raspberry

https://www.linurs.org

Raspberry Guide

Raspberry Guide

Table of Contents

List of Figures

List of Tables

Chapter 1. Copyright

Chapter 2. Raspberry

The very big success of the Raspberry PI http://www.Raspberrypi.org/ has triggered lot of different hardware variants. It uses SoC (System on Chip) from Broadcom having ARM CPUs, GPU and Peripherals. PI stands for Python Interpreter as a goal was to have a computer being able to be used for python.

Important

The standard Raspberry USB power supplies have 5.1V and not 5.0V. Using a 5.0V might work but might print under-voltage warnings to the screen. Obvious long cables can cause the same effect due to the voltage drop.

Figure 2.1. Raspberry PI first model

Other sites: https://elinux.org/R-Pi_Hub

Chapter 3. Raspberry PI OS Installation

Raspberry PI OS is a port of Debian. https://www.raspberrypi.com/documentation/

Debian is know to be a stable distribution but has unfortunately some packages that are many years old.

Important

Be carefully what gets installed. When installing a package then Debian installs also depended packages. But when the package gets removed Debian is unable to know why packages are installed and keeps the depended packages on the system. Those packages remain on the system, take memory space, get updated, call for other depending packages and might cause nasty update errors. The way out is a fresh installation, this should be done anyway every couple of years since Debian is not a rolling release distribution.

Conclusion, try out things on Debian using a fresh SD Card. If not happy, reformat it. If happy, have fun and repeat it on the target system.

cat /etc/os-release shows what is used. Be aware every 2 to 3 a new major release will come and fresh install might be required. The desired fun packages might be again many years old.

Debian #	Name	Release	End of Life
13	Trixie	2025-08-09	2030-06-30
12	Bookworm	2023-06-10	2026-06-10
11	Bullseye	2021-08-14	2024-08-14

There are ways to get new versions of the fun packages, as deviating from the official repositories or visualization. Visualization for an embedded device as a raspberry sounds odd. Using a rolling release Linux distribution as Gentoo might be more adequate in this situation.

cat /sys/firmware/devicetree/base/model will tell the model

cat /proc/cpuinfo show the model and using the Revision number the exact model including the memory size.

vcgencmd get_config int shows also the memory size

Finally **uptime** shows how long the Raspberry is up and running

Prepare the SD Card

There is a big ongoing evolution of **rpi-imager** https://www.raspberrypi.com/software/. So use version 1.9.6 or newer to create the SD Card. Newer **rpi-imager** versions support the new models, a more safe setup and more features. Also headless installation (no need to plug in keyboard and monitor) using OS customizations is supported.

Note

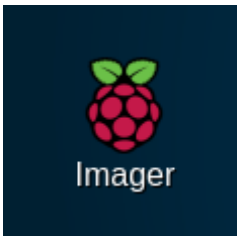
rpi_imager version 2.0.0 has been arrived. It comes as AppImage.

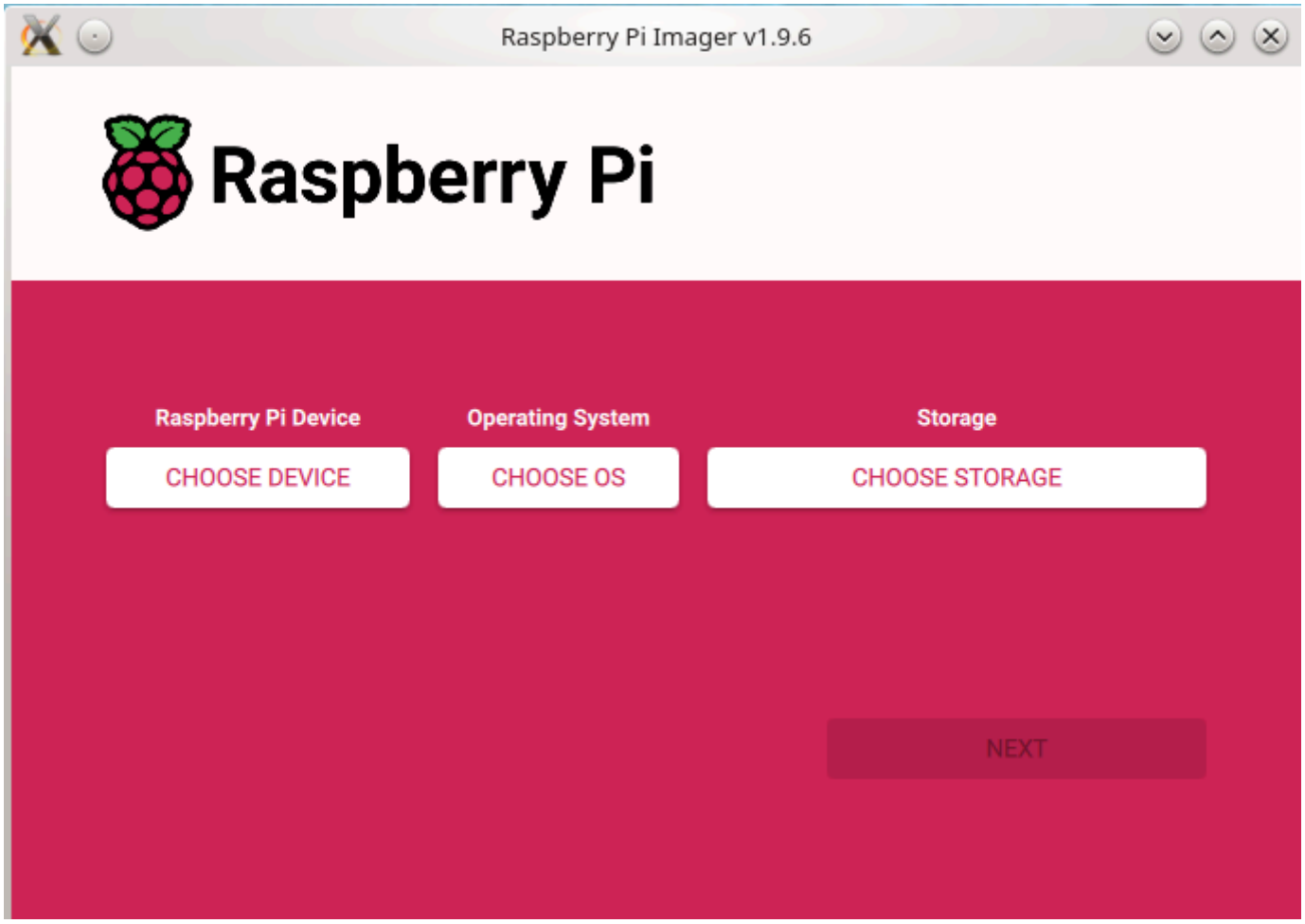
AppImage can run directly under Linux (when the exec permission is set) and contain all required libraries.

It refuses to run without having root permissions. If running as root if allowed by the system administrator do: **sudo ./imager_2.<*x*>.<*y*>_amd64.AppImage**

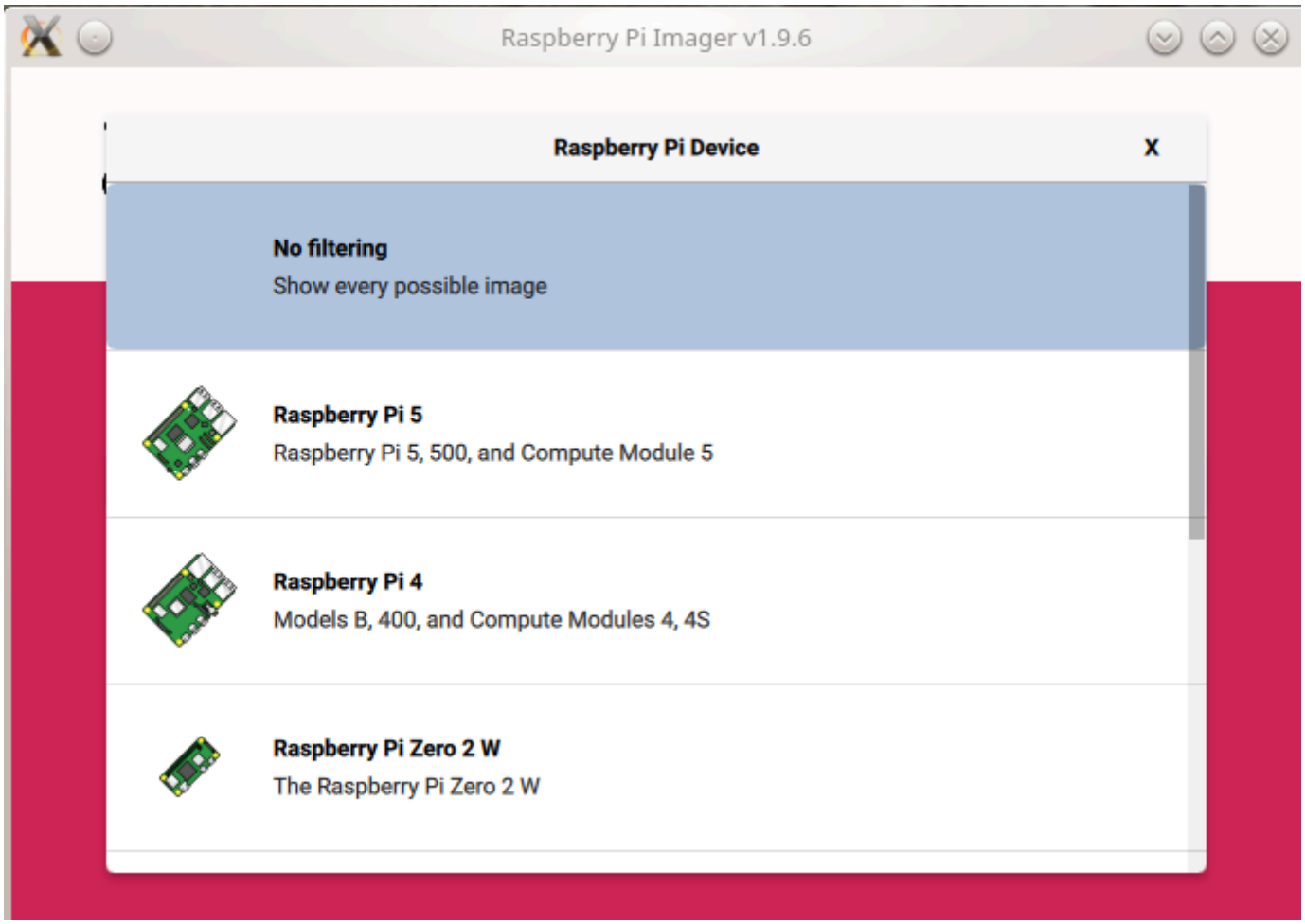
Important

For Raspberry Lite versions do not yet turn on **rpi-connect** within the **rpi-imager**. This might not work. **rpi-connect** can be enabled later



