The PiLogger One Manual

Table of Contents

1 Introduction

The PiLogger One is an extension card for the Raspberry Pi. Adding the PiLogger One implements an autonomous working 4-channel measuring system connected via I²C bus to the Raspberry Pi.
The main focus is on measuring electric power (DC power meter) by measuring voltage and current, accompanied by a pulse counter, which enables measurements of turn rates or flow rates, like wind speed or hot water flow rate. Additionally a sensor input for temperature measurements is provided.
More details are provided in the chapter 'Technical Data'.
This combination is especially suitable for monitoring small generators of green power like wind mills or solar panels and their associated batteries.
The special point is the concept of an autonomous working peripheral unit. The PiLogger comes with its own timebase, which allows precise timed intervals between measurements independent of the actual activity on the Pi. This also enables low power consumption as the Pi could idle in low power mode especially when doing long period logging.
The PiLogger takes over the tasks of performing the periodically measurements and calculating the power and the moving average values. The Raspberry Pi just has to retrieve the results at the desired logging intervals, could do additional post processing, if needed, and takes care of storing the results. The respective application program interface is described in an own chapter.
Enabled by the versatility of the Rasberry Pi – especially the capability of wireless networking – a bunch of possibilities arises, for instance a small web server for graphical presentation of the measurement results.
The PiLogger base software is meant for just giving a starting point for this opportunities. It serves for the purposes of first operation, configuration and testing. It is free demonstration software and therefore it enables learning and is suitable as a starting point for own projects.
This manual is intended to give a step by step tutorial also for beginners.
Advanced users of the Rasberry Pi could omit the first chapters. The later chapters provide the technical reference and documentation for a successful integration of the PiLogger to your own projects.
In the meantime there are a lot of variants of the Rasberry Pi, but thanks to the reduced form factor of the PiLogger One, which is not a 'hat board', it is also compatible to the first generation of Rasberry Pi equipped with a 26 pin header.
A good chance for an old Pi 1 to be reactivated !

We wish you a lot of fun and success with the PiLogger !

2 Safety Instructions

The following safety instructions and hazard warnings are meant to protect your health and the integrity of the device. Please read them complete and carefully:

- Before touching the printed circuit board (PCB) and before installing and wiring, appropriate measures against electrostatic discharge must be taken (e.g. grounding guard, non-conductive mounting plate etc.).
- Mounting and wiring must be done in a not powered condition. All power supplies must be switched off.
- The product is suitable for use in dry and closed rooms only.
  It must not get exposed to condensing humidity or even water, this would lead to damage.
- Protect the product against cold temperatures, heat, direct sun exposure, dust and dirt.
- Handle the product with care. Impacts and mechanical stress, also falling from even small height might lead to damages.
- This product is not a toy. Keep it away from small children. Keep it in safe places where small children can not reach it.
- Don't let the device, as well as the packaging material, lay around unattended, it could become a dangerous toy for small children.

⚠	*Take care on good isolation and protection against touching contacts when installing the measurement wiring. Risk of electric shock !*

⚠	*Take care on strain relief when installing the measurement wiring and a stabile mechanical setup in general to avoid short cuts !*

3 Preparing the Raspberry Pi

3.1 Choice of the Operating System

The free of charge provided demonstration software for the PiLogger is written in 'Python 3' utilizing the graphical extension 'tkinter'.

As the Rasberry Pi originally was intended as an easy to use platform for Python, it suggests itself to choose the officially recommended operating system 'Raspbian'. It is based on Debian Linux with really excellent adaptions for the hardware and it gains from broad support. And additionally it is almost open source and free of charge, except for some special components (Broadcom) and Java (Oracle) !

Those users, which are already familiar in setting up Raspbian on a Rasberry Pi can skip this chapter and proceed to chapter 4 'Setting Up the Base Software'.

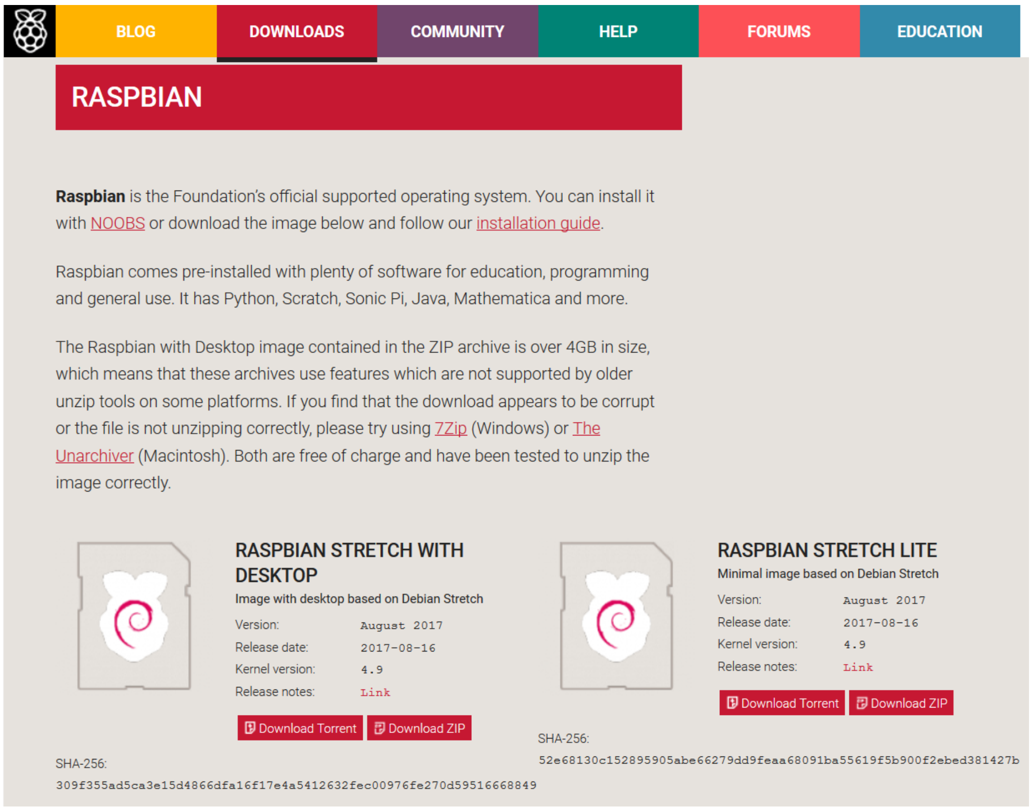
3.2 Download Raspbian

3.2.1 Raspbian or Raspbian Lite

On the download page at www.raspberrypi.org following the link to Raspbian, there are two versions listed : 'Full Desktop Image' and 'Lite: Minimal Image'.

An 'Image' means a big file containing the complete content of a data carrier as a 1:1 copy.

For the following we want to use the graphical interface, therefore the choice here clearly is the full desktop version !



RASPBIAN

Raspbian is the Foundation's official supported operating system. You can install it with NOOBS or download the image below and follow our installation guide.

Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more.

The Raspbian with Desktop image contained in the ZIP archive is over 4GB in size, which means that these archives use features which are not supported by older unzip tools on some platforms. If you find that the download appears to be corrupt or the file is not unzipping correctly, please try using 7Zip (Windows) or The Unarchiver (Macintosh). Both are free of charge and have been tested to unzip the image correctly.

RASPBIAN STRETCH WITH DESKTOP

Image with desktop based on Debian Stretch

Version:	August 2017
Release date:	2017-08-16
Kernel version:	4.9
Release notes:	Link

SHA-256:
309f355ad5ca3e15d4866dfa16f17e4a5412632fec00976fe270d59516668849

RASPBIAN STRETCH LITE

Minimal image based on Debian Stretch

Version:	August 2017
Release date:	2017-08-16
Kernel version:	4.9
Release notes:	Link

SHA-256:
52e68130c152895905abe66279dd9feaa68091ba55619f5b900f2ebed381427b

Of course you could use the PiLogger also in a so called 'headless' system, which means without a graphical user interface. But this is not the subject of this manual.

In case the download is done using a Windows PC the SHA-256 checksum, which is given on the download page, optionally could be calculated for instance also by the tool 'Win32 Disk Imager' (https://sourceforge.net/projects/win32diskimager/).
This tool we will use later to write the image to the SD card.
After starting the program we first have to specify the image file at the respective input filed. This can be done by browsing too by clicking on the small folder button right to the input field.
Because the given checksum is meant for the .zip archive file, we have to set the file type filter to *.* to find the downloaded file and select it.
After that we go to the area ‘Hash' to the drop-down choice box showing 'None' and select SHA256. Now the calculation of the checksum is started by clicking on the button 'Generate' right next to it (this will take some time for 1,6 GB):

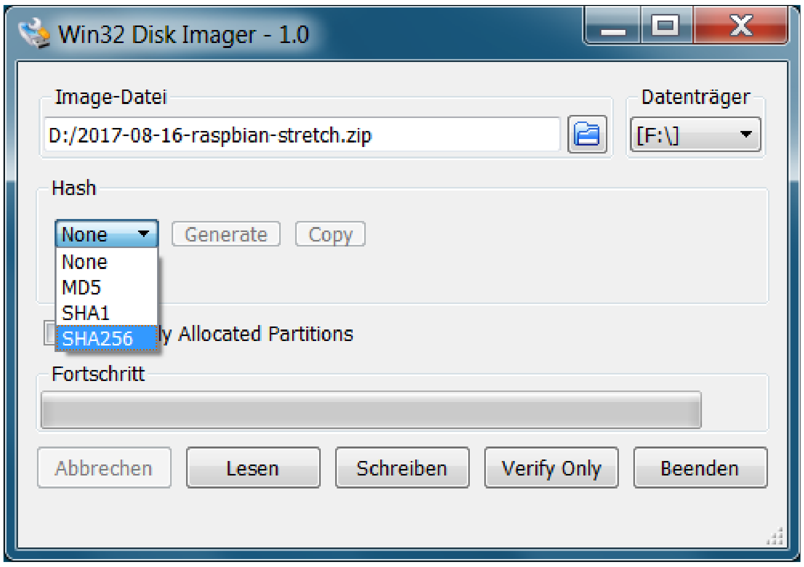
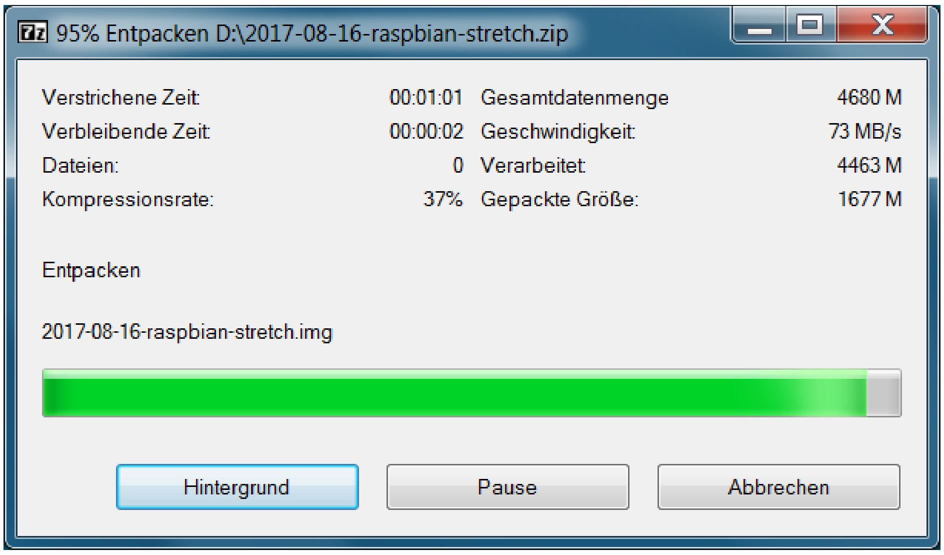


Image-Datei: D:/2017-08-16-raspbian-stretch.zip
Datenträger: [F:\]
Hash: None

By comparing the checksum values you are able to check whether the file is corrupted or manipulated – which might be especially reasonable if you didn't download the file yourself, but instead got it from a friend on an USB stick, for instance.

3.2.2 Unpacking the archive file

The downloaded file is an .zip archive file, which contains the real image file in a compressed format. In this state the file can’t be written to the SD card. It has to be decompressed before – it has to be unpacked.
With the most recent Raspbian versions you may encounter problems using the onboard unarchiver of the windows explorer. As stated on the raspberrypi.org download page, you should use the application 7zip to unpack. It can be downloaded at:
https://sourceforge.net/projects/sevenzip/
This application behaves like a usual file manager. After selecting the image file within the archive you can start the unpacking either by using the context menu (right click) or by drag and drop:


95% Entpacken D:\2017-08-16-raspbian-stretch.zip

Verstrichene Zeit:	00:01:01	Gesamtdatenmenge	4680 M
Verbleibende Zeit:	00:00:02	Geschwindigkeit:	73 MB/s
Dateien:	0	Verarbeitet:	4463 M
Kompressionsrate:	37%	Gepackte Größe:	1677 M

Entpacken

2017-08-16-raspbian-stretch.img

Take care to remember the target folder for the image file !

3.3 Preparing the (micro) SD Card

3.3.1 Which SD Card ?

For the SD card or micro SD card (only generation 1 of the Raspberry Pi has a big card holder for a classic SD card) it is recommended to use a quite fast version, because the Raspberry Pi does use it like a hard disk. The memory size should be in the range of 8 GByte to 32 GByte. This implies that the card is a SDHC type of card (high capacity). An 16 GByte card with a speed grade of 40 MByte/sec or faster, today is probably the smallest available size in this class and therefore will be the best trade-off in terms of cost. With default settings the Rasberry Pi will achieve a speed rate of around 21 MByte/sec at the SD card interface. When using a card with UHS-1 / class 10 speed specification this can be boosted up to around 42 MByte/sec by using a special overclock setting (see appendix) – but always limited by the real performance of the used SD card.

With the Rasberry Pi 3 using this overclock setting will prevent the detection of the built in bluetooth/wifi module.

The extensively advertised speed rates almost are given for a continuous reading mode (streaming), which gives the more impressive figures. But for the general behaviour in a normal desktop usage the card better should be optimised for reading and writing of small blocks of 4 Kbyte (sometimes also advertised) – for this mode values in the range of 3 Mbyte/sec already are excellent.

3.3.2 Writing the Image to a SD Card

In the case a Windows PC is available, the application 'Win32DiskImager' can be recommended, as it makes writing an image to the SD card very easy and comfortable. After starting the application the above window appears. Now you have to give the location of the image file to the input field named 'Image File'. Alternatively you can choose the file by browsing, using the button with the folder icon right next to the input field.

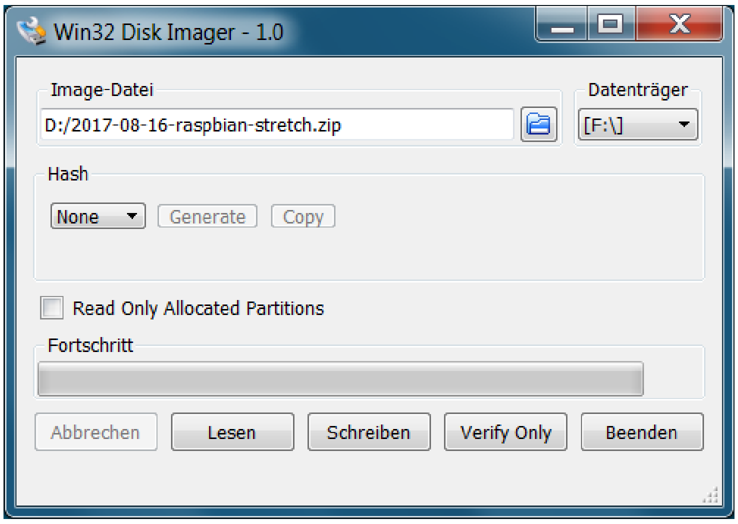

Win32 Disk Imager - 1.0

Image-Datei: D:/2017-08-16-raspbian-stretch.zip

Datenträger: [F:\]

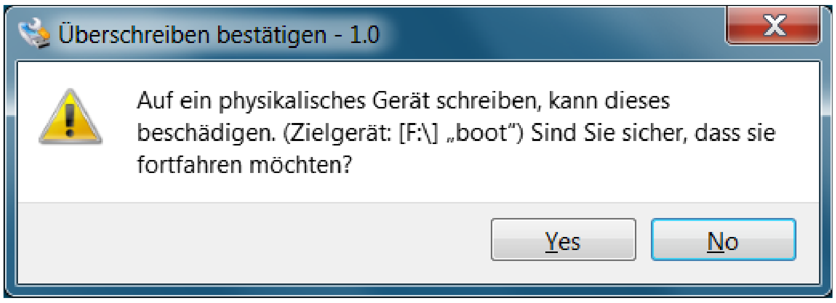
Hash: None

Read Only Allocated Partitions: ☐

More to the right side, in the area 'Device', you will find a fold out list for the drive letter to be used as card reader – *attention : don't mix it up with a different drive for removable memory !*

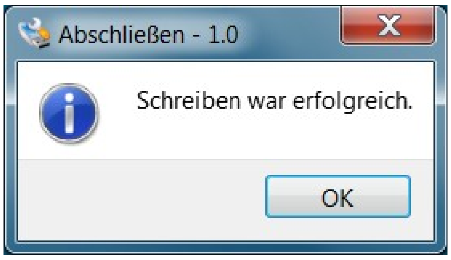
The image copy process now is to start by a click on the button 'Write'.

Subsequently a confirmation window appears, wherein the text is warning about corruption of the device – this sounds much more dramatic then probably meant. The window title gives the hint that it is mainly about overwriting existing data finally on the target device. Also in our case there will be a file system written to the card, which Windows is not able to read – don't panic, by this the card is not really destroyed.


Überschreiben bestätigen - 1.0

Auf ein physikalisches Gerät schreiben, kann dieses beschädigen. (Zielgerät: [F:\] „boot") Sind Sie sicher, dass sie fortfahren möchten?

By clicking on 'Yes' the copy process will really start now. It takes some time. When it's finally done, hopefully this message window will appear:


Abschließen - 1.0

Schreiben war erfolgreich.

Click on 'OK' now. The message window disappears and the main window will be visible again. Here you finish the application by clicking on 'Exit'.

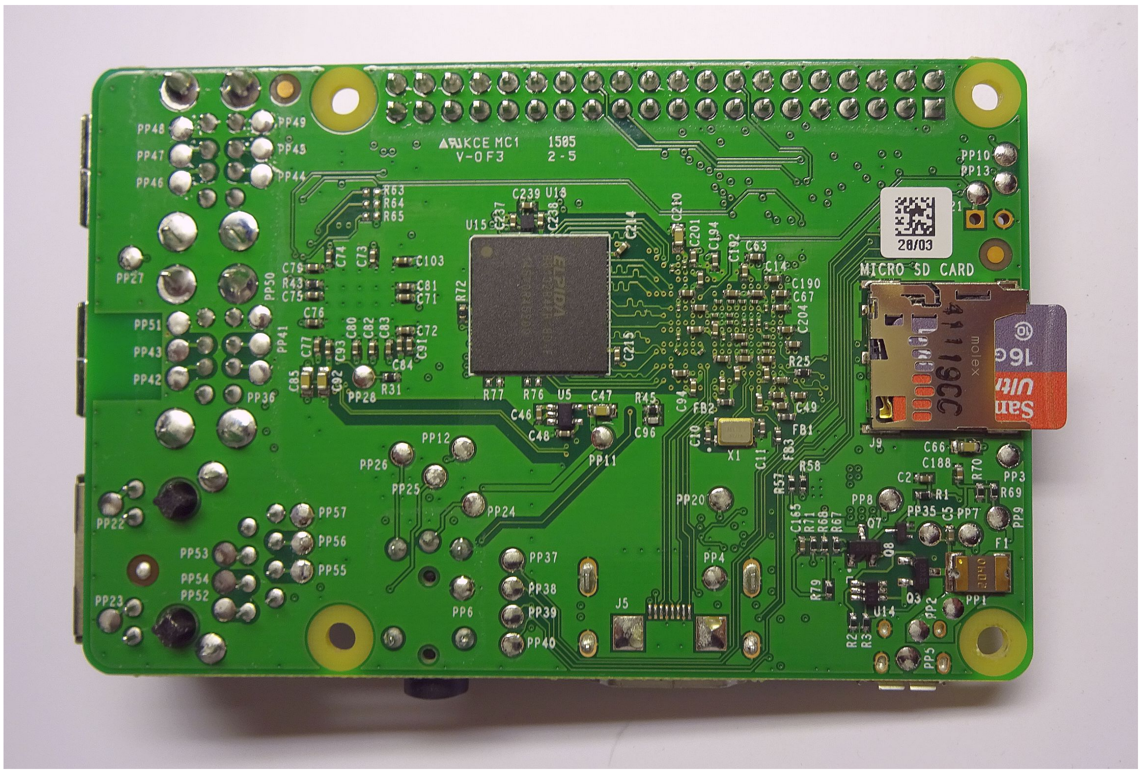
Now don't forget to remove the card reader safely before taking out the SD card.

Removable flash memory has the awkward habit to operate on it's own internally without showing to the user. When interrupted at an internal write operation by a sudden power loss – just a normal removal – even nowadays it may happen that the flash memory becomes broken.

3.4 Setting Up the Raspberry Pi

3.4.1 Inserting the SD Card at the Raspberry Pi

The freshly prepared SD card can now be plugged into the appropriate slot at the Pi – but the Pi must be unpowered to do so!
It is recommended to do this when having nothing connected at all, because this makes the handling more easy and safe.



3.4.2 Connecting Power Supply, Display, Keyboard and Others

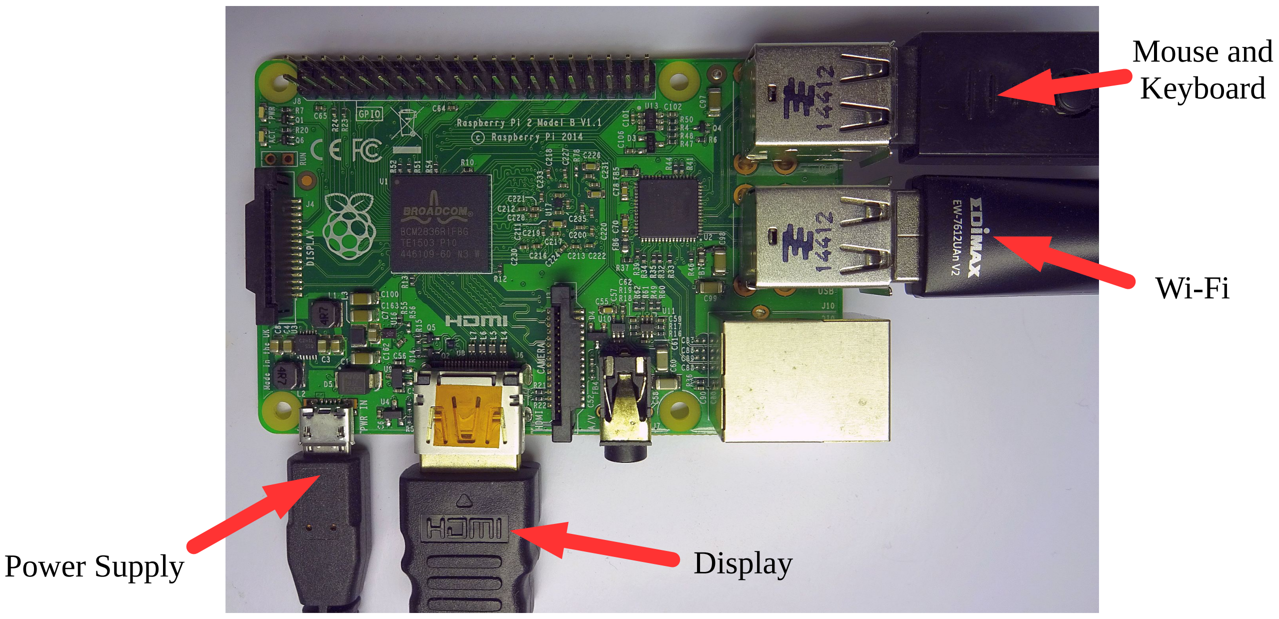
Now the Pi needs a basic environment for it's first boot up:

- **Power supply : without electricity nothing works**
  The Raspberry Pi needs an USB power supply with micro USB pigtail.
  All USB power supply units will provide nominal 5 V output voltage, but their load current capability differs a lot. The weakest models, which are probably the cheapest, have to be able to deliver at least 500 mA, because of the USB specification.
  Nevertheless, this will not be suitable for most of the Pi models and their respectively connected USB peripherals. 1000 mA or 1 A should be the minimum load current capability, for the new Raspberry Pi 3 even 2,5 A are required.
  Under the following described preparation procedure please do not switch on the power supply – e.g. don't plug in the wall plug.
- **Output : humans need a kind of display**
  The simplest and most convenient way to connect a display to the Raspberry Pi is to use the HDMI connector. In case there is no modern type of PC display with HDMI

input available and the TV set in the living room is momentary used in a different way, an old discarded display with a DVI input will also work perfect. For that purpose there are affordable adapter cables available, connecting the HDMI output to the DVI input. The resolution and the aspect ratio are not that important, because the supported range is big – so it works (in most cases).
In principle the Raspberry Pi can also be operated remote controlled via a network connection – but because this is not that easy and intuitive we won't go into details here.
If the display is equipped with loudspeakers and has a HDMI input, the Pi also can output audio via the HDMI connection. Alternatively you can connect an analogue audio device to the 3,5 mm audio jack. This may be a headphone or an active stereo speaker set.

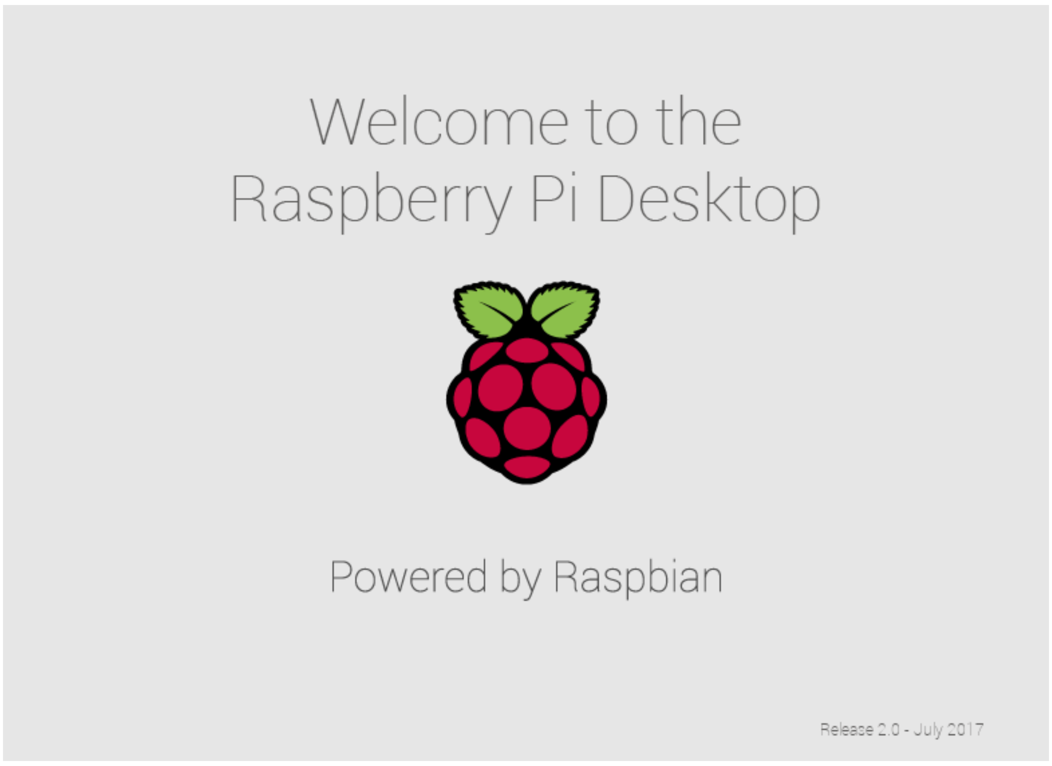
- **Input : the Pi needs Input**
  The easiest solution for an input device is a simple USB keyboard. As using a pointing device like a mouse also is recommended, this additional device will also use an USB port. Therefore a wireless mouse and keyboard combination would save one port and would draw less current than using 2 ports while allowing comfortable working.
- **Network : connections are useful**
  Via a direct internet connection you can do the necessary first updates fast and convenient. But there is more benefit: Raspbian provides all needed tools to surf and search out of the box. Basically, on older versions, there was an application called 'Epiphany' installed, which is a light weight web browser for the Gnome desktop – with the newest versions Chromium will be installed, which is the open source variant of the Google Chrome browser.
  Depending on the given infrastructure you could use a direct cable connection to your router (*not with model A+*) or use a USB Wi-Fi stick. Considering the expected use case with the PiLogger, a wireless connection might be preferable.
  The latest addition to the Raspberry Pi family, the Rasberry Pi 3 Model B, already brings an onboard Wi-Fi module.

