/> Min	<input type="checkbox"/> Max
Ohm	<input type="checkbox"/> Ohm			Ohm	<input type="checkbox"/> Ohm		
Wh Dauer	<input type="checkbox"/> Bila	<input type="checkbox"/> Ertr	<input type="checkbox"/> Verb	Wh Dauer	<input type="checkbox"/> Bila	<input type="checkbox"/> Ertr	<input type="checkbox"/> Verb
Wh Tag	<input type="checkbox"/> BilTg	<input type="checkbox"/> ErtTg	<input type="checkbox"/> VerTg	Wh Tag	<input type="checkbox"/> BilTg	<input type="checkbox"/> ErtTg	<input type="checkbox"/> VerTg

On smaller screens (e.g. smartphone vertical), the two halves (left and right) appear one below the other.

Both halves show the same selection options. This means that each series of measured values can be activated for the left or right vertical axis.

If at least one series of measured values is activated for the right-hand axis, the second value axis is also displayed on the right-hand side of the diagram area. Clicking or tapping activates or deactivates the display of the respective series of measured values. If an already activated measured value is activated on the other axis, it is automatically deactivated on the first axis.

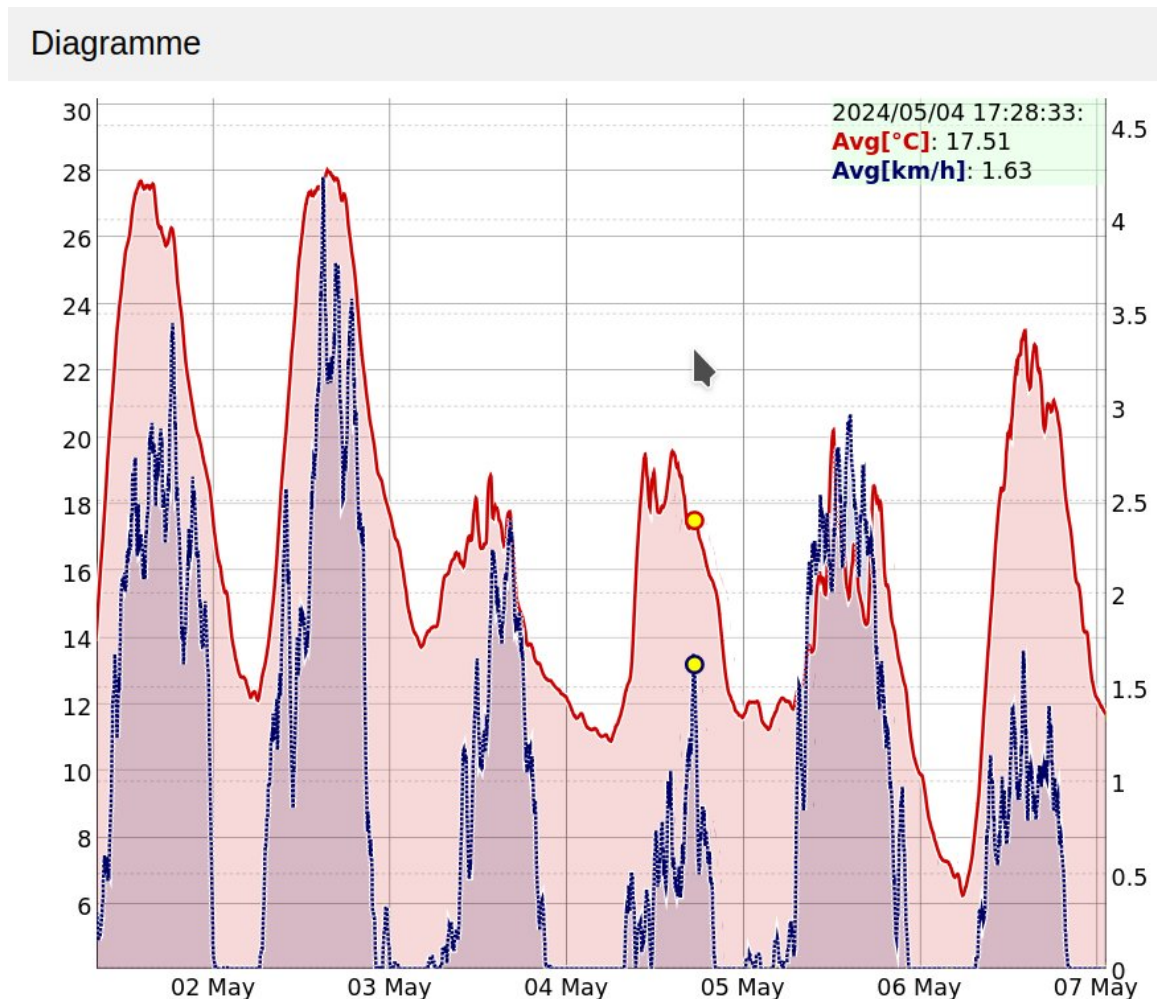
This setting is also saved in the 'local storage'.

6.4.4 Showing a single measured value

On screens with mouse operation, a marker in the form of a yellow-filled circle appears immediately when the mouse pointer is in the diagram area; this follows the horizontal mouse pointer position but remains on the measurement curve. If several measured value curves are displayed, each curve has its own marker. The respective individual measured values at the marker point are displayed at the top right in the legend together with the date and time.

On touch screens, this marker is activated with a short, light tap at a desired point in the diagram. The tap must not be too long and must not contain any movement, otherwise it will be interpreted as a gesture. Pressing down hard can also be misinterpreted as a movement.

Here is an example of a mouse pointer-controlled measured value display:



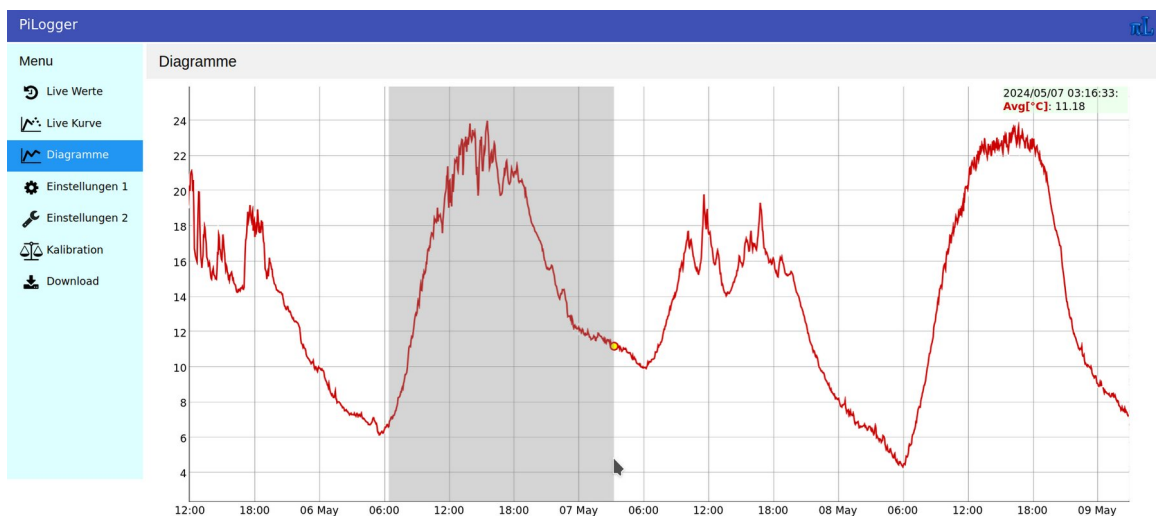
6.4.5 Panning and zooming

The displayed curve area within the diagram can be moved and zoomed. This is possible both with the mouse and with touch, but is very different.

6.4.5.1 Mouse operation

◆ Zooming

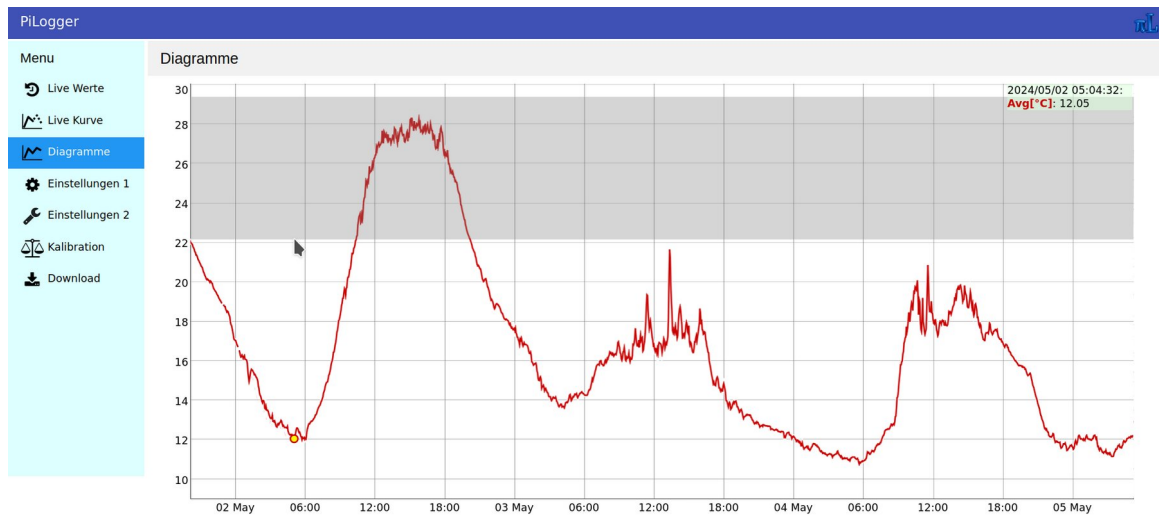
To zoom in on an area with the mouse, we move to a position for the desired start of the area to be zoomed - press and hold the left mouse button and drag the mouse pointer to the desired end of the zoom area. The area is highlighted in color. Then we release the left mouse button and the diagram is enlarged to the desired area.



The screenshot shows the marking of a time range. After zooming, it looks like this:



A vertical range, i.e. a range of values, can also be zoomed in the same way. Here the vertical marking of the range:



With this result:



◆ Reset zoom

Double-click in the diagram area to reset the zoom. The automatic full view of all data is displayed again.

◆ Moving

To move the diagram area (Pan), place the mouse pointer anywhere in the diagram area and press and hold the "Shift" key. If we now move the mouse pointer, we move the diagram area. After releasing the 'Shift' key, we have the normal movement of the marker with the mouse pointer again.

If the diagram is not zoomed, we can only move it horizontally. When zoomed, the movement is also possible vertically or in combination.

◆ Reset move

Here too, a double-click anywhere in the diagram area takes you back to the standard display.

6.4.5.2 Touch operation

With touch operation, the fluidity of the control depends heavily on the browser used. For example, the Firefox browser for Android always reacts to the two-finger gesture with a zoom change - but this is not what is meant here - which leads to severe delays or even an unintentional zoom of the entire page content. The best way to undo this is in the navigation menu area.

- ◆ Zooming

Zooming on a touch screen is done with a two-finger gesture. Simply place two fingers simultaneously in the diagram area and then move the fingers away from each other on the screen to enlarge the area under the fingers - or move them towards each other to reduce the area (pinch). This works both horizontally and vertically at the same time.

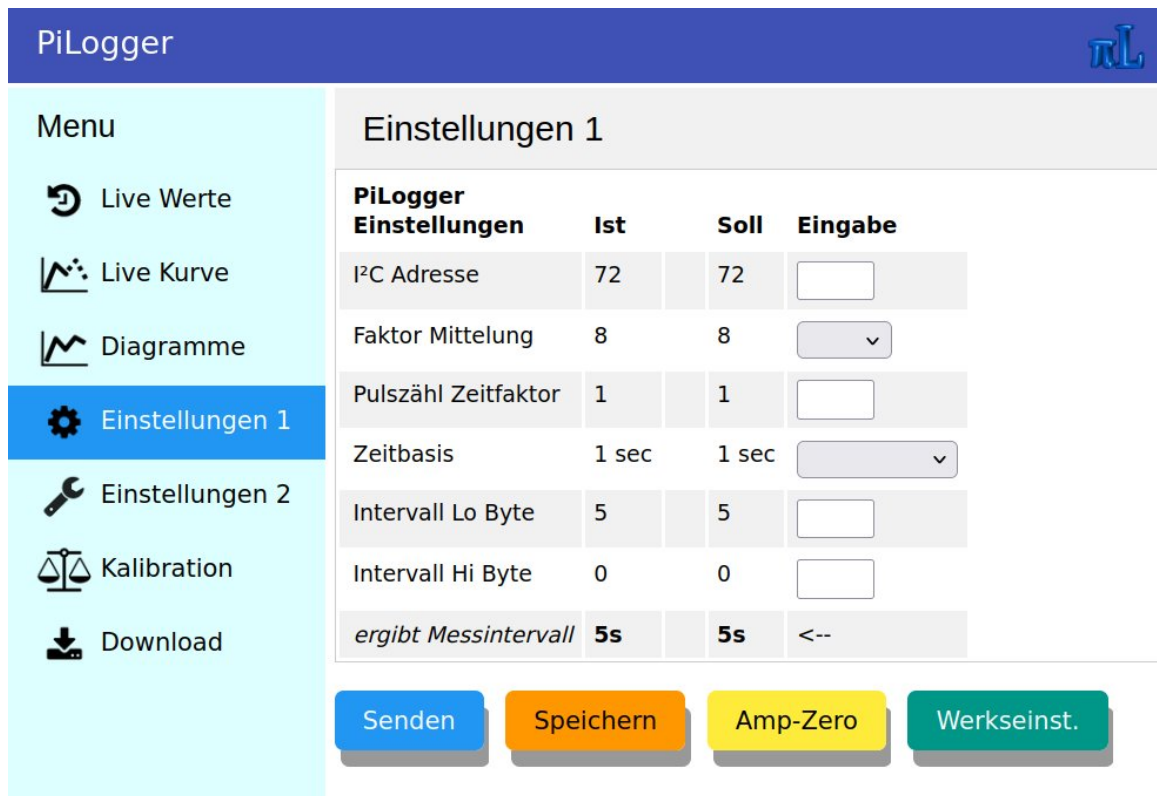
- ◆ Moving

Moving is done by keeping your finger on the screen and then moving the finger in any direction - in the diagram area. Two fingers can also be used for this. It can therefore also be combined with zooming.