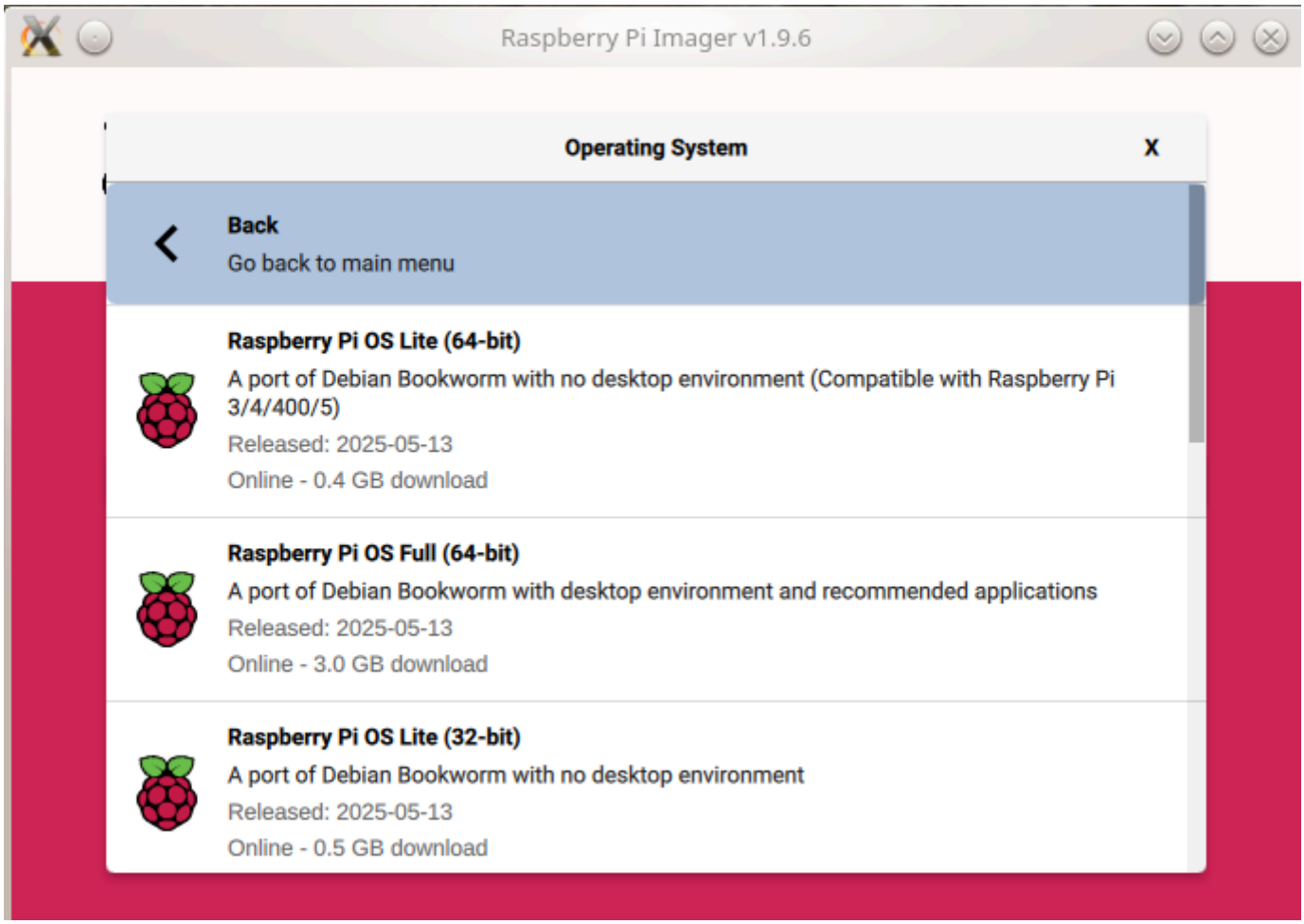


Choose the device

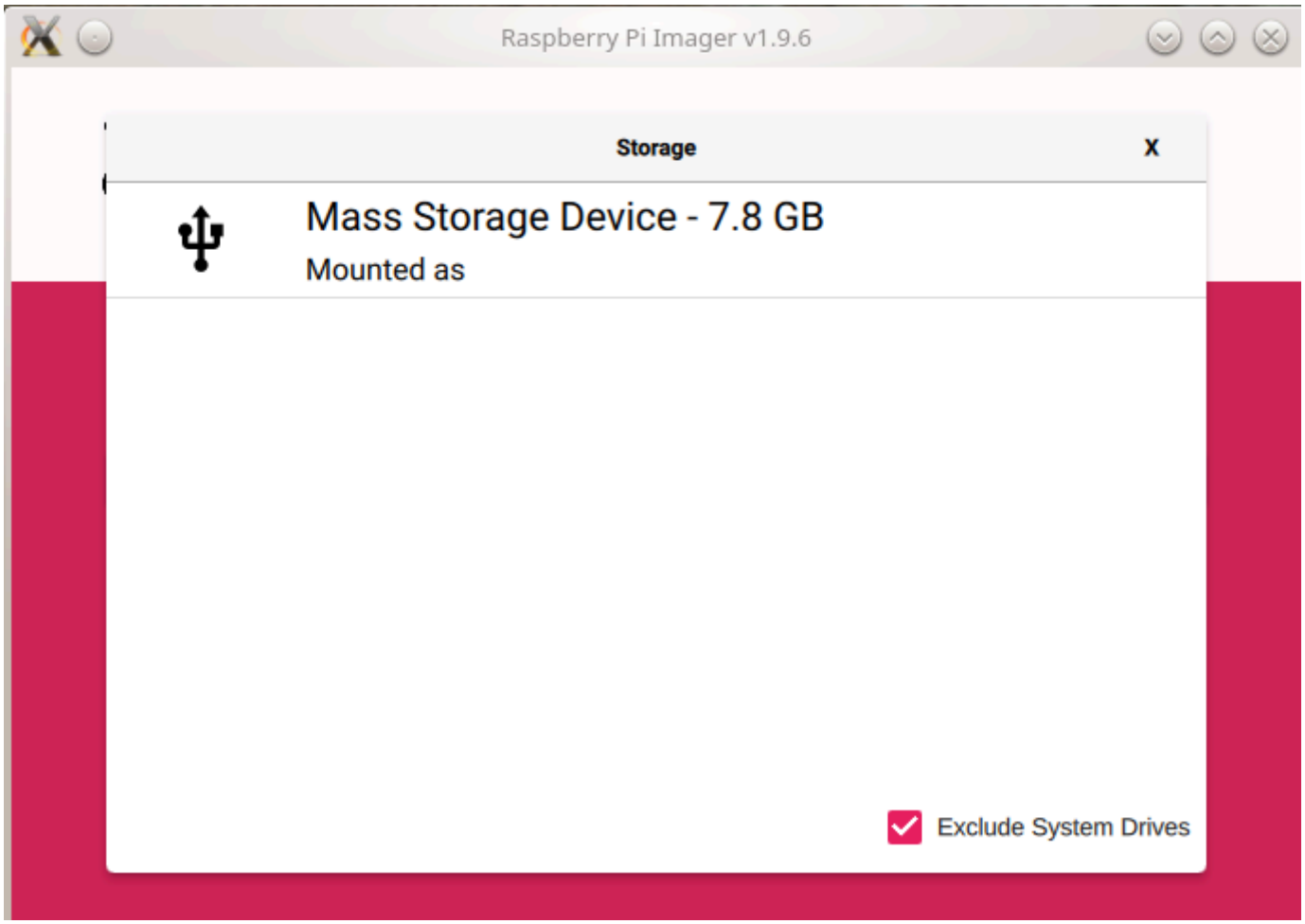


Select one out of the many images.

For headless systems (without graphical desktop, monitor and mouse and therefore suited for IoT devices) use Raspi PI OS Lite, it is less problematic for sw updates, faster and use less memory space.

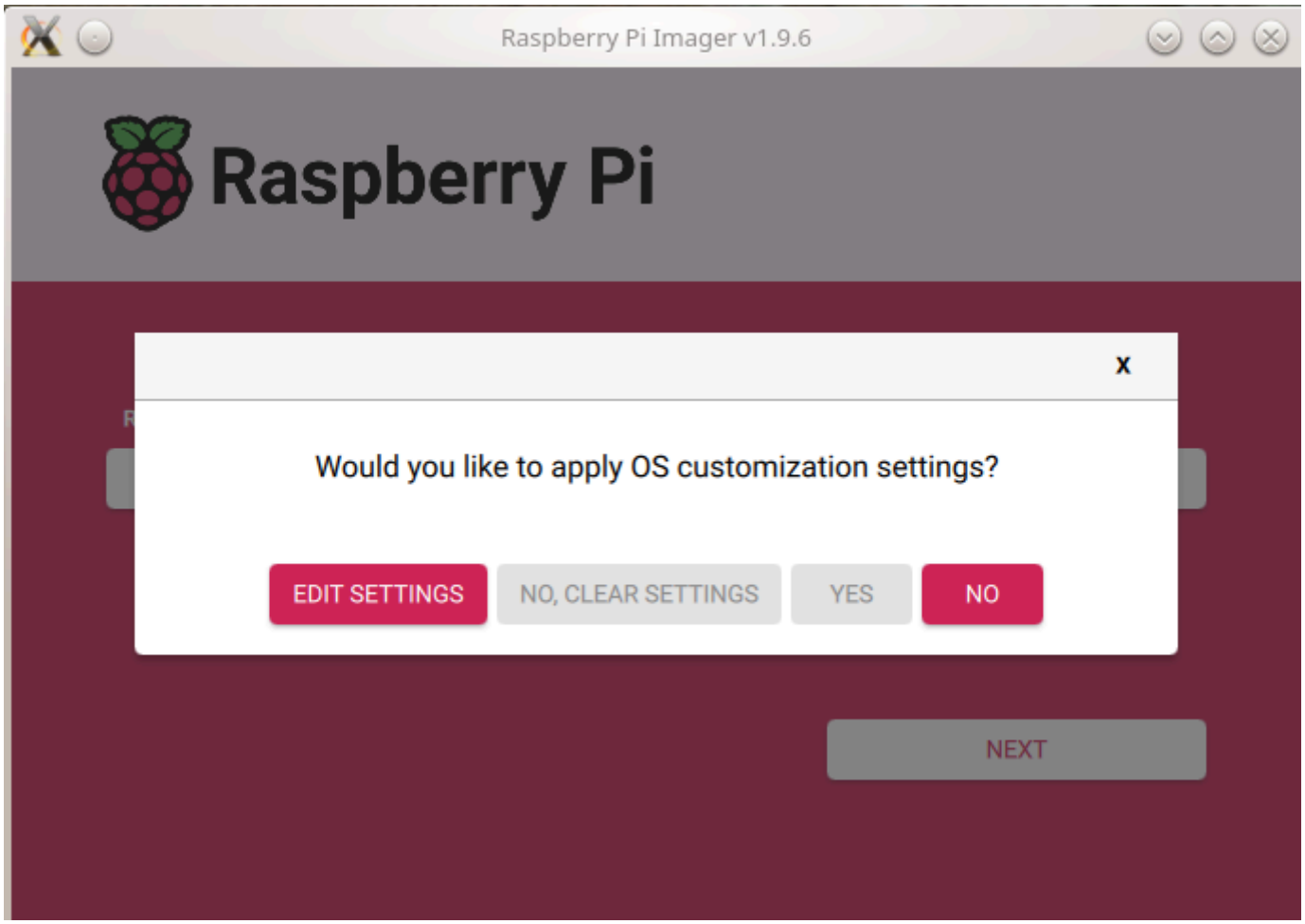


Now take a SDCard an plug it into the computer. Be aware that lifetime of SD Cards are limited, use therefore ones with higher or industrial quality. Newer Raspberry models have support for other media than SD Cards. The smaller the card the better, it is easy possible to move a small disk image to a larger card, the other way around not.

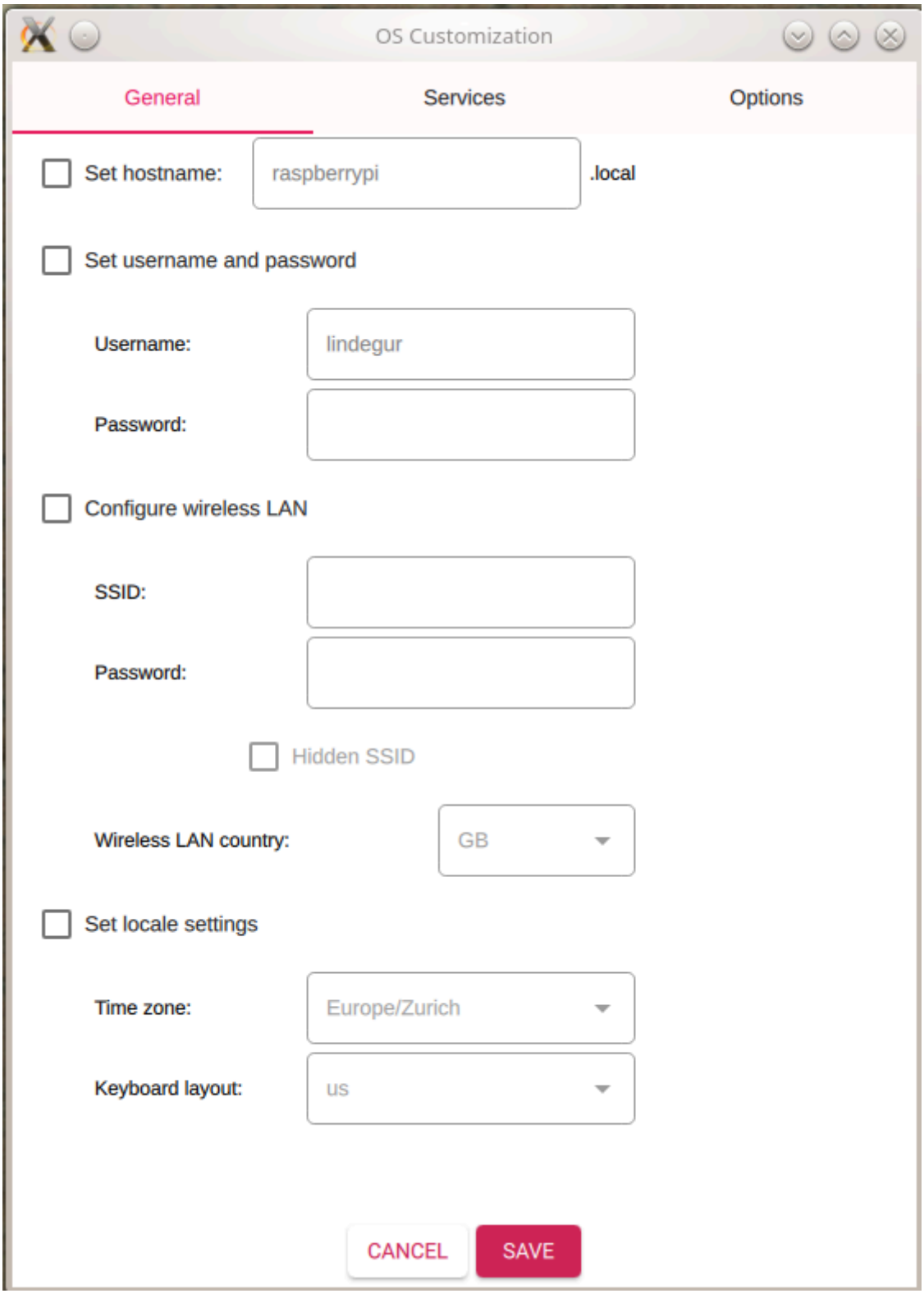


Important

For Headless systems it is important to not skip the OS customization. The OS customization allows to put all what is required to connect to the device via the network and use **ssh** to do the work. So click Edit settings:

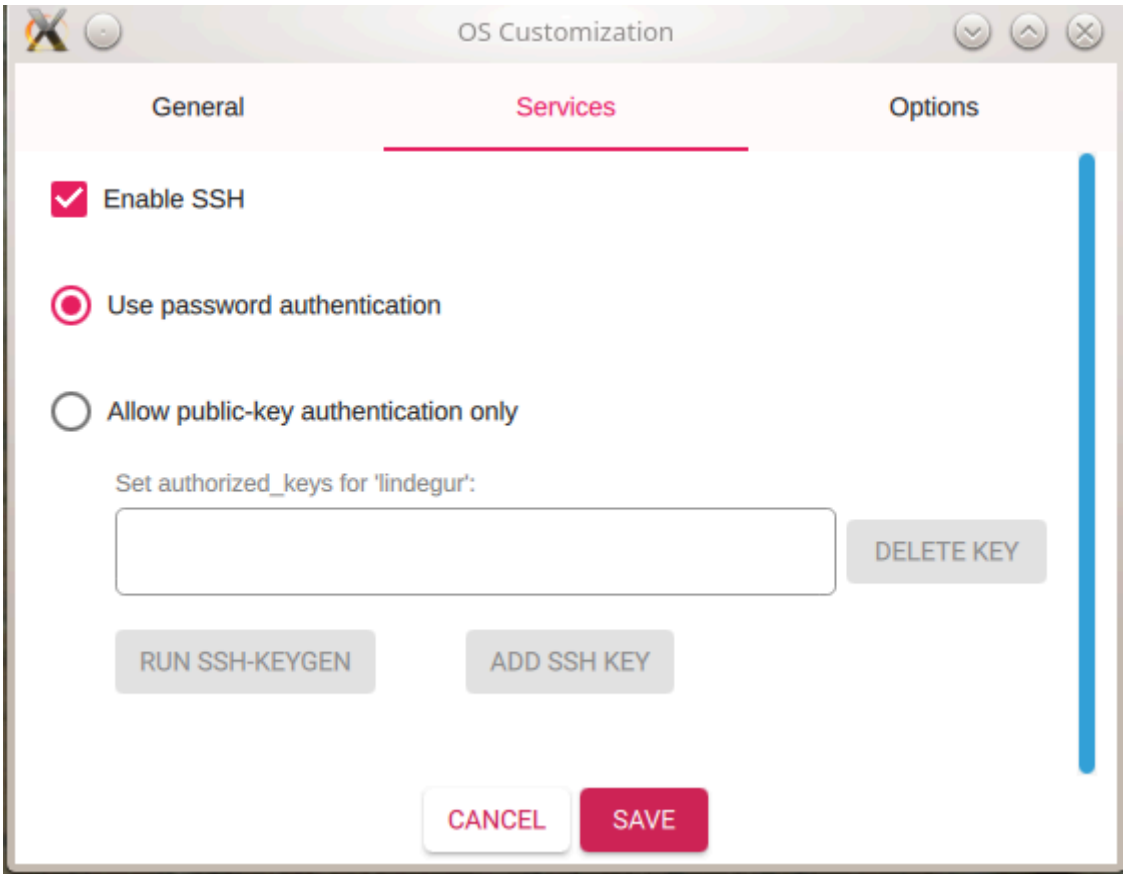


Set now hostname, and username, password, time zone and optionally WLAN settings:

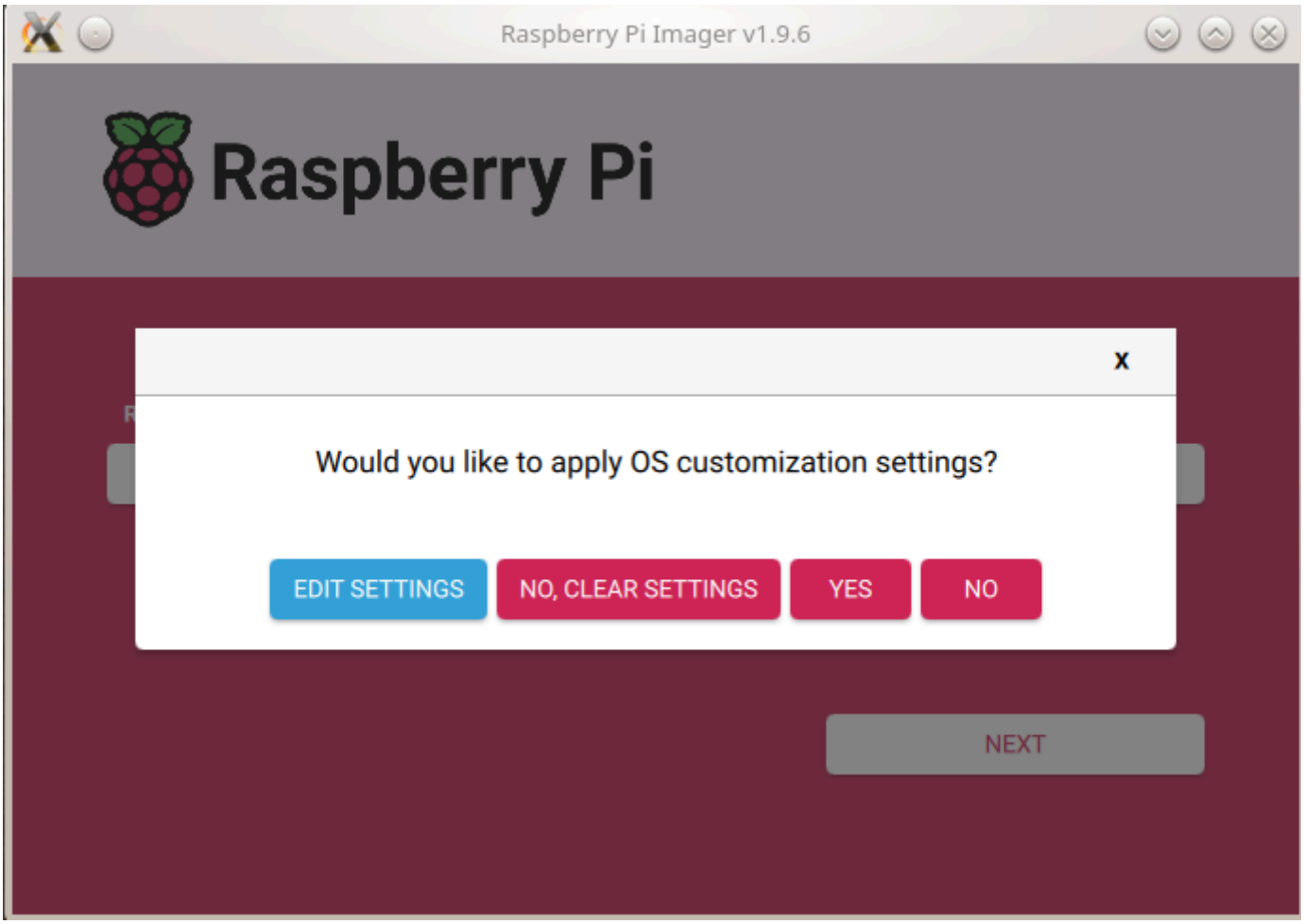


Important

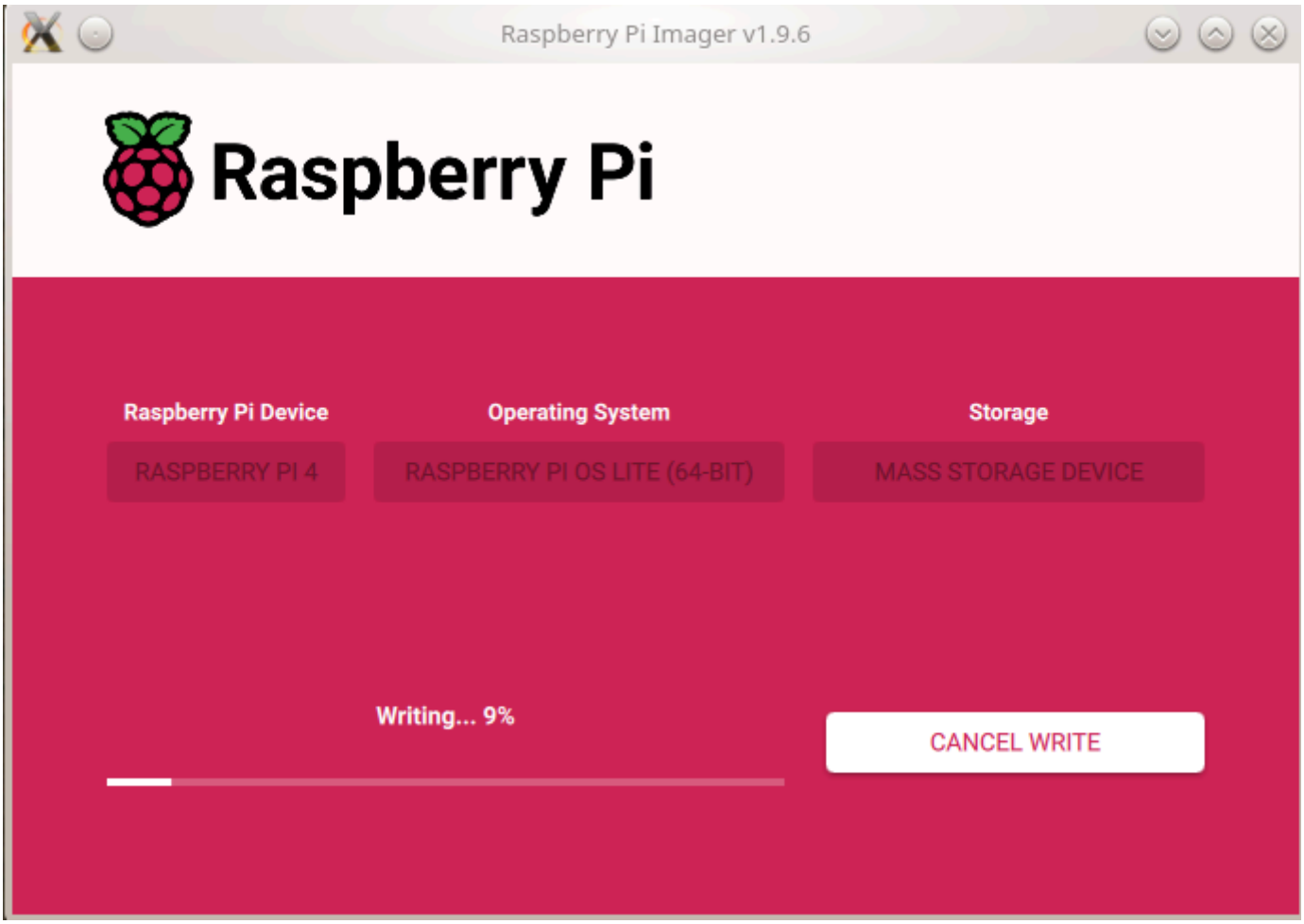
Enable SSH for headless systems. Click run-ssh-keygen to get a the necessary keys to have public-key authentication. This enables the computer used to create the SD card to login using ssh-keys. Click use password authentication as well even public-key authentication is more safe, this allows other computers not having the keys yet to login.



When done click Save and Yes to use the OS Customization and to erase the SD card.



rpi-imager then downloads the OS from the Internet and programs it to the SD Card.



Plug in the SD Card and boot

Check the local router to see if the raspberry appears on wired Ethernet and if configured on WLAN.

Important

Be patient the raspberry has a lot to do and will do a reboot

If nothing seem to happen attach a screen.

Check the network **sudo ifconfig**

ping -c3 <*wired IP*> and **ping -c3 <*wlan IP*>**

Then log in using **ssh**

Power up and using keyboard and screen

When not selecting the right OS customizations then some important final settings are required and the access via network can not be established.

Therefore attach a monitor, keyboard and mouse, plug in the SDcard, then power up.

The first HDMI is close to the USB-C connector. However both work well so use the second connector if there is a mechanical conflict with a HDMI adapter.

If it boots up into a black screen, then it might be an issue with Extended Display Identification Data (EDID) of the monitor and video driver.

If so, add the video mode to `/boot/cmdline.txt`

```
console=serial0,115200 console=tty1 \
root=PARTUUID=37c8e4ff-02 rootfstype=ext4 fsck.repair=yes \
video=HDMI-A-1:1920x1080@60D rootwait
```

Or

`/boot/config.txt` uncomment hdmi_safe=1 if no booting text appears.

In the past the user name was **pi** and the password **Raspberry** (or if keyboard layout not matches the default US keyboard layout **Raspberrz** might work).

Since this is obvious a security issue the newer Raspberry PI OS asks for a user name and password.

Important

Consider to give a temporary password as 123456789. The reason for that is that the keyboard layout probably does not match to your hardware (except when having a US keyboard). Once the keyboard layout is ok, replace the password with strong one.

It is now necessary to create a user and a password having attached a keyboard and a screen (ssh will not work, even there is the "temporary" pi user on the SD-card)

Having attached the screen and the keyboard, the ip address can be easily seen using **sudo ifconfig**

ssh to use the Network

If the ip address is still unknown use a computer to find the Raspberries IP address that got assigned via DHCP. The web page of the router will show the IP address.

If zeroconfig is configured then use Raspberrypi.local as <*ip*>

As regular user use **ssh <*ip address*>** or if working with as different user **ssh <*username*>@<*ip address*>** on a computer (or **putty** if Windows) to access the Raspberry

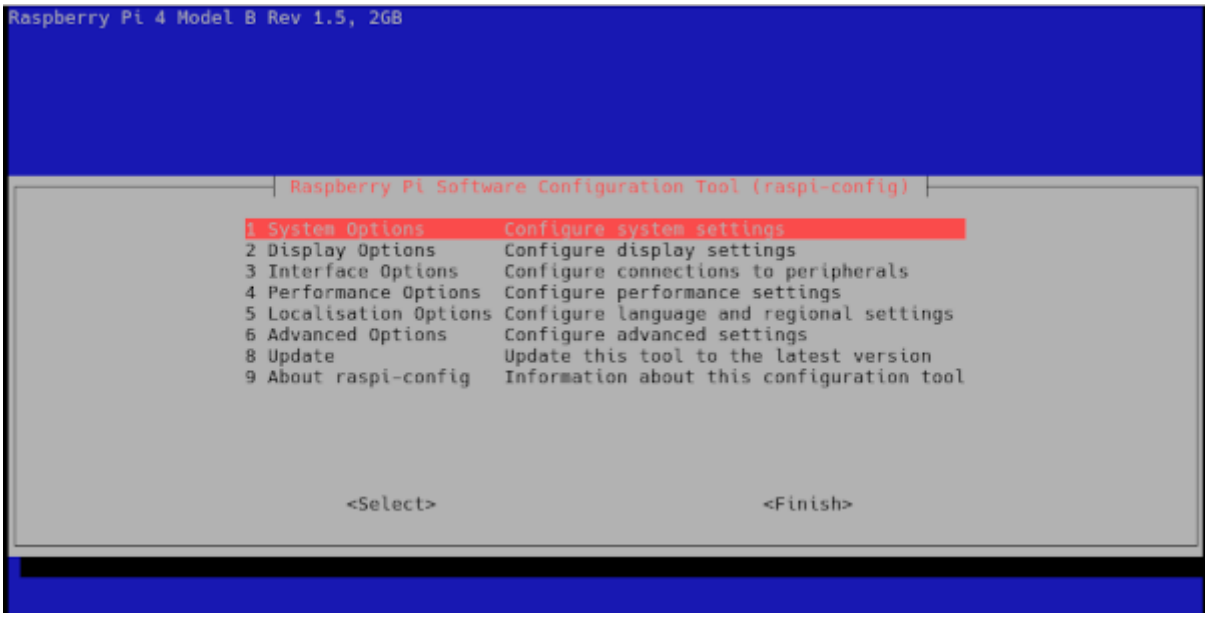
Important

When working frequently with raspberry devices the computer will detect that a different/new hardware got plugged in and becomes scared. Since this is not a cyber attack delete in the `~/.ssh/known_hosts` file the keys created for the previous device and restart ssh.