The completely connected Rasberry Pi, for example the model 2 B, should look like this:



3.4.3 First Boot, Configuration

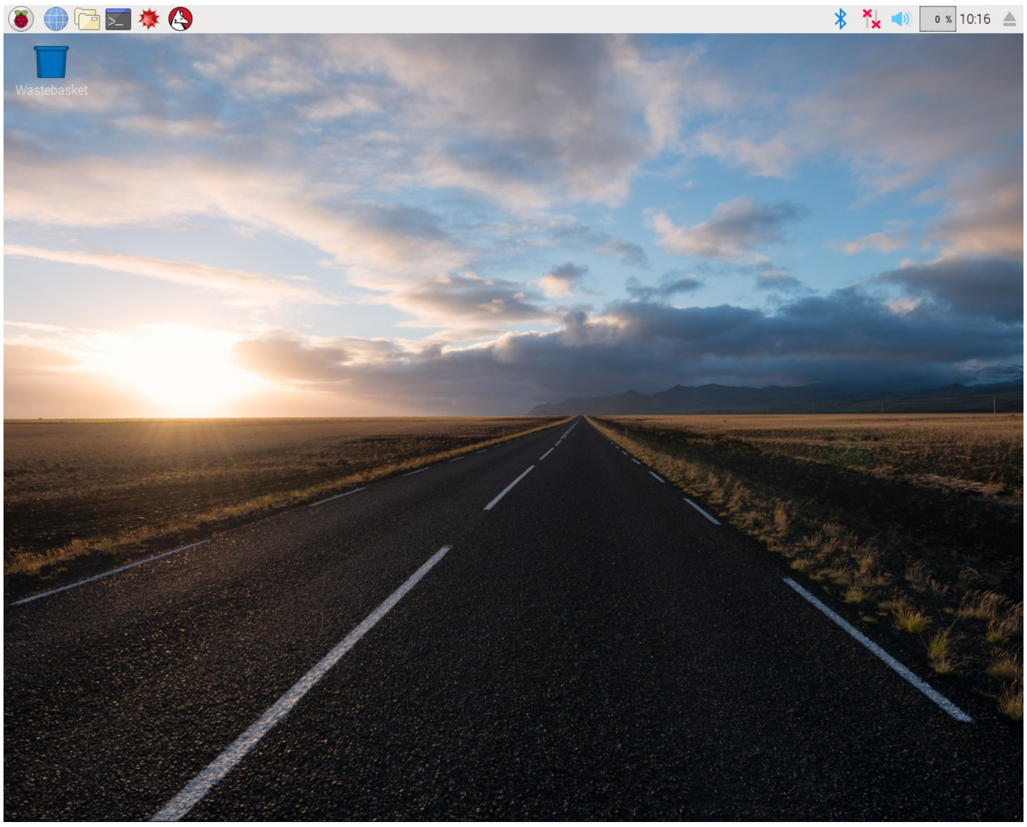
Now it is time to switch on the power and to watch the Raspberry Pi at his first boot.
On the display you now should see some raspberries – each one standing for a found processor core. With a Raspberry Pi 3 there should be 4 of them.
Then for a short time a message will state that the file system got expanded (of the SD card) and a reboot will follow – all fine so far.
With the next boot up the following welcome screen will appear the first time:


Welcome to the
Raspberry Pi Desktop

Powered by Raspbian

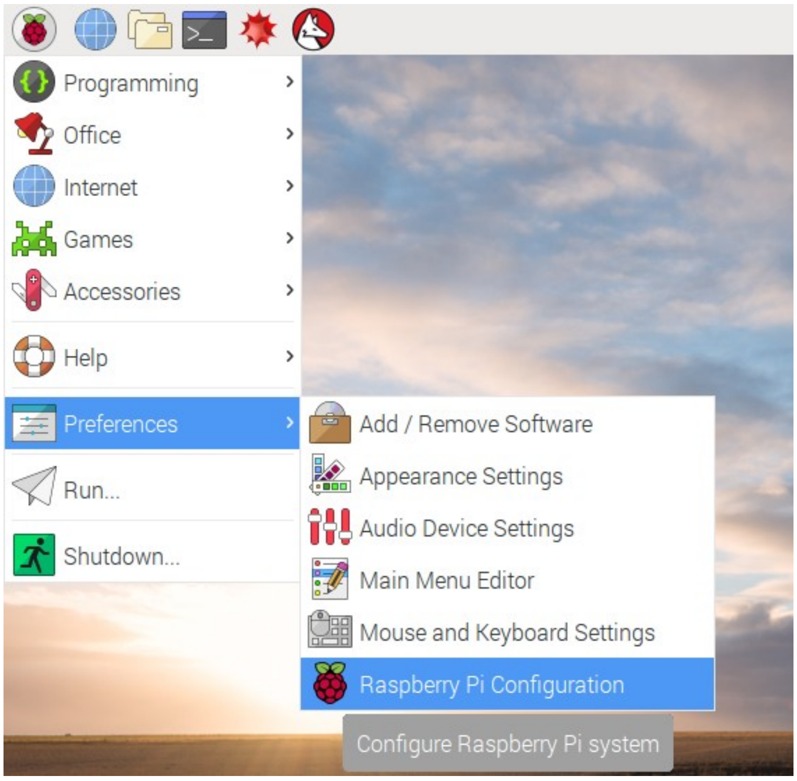
Release 2.0 - July 2017

Now it takes a short time until actually the desktop screen will appear:

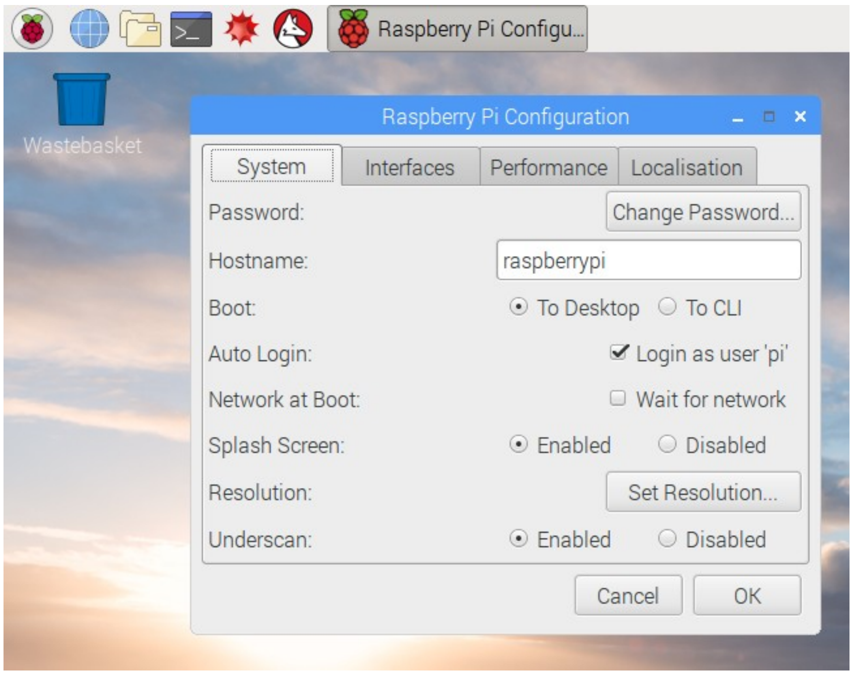


The taskbar is on top of the screen by default and the system language is set to English.

Now some basic things have to be set up, for instance the system language – if it doesn't fit. For that purpose a tool is provided, called 'raspi-config'.
Since Raspbian Jessie a graphical version of this tool is provided, which can be started like every other application in the start menu.
So let's go to 'Menu' > 'Preferences' > 'Rasberry Pi Configuration' :



With this tool all necessary system settings are done quite comfortable. The application starts with this screen:



Password: Change Password...
Hostname: raspberrypi
Boot: [✓ To Desktop] To CLI
Auto Login: [✓ Login as user 'pi']
Network at Boot: Wait for network
Splash Screen: [✓ Enabled] Disabled
Resolution: Set Resolution...
Underscan: [✓ Enabled] Disabled

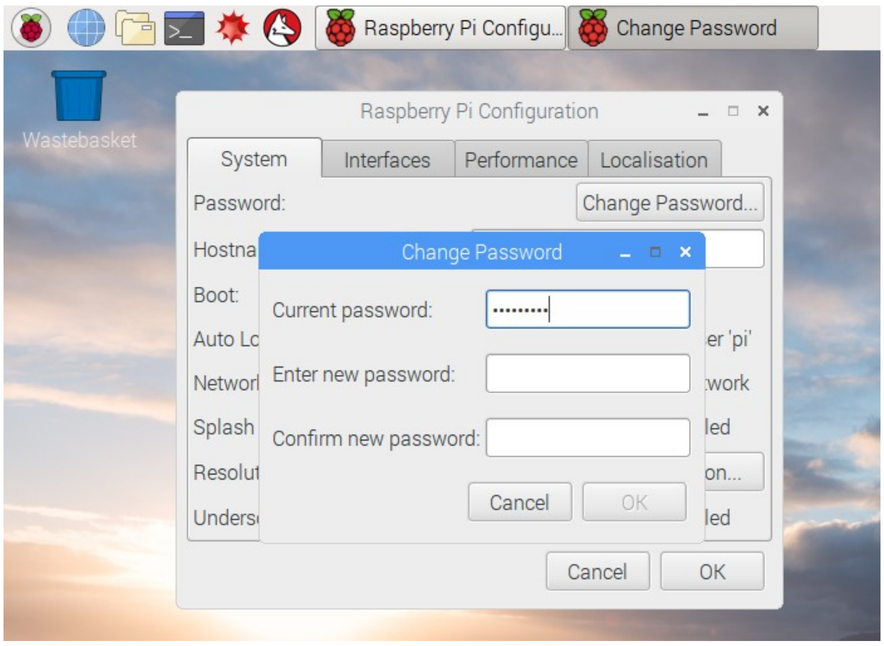
In the following all reasonable settings for the use of the PiLogger are shown step by step.

3.4.3.1 System

On the first tab 'System' you will find these options:

- **Change Password**

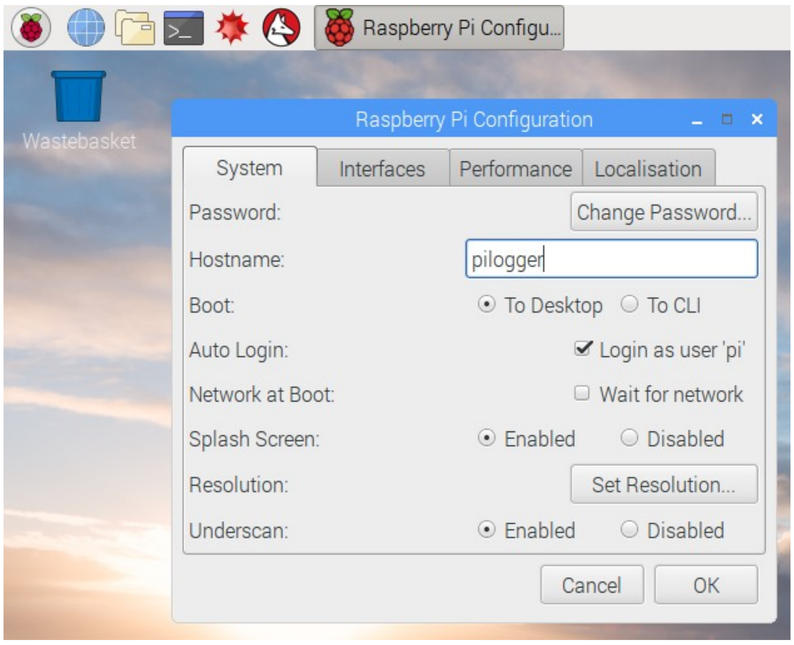
  Here you may change the password used for the system logon for the default user 'pi', which had been set up by default. Also by default the password was set to 'raspberry'. Because all the Pi's will have this password as a start value, this is like an open door. It is therefore absolutely necessary to define a new and safe password here (*memorize it and/or write it down !*) :



Current password: ·········
Enter new password:
Confirm new password:

This is just a first small step to secure your Rasberry Pi, especially if it’s planned to operate it with a connection to the internet. Further measures for security you will find at : https://www.raspberrypi.org/documentation/configuration/security.md

- **Hostname**
  Hostname is the name of this Rasberry Pi shown at a connected network. Especially if more than one Pi are active on the same network, you would like to identify them by meaningful unique names. Of course we recommend 'pilogger' here:



Password: Change Password...
Hostname: pilogger
Boot: [✓ To Desktop] To CLI
Auto Login: [✓ Login as user 'pi']
Network at Boot: Wait for network
Splash Screen: [✓ Enabled] Disabled
Resolution: Set Resolution...
Underscan: [✓ Enabled] Disabled

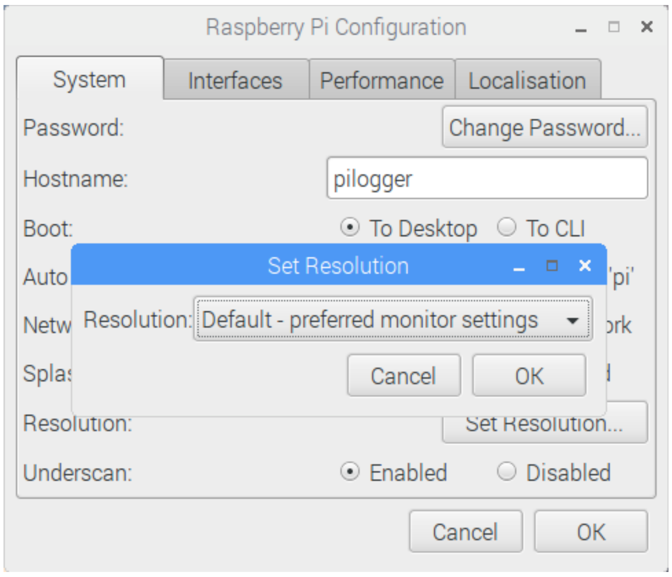
- **Boot 'To Desktop' or 'To CLI'**
  Here you can choose whether the system shall come up with the graphical user interface or shall stay at the command line interface at boot time. Because we do need the graphical user interface with the PiLogger base software, the default setting 'To Desktop' shall stay activated (unchanged).

- **Auto Login**
  This option is activated by default, leading to an automatic and quiet login of the user 'pi' at start up. By this the startup of the Rasberry Pi becomes very quick and easy, but it allows everyone to do this without requiring a password. For a system meant for development and testing this should be no problem – if you want to make your system more safe, you should change this option.

- **Network at Boot**
  Here you can choose whether the system shall wait on a successful network connection at boot time or not. For a system which is actually demanding a working network connection, e.g. a web server, this might be reasonable. But in general use cases it is appropriate to leave this setting deactivated. By that the system start will not be delayed additionally.

- **Splash Screen**
  This option is activated by default, thus showing a graphical welcome screen at boot up instead of the common text messages of a Linux system.
  A matter of personal flavour.

- **Resolution**
  Here you can chose from a small list of special display formats, especially for LCDs with lower resolutions. This is useful for instance when using the Raspberry Pi monitor.



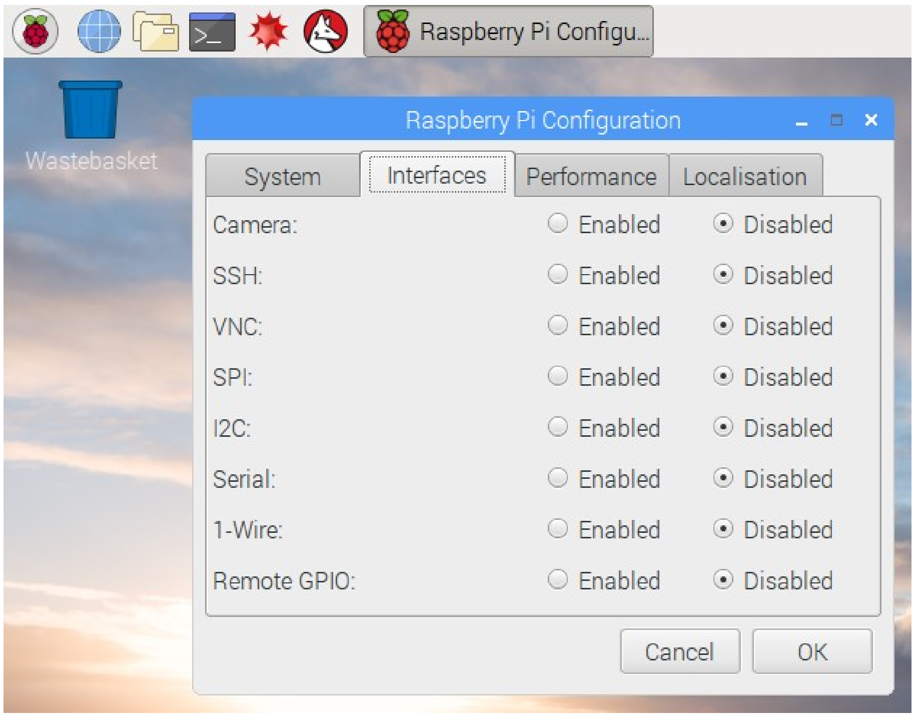
Password:
Hostname: pilogger
Boot: [✓ To Desktop] To CLI
Resolution: Default - preferred monitor settings
Underscan: [✓ Enabled] Disabled

If your display worked fine up to here, there is no reason to change it.

- **Underscan**
  This setting is related to the display output handling. Some displays – especially TV sets – do not show a few pixel at the edges of the frame, cropping a part of the content. Activating the option 'Underscan' can prevent this, but may lead to a black frame around the content on some displays. Just test it with your display !

3.4.3.2 Interfaces

On the next tab 'Interfaces' you will find the following options:



Camera:	Enabled	[✓ Disabled]
SSH:	Enabled	[✓ Disabled]
VNC:	Enabled	[✓ Disabled]
SPI:	Enabled	[✓ Disabled]
I2C:	Enabled	[✓ Disabled]
Serial:	Enabled	[✓ Disabled]
1-Wire:	Enabled	[✓ Disabled]
Remote GPIO:	Enabled	[✓ Disabled]

- **Camera**
  If you want to use the special interface for a camera (CSI), this option should be enabled. If you don't want to connect a camera, leaving this setting to 'disabled' will save some system resources (boot time, memory).
- **SSH**
  The abbreviation SSH stands for '**S**ecure **Sh**ell'. This is an interface allowing the remote control of the Rasberry Pi via another computer. For this purpose a terminal application is used on the remote computer to provide a window showing the command line interface (CLI) of the Pi. The term 'secure' means, the connection is secured by an authentication and by encryption. SSH can be used via a direct serial connection (UART; accessible at expansion header) or via a network connection.
  In case you connect your Pi to the public internet, you should provide secure passwords for all users before activating SSH.
  Recommendation: keep it disabled until you really need it.
- **VNC**
  VNC (Virtual Network Computing) offers the possibility to remote control the Raspberry Pi via a network connection. For this a server application running on the Pi transfers the current graphical screen content to the remote computer and in turn receives the input from the remote keyboard and mouse or touch panel. Therefore on the remote (controlling) computer the suitable client software is needed.
  Raspian comes with the application RealVNC, which can be used for free for private, non-commercial purpose. To achieve a secured and encrypted connection via the internet a

registration and setting up an account is required. The connection than will be accomplished via a cloud server.
More detailed information is available under :
https://www.realvnc.com/en/connect/docs/raspberry-pi.html
Attention: Activating this option will lead to automatically starting the VNC server at every boot up !

- **SPI**
  This option would activate the serial peripheral interface (SPI), which is available at the GPIO port of the Rasberry Pi. The PiLogger One does not use this interface, so with this focus it could be left disabled.
- **I2C**
  This interface is essential when using the PiLogger One.
  So activating this option is a must!
  More details on this kind of interface you will find in chapter 4.1 and following.
- **Serial**
  In a default install of Raspbian, the primary UART (serial0) is assigned to the classical Linux console. The Rasberry Pi provides 2 interfaces of this kind, which are assigned differently depending on the model.
  This is the serial interface above mentioned at 'SSH'. It provides a direct connection to the command line interface of the Rasberry Pi. Physically it is accessible also at the GPIO extension connector.
  At the various Raspbian versions this interface is sometimes activated and sometimes not. In some cases the Linux-Console is the last rescue option for a troubled system, therefore here the recommendation is: Set it to activated.

3.4.3.3 Performance

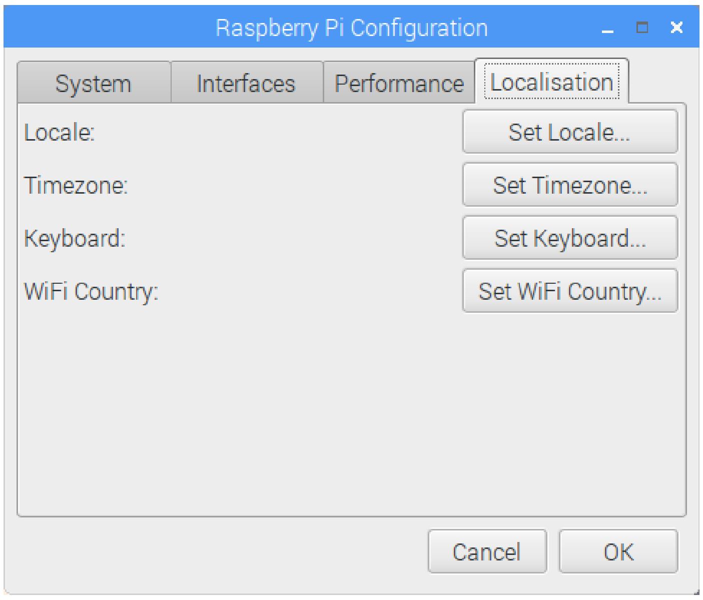
The next section is named 'Performance'. Here you could change two basic settings related to the system performance of your Rasberry Pi : the clock rate of the processor and the memory size for the graphics processing unit (GPU). The possible settings differ corresponding to the used model of Rasberry Pi.
Basically there are only a few cases for which it does make sense to change these settings, for instance if the Pi shall serve as a media center. For the use case 'data logger' the main focus will be on saving energy, so an increase of the system power is not really desired.
The following just gives an illustration on the setting possibilities:

- **Overclock**
  None (700 MHz) ; Modest (800 MHz) ; Medium (900 MHz) ; High (950 MHz) ; Turbo (1000 MHz)

- **GPU Memory**
  16 ; 32 ; 48 ; **64** ; 80 ; 96 ; 112 ; 128...880 [MByte]

3.4.3.4 Localisation

This section is named 'Localisation'. This term is used to describe the software support for different countries and languages. So it's about system settings for country, language, time zone and similar.

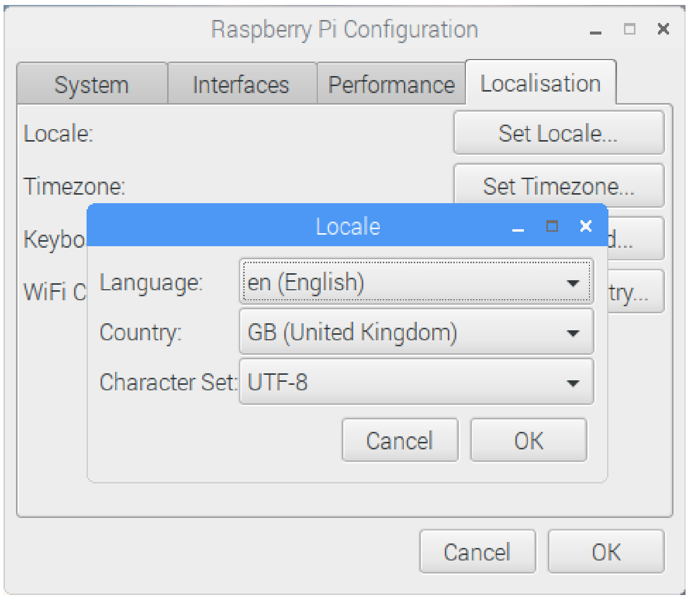


Locale: Set Locale...
Timezone: Set Timezone...
Keyboard: Set Keyboard...
WiFi Country: Set WiFi Country...

By default the settings start value is 'English UK' – except the timezone, which is set to UTC. So, who wants to stay with that, may proceed to the point 'Timezone'. All others have to adapt every option once.

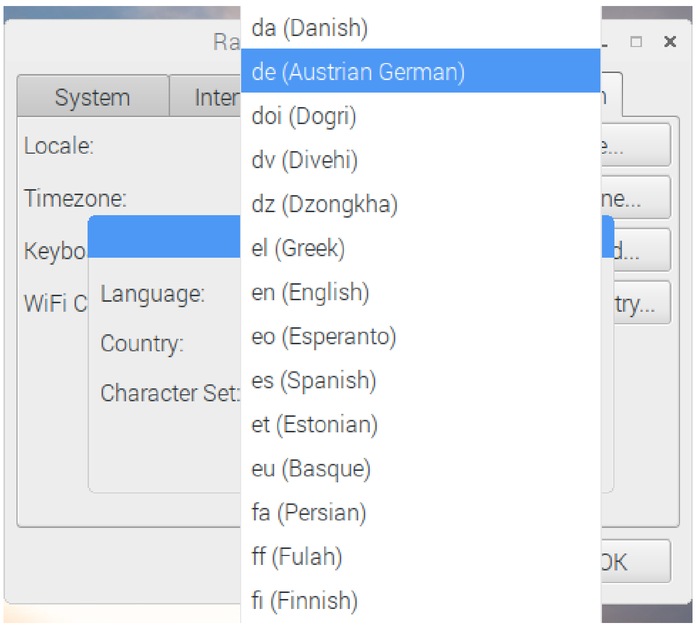
List of menu options:

- **Locale**
  This is the main option for choosing language, country and lettering. Clicking on 'Set Locale' will open this sub menu:

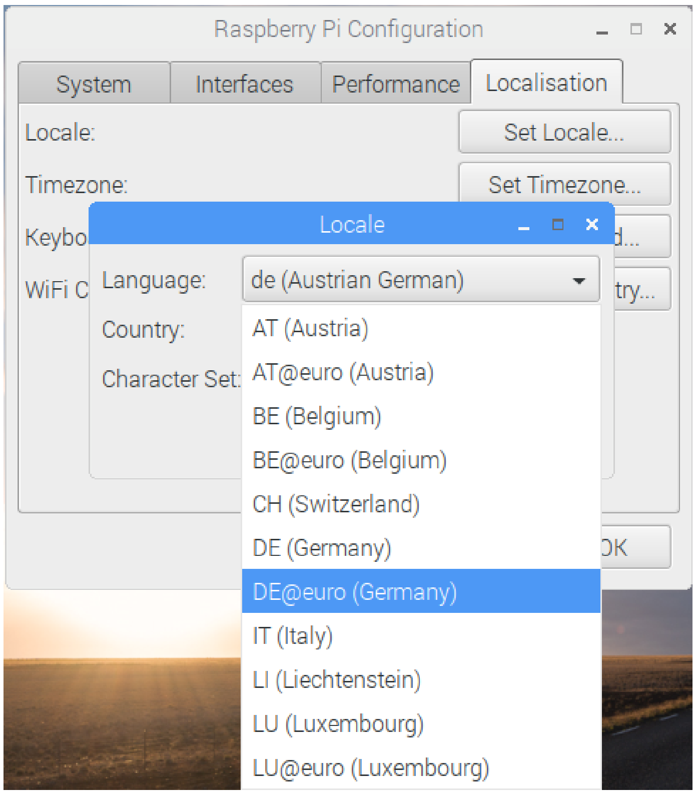


Language: en (English)
Country: GB (United Kingdom)
Character Set: UTF-8

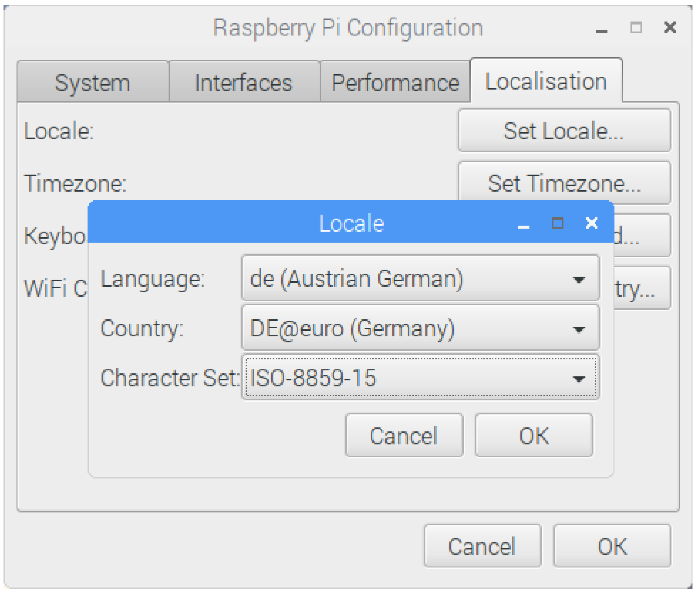
Clicking on the choice box 'Language' will give this screen:



You have to choose the wanted language by the international 2 letter abbreviation, in the given example it's 'de (Austrian German)'. By that choice the next choice box 'Country' will be reduced to the remaining options automatically:



To get the Euro-Sign (€) integrated as usual it is recommended to choose the variant 'DE@euro (Germany)'. Again by that choice of the next choice box 'Character Set' automatically is reduced to the only possible option 'ISO-8859-15' in this example:

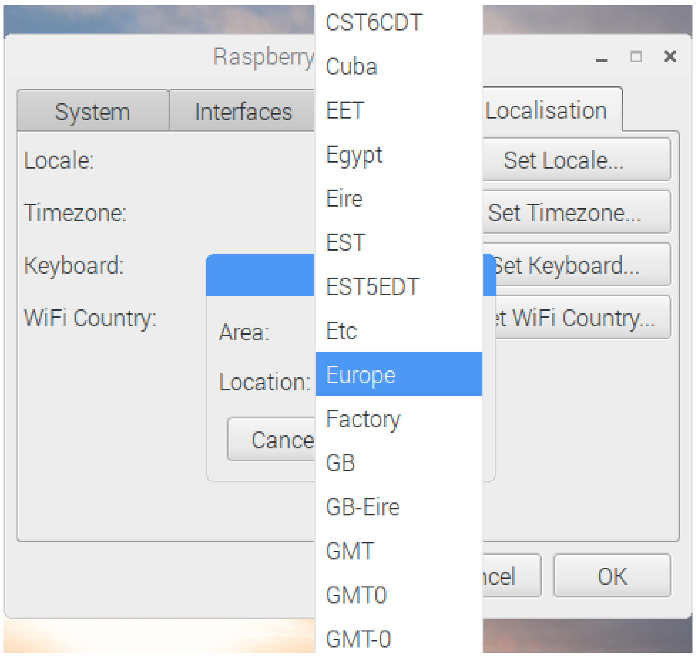


Language: de (Austrian German)
Country: DE@euro (Germany)
Character Set: ISO-8859-15

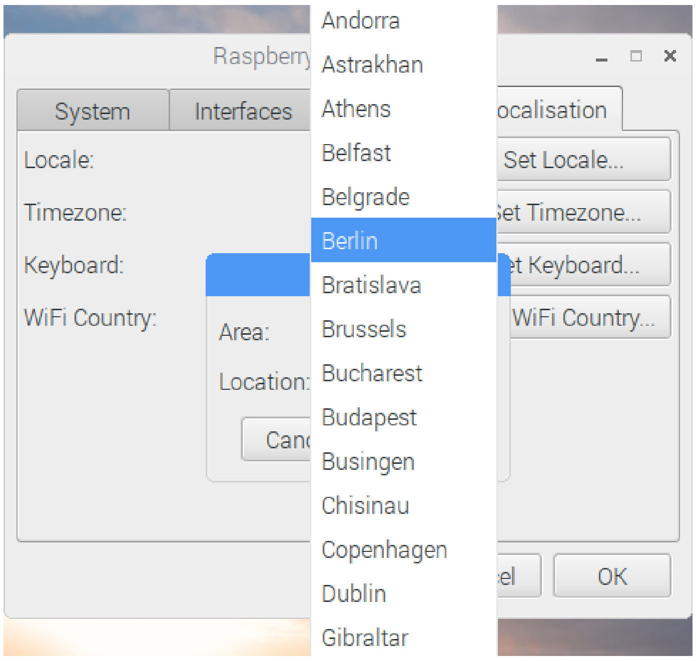
By clicking on 'OK' this sub menu is completed.