- ◆ Resetting

Even with touch operation, zoom and shift are reset by double-tapping in the diagram area.

6.5 The page **Settings 1**



PiLogger Einstellungen	Ist	Soll	Eingabe
I ² C Adresse	72	72	<input type="text"/>
Faktor Mittelung	8	8	<input type="text" value="v"/>
Pulszahl Zeitfaktor	1	1	<input type="text"/>
Zeitbasis	1 sec	1 sec	<input type="text" value="v"/>
Intervall Lo Byte	5	5	<input type="text"/>
Intervall Hi Byte	0	0	<input type="text"/>
ergibt Messintervall	5s	5s	<--

Buttons: Senden, Speichern, Amp-Zero, Werkseinst.

All settings directly related to the PiLogger itself are managed on this page.

As soon as the window is opened, the values in the left-hand column 'Actual' are automatically read once by the PiLogger.

New values can be entered for the individual parameters in the 'Input' column on the right. The 'Target' column lists the values that will be used for the next 'Send' or 'Save'.

These are in detail:

- **I²C address**

This parameter is specified as a decimal value and is the bus address to which the PiLogger responds. This address may be in the range 4 to 124 (0x04 to 0x7C).

The default address is 72 (0x48). Only in exceptional cases should there be a reason to change this address - for example, if two PiLoggers are to be operated on one Raspberry Pi.

In order for a changed I²C address to be saved permanently in the PiLogger, you must then press 'Save' once - see below.

- **Averaging factor**

The PiLogger continuously calculates a moving average for each of the 5 retrievable measured values. The weighting factor used for this can be selected here.

Valid values for this parameter M are between 1 and 7, whereby the averaging factor is then $1/(2^M)$. The averaging factor can therefore be set between $\frac{1}{2}$ and $\frac{1}{128}$. The current measured value is weighted with this factor and added to the previous average value.

- **Pulse counting time factor**

A multiplication factor can be set here for the measurement time at the pulse counter input. With small measurement intervals in particular, many measurements with zero pulses would otherwise be generated, which would not provide a meaningful value for the wind speed, for example. If, for example, the measurement interval for voltage, current and power is set to $\frac{1}{4}$ second (250 msec), a factor of 40 can be used here to ensure that the values for wind speed are determined every 10 seconds.

- **Time base**

The PiLogger works with an internal time base that is used for the measuring interval. Here you can choose between 4 basic time units:

0 : 1 sec

1 : 250 msec

2 : 15.625 msec

3 : 1.953125 msec

This determines the total adjustable time range for the measuring interval and the resolution within it. With the following 2 values for the factor, a multiple of this time base of $1x \dots 65535x$ can be set as the measuring interval.

Important : The internal measuring rate of the PiLogger is set here - not the data request rate. The measured values are continuously averaged in the PiLogger. The measurement rate can and should be higher than the polling rate (time interval smaller). More on this under 6.6 'Settings 2 - Log factor'

The smallest measurement interval must not be less than 7.8 msec, i.e.

$4 \times 1.953125 \text{ msec} = 7.8125 \text{ msec}$. This corresponds to 128 measurements per second.

The maximum time interval that can be set is 65535 sec, which corresponds to 18 hours, 12 minutes and 15 seconds.

The WebMonitor software calculates the resulting measuring interval in seconds from the currently set values each time. You can therefore test the new measurement interval live by changing the values and sending them to the PiLogger and see the value in seconds. The LED on the PiLogger flashes once for each measurement (approx. 5 msec on).

- **Interval Low Byte**

Low-value byte of the factor for the measuring interval setting.

- **Interval High Byte**

High-value byte of the factor for the measuring interval setting.

The total factor is calculated as $H \times 256 + L$. The minimum value is 1.

- **Button 'Send'**

Newly entered values are sent to the PiLogger by clicking on the 'Send' button and take effect immediately. If the measuring interval is changed, this can be seen very clearly from the activity of the PiLogger LED.

- **Button 'Save'**

If the changed setting values are to be saved permanently so that the PiLogger works with these settings again after a power cycle, these values must be saved in the PiLogger's internal flash memory. To do this, click the 'Save' button once.

- **Button 'Amp-Zero'**

The button in the bottom line labeled 'Amp-Zero' sends a command to the PiLogger to set the current average value for the current as the zero value.



Attention: This command may only be used if there is really no current flowing through the sensor at the moment, i.e. nothing is connected to the screw terminals, for example.

This calibrates out sample scatter and other tolerances in the current measuring branch. Before use, the PiLogger should have been active for about 20 minutes so that it has settled thermally.

Pressing the 'AmpZero' button can be repeated as often as required.

If the result is stable and close to zero after a few minutes, this setting must be saved permanently in the PiLogger by pressing the 'Save' button so that this correction is also active after a restart.

This calibration is useful once during initial commissioning and, if necessary, if a clear misalignment is detected after some time in operation or the setup has changed.

- **Button 'Factor settings'**

In the event that the status of the PiLogger is unclear, for example due to extreme measurement intervals, this button can be used to send a command to the PiLogger to restore the default settings.

However, the PiLogger only loads these default settings when it is restarted - this means that the PiLogger must be switched off for a few seconds and then switched on again. To do this, the 'Shutdown' button under 'Settings 2' can be used to briefly disconnect the Raspberry and PiLogger from the power supply.

6.6 The page **Settings 2**

PiLogger

Menu

- Live Werte
- Live Kurve
- Diagramme
- Einstellungen 1
- Einstellungen 2
- Kalibration
- Download

Einstellungen 2

Raspberry Einstellungen	Ist	Soll	Eingabe
Messintervall (PiLogger Interrupt)	5s		
Log Faktor	36	36	<input type="text"/>
<i>ergibt Log-Intervall</i>	3m 0s	3m 0s	
Statistik Reset zum Log-Zeitpunkt	ja	ja	<input type="button" value="v"/>
Faktor Pulse 1 Sensorkonstante	0.36	0.36	<input type="text"/>
Einheit Pulse 1 Haupteinheit	km/h	km/h	<input type="text"/>
Faktor Pulse 2 (Neben)	0.1	0.1	<input type="text"/>
Einheit Pulse 2 Nebeneinheit	m/s	m/s	<input type="text"/>
Typ Temp Sensor	PTC Pt1000	PTC Pt1000	<input type="button" value="v"/>
Split Log-Datei	ja	ja	<input type="button" value="v"/>
Zeilenzahl für Split Log-Datei	3360	3360	<input type="text"/>
<i>Neuer Wert ergibt:</i> 7d 0h 0m 0s Zeit 880 kB			

Speichern

Reset Energiezähler

Split Log jetzt

Neustart

Herunterfahren

This page contains settings for the Python program itself, so it is the Raspberry Pi (ESP32) settings page.

These settings are saved in a separate file 'PiLogger_Config.json' in the program directory.

Some of the PiLogger settings are also saved there so that, for example, the program can also address the PiLogger if the I²C address is changed.

Here too, the current values in the left-hand column 'Actual' are retrieved from the Raspberry when the page is called up. The new values to be saved are listed in the 'Target' column on the right and the values are changed accordingly in the 'Input' column.

The first line shows the time currently set for the PiLogger's measurement interval. This is the basis for the logger settings.

The settings in detail:

- **Log Factor / Log-Interval**

The measuring interval is multiplied by the 'Log factor' set here to set the log interval. The resulting log interval is displayed in the line below.

This interval is the time interval between the measuring points of the recorded measured values.

With this time interval, the Raspberry Pi (or ESP32) fetches the data from the PiLogger One via I²C. The I²C query of all data with 2 blockread operations takes 14.7 msec at 100 kBaud. Since the program loop (depending on the computer speed) consumes up to approx. 75 msec itself and the SD card write process also takes time, the log interval should not be shorter than approx. 100 ms.

In the case of short intervals, please test the reliability with the computer actually used before continuous use. If necessary, the underlying measurement interval can also be extended.

The measured values are saved with a time stamp in the 'logdata.csv' file in the '/home/user/pilogger' directory.

- **Reset statistics at log time**

This field is a yes/no decision by means of a drop-down list.

This function means that every time an entry is made in the log file, a command is also sent to the PiLogger to reset the statistics values 'Average', 'Minimum' and 'Maximum' to the current measured values.

These values thus become the statistical values that apply to exactly one log interval. This is particularly important for the energy meters, which add up the product of power times time (interval) per log interval.

For this reason, the default setting for this item is 'active' - i.e. 'yes'.

- **Factor Pulse 1**

The conversion factor for the pulse count input is set here.

The basic unit is pulses per second. The measuring time interval is also taken into account by the PiLogger WebMonitor software. The sensor constant must be entered here, taking into account the unit entered in the next field.

For example, for a wind sensor, the distance per pulse multiplied by the time factor of the desired display unit.

In the example shown, this is 0.36 km/h - which corresponds to 0.1 m per pulse per 3600 seconds.

This field applies to the main display (large) on the 'Live values' page.

- **Unit Pulse 1**

This is the associated unit of measurement for converting the pulse counter readings.

This field applies to the main display (large) on the 'Live values' page.

- **Factor Pulse 2**

As under 'Factor Pulse 1', but here for the secondary display (small) on the 'Live values' page. Only the value for the main unit is saved in the log file.

- **Unit Pulse 2**

This field applies to the secondary display (small) on the 'Live values' page.

- **Type Temp Sensor**

This is a drop-down list with currently 6 entries:

NTC 10k B3928

NTC 10k B3477

NTC Tabelle

PTC Pt1000

PTC KTY81-110

PTC Tabelle

Depending on the temperature sensor connected, the necessary conversion of the measured values into temperature values is defined here.

These conversion algorithms are available in the WebMonitor software.



Attention: The jumper on the PiLogger One must be set according to the class of the connected sensor. It is set to the 'NTC' position at the factory.

A sensor with 10 kOhm at 25°C is suitable as an NTC, a platinum sensor with 1000 Ohm at 0°C is provided as a PTC.

See also chapter 7.3 'Temperature sensor input'.

- **Split log file**

This field is a yes/no decision using a drop-down list.

If this function is activated, a new log file is automatically created after the number of logged lines specified in the next field.

If this function is not activated, the log file is continued indefinitely. The log file can still be split manually using the 'Split log now' button - see below.

- **Number of lines for split log file**

The number of lines at which a new log file is started is specified here. Each newly created log file is given the time stamp of the new creation in its name.

Depending on the log interval, very large files can quickly be created, which then take a correspondingly long time to transfer for the diagram display. An even larger file is actually transferred for the diagram display, which also contains the data from the previous log file. This means that immediately after the automatic creation of a new log file, at least the previous period can be displayed as a curve. The line below shows the time period and file size for each log file with the input values.

- **Button 'Save'**

Pressing this button saves the values from the 'Target' column in the 'PiLogger_Config.json' configuration file.

In addition, the program is reinitialized with these values.

- **Button 'Reset energy meters'**

This button immediately sets all energy meters to zero, i.e. both the continuous meters and the daily meters. Including the associated time counters.

- **Button 'Split log now'**

If this button is pressed, a new log file is created immediately. The previous data file can still be downloaded and displayed - see 6.8 'The page Download'.

The data in the previous log file is now the display preprocessing for the time diagrams.

If this part is no longer to be displayed, for example because the settings have been changed significantly (other pulse units, other calibration values, etc.), the 'Split log now' button must be pressed again. This creates a useless log file without measured values or only very few measured value lines. If necessary, it can be deleted on the 'Download' page.

