# pi\_ager install on Raspberry Pi 3/Pi 4/ Pi zero w under Pi OS (32-bit) with desktop and recommended software or Pi OS Lite bullseye

* For Pi 4/3: Download and install Raspberry Pi OS with desktop and recommended software from <https://www.raspberrypi.org/software/operating-systems/>
* For Pi zero: Download and install Raspberry Pi OS Lite from <https://www.raspberrypi.org/software/operating-systems/>
* Enable SSH for remote access

sudo touch /boot/ssh

* Setup WLAN configuration

Generate file wpa\_supplicant.conf in /boot:

country=DE

ctrl\_interface=DIR=/var/run/wpa\_supplicant GROUP=netdev

update\_config=1

network={

ssid="WLAN SSID"

psk="WLAN PASSWORT"

}

You may also use Raspberry Pi Imager tool to setup WLAN and an user with password. The new Pi OS bullseye does not provide a default user anymore for security reasons.

* To allow root login:

In /etc/ssh/sshd\_config : remove # from PermitRootLogin and replace prohibit-password to yes. Then restart ssh server: sudo service ssh restart

Set your root password with: sudo passwd root

* Run raspi-config and disable login shell on serial port and enable serial port
* Edit config.txt in /boot to support I2C and SPI devices:

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

# Use Pi-Ager Pins 11/13 GPIO 17/27 for I2C

dtoverlay=i2c-gpio,bus=3,i2c\_gpio\_sda=17,i2c\_gpio\_scl=27

# Use Pi-Ager Pin 16 for MCP3204

dtoverlay=spi1-1cs,cs0\_pin=16

at the end of config.txt add the following lines to support bluetooth and Nextion TFT displays

via serial port /dev/serial0 :

[all]

start\_x=0

enable\_uart=1

dtoverlay=miniuart-bt

force\_turbo=1

* Add in /boot/cmdline.txt at the end of line this to enable USB camera with fswebcam :

dwc\_otg.fiq\_fsm\_mask=0x3

* Disable man-db.service and man-db.timer

sudo systemctl disable man-db.timer

sudo systemctl disable man-db

• Reboot system

* Edit /etc/modules to load i2c-dev at boot, add this line :

i2c-dev

* Add file :

sudo touch /etc/modprob.d/raspi-blacklist.conf

* Install git: sudo apt install git
* Get a copy from Pi-Ager repository to your local system:

git clone –b entwicklung <https://github.com/Tronje-the-Falconer/Pi-Ager>

All project file are now in the folder ./Pi-Ager/

Change working directory to Pi-Ager

cd Pi-Ager

* Copy setup.txt from local repository to /boot/ and edit it as needed:

sudo cp ./boot/setup.txt /boot/

* Copy /etc/modprobe.d/Pi-Ager\_i2c\_off.conf.on from local repository to /etc/modprobe.d/

sudo cp ./etc/modprobe.d/Pi-Ager\_i2c\_off.conf.on /etc/modprobe.d/

* Reboot system
* Update system:

sudo apt-get update

sudo apt-get upgrade

--------------------------------------- php8.2 nur für rpi zero 2w ---------------------------------------------

• Install php 8:

Connect to Raspberry Pi via SSH and execute command to download GPG key:

sudo wget -qO /etc/apt/trusted.gpg.d/php.gpg <https://packages.sury.org/php/apt.gpg>

Add PHP repository:

echo "deb https://packages.sury.org/php/ $(lsb\_release -sc) main" | sudo tee /etc/apt/sources.list.d/php.list

sudo apt update

sudo apt upgrade

sudo apt install php8.2-common php8.2-sqlite3

* Install additional modules for php8.2:

sudo apt install php8.2-mbstring php8.2-zip php8.2-curl

-----------------------------------------------------------------------------------------------------------

--------------------------------------- php 7.4 für alle raspis --------------------------------------

• Install php 7.4

sudo apt install php-fpm php-cli

sudo apt install php

sudo apt install php-cgi php-sqlite3

* Install additional modules for php7.4:

sudo apt install php-apcu php-mbstring php-zip

-----------------------------------------------------------------------------------------------------------