- **Timezone**

Here you will set the timezone according to the one the Raspberry Pi will be operated. This will allow the operating system to deal with the date and time formats correctly. A click on 'Set Timezone' will show up this sub menu:



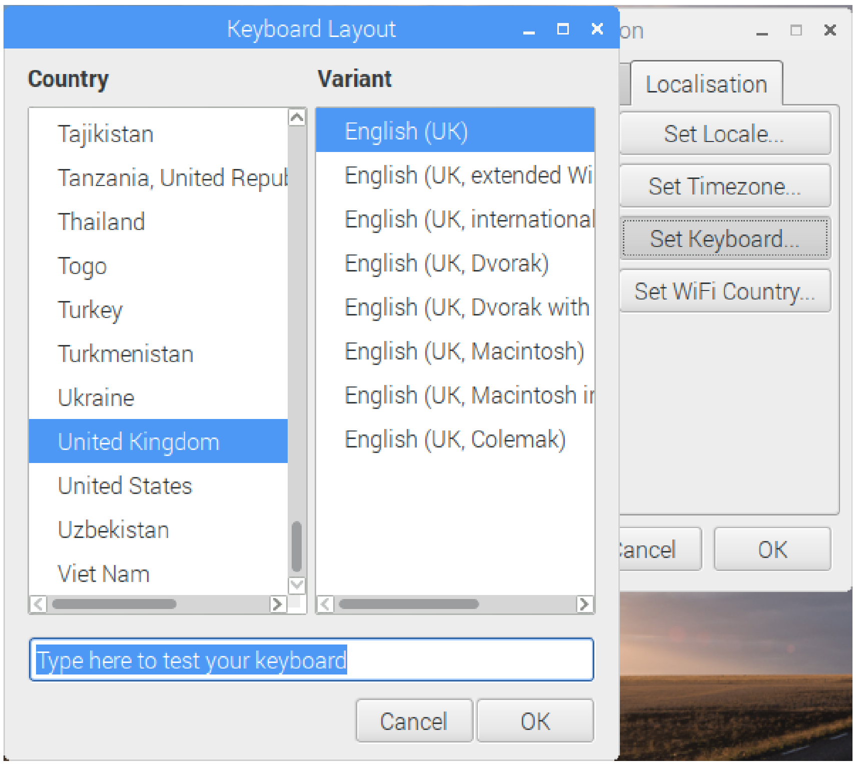
The main choice is done in the first list 'Area'. There are general options like 'GMT' (Greenwich Mean Time) by which you can choose the timezone directly by the standard time scheme. Alternatively you can use more meaningful options related to geological regions like 'Europe'. When choosing 'Europe', the second choice list 'Location' will be reduced automatically to the matching main towns:



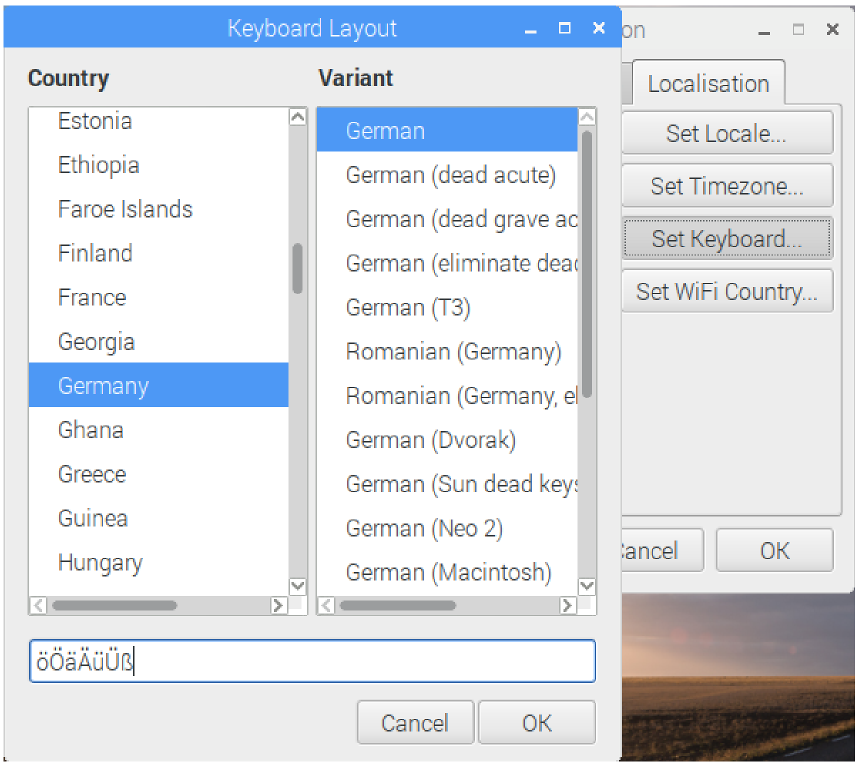
After choosing the matching city, in the given example 'Berlin', a click on 'OK' completes this sub menu.

- **Keyboard**

This one is about the correct adaption to the keyboard layout. One click on 'Set Keyboard' will show up this sub menu:



We will start at the left column 'Country' picking the correct country. After that the right column 'Variant' will show the valid variants for that choice.
Sticking to the example, choosing 'Germany' in 'Country' will show up an astonishing long list of variants:

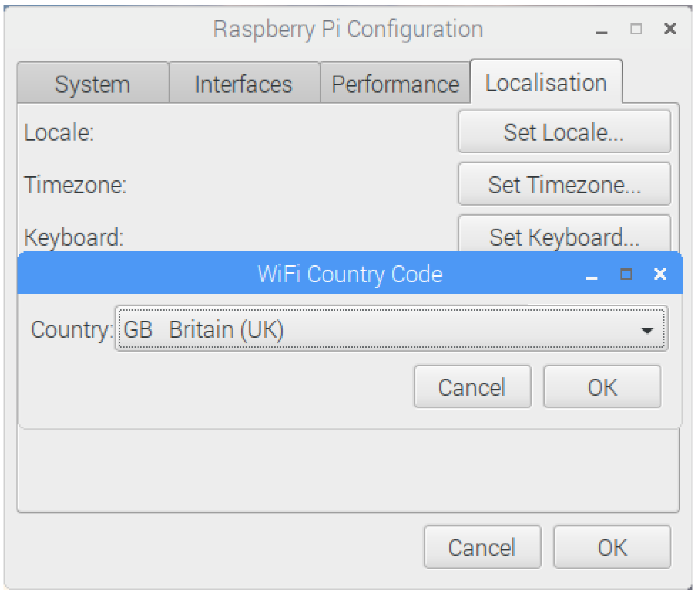


öÖäÄüÜß

This is because there are many different keyboard layouts available with even different numbers of keys. In most cases the first entry is a good starting point. In case of doubts you can test your keyboard by typing to the line marked 'Type here to test your keyboard'. For a german keyboard testing the special keys 'äÄöÖüÜß' will show up quickly whether the chosen option is working accordingly.
Finally a click on 'OK' will complete this menu.

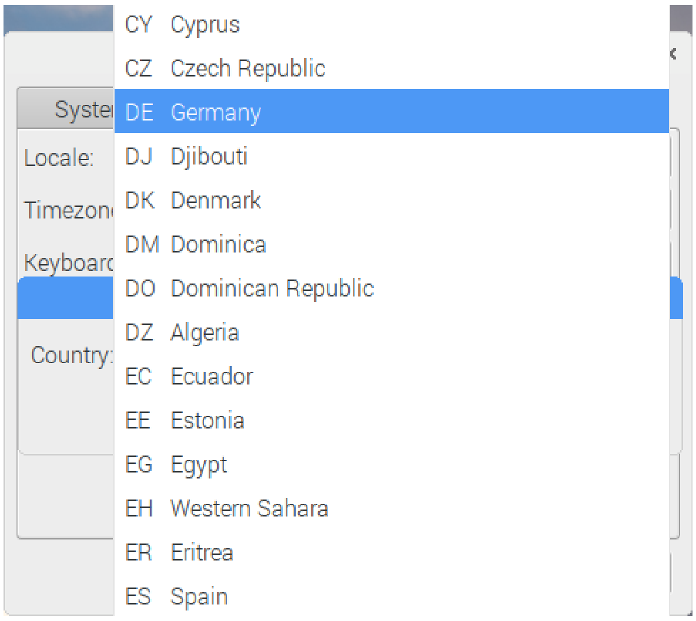
- **WiFi Country**

This menu entry is meant for setting the Wi-Fi module to the correct configuration corresponding to the country it is operated.



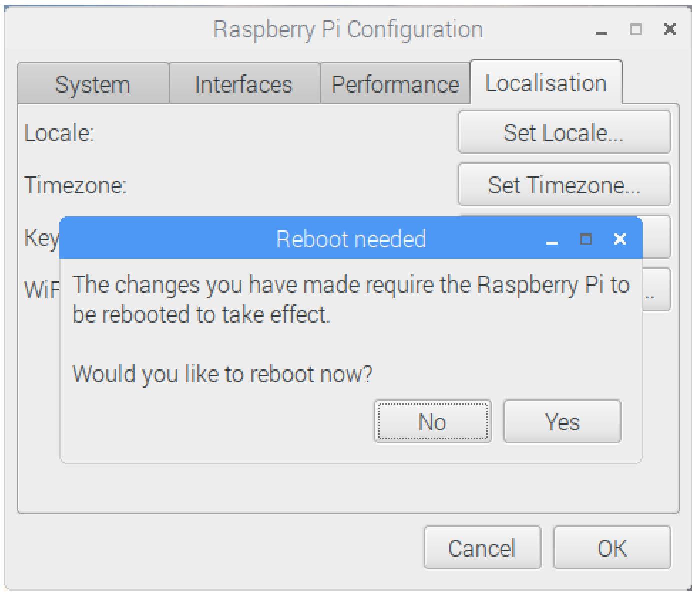
Country: GB Britain (UK)

For example we choose 'DE Germany' again:



One click on 'OK' finishes this sub menu.

Now all necessary settings are done and we can also close the main menu by clicking on 'OK'.



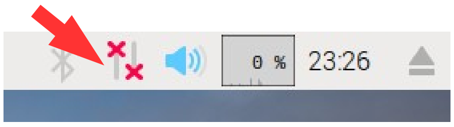
The changes you have made require the Raspberry Pi to be rebooted to take effect.

Would you like to reboot now?

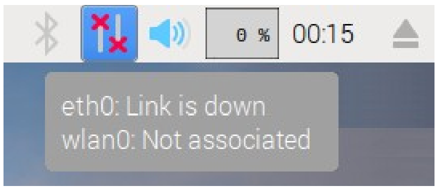
Now a confirmation window appears, which informs us about a necessary reboot to get the new settings effective and asks whether we would like to reboot now. Because we are done with all necessary settings, it's reasonable to do so – just click on 'Yes'.

3.4.4 Connecting to the Internet

As the first configuration is done and the Rasberry Pi is rebooted, it is time to set up and test the network connection.
On the right side of the task bar you will find an icon indicating the status of the network connections:



In the shown example the icon indicates by two red 'x' that no connection is established.
When moving the mouse cursor over the icon, additional informations will pop up:

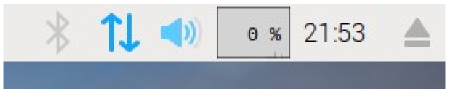


eth0: Link is down
wlan0: Not associated

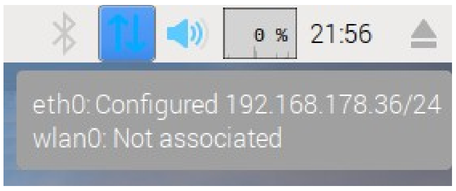
The line 'eth0: Link is down' is related to a direct cable link, stating there is no connection, e.g. no cable plugged in (eth0 = Ethernet interface number 0).
The next line 'wlan0: Not associated' only appears in case there is a Wi-Fi stick attached (as in 3.4.2), the stick was successfully detected and is supported normally (like most of them). For that reason it is already managed as a possible network connection here.
But in the shown example it did not yet establish a working link to a network.

- **Direct cable link**

For establishing a network connection via an ethernet cable you just have to plug in a patch cable to your router. When the cable is plugged in, the link will be established automatically. On that the control light at the Rasberry Pi Ethernet jacket will blink and also the indicator icon will be animated on the screen. A successful connection will be shown like this:



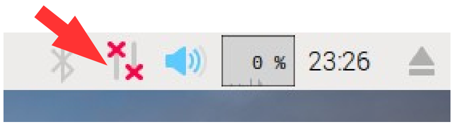
Hovering the mouse cursor on the icon will give this additional information:



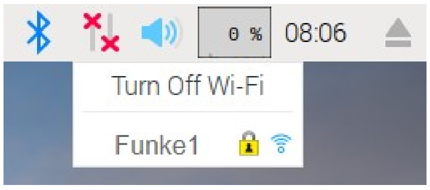
eth0: Configured 192.168.178.36/24
wlan0: Not associated

- **Wi-Fi link**

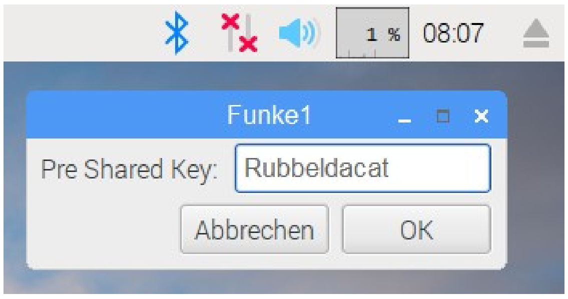
  In the case of a Wi-Fi link now the first registration at the access point is necessary. Therefore we do a left click on the network icon:



Subsequently all found Wi-Fi networks are shown with their protection state and their signal strength:

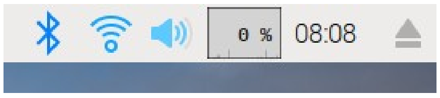


With a left click on the name of your home network you will start the registration process, in this example it's 'Funke1'. On that an input mask opens, requesting the access key:



Pre Shared Key: Rubbeldacat

After typing the access key and finishing the input with a click on 'OK', the window will close. Now the network icon changes to an animated wireless icon, which remains stable in case of a successfully established link.

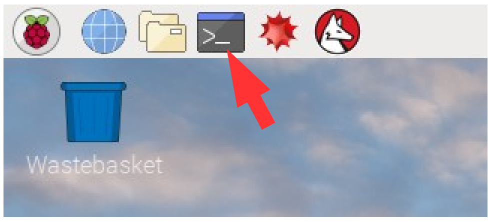


3.4.5 General Update

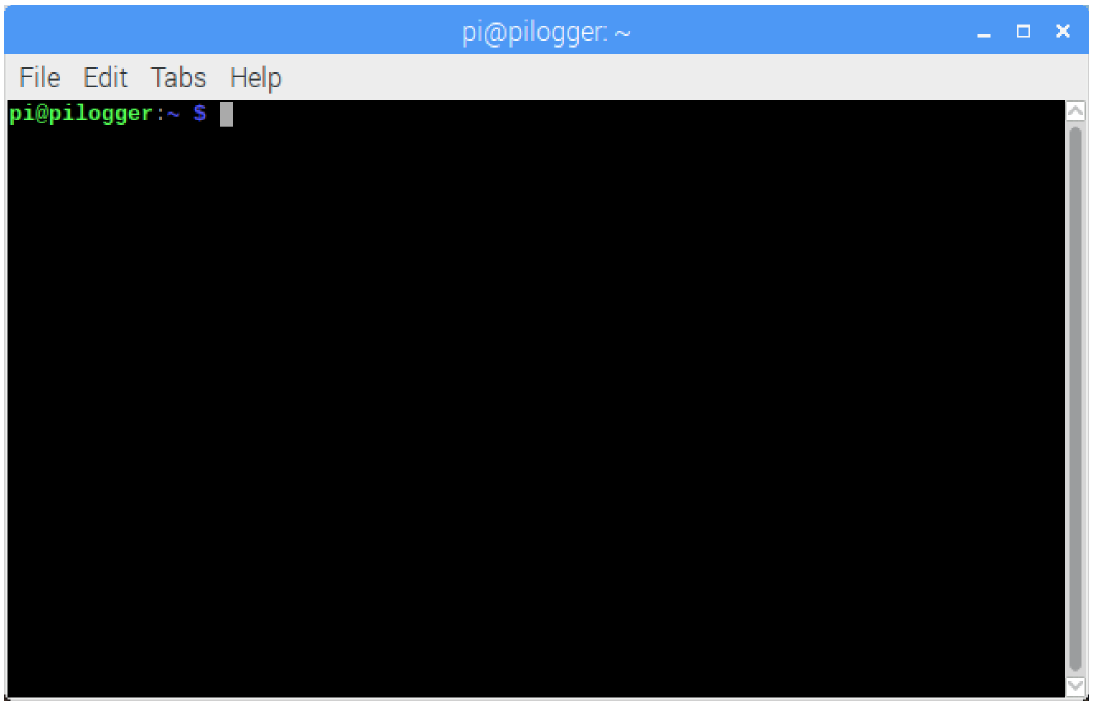
The fresh installation of the operating system will only be really complete and safe after doing a first general update. This means to look for available software updates and install them. Especially security relevant updates are distributed as fast as possible, not waiting for the next planned release of a complete installation image.

The procedures described in the following presume a working internet connection.

The first part is about the so called firmware. This part of software contains the Raspberry Pi specific hardware adaptions for the operating system (the adopted Linux Kernel). For that purpose we start a console window – this is a kind of graphic version of the command line interface. It presents a pure text screen within a window on the desktop.
We start this application by clicking on the console icon in the task bar:



When hovering with the mouse cursor on this icon we will get the additional information that this application is called 'LXTerminal'. That's correct – that's the application we want to start. As already said, this application emulates a simple text screen, like a historical 'terminal' or 'console', which were the first displays for a (remote) 'host' computer.
Exactly this output you also could get remotely via SSH.
This window will look like this:



```
pi@pilogger:~ $
```

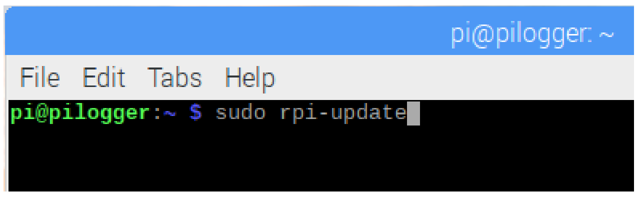
The window title line indicates by the text 'pi@pilogger' that the active session is associated to the user 'pi' logged in on the computer 'pilogger'. The colon and the tilde indicate that at current time the directory 'home' of the current user 'pi' is active and used as a base for path definitions.

The same text we find in the interactive area of the window, followed by the dollar character, which is the original so called prompt. Historically the prompt was just a single character, which asked the user for input. The computer indicates by that, that all tasks are complete and it is ready for new input, especially for the next command.
Nowadays the complete text including the '$' is called prompt – the extended prompt. Behind the prompt, with one space in between, a grey rectangle is blinking – the text cursor. At this place the text will appear, which is typed by the user at the keyboard – this is the command line.

For our task, to update the firmware of the Raspberry Pi, we have to input the following text literally:

```
sudo rpi-update
```

This should look this way on the screen:



```
pi@pilogger:~ $ sudo rpi-update
```

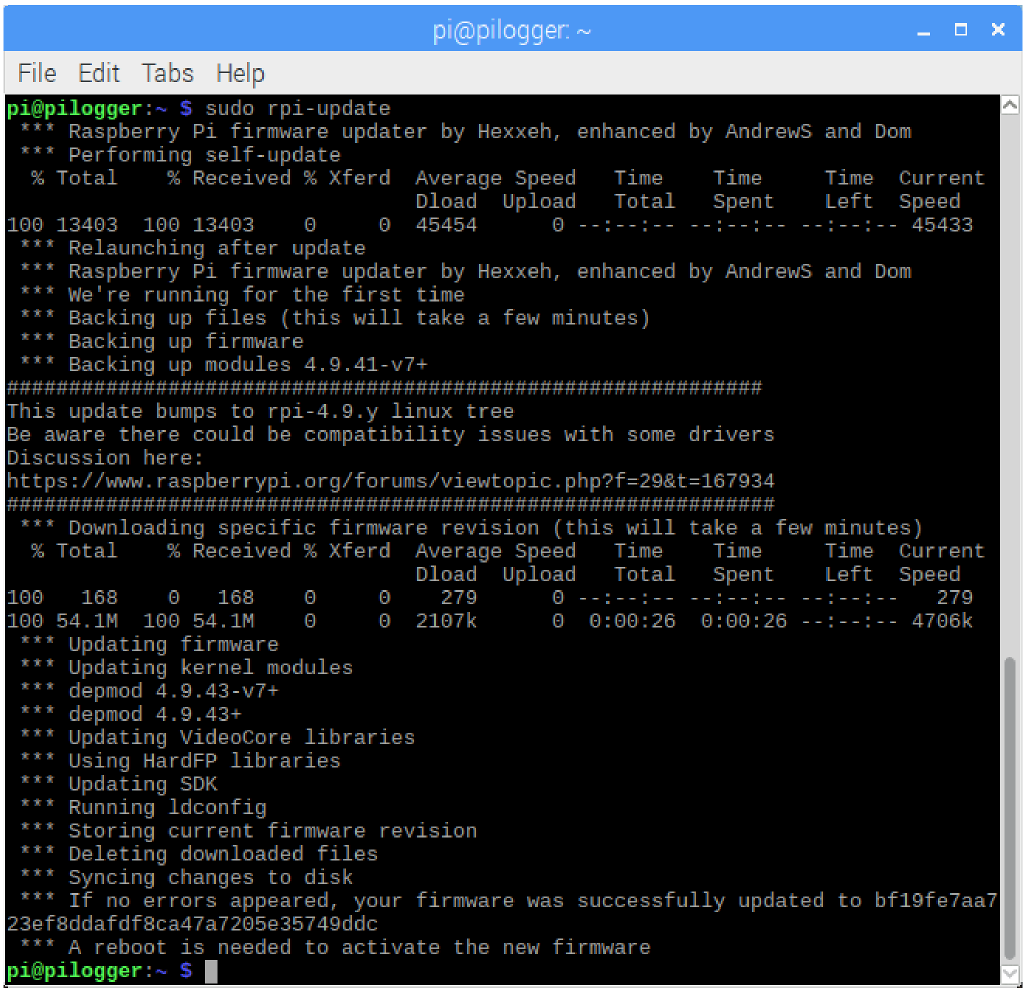
'sudo' is a system command, which assigns super user rights to the following command – in this case the rights of the user 'root'. By this nearly all security limitations are suspended. Normally such a rights elevation demands an authorisation with a special password. But, because the Rasberry Pi was configured to automatic login with the only one administrator account, here the password request is missing.

As already said above, this may be tolerable for a development system – but for a production system, which is meant to work unattended, this is a security risk, which should be avoided by changing this configuration in time !

The second part 'rpi-update' invokes a special tool, which will use an existing internet connection to connect to update server. It will ask for a newer version of the firmware and, if available, start the update process.

The input is completed by pressing the key 'return' (↵). By this the command line is completed and the command is send – the execution starts now.
While the execution of the task several messages are displayed and if there is a new version to install the whole process will take a few minutes.

The complete output of a successful run should look like this:



pi@pilogger: ~

File Edit Tabs Help

```
pi@pilogger:~ $ sudo rpi-update
 *** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
 *** Performing self-update
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 13403  100 13403    0     0  45454      0 --:--:-- --:--:-- --:--:-- 45433
 *** Relaunching after update
 *** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
 *** We're running for the first time
 *** Backing up files (this will take a few minutes)
 *** Backing up firmware
 *** Backing up modules 4.9.41-v7+
#############################################################
This update bumps to rpi-4.9.y linux tree
Be aware there could be compatibility issues with some drivers
Discussion here:
https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=167934
##############################################################
 *** Downloading specific firmware revision (this will take a few minutes)
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   168    0   168    0     0    279      0 --:--:-- --:--:-- --:--:--   279
100 54.1M  100 54.1M    0     0  2107k      0  0:00:26  0:00:26 --:--:-- 4706k
 *** Updating firmware
 *** Updating kernel modules
 *** depmod 4.9.43-v7+
 *** depmod 4.9.43+
 *** Updating VideoCore libraries
 *** Using HardFP libraries
 *** Updating SDK
 *** Running ldconfig
 *** Storing current firmware revision
 *** Deleting downloaded files
 *** Syncing changes to disk
 *** If no errors appeared, your firmware was successfully updated to bf19fe7aa7
23ef8ddafdf8ca47a7205e35749ddc
 *** A reboot is needed to activate the new firmware
pi@pilogger:~ $ 
```

At the end of the output the prompt shows up again – as explained above – to indicate the computer is ready for new input.
The last message of the firmware update process states that a reboot is necessary to activate the new firmware. For the planned second part this reboot is not really needed, but for those, who want to keep on the safe side, here is the command to reboot the Pi instantly:

```
reboot
```

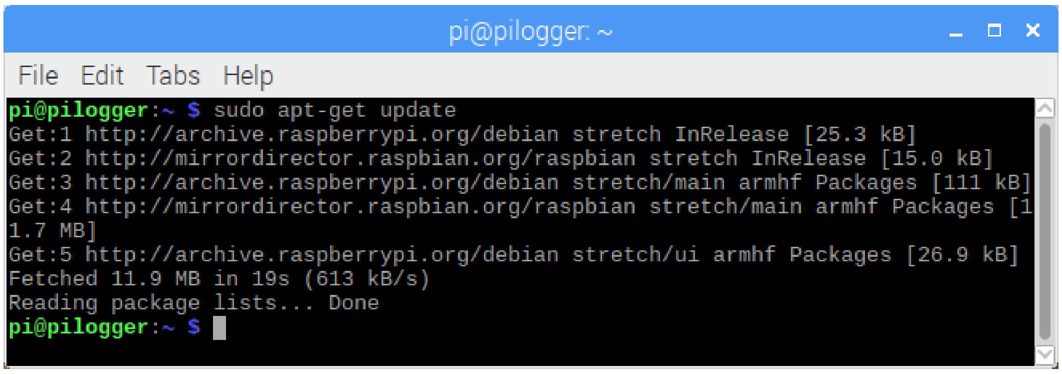
This procedure is fast and easy.

Now we can prepare a general upgrade of the Raspbian distribution. For that the local database of the installed and the available software packages has to be updated.