- **Button 'Restart'**

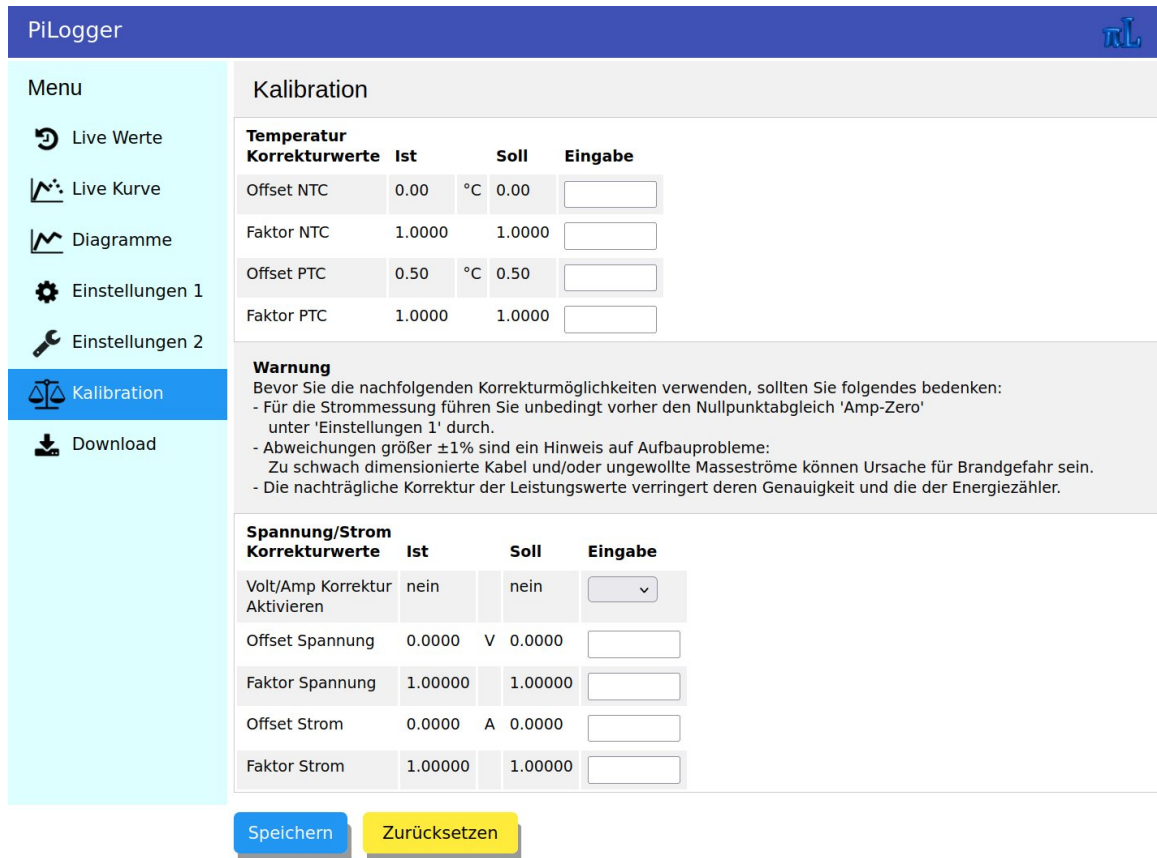
This button offers the possibility to restart the Raspberry in a controlled manner. In contrast to disconnecting the power supply, this method prevents data loss or even rendering the SD card unusable.

- **Button 'Shut down'**

This button offers the option of shutting down the Raspberry in an orderly manner. In contrast to disconnecting the power supply, this method avoids data loss or even rendering the SD card unusable.

The power supply can be safely disconnected after the Raspberry Pi LED flashes the last time (4 times evenly).

6.7 The page *Calibration*



PiLogger

Menu

- Live Werte
- Live Kurve
- Diagramme
- Einstellungen 1
- Einstellungen 2
- Kalibration**
- Download

Kalibration

Temperatur Korrekturwerte	Ist	Soll	Eingabe
Offset NTC	0.00	°C 0.00	<input type="text"/>
Faktor NTC	1.0000	1.0000	<input type="text"/>
Offset PTC	0.50	°C 0.50	<input type="text"/>
Faktor PTC	1.0000	1.0000	<input type="text"/>

Warnung
 Bevor Sie die nachfolgenden Korrekturmöglichkeiten verwenden, sollten Sie folgendes bedenken:
 - Für die Strommessung führen Sie unbedingt vorher den Nullpunktgleich 'Amp-Zero' unter 'Einstellungen 1' durch.
 - Abweichungen größer $\pm 1\%$ sind ein Hinweis auf Aufbau Probleme:
 Zu schwach dimensionierte Kabel und/oder ungewollte Masseströme können Ursache für Brandgefahr sein.
 - Die nachträgliche Korrektur der Leistungswerte verringert deren Genauigkeit und die der Energiezähler.

Spannung/Strom Korrekturwerte	Ist	Soll	Eingabe
Volt/Amp Korrektur Aktivieren	nein	nein	<input type="text" value="v"/>
Offset Spannung	0.0000	V 0.0000	<input type="text"/>
Faktor Spannung	1.00000	1.00000	<input type="text"/>
Offset Strom	0.0000	A 0.0000	<input type="text"/>
Faktor Strom	1.00000	1.00000	<input type="text"/>

Speichern Zurücksetzen

The 'Calibration' page offers the option of setting correction values for the temperature measurement and also for the voltage and current measurement. This is basically a simple linear conversion with a shift (offset) and a gradient (factor).

This correction calculation is always useful for temperature measurement and enables a classic 2-point calibration according to the definition of the Celsius scale:

To do this, first expose the temperature sensor to an ice water bath (assuming good, watertight insulation of the connections) and take the resulting display value negative (times -1) as the 'offset'. The temperature sensor is then exposed to boiling water and the 'factor' is set to '100/display value'.

The default setting is for both temperature sensor classes:

'0' for the offset and '1' for the factor → this means: no change.

The correction of the voltage and current values, on the other hand, should only be activated if the following is considered:

- If the amp values deviate, always carry out the zero point adjustment 'Amp-Zero' under 'Settings 1' first - see page 77.

- If the deviations of the voltage measurement are greater than $\pm 1\%$ of the measured value, this is an indication of a possible structural problem:
 - Connection cables that are too weak lead to voltage drops and can pose a safety problem due to self-heating. The PiLogger measures the voltage between its 'OUT+' and 'OUT-' terminals. A voltage drop on the supply cable can cause this voltage to deviate from the voltage at the generator or battery. Make sure that the cables are not overloaded and that the contacts are well connected (do not generate any significant voltage drop). The cables should also always be kept as short as possible. If, for example, the battery voltage is to be used as a reference, it makes sense to mount the PiLogger closer to the battery than to the generator.
 - Additional currents on the negative cable of the battery, which do not also flow on the positive cable measured by the PiLogger, lead to a voltage increase at the measuring point of the PiLogger - a ground offset.
- If the voltage and current value correction is activated, the power values calculated directly by the PiLogger are also corrected accordingly. This means additional division and multiplication operations, which reduce the accuracy of the power values. And thus, of course, also the accuracy of the energy meters.

In general, an optimized structure is preferable to a subsequent correction calculation.

Therefore, the default setting here is 'no' for 'Correction calculation deactivated'.

- **Button 'Save'**

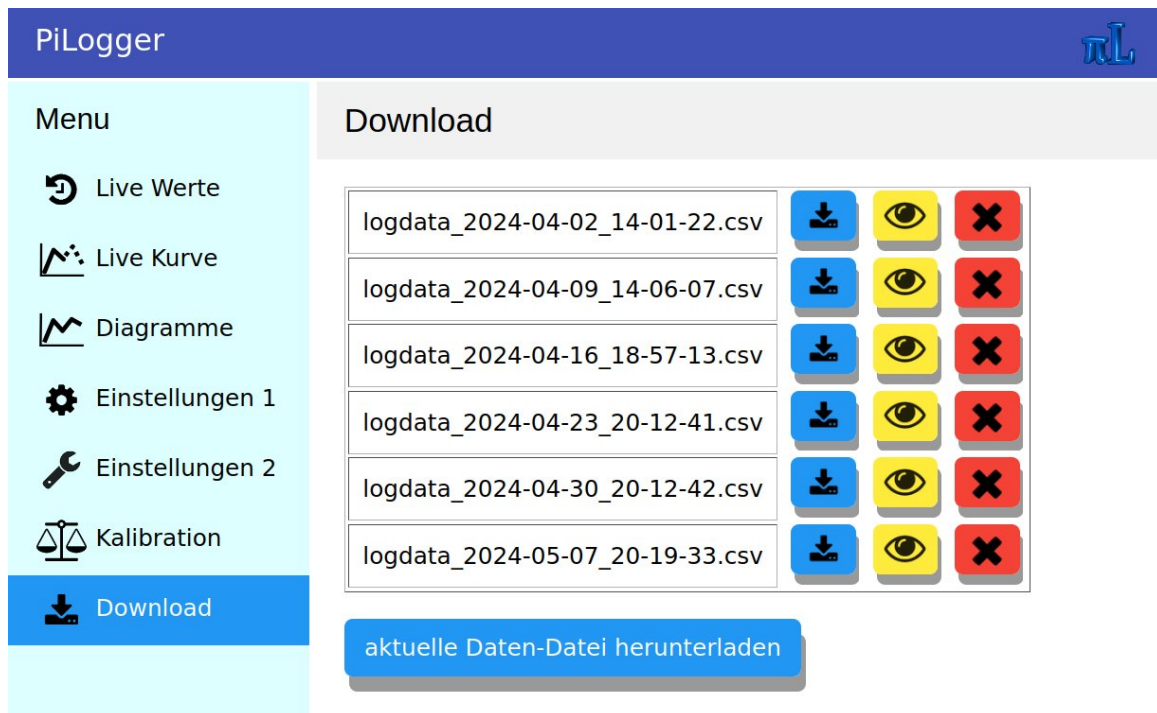
Pressing this button saves the values from the 'Target' column in the 'PiLogger_Config.json' configuration file.

In addition, the program is reinitialized with these values.

- **Button 'Reset'**

This button resets all values on this page to their default values and saves them *immediately*.

6.8 The page **Download**



This page allows you to download, view and delete the log data from the Raspberry (or ESP32).

All log files are '.csv' files. However, in order to load them into a German-language spreadsheet program, a text program must be used to replace '.' with ','. The '.csv' files use the American-English representation for the number display in order to be compatible with the diagram library. Without this substitution, this leads to a misinterpretation during import. The semicolon (;) is used as a field separator, which is usually recognized by the import filter, but sometimes has to be set separately.

When the page is called up, the directory of all available 'logdata_xxx.csv' files is freshly queried from the Raspberry (ESP32).

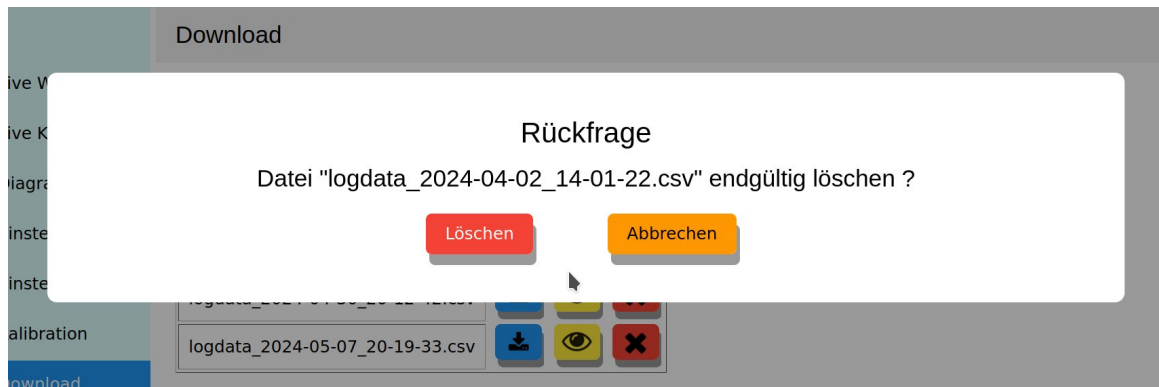
Behind each of these files there are 3 buttons:

The first blue area with the download symbol starts a normal file download for this file. Depending on the configuration of the calling browser, a dialog window appears asking whether and where the file should be saved.

The second yellow area with the eye symbol allows you to view this file using the diagram function. As usual, the Diagrams page is called up and the average temperature is displayed by default. This allows data from previous periods to be viewed.

