

## pi\_ager install on Raspberry Pi 3/Pi 4/Pi zero (2)w under Pi OS 12 Lite bookworm

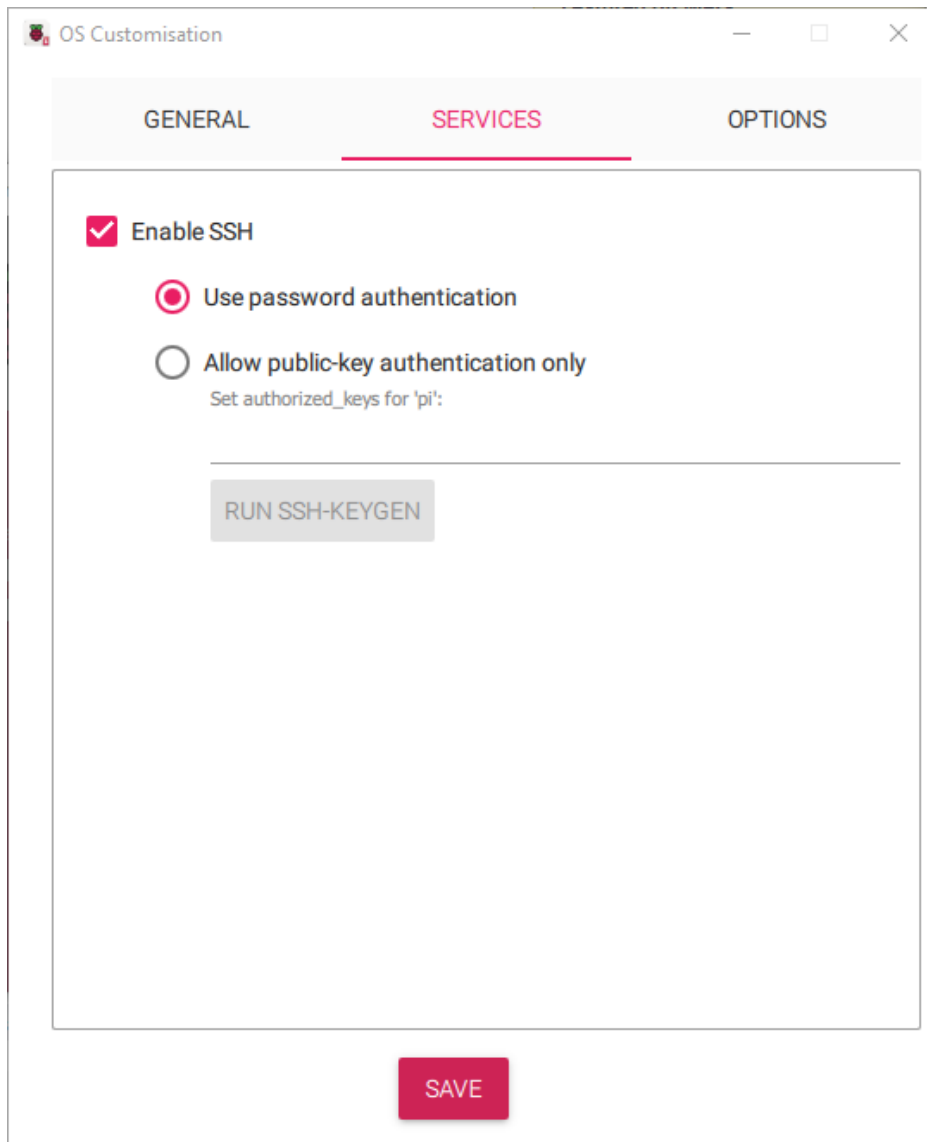
- **For all Pi:**  
Download and install Raspberry Pi Imager v1.8.4 or later.  
Start Raspberry Pi Imager, then select:  
Raspberry Pi Device:  
NO FILTERING  
Operating System:  
Raspberry Pi OS Lite (32-bit) ( A port of Debian Bookworm with no desktop environment)  
Storage:  
Use an USB Reader and choose your SD-Card, min. 8 GB  
Then click 'NEXT'  
In popup Window 'Use OS customisation' select 'EDIT SETTINGS'.  
In the 'GENERAL' Tab edit your settings:

The screenshot shows the 'OS Customisation' window with the 'GENERAL' tab selected. The window has three tabs: 'GENERAL', 'SERVICES', and 'OPTIONS'. The 'GENERAL' tab contains several settings:

- ☒ Set hostname: RaspberryPi.local
- ☒ Set username and password
  - Username: pi
  - Password: [masked]
- ☒ Configure wireless LAN
  - SSID: [masked]
  - Password: [masked]
  - ☐ Show password ☐ Hidden SSID
  - Wireless LAN country: DE
- ☒ Set locale settings
  - Time zone: Europe/Berlin
  - Keyboard layout: de

A red 'SAVE' button is located at the bottom center of the window.

In the 'SERVICES' Tab enable SSH:



Then 'SAVE' and apply OS customisation settings with 