After a reboot we again have to open a terminal window and now input the following line:

```
sudo apt-get update
```

With that line the tool 'apt-get' is invoked with super user rights and instructed to update the package database. This process will take not much time – the younger the installed Raspbian release, the shorter. The screen output will like this one:

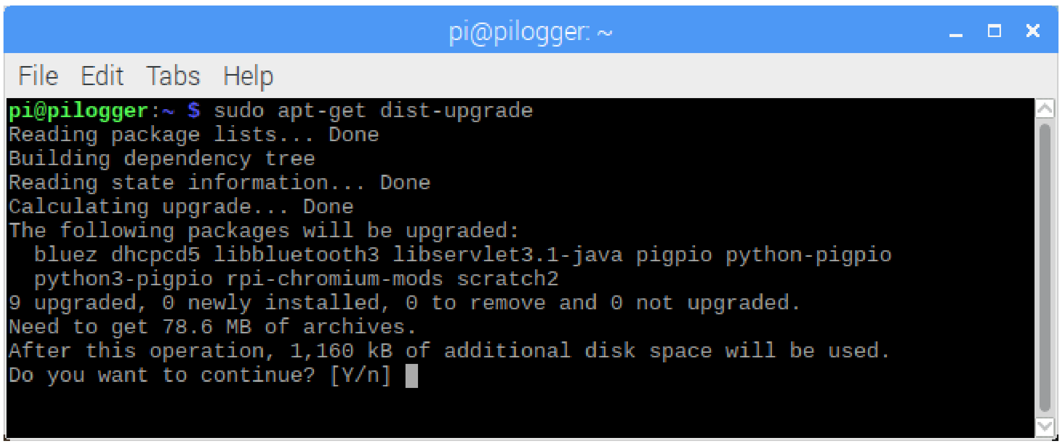


```
pi@pilogger:~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org/debian stretch InRelease [25.3 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian stretch InRelease [15.0 kB]
Get:3 http://archive.raspberrypi.org/debian stretch/main armhf Packages [111 kB]
Get:4 http://mirrordirector.raspbian.org/raspbian stretch/main armhf Packages [1
1.7 MB]
Get:5 http://archive.raspberrypi.org/debian stretch/ui armhf Packages [26.9 kB]
Fetched 11.9 MB in 19s (613 kB/s)
Reading package lists... Done
pi@pilogger:~ $ 
```

Now we continue by inputting this line:

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

By this the real update of the Raspbian installation is started. At first it determines which packets have to be changed. Then it will ask you whether these changes shall actually be installed:



```
pi@pilogger:~ $ sudo apt-get dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  bluez dhcpcd5 libbluetooth3 libservlet3.1-java pigpio python-pigpio
  python3-pigpio rpi-chromium-mods scratch2
9 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 78.6 MB of archives.
After this operation, 1,160 kB of additional disk space will be used.
Do you want to continue? [Y/n] 
```

After giving the reply 'Y' with completing by 'return', the main part of the update will start. Depending on how fresh the used Raspbian image is, there might be a lot of packages to be updated. This could last quite long – could be around one hour. The shown example is done on a real fresh installation, so it's quite short.
After finishing this update, including an eventual reboot, the basic set up of your Rasberry Pi is completed.

4 Setting Up the PiLogger Base Software

To use the PiLogger base software some preconditions have to be fulfilled, which conveniently all are pre-installed since Raspbian Version 'Stretch':

- The I²C kernel driver to support this hardware interface must be loaded
  → which we already activated in 3.4.3.2 with 'raspi-config'
- The command line tool 'i2c tools' is installed (details in attachment 13.1)
- The I²C support for Python 'python3-smbus' is installed
- Python with version 3.x is installed
- The graphic extension 'tkinter' for Python is installed

Therefore we could start immediately – but sometimes things go wrong when updating, so in the next step we have to activate an alternative I²C driver first.

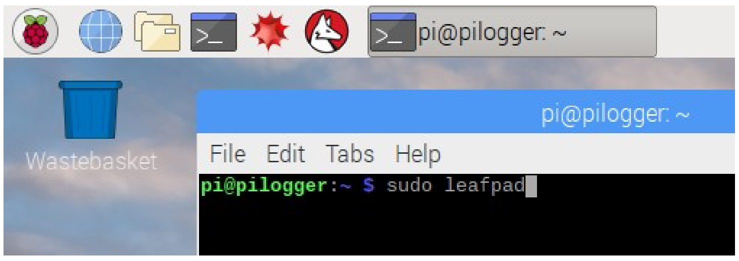
4.1 Activating an alternative driver for I²C

Since Raspbian changed to kernel version 4.9.x by default a new driver for the I²C interface is configured, which at this time has an issue.
As long as this problem persists we have to implement a workaround by using the older driver (fallback).

For that we open a terminal window and input the line:

```
sudo leafpad
```

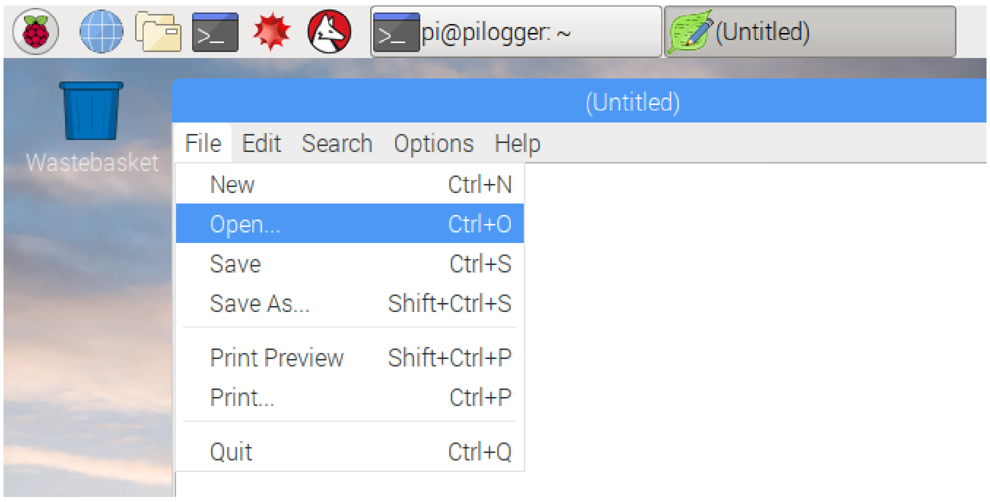
This will look like this:



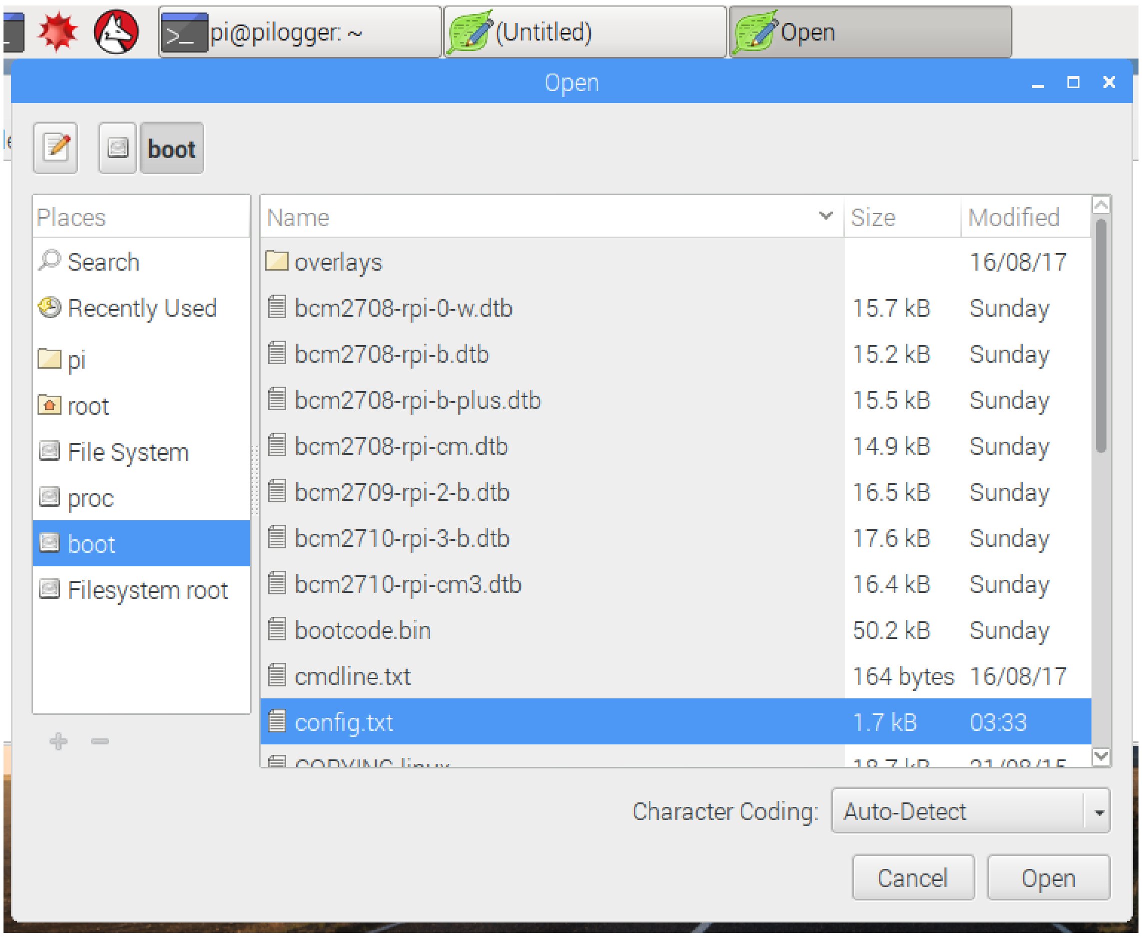
```
pi@pilogger:~ $ sudo leafpad
```

This command opens the text editor 'leafpad' with administrator rights, so we can edit a special system file.

Within the now opened editor window we go to 'File' → 'Open' :



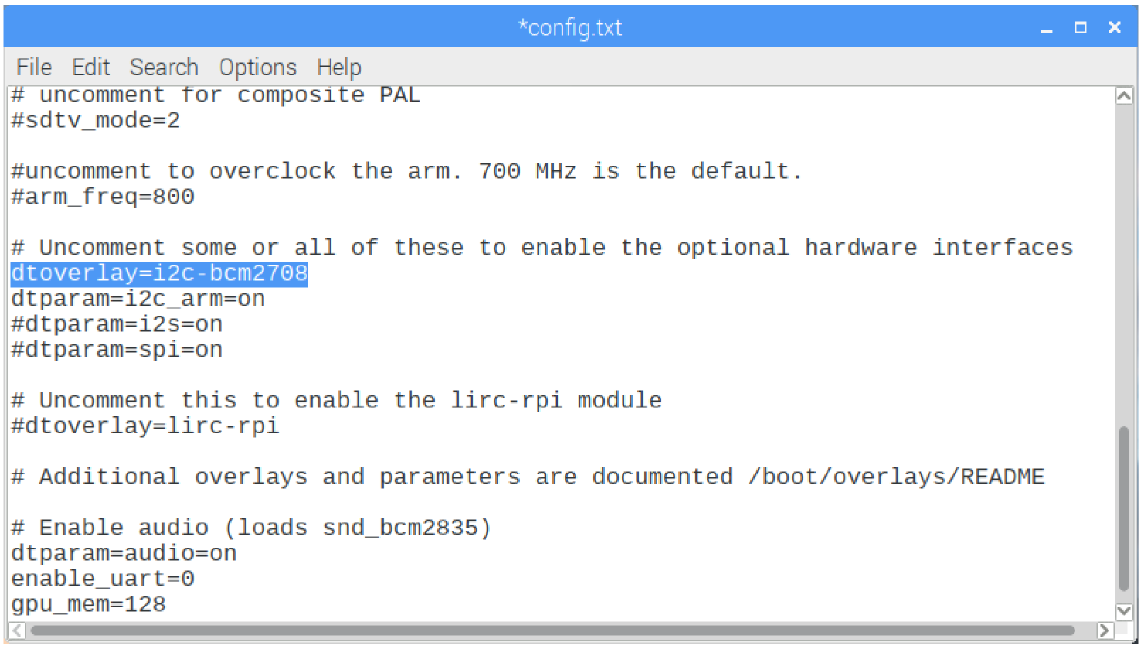
This will open an additional dialogue window in which we will browse for the text file:



Name	Size	Modified
overlays		16/08/17
bcm2708-rpi-0-w.dtb	15.7 kB	Sunday
bcm2708-rpi-b.dtb	15.2 kB	Sunday
bcm2708-rpi-b-plus.dtb	15.5 kB	Sunday
bcm2708-rpi-cm.dtb	14.9 kB	Sunday
bcm2709-rpi-2-b.dtb	16.5 kB	Sunday
bcm2710-rpi-3-b.dtb	17.6 kB	Sunday
bcm2710-rpi-cm3.dtb	16.4 kB	Sunday
bootcode.bin	50.2 kB	Sunday
cmdline.txt	164 bytes	16/08/17
config.txt	1.7 kB	03:33

In the left column 'Places' we now click on 'boot', which show up the content of that folder (in this case actually a partition) on the right panel.
In the right panel we chose the file 'config.txt' and then click right at the bottom on the button 'Open’.

Now the file is opened and we scroll to that part of the text, where the I²C interface already was activated by the tool 'raspi-config':
'dtparam=i2c_arm=on'
and insert a new line before:



```
# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
dtoverlay=i2c-bcm2708
dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
enable_uart=0
gpu_mem=128
```

'dtoverlay=i2c-bcm2708'

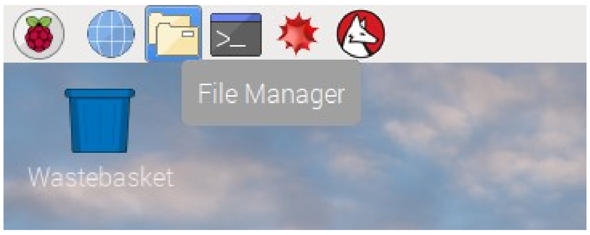
After that we save the changed file ('File' → 'Save') and close the text editor ('File' → 'Quit').
Now we also close the terminal window and reboot the Rasberry Pi.
By this now the older I²C driver is activated.

4.2 Setting Up the PiLogger Base Software

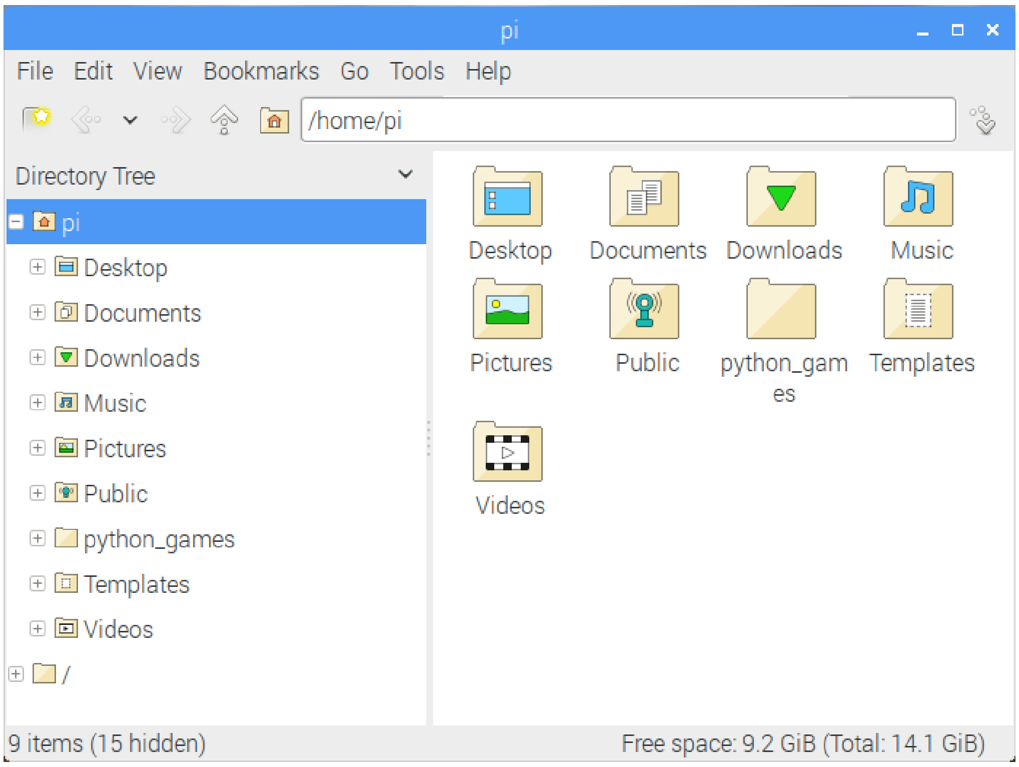
The PiLogger base software does not need a real installation. You only have to retrieve an archive file, which should be unpacked into a folder in a reasonable place. The recommended place from this manual should be followed in a first step to keep consistent – choosing a different place can be done easily later.

4.2.1 Create a New Folder

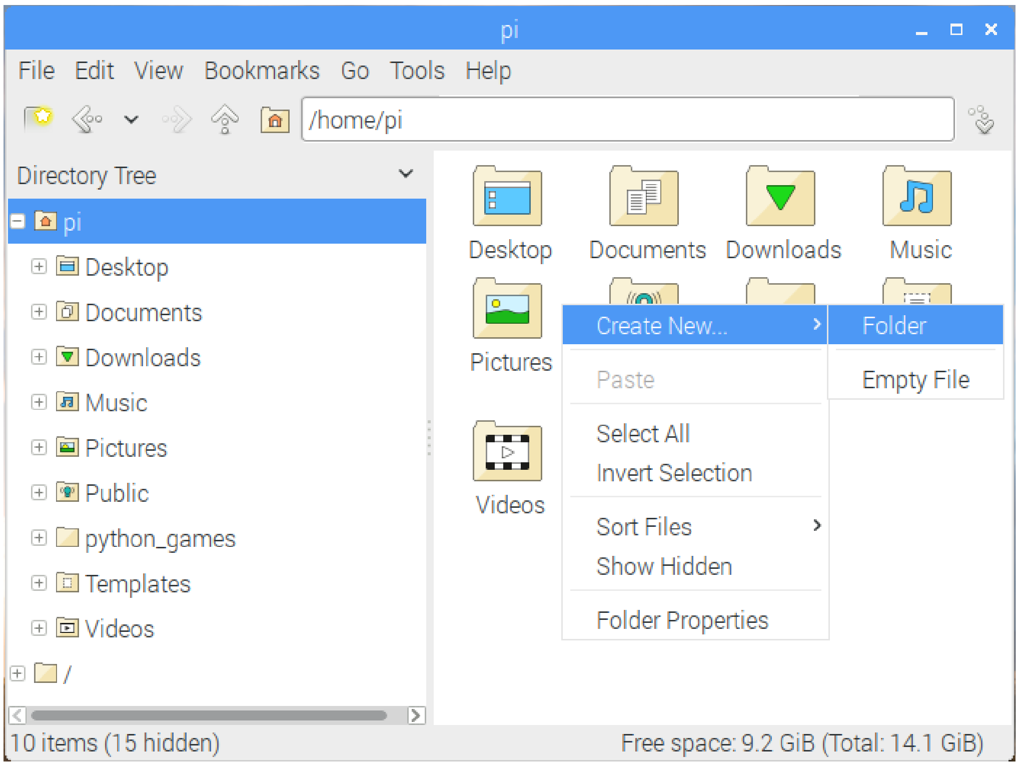
In a first step we prepare that mentioned target folder. We start the application 'file manager' by a mouse click on the folder icon in the task bar:



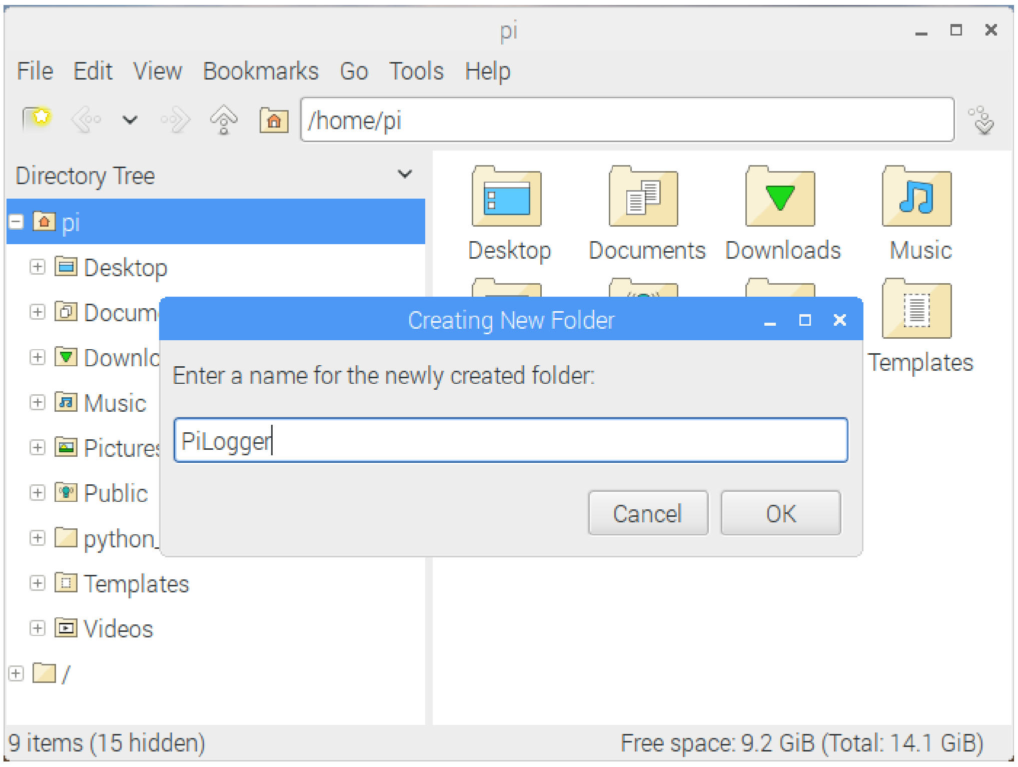
By default the file manager will set his start up window to the directory '/home/pi' – the home folder for the user 'Pi':



This is a good place to create a new folder for the PiLogger. To do so, we do a right mouse click in the main window to open the context menu and move via 'Create New' to the right to 'Folder':



After a left mouse click a dialogue appears to input the name of the new folder:

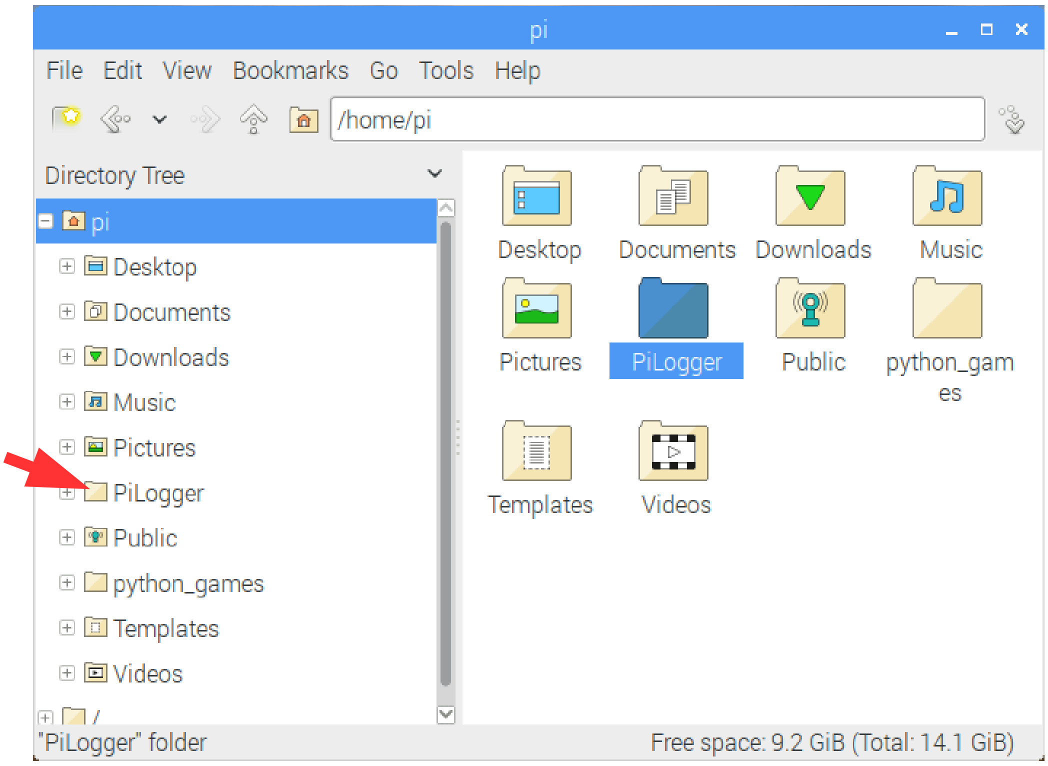


Creating New Folder

Enter a name for the newly created folder:

PiLogger

After typing 'PiLogger' to the input mask and clicking on the button 'OK', the requested new folder integrates in this way:

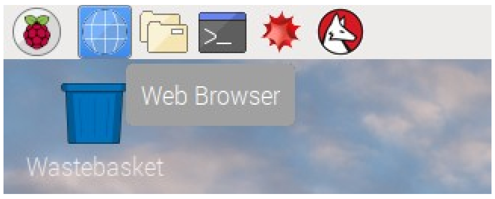


Now this is the home folder for the base software. The file manager can be closed now (*clicking on the X in the upper right corner*).

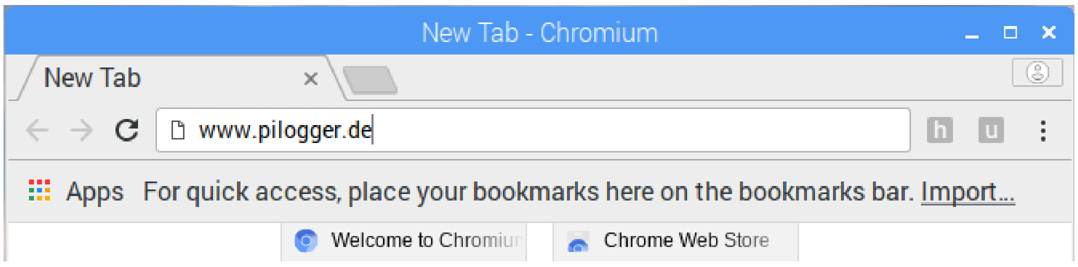
4.2.2 Download the Archive

The archive containing the necessary files is named 'PiLogger_1v0_EN.zip' (*or a respective newer version*) and is provided at the same place where also this manual was found: www.pilogger.de → 'Download' → 'Software'.

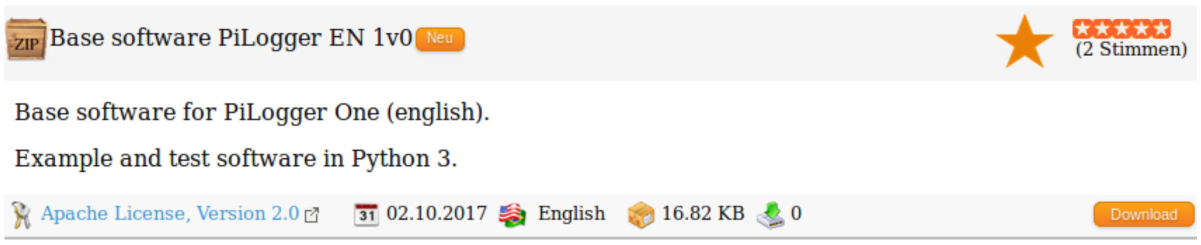
Because Raspbian comes with a web browser already installed, we can use it directly if a connection to internet is available. To do so, we start the 'Web Browser' by clicking on the globe icon in the task bar:



This will open a window with the application 'Chromium', which is the open source fork of the Google browser 'Chrome'. We input 'www.pilogger.de' to the address field:



At the PiLogger home page we navigate to 'Download', where we branch to the category 'Software' and finally find this archive:


Base software PiLogger EN 1v0 Neu

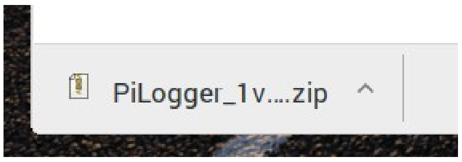
(2 Stimmen)

Base software for PiLogger One (english).

Example and test software in Python 3.

Apache License, Version 2.0 02.10.2017 English 16.82 KB 0

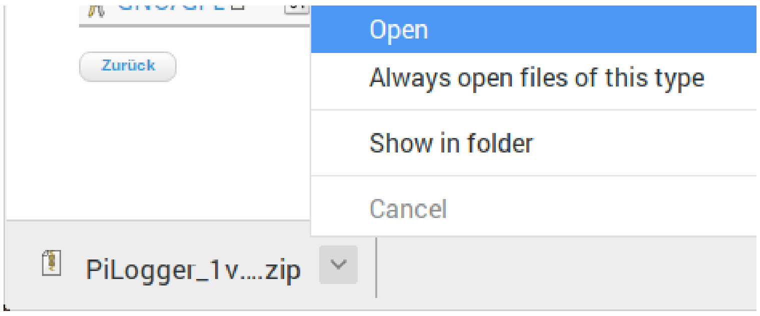
After clicking on the button 'Download' the process will start in the background and just shows the result in the left down corner:



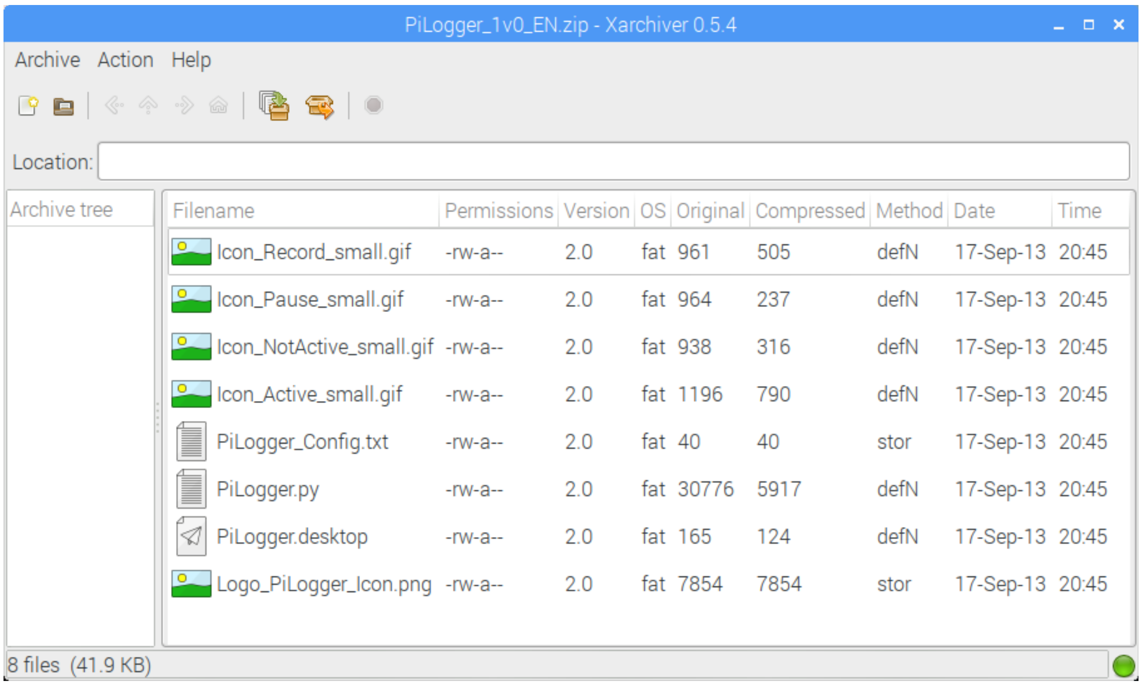
So the archive was downloaded successfully. Now we have to unpack it.