The third red area with the X symbol deletes this data file from the SD card of the Raspberry Pi (ESP32). As this process is final, a confirmation prompt is first interposed via an overlay window to enable an abort in the event of an accidental

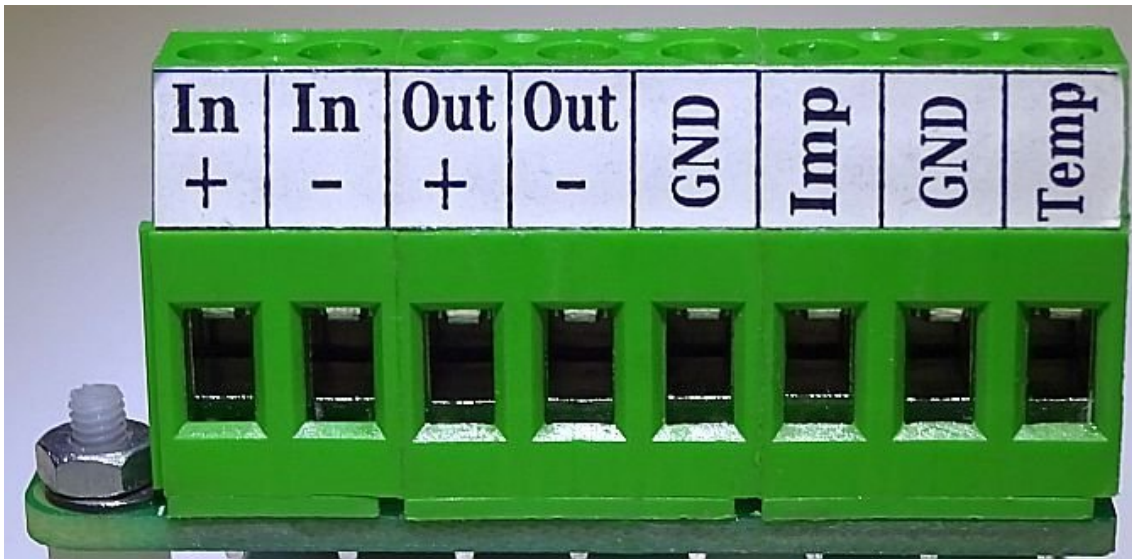
operation:



Under the file directory there is also the 'Download current data file' button. This allows you to download the currently used log file 'logdata.csv'. As this file is currently being used for logging, it is of course not possible to delete it, and it is also not useful to display it at this point, as this is done via the normal call-up of the diagram page.

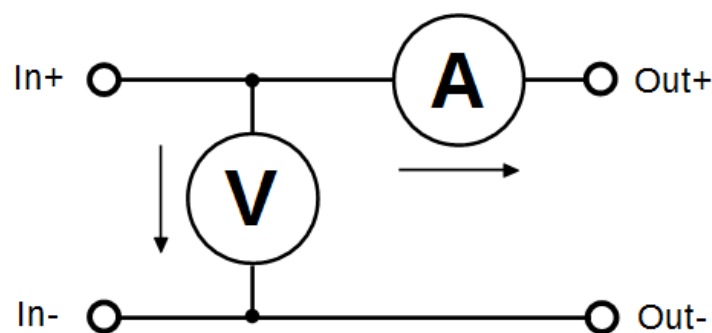
7 Connection terminals

The measurement inputs of the PiLogger are all located on the 8-pin terminal block:



7.1 DC voltage and current measurement

The first 4 terminals on the left are wired for measuring voltage and current in such a way that the first two terminals form the input and the next two terminals form the output for looping through. The two terminals marked '-' are connected directly to each other, while the current sensor is connected in series to the terminals marked '+':



The internal resistance of the current sensor is really low at approx. 2 mΩ, but 24 mV is lost at 12 A, which corresponds to a power of 288 mW.

This is therefore a systematic measurement error of this arrangement.

It is greatest at low voltages and high currents. In the above example at 12 A and 3 V, however, it is less than 1% at 0.008.

The fact that cables with a sufficient cross-section must be used for currents of up to 12 A is also explained by this consideration of the loss of seemingly small resistances.

Even if according to VDE a 0.75 mm² single cable (90°C) can be loaded with up to 12 A, this load capacity is reduced to 35% at an ambient temperature of 85°C! If the setup is heated, e.g. by solar radiation, you need significantly thicker cables - in this case 4 mm² - to ensure a safe setup.

The PiLogger terminals can accommodate these 4 mm² (stranded wire with ferrules).

A cable with a copper cross-section of 4 mm² and a length of 4 m already has a resistance of approx. 17.2 mΩ and thus causes a loss of almost 2.5 W at 12 A ! And this happens again in the return conductor, making a loss of almost 5 W !

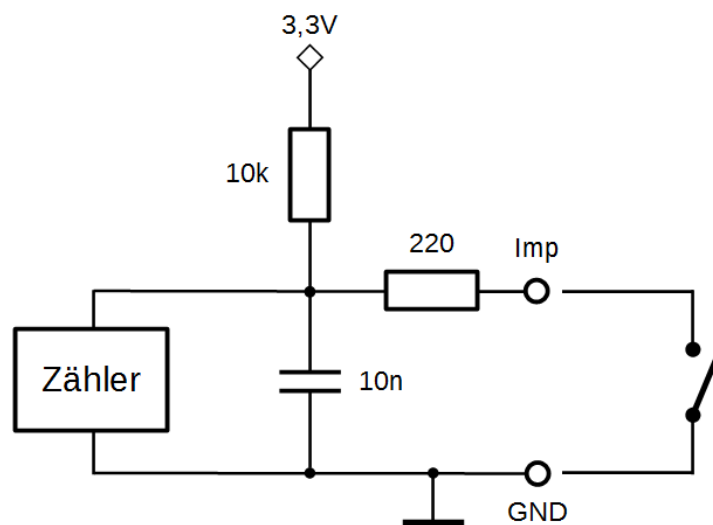
So please keep the cable length as short as possible.

7.2 Pulse counting input

The next two terminals are labeled 'GND' and 'Imp'. They form the connections for the pulse counter input.

The input is designed for use with mechanical switches, such as reed contacts, as well as open collector outputs (OC). In the idle state, it shows around 3.1V at the 'Imp' connection against the reference ground 'GND'. If the 'Imp' connection is short-circuited to 'GND', this edge is counted as a pulse.

Its main elements look like this:



The pull-up resistor forms a time constant with the capacitor for the recovery of the input for the next negative edge. The series resistor in the input forms a low-

pass filter with the capacitor. Both together ensure debouncing and a reduction in sensitivity to interference pulses.

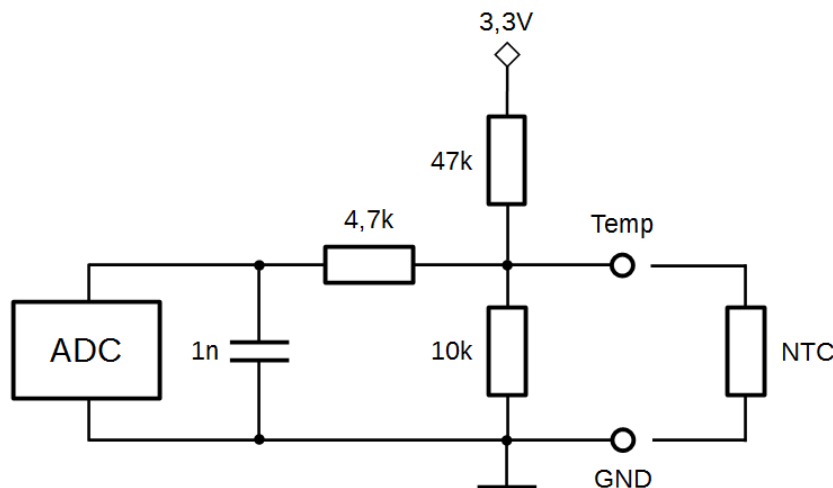
At 50% pulse-pause ratio, the maximum frequency is around 2 kHz. The resistance of the closed switch (or open collector) must be below 3 k Ω in order to enable reliable low detection.

7.3 Temperature sensor input

The two terminals on the right are labeled 'GND' and 'Temp'. They are intended for the connection of a temperature sensor.

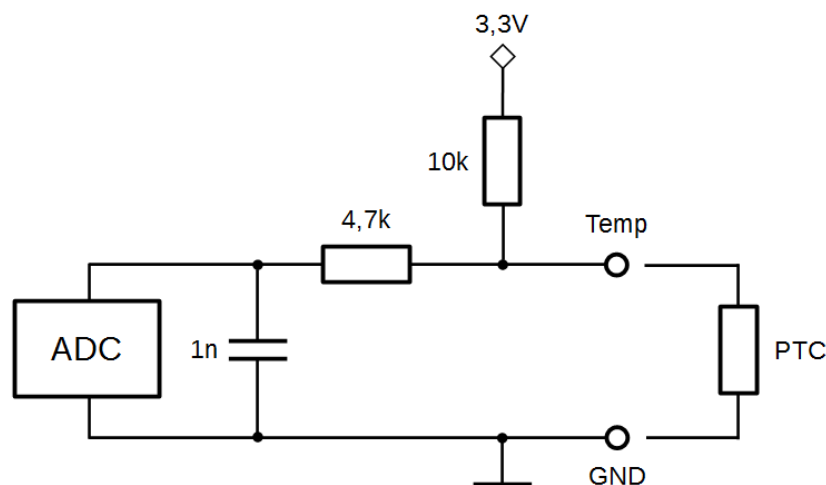
The input can be set to 2 types of temperature sensors using the shorting jumper JP1:

1. JP1 in position 1-2 : **NTC**



In this position, the circuit is designed for the connection of an NTC (resistor with negative temperature coefficient) with a nominal resistance of 10 k Ω .

2. JP1 in position 2-3 : **PTC**

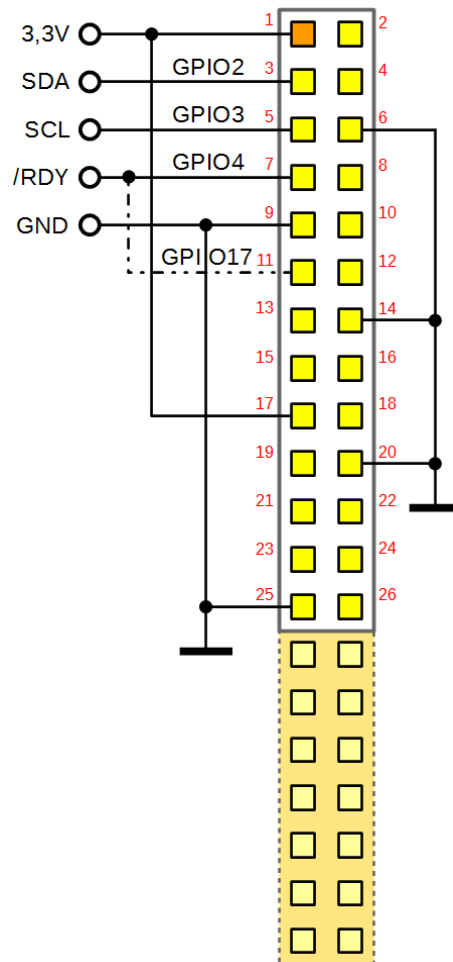


In this position, the circuit is designed for the connection of a PTC (resistor with positive temperature coefficient) with 1 k Ω nominal resistance.

More details on temperature sensors follow in chapter 10.

8 Pin assignment GPIO strip

The PiLogger only uses 5 of the first 8 pins of the Raspberry Pi GPIO strip. Further pins are connected - but only to increase contact reliability (3.3V & GND) and as an alternative (/RDY).



With the signal '/RDY' (= busy) the PiLogger One informs the Raspberry Pi about its measuring activity. The status corresponds to that of the LED on the PiLogger. A 'High' (+3.3V) means that a measurement is currently running - the LED lights up. When the signal changes to 'Low', the measurement is complete - the LED goes out.

This negative falling edge is configured as an interrupt on GPIO4 (pin 7) of the Raspberry Pi in order to retrieve and log the fresh measured values with the correct time stamp.

9 Application programming interface

The PiLogger One uses the I²C interface to communicate with the Raspberry Pi (or ESP32). The Raspberry Pi is the master and the PiLogger is the slave in this connection.

The I²C bus (or SMBus) uses the hardware protocol defined by the specification (http://www.nxp.com/documents/user_manual/UM10204.pdf).

The SMBus is a further development with some additions that are not or only partially supported by pure I²C devices (ICs).

This protocol is handled in Linux by the kernel driver 'i2c-dev' and made accessible as a device with the 'i2c-dev' module from the 'user land', i.e. the application area.

The 'python3-smbus' wrapper then gives Python access to the I²C bus via the kernel driver. Alternatively, the 'I2C-Tools' can be used from the command line (see appendix).

The latter two provide a set of functions (methods) with which all necessary operations can be carried out on the I²C bus or SM bus.

Below is the message catalog that defines the communication with the PiLogger One. As the I²C bus or SM bus is byte-oriented, all accesses are controlled via virtual 1-byte addresses - the registers.

The PiLogger One has two classes of registers:

Configuration registers and measured value registers.

In addition, there are direct commands that are triggered by writing to the relevant register (with and without password).

9.1 Configuration registers

The following table lists all configuration registers of the PiLogger One.

These registers are in the range 0x01 to 0x16.

Their respective meaning has already been described in chapter 6.5.