* Install lighttpd:

sudo apt-get install lighttpd

sudo systemctl status lighttpd

sudo nano /etc/lighttpd/lighttpd.conf

and change/add Parameter

server.document-root = "/var/www/html"

to

server.document-root = "/var/www"

add “mod\_auth” to server.modules list

add “mod\_authn\_file” to server.modules list

Save and close nano

sudo usermod -G www-data -a pi

sudo chown -R www-data:www-data /var/www

sudo chmod -R 755 /var/www

* Reboot system

For testing the web server, generate html-page:

sudo nano /var/www/test.html

with content:

<html>

<head><title>Test-Seite</title></head>

<body>

<h1>Das ist eine Test-Seite.</h1>

</body>

</html>

Enter your IP Address (or localhost) into the browser followed by /test.html

In addition we need .htcredentials to contain user and password.

Fort that we use the Online-Tool https://websistent.com/tools/htdigest-generator-tool/

Username: pi-ager

REALM: Pi-Ager

Password: raspberry

Caution! All entries are case sensitive!

Open this file now

sudo nano /var/.htcredentials

and fill in the string output from the generator tool.

Save file with “STRG+o”, “RETURN” and close with “STRG+x”

Now we have to setup the password authentification in lighttpd:

sudo nano /etc/lighttpd/conf-available/05-auth.conf

The following lines are added under server.modules += („mod\_auth“) :

auth.backend = "htdigest"

auth.backend.htdigest.userfile = "/var/.htcredentials"

auth.require = ( "/settings.php" =>

(

"method" => "digest",

"realm" => "Pi-Ager",

"require" => "user=pi-ager"

),

"/admin.php" =>

(

"method" => "digest",

"realm" => "Pi-Ager",

"require" => "valid-user"

),

"/webcam.php" =>

(

"method" => "digest",

"realm" => "Pi-Ager",

"require" => "valid-user"

),

"/notification.php" =>

(

"method" => "digest",

"realm" => "Pi-Ager",

"require" => "valid-user"

)

)

Then we activate this modul:

sudo lighty-enable-mod auth

In addition we have to edit :

sudo nano /etc/lighttpd/conf-available/15-fastcgi-php-fpm.conf

change : "socket" => "/run/php/php8.2-fpm.sock"

or if php7.4 installed: "socket" => "/run/php/php7.4-fpm.sock"

save and close nano

Now enable these modules:

sudo lighty-enable-mod fastcgi-php-fpm

Now reload the the webserver:

sudo service lighttpd force-reload

Now continue to install additional modules:

* Install rust, needed for pip3 install cryptography:

<https://github.com/rust-lang/rustup/issues/1085>

There are two environment variables to control rustup's behavior: RUSTUP\_HOME and CARGO\_HOME. The former controls where to install/find the actual Rust toolchains and the latter where to install the rustup wrapper (in disguise of rustc etc.) After a successful installation, you'll want to configure them for ordinary users' convenience.

So I'd recommend you to install everything under /usr/local/share/rust (maybe /opt/rust ?) and write a simple Bash wrapper to configure the environment variables.

First, run these command as root:

# RUSTUP\_HOME=/opt/rust

# export RUSTUP\_HOME

# CARGO\_HOME=/opt/rust

# export CARGO\_HOME

# curl https://sh.rustup.rs -sSf | sh -s -- -y --no-modify-path

You'll get files such as

/opt/rust/toolchains/stable-x86\_64-unknown-linux-gnu/bin/rustc

/opt/rust/toolchains/stable-x86\_64-unknown-linux-gnu/lib/rustlib/x86\_64-unknown-linux-gnu/lib/libstd-13f36e2630c2d79b.rlib

/opt/rust/bin/rustc

/opt/rust/bin/rustup

Then, install this shell script as /usr/local/bin/rustc and

as /usr/local/bin/rustup and as /usr/local/bin/cargo :

#!/bin/sh

RUSTUP\_HOME=/opt/rust exec /opt/rust/bin/${0##\*/} "$@"

and repeat that for all the commands under /opt/rust/bin/ (you can use ln(1) for your convenience.)

Then ordinary users will be able to use any of standard Rust commands except they cannot install/update Rust toolchains and standard libraries. In particular Cargo downloads will be stored in each user's ~/.cargo.