If **ssh** is not enabled yet, insert the SDcard in a Linux computer and observe **fdisk -l** and then **cat /etc/mtab**. The SDcard will have two partitions: a FAT partition called boot and a Linux partition called rootfs. Windows PCs will most probably struggle with such a SDcard. Enable ssh: **cd /run/media/<*username*>/bootfs/** and create the empty `ssh` file in the boot partition **touch ssh**. Insert the SDcard, connect the Raspberry to the network and power up.

First steps

sudo raspi-config



adjusts the important settings

- Verify the correct keyboard layout, and make sure a strong password is used.

  ## Important

  Just change the password once you are convinced the keyboard layout is correct. If a disaster happens delete the users password hash in `/etc/shadow` to reset the password.

- ## Important

  Do not enable RPI Connect when having a headless system. Since this will install a graphical desktop and Debian will not be able to remove all the depended packages afterward. **sudo apt install rpi-connect-lite** is the way to go for headless systems.

- The SD-card will probably be bigger than the image file. This can be fix by Advanced Options => Expand Filesystem to the SD card size.
- There is also an update command for **raspi-config**

quit **raspi-config** and work in the console

- It is now time to update the Raspberry OS **sudo apt-get update && sudo apt-get upgrade -y && sudo apt-get autoremove && sudo apt-get autoclean**
- Consider to install **mc** the midnight commander (console based file manager)

  sudo apt-get install mc to install it. When edit the first time a file with **mc** it asks for the editor select **/usr/bin/mcedit** for the **mc** editor

To run commands as root: **sudo <*command*>**

sudo su - changes to the root console and after this entering **sudo** is no more required. This is convenient when use it with tab completion as for commands like **cat /etc/wireguard/publickey** but it has a disaster potential, so use it with care.

Raspberry PI OS networking

Per default DHCP is configured. However if the raspberry works as server and has to be reached via the network a static address is desired.

Networking with Debian 12 and newer

Debian 12 Bookworm and Debian 13 Trixie use Networkmanager to configure.

sudo ip a to see what is active

sudo nmtui to get the text user interface of Networkmanager. Edit a connection, select IP4 to manual and enter a manual IP4 address

Address 192.168.1.<*n*>/24

Gateway 192.168.1.1

DNS 8.8.8.8

disable IP6 and save it and **sudo systemctl restart NetworkManager** to restart it

sudo ip a to see what happened, if connected with ssh then the address remains an a **sudo reboot** is required

Advanced setup using nmcli

nmtui is limited, for more settings use **nmcli**

nmcli connection show

In the following use case the raspberry is connected via eth0 to mobile router having low band width access to the Internet but therefore physically portable and cheap. However when moved to the office for development low bandwidth is an obstacle. Turning on to fast wlan0 would be a booster but the Raspberry would still use eth0 since it has a lower metric value. It would however switch to wlan0 when the eth0 cable gets removed. **nmtui** can not do this but **nmcli** can!

To give priority to wlan0 its metric must have a lower value.

nmcli connection show to see the names of the connections

sudo nmcli connection modify "<*wlan SSID*>" ipv4.route-metric 50 set a low metric and therefore high priority

sudo nmcli connection modify "Wired connection 1" ipv4.route -metric 200 set a high metric and therefore low priority

sudo nmcli connection up "<*wlan SSID*>" takes the modification

sudo nmcli connection up "Wired connection 1" takes the modification

To see the result: **ip route show**

```
default via 192.168.1.1 dev wlan0 proto dhcp src 192.168.1.161 metric
50 default via 192.168.20.1 dev eth0 proto dhcp src 192.168.20.5
metric 200
```

curl -s https://ifconfig.me will show the wan Internet address of the wlan0 router and uses high the speed connection.

sudo ping -c3 192.168.20.1 will still find the eth0 router

Observe the files in `/etc/NetworkManager/system-connections`, they contain now the new metric values.

Disable wlan0 and **curl -s https://ifconfig.me** will show the wan Internet address of the eth0 router and use the low speed mobile connection.

Networking with Debian 11

For Debian 11 Bullseye to have a static address edit **sudo nano /etc/dhcpcd.conf** or **sudo mc**

Important

Give a static address if the raspberry works as server and will get accessed over the network. It is also possible to turn on networkmanager instead of dhcpcd and then the steps as for Debian 12 can be done.

Changing Hostname

Having meaningful hostnames are vital in a network. There are many ways to change the hostname, but after reboot and Raspberry PI OS Trixie the old hostname comes back. The reason is that after boot it takes the hostname from `/boot/firmware/user-data`

RPI Connect

The Raspberry foundation offers a cloud service to access a raspberry device from everywhere.

The free version is limited to get access to a single user. There is also a non-free version for multiple and therefore company users.

There is no need to setup the local router or install a VPN. See https://www.raspberrypi.com/documentation/services/connect.html or https://rptl.io/rpi-connect

A device is defined by a UUID that the raspberry foundation creates once it connects to a device. Copy SD cards or using a SD card on a new device will not work. Additional there is a unique number in the hardware: **cat /proc/cpuinfo | grep Serial**

Important

VERY IMPORTANT, the **rpi-connect** version 2.5.2 installs a graphical desktop on headless systems. If this is not desired **sudo apt install rpi-connect-lite**. If it gets installed accidentally then Debian is unable to identify the unnecessary packets and is unable to remove them. A fresh install is required.

If a graphical desktop is ok do **sudo apt install rpi-connect** or select it using **sudo raspi-config** or enable it even during the SD card creation within **rpi-imager**

To turn rpi connect on run as current user **rpi-connect on** and **rpi-connect off** would turn it off

rpi-connect signin will get a link as https://connect.raspberrypi.com/verify/<random key> that when it is opened in a web browser lets create/select an account and adds the device.

connect.raspberrypi.com is the link to re-open it later in a browser where sign in using a raspberry ID is required.

rpi-connect status, **rpi-connect doctor** and **rpi-connect help**

Devices with desktop allow to be connected via desktop sharing or console

rpi-connect-lite

For **rpi-connect-lite** a user needs to enter the command **rpi-connect on** after a reboot.

This can be automated by a user systemd service that is run without having the user logged in.

Important

important is that this service is started without the user needs to be logged in.

This is done by **sudo loginctl enable-linger <*username*>**

Then **systemctl --user daemon-reload**

To disable it **sudo loginctl disable-linger <*username*>**

To verify **loginctl show-user <*username*>**

rpi-connect status if it works

Service file for rpi-connect-lite

As **dpkg -L rpi-connect-lite** shows a `/usr/lib/systemd/user/rpi-connect.service` is provided. As **systemctl --user status rpi-connect** shows this service will be started but might fail.

An improved service can be used instead (as for version 2.5.2). Create a user systemd service directory **mkdir -p ~/.config/systemd/user/**

Add a file to it **touch ~/.config/systemd/user/rpi-connect-lite.service**

Add the content

```
[Unit]
Description=RPi Connect Lite (User Service)
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
ExecStart=/usr/bin/rpi-connect on
Restart=always
RestartSec=5

[Install]
WantedBy=default.target
```

systemctl --user enable --now rpi-connect-lite.service (**--now** does also **systemctl --user start rpi-connect-lite.service**)

Finally **systemctl --user status rpi-connect-lite.service** will show if the service is happy

Wireguard on Raspberry

sudo apt-cache policy wireguard to see if it is in the repository

sudo apt install wireguard -y to get it

To created the keys

sudo su to become root and able to do:

cd /etc/wireguard

$(umask 077; wg genkey | tee privatekey | wg pubkey > publickey) produce new private and public keys. After that **exit** to become a regular user again.

Important

Especially the `privatekey` should be kept in a secret location, it is also recommended to recreate new keys for new things and changing them would not hurt either. However when having lots of devices connected to a central server over long distances, it is a disaster loosing the `privatekey` of the server. It is therefore recommended to do a backup of it.

When changing or updating the server hardware then use the keys from the old server. Prepare everything and then change the routers port forwarding from the old server to the new server.

Now it needs to be decided if the raspberry will act as a server or a client.

- Server Examples:
  - A Raspberry at home that runs 24h a day and can be accessed from everywhere used router to a other location is a typical wireguard server.
  - It will be used as a webserver
- Client Examples:
  - A IoT device located somewhere that needs access to the home network is a typical wireguard client

sudo touch /etc/wireguard/wg0.conf and edit the wireguard configuration file.

Note

When having multiple tunnels do not call them `wg0.conf` and `wg1.conf` give more meaningful names as `wg_<`*`tunnel destination`*`>.conf`

Wireguard Client Configuration

Put an entry in the wireguard server to know the raspberry public key and its VPN wireguard address.

```
[Interface]
Address = <Raspberry VPN wireguard address>
PrivateKey = </etc/wireguard/privatekey of the raspberry>

[Peer]
PublicKey = </etc/wireguard/publickey of the wireguard server>
AllowedIPs = <wireguard address of the server> <optional Addresses that will be
Endpoint = <url or IP address of the server>:51820
```

Important

A tunnel can not be created to a client, the tunnel must be created by the client. Methods to create the tunnel from the client:

- **rpi-connect**
- user accesses it from the clients local network
- use monitor and keyboard attached to the raspberry
- cron job

Wireguard Server Configuration

and edit it

```
[Interface]
Address = <Raspberry VPN wireguard address>
ListenPort = 51820
PrivateKey = </etc/wireguard/privatekey of the raspberry>

[Peer]
PublicKey = </etc/wireguard/publickey of the client device>
AllowedIPs = <wireguard address of the client device>
```

For every device added to the server add a [Peer] entry

Turning on Wireguard

sudo wg-quick up wg0 to bring it up

sudo ifconfig to see if it is there

sudo wg show to see

sudo wg-quick down wg0 to bring it down

To have the tunnel working obviously the server needs to know about the raspberry. Do the necessary work there and restart the wireguard server.

sudo systemctl enable wg-quick@wg0 to have it started automatically

sudo systemctl restart wg-quick@wg0

Important

Stopping the tunnel breaks it. Don’t saw off the branch you’re sitting on, use restart.

```
PersistentKeepalive = 25
```

Can be put optional into `wg0.conf`. It sends out every 25 seconds something so the tunnel will not collapse when not used. Without this the routers on the way of the tunnel will collapse it when not in use.

Raspberry PI Zero W

Low cost and low power but still have Linux makes the Raspberry PI Zero W attractive.

Note

W stands for WLAN there is also a version without WLAN. Since WLAN is the only network connection it might be vital.

It can be setup using the **rpi-imager** to have username, password and connectivity to the WLAN.

Note

It is 32bit and significant less powerful than the top Raspberry models. So be patient on the first boot to have it done its first setup and to connect to the WLAN. Using a HDMI mini (fortunately not micro) adapter allows connecting a monitor to observer what is going on.

The micro USB connector located in the middle is USB On-The-Go (USB OTG). Using a USB OTG cable, Keyboards, Mouse or USB hub can be connected.

The micro USB connector on the edge is the power in connection.

Pin 1 of the GPIO header is located close to the SDcard connector (row closer to the SDcard)

There are also two 2pin headers that could be soldered. The TV provides a video signal and the RUN can be connected to a reset button.

https://learn.sparkfun.com/tutorials/getting-started-with-the-raspberry-pi-zero-wireless/all

Chapter 4. Maintaining Raspberry PI OS

Update Raspberry PI OS

To update a system

sudo apt update

sudo apt full-upgrade

sudo apt-get autoremove

sudo apt autoclean

Or all in one

sudo apt-get update && sudo apt-get upgrade -y && sudo apt-get autoremove && sudo apt-get autoclean

sudo reboot after the update to have all including new kernel started. if the file `/var/run/reboot-required` exist then a reboot is necessary

cat /var/log/apt/history.log shows when last update has been done

sudo shutdown before removing power would be the clean way

sudo df -T to check memory size of the SD card

sudo apt-get clean do remove temporary files from the SD card

Update the Raspberry PI OS using ssh and screen

When connecting via **ssh** to a remote computer and then closing the **ssh** connection closes also the terminal and stops what is in progress. Within the **ssh** console the program **screen** allows to connect to a local terminal (screen session) using **screen -S <*name*>** and then detach **Ctrl**+**d** (or **Ctrl**+**a** followed by **d**), **screen -ls** shows the sessions and session ID's and later on re-attach **screen -r** or **screen -r <*session ID*>**

Ctrl+**a ?** gives help

Backup a Rasperry SD

SDcard to an other SDcard

Raspberry PI OS comes with **piclone** that is a gui application https://github.com/Raspberrypi-ui/piclone.

Do not click the UUID option for a clone. It might happen that side effects might occur due to different UUID.

To not accidentally overwrite the wrong disk, install Raspberry PI OS Desktop in a Virtualmachine as Virtualbox.

To not mix up the source and the destination SDcard, use a simple SD card to USB Adapter for the source and a Multi Memory Card for the destination. The **piclone** shows additionally to `/dev/sd<x>` some device information.

SDcard to a backup file

Write the SDcard to the file

Insert SDcard to be backup-ed into a PC, check with **fdisk -l** or **blkid** what `/dev/sd<x>` or `/dev/mmcblk<x>` it gets and run

dd if=/dev/sd<*x*> bs=4M status=progress | gzip -c > ~/Backup.img.gz

This takes a long time (the SDcard is generally the bottle neck not the USB) observe the MB/s value to get a rough idea how long it takes (it can take more than one hour) and obviously during this time the Raspberry can not run. This might be an issue.

To see exactly how much time it uses type:

time dd if=/dev/sd<*x*> bs=4M status=progress | gzip -c > ~/Backup.img.gz

gzip does a good job, so the file size is surprisingly small.

Write the file back to a SDcard

To an equal or bigger SDcard

To create a cloned SDcard plug in an empty card with at least the same size check it name **fdisk -l** or **blkid** and run

gunzip -c ~/Backup.img.gz | dd of=/dev/sd<*x*> bs=4M status=progress && sync

or to see the time it takes

time gunzip -c ~/Backup.img.gz | dd of=/dev/sd<*x*> bs=4M status=progress && sync

If the card is bigger than necessary then **Raspberry-config** can expand the file-system to use the complete card. The bigger the card the longer it takes to wear-out.

If **Raspberry-config** is not available as when having an other Linux distribution run **e2fsck /dev/sd<*x*>3** to assure having a clean partition. There are different ways.

Use **fdisk** to create a 4th partition **/dev/sd<*x*>4** and use it to store data

resize2fs /dev/sd<*x*>3 and run **e2fsck /dev/sd<*x*>3** to check.