4.2.3 Unpack the Archive

Directly within the message box we now click on that small upwards pointing arrow at the right to open a list with possible actions:



We choose the option 'Open' by clicking on it. By this the archive file is directly opened with the tool 'Xarchiver' and we at once can see the content of it:

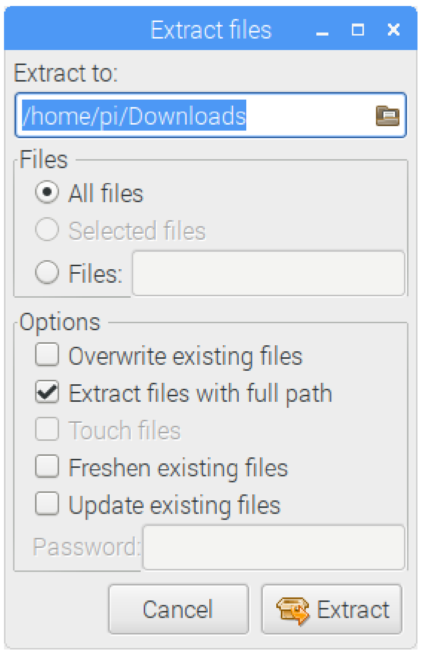


Filename	Permissions	Version	OS	Original	Compressed	Method	Date	Time
Icon_Record_small.gif	-rw-a--	2.0	fat	961	505	defN	17-Sep-13	20:45
Icon_Pause_small.gif	-rw-a--	2.0	fat	964	237	defN	17-Sep-13	20:45
Icon_NotActive_small.gif	-rw-a--	2.0	fat	938	316	defN	17-Sep-13	20:45
Icon_Active_small.gif	-rw-a--	2.0	fat	1196	790	defN	17-Sep-13	20:45
PiLogger_Config.txt	-rw-a--	2.0	fat	40	40	stor	17-Sep-13	20:45
PiLogger.py	-rw-a--	2.0	fat	30776	5917	defN	17-Sep-13	20:45
PiLogger.desktop	-rw-a--	2.0	fat	165	124	defN	17-Sep-13	20:45
Logo_PiLogger_Icon.png	-rw-a--	2.0	fat	7854	7854	stor	17-Sep-13	20:45

8 files (41.9 KB)

The file 'PiLogger.py' is the actual Python program. All other files are additional resource files, which will be explained later.

To unpack all files to the previously prepared folder, we now click most right active icon in the upper icon bar. This will open a dialogue looking like this:

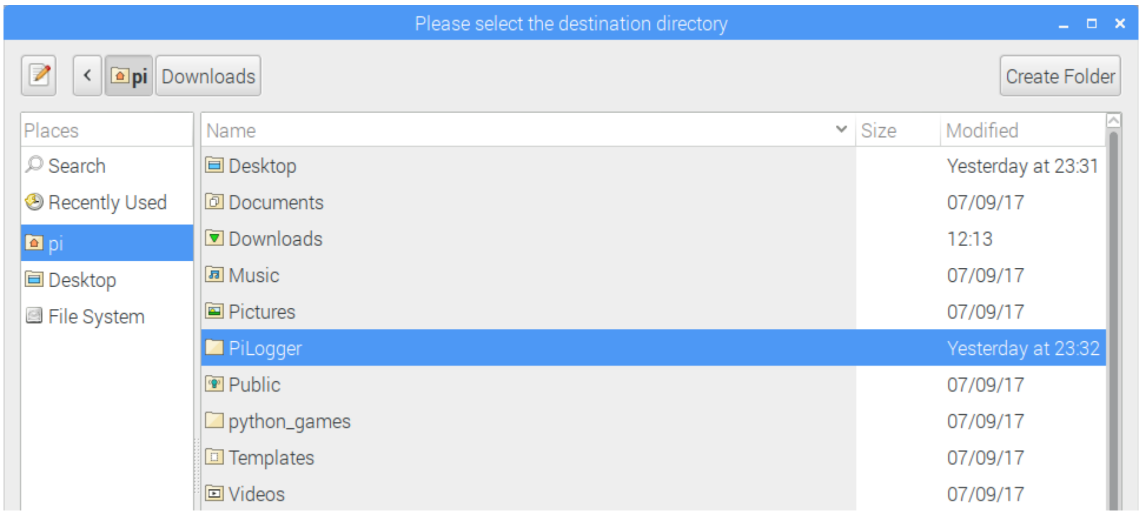


Extract to:
/home/pi/Downloads

Files
● All files
○ Selected files
○ Files:

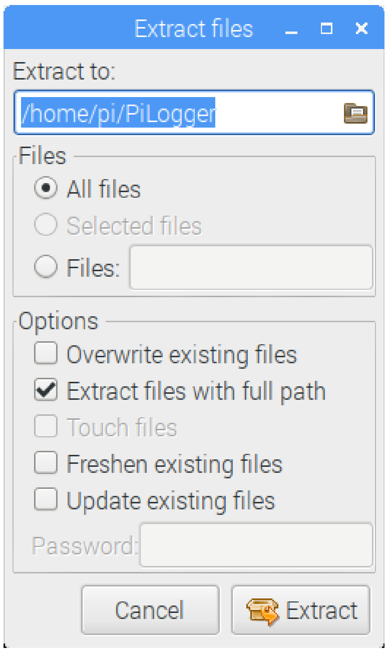
Options
☐ Overwrite existing files
☑ Extract files with full path
☐ Touch files
☐ Freshen existing files
☐ Update existing files
Password:

The option 'All files' in the section 'Files' has to be checked, especially if before one of the files was selected. After that we click on the folder icon right in the field 'Extract to:', as here the folder of origin is still given. Now this dialogue will show up:

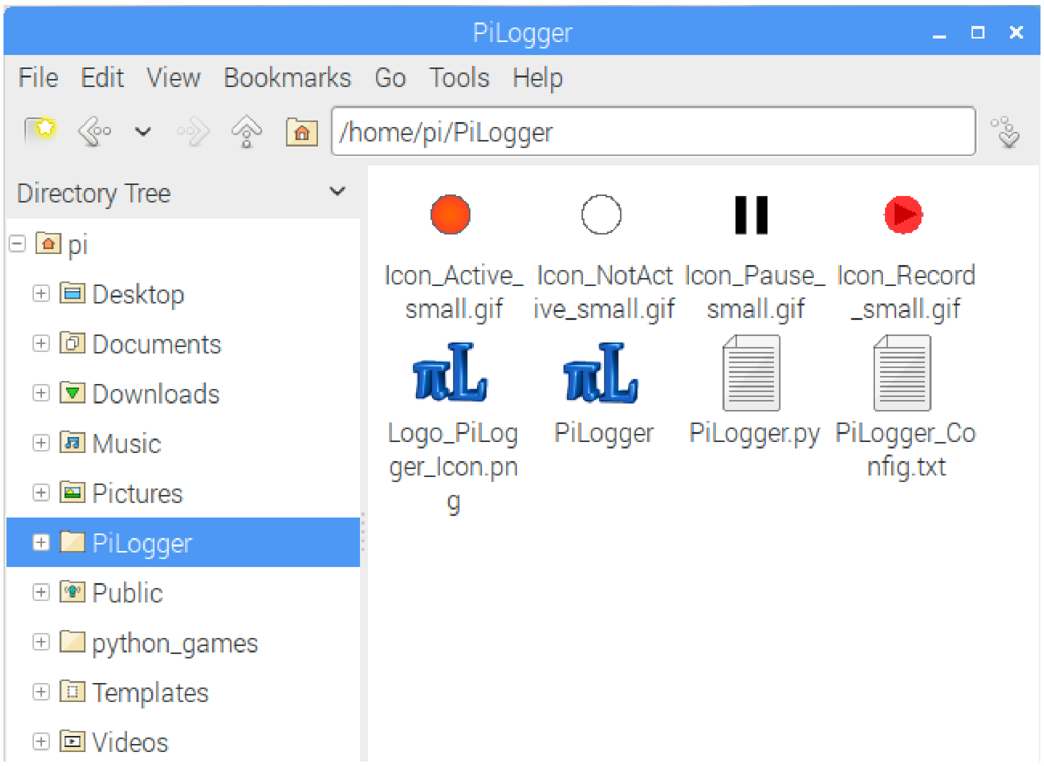


Name	Size	Modified
Desktop		Yesterday at 23:31
Documents		07/09/17
Downloads		12:13
Music		07/09/17
Pictures		07/09/17
PiLogger		Yesterday at 23:32
Public		07/09/17
python_games		07/09/17
Templates		07/09/17
Videos		07/09/17

Here we navigate to the place 'pi' and then to the folder 'PiLogger'. After selecting it via a left mouse click we can click on the button 'Open'. By that the windows closes and the path will be taken over to the previous dialogue.



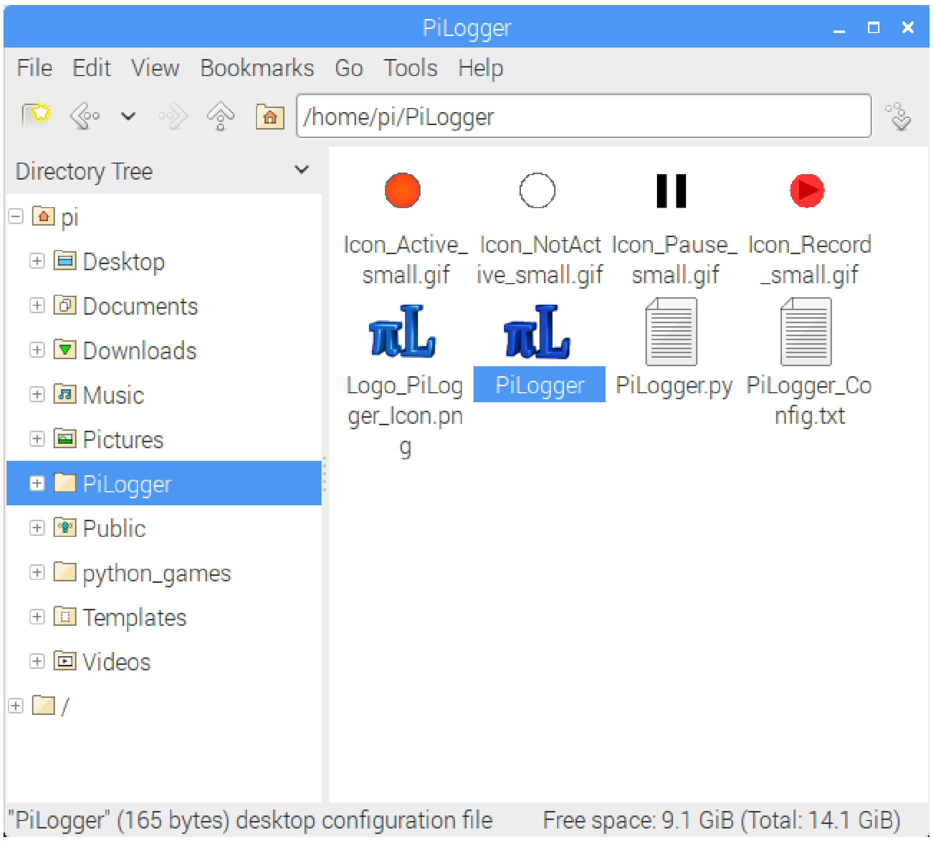
Now we click on the button 'Extract' in the down right corner. The dialogue window closes and the extract process is started. This doesn't take much time and won't show further messages. We close the 'Xarchiver' and open the file manager instead. Here we navigate again to our folder 'PiLogger':



Now here we see the extracted files with the view option 'Icon'.

4.2.4 Setting Up a Desktop Starter

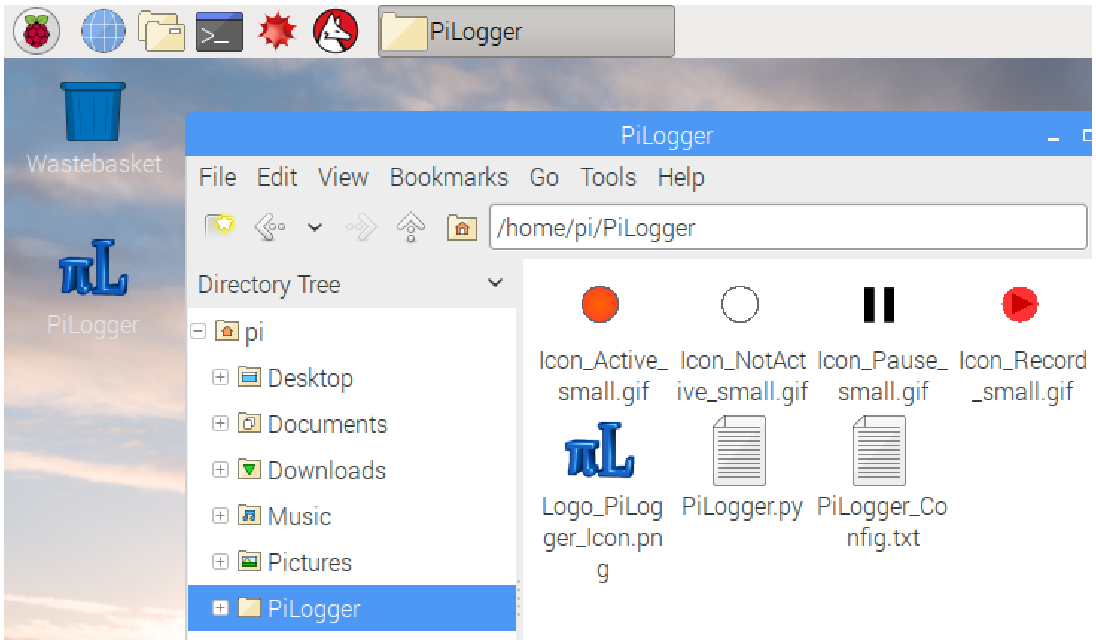
One of the additional files has the file type suffix '.desktop'. This type of suffix is not shown in the file manager, because this is a special file for a defined use – a system file. Viewing at the file with the 'Xarchiver' showed the real type suffix, but at the file manager this suffix is suppressed and instead a system file icon is shown, or as with our file, the icon referenced by the file is shown – in this case the 'PiLogger' icon.
This file is meant to place an icon on the desktop by a selected application can be started directly. If we select this file in the folder 'PiLogger' by a left mouse click, the status line of the file manager informs us, that this file is a desktop configuration file:



"PiLogger" (165 bytes) desktop configuration file Free space: 9.1 GiB (Total: 14.1 GiB)

We now again click the selected file with the left mouse button and hold that button pressed – now we can move the icon attached to the mouse pointer and drag it simply to the free space on the desktop under the wastebasket icon. There we release the mouse button (Drag and Drop). By this the file was moved from the folder 'PiLogger' to the folder 'Desktop'.

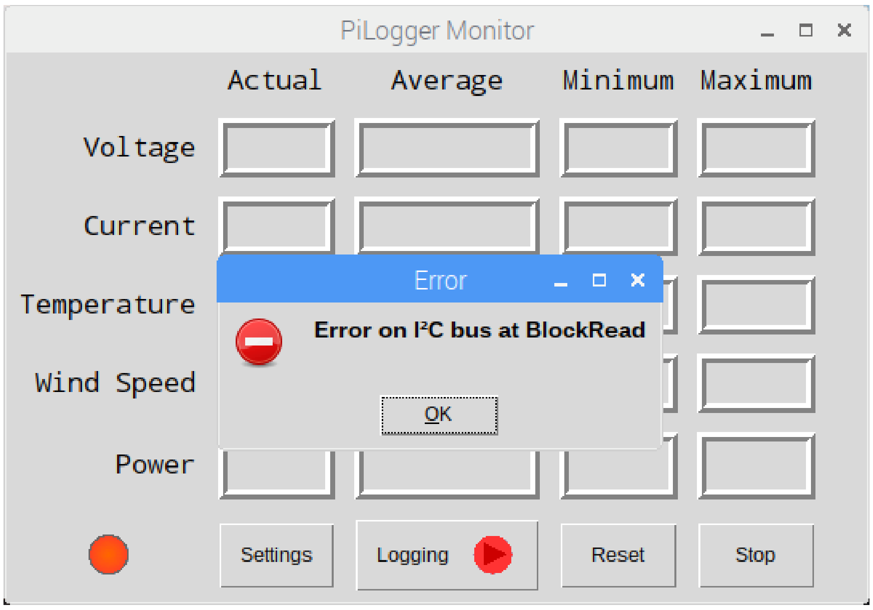
Now it should look like this:



Now we can close the file manager window.
The PiLogger base software setup is completed.

4.2.5 First Test Run (Without Hardware)

To check whether up to now all works fine and to get a first impression, we now test the desktop starter by double-clicking on it. As a first reaction the mouse pointer should show an hour glass indicating activity. Then in the middle of the screen these two windows should appear:



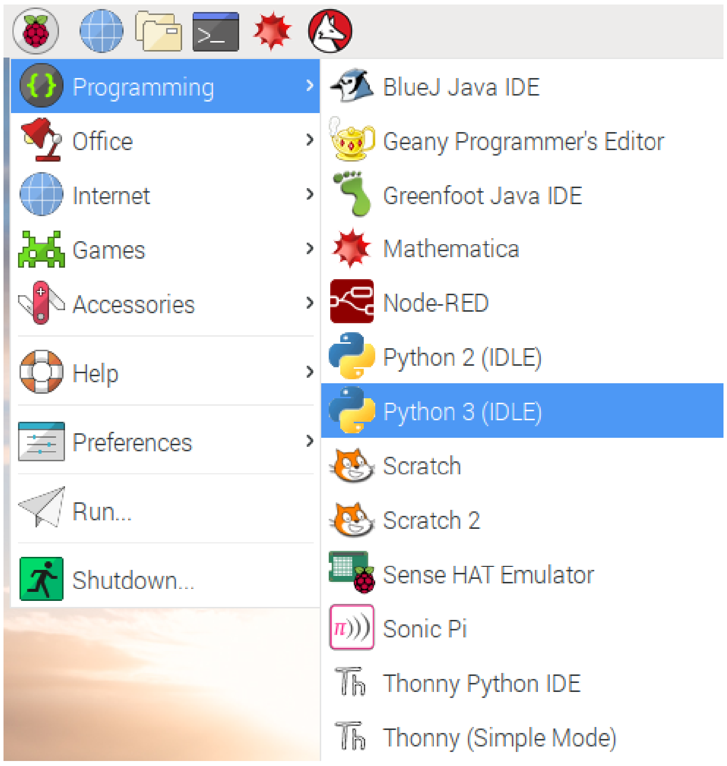
Error on I²C bus at BlockRead

The first and bigger window is named 'PiLogger Monitor', which is the main window of the base software. The second window laid over on top is an error message generated by the application, stating that there was an error on the I²C bus. This not surprising, as we did not install the PiLogger at all up to now.

Therefore this is ok and we acknowledge this by clicking on the 'OK' button.
By that both windows will be closed, because the application is terminated.
So up to now all things went fine.

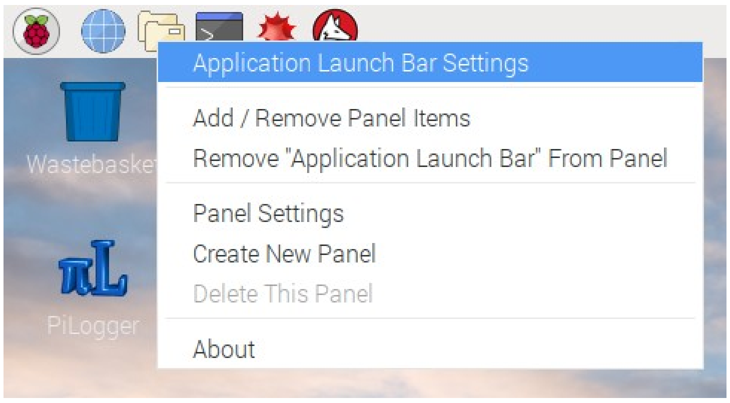
4.2.6 Quick Starter for Python

To view and edit the base software, we need a so called integrated development environment for Python. This could be the classical 'IDLE for Python 3' or since Raspbian Stretch we could also choose 'Thonny'. These applications normally are invoked via the start menu:

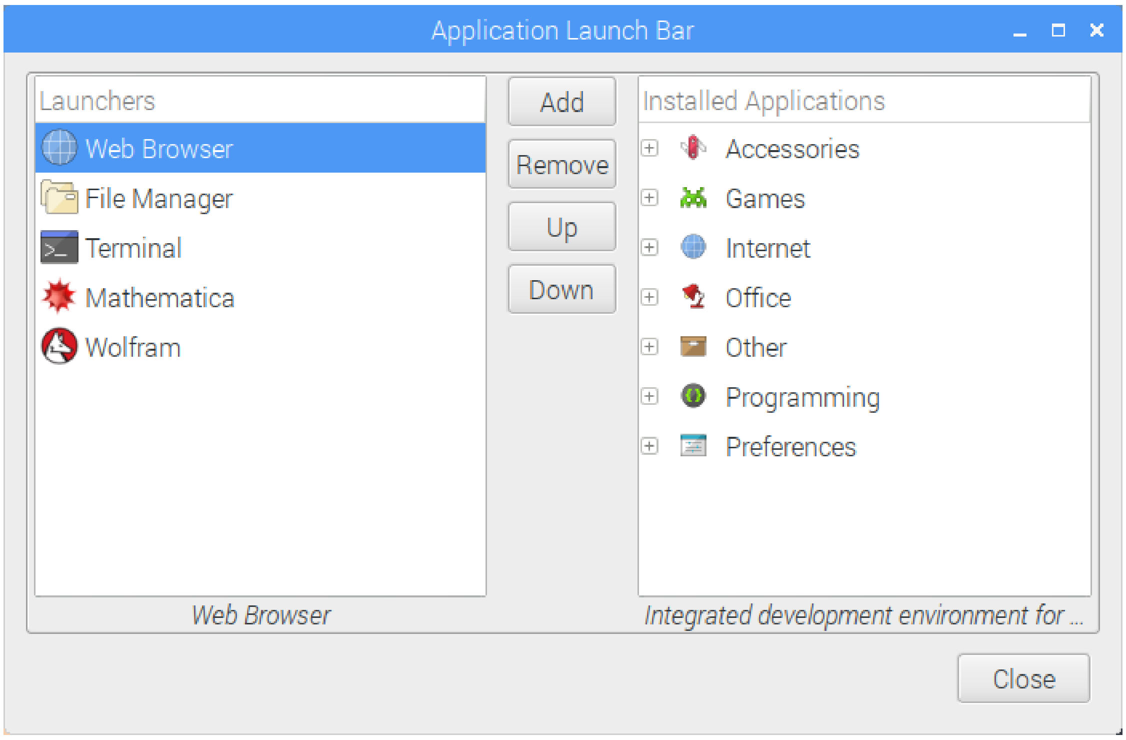


So it’s: 'Menu', 'Programming' → 'Python 3 (IDLE)'.

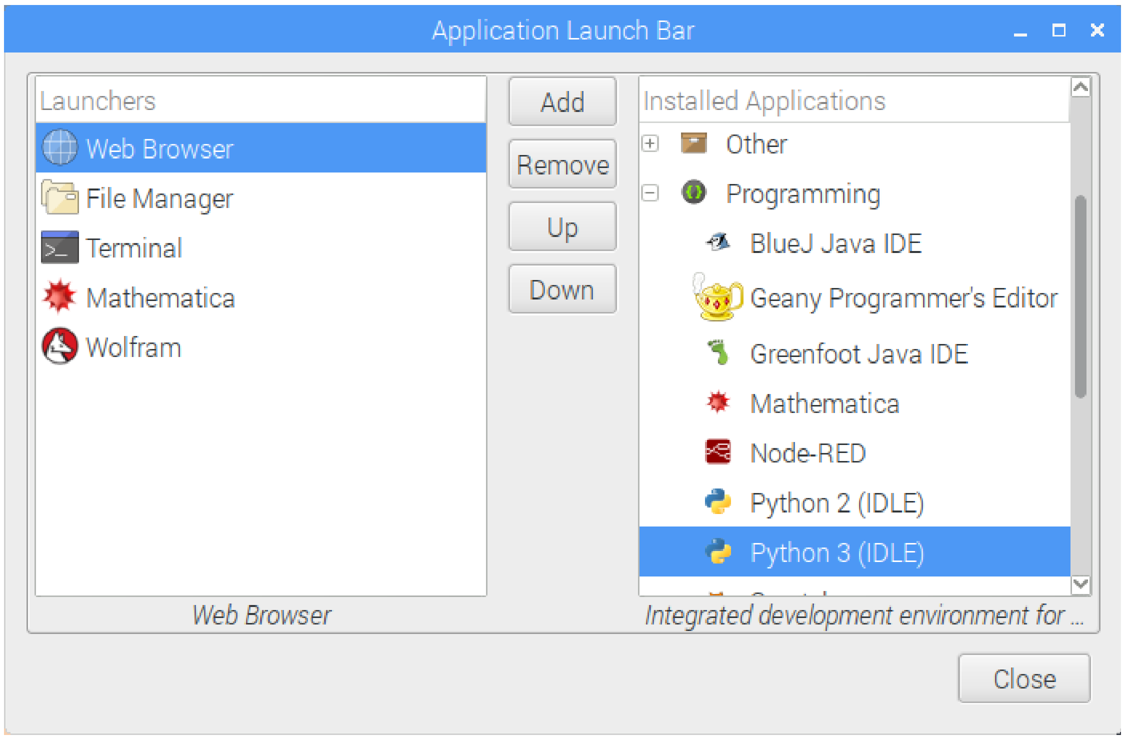
This can be made more comfortable by adding a quick start icon to the task bar like those for the web browser, the file manager or the terminal.
To do so, we click at on of those already existing quick start icons with the right mouse button and by that open the context menu:



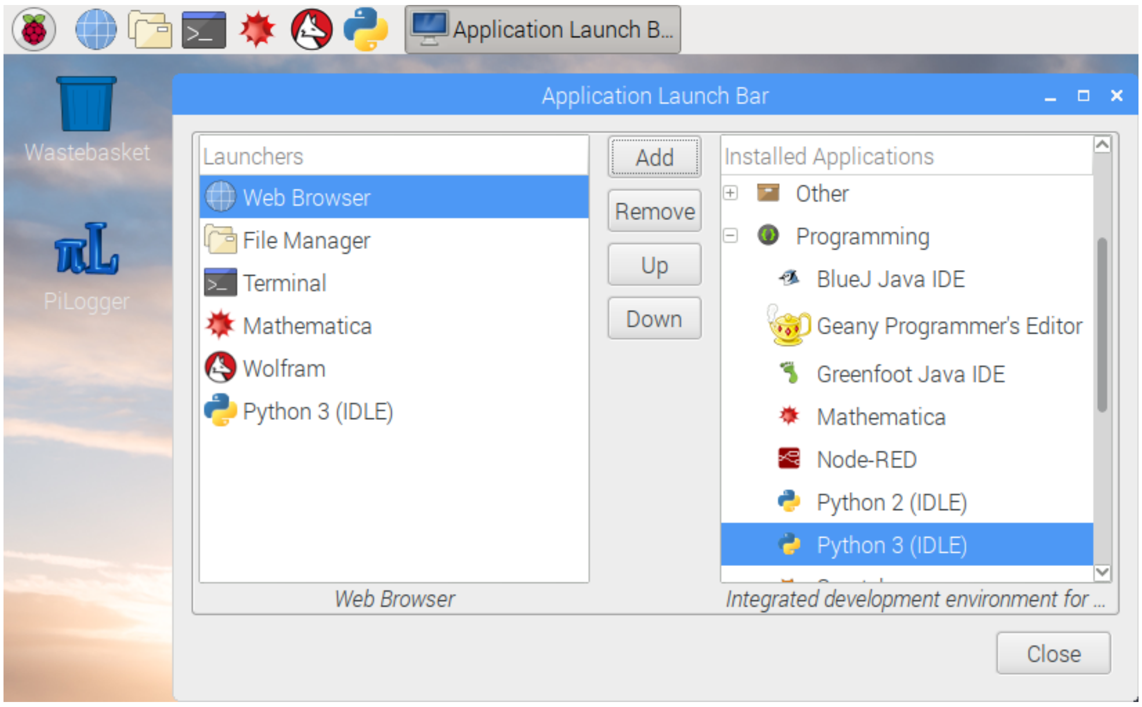
Here we navigate to the first item 'Application Launch Bar Settings' and click on it. By that this window will appear:



In the left column we now see all quick start icons already defined and in the right column all additionally available quick start icons, arranged in a tree structure. In the right column we now click on that small '+' sign in front of 'Programming':



In the now opened list we select the line 'Python 3 (IDLE)' and then click on the button 'Add' in the middle:



This will cause the quick start icon to appear in the left column on one hand and also to appear in the task bar on the other. We now close this window by clicking on the button 'Close'.