* Install smbus

sudo apt-get install python3-smbus

* Install sqlite3:

sudo apt install sqlite3

* Install pip3

sudo apt install python3-pip

* Install DHT sensor support

sudo pip3 install Adafruit-DHT

* Install SHT1x sensors

sudo pip3 install pi-sht1x

* Install libgd-dev (needed for new version of fswebcam)

sudo apt install libgd-dev

* Install openssl dev package

sudo apt install libssl-dev

* Install fswebcam:

sudo apt install fswebcam

* Install influxdb

sudo pip3 install influxdb

* Install cryptography

sudo pip3 install cryptography

rust no longer needed! Now remove /opt/rust folder to recover 1.2GB disk space

sudo rm –rf /opt/rust/

* Install wiringpi:

sudo apt install wiringpi

* Install wiringpi new version with Pi4 support :

cd /tmp

wget https://project-downloads.drogon.net/wiringpi-latest.deb

sudo dpkg -i wiringpi-latest.deb

* Copy gpio to /usr/local/bin

sudo cp /usr/bin/gpio /usr/local/bin

sudo chmod 4755 /usr/local/bin/gpio

* Install PiShrink

wget <https://raw.githubusercontent.com/Drewsif/PiShrink/master/pishrink.sh>

chmod +x pishrink.sh

sudo mv pishrink.sh /usr/local/bin

* Nextion serial client (HMI Dislplay support)

sudo pip3 install nextion

* Install lsof command:

sudo apt update

sudo apt install lsof

* Install Locale en-GB and de-DE UTF-8 using

sudo raspi-config

* Enable Serial Interface, disable login, needed for HMI Nextion Display

sudo raspi-config

* Install zip and unzip:

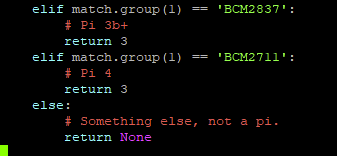
sudo apt install zip unzip

* Workaround for Adafruit\_DHT for Pi4:

In "/usr/local/lib/python3.7/dist-packages/Adafruit\_DHT/platform\_detect.py", you can add the following at line #112 in the elif ladder, so it should workaround the issue.

elif match.group(1) == 'BCM2711':

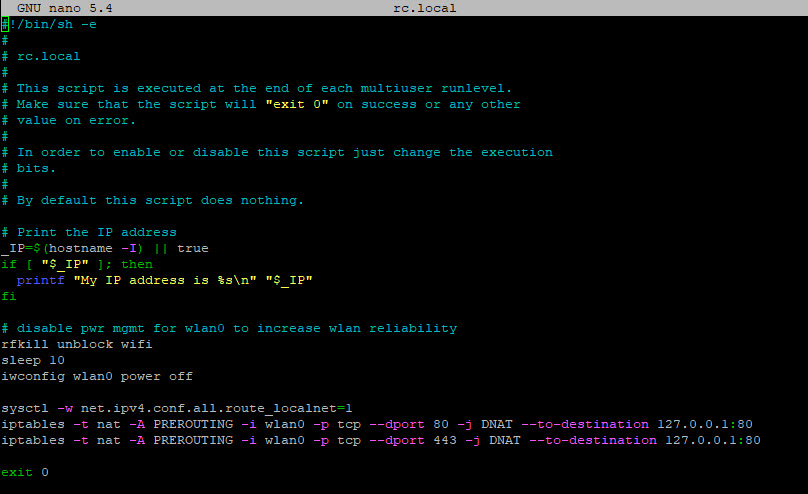
return 3



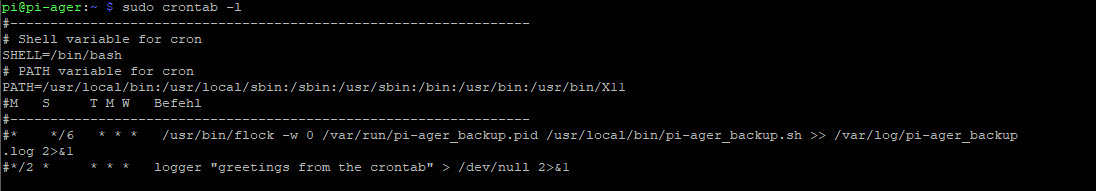
* Unblock wifi for Pi4, add rfkill unblock wifi and disable power management for wlan0:

cd /etc

sudo nano rc.local



* Only if wanted: Generate/edit crontab to prepare for automatic enable pi-ager\_backup.sh



Use visudo to edit /etc/sudoers, so that the www-data User (User of Website) can execute /var/sudowebscript.sh :

sudo visudo

and then in sudoers following

...