Use **fdisk** delete the 3rd partition and recreate it to have the full size. Then exit **fdisk** with **w** and run **e2fsck /dev/sd<*x*>3** to check.

To a smaller or any SDcard

Use the original SDcard or do the steps above on a equal or bigger SDcard. Plug this source SDcard in the PC and mount it so it appears under `/run/media`.

Then plug in a new SDcard that can also be smaller as long as the files fit. If possible plug it into an embedded SDcard slot so it appears under `/dev/mmcblk<x>` so the probability to make the following steps on the wrong SDcard is reduced. Use **fdisk** to create the empty destination SDcard with the size, partitions and filesystems as desired:

/dev/mmcblk<x>1 128M c W95 FAT32 (LBA)

/dev/mmcblk<x>2 2G 82 Linux swap / Solaris

/dev/mmcblk<x> <rest of it>G 83 Linux

mkfs.vfat -F 32 /dev/mmcblk<x>p1

mkfs.ext4 -T small /dev/mmcblk<x>p3

mkswap /dev/mmcblk<x>p2

swapon /dev/mmcblk<x>p2

and then do **blkid** to see what is there and do for all data partitions

cp -arT <source> <destination>

Important

Be aware that this way the UUID of the partitions will be different. `/etc/fstab` might be required to adapt this.

Chapter 5. Working with Raspberry PI OS

Mounting USB devices

Create a mounting point **sudo mkdir /mnt/usb**

Identify the device **sudo fdisk** -l as `/dev/sda1`

mount manually **sudo mount /dev/sda1 /mnt/usb -o uid=pi,gid=pi**

unmount manually **sudo umount /mnt/usb**

or add it to `/etc/fstab` to have it automatically mounted at boot

No automount happens when usb sticks are plugged in.

For usb devices add the nofail option to `/etc/fstab` otherwise there will be a boot blockage of the raspberry when the usb device is not plugged in

```
/dev/sda1 /mnt/usb    vfat    defaults,nofail   0       2
```

Exchange files with a Raspberry

Exchange files with mc

mc (the midnight commander) has many options to manipulated file on a Raspberry. Since it is text based it runs well on a Raspberry via ssh. It can however run well on a full blown PC and then show in **mc** the Raspberries directory. On way for that is using File Transfer over Shell file system. it uses a command as **sh://<*user*>@<*Raspberries IP*>/home/<*path*>**

Exchange files using automounter

On a master PC automount for home directories on the raspberry can be installed. This way the raspberry will appear in the master PC's file-system under `/mnt/auto`. This allows to use tools as **unison** to synchronize the files between both. See NFS section in this book

Switching from graphical desktop to headless and back

When most of the time no desktop is required, Power and energy can be reduced and lifetime at least for the fan can be increased by switching the device to command line.

sudo systemctl set-default multi-user.target will make that the device boots in command line mode.

Important

However on Trixie the auto-login gets lost when select boot to CLI via desktop and therefore **rpi-connect** with share screen will not work. Auto-login can be re-enable by re-adding to `/etc/lightdm/lightdm.conf`

```
autologin-user=<user name>
```

sudo reboot now

systemctl get-default to confirm it

sudo systemctl set-default graphical.target then brings desktop back.

Chapter 6. Add SW to Raspberry PI OS

Commands to add, search and observe packets

`/etc/apt/sources.list` contains the links to the packet sources

sudo dpkg-query -l to see what packages are installed

sudo dpkg-query -l | grep <*packagename*> for a specific package

sudo apt-get install --simulate dbus-x11 shows what happen when installed inclusive dependencies

sudo apt-get install mc to install midnight commander

sudo apt search <*packagename*> to see what package can be or is installed. If the word automatic appears, then it got installed as a dependency

remove **sudo apt-get remove --auto-remove <*packagename*>** and **sudo apt-get purge <*packagename*>** and **sudo reboot**.

apt-show-versions -a <*packetname*> shows versions available, installed apt-show-versions might have to be installed first

If the package is not in the repository and just available as *.deb file do with checking and installing dependencies:

sudo apt install ./<*package*>.deb

Or without

sudo dpkg -i <*package*>.deb

dpkg -L <*package*>.deb shows what directories and files got installed

mariadb on Raspberry

Install mariadb https://www.digitalocean.com/community/tutorials/how-to-install-mariadb-on-debian-10

sudo apt update

sudo apt install mariadb-server

sudo mysql_secure_installation this will ask for yes and no, use the defaults

Depending on the configuration root may run **mysql** without password

Run **sudo mysql** or **sudo mysql -u root -h localhost -p** the sql prompt appears

NFS server with auto mount on Raspberry

The raspberries `/home/<path>` directory can be mounted to other computers filesystem

sudo apt install nfs-kernel-server

`/etc/exports` need to hold what will be exported

`/home/<path>  192.168.1.0/24(rw,no_subtree_check,async)`

sudo systemctl restart nfs-kernel-server

On the client system as root (or **sudo**) type **mount -t nfs 192.168.1.<*x*>:/home/<*path*> /<*mounting point*>** to mount it and **umount /<*mounting point*>** to unmount it.

Autofs can be used to easily mount the filesystem at a client and without root permission. Autofs does not need to be installed on the server Raspberry, it needs to be installed just on the client.

For a Gentoo Linux client the following must be added to `/etc/autofs/auto.misc` on Raspberries it is `/etc/auto.misc`

`<mounting point>  -fstype=nfs,rw <Raspberry server ip>:/home/<path>`

/etc/init.d/autofs restart

If the rest of the autofs configuration is done the `/home/<path>` directory appears on the client system under `/mnt/auto/<mounting point>`

Important

Since two computers are involved the user and group id numbers might not match and cause access issues.

Samba

Samba allows to access data via a Windows drive. There are two ways: the shared drive can be on the Raspberry or on the Windows machine.

First samba needs to be added to the Raspberry https://www.raspberrypi.com/documentation/computers/remote-access.html#samba-smbcifs:

sudo apt install samba samba-common-bin smbclient cifs-utils

The configuration is in `/etc/samba/smb.conf` and can/should be verified with **testparm**

testparm -v -s returns all the global parameters without prompt for the dump

The Raspberry might run as server 24h a day whereas the window laptop might be turned on occasionally. So the Raspberry should host the shared network drive.

To let the Windows PC know what they will see modify `/etc/samba/smb.conf` to have something as:

workgroup = <RASPISERVERWORKGROUP>

sudo smbpasswd -a <*username*> is used to set a samba password for a user. Without samba passwords anonymous logn might work but access user home directories fail.

smbclient -L localhost -U% executed on the Raspberry will test the samba server without asking for a password. **smbclient -L localhost** will do the same but as user and asking for the samba password.

Finally there is also **sudo smbstatus**

Access the samba server from an other linux client

It is more native to use NFS for Linux systems, but using samba might be a good test before trying to connect Windows to Linux.

If the two usernames (=sharenames) are the same, the users home directory on the samba server can be accessed by the client as: **smbclient //192.168.1.<*x*>/<*username*>**

A prompt appears were **help** list the commands, **cd**, **ls** lets browsing in the share names directory

mount -t cifs //192.168.1.<*x*>/<*username*> /mnt/<*some name*> -o uid=<*n*>, gid=<*m*>, username=<*username*>, password=<*escaped password*> mounts it

> **Important**
>
> Passwords usually contain special characters as $, those characters need to be escaped as \$

umount /mnt/<*some_name*> unmounts it

Access the samba server from a Windows client

Adding a share is straight forward, just adding ip address and sharename **\ \192.168.1.<*x*>\<*username*>** as network drive or network address.

Adding a share is straight forward, just adding ip address and sharename **\ \192.168.1.<*x*>\<*username*>** as network drive or network address.

Apache

sudo apt install apache2 -y gets it

sudo systemctl status apache2 shows if it is running

Typing into a browser the Raspberries Ip Address will show the file under `/var/www/html/index.html`

PHP server side scripting

To get php (server side scripts) for apache **sudo apt install php libapache2-mod-php -y**

Create a test php script

cd /var/www/html

sudo touch php.php

sudo mc and add

```
<html>
  <body>
    <?php
      phpinfo();
```

```
    ?>
  </body>
</html>
```

As **whoami** shows PHP script run as user www-data.

PHP has just the rights of the user www-data and might not run system commands commands

There are very good reason why this is like that.

It is more secure to run a cron service and write data to the disk (ram disk `/run` if it changes frequently) then have php directly get the data.

Docker

Debian is not known to have the newest packet versions.

Also complexer application as homeassistant require containers as docker.

To get out of this dependency nightmare containers as docker can be used to supply all what is depends with the application.

Install docker and run hello-world

see https://docs.docker.com/engine/install/debian/.

This is done by adding the docker repository.

sudo apt-get update as usual work with the newest things

Add Docker's official GPG key: **sudo apt-get install ca-certificates curl**

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

Add the repository to Apt sources:

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update to get aware of the things above

then install docker **sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**

To have docker running as none root,

sudo usermod -aG docker $USER add the user to the docker group

newgrp docker to let the shell know about it. Alternatively logout and login.

Logout and login again and run **docker run hello-world**

The command looks for the hello world container on the raspberry and if not found it down loads it from the docker hub.

Run docker

As it already could be noted when docker runs it checks locally if it finds a container if not it tries to download it from the Internet. **docker ps** shows what docker is running.

The docker applications are stored once under `/var/lib/docker`.

The -v option behaves as a link between system and the docker container.

As example **-v /opt/docker/homeassistant:/config** creates a link between the systems `/opt/docker/homeassistant` directory and the dockers `config`

Home Assistant

https://www.home-assistant.io/, the docs https://www.home-assistant.io/docs/ and the forum https://community.home-assistant.io/

When having dedicated Raspberry device for Home Assistant follow guides as https://www.home-assistant.io/installation/raspberrypi/ and get directly a Raspberry Home Assistant SD card image. This way gets the Home Assistant Operating System.

It is also possible to install Home Assistant OS in Virtual Box.

See https://www.home-assistant.io/installation/linux/ after it is installed virtual box starts showing a text console running the Home Assistant CLI (a console line interface). when typing **login** the prompt changes to the linux root. **localectl status** and **localectl set-keymap de_CH-latin1** to get swiss german key map. **exit** bring back the `ha>` prompt. **host shutdown** shuts down Home Assistant in the proper smooth way.

It also tells to use http://homeassistant.local:8123 and http://homeassistant.local:4357/

If the Raspberry is serving primary for other purposes than Home Assistant then Home Assistant can be added and installed in a container https://www.home-assistant.io/installation/

Note

Home Assistant Operating System will come with the App Store, allowing to install additional supported applications to the Home Assistant OS.

Home Assistant in a container makes use of the running host system. It does not come with the App Store but the additional applications (and any other application) can be installed the host system using its package manager.

Install Home Assistant in docker

See https://www.home-assistant.io/installation/alternative/#install-home-assistant-container or https://www.home-assistant.io/installation/raspberrypi-other. There are different container solutions but the guide focuses on docker. Docker must be first installed

Run the docker home assistant container

Before starting the installing it decide for:

- Time Zone TZ=<*Europe/Zurich*>
- Path where the home assistant docker configuration goes as `/opt/docker/homeassistant`.

Then create this directory **sudo mkdir -p /opt/docker/homeassistant**

Give permission for the local user

sudo chown -R $(id -u):$(id -g) /opt/docker/homeassistant

docker run -d --name homeassistant --privileged --restart=unless-stopped -e TZ=Europe/Zurich -v /opt/docker/homeassistant:/config -v /run/dbus:/run/dbus:ro --network=host ghcr.io/home-assistant/home-assistant:stable

If successful docker starts and since home-assistant docker container is not on the local machine it gets downloaded from docker hub and this takes a while.

--network=host makes that the local machines IP address can be used and not the docker one showed by **sudo ifconfig**

docker ps -a shows if homeassistant is running.

For automatically startup a systemd file needs to be created: **sudo touch /etc/systemd/system/homeassistant.service**

```
[Unit]
Description=Home Assistant Container
Requires=docker.service
After=docker.service

[Service]
Type=oneshot
RemainAfterExit=yes
WorkingDirectory=/opt/docker/homeassistant
ExecStart=/usr/bin/docker run -d --name homeassistant --privileged --restart=un
ExecStop=/usr/bin/docker stop -t 60 homeassistant
ExecStopPost=/usr/bin/docker rm -f homeassistant

[Install]
WantedBy=multi-user.target
```

docker ps -a check if docker still runs homeassistant

docker stop homeassistant stop it

docker rm homeassistant remove temporary files

sudo systemctl daemon-reload make the system aware of the new service

sudo systemctl enable homeassistant.service tell it to start it automatically next boot

sudo systemctl start homeassistant.service start it manually

sudo systemctl status homeassistant.service check its status

sudo journalctl -u homeassistant.service -f check its log

to update homeassistant

Stop the service **sudo systemctl stop homeassistant.service**

get the update **docker pull ghcr.io/home-assistant/home-assistant:stable** it will also tell if there is an update

docker rm homeassistant remove temporary files

sudo systemctl start homeassistant.service start it again

docker exec -it homeassistant bash

wget -O - https://get.hacs.xyz | bash -

restart and add the HACS integration

Run Home Assistant

Once installed, it can be accessed as http://<*ip address*>:8123

There is also an app available for mobile devices

Some terms to know

- Integrations are ways to communicate to devices or protocols
- Entity ID are identifiers of states to be exchanged. States are values as on or off, or a numeric value and can be seen as data. State changes can be used in triggers as events to be processed.
- Helpers can take their data from everywhere including Home Assistant User Interface. So helpers do not need hardware. Helpers can count
- Dashboards can be created holding Cards and the cards hold Entities. There is also the Grid Card that holds other Cards and allows therefore to group Cards.
- Scripts are as automations but do not have triggers. Scripts have therefore be used in automations.
- Scenes allow to send actions to multiple devices instead of sending actions to multiple times to individual devices
- Blueprints are automation templates and scripts to not need ti start from zero
- Tags are NFC or QR tags that can be used as triggers
- Sensors are entities that provide a value
- A switch can provide its value but can also be toggled, it is therefore bi-directional
- A light is like a switch but has additional functionality, as brightness and RGB color

`/home/homeassistant/.homeassistant` hold the data including yaml files

Automations

Automations is the logic added to Home Assistant that takes inputs or events and controls outputs or actions. It is what Home Assistant does automatically. Automations have triggers.