Now the Python development environment can be started by one simple click. Just test it!

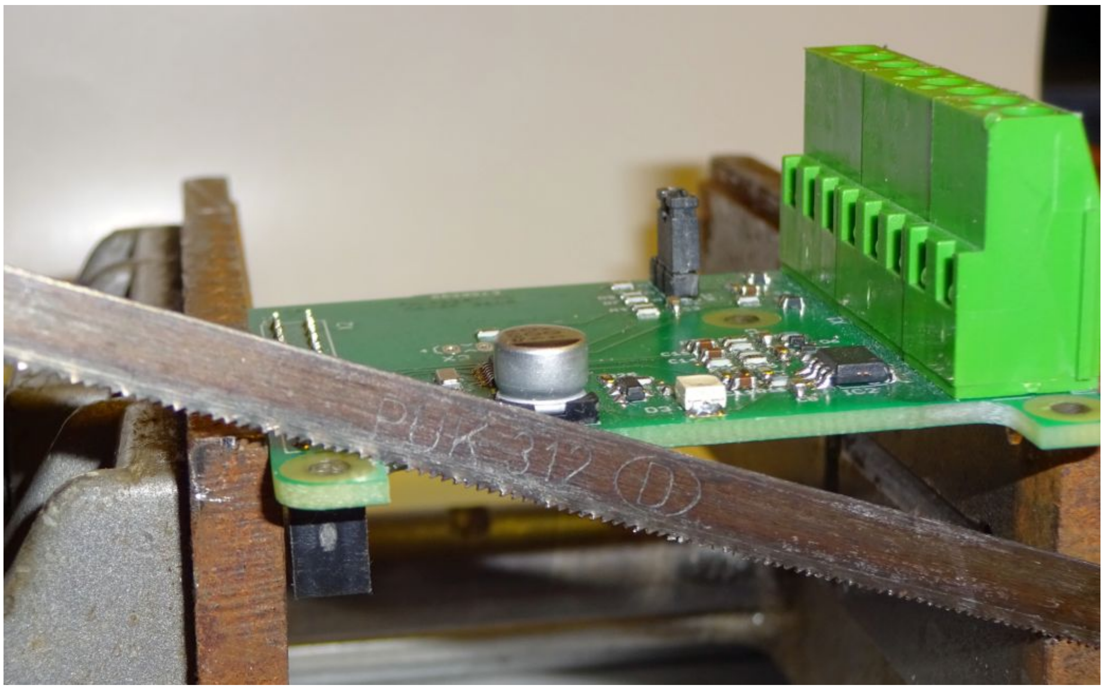
5 Installation and First Operation

5.1 Preparation

The PiLogger One module is suitable as well for the Raspberry Pi 1 which provides a 26 pole pin header, as for the more recent Raspberry Pi's providing a 40 pole pin header.

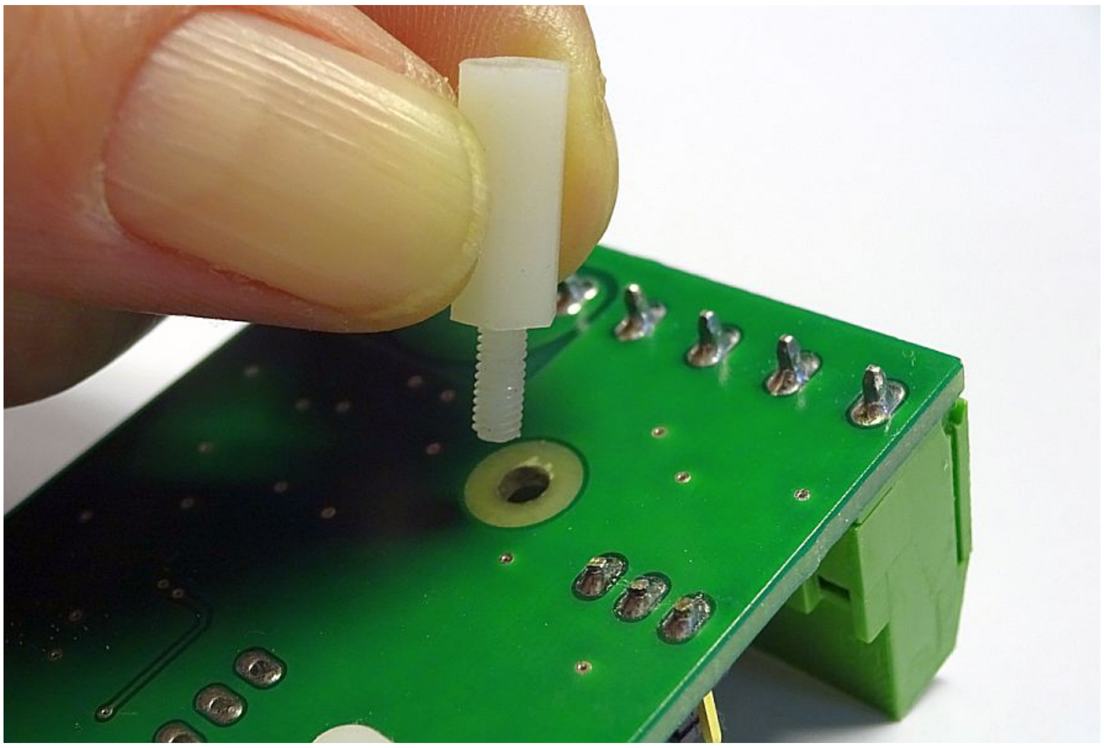
If it is used with a Rasberry Pi with 26 pole pin header, only one screw hole close to the terminal block can be used. The two overhanging edges with screw holes in this case are useless.

If needed, for instance when the combination shall be installed into a small enclosure, these two edges could be removed carefully. This could be done with a small hand hacksaw, presumed the module is fixated safe, but without too much pressure – expressively without bending ! Take care not to damage the pin header socket or any other components !
This should look similar to this:



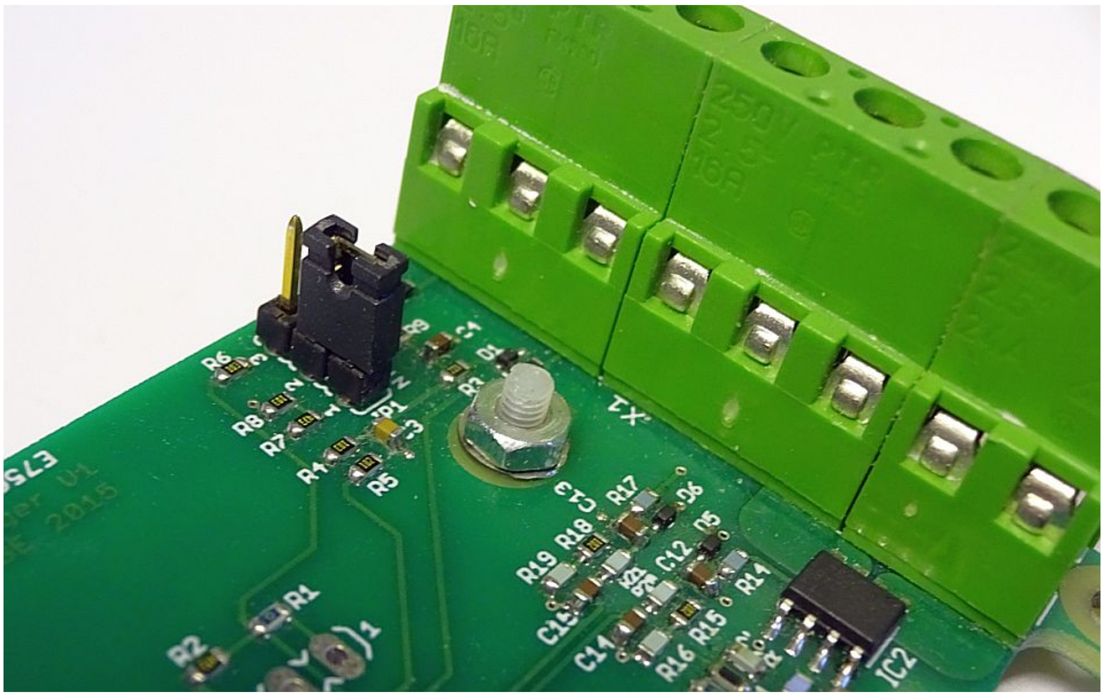
For Raspberry's with 40 pole pin header all 3 screw holes shall be used. Although there is no matching hole on the Raspberry Pi for the stand off in the middle, normally there are no components on the top of the PCB, which could be damaged. So there is a free space where the third nylon stand off could land on. (*Please double check with the actual sample of Raspberry Pi you are using to be safe !)*

Now the included stand off's have to be mounted to the screw holes of the PiLogger One. For that the threaded end of each stand off is plugged to the hole of the PiLogger module from the bottom side:

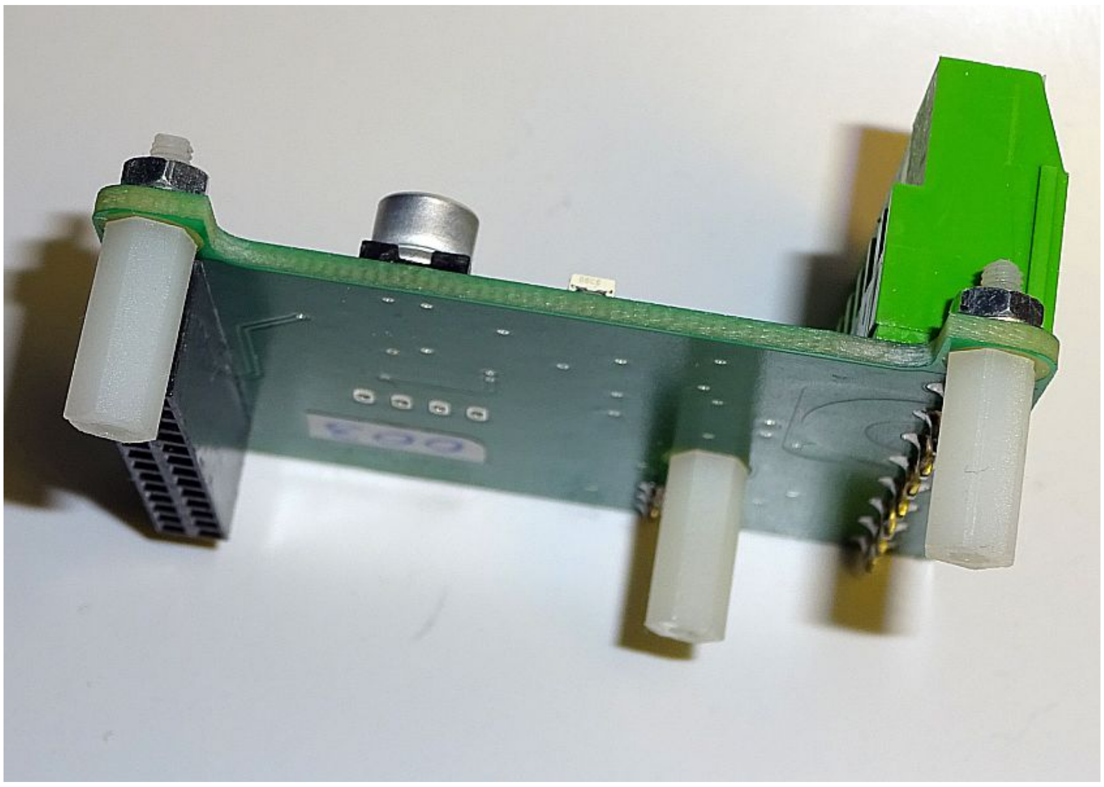


After that the bolt has to be held in the hole, while turning around the module. Now a washer has to be placed on the threaded end, followed by a nut. The nut has to turned to fix the bolt slightly in that way the bolt is not shaking anymore, but can be shifted to correct the position.

The hole in the middle on the top should look like this:



This shows all 3 from the side:



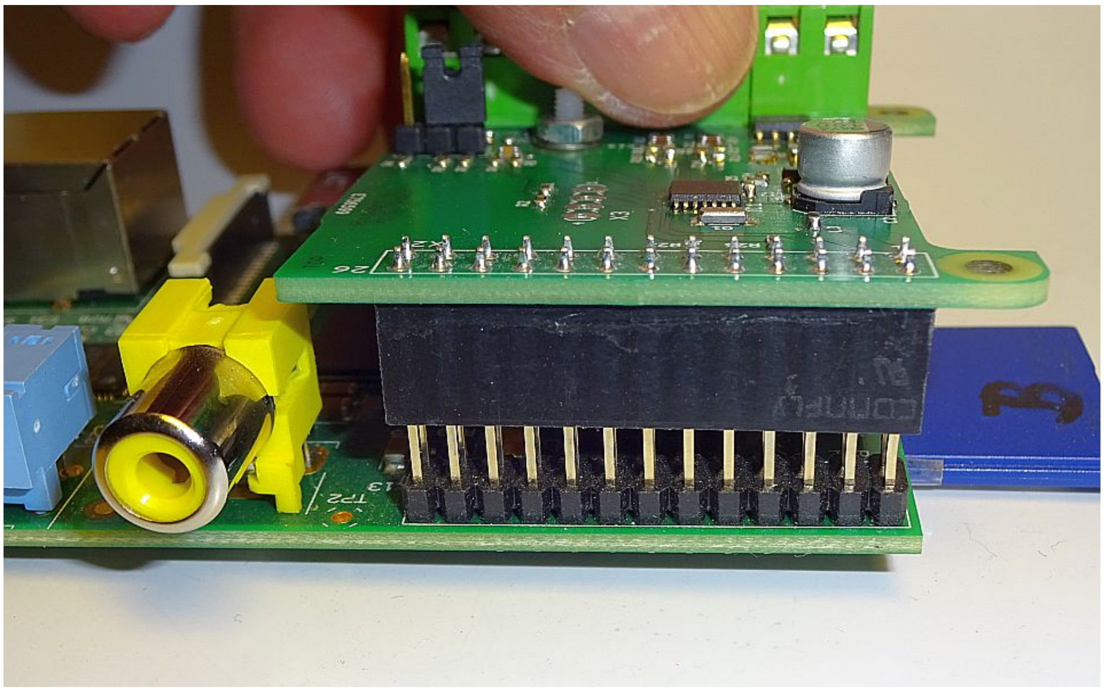
Now the module is ready to be attached.

5.2 Attaching the Module

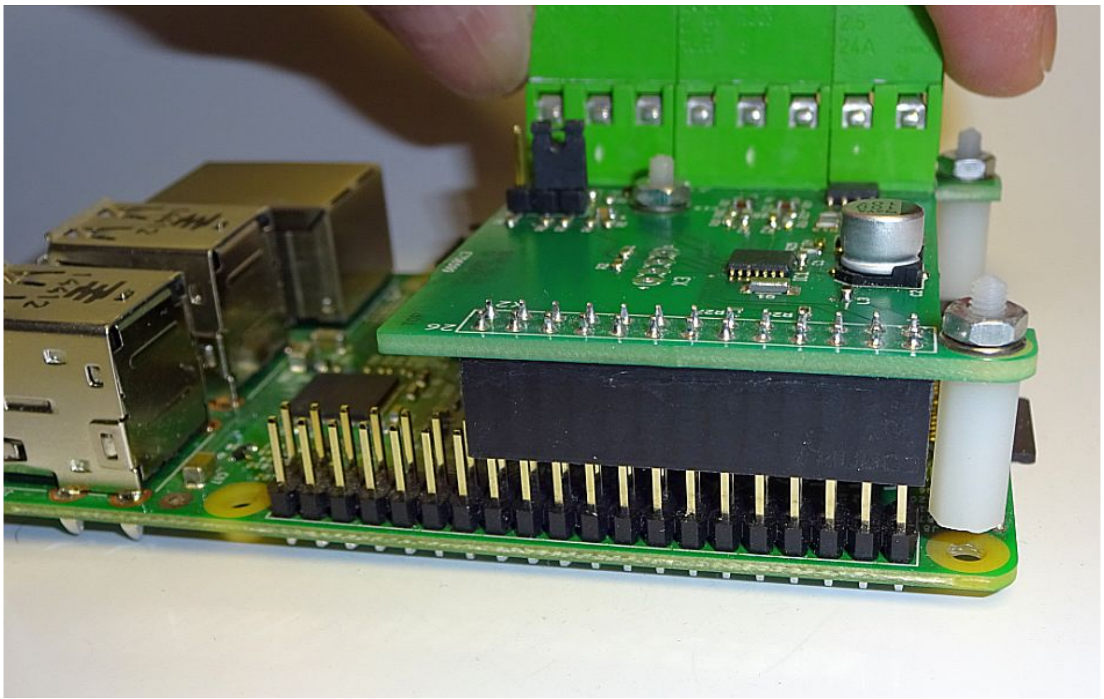
The PiLogger module is now placed on the Raspberry Pi. For that at least the power supply has to be unplugged – it is recommended to remove all cables and peripherals before continuing.

The 26 pole pin header socket has to be adjusted in that way, that the pins of the outermost corner of the Raspberry Pi match exactly with the sockets of the outermost corner with screw hole of the PiLogger (pin1 to socket1).

At first here shown at a Raspberry Pi 1:

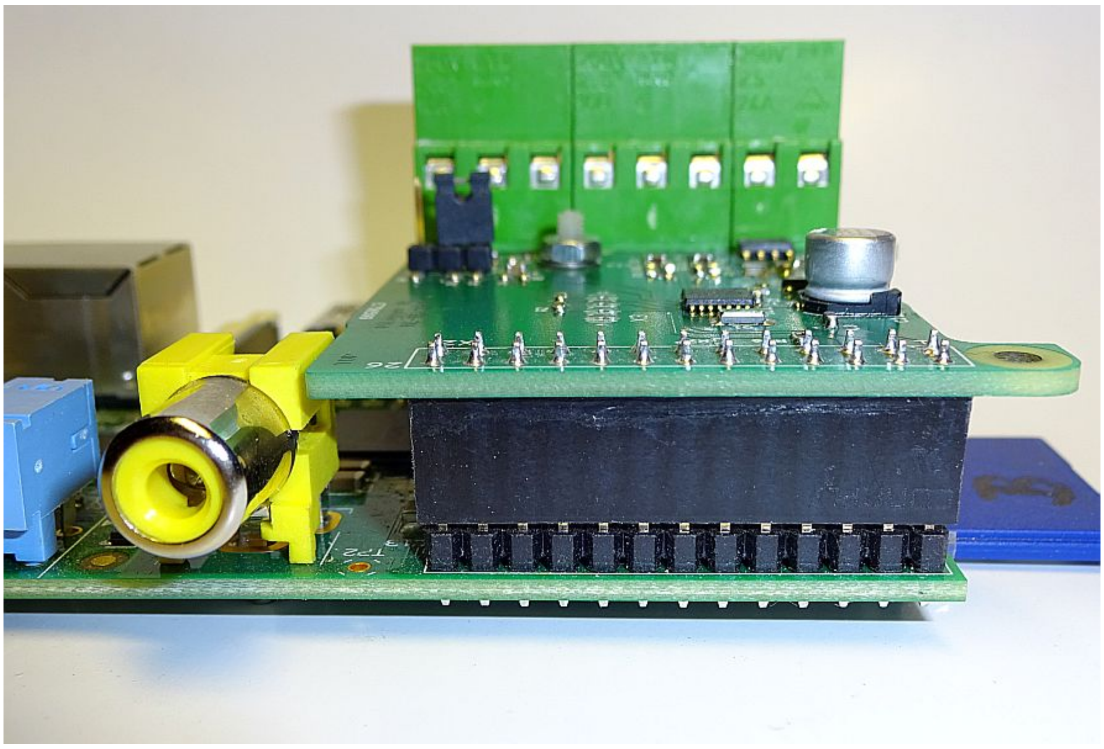


And here shown at a Raspberry Pi 2 B:

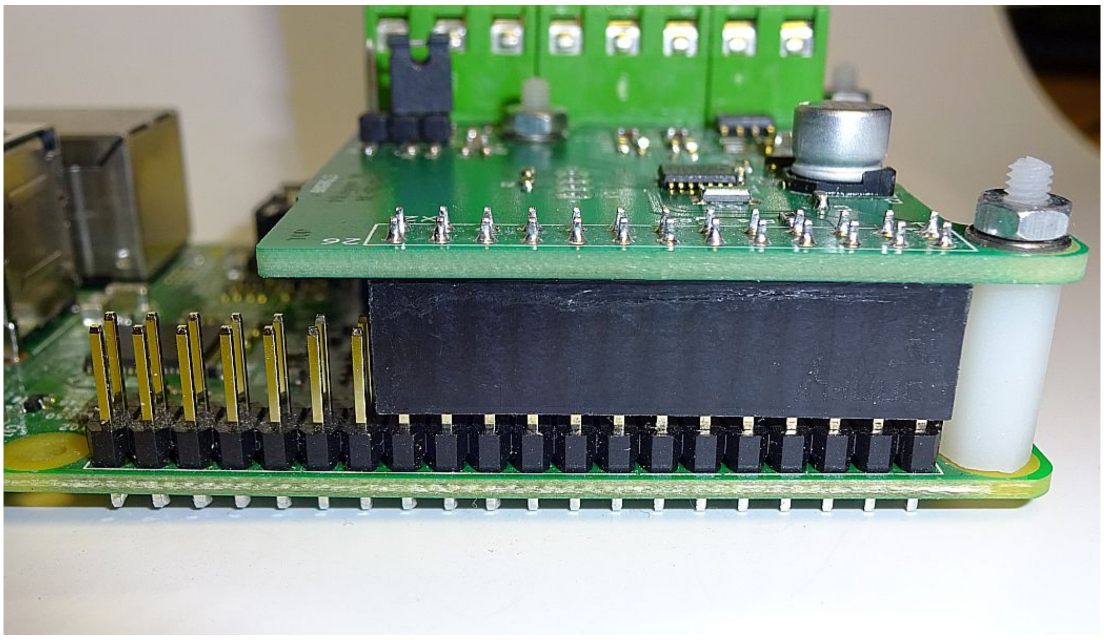


Now the PiLogger PCB has to be pressed down gently until the stand off's touch the Raspberry Pi PCB.

With a Raspberry Pi 1 this should look like this:

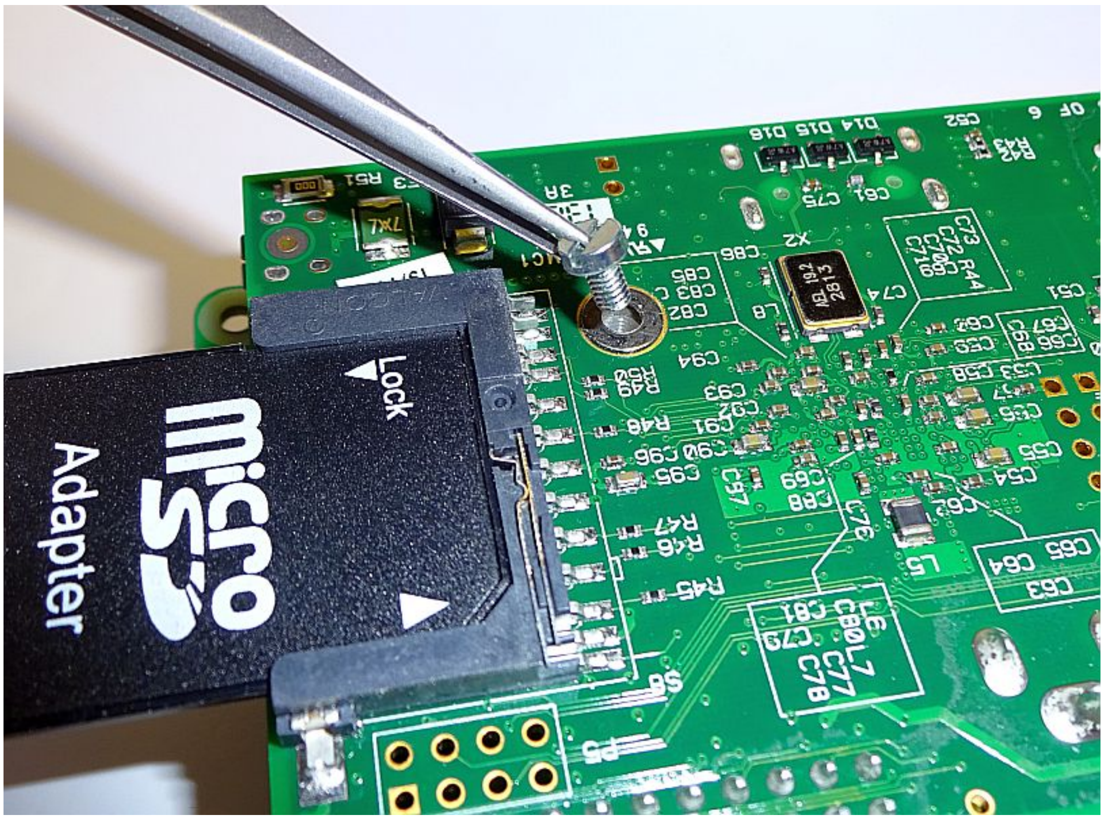


And with a Raspberry Pi 2 B it should look like this:

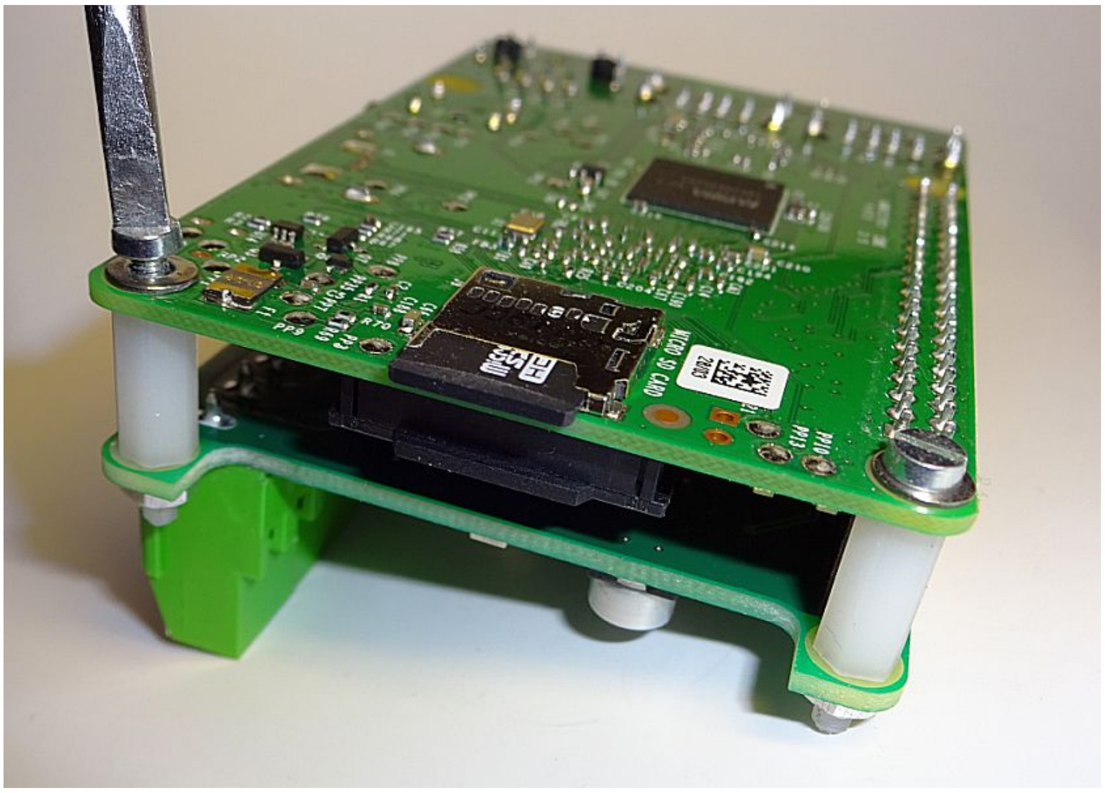


This means: The pin header socket actually does not hit the socket base of the pin header, but the contacts will have sufficient overlap.

Now we will turn the combination head down. On the bottom side of the Raspberry Pi we place a washer on every hole, where a stand off is behind.



Subsequently we turn in one screw to each stand off by hand and not too tight, so they fit accurately without any bending in case there are 2 of them. Subsequently we tighten them by using a screw driver just moderate:



The PiLogger module now is ready mounted.