The registers from 0x01 to 0x06 are valid addresses for write access.

The corresponding registers from 0x11 to 0x16 are the respective addresses for read access.

These registers each hold 1 byte. This means that exactly 1 data byte is expected when writing and exactly 1 data byte is returned when reading.

In addition, there are 3 commands (0x07 , 0x0D , 0x0F) that are triggered by writing to this register if the expected fixed value - the password - is also sent as a value (data byte). This is intended to reduce the probability of inadvertent execution, e.g. due to disrupted communication.

Lesen/Schreiben	Register	1 Byte		
Konfiguration	Write	Read	Wert	
I ² C Slave Adresse	0x01	0x11	default 0x48	0x04...0x7C
Messzyklus Basis	0x02	0x12	0 = 32768 = 1 sec 1 = 8192 = 250 ms 2 = 512 = 15,625 ms 3 = 64 = 1,953125 ms	
Messzyklus Faktor 1	0x03	0x13	1 – 255 low	1...65535
Messzyklus Faktor 2	0x04	0x14	0 – 255 high	
Mittelungsfaktor	0x05	0x15	1 – 7	2 ^x : 2...128
Zeitexpansion Pulszählung	0x06	0x16	1 – 255	
Amp Avg zu Null setzen	0x07	-	0x44	Nur Schreiben
Auf Werkseinstellung	0x0D	-	0xAB	Nur Schreiben
Konfig Speichern (Flash)	0x0F	-	0x55	Nur Schreiben

The usage in Python looks like this:

The first instruction is at the very beginning of the program and integrates the 'SMBus' module. The second instruction assigns the value of the bus to be used to the variable 'pillogger'.

```
import smbus
```

```
pillogger = smbus.SMBus(1) ; for Raspberry Pi Typ A (256 Mbyte): smbus.SMBus(0)
```

Reading a 1-byte register:

```
wert = pillogger.read_byte_data(address, register)
```

Example:

```
timebase = pillogger.read_byte_data(0x48, 0x12)
```

Writing a 1-byte register:

```
pillogger.write_byte_data(address, register, databyte)
```

Example:

```
pillogger.write_byte_data(0x48, 0x02, 0x01)
```

9.2 Measured value registers

The following table lists all measured value registers of the PiLogger One. These registers are in the range 0x20 to 0x88.

The registers in the range from 0x20 to 0x63 without [0x28,0x38,0x48,0x58] are valid access addresses for read access.

These registers each hold 2 bytes. This means that exactly 2 data bytes are returned when reading. These 2 bytes each represent a word - i.e. a 16-bit number.

The measured values in the register range 0x20 to 0x27 (power) must be combined in pairs to form 4-byte values, i.e. 32-bit numbers.

Registers 0x70 and 0x80 each initiate the reading of an entire 32-byte block. Although these 2 blocks overlap, so that the values for temperature and pulse counter are read twice when both blocks are read, this query method is the fastest and most effective if all values are required.

In addition, there are 7 commands (0x28...0x88) that are triggered by writing to this register. Writing to these registers resets the statistics values of the associated base registers (to the current value).

Nur Lesen	Register	2 Byte			Avg, Min & Max
Messwerte	Momentan	Mittelwert	Minimum	Maximum	Reset
Leistung	0x20 0x21	0x22 0x23	0x24 0x25	0x26 0x27	0x28
Temperatur	0x30	0x31	0x32	0x33	0x38
Impulseingang	0x40	0x41	0x42	0x43	0x48
Spannung	0x50	0x51	0x52	0x53	0x58
Strom	0x60	0x61	0x62	0x63	0x68

Block Lesen	Befehl	Register			Reset
Block1 Lesen 16x Word	0x70	0x20...0x43			0x78
Block2 Lesen 16x Word	0x80	0x30...0x63			0x88

The usage in Python looks like this:

The first instruction is at the very beginning of the program and integrates the 'SMBus' module. The second instruction assigns the value of the bus to be used to the variable 'pillogger'.

```
import smbus
```

```
pillogger = smbus.SMBus(1) ; for Raspberry Pi Typ A (256 Mbyte): smbus.SMBus(0)
```

Reading a 2-byte register:

```
wert = pilogger.read_word_data(address, reg)
```

Example:

```
tempmittel = pilogger.read_word_data(0x48, 0x31)
```

Reading a 4-byte value:

```
wertL = pilogger.read_word_data(address, reg)
wertH = pilogger.read_word_data(address, reg+1)
wert = wertH * 65536 + wertL
```

Example:

```
LeistungMaxL = pilogger.read_word_data(0x48, 0x26)
LeistungMaxH = pilogger.read_word_data(0x48, 0x27)
LeistungMax = LeistungMaxH * 65536 + LeistungMaxL
```

Reading a 32-byte block:

```
werte = pilogger.read_i2c_block_data(address, blockreg) ; block 1 = 0x70 , block 2 = 0x80
```

Example:

```
werte = pilogger.read_i2c_block_data(0x48, 0x70)
```

Reset the statistical values of a block:

```
pilogger.write_byte(address, blockreg) ; block 1 = 0x78 , block 2 = 0x88
```

Example:

```
pilogger.write_byte(0x48, 0x78)
```

9.3 Value ranges and scaling

The measurement results are output by the PiLogger as raw values with different value ranges. The following table shows the value range and the associated scaling factors for each channel:

Messwert	Format	Wertebereich	Standard Maßfaktoren			
Leistung	signed 32 bit	-2147483648... +2147483647	918/2147483648	W	-918...+918	1/2339236
Temperatur	unsigned 16 bit	0...+65535	0,00915541	mV	0...600mV	1/109,2267
Impulseingang	unsigned 16 bit	0...+65535	0,2	m/s		
Spannung	unsigned 16 bit	0...+65535	0,000915541	V	0...60V	1/1092,267
Strom	signed 16 bit	-32768... +32767	0,000466933	A	-15,3...+15,3A	1/2141,634

These are the respective full scale limits, the usable measuring ranges are always slightly smaller (see 'Technical data').

The pulse counting channel is an exception. Here, the raw value is always pulses

per measuring interval - the interpretation depends on the sensor used (see 6.6 'Settings 2 - Pulse factor'). The standard measurement factor here is only the default from the WebMonitor software, which can of course be adjusted.

The temperature measurement is an internal voltage measurement for the PiLogger. The conversion to temperature values depends on the temperature sensor used. The conversion therefore takes place in the WebMonitor software.

10 Temperature sensor tables

The PiLogger One temperature sensor input is optimized for two common temperature sensors. One is a so-called Pt1000, a sensor based on platinum (Pt) with a positive temperature coefficient (PTC) and a typical resistance of 1000 Ω at 0°C in the Pt1000 version. This type is quite accurate and standardized.

The other is a so-called NTC, i.e. a temperature-dependent resistor with a negative temperature coefficient, of which there are unfortunately many different variants. In the 'NTC' setting, the PiLogger input is optimized for a type that is quite common in heating systems and has a typical resistance of 10 k Ω at 25°C (e.g. Epcos B57891S0103H008). An NTC typically covers a larger resistance range, which makes the measurement somewhat easier - but the characteristic is significantly more non-linear, which in turn requires more approximation effort.

Both types have further advantages and disadvantages, which we will not go into here, but the high measurement resolution of the PiLogger (16 bit) means that very accurate temperature measurements can be achieved with both types.

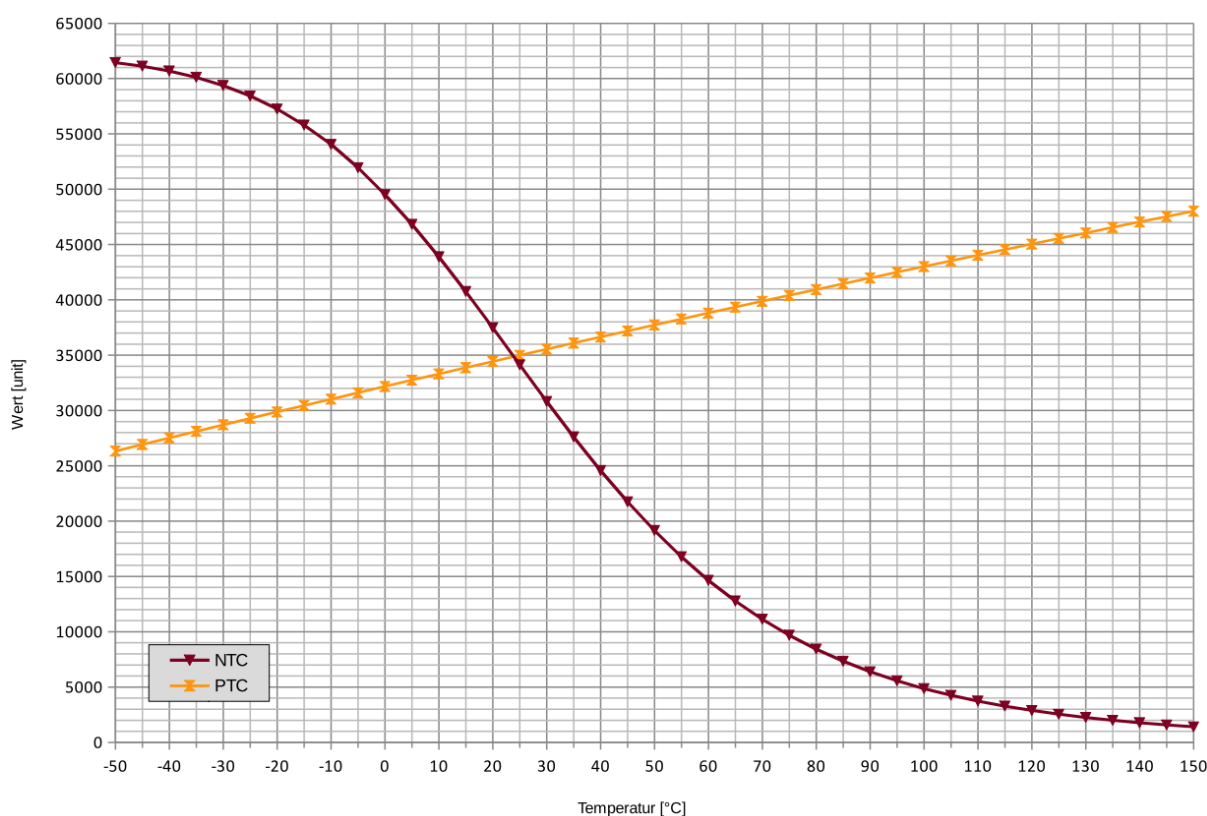
The following table shows the resistances at temperatures in the range -55°C to 155°C for the two typical representatives:

Pt1000				NTC 10k			
T [°C]	R [Ω]		T [°C]	R [Ω]		T [°C]	R _{nom} [Ω]
-55	783,30		55	1213,24		55	3039
-50	803,14		60	1232,46		60	2536
-45	822,96		65	1251,64		65	2128
-40	842,74		70	1270,79		70	1794
-35	862,50		75	1289,92		75	1518
-30	882,23		80	1309,01		80	1290
-25	901,93		85	1328,07		85	1100
-20	921,60		90	1347,10		90	942
-15	941,25		95	1366,09		95	809
-10	960,86		100	1385,06		100	697
-5	980,44		105	1403,98		105	604
0	1000,00		110	1422,87		110	525
5	1019,53		115	1441,72		115	457
10	1039,03		120	1460,54		120	400
15	1058,50		125	1479,31		125	351
20	1077,94		130	1498,04		130	308
25	1097,35		135	1516,74		135	272
30	1116,74		140	1535,38		140	240
35	1136,09		145	1553,99		145	213
40	1155,42		150	1572,55		150	189
45	1174,73		155	1591,06		155	168
50	1194,00						

With the wiring from chapter 7.3, this results in these value tables:

Pt1000					NTC 10k				
T[°C]	Wert		T[°C]	Wert	T[°C]	Wert		T[°C]	Wert
-55	25718		55	38262	-55	61682		55	16767
-50	26319		60	38800	-50	61441		60	14645
-45	26918		65	39334	-45	61114		65	12772
-40	27513		70	39866	-40	60679		70	11126
-35	28106		75	40396	-35	60107		75	9680
-30	28695		80	40923	-30	59367		80	8423
-25	29281		85	41447	-25	58424		85	7328
-20	29864		90	41968	-20	57247		90	6382
-15	30444		95	42487	-15	55797		95	5563
-10	31021		100	43002	-10	54054		100	4854
-5	31595		105	43516	-5	51942		105	4249
0	32167		110	44026	0	49509		110	3726
5	32735		115	44534	5	46824		115	3272
10	33300		120	45039	10	43884		120	2878
15	33863		125	45541	15	40748		125	2539
20	34422		130	46040	20	37484		130	2245
25	34979		135	46537	25	34124		135	1988
30	35533		140	47031	30	30815		140	1763
35	36085		145	47522	35	27607		145	1566
40	36633		150	48010	40	24561		150	1395
45	37179		155	48496	45	21740		155	1244
50	37722				50	19144			

As a diagram, it looks like this:



These tables can be used to look up the associated temperature for a PiLogger measured value in the application software (look up table). Intermediate values must be calculated by linear interpolation.