Automations can be created and edited via the gui or yaml.

The following example turns on an output of a tasmota device at 8 o'clock:

```
alias: Tasmota Switch On at 20:00
description: Turn on switch.tasmota2 at 20:00
trigger:
  - platform: time
    at: "20:00:00"
action:
  - service: switch.turn_on
    target:
```

```
      entity_id: switch.tasmota2
    data: {}
```

Or turn it on at sunset:

```
alias: Tasmota Switch On at sunset
description: Turn on switch.tasmota2 at sunset
trigger:
  - platform: sun
    event: sunset
    offset: "0"
condition: []
action:
  - service: switch.turn_on
    target:
      entity_id: switch.tasmota2
    data: {}
mode: single
```

And a humidity alarm:

```
alias: Humidity Alarm Bathroom On
description: Humidity is above level
trigger:
  - platform: numeric_state
    entity_id: sensor.shellyplusuni_a0a3b36865f4_humidity
    above: 45
condition: []
action:
  - type: turn_on
    device_id: f9ecf0bcd602949c51a780527ac19b57
    entity_id: switch.shellyplusuni_a0a3b36865f4_switch_0
    domain: switch
mode: single
```

Add a shelly to home assistant

Shelly's https://www.shelly.com/ appear as WLAN servers creating their own network and have ip addresses as http://192.168.33.1

The first step is using a WLAN capable PC and check for available WLAN networks as SSID: Shelly1PMMiniG3-<*number*> . Then connect to the network. After this open a browser with http://192.168.33.1 and the Shelly's device web page appears

Now configure the WLAN settings so it uses the same network as the home assistant device. A static address can be set to have a defined IP address. Then enable the network to have now the Shelly accessible to home assistant and it stops blinking its red LED. It however stays also in its SSID: Shelly1PMMiniG3-<*number*> using http://192.168.33.1. Everybody could connect to it, so it is a cyber security issue, set a user name and password and to be more safe disable the shelly's access point feature. This however has the risk that a factory reset might be required in the future, if password or user names will be lost or if the WLAN home network changes.

Important

Enable access point when modifying something on the WLAN home network. If it goes wrong then the Shelly can be accessed otherwise a factory reset is required.

Factory resets could be done by pressing the buttons on the shelly's for longer than 10s or if no button is present by multiple quick power on off cycles.

Since it is now connected to the Internet it is time to check for firmware updates.

The Shelly has lot of functions and features to configure and operate itself without using home assistant. If using home assistant, it needs therefore be taken care that no interference between the home assistant and the local Shelly configuration occurs.

The Shelly and home assistant must communicate themselves using one out of the many protocols available. However both must use and understand the same protocol.

Many Shelly's are directly found by Home Assistant and will be directly integrated in its GUI.

However some Shelly's as Generation 3 are not detected yet and can be used by defining their protocol. One common protocol (not just for Shelly's) is MQTT. If not having Home assistant OS or Supervised then a MQTT broker is missing so **sudo apt install mosquitto mosquitto-clients** and configure **mosquitto**.

Add OctoPrint to Home Assistant

Note

If it is discovered automatically check the IP address. If the address is a wireguard address then ignore the device and add a OctoPrint device with the local private network address

When adding the OctoPrint device to Home assistant then the OctoPrintes user name needs to be entered and OctoPrint started. OctoPrint then asks to grant the connection.

There are different implementations for webcams in OctoPrint. OctoPrint tries to use them but not to set them up. Therefore the Webcam can use the Home Assistant MJPEG IP Camera Integration using a stream as http://<ip>:8080/stream. The Picture Entity Card can then be used to show the camera stream.

configuration.yaml

The GUI has its limits for configuration especially when integrations are missing or generic communication protocols are used. In this case the configuration file is usually in `/home/homeassistant/.homeassistant/configuration.yaml` needs to be edited.

If working in a console work as user homeassistant:

sudo -u homeassistant -H -s

In Home Assistant Gui Developer Tools CHECK CONFIGURATION https://www.home-assistant.io/common-tasks/core/#configuration-check or **source /srv/homeassistant/bin/activate** and **hass --script check_config**

For new items a restart is necessary after that modifications need just a YAML reload, click an green tick should appear.

To get a modular configuration `configuration.yaml` can includes other yaml files. To move out the mqtt stuff in a separate `mqtt.yaml` file add to `configuration.yaml`

```
mqtt: !include mqtt.yaml
```

then create the `mqtt.yaml` without repeating mqtt again:

```
sensor:
  - name: "raspi_soc_temperature"
    state_topic: "raspi/soc/temperature"
```

MQTT support

It is assumed that a MQTT broker as mosquitto is running and configured in Home Assistant running.

Read from a custom MQTT sensor

MQTT and subscribe values

Test first that MQTT is able to communicate.

Create a script that reads and publishes the raspberries SoC temperature:

```
#! /bin/bash
mosquitto_pub -t "raspi/soc/temperature" -m "$(/usr/bin/vcgencmd measure_temp |
```

sed is used to just get the numerical value. This script could be added to **cron**

Check in an other console that subscribing is working:

mosquitto_sub -t "raspi/soc/temperature" -u <*username*> -P <*password*>

Now it is time to edit `/home/homeassistant/.homeassistant/configuration.yaml`

```
mqtt:
  sensor:
    - name: "raspi_soc_temperature"
      state_topic: "raspi/soc/temperature"
```

In Home Assistant -> Developer Tools -> CHECK CONFIGURATION and reload the YAML

A new entity sensor.raspi_soc_temperature should now appear and be used as in a gauge card.

MQTT and subscribe Json

The payload might be json instead of single data. For every data structured in Json a separate MQTT entry must be done, some data is nested so the path needs to be included. The entries would look as:

```
sensor:
  - name: "raspi_soc_temperature"
    state_topic: "raspi/soc/temperature"
  - name: "raspi_manager_battery_soc"
    state_topic: "raspi/manager"
    value_template: "{{ value_json['battery' ].soc }}"
  - name: "raspi_manager_power"
    state_topic: "raspi/manager"
    value_template: "{{ value_json.power }}"
```

Write to a custom MQTT sensor

MQTT and send to Shelly Mini1PMG3

Shelly Mini1PMG3 is not yet directly supported in Home Assistant and has a rather complicated payload to be turned on. Everything is therefore be packed into payload_on and payload_off

```
switch:
  - name: "mini1pmg3"
```

```
    command_topic: "shelly1pmminig3-5432044f7414/rpc"
    payload_on:  '{"id": 1, "src": "hass", "method":"Switch.Set", "params":{"id
    payload_off: '{"id": 1, "src": "hass", "method":"Switch.Set", "params":{"id
    retain: false
```

There is no state_topic and value_template therefore it works just in the publishing direction. To have mosquitto know the payload in case the Shelly Mini1PMG3 is powered off and will be powered on later retain: true must be set.

To have a switch bidirectional, a state_topic must be added, since the switch status is inside json a value_template is required and since the state data is different from command data, state_on and state_off is defined. To work it well during power on an off retain is set to true:

```
switch:
  - name: "mini1pmg3"
    command_topic: "shelly1pmminig3-5432044f7414/rpc"
    payload_on:  '{"id": 1, "src": "hass", "method":"Switch.Set", "params":{"id
    payload_off: '{"id": 1, "src": "hass", "method":"Switch.Set", "params":{"id
    retain: true
    state_topic: "shelly1pmminig3-5432044f7414/status/switch:0"
    value_template: "{{ value_json.output }}"
    state_on: true
    state_off: false
```

command line support

Results from command line calls can be used as entities by adding to `configuration.yaml`

```
command_line:
  - sensor:
      name: "Raspberry Pi CPU Temperature"
      unique_id: rpi_cpu_temperature
      command: "cat /sys/class/thermal/thermal_zone0/temp"
      scan_interval: 60
      unit_of_measurement: "#C"
      value_template: "{{ (value | float / 1000) | round(1) }}"
```

rest API support

Rest API calls can be used by adding to configuration.yaml:

```
rest_commend:
```

Clean entities

Give it a unique id so the gui editor can find it. It is recommended to put a MAC address in the identifier to assure it stays unique

```
unique_id: "raspi4_soc_temp-e45f016dc555"
```

Give a unit to the value

```
unit_of_measurement: "W"
```

MQTT and mosquitto

To install **sudo apt install mosquitto mosquitto-clients**

sudo systemctl status mosquitto to see if it is running

sudo cat /var/log/mosquitto/mosquitto.log to see if it is happy

`/etc/mosquitto/mosquitto.conf` for the configuration

As https://mosquitto.org/documentation/authentication-methods/ says authentication is require one way is using hashed passwords and user names

sudo mosquitto_passwd -c <*e.g. /etc/mosquitto/passwd*><*username*> to create a password file

sudo mosquitto_passwd -D <*password file*><*username*> remove user

sudo mosquitto_passwd <*password file*><*username*><*password*> add/update

and create a `/etc/mosquitto/conf.d/password.conf file` containing

```
listener 1883 0.0.0.0
allow_anonymous false
password_file /etc/mosquitto/passwd
```

To allow remote access

Important

Don't use $ character in the password

Test **mosquitto** in two consoles

mosquitto_sub -h localhost -t "test/topic" -u <*username*> -P <*password*>

mosquitto_pub -h localhost -t "test/topic" -m "Hello" -u <*username*> -P <*password*>

Configure **mosquitto** integration and give as IP 127.0.0.1 with port 1883 and user name and password. The **mosquitto** integration gui allows to subscribe and publish MQTT topics. As first test it can communicate to console windows.

The topics is like a directory tree structure. It allows to use wildcards to adress more than one topic:

- home/+/temperature matches home/livingroom/temperature home/kitchen/temperature but not home/upstairs/bedroom/temperature and home/temperature
- home/# matches home/livingroom/temperature home/kitchen/humidity home/garage/light/status home

MQTT is known to be robust for devices that appear and disappear. This however has to be told to **mosquitto**. QoS (Quality of Service) are options that deal with such nodes. In QoS=1 at least once, it is assured that the message appears but it can appear multiple times. In QoS=2 it is assured that the message appears just one time. In both options **mosquitto** needs to store the message, QoS=2 requires more work and network traffic. The default behavior is QoS=0 meaning messages are sent just once without taking care if they arrive. QoS is passed with the command line option **-q** or **--qos**. The publisher as well as the subscriber can set those options based on network quality and importance of the pay load. Shelly's seem to use fix QoS=1

man mqtt

man mosquito_sub

man mosquito_pub

MQTT from Shelly

The next steps is knowing how to communicate to the Shelly's

mosquitto_sub -h 192.168.1.5 -p 1883 -t shelly1pmminig3-5432044f7414/status -u <*username*> -P <*password*> subscribe the status or

mosquitto_sub -h 192.168.1.5 -t "shelly1pmminig3-5432044f7414/#" -u <*username*> -P <*password*> to get everything from this device

and then

mosquitto_pub -h 192.168.1.5 -p 1883 -t shelly1pmminig3-5432044f7414/command -m status_update -u <*username*> -P <*password*> to ask for the status

mosquitto_pub -h 192.168.1.5 -p 1883 -t shelly1pmminig3-5432044f7414/rpc -m '{"id":"Shelly.GetStatus", "src":"devices/shelly1pmminig3-5432044f7414/messages/events", "method":"Shelly.GetStatus"}' -u <*username*> -P <*password*> new way to get the status

MQTT to Shelly

mosquitto_pub -h 192.168.1.5 -t "shelly1pmminig3-5432044f7414/rpc" -m '{"id": 1, "src": "whatever", "method":"Switch.Set", "params":{"id":0,"on":true} }' -u <*username*> -P <*password*> to turn on the switch

mosquitto_pub -h 192.168.1.5 -t "shelly1pmminig3-5432044f7414/rpc" -m '{"id": 1, "src": "whatever", "method":"Switch.Set", "params":{"id":0,"on":false} }' -u <*username*> -P <*password*> to turn off the switch

Important

src might not be optional, so without src the Shelly might ignore the command. It is an indicator from where the command comes and can have nonsense data as "whatever"

Media Server

Typically a TV with its remote controller and built in software acts as user interface and is the DLNA client.

Therefore the DLNA server is not made visible to the regular user and a simple server as minidlna appears the same way as a featured one as gerbera.

minidlna

See **man minidlnad** and **man minidlna.config**

The configuration file is `/etc/minidlna.conf` and `/etc/defaults/minidlna`. It holds all the paths to be used as media directory and log file path.

The path to the directory or directories containing the media has to be set.

minidnlad the minidlna daemon is meant to be run as user and group **minidlna**

The minidlna daemon is started as usual **sudo systemctl start minidlna**

To see if it is happy **sudo cat /var/log/minidlna/minidlna.log**

The database is at `/var/cache/minidlna`

minidlna database and cache

With the minidlna daemon stopped **sudo systemctl stop minidlna** run **sudo -u minidlna mindlnad -R** to rebuild the media cache, observe in the console. It will not stop.

minidlna autofs issue

If **sudo systemctl status autofs** shows nothing, The **sudo systemctl status minidlna** might show an error as: `/lib/systemd/system/minidlna.service:4: Failed to add dependency on autofs, ignoring: Invalid argument`

The `/lib/systemd/system/minidlna.service` might ask for **autofs**, but the setup does not need **autofs**. Therefore delete autofs in the following line in `/lib/systemd/system/minidlna.service`

```
After=local-fs.target remote-fs.target autofs
```

sudo systemctl daemon-reload and **sudo systemctl start minidlna**

inotify errors

sudo cat /var/log/minidlna/minidlna.log might show errors as `monitor.c:222: warn: WARNING: Inotify max_user_watches [29927] is low or close to the number of used watches [393] and I do not have permission to increase this limit. Please do so manually by writing a higher value into /proc/sys/fs/inotify/max_user_watches.`

I notify watches are used to detect if files and directories get changed. There is a limit for those watches. It can be observed with: **sudo cat /proc/sys/fs/inotify/max_user_watches**

The number can be temporary increased by **sudo sysctl fs.inotify.max_user_watches=<*n*>**

for permanently edit `/etc/sysctl.conf`

`fs.inotify.max_user_watches=<n>`

and **sudo sysctl -p**

It is also possible to not use inotify watches. Add

```
inotify = no
```

in `/etc/minidlna.conf`

Since inotify watches use a lot of resources, disabling inotify watches might be considered for big media data on a small devices with not much data that changes. In this case the the **minidlnad** must be stopped from time to time and restarted with **minidlnad -r** to force a rescan of the data.

udp_notify errors

sudo cat /var/log/minidlna/minidlna.log might show errors as `minissdp.c:324: error: sendto(udp_notify=10, 192.168.10.1): Required key not available`

In this case the IP address comes from an additional wireguard interface. Setting in `/etc/minidlna.conf`

```
network_interface=eth0
```

removes those errors

Gerbera

https://gerbera.io/ is a media server an evolution of mediatomb, it allows devices as TV's to access media data.

sudo apt search gerbera to see what version would be installed and check the Internet what version is actual. Since Debian might offer just a outdated version consider to move away from the official Debian repository and add the one from gerbera, visualization, use a gentoo linux or use the easier package minidlna.

sudo apt install gerbera to get it

gerbera --version shows what version got installed

Do not run gerbera as user , use the systemd service to run https://docs.gerbera.io/en/stable/daemon.html#daemon

sudo useradd --system gerbera

sudo mkdir /etc/gerbera

sudo chown -Rv gerbera:gerbera /etc/gerbera

sudo systemctl daemon-reload to include the service that came with the installation

sudo systemctl start gerbera

As **sudo systemctl start gerbera** shows the web user interface can be reached as http://<hostname or ip address>:49152 if it is disabled enable the ui in `/etc/gerbera/config.xml`

Note

For security reasons it is recommended to have the ui disabled (it is therefore also disabled on a fresh install)

Change the name in `/etc/gerbera/config.xml` to something unique

Chapter 7. Administrator tasks for the Raspberry PI OS

Debian Repositories

cat /etc/apt/sources.list is the old way to list repositories. It usually holds today the official repositories to be used. Adding additional repositories is not advised since the file could become messy.

ls /etc/apt/sources.list.d/ is the directory to hold repositories in a modern, clean and modular way. Every repository has its *.list file.