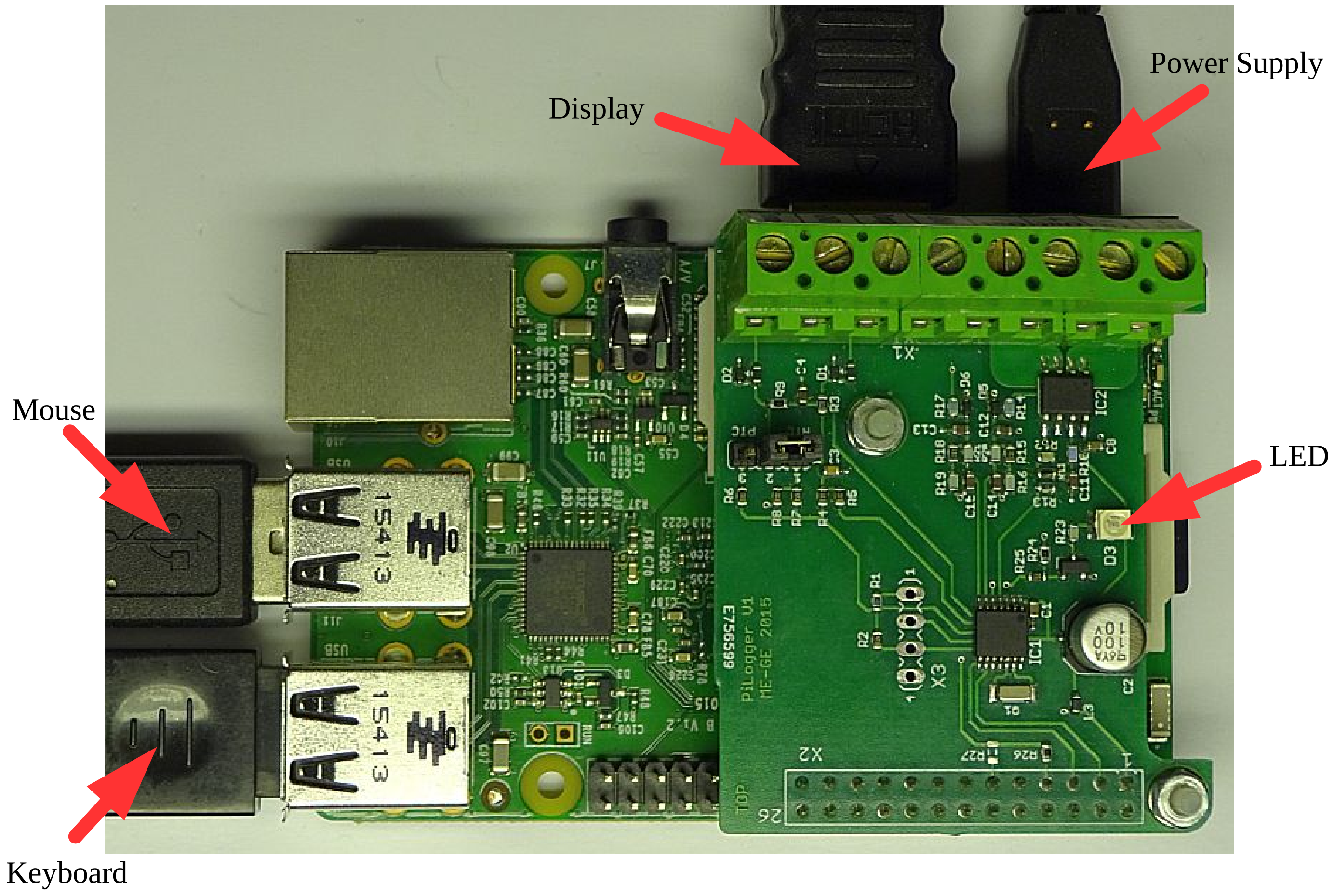
The pack of two now can be placed the normal way on the desk.

5.3 First Operation

Before powering up the Raspberry Pi with PiLogger the first time, of course we have to reconnect all cables and peripheral parts like in chapter 3.4.2 .

As the last part of that we put the power supply to the wall plug – and at the same time with the normal boot actions of the Raspberry Pi the PiLogger instantaneously begins to flash his LED once in a second.
This is according the factory settings and proofs it's working correctly!
Every time the Raspberry Pi is powered up, the PiLogger starts working autonomously with his stored parameters.

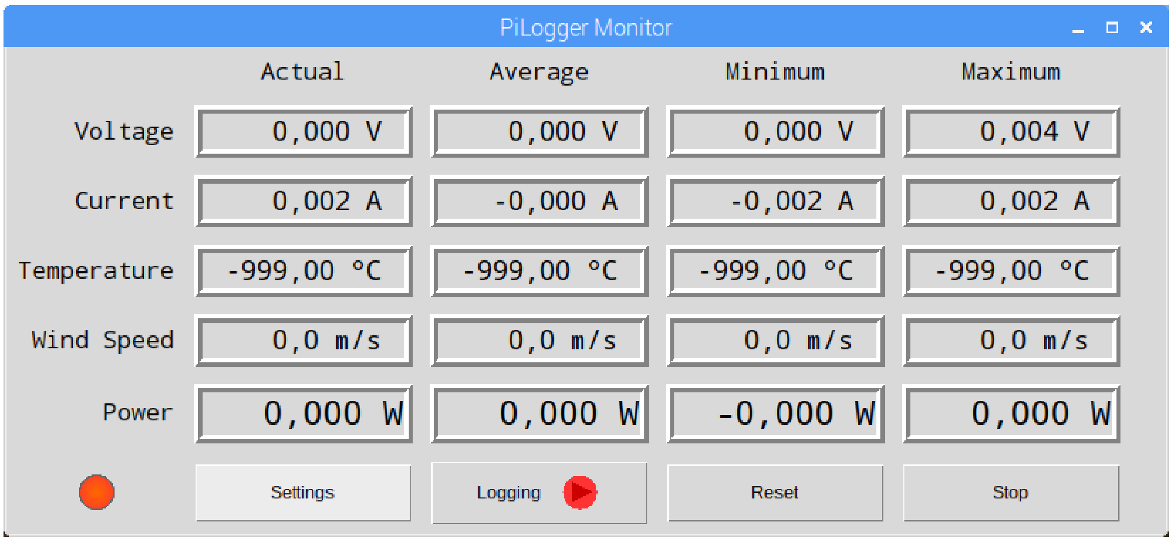
As an example here a PiLogger mounted to a Rasberry Pi 3 is shown:



6 Operation Instructions for the Base Software

6.1 Monitor Window

After the Rasberry Pi has booted completely, we start the PiLogger base software by double clicking on the PiLogger desktop icon. This will show up a window named 'PiLogger Monitor':

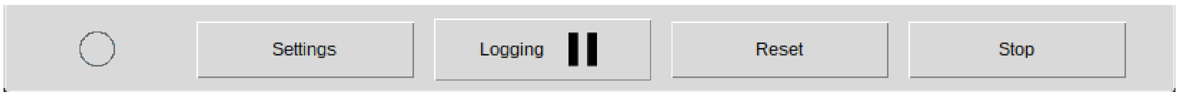


	Actual	Average	Minimum	Maximum
Voltage	0,000 V	0,000 V	0,000 V	0,004 V
Current	0,002 A	-0,000 A	-0,002 A	0,002 A
Temperature	-999,00 °C	-999,00 °C	-999,00 °C	-999,00 °C
Wind Speed	0,0 m/s	0,0 m/s	0,0 m/s	0,0 m/s
Power	0,000 W	0,000 W	-0,000 W	0,000 W

Within this window all measurement values are displayed live. *The label 'Actual' for the most left column is meant in the sense of the current value, not to be mixed up with the electrical current.* Because at this time nothing is connected to the PiLogger inputs, we see a near zero value for the voltage, a near zero value for the current and therefore a zero for the power too. Without an appropriate sensor connected also the value for the wind speed will remain at zero. The temperature is shown with a clearly invalid value, because without an appropriate temperature sensor the measurement value will stick to a range which does not make sense.
In the lower left corner there is a round indicator field, which was active at the time the screenshot was taken, indicating that the values shown at the moment are stored to the logfile at this time. With the preset values the actualisation of the displayed values will be done once per second and storing the measurement values will be done once in 10 seconds – these settings of course can be configured.

- **Button 'Settings'**
  On the right side of the log-indicator there is a button 'Settings', which will invoke a menu for all the configuration options like the previously mentioned. Details will follow in section 6.2 .

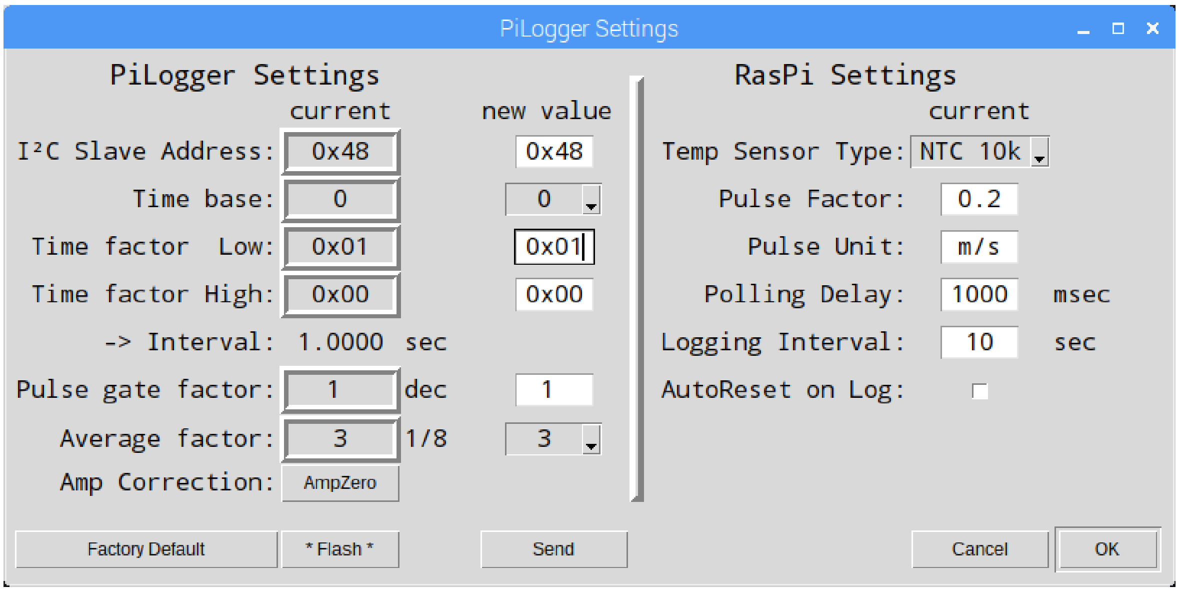
- **Button 'Logging'**
  More to right the button 'Logging' is the next one. It is provided with an integrated status indicator. This means: the icon within the button will change according the current status. A click on the button will pause the logging (storing to file) and a subsequently click will resume the logging. The paused status will look like this:



- **Button 'Reset'**
  The next button labeled 'Reset' is meant to set the statistic values 'Average','Minimum' and 'Maximum' back to the current measurement values. This reset actually is performed at the PiLogger, where these values are calculated. Therefore the reaction of the display is delayed, as it will happen with the next readout of the values from the PiLogger.
- **Button 'Stop'**
  The last button labeled 'Stop' is actually the button to exit the program. Clicking on it will terminate the Python program and close the window – but the PiLogger will continue to work autonomously.

6.2 Configuration Window

By the button 'Settings' an additional window named 'PiLogger Settings' is invoked, which will be laid over the monitor window:


PiLogger Settings

PiLogger Settings

	current		new value
I²C Slave Address:	0x48		0x48
Time base:	0		0
Time factor Low:	0x01		0x01
Time factor High:	0x00		0x00
-> Interval:	1.0000	sec	
Pulse gate factor:	1	dec	1
Average factor:	3	1/8	3
Amp Correction:	AmpZero		

RasPi Settings

	current	
Temp Sensor Type:	NTC 10k	
Pulse Factor:	0.2	
Pulse Unit:	m/s	
Polling Delay:	1000	msec
Logging Interval:	10	sec
AutoReset on Log:	☐	

This window is basically divided in two areas. The left half refers to settings related to the PiLogger. The right half refers to settings of the Python program itself, so referring to the Raspberry Pi side.

6.2.1 PiLogger Settings

The framed values of the left column 'current' are automatically read once from the PiLogger as soon as the window is invoked.
New values for each of the parameters can be given to the fields of the right column 'new Value'. Here are the details for each item:

- **I²C Slave Address**
  This parameter has to be given as hexadecimal value. It is the bus address on which the PiLogger will respond. This address is valid in the range 0x04 to 0x7C.
  The default address is set to 0x48 (72 dec). Only in rare cases there might be a reason to change this address – for instance if two PiLogger shall be operated at the same Rasberry Pi.

  > *Attention :*
  > *When this address is changed, immideatly the PiLogger base software is not able anymore to talk to the PiLogger !*
  > *Then you have to change the configuration file 'PiLogger_Config.txt' manually to the new address !*
  > *The new address will be activated immideatly temporally, if the button 'Send' is been clicked.*
  > *It only becomes permanently changed, if the button 'Flash' is used after a reboot of the base software with a changed configuration file.*

- **Time base**
  The PiLogger operates with an internal time base to derive the time interval between two measurements. Here you can choose from 4 base units to use:
  0 : 1 sec
  1 : 250 msec
  2 : 15,625 msec
  3 : 1,953125 msec
  This will define the total range for the settable time interval and the resolution for it. Together with the two following parameters a multiple of this time base unit can get set as measurement interval in the range from 1x … 65535x .
  The minimal measurement interval should be longer than 99 msec, as one measurement takes around 84 msec. This leads to 51 x 1,953125 msec = 99,609375 msec. As maximal measurement interval you can set 65535 sec, which means 18 hours, 12 minutes und 15 seconds.
  The base software will calculate the resulting measurement interval in seconds at every change from the currently set values. Therefore you can test a new measurement interval live by changing the values and sending them to the PiLogger. With that you get a display of the time in seconds.
  The LED of the PiLogger is flashing once with each measurement (around 84 msec on).

- **Time factor Low**
  This is the lower significant byte for the measurement interval factor.

- **Time factor High**
  This is the higher significant byte for the measurement interval factor.

The total factor calculates to H x 256 + L. Minimal value is 1.

- **Pulse gate factor**
  Here you can set a multiplication factor for the measurement time with the pulse count input. Especially with short measurement intervals there would be a lot of zero counts, which do not make sense when for instance representing wind speed. If, for instance, the measurement interval for voltage, current and power is set to ¼ second (250 msec), applying a factor of 40 will lead to an effective measurement interval of 10 seconds for the wind speed.

- **Average factor**
  The PiLogger continuously calculates a moving average for each of the 5 retrievable measurement results. The applicable weighting factor for this calculation can be chosen here. The valid range for this parameter M is 1 to 7, resulting to an averaging factor of 1/(2^M) . So the averaging factor can be set from ½ to 1/128. The current measurement value will be weighted by this factor and added to the old average value.

- **Button 'AmpZero'**
  The button 'AmpZero' in the line 'Amp Correction' will send a command to the PiLogger, causing it to set the current average value to be the zero reference.

  Attention : This command should only be applied if there really is no current flow through the sensor – for instance nothing is connected to the respective terminals.

  By this device to device deviations and other tolerances of the current measurement loop are calibrated. Before performing this action the PiLogger should be operated around 10 minutes to be thermally balanced.
  To perform this calibration is reasonable once at the time of first operation and may be later if there are significant deviations noticeable caused by slow changes.

- **Button 'Factory Defaults'**
  In case you get lost with, for instance, by setting extreme measurement intervals, this button will send a command to the PiLogger to revert to the default values.
  Of course this command will only be operable if there is still a valid I²C connection.

- **Button 'Send'**
  The values of the column 'new Value' will be send to the PiLogger when you click on the button 'Send'. They are activated immediately. This can nicely be watched by the activity in the monitor window in the background. And of course by the flashing of the PiLogger LED.

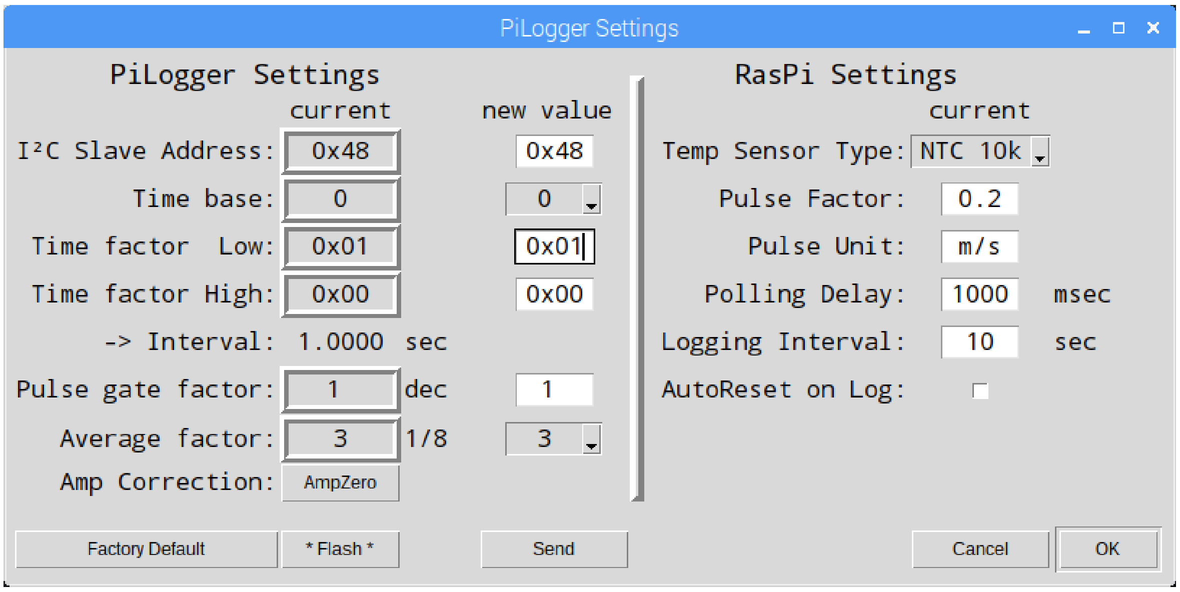
- **Button 'Flash'**
  If you want to make changed configuration parameters permanently, which means the PiLogger will also use these settings after a power cycle, these parameters have to be stored to the internal flash memory of the PiLogger. For that you only have to click on the

button 'Flash' once.

Attention: *For this action there is no direct feedback at all !*
Success can only be checked by performing a power cycle.

6.2.2 RasPi Settings

The right side of the window contains the settings for the Python program itself. So it is the Rasberry Pi settings part.



PiLogger Settings

PiLogger Settings	current		new value
I²C Slave Address:	0x48		0x48
Time base:	0		0
Time factor Low:	0x01		0x01
Time factor High:	0x00		0x00
-> Interval:	1.0000	sec	
Pulse gate factor:	1	dec	1
Average factor:	3	1/8	3
Amp Correction:	AmpZero		

RasPi Settings	current	
Temp Sensor Type:	NTC 10k	
Pulse Factor:	0.2	
Pulse Unit:	m/s	
Polling Delay:	1000	msec
Logging Interval:	10	sec
AutoReset on Log:	☐	

These settings are stored into a separate external file, called 'PiLogger_Config.txt', also located in the PiLogger folder.
Some of the PiLogger settings are also stored to this file, for instance the I²C address, so the program is able to talk the PiLogger, even if this address had been changed.

The RasPi settings in detail:

- **Pull Down List 'Temp Sensor Type'**
  With this pull down list you can select from 2 options: 'NTC 10k' and 'PT1000'. Depending on the actual connected temperature sensor the appropriate method to calculate the temperature from the measurement value is defined with this selection. These two algorithms are provided by the base software.

Attention:
The jumper at the PiLogger has to be set to the appropriate base type NTC or PTC to adapt the input circuitry.

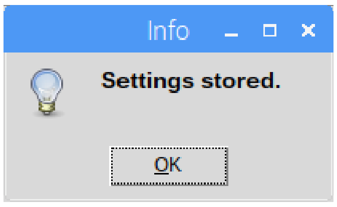
For NTC sensor type a 10 kOhm at 25°C is recommended and for PTC type a platinum sensor with 1000 Ohm at 0°C is recommended.

- **Pulse Factor**
  Here you can set the conversion factor for the impulse counting input.
  This is the characteristic sensor constant. For instance with a wind speed sensor this would be the distance per pulse. The actual measurement time interval will be considered by the PiLogger base software. With the shown example this is set to 0,2 m / Pulse – but of course for the complete calculation also the unit parameter from the next field has to be considered.
- **Pulse Unit**
  This is the associated unit of measure for the pulse count values.
  For instance here can be given 'km/h' instead of 'm/s', but of course then the sensor constant has to be adapted – with the given example this leads to 0,72 for 'km/h' instead of 0,2 for 'm/s'.
- **Polling Delay**
  With this parameter you can define how frequently the base software shall poll the PiLogger.
  This mainly is related to the live display of the measurement results, but has to be seen in conjunction with the following parameter 'Logging Interval'.
  If the live monitoring is not the main intention of the planned application, here also a very long time span can be set, which matches with the desired logging interval.
- **Logging Interval**
  Here the time between 2 logfile entries is set.
  It is reasonable the set this interval to an integer multiple of the polling interval – because the application is designed that way ;-)
  This interval represents the time distance between the logged measurement results.
  The measurement results are stored together with a timestamp to the file 'logdata.csv' in the folder '/home/pi/PiLogger'.
- **AutoReset on Log**
  This field as a checkbox. If the box is left empty, this function is not active. If this box is checked, so filled with a hook, this function is active.
  If this function is activated, every time a line is added to the logfile, also a command is send to the PiLogger to reset the statistical values 'Average', 'Minimum' and 'Maximum' to the current measurement value.
  By that the logged values represent exactly the statistical values for their time interval.
  This is for instance reasonable for the case that also additionally the energy amount shall be counted, that is power times time interval.

■ **Button 'Cancel'**
  The button 'Cancel' closes the window 'PiLogger Settings' without storing the current configuration.

- **Button 'OK'**

  The button 'OK' closes the window 'PiLogger Settings' and stores the current settings to the file 'PiLogger_Config.txt' in the folder '/home/pi/PiLogger'.
  This feedback message will appear:



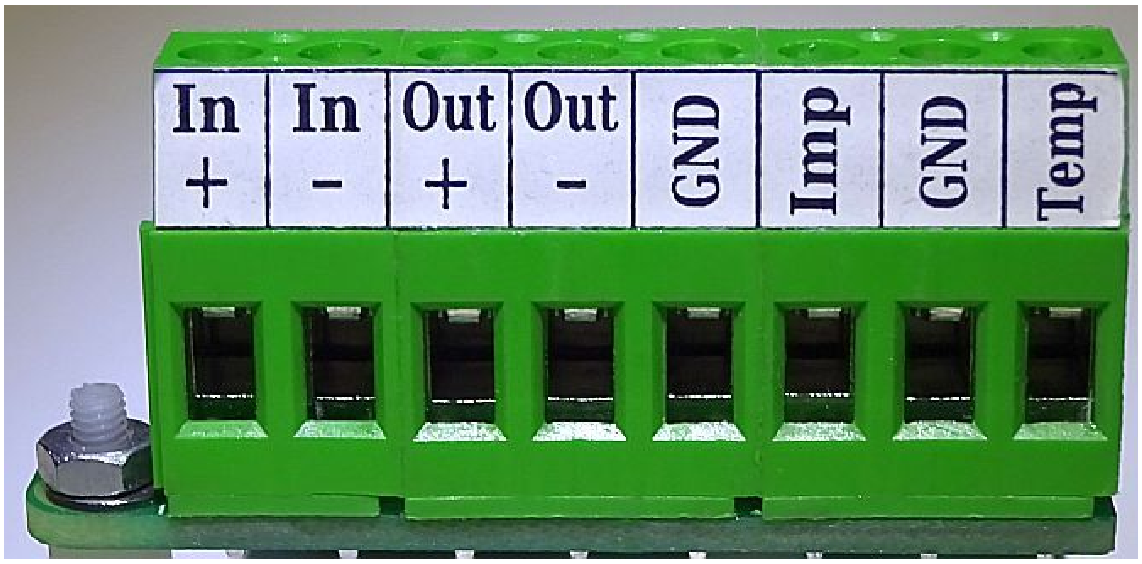
Settings stored.

This message disappears after a click on 'Ok'.

Now all important things about the operation of the base software are told.
It’s time now to produce real measurement results. For that it’s necessary to connect the terminal block to sensors and other external circuitry.

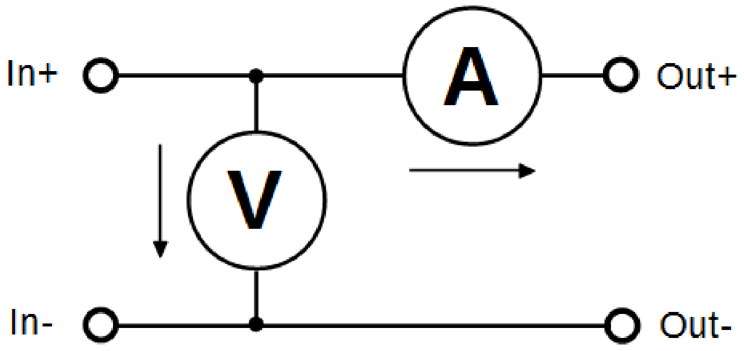
7 Terminal Block

The measurement inputs of the PiLogger all are located at the 8 pole terminal block:



7.1 DC Voltage and Current Measurement

Starting on the left, the first 4 terminals are assigned for measuring voltage and current of one DC path. They are connected that way, the first two terminals are meant as input and the next two terminals are meant as output, forming a loop through. The two terminals labeled with '-' are directly connected to each other, while the two terminals labeled with a '+' are connected to each other via a current sensor:



Although the ohmic resistance of the current sensor is only around 2 mΩ, and therefore really low, however at a current of 12 A it will cause a drop of 24 mV, which corresponds to a power loss of 288 mW.
So this is a systematic measurement error of this setup.
With low voltages and high currents this error maximises. But with the above example at 12 A and 3 V it is still only 0,008 so less than 1%.

The fact, that one must use a sufficient wire gauge when dealing with currents as high as 12 A becomes understandable with this calculation of power loss at those apparently low

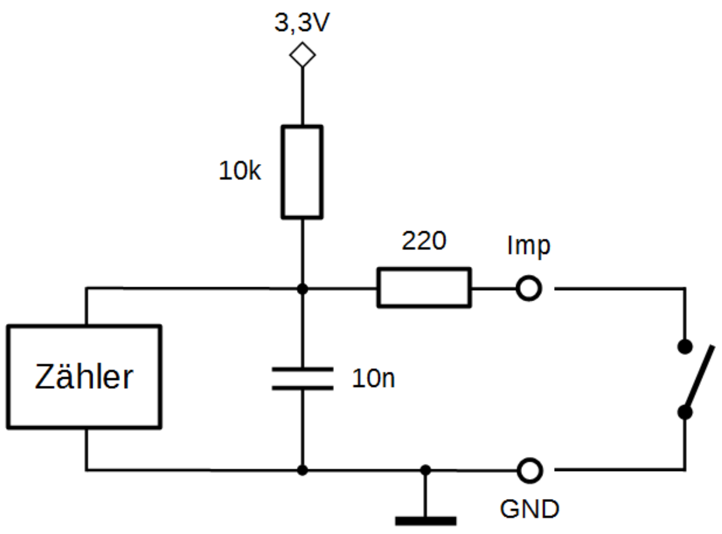
resistances.
Although a single wire cable with 0,75 mm² (90°C) according VDE may be charged up to 12 A, this loading capacity is reduced to 35% when the ambient temperature is 85°C !
In case the setup is heated up, for instance by sun radiation, much thicker cables are needed – in this case 4 mm² cross-sectional area – to achieve a safe setup.
The terminals of the PiLogger are able to deal with cross-sectional areas of up to 4 mm² (stranded wire with end sleeve).

A cable providing 4 mm² cross-sectional area of copper and having a length of 4 m, will have a resistance of around 17,2 mΩ and therefore will cause a power loss of nearly 2,5 W at a current flow of 12 A ! And this happens in the return conductor once again, so the power loss sums up to nearly 5 W !

So please also keep the cable length as short as possible.

7.2 Pulse Counter Input

The next two terminals are labeled with 'GND' and 'Imp'. These are the input connectors for the pulse counter.
This input is designed for the use with mechanical switches, like Reed contacts, as well with open collector outputs (OC). In open state the terminal 'Imp' will show around 3,1 V with respect to the reference ground at the terminal 'GND'. When the terminal 'Imp' is shorted to 'GND', this low going edge will be count as one pulse.
The main elements of this input show up this way:

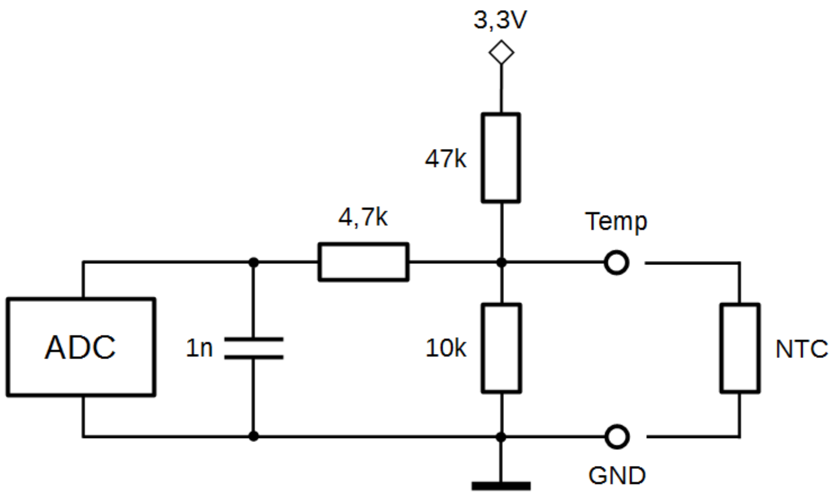


The capacitor in conjunction with the pull up resistor will have a time constant for the recovery of the input for the next negative edge. The series resistor together with the capacitor forms a low pass filter. This provides debouncing and reduced sensitivity against noise.
At a duty cycle of 50% the maximum input frequency is about 2 kHz. The resistance of the closed switch or open collector must be below 3 kΩ to enable safe low detection.