#User privilege specification

root ALL=(ALL:ALL) ALL

...

adding:

www-data ALL=NOPASSWD:/var/sudowebscript.sh, /var/show\_wifi\_connections.sh, /var/updatessid.sh

Save and exit.

* Install autohotspot

Download the AutoHotspot-Setup.tar.xz archive to the current folder using the command

curl "https://www.raspberryconnect.com/images/hsinstaller/Autohotspot-Setup.tar.xz" -o AutoHotspot-Setup.tar.xz

Unarchive the file to the curent folder using the command

tar -xvJf AutoHotspot-Setup.tar.xz

If you are using the Desktop then you can right click on the AutoHotspot-Setup.tar.xz file and select Extract Here

change directory to the Autohotspot folder with

cd Autohotspot

Run the script with the command

sudo ./autohotspot-setup.sh

This script will fail if sudo is not used.

You will presented with these menu option, choose option 2:

1 = Install Autohotspot with eth0 access for Connected Devices

2 = Install Autohotspot with No eth0 for connected devices

3 = Install a Permanent Access Point with eth0 access for connected devices

4 = Uninstall Autohotspot or permanent access point

5 = Add a new wifi network to the Pi (SSID) or update the password for an existing one.

6 = Autohotspot: Force to an access point or connect to WiFi network if a known SSID is in range

7 = Change the access points SSID and password

8 = Exit

Option 1: Install Autohotspot with eth0 access for Connected Devices

Once installed and after a reboot the Raspberry Pi will connect to a router that has previously been connected to and is listed in /etc/wpa\_supplicant/wpa\_supplicant.conf. If no router is in range then it will generate a WiFi access point.

This will have an SSID of RPiHotspot and password of 1234567890

Use option 7 to change the access point password and also the SSID if required

If an ethernet cable is connected to the Pi with access to the internet then it will allow devices connected to the access point to connect to the internet or local network.

Once a connection to the access point has been made you can access the Raspberry Pi via ssh & VNC with

ssh pi@192.168.50.5

vnc: 192.168.50.5::5900

for webservers use http://192.168.50.5/

Option 2: Install Autohotspot with No eth0 for connected devices

This option is similar to option 1 but connected devices have no network/internet connection if an ethernet cable is connected.

The Pi itself can use the eth0 connection and also be accessed from a device on the etho network.

This has been designed so you can access only the Pi from a Laptop, tablet or phone.

The access point SSID will be RPiHotspot with a password of 1234567890

Once a connection to the access point has been made you can access the Raspberry Pi via ssh & VNC with

ssh pi@10.0.0.5

vnc: 10.0.0.5::5900

for webservers use http://10.0.0.5/

Option 3: Install a Permanent Access Point with eth0 access for connected devices

This is for a permanent WiFi access point with network/internet access for connected devices.

The Raspberry Pi will only have network and internet access when an ethernet cable is connected.

Once a connection to the access point has been made, you can access the Raspberry Pi via ssh & VNC with

ssh pi@192.168.50.10

vnc: 192.168.50.10::5900

for webservers use http://192.168.50.10/

Additional setup is required if you wanted to use a second WiFi device to connect to the internet rather than a ethernet conection. This is a planned future option.

Option 4: Uninstall Autohotspot or Permanent Access Point

This will disable the setup of any of the three setups and return the Raspberry Pi to default Wifi settings.

Hostapd & dnsmasq will not be uninstalled just disabled. This is so a hotspot setup can be re-installed without access to the internet.

Option 5: Add a new wifi network to the Pi (SSID) or update the password for an existing one

If you are using either of the autohotspot setups in access point mode and wish to connect to a local WiFi network. You will be unable to scan for any networks as the desktop wifi option will be disabled, shown as red crosses. You can manually add the details to /etc/wpa\_supplicant/wpa\_supplicant.conf if you know them.

This option will allow you to scan for local WiFi networks and update the Pi. If you then reboot or use the Force... option 6 ,see below.

This option only works for WiFi networks where only a password is required. If a username is required this will not work.