All repositories in `/etc/apt/sources.list` and `/etc/apt/sources.list.d/` are considered.

Add additional repositories

In Debian many packages are very old or do not exist. A way out is adding additional repositories.

`/etc/apt/sources.list.d/` is the directory to hold additional repositories as *.list file.

As **cat /etc/apt/sources.list.d/docker.list** shows

```
deb [arch=arm64 signed-by=/etc/apt/keyrings/docker.asc] https://download.docker
```

As seen above, repositories need to have a keyring.

If not already exist, create a place for the key-rings **sudo -p /etc/atp/keyrings**

Then download the public pnp key (assuming the curl and ca-certificate packages are already installed) **sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc**

Having the key, the repository can be added as to `/etc/apt/sources.list.d/` as `*.list` file containing the output of the echo command:

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/debian $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

This command finds out the architecture used **dpkg --print-architecture** and Debian version **echo $(. /etc/os-release && echo "$VERSION_CODENAME")**

> /dev/null hides the normal output but would show errors

What the link will find can be opened in a web browser https://download.docker.com/linux/debian/dists/bookworm/stable/binary-arm64/

sudo apt update makes it aware that there is a new repository

sudo apt install <*package*> installs a package from the newly added repository

Debian system integrity checks

sudo dpkg --audit and **sudo dpkg -C** should not return errors and warnings

If **sudo dpkg --audit** clams, create a file with the list of packages and then **sudo xargs -a <*list of packages*>.txt apt-get install --reinstall**

sudo apt-get update and **sudo apt-get install --reinstall $(dpkg -l | awk '/^ii/ {print $2}')** will reinstall everything

sudo apt-get install debsums and **sudo debsums -s** will check the system files

sudo apt-get check

sudo touch /forcefsck and **sudo reboot** to test the disk or unplug the SDcard and do **sudo fsck -f /dev/sd<*x*>**

dmesg | less and **journalctl -p err -b**

User Management

In the past the Raspberry user name was **pi** and and often used with the default password. This is easy to remember but the opposite of being secure. Fortunately newer Raspberry OS versions ask for a user name and password during installation.

Important

For old Raspberry installations just delete the user **pi** when having a new user that is capable of using the **sudo** command. Use the user **pi** to setup permission for the new user

Important

Be aware that changing users uid is not possible while being connected via ssh and logged in as the user being changed. In worst case a temporary user needs to be created or monitor and keyboard be attached to the raspberry.

The following shows how to remove the **pi** user from an old Raspberry installation:

Note

Groups must exist before a user can be assigned to the group. To create a group **sudo groupadd -g <*gid*><*groupname*>**

sudo adduser --uid <*uid*> --gid <*gid*><*username*> creates a new user and files from `/etc/skel` are copied over to the new `/home/`<*username*> directory. **-u** and **-g** are optional but it is wise to assign to the **<*username*>** the same numeric values **uid** and **gid** among different raspberry devices, this makes networking much easier. The user create during stetting up Raspberry OS will get **uid** and **gid** 1000.

lslogins -u shows the logins on the system

getent passwd <*username*> to see its password entry, x is there for historical reasons to tell that the password is encrypted and just a password salted hash is in `/etc/shadow`

groups <*username*> shows what groups the user belongs

groups pi shows what groups **pi** belongs

To do the same as pi copy all groups of pi to the following command

sudo usermod <*username*> -a -G pi,adm,dialout,cdrom,sudo,audio,video,plugdev,games,users,input,render,netdev,spi,i2c,gpio

To work with **sudo** as pi (do not use a regular editor since mistyping could end up in troubles)

sudo cp /etc/sudoers.d/010_pi-nopasswd /etc/sudoers.d/010_<*username*>-nopasswd and then

sudo visudo -f /etc/sudoers.d/010_<*username*>-nopasswd replace pi with <*username*> inside the new copy. Run also **sudo visudo -cf /etc/sudoers.d/010_<*username*>-nopasswd** to thest its syntax.

Important

Use **visudo** as editor since it checks the syntax, do not use a standard editor

If very sure **pi** is no more needed and other users can do **sudo**, **pi** can be removed.

sudo deluser --remove-home pi to delete a user and its home directory,

It might be necessary to remove the pi group before

sudo usermod -g users pi to not use the pi primary group anymore

sudo groupdel pi

vcgencmd

See https://www.Raspberrypi.com/documentation/computers/os.html#vcgencmd and **man vcgencmd**

vcgencmd get_throttled if hex 0 then ok, otherwise check the manual

vcgencmd measure_temp

vcgencmd measure_clock arm

cron

cron allows starts periodically commands. It uses cron tables for that that can be shown by **sudo crontab -l**. The table can be edited a safe way by **sudo crontab -e**

as example

```
0 2 * * * <path>/backup.sh
```

starts everyday at 2 AM a backup

Ram Disks

Do not write frequently to the SD Card. Consider writing data that does not have to be persistent to a ram disk.

The ram disks `ls /dev/ram*` seem to be leftover from early Debian versions and are not used. They could be used however using **mount -t tmpfs -o size=256M tmpfs /mnt/tmpfs** is simpler and more advanced.

More easy is `/run/user`. This is also ram disk as **cat /etc/mtab | grep tmpfs** shows and has even the access right set for users.

Emergency mode

If there are issues with boot the Raspberry might start in emergency mode.

One simple reason for that is that a USB drive defined in `/etc/fstab` is not plugged in.

Guests

If guests have access to the Raspberry it is not that wise to have them having access to the main user account and doing things as **sudo <*command*>**.

sudo groupadd -g 1002 guest create a guest user group

sudo useradd -m -s /bin/bash -u 1002 -g 1002 guest create the guest account

sudo passwd guest give a guest password as welcome

groups guest check if the guest does not belong to groups as sudoers

Raspberry OS uses **ligthdm** as login manager `/etc/lightdm/lightdm.conf` is its configuration file

comment the line

`#autologin-user=<username>`

Next time all users must login. Access and login are still possible with **ssh** and **rpi-connect**

Chapter 8. Hardware

Cooling

Type **/usr/bin/vcgencmd measure_temp** or simply **vcgencmd measure_temp** to see the SoC temperature

and then decide the way how to cool the Raspberry.

Important

Rule of thumb: Stay under 60°C but spend the money for a heat-sink. If it gets too hot use a PWM fan.

The cooler the better, it will live longer.

Aluminum cases acting as heat-sink are a nice option but glue to the chips and are therefore not well suited if the case needs to be frequently opened.

Reduce power helps to dissipate less https://Raspberrypi-guide.github.io/electronics/power-consumption-tricks https://learn.pi-supply.com/make/how-to-save-power-on-your-Raspberry-pi/ https://blues.io/blog/tips-tricks-optimizing-Raspberry-pi-power/

If desktop is just something that occasionally is used, boot to command line mode.

watch -n 2 vcgencmd measure_temp and observe the effects

Fan control

If reducing power, and passive heat sinks do not bring the result a fan has to be used.

2 wire fans can be connected to the connector GPIO 5V power pin 2 (or pin 4) and Ground pin 6. 2 wire fans are not recommended, since they run permanently, making noise and get damaged after a while.

Better are 3 wire fans, the fans can be PWM temperature controller via its third pin GPIO14 (pin 8) that is configured using the performance entry in **sudo raspi-config**

Raspberry 5 fan

The Raspberry model 5 has a dedicated 4 pin fan connector that is controlled by the firmware and configured in `/boot/firmware/config.txt` see `/boot/firmware/overlays/README` for the parameters that can be added

ls /sys/class/hwmon/hwmon2/ shows files having status information about the fan

cat /sys/class/hwmon/hwmon2/fan1_input shows the fan speed

cat /sys/class/hwmon/hwmon2/pwm1 shows the pwm speed of the fan

The result of the configuration can be found in `/proc/device-tree/cooling_fan` however this is binary data. **dtc** is a complicated tool to read the device-tree.

If the raspberry 5 runs with a graphical desktop and no significant load its temperature rises versus 50°C the default temperature when fan starts to spin. If no screen is attached it is worth to disable the graphical desktop or install a headless Raspberry OS.

Compared to a Raspberry 4 the Raspberry 5 gets hotter.

Argon fan hat

See https://argon40.com or https://argon40.com/products/argon-fan-hat

Make sure heatsinks are not too tall

Important

The manual tells to run **curl https://download.argon40.com/argonfanhat.sh | bash** this run the script directly when it is coming from the Internet. This means running something unknown without seeing it and even with **sudo** inside. This should scare every system administrator.

A more safe approach is taking a fresh and empty Raspberry PI OS installation and **curl https://download.argon40.com/argonfanhat.sh > argonfanhat.sh** to save the script, observe it (it has 800 lines). The script installs and downloads a lot and is quite complex.

Run it as **bash -x argonfanhat.sh | tee argon_install.log** then observe `install.log` what happened.

sudo systemctl list-unit-files | grep argon to look for systemd units

Creates symlink `/etc/systemd/system/multi-user.target.wants/argononed.service` to `/lib/systemd/system/argononed.service`

sudo find /etc /usr -iname '*argon*' to look for configurations and scripts

`/etc/argon/argon_uninstall.sh` shows also things that got installed

Final Comment: Is it worth all of this just for a fan?

argonone-config to configure

cat /etc/argononed.conf shows the temperature fan speed table

Table 8.1. Default fan speed table

Temperature [°C]	Speed [%]
55	30
60	55
65	100

argonone-uninstall to uninstall

The features for the button are:

- short press when off turns it on
- short press when on does nothing
- long press when on put Raspberry into soft shutdown
- double press when on reboots Raspberry

Camera

USB cameras following v4l2 are supported by the Debian based Raspberry PI OS, however the Raspberry PI OS standard way is using the camera stack libcamera and picamera2 a python API binding that is optimized for the raspberries camera interface hardware.

Important

If a virtual python environment is used allow it to use picamera2 installed in the system,, so do something as: **python -m venv .venv --system-site-packages**

rpicam-apps

The **rpicam-apps** package **apt show rpicam-apps** comes with the **rpicam-hello** program and others. It uses DRM to show the preview in case the system has no graphical desktop.

rpicam-hello -t0 shows a preview, without **-t0** is times out after 5 seconds

rpicam-jpeg --output "$(date +%Y-%m-%d_%H-%M-%S).jpg" to take pictures

rpicam-hello supports post processing https://www.raspberrypi.com/documentation/computers/camera_software.html#post-processing-with-rpicam-apps. Post processing is configured in a `*.json` file that will be passed via command line. A stage processes camera images. Multiple stage can operate sequentially and are put into the `*.json` file https://github.com/raspberrypi/rpicam-apps/tree/main/assets.or `/usr/share/rpi-camera-assets` hold such *.json files

Important

Open and observe those `*.json` files thy might write to locations as `/home/pi` that might not exist. In such cases create copies and adapt.

cd /usr/share/rpi-camera-assets and **rpicam-hello --post-process-file negate.json**

Important

Some `*.json` files contain in their name _cv. Some of those files as `sobel_cv.json` work. Others as `face_detect_cv.json` will fail due to missing OpenCV features..

sudo apt install python3-opencv might improve however to have OpenCV (and also TFlite) support **rpicam-apps** and **libcamera** must be recompiled to use it https://www.raspberrypi.com/documentation/computers/camera_software.html#build-libcamera-and-rpicam-apps

The sample program can be started with the verbose **-v 2** option, this prints metadata as for object detection and bounding boxes into the console: **rpicam-hello -t 0s --post-process-file /usr/share/rpi-camera-assets/imx500_mobilenet_ssd.json --viewfinder-width 1920 --viewfinder-height 1080 --framerate 30 -v 2**

This is debug information and therefore not well documented and not recommended for productive solutions.

AI Camera

In edge computing the camera gets CPU power. The AI camera has the imx500 chip from Sony that has Iimage Signal Processor and a AI accelerator that can load a neuronal network.