The PiLogger WebMonitor software takes a different approach. It calculates the temperature value for each measured value using an approximation function (formula for an approximate value for the given curve).

For the Pt1000 it uses:

Measured value M in 25000...50000:

$$T = M^2 / 23006845.6 + M / 167.473355 - 237.0911$$

(quadratic approximation)

This is not sufficient for the NTC 10k. The curve changes so much over the temperature ranges that the deviation from the original curve would be too great with a single approximation formula. For this reason, the range from -40°C to 140°C is divided into 3 sub-ranges, for each of which a different approximation formula is used:

With $m = M / 65536$;

Measured value M in 1763...8423 (+140...+80 °C) :

$$T = m^3 * -50245.929 + m^2 * 16224.33 + m * -2068.25 + 184.196$$

Measured value M in 8424...54054 (+80...-10 °C) :

$$T = m^3 * -247.0087 + m^2 * 394.7776 + m * -305.093 + 111.6473$$

Measured value M in 54055...60679 (-10...-40 °C) :

$$T = m^3 * -18593.11 + m^2 * 47031.885 + m * -39846.685 + 11292.757$$

(cubic approximation)

This is a lot of computing work, especially for the NTC, but no problem for the Raspberry Pi.

The accuracy of the temperature measurement is therefore already very good, but it still depends crucially on the individual sensor used. The deviations can quickly amount to +/- 3° or more. A calibration should therefore be carried out for accurate absolute measurements.

For example, a 2-point calibration against a known good (calibrated) liquid thermometer (alcohol column).

Or the classic 2-point calibration with an ice water bath (0°C) and boiling water (100°C) - the definition of the Celsius scale. For this, the correction value is determined at 0°C, which must be added to the raw measured value so that the display value is correctly 0°C (the offset) and then the linear correction value with which the intermediate value must be multiplied so that the display at 100°C also fits. This should enable an accuracy of 0.1°C to be achieved in the range -10°C to +120°C.

11 Technical data

Dimensions : 56 x 47 x 29 mm (L x W x H)

Weight : 29 g

Power supply : 3.3 V from Raspberry Pi via pin connector

Current consumption : 5 ... 6 mA

Measurement duration approx. 84 ms (Python loop; depending on the computing speed of the guest computer)

Query of all values (2 blocks) : ca. 14,7 ms (@100 kbaud , BCM2708)

Interval range : 100 msec ... 18h 12min 15sec

Measurement ranges:

Voltage : 0 ... 50 V DC voltage (max. 60 V, resolution < 1 mV)

Current : -12 ... +12 A direct current (max. +/- 15 A, resolution < 0.5 mA)

Power : -600 W ... +600 W (max. +/- 900 W)

Pulse counter : 0 ... 65535 Cnt/Interval

Temperature : depending on the sensor used (see chap. 10), resolution < 0.01 K

12 Calibration

Although the PiLogger is built with close-tolerance components and uses a 16-bit A/D converter, it is not a calibrated measuring instrument.

The PiLogger is not factory calibrated to any standard.

For precision measurements, each unit should be calibrated against a suitable measurement standard.

The resulting correction values can then simply be incorporated into the software running on the Raspberry Pi (or ESP32) (e.g. in the Python program of the WebMonitor software) to ensure that the values displayed or stored in the log file are as accurate as possible.

As described in Chapter 6.7 'The page *Calibration*', the WebMonitor software already offers an option for a simple linear correction of the measured values in the user interface.

This is usually completely sufficient for the temperature values, for example, as the characteristics of the temperature sensors used are already well approximated with the help of the typical curves, meaning that only the sensor and PiLogger scatter need to be corrected.

The following describes exemplary recipes for calibrating the 4 basic measurements.

12.1 Temperature measurement calibration

If the temperature sensor used is waterproof, calibration according to the classic Celsius scale is recommended. The deviations of this historical scale are less than 0.06 K compared to the definition of temperature via the Boltzmann constant introduced in 2019 in the range up to 300 °C. The deviations caused by deviating air pressure during measurement are significantly greater.

The historical Celsius scale uses the melting point of water (0 °C) as the lower fixed point and the boiling point of water (100 °C) as the upper fixed point. And this is at normal pressure. For an open vessel, this means that the surrounding air pressure must be 1.01325 bar or 1013.25 hectopascals.

Procedure:

- Operate the PiLogger with connected temperature sensor and display option for about 20 minutes to reach a steady-state thermal condition.
- In the meantime, prepare a water bath with ice cubes in an open container. The sensor must be fully immersible in the water between the ice cubes. The ice cubes themselves are too cold and deeper water is too warm. A thermally steady state is also important here.

- Now insert the sensor completely into the meltwater and observe the temperature display. After some time, a stable value close to 0 °C should be reached.
Note the actual value.
- Take the noted value as a negative value (times -1) and enter it as an offset on the Calibration page of the WebMonitor software (for the sensor type used).
Immediately after saving, the display under 'Live values' should actually be 0 °C. If the ice bath still contains sufficient ice - i.e. the melt water can still be used as a reference - any remaining deviation can be reduced even further.
- Now remove the sensor from the ice bath and prepare a container with boiling water instead - such as an open kettle.

- Now immerse the sensor completely in the boiling water. The sensor must not come into contact with the base or the heating element !

Caution: Risk of scalding!

It is best to hold the sensor by the connection cable with long pliers !

The sensor should be held as horizontally as possible, but completely covered in a higher layer of water.

Observe the temperature display and note the value that remains stable at around 100 °C after a while.

- Calculate the correction factor using the noted display value:

Factor = $100 / \text{display value}$

Enter the factor for the sensor type used on the 'Calibration' page in the WebMonitor and save. The display under 'Live values' should now actually show 100 °C. If the deviation is still unsatisfactory, determine a new factor again:

$\text{NewFactor} = 100 / \text{Display value} \times \text{OldFactor}$

And enter and check the new factor.

After carrying out this procedure, an accuracy of +/- 0.1 °C should be achieved, especially with a Pt1000 sensor.

The deviations that occur in practice during measurement, for example due to draughts or solar radiation, are quickly 30 times greater.

12.2 Pulse counting calibration

As the name suggests, this is a counter.

It is about recording pulses per time unit. The accuracy therefore essentially depends on the accuracy of the time generator. In the PiLogger, this is a crystal-controlled The quartz is a so-called clock crystal with 32.768 kHz and an accuracy of +/- 20 ppm (**p**arts **p**er **m**illion) - i.e. +/- 0.002 %, corresponding to around +/- 0.7 Hz.

Although there is no adjustment option at this point, the overall accuracy can be corrected in conjunction with the sensor used.

In the WebMonitor software, on the 'Settings 2' page, there is the sensor constant (see 6.6) - the 'Pulse factor'.

This is therefore a linear factor that is typical for the sensor used (design-related). By adjusting this factor, any deviation of the measured values from the expected values can be corrected.

The prerequisite for this is, of course, a test arrangement with known, reliable expected values.

For a flow sensor, for example, a set water flow rate can be checked by measuring the amount of water over a period of time which is also measured. The measuring standards used here are a measuring vessel of known volume (bucket) and a precise stopwatch.

Or for a wind sensor a known wind speed in a wind tunnel... If a calibrated second wind measuring device is available and both devices can be exposed to the same wind flow without falsification, the display of the PiLogger can be adjusted to this reference using the 'Pulse factor'.

12.3 Voltage measurement calibration

The PiLogger WebMonitor software also offers the option of correcting the measured voltage values with offset and factor.

See chapter 6.7, in particular the notes on possible set-up problems as the reason for a measured value deviation.

The PiLogger measures the voltage at the 'Out+' and 'Out-' terminals.

If the displayed voltage is to be compared with a reference measuring device, for example a multimeter with high accuracy (< 0.5 %), the measuring tips of this reference device must be applied to these PiLogger terminals. The measurement at a more distant location - for example at the battery terminals - is distorted by the voltage drop across the cables.

The higher the current currently flowing through the cables, the greater the resulting voltage difference between the measurement locations.

In some cases, however, it is precisely this systematic measurement difference that can be the motivation for a measured value correction:

If the goal is to accurately monitor the battery condition, the battery voltage is the more important value for the energy balance. If the aim is to monitor the power generated by the generator (wind, solar), the generator voltage is the more interesting value. If at all possible, this should be taken into account when choosing the location of the PiLogger and the cables should always be as short as possible and the cable cross-section as large as possible.

The calibration process basically looks like this:

- Zero point adjustment:

Remove the cables from the 'In+', 'In-', 'Out+' and 'Out-' terminals.

Connect the 'Out+' and 'Out-' terminals together with a shorting jumper.

The voltage display should now be stable close to zero, i.e. approx. 0 ... 10 mV.

If this is not the case, there may be a problem with the power supply of the Raspberry Pi (or ESP32)

→ As a rule, no offset should be necessary under these test conditions.

- Linear deviation:

Connect a stable voltage of preferably 30 V (half the maximum voltage) from a laboratory power supply to the 'Out+' and 'Out-' terminals.

Measure the voltage at the PiLogger terminals with a known good (calibrated) measuring device and fine-tune if possible.

Calculate the correction factor :

$\text{Factor Voltage} = U_{\text{Reference reading}} / U_{\text{PiLogger reading}}$

If no laboratory power supply unit is available, the battery can be used as an alternative in idle mode (nothing else connected) - this means that the voltage is not at half full scale, but in the most interesting range.

Please note: The result can only be as good as the reference measuring device. The reference measuring device should be as accurate and calibrated as possible.

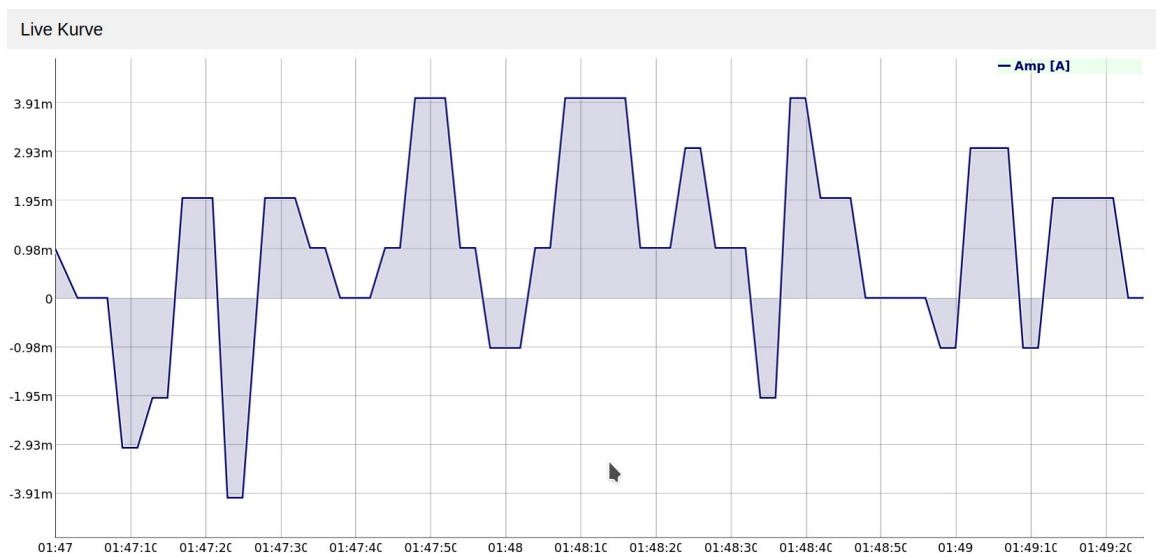
12.4 Current measurement calibration

The measured current values can also be corrected in the PiLogger WebMonitor software using an offset and factor.

The PiLogger works with a Hall sensor for current measurement. The analog output signal must have exactly half the output voltage at zero current and the analog-to-digital converter must output exactly zero at this voltage. Both deviations must be adjusted to each other. This may need to be repeated after some time due to drift and external magnetic fields.

For this purpose, the PiLogger has the 'Amp-Zero' function, which can be executed via the WebMonitor software - see chapter 6.5 .

During execution, no current must flow through the sensor - for example, nothing must be connected to the 'In+' terminal. When 'Amp-Zero' is executed, the current average current value is taken and set to zero.



If the adjustment is successful, the live value for the current is between -10 mA and +10 mA. This can be easily observed on the 'Live curve' page.

This zero point adjustment must always be carried out before any calibration using offset and factor.

The current calibration procedure is then as follows:

- **Current offset**

Normally, the adjustment using 'Amp-Zero' should be sufficient here.

In some cases, however, a (predominantly) constant current - for example for a DC/DC converter to supply the Raspi - is to be compensated. In this case, this current must be entered as a negative value (times -1) and saved.

- **Current factor**

To calibrate the linear deviation of the measured value, a constant current must first be generated in the medium measuring range.