7.3 Temperature Sensor Input

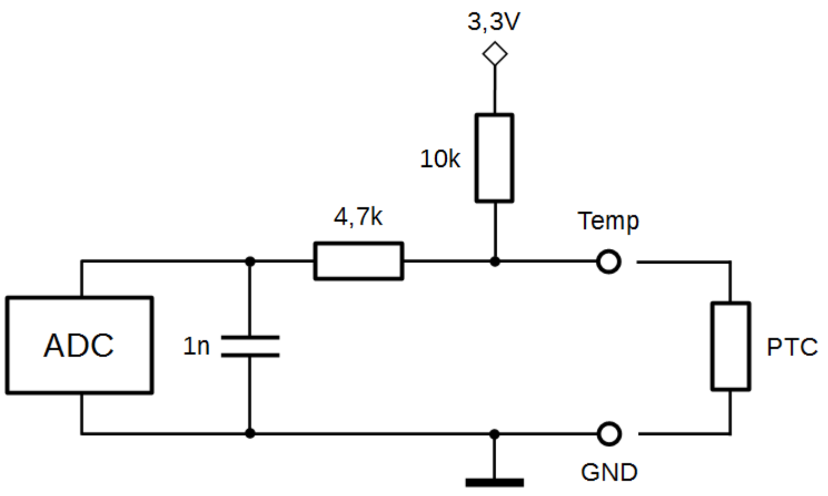
The two terminals at the right are labeled with 'GND' and 'Temp'. They are designed to connect a temperature sensor.
By the jumper JP1 this input can be adapted to work with 2 types of temperature sensors:

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



Set to this position the circuitry of this input is configured for a NTC (Resistor with **N**egative **T**emperature **C**oefficient) with a nominal value of 10 kΩ.

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

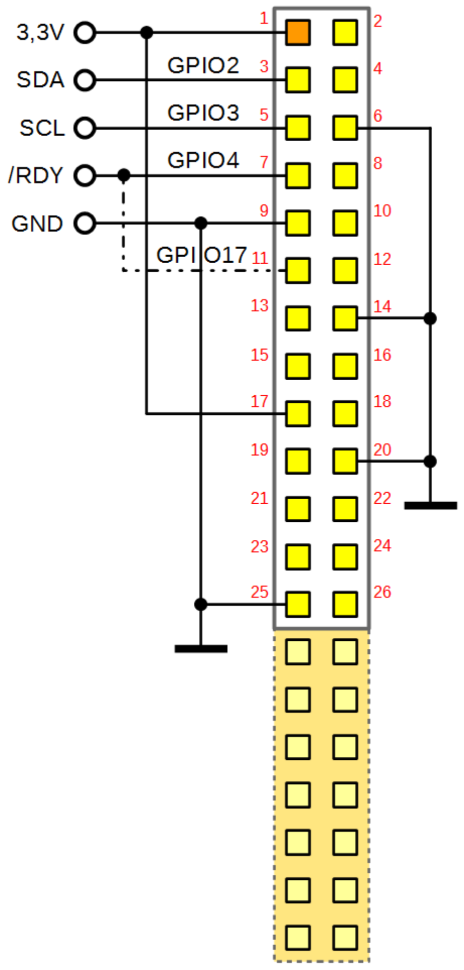


Set to this position the circuitry of this input is configured for a PTC (Resistor with **P**ositive **T**emperature **C**oefficient) with a nominal value of 1 kΩ.

More details on temperature sensor are given in chapter 10.

8 Pin Assignment GPIO Connector

The PiLogger only uses 5 pins of the first 8 pins at the Raspberry Pi GPIO header.
There a few more pins connected – but this is just for increasing the contact reliability (3,3V & GND) and as an option (/RDY).



By the signal '/RDY' (= busy) the PiLogger One informs the Rasberry Pi about his current measurement activity. The state of the signal corresponds to the state of the LED at the PiLogger. A 'High' (+3,3V) means, a measurement is currently performed – the LED is on. When the signal changes to 'Low', the measurement is done – the LED goes off.
This negative going edge (falling edge) is connected to the GPIO4 (pin 7) of the Rasberry Pi, which can be configured to raise an interrupt to retrieve the fresh measurement values with the correct timestamp and to store it to the logfile.

9 Application Program Interface

The PiLogger One utilizes the I²C interface to communicate with the Rasberry Pi.
At this connection the Rasberry Pi is master and the PiLogger is slave.
The I²C bus (resp. SMBus) on this uses the hardware protocol defined by the official specification (http://www.nxp.com/documents/user_manual/UM10204.pdf).
The SMBus actually is an evolvement of the I²C bus with some extensions, which are not or only partly supported by pure I²C devices (ICs).
This protocol is handled within Linux by the kernel driver 'i2c-dev' and is made accessible to the so called 'user land' – the area of applications – by the module 'i2c-dev'.
Via the wrapper 'python3-smbus' as next step Python is enabled to get access to the I²C bus via the kernel driver. Alternatively the 'I2C-Tools' can be used from the command line interface (see appendix).
The last two provide a set of functions (methods) by which all necessary operations on the I²C Bus, respectively SM bus can be performed.
In the following the message catalogue will be described, which defines the communication with the PiLogger One.
Because the I²C bus resp. SM bus is byte oriented, all accesses are controlled by virtual one-byte-addresses – the registers.
The PiLogger One knows two classes of registers:
Configuration registers and measurement result registers.
Additionally there are direct commands, which are triggered by writing to the respective register (with and without password).

9.1 Configuration Registers

The following table lists all configuration registers of the PiLogger One.
This registers are in the range 0x01 to 0x16.
Their particular meaning was described already in chapter 6.2.1 .

The registers from 0x01 to 0x06 are valid addresses for a writing access.
The corresponding registers from 0x11 to 0x16 are the respective addresses for a read access.

These registers hold 1 byte each. This means, that with a write exactly 1 data byte is expected and with a read exactly 1 data byte is returned.

Additionally there are 3 commands (0x07 , 0x0D , 0x0F), which are triggered by writing to this register, if the send value (data byte) is the expected fix value – the password. This is implemented this way to lower the risk of triggering by accident, for instance, due to disturbed communication.

R/W	Register	1 Byte		
Configuration	**Write**	**Read**	**Value**	
I²C Slave Adress	0x01	0x11	default 0x48	0x04…0x7C
Measurement Cycle Time Base	0x02	0x12	0 = 32768 = 1 sec 1 = 8192 = 250 ms 2 = 512 = 15,625 ms 3 = 64 = 1,953125 ms	
Cycle Time Factor 1	0x03	0x13	1 – 255 low	1…65535
Cycle Time Factor 2	0x04	0x14	0 – 255 high	
Averaging Factor	0x05	0x15	1 – 7	2^x : 2…128
Time Factor Pulse	0x06	0x16	1 – 255	
Set Amp Avg to Zero	0x07	-	0x44	Write only
Set Factory Default	0x0D	-	0xAB	Write only
Store Config (flash)	0x0F	-	0x55	Write only

The usage within Python will look as follows:

The first instruction is located right at the beginning of the program and will include the code from the module 'SMBus'. The second instruction sets the variable 'pilogger' to the value of the bus to be used.

```
import smbus
pilogger = smbus.SMBus(1)      ; for Raspberry Pi type A (256 Mbyte): smbus.SMBus(0)
```

Read one 1-byte-register:

```
value = pilogger.read_byte_data(address, register)
```

Example:

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

Write one 1-byte-register:

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

Example:

```
pilogger.write_byte_data(0x48, 0x12, 0x01)
```

9.2 Result Registers

The following table lists all result registers of the PiLogger One.
These registers are in the range 0x20 to 0x88.
The registers in the range 0x20 to 0x63 without [0x28,0x38,0x48,0x58] are valid addresses for a read access.
These registers hold 2 bytes each. This means, that a read from this address will return exactly 2 data bytes. These 2 bytes mean one word each – representing a 16 bit number.

The measurement values from the registers in the range 0x20 to 0x27 (power) have to be combined to 4 byte values each, as they represent 32 bit numbers.

The registers 0x70 and 0x80 will trigger a block read operation.
These 2 blocks do overlap, which will lead to a double read of the results for temperature and pulse count – despite of that, this retrieving method is the fastest and most effective way, if all values are needed.

Additionally there are 7 commands (0x28...0x88), which are triggered by writing the respective register. Writing to these registers will reset the corresponding statistical values (to the current value).

Read only	**Register**	**2 Byte**			Avg, Min & Max
Measurement Result	Actual	Average	Minimum	Maximum	**Reset**
Power	0x20 0x21	0x22 0x23	0x24 0x25	0x26 0x27	0x28
Temperature	0x30	0x31	0x32	0x33	0x38
Pulsecount	0x40	0x41	0x42	0x43	0x48
Voltage	0x50	0x51	0x52	0x53	0x58
Current	0x60	0x61	0x62	0x63	0x68

Block Read	**Command**	**Register**			**Reset**
Block1 Read 16x Word	0x70	0x20...0x43			0x78
Block2 Read 16x Word	0x80	0x30...0x63			0x88

The usage within Python will look as follows:

The first instruction is located right at the beginning of the program and will include the code from the module 'SMBus'. The second instruction sets the variable 'pilogger' to the value of the bus to be used.

```
import smbus
pilogger = smbus.SMBus(1)       ; for Raspberry Pi type A (256 Mbyte): smbus.SMBus(0)
```

Read one 2-byte-register:

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

Example:

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

Read one 4-byte-value:

```
valueLo = pilogger.read_word_data(address, reg)
valueHi = pilogger.read_word_data(address, reg+1)
value = valueHi * 65536 + valueLo
```

Example:

```
powerMaxLo = pilogger.read_word_data(0x48, 0x26)
```

```
powerMaxHi = pilogger.read_word_data(0x48, 0x27)
powerMax = powerMaxLo * 65536 + powerMaxHi
```

Read one 32-byte-block:

```
valarray = pilogger.read_i2c_block_data(address,blockreg)  ; block1=0x70, block2=0x80
```

Example:

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

Reset statistics of one block:

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

Example:

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

9.3 Value Ranges and Scale Factors

The PiLogger outputs the measurement results as raw values with different ranges and scales. The following table shows the ranges and according scale factors for each channel:

Channel	Format	Range	default scale factors			
Power	signed 32 bit	-2147483648... +2147483647	918/2147483648	W	-918...+918	1/2339236
Temperature	unsigned 16 bit	0...+65535	0,00915541	mV	0...600mV	1/109,2267
Pulsecount	unsigned 16 bit	0...+65535	0,2	m/s		
Voltage	unsigned 16 bit	0...+65535	0,000915541	V	0...60V	1/1092,267
Current	signed 16 bit	-32768... +32767	0,000466933	A	-15,3...+15,3A	1/2141,634

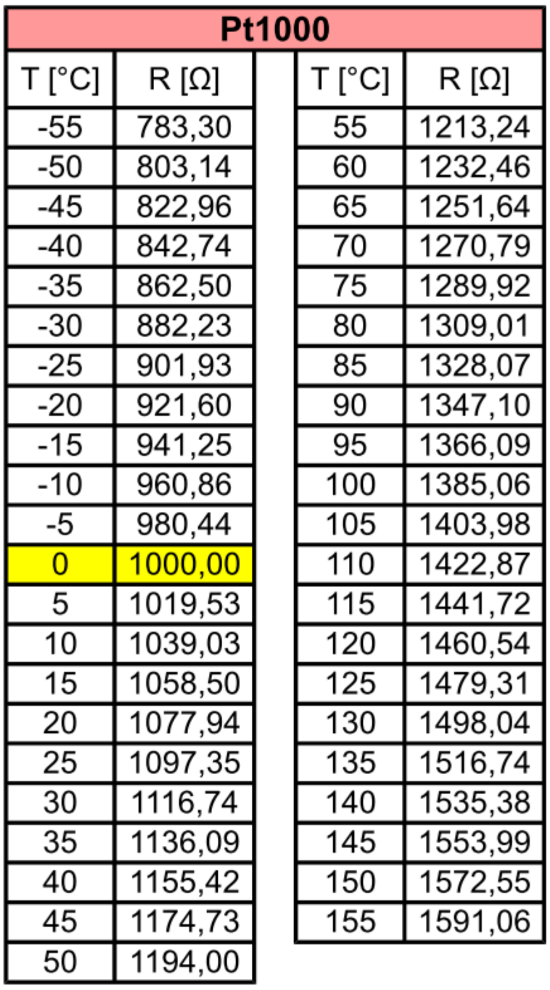
The given values are the respective full scale limits, the useable measurement ranges are in fact a little bit smaller (see 'Technical Data').

The pulse count channel is an exception to this. On this channel the raw value is always pulse per time interval – the interpretation depends on the used sensor (see 6.2.2 RasPi settings – Pulse factor). The default scale factor for this is just a setting within the base software, which of course can be adapted.

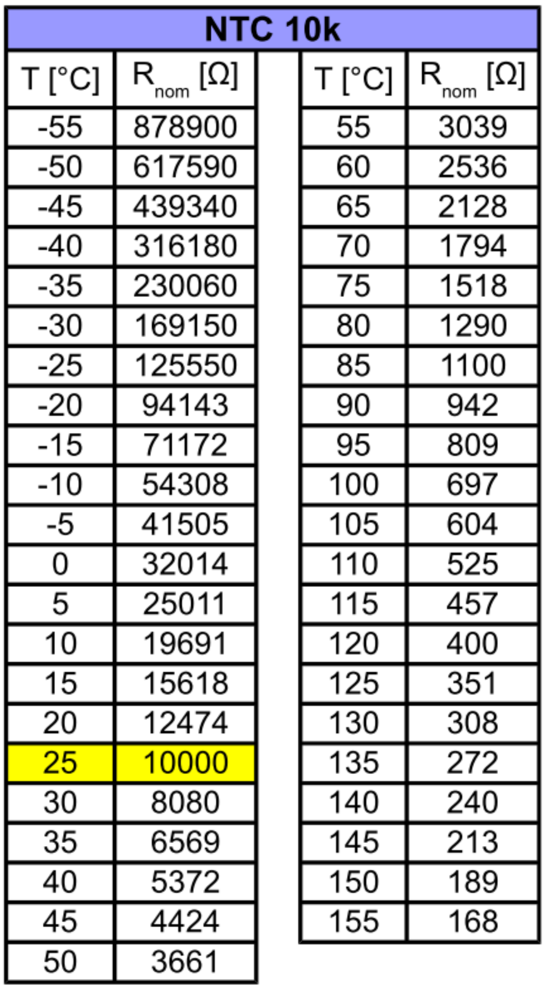
The temperature measurement for the PiLogger internally is a voltage measurement. The conversion calculation to derive the temperature values depends on the actually used temperature sensor. The conversion therefore is done in the base software.

10 Temperature Sensor Tables

The temperature sensor input of the PiLogger One is optimised for two widely spread types of temperature sensors.
One type is a so called Pt1000, a sensor based on platinum (Pt), which has a positive temperature coefficient (PTC) and which will show a typical resistance of 1000 Ω at 0°C, if of the variant Pt1000. This type is quite accurate and it is standardised. The other one is a so called NTC, so it’s a temperature depending resistor with a negative temperature coefficient, which unfortunately exists in many variants. With setting the PiLogger input jumper to 'NTC' the input is optimised for a common type in the heating installation industry, having a typical resistance of 10 kΩ at 25°C (e.g. Epcos B57891S0103H008).
A NTC typically spreads over a huge range of resistance, which makes it more easy to measure it – but on the other hand its characteristic is much more non linear, which increases the effort for approximation. Both types do have further pros and cons, which we won’t discuss here, but due to the high measurement resolution of the PiLogger (16 bit), both types enable very accurate temperature measurements.
The following table gives the resistance over temperature for both typical sensor types in the range of -55°C bis 155°C :



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



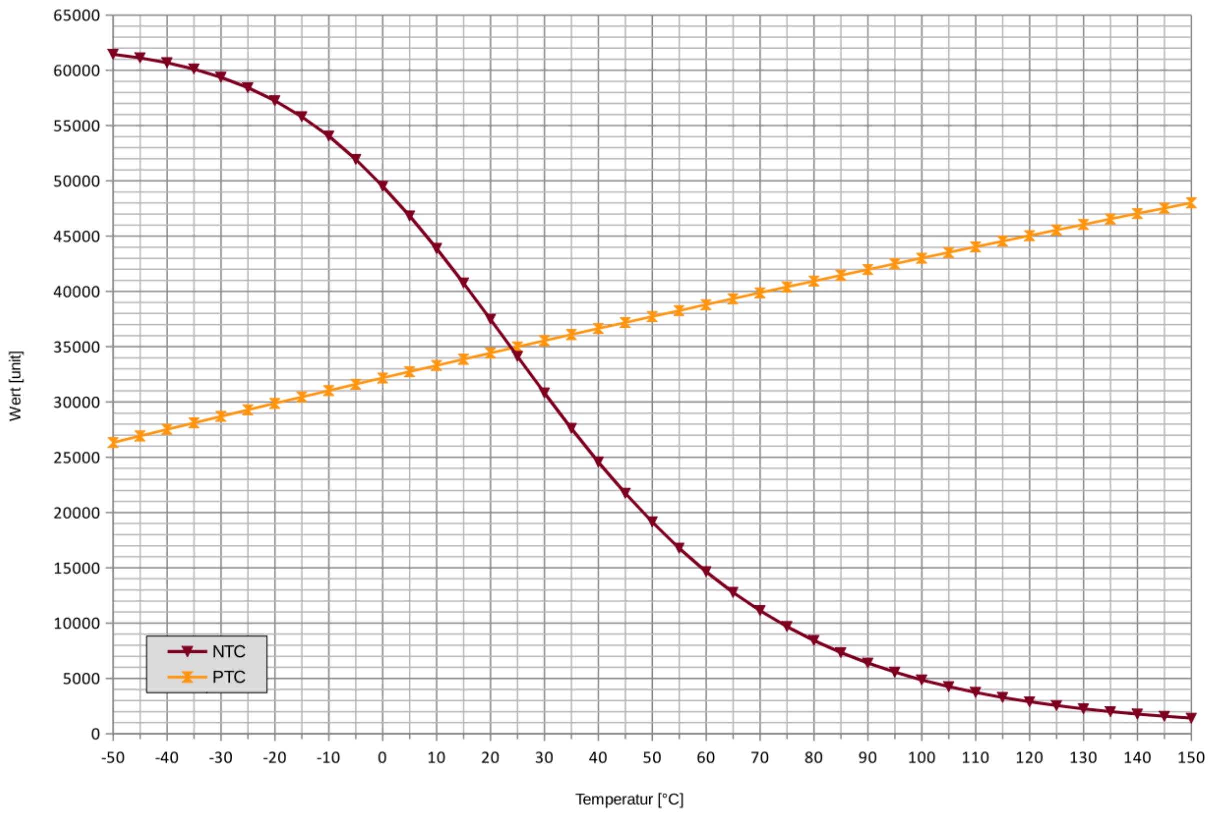
NTC 10k			
T [°C]	R_{nom} [Ω]	T [°C]	R_{nom} [Ω]
-55	878900	55	3039
-50	617590	60	2536
-45	439340	65	2128
-40	316180	70	1794
-35	230060	75	1518
-30	169150	80	1290
-25	125550	85	1100
-20	94143	90	942
-15	71172	95	809
-10	54308	100	697
-5	41505	105	604
0	32014	110	525
5	25011	115	457
10	19691	120	400
15	15618	125	351
20	12474	130	308
25	10000	135	272
30	8080	140	240
35	6569	145	213
40	5372	150	189
45	4424	155	168
50	3661		

With the input circuitry given in chapter 7.3 these tables for the values result:

Pt1000	
T[°C]	Wert
-55	25718
-50	26319
-45	26918
-40	27513
-35	28106
-30	28695
-25	29281
-20	29864
-15	30444
-10	31021
-5	31595
0	32167
5	32735
10	33300
15	33863
20	34422
25	34979
30	35533
35	36085
40	36633
45	37179
50	37722
55	38262
60	38800
65	39334
70	39866
75	40396
80	40923
85	41447
90	41968
95	42487
100	43002
105	43516
110	44026
115	44534
120	45039
125	45541
130	46040
135	46537
140	47031
145	47522
150	48010
155	48496

NTC 10k	
T[°C]	Wert
-55	61682
-50	61441
-45	61114
-40	60679
-35	60107
-30	59367
-25	58424
-20	57247
-15	55797
-10	54054
-5	51942
0	49509
5	46824
10	43884
15	40748
20	37484
25	34124
30	30815
35	27607
40	24561
45	21740
50	19144
55	16767
60	14645
65	12772
70	11126
75	9680
80	8423
85	7328
90	6382
95	5563
100	4854
105	4249
110	3726
115	3272
120	2878
125	2539
130	2245
135	1988
140	1763
145	1566
150	1395
155	1244

Given as a diagram this looks like this:



These tables can be used as look up tables by the application software. So, for a measurement value it gives the corresponding temperature, whereas values inbetween two given one, have to be calculated by linear interpolation.

The PiLogger base software does another approach. It calculates the temperature value for each measurement value by using an approximation formula. This formula tries to match the original curve as close as possible by simple math operations.

For the Pt1000 this formula is used:
Measurement value M in range 25000…50000:
$T = M^2 / 23006845{,}6 + M / 167{,}473355 – 237{,}0911$
(quadratic approximation)

For the NTC 10k this is not good enough. The curve characteristic does change that massively over the temperature ranges, that with only one formula the deviation from the original curve gets to big. Therefore the complete range from -40°C bis 140°C is parted in 3 sub ranges, using a different formula for each sub range:

With m = M / 65536 ;
Measurement value M in range 1763...8423 (+140...+80 °C) :
$T = m^3 * -50245{,}929 + m^2 * 16224{,}33 + m * -2068{,}25 + 184{,}196$

Measurement value M in range 8424…54054 (+80…-10 °C) :
$T = m^3 * -247{,}0087 + m^2 * 394{,}7776 + m * -305{,}093 + 111{,}6473$

Measurement value M in range 54055...60679 (-10…-40 °C) :
$T = m^3 * -18593{,}11 + m^2 * 47031{,}885 + m * -39846{,}685 + 11292{,}757$
(cubic approximation)

This is, especially for the NTC, a lot a calculation effort, but this is not a problem for the Rasberry Pi.
The precision of the temperature measurement by that already gets very good, but it still depends essentially on the tolerances of the used sensor. The deviation easily may be +/- 3° or even more. For more accurate absolute values a calibration must be performed. For instance a 2-point-calibration against a known good (calibrated) liquid thermometer (alcohol column).
Or the classical 2-point-calibration with an ice water sludge (0°C) and boiling water (100°C) – the definition of the Celsius scale. For that you will determine the correction value at 0°C, which has to be added to the raw value to achieve the displayed value is exactly 0°C ist (offset) and subsequently the linear correction value, which has to be multiplied to the intermediate value to match the display additionally at 100°C.
By that an accuracy of 0,1°C in a range of -10°C to +120°C should be achievable.

11 Technical Data

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

Weight : 29 g

Supply voltage : 3,3 V from Raspberry Pi via extension pin header

Supply current : 5 … 6 mA

Time for one measurement ~ 84 ms

Time for polling all values (2 blocks) : ca. 14,7 ms (@100 kbaud , BCM2708)

Range for interval time : 100 msec … 18h 12min 15sec

Measurement ranges:

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

Current : -12 … +12 A DC 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 used sensor (ref. Chap. 10), Resolution < 0,01 K

12 Calibration

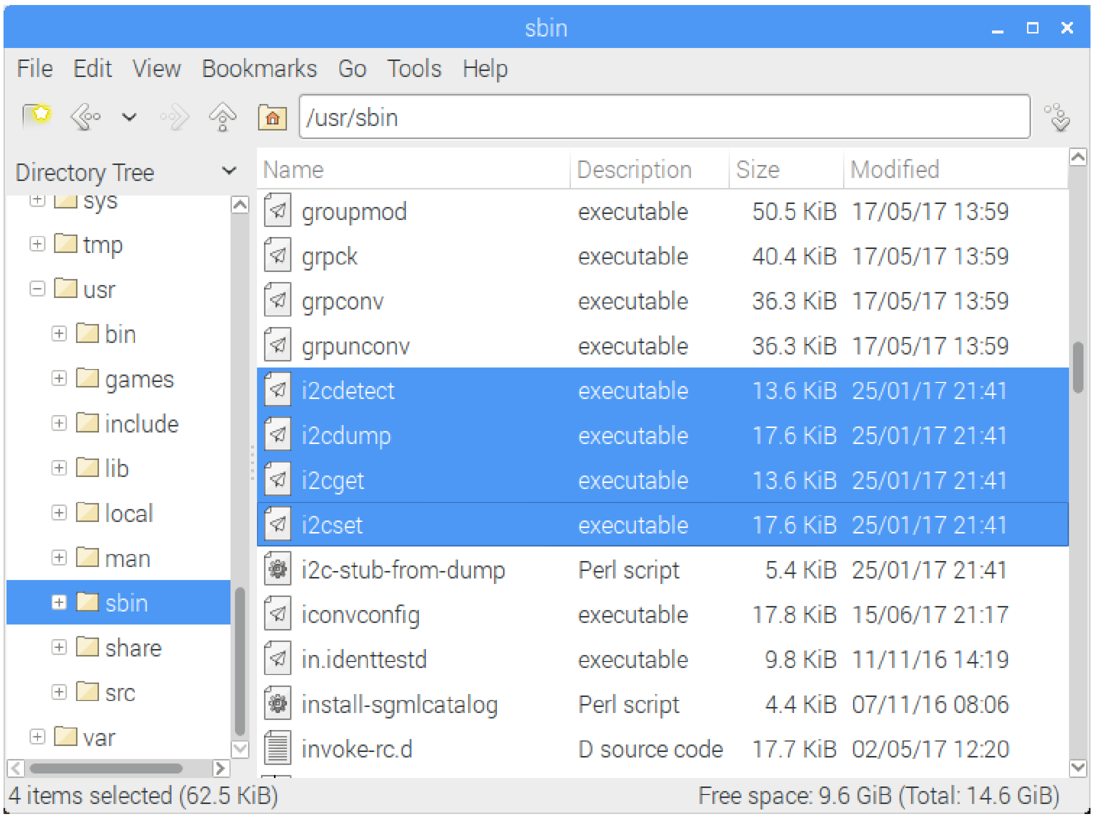
Although the PiLogger is built with low tolerance components and internally works with a 16 Bit A/D converter, it is *not* a calibrated measuring instrument.
It is not calibrated by factory against any official standard.
To perform precision measurement each sample should be calibrated using a suitable reference. The resulting correction values can easily be incorporated with the software running on the Rasberry Pi (for instance in the Python code of the Base Software).
By doing so, best precision of the values shown or stored to the logfile can be achieved.

13 Appendix

13.1 The I²C Tools

With the installation of the SMbus module automatically also the modules of the 'i2c-tools' are installed, which can be used directly from the command line.

The I²C tool set includes 4 programs: 'i2cdetect', 'i2cdump', 'i2cget' and 'i2cset'.
They can be found in the folder '/usr/sbin':



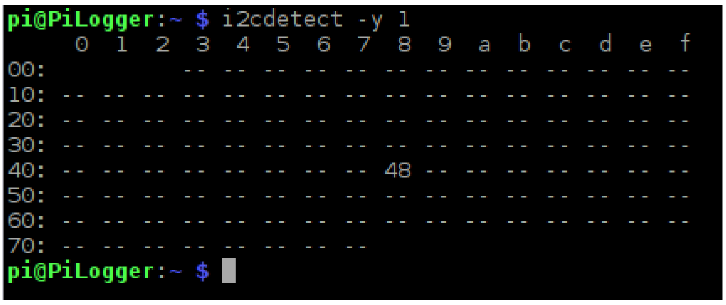
Name	Description	Size	Modified
groupmod	executable	50.5 KiB	17/05/17 13:59
grpck	executable	40.4 KiB	17/05/17 13:59
grpconv	executable	36.3 KiB	17/05/17 13:59
grpunconv	executable	36.3 KiB	17/05/17 13:59
i2cdetect	executable	13.6 KiB	25/01/17 21:41
i2cdump	executable	17.6 KiB	25/01/17 21:41
i2cget	executable	13.6 KiB	25/01/17 21:41
i2cset	executable	17.6 KiB	25/01/17 21:41
i2c-stub-from-dump	Perl script	5.4 KiB	25/01/17 21:41
iconvconfig	executable	17.8 KiB	15/06/17 21:17
in.identtestd	executable	9.8 KiB	11/11/16 14:19
install-sgmlcatalog	Perl script	4.4 KiB	07/11/16 08:06
invoke-rc.d	D source code	17.7 KiB	02/05/17 12:20

4 items selected (62.5 KiB) Free space: 9.6 GiB (Total: 14.6 GiB)

Especially the tool 'i2cdetect' can be very useful to check the connection and to verify the actual address of the PiLogger.
Execution is started within a console window by this line:

```
i2cdetect -y 1
```

The option -y causes to suppress the obligatory confirmation prompt. The number '1' gives the value for the bus to use (with Rasberry Pi type A this has to be '0'). The displayed result should look 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 tool 'i2cdump' is basically a block read tool and is specially designed for reading older and smaller EEPROMs – please don’t use with the PiLogger.
The tools 'i2cget' and 'i2cset' are meant for reading and writing single registers of an I²C bus device. These are the basic elements for all I²C bus operations and by these tools they are applicable from the command line interface. The respective documentations 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

13.2 Boosting SD Card Speed

The SD card interface can be set to double speed by changing a boot configuration option. By default the processor module is clocked with 50 MHz, but by adding a line to the file 'config.txt' in the folder './boot' the clock can be set to 100 MHz, which in fact is a kind of overclocking.

With the Rasberry Pi 3 using this overclock setting will prevent the detection of the built in bluetooth/wifi module.

For writing to the file 'config.txt' administrator privileges are necessary. The shortest way is to invoke the editor 'Leafpad' directly with super user rights and the name of the file:

```
sudo leafpad /boot/config.txt
```

Then this line is inserted at the end of the file:

```
dtoverlay=sdhost,overclock_50=100
```

Subsequently the file is saved, the editor is closed and the Raspberry Pi is rebooted.

Further informations on the topic SD Card and interface speed are available here:
https://www.sdcard.org/consumers/speed/bus_speed/