See https://www.raspberrypi.com/products/ai-camera/ and https://www.raspberrypi.com/products/ai-camera/

To get the firmware **sudo apt install imx500-all** to get the firmware **sudo reboot** to get it recognized by the system

`/usr/share/imx500-models/` holds neuronal networks in `*.rpk` format to be loaded into the AI camera

The object detection sample program is started as **rpicam-hello -t 0s --post-process-file /usr/share/rpi-camera-assets/imx500_mobilenet_ssd.json --viewfinder-width 1920 --viewfinder-height 1080 --framerate 30**

The pose detection is started as **rpicam-hello -t 0s --post-process-file /usr/share/rpi-camera-assets/imx500_posenet.json --viewfinder-width 1920 --viewfinder-height 1080 --framerate 30**

For advanced things https://mlsysbook.ai/kits/contents/raspi/raspi.html

picamera2

picamera2 ttps://github.com/raspberrypi/picamera2https://pip-assets.raspberrypi.com/categories/652-raspberry-pi-camera-module-2/documents/RP-008156-DS-2-picamera2-manual.pdf?disposition=inlinehttps://pip-assets.raspberrypi.com/categories/652-raspberry-pi-camera-module-2/documents/RP-008156-DS-2-picamera2-manual.pdf?disposition=inline is based on libcamera and is a python library for Raspberry Pi cameras. As **sudo dpkg-query -l | grep picamera2** shows, **python3-picamera2** is already installed.

However **dpkg -L python3-picamera2** shows that it got installed without its valuable examples.

Those example python scripts are a good start to create applications.

Therefore **git clone https://github.com/raspberrypi/picamera2.git**

Using picamera2 run **python imx500_object_detection_demo.py --model /usr/share/imx500-models/imx500_network_ssd_mobilenetv2_fpnlite_320x320_pp.rpk**

python imx500_object_detection_demo.py -h shows the help

--iou <*values between 0.0 and 1.0*> Intersection-over-Union threshold handles how overlapped boxes are dealt

--max-detections <*n*> defines maximum of boxes to be displayed

python imx500_object_detection_demo.py --print-intrinsics shows information and exits.

`/usr/share/imx500-models/` or https://github.com/raspberrypi/imx500-models/tree/main contain the models and lists what example applications can be used

git clone https://github.com/raspberrypi/imx500-models.git to get the newest models as the yolo models that require the command line option to find the label file **--labels picamera2/examples/imx500/assets/coco_labels.txt**

https://github.com/raspberrypi/picamera2/tree/main/examples/imx500 contains the example programs for the AI camera.

For the object detection there are different versions

imx500_object_detection_demo_mp.py that uses multiprocess and gives smoother video. This demo comes with a menu that allows to take screen shots

imx500_object_detection_injection_demo.py allows customization by injecting the results into the camera pipeline

Streaming video

RTSP is a standard for streams it is robust but creates delays. WebRTC is commonly used for browsers and has little delay.

Mediamtx

RTSP is mature and can stream directly to other devices that understand and use RTSP. But today WebRTC might be preferred since it runs in browsers and produces significantly less delay.

On the local machine RTSP can therefore be streamed into mediamtx and mediamtx then forwards it to WebRTC.

Mediamtx is then used to establish the connection to different IP addresses and is not busing shuffling the data. It however could optionally transcode streams.

RTSP and mediamtx

mediamxt is known to work reliable as streaming session manager that supports RTSP, WebRTC and others. It is probably not available for the linux distribution. Since it is just a simple binary it can be installed manually. **sudo mkdir /opt/mediamtx** and **cd /opt/mediamtx**

sudo wget https://github.com/bluenviron/mediamtx/releases/download/v1.16.0/mediamtx_v1.16.0_linux_arm64.tar.gz and **sudo tar xzf mediamtx_v1.16.0_linux_arm64.tar.gz**

add to `mediamtx.yml`

```
paths:
  stream1:
    source: rpiCamera
```

start **/opt/mediamtx/mediamtx /opt/mediamtx/mediamtx.yml**

ffplay rtsp://192.168.1.134:8554/stream1 to get the RTSP stream

or in the browser tp://192.168.1.134:8889/stream1/ on the other computer using WebRTC

Now adapt the systemd service `/etc/systemd/system/mediamtx.service`

```
[Unit]
Description=MediaMTX RTSP server
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
ExecStart=/opt/mediamtx/mediamtx /opt/mediamtx/mediamtx.yml
Restart=always
RestartSec=2

# Security / cleanliness
User=<user name>
WorkingDirectory=/opt/mediamtx
LimitNOFILE=1048576

[Install]
WantedBy=multi-user.target
```

and **sudo systemctl daemon-reload**

Adding audio to the stream

The source rpiCamera and also **rpicam-vid** do not allow adding audio from an external device.

It is also possible to separate **mediatmx** from rpiCamera to get more flexibility

```
paths:
  cam:
    source: udp://127.0.0.1:1234
```

then launch **rpicam-vid** or an other program to feed it, first without audio

rpicam-vid -t 0 -n --codec libav --low-latency --libav-format mpegts -o udp://127.0.0.1:1234?pkt_size=1316

To get audio **rpicam-vid** has to write into a pipe and **ffmpeg** that reads it and adds an audio stream

rpicam-vid -t 10000 --width 1920 --height 1080 --framerate 30 --codec h264 --inline -o - | ffmpeg -i - -f alsa -ac 1 -ar 44100 -i plughw:2,0 -t 10 -c:v copy -c:a aac -b:a 128k recording.mp4

to reduce delay **rpicam-vid -t 0 -n --codec libav --low-latency --libav-format mpegts -o - | \ ffmpeg -y -loglevel error \ -thread_queue_size 32 -i pipe:0 \ -f alsa -thread_queue_size 32 -ac 1 -i plughw:2,0 \ -c:v copy \ -c:a aac -b:a 128k \ -fflags nobuffer+flush_packets \ -flags low_delay \ -f mpegts "udp://127.0.0.1:1234?pkt_size=1316"**

Once the command works a systemd service can be created **cat /etc/systemd/system/rpi-camera.service**

```
[Unit]
Description=Raspberry Pi Camera Audio-Video Stream
After=network.target mediamtx.service
Requires=mediamtx.service

[Service]
User=<username>
Group=video
ExecStart=/bin/bash -c "rpicam-vid -t 0 -n --codec libav --low-latency --libav-f
ffmpeg -y -thread_queue_size 32 -i pipe:0 \
-f alsa -thread_queue_size 32 -ac 1 -i plughw:2,0 \
-c:v copy -c:a aac -b:a 128k \
-fflags nobuffer+flush_packets -flags low_delay \
-f mpegts udp://127.0.0.1:1234?pkt_size=1316"

Restart=always
RestartSec=5
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

Important

ps -a should not list a running **ffmpeg** or **rpicam-vid** process otherwise the service will fail due to busy audio or camera. Therefore **pkill -9 ffmpeg** and **pkill -9 rpicam-vid**

WebRTC

WebRTC requires Opus Codec for Audio

```
rpicam-vid -t 0 -n --codec libav --low-latency --libav-format mpegts -o - | \
ffmpeg -y \
  -thread_queue_size 1024 -i pipe:0 \
  -f alsa -thread_queue_size 1024 -ac 1 -i plughw:2,0 \
  -c:v copy \
  -c:a libopus -ar 48000 -ac 2 \
  -f mpegts "udp://127.0.0.1:1234?pkt_size=1316"
```

RTSP and rpicam

This way is faster to set up but will not support audio.

rpicam-vid -t 0 -n --codec libav --libav-format mpegts -o - | cvlc stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/stream1}' puts a stream to the raspberries RTSP server.

On an other device **ffplay rtsp://<ip-addr-of-server>:8554/stream1 -fflags nobuffer -flags low_delay -framedrop** will then show the video

To have this more automated create a script as `/opt/rtsp/mediamtx.sh`

```
#!/bin/bash
# RTSP stream using command from Raspberry Pi documentation

exec rpicam-vid -t 0 -n --codec libav --libav-format mpegts -o - | \
     cvlc stream:///dev/stdin --sout "#rtp{sdp=rtsp://:8554/stream1}"
```

A simple service for systemd **sudo touch /etc/systemd/system/rpi-rtsp.service**

```
[Unit]
Description=Raspberry Pi Camera RTSP Stream
After=network.target
Wants=network.target

[Service]
Type=simple
User=<username>
ExecStart=/opt/rtsp/rpi-rtsp-stream.sh
Restart=always
RestartSec=5
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

sudo systemctl daemon-reload

Note

even it is a system process a User gets used since root might fail

As **ss -ltnup | grep 8554** shows

```
tcp    LISTEN    0    4096    0.0.0.0:8554    0.0.0.0:*    users:
(("vlc",pid=1937,fd=3))
```

```
tcp LISTEN 0 4096 [::]:8554 [::]:* users:(("vlc",pid=1937,fd=5))
```

vlc acts as RTSP server but it is limited. It is better to install **mediamxt**

Audio

In the graphical desktop the audio output as for HDMI, 3.5mm Jack and USB Audio device can be selected using the speaker icon. There is also a mic icon where the microphone source can be selected as from the USB device.

USB audio devices are automatically detected.

Audio output

aplay -l or **cat /proc/asound/cards** to see what is there

aplay -D hw:2,0 /usr/share/sounds/alsa/* will speak or fail with a channels error. This might be due to mono files and the device does not support this. **file /usr/share/sounds/alsa/*.wav** confirms this. In this case use the alsa plugins **aplay -D plughw:2,0 /usr/share/sounds/alsa/***

Audio input

arecord -d 3 -f cd -D hw:2,0 test_rec.wav it might also have a format issue with **-f cd**

arecord -d 3 -f cd -D plughw:2,0 test_rec.wav will work

However it is also possible to record the way the hardware can do it.

cat /proc/asound/card2/pcm0c/info shows about the card and

arecord -D hw:2,0 --dump-hw-params /dev/null shows what it supports as **S16_LE** for the microphone attached to the USB audio device

arecord -d 3 -D hw:2,0 -f S16_LE -c 1 -r 44100 test_rec.wav

aplay -D plughw:2,0 test_rec.wav

GPIO

Raspberry-gpio get to see the status and how the pins are configured

Raspberry-gpio get <*n*> to see just one pin

Important

As common today the pins are designed for 0 to 3.3V

To attach something to the GPIO see https://elinux.org/RPi_Low-level_peripherals. As most high density microchips the level are 3V3 so do not attach 5V Logic. There is on pin 1 a 3V3 50mA Power output and on poin2 and pin 4 a 5V power output. The 5V power comes more or less from the 5V power supply and can therefore supply up to 300mA current. Signals on the connector can be programmed as parallel IO, UART, I2C or SPI.

UART appears as `/dev/ttyAMA0` . The pins are:

- pin 6 is ground

- pin 8 is TXD GPIO14
- pin 10 RXD GPIO15

There is a DSI (Display Serial Interface) on S2 and a MIPI CSI-2 (Camera Serial Interface) on S5.

I2C

Make sure that I2C is enabled, if not do it using **sudo raspi-config**. Install the i2c tools **sudo apt-get install i2c-tools**

If enabled **i2cdetect -l** shows the raspberries I2C host interfaces. A Raspberry Zero has two of them:

```
i2c-1 i2c bcm2835 (i2c@7e804000) I2C adapter

i2c-2 i2c bcm2835 (i2c@7e805000) I2C adapter
```

sudo i2cdetect 1 will show what is attached on i2c-1 (as example a LM75 chip)

sudo i2cdump -y 1 0x4f will read the LM75 chip

sudo i2cget -y 1 0x4f 0x00 will read 8 bit from its register 0

sudo i2cget -y 1 0x4f 0x00 w will read 16bit from its register 0

The i2c clock frequency can be modified in `/boot/firmware/config.txt`

```
dtparam=i2c_arm=on,i2c_arm_baudrate=10000
```

to have 10kHz and the do **sudo reboot**

Watchdog

The watchdog is a service that can monitor different things and when something fails a hardware reset can happen. A lot of things must correctly run fine before the watchdog gets alive, so it will not act on hardware failures, its targets are detecting software issues and remove them by a hardware reboot.

Important

It can also send out a e-mails but this requires a complex email setup without lot of control what is sent.

To get it **sudo apt-get install watchdog** and then **man watchdog** and **man watchdog.conf**

For the Raspberry there is the kernel module bcm2835_wdt that creates the file `/dev/watchdog`. For devices not having a hardware watchdog there is a software watchdog driver coming with the kernel source.

basename $(readlink /sys/class/watchdog/watchdog0/device/driver) will show the device driver used

Writing the the dev file will start the the watchdog **echo 1 | sudo tee /dev/watchdog** or **sudo bash -c 'echo 1 > /dev/watchdog'**. A reset will occur after the default time of 60s.

It is not limited to kick (pet) the watchdog as **man watchdog** and the configuration file `/etc/watchdog.conf` show.

To start the watchdog automatically and prevent resets, the watchdog daemon is required.

Configure in `/etc/watchdog.conf` the device to take (or uncomment)

```
watchdog-device = /dev/watchdog
```

Important

The file `/etc/watchdog.conf` must have configured some tests otherwise the watchdog will not do anything

Make sure the repair and test scripts use absolute path even if they are in /etc/watchdog.d

Do not put comments after the commands. Use separate lines

Then **sudo systemctl enable watchdog** and **sudo systemctl start watchdog**

Check with **sudo systemctl status watchdog** to see what can and what will trigger the watchdog

To test it stop the service **sudo systemctl stop watchdog** however it is allowed to stop the watchdog service and therefore nothing will happen.

As **sudo lsof /dev/watchdog** shows (**sudo apt install lsof**), there is a wd_keepal process running that pets the dog:

```
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME wd_keepal 1788 root 4w CHR 1
```

sudo kill -9 <*PID*> will kill the watchdog process wd_keepal then reset will occur.

Once triggered the watchdog requires periodical kicks to not reset, this makes sure the watchdog process is alive. Tests are as:

1. CPU load not to high
2. Memory usage (used and free)
3. Machine temperature too high (can also create warnings)
4. Status of a file
5. Check for running processes and daemons using files containing the PID
6. Ping IP4 addresses

journalctl -u watchdog -f will live (follow) show what the watchdog (unit) logs

The `/var/log/watchdog` directory is used for what the repair and test scripts write. A common place for the scripts is `/etc/watchdog.d`

Repair script

Important

The repair script is called on every interval to find out if something is wrong. The watchdog will not tell if it has found an error.

The repair script is called by the watchdog with $1 = test. This means please test.

If ok return 0 otherwise 1.

With 1 it is indicated that there is a problem. In this case the test script is recalled by the watchdog with $1 = repair. The repair script can do something and when fixed return 0 or return 1 and simply wait until the watchdog triggers.

The test script might use $# number of arguments, $@ all arguments, or $2, $3 to get additional information from the watchdog. The contents depends on the errors.

Test script

The test script is run every interval. If it returns 0 everything ok if it returns 1 then it detected and error.

raspberry