Option 6: Autohotspot: Force to an access point or Force to WiFi network if a known SSID is in range

This option is only for the Autohotspot setups.

If you are at home and connected to your home network but would like to use the hotspot. This option will force the pi to access point mode and will ignore your home network untill the next reboot. If you use this option again while in access point mode, it will attempt to connect to a known WiFi network. This will go back to the access point if no valid WiFi network is found or there is a connection issue.

Option 7: Change the Pi's access point SSID and Password

By default the access point SSID is RPiHotSpot with a password of 1234567890. Use this option to change either or both SSID and Password.

You will be prompted to change both but if you make no entry and press enter the existing setting will be kept.

The password must be at least 8 characters.

Option 8: Exit

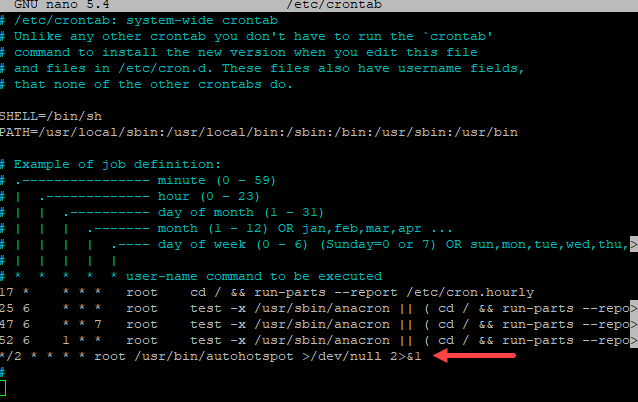
Exit the script.

Complete the Setup:

If you have installed a hotspot setup with options, 1,2 or 3 then Reboot to activate the setup.

To test the Hotspot mode for the Autohotspot setups; run the installer again and select option 6 to force the Pi into Hotspot mode. You can then test that everything is working ok.

When you have finished either choose option 6 again to reconnect to your router or reboot.



To enable automatic hotspot, the crontab must be edited. The system checks WLAN connect status every 2 minutes and turns on hotspot if WLAN is disconnected.

* Install Bluetooth modules to support Bluetooth Temp./Hum Sensor from Xiaomi
* sudo pip3 install bluepy requests
* sudo apt install bluetooth libbluetooth-dev
* sudo pip3 install pybluez pycryptodomex
* cd /opt
* sudo git clone <https://github.com/JsBergbau/MiTemperature2.git>
* copy MiCallback.sh from repository /opt/MiTemperature2 to Pi and enable execute rights for all
* sudo chmod +x MiCallback.sh
* copy my\_thermometer.txt from repository /opt/MiTemperature2 to Pi and enable write rights for all
* sudo chmod 666 my\_thermometer.txt
* Now copy all files and folders from your local git repository /var/www to /var/www/
* from local repository /opt/pi-ager/ to /opt/pi-ager/
* from local repository /var/sudowebscript.sh to /var/
* sudo chown –R www-data:www-data /var/www
* sudo usermod –G gpio –a www-data
* sudo chmod 666 /var/www/logs/logfile.txt
* sudo chmod 755 /var/www/logs/
* sudo chmod 664 /var/www/config/pi-ager.sqlite3
* sudo chown -R www-data:www-data /var/www/config/
* sudo chmod 777 /var/www/config/
* sudo chmod 555 /var/sudowebscript.sh
* from local repository /usr/local/bin/\*.sh copy all to /usr/local/bin/

(pi-ager\_backup.sh, pi-ager\_image.sh,setup\_pi-ager.sh)

Set +x mode to the scripts :

sudo chmod +x /usr/local/bin/\*.sh

* from local repository /lib/systemd/system copy the following files to

/lib/systemd/system/ :

pi-ager\_main.service

setup\_pi-ager.service

* from local repository /usr/bin copy the following file to /usr/bin/. This is a newer version of fswebcam with re-get frame on error.

fswebcam

Set +x mode to fswebcam:

sudo chmod +x /usr/bin/fswebcam

* from local repository /usr/share/man/man1/fswebcam.1.gz copy the following file to /usr/share/man/man1/

fswebcam.1.gz

* Enable setup\_pi-ager.service to initialize system with data from /boot/setup.txt after next reboot:

sudo systemctl enable setup\_pi-ager

sudo reboot