1. To do this, a suitable constant voltage source (laboratory power supply or battery) is connected to the 'In+' and 'In-' terminals - for example 12 V.
2. For a reference current of approximately 5 A, for example, a high-load resistor with 2.2 Ω is now required.

Attention : *With these example values, the resistor must be able to handle 65.5 W and will get hot!*

The resistance is now measured with a known accurate measuring device, i.e. the resistance in ohms is determined exactly.

3. The resistor is now connected as a load to the 'Out+' and 'Out-' terminals.
4. Now, with a known accurate measuring device and with the voltage source switched on, the voltage actually present is measured directly at the terminals of the load resistor.

With these 2 measured values, we can now calculate the actual flowing reference current (Ohm's law):

$$I_{\text{Reference}} = U_{\text{Resistor}} / R_{\text{measured}}$$

Note: If the reference current is to be as even a value as possible - i.e. exactly 5 A in the example above - then an adjustment can be made here by changing the voltage (laboratory power supply). However, this is only helpful for comparison purposes.

5. The live value for the current is now monitored and read using the PiLogger WebMonitor.

We calculate the current correction factor as follows :

$$\text{Factor Current} = I_{\text{Reference}} / I_{\text{Reading}}$$

We enter this factor on the 'Calibration' page, set the value for 'Activate volt/amp correction' to 'Yes' and save everything.

6. Now check the live current value again and, if necessary, repeat the procedure to improve the result.

13 Integration into smart home systems

There are basically two ways to integrate the PiLogger One into smart home systems:

Direct - the PiLogger is operated as an extension on the Raspberry Pi on which the Smart Home system is running and is therefore an I²C multi-sensor. This assumes that the PiLogger cabling is possible in the vicinity of the Raspberry Pi.

Via network - the PiLogger is operated on a smaller, separate guest computer with the WebMonitor and is accessible in the local network.

From version 0.16 of the WebMonitor for Raspberry Pi or 0.4 for ESP32 there is a special additional data query path /rawdata/ which returns the data in an easily transferable format.

(So upgrade the PiLogger WebMonitor software if necessary).

This is a simple form of REST interface and is supported by many smart home systems.

This approach requires one more small computer, but offers maximum flexibility.

There are a number of different smart home solutions in the Raspberry Pi environment.

Here are a few examples - without any claim to completeness:

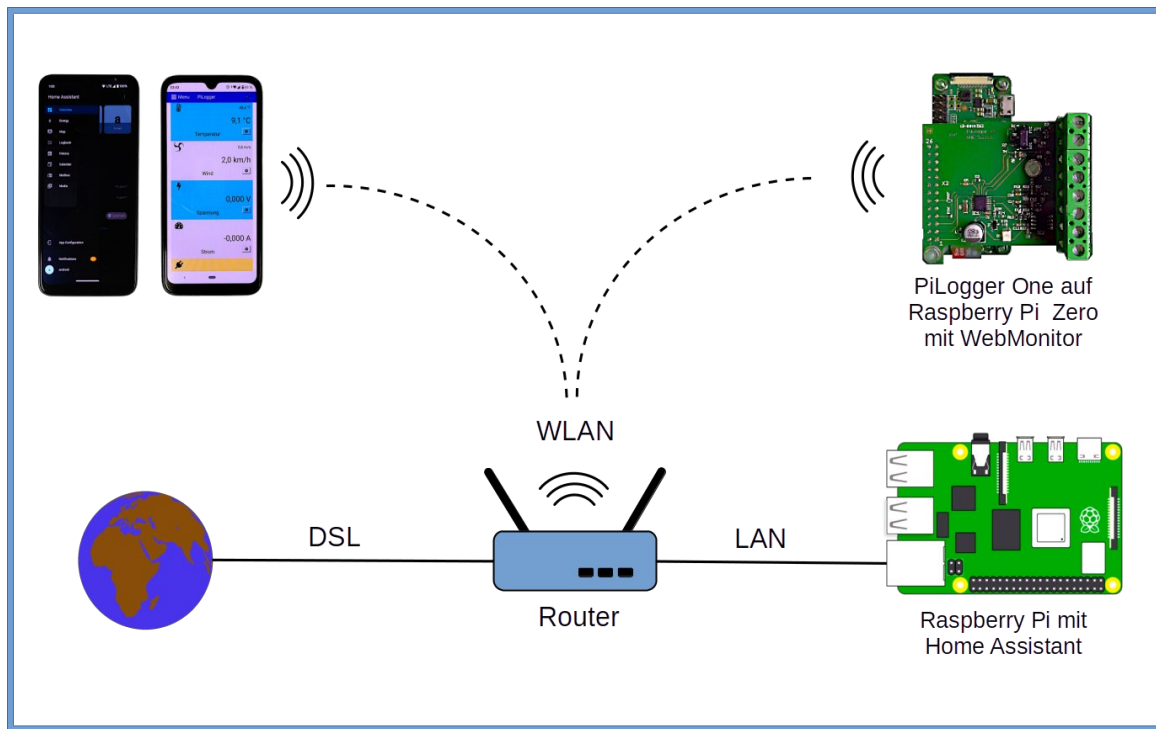
- Home Assistant
- ioBroker
- FHEM
- openHAB
- Gladys Assistant
- nymea
- RaspberryMatic
- Homebridge

The following are instructions for a Home Assistant connection via the network. Further instructions will follow as soon as available ...

13.1 Home Assistant

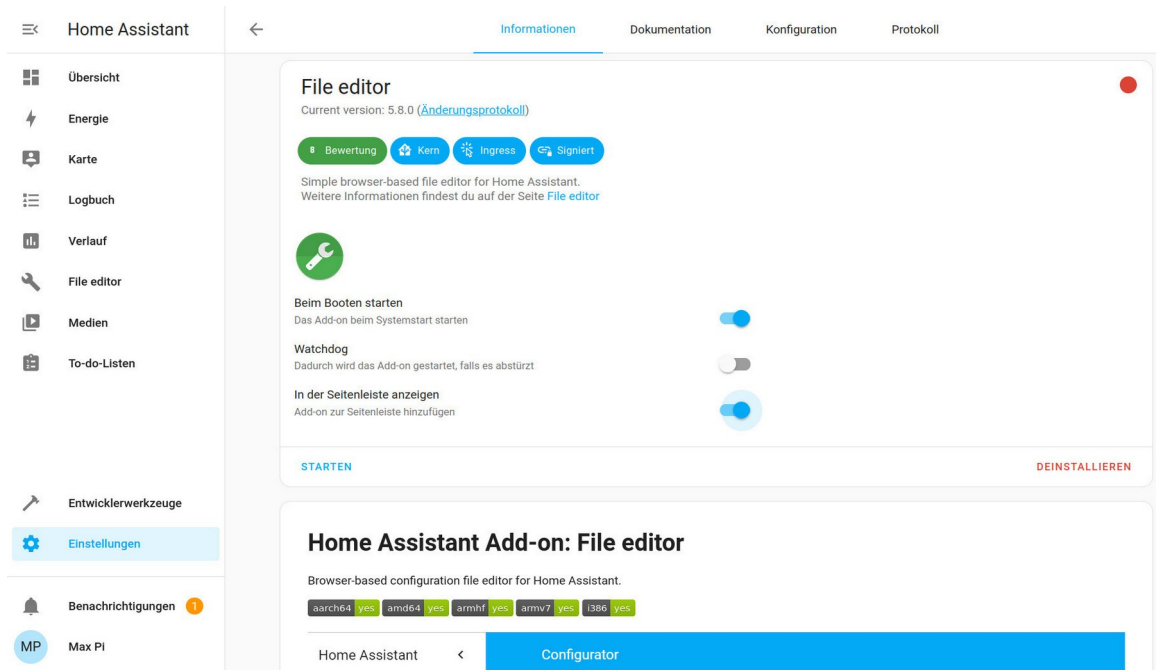
If the PiLogger One is operated on a Raspberry Pi or an ESP32 with the WebMonitor, the WebMonitor provides a web interface that can be called up with any device in your own home network using the web browser.

The browser constantly retrieves the live data from the Raspi (or ESP32) - *this can also be done by an existing 'Home Assistant' instance.*



This means that automation systems in Home Assistant can also use the measurement data from the PiLogger as a basis - for example, to retract the awning if the wind is too strong.

To integrate the PiLogger, the 'configuration.yaml' file must be edited in Home Assistant. This can be done in the HA web interface with the 'File editor'. In the current version 2024.3.3, this add-on is not on board by default and must be installed manually.



Now this text block must be inserted at the end of the 'configuration.yaml' file:

```
rest:
- resource: http://192.168.178.20:8080/rawdata/
  scan_interval: 10
  sensor:
  - name: "PiLo01 Temperatur"
    unique_id: "pilo_temp"
    value_template: '{{value_json.PiLoTemp1}}'
    device_class: "temperature"
    unit_of_measurement: "°C"
  - name: "PiLo02 Wind"
    unique_id: "pilo_wind"
    value_template: '{{value_json.PiLoWind1}}'
    device_class: "wind_speed"
    unit_of_measurement: "km/h"
  - name: "PiLo03 Spannung"
    unique_id: "pilo_volt"
    value_template: '{{value_json.PiLoVolt}}'
    device_class: "voltage"
    unit_of_measurement: "V"
  - name: "PiLo04 Strom"
    unique_id: "pilo_amps"
    value_template: '{{value_json.PiLoAmps}}'
    device_class: "current"
    unit_of_measurement: "A"
  - name: "PiLo05 Leistung"
    unique_id: "pilo_watt"
    value_template: '{{value_json.PiLoWatt}}'
    device_class: "power"
    unit_of_measurement: "W"
  - name: "PiLo06 TagesBilanz"
    unique_id: "pilo_dayenergy"
    value_template: '{{value_json.DayEnergy}}'
    device_class: "energy_storage"
    unit_of_measurement: "Wh"
  - name: "PiLo07 TagesErtrag"
    unique_id: "pilo_dayharvest"
    value_template: '{{value_json.DayHarvest}}'
    device_class: "energy"
    unit_of_measurement: "Wh"
  - name: "PiLo08 TagesVerbrauch"
    unique_id: "pilo_dayconsumption"
    value_template: '{{value_json.DayConsumption}}'
    device_class: "energy"
    unit_of_measurement: "Wh"
  - name: "PiLo09 DauerBilanz"
    unique_id: "pilo_permenergy"
    value_template: '{{value_json.PermEnergy}}'
    device_class: "energy_storage"
    unit_of_measurement: "Wh"
  - name: "PiLo10 DauerErtrag"
    unique_id: "pilo_permharvest"
    value_template: '{{value_json.PermHarvest}}'
    device_class: "energy"
    unit_of_measurement: "Wh"
  - name: "PiLo11 DauerVerbrauch"
    unique_id: "pilo_permconsumption"
    value_template: '{{value_json.PermConsumption}}'
    device_class: "energy"
    unit_of_measurement: "Wh"
  - name: "PiLo12 Dauer Zeit"
    unique_id: "pilo_permenertime"
    value_template: '{{value_json.PermEnerTime}}'
    device_class: "duration"
    unit_of_measurement: "min"
```

This text block can be downloaded here: [HA Configuration](#)

Of course, the actual IP address of your own PiLogger in the home network must be entered in the second line 😊

Note: This also means that it must be specified in the router that the PiLogger is always permanently assigned this one IP address.

The polling rate is specified with 'scan_interval' in seconds. It should be adapted to your own requirements.

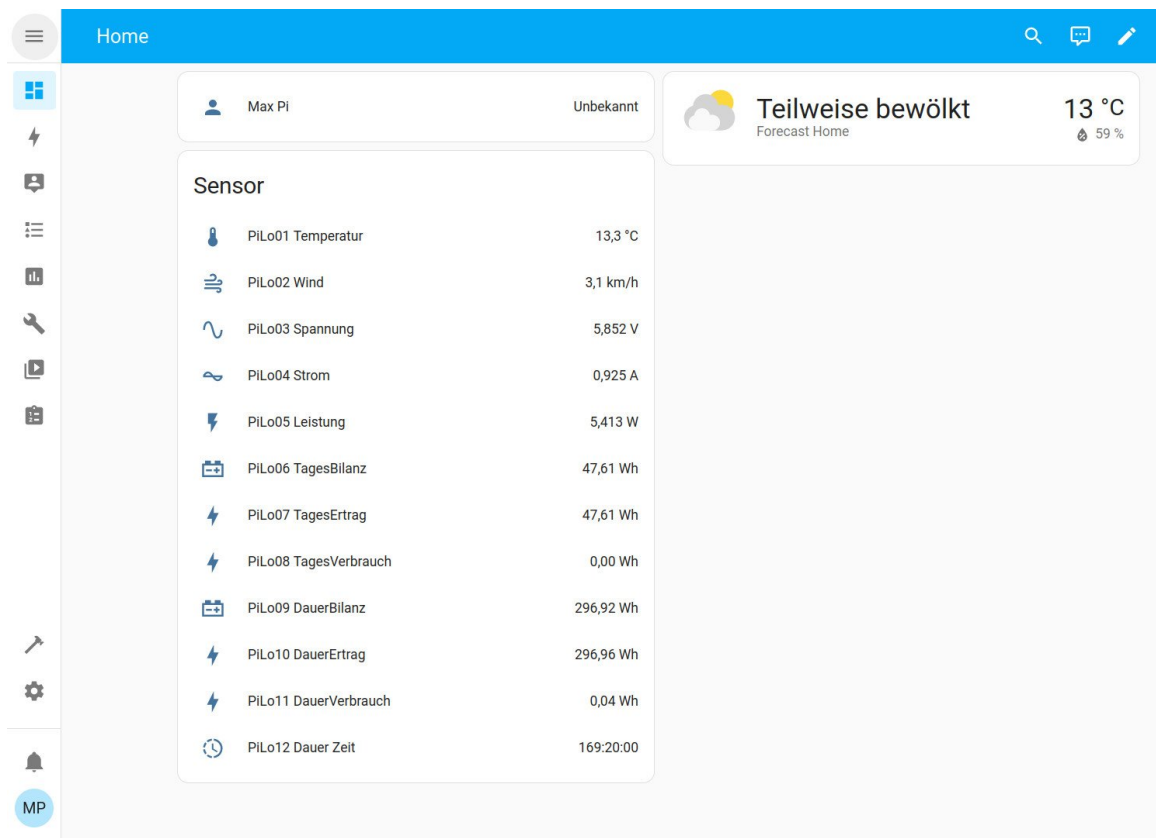
You can also specify meaningful names for the various measured values here - but these can also be changed later in HA.

The file must now be saved and closed. Home Assistant must then be restarted:

- Settings
- click on the 3 dots (menu) at the top right
- Select 'Restart Home Assistant'
- Click on the 'Restart Home Assistant' option in the pop-up window
- then finally confirm in the query window

... and wait : the progress is reported at the bottom left until HA is fully loaded again.

Now go to the 'Overview' page and - if everything has worked - a new 'Sensor' panel with the PiLogger values will appear:



This means that nothing stands in the way of integrating individual values into your own dashboards (overview pages).

The entities that have now been added to the system can now also be used for automation.

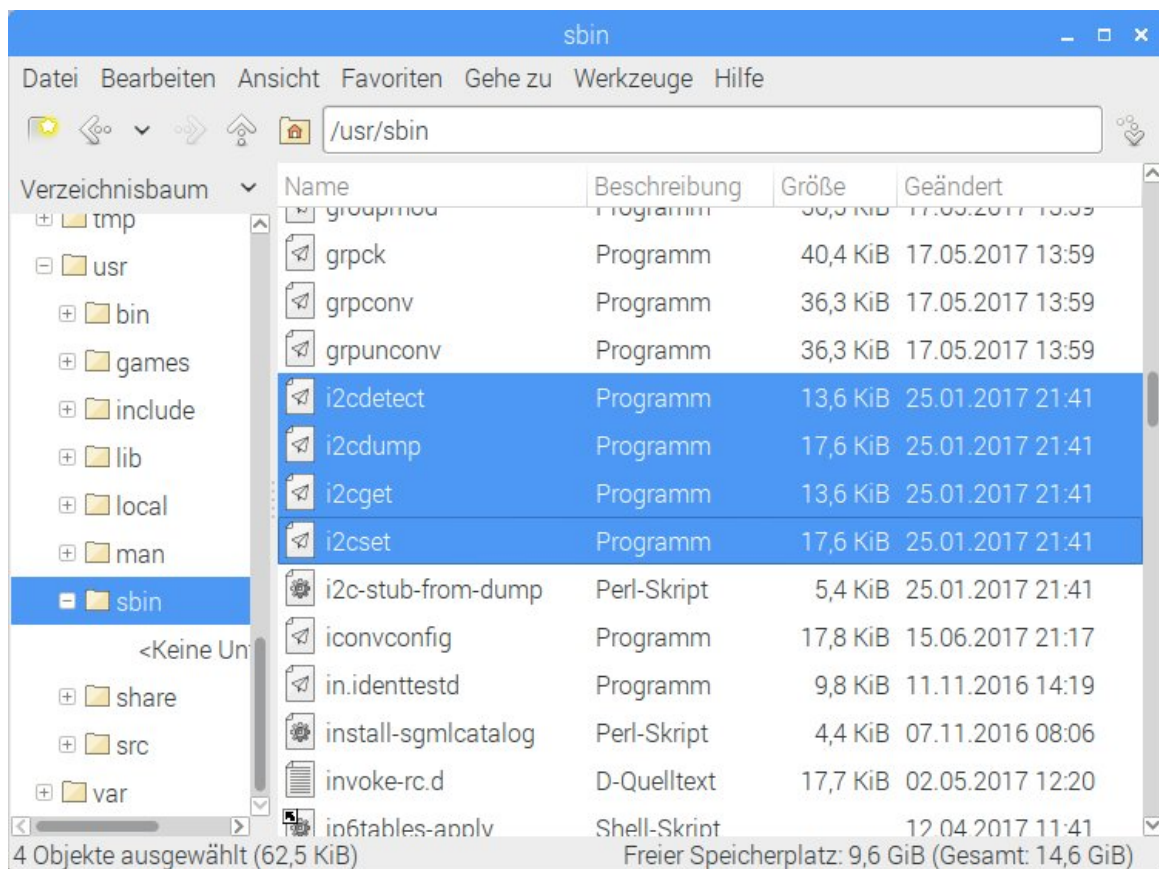
14 Appendix

14.1 The I²C Tools

This chapter applies to Raspberry Pi's on which Raspberry Pi OS is installed. This is a complete Linux operating system based on the Debian distribution.

Installing the Smbus module automatically installs the 'i2c-tools' modules, which can be called up via the command line.

The I²C tools consist of 4 programs: 'i2cdetect', 'i2cdump', 'i2cget' and 'i2cset'. They are located in the '/usr/sbin' folder:



The 'i2cdetect' tool in particular can be very useful for checking the connection and verifying the actual address of the PiLogger. The call is made in the console window with this line:

```
i2cdetect -y 1
```

The -y option causes the confirmation prompt that normally appears to be suppressed. The '1' indicates the bus to be used (for Raspberry Pi type A this would be '0'). The display then looks something like this:

```

pi@PiLogger:~$ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  48  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@PiLogger:~$

```

The official documentation can be found here:

<http://www.lm-sensors.org/wiki/man/i2cdetect>

The 'i2cdump' tool is basically a block read tool and is specially designed for reading out older, small EEPROMs - please do not use it with the PiLogger.

The tools 'i2cget' and 'i2cset' can be used to read and write individual registers of an I²C bus module. These are the basic blocks for the I²C bus operations that can be used in the console window. The respective documentation can be found here:

<http://www.lm-sensors.org/wiki/man/i2cdump>

<http://www.lm-sensors.org/wiki/man/i2cget>

<http://www.lm-sensors.org/wiki/man/i2cset>

14.2 The concept of the PiLogger WebMonitor

In this chapter we describe the operation and approach of the WebMonitor software for the Raspberry Pi - the differences for the ESP32 port are described elsewhere.

This is intended to help understanding and also as a suggestion for your own adaptations or projects - PiLogger WebMonitor is free open software (FOS).

The PiLogger WebMonitor is a network application (WebApplication) that uses the capabilities of a Raspberry Pi to provide both a network connection and access to the GPIOs (General Purpose Input/Output; Universal Input and Output Connectors).

The core of this web application is the Python web framework 'Bottle' by Marcel Hellkamp (<https://bottlepy.org/>).

"Bottle is a fast, simple and lightweight WSGI (Web Server Gateway Interface) micro web framework for Python. It is distributed as a one-file module and has no dependencies other than the Python standard library."

It is offered under MIT license and can be installed from the Raspberry Pi OS software repositories. The built-in server for development purposes is completely sufficient for a private (not Internet-connected) website. Bottle can be expanded with extension modules (plugins) - which will follow in terms of access control with 'Bottle-Cork', for example.

Instead of the built-in server, many other servers can be used as an alternative -

see Bottle documentation.

Starting from this basic building block, it makes sense to solve the rest directly in Python. The main program on the Raspberry is therefore a Python program (usually called a script - although Python is a fully-fledged high-level language). The program file is called 'PiLogger-bottle.py' and can be downloaded as part of the archive 'PiLo-WebMon.zip':

<https://www.pilogger.de/index.php/de/download-de/send/2-software/8-pilo-webmon>

The prerequisite is an activated I²C interface with an activated alternative I²C driver, as well as the installation of 'python3-smbus', the module for Python to access the I²C bus. In addition, the installation of 'python3-gpiozero', a Python extension for direct access to the GPIOs. This allows the interrupt line of the PiLogger One to be evaluated as such.

This is the basic configuration for all I²C sensors that are to be used with the Raspberry. The PiLogger One is a multi-sensor, so to speak. The application programming interface (API) is described in chapter 9.

The Python script now performs the main task - logging by reading the measured values from the PiLogger One, post-processing them and saving them on the SD card of the Raspberry Pi. Instead of using a large database, the data is saved in a plain text file in append mode. This is a classic CSV file (comma separated values) that records the measured values line by line.

The Python script now also offers the option of outputting web pages to a browser via the 'bottle' module. What these HTML files can do has been described in detail in chapter 6.

This provides a cross-platform user interface with all modern display options.

In order to minimize data traffic and, in particular, to avoid having to address external Internet servers, no third-party web fonts and no huge universal frameworks are used. In fact, a common CSS file is used for all pages to define the display style. This also includes the use of a single image file containing all the required icons as so-called sprites. The exception here is the additional file for the so-called favicon - i.e. the small icon for page recognition. Thus, the call of a page (except diagrams) is answered with 4 small files (1 document, 1 stylesheet, 1 img iconset & 1 img favicon). These resources are static and are usually cached by the browser. This means that the next page request is faster.

The actual data to be displayed is loaded dynamically from the Raspberry using XHR. This technology is called Ajax and XHR stands for 'XMLHttpRequest'. The core here is that a JavaScript script is executed in the browser, which in our case is embedded directly in the HTML page so that no further file has to be requested.

This JavaScript script executes the aforementioned requests for the data to be displayed, which are then sent as a small JSON file from the Raspberry. In the opposite direction, the data is sent to the server with the transmitted navigation address as so-called query parameters (url-data). This is also an XHR - only here the response from the server is simply ok or nok - i.e. a pure acknowledgement of success.

Another important component of the PiLogger WebMonitor software is the JavaScript library 'dygraphs' by Dan Vanderkam (<http://dygraphs.com/>).

This so-called charting library is used on the 'Diagrams' page to generate the measured value time diagrams.

For this purpose, 2 additional files are requested from the server on this page: 'dygraph_p.min.js' - the charting library in minimized form and 'dygraph_p.css' with the chart-specific style specifications.

The file 'dygraph_p.min.js' is based on the original version 2.1.0 and contains patches (repairs) for touch operation.

This JavaScript module retrieves the log data file 'showdata.csv' from the Raspberry when the 'Diagrams' page is called up, parses the data and generates a time diagram of the series of measured values as a graphic on an assigned area of the page.

This charting library can handle even very extensive measurement series in a surprisingly short time. The longest time is usually required to transfer the data file. This depends primarily on the connection quality and the size of the file. It should be borne in mind that this data file can be up to twice as large as the set data file size, as it is formed from the previous log file plus the current data file - in order to always be able to display a meaningful observation period.

Up to now, all data and communication has been purely between the Raspberry Pi as a server and a user computer logged into the local network as a client via its own router.

In order for the Raspberry Pi to add a correct time stamp to the measurement data when logging, it must synchronize its internal time with a time server. This is done by default with an Internet time server via NTP (**N**etwork **T**ime **P**rotocol). This Internet access can also be avoided with a simple measure:

The router itself synchronizes its internal time with an Internet time server, usually a time server of the network provider. Most routers offer the option of making this time available as a time server for the local internal network - see chapter 4.3.1 . If this requirement is met, the Raspberry can now synchronize locally with the router.

Now only the regular (daily) requests to the Raspberry Pi OS update servers remain. If the Raspberry is intended exclusively as a logger and strictly behind the router's firewall, an update is generally not security-relevant. Especially as these

update requests only update the database anyway without a configured auto-update - and otherwise an unattended update can interrupt logger operation. These update requests can optionally be switched off - see chapter 4.3.2 .

Currently, access to all functions of the PiLogger WebMonitor is open to all users of the home network. This can be changed with an extension module for 'Bottle' such as 'Bottle-Cork' - todo...

If you want to use the PiLogger WebMonitor remotely via the Internet, you should do this via a VPN (virtual private network) - i.e. via a secure point-to-point connection.

PiLogger WebMonitor thus also shows that 'Smart Home' and 'Internet of Things' are also possible without the cloud and third-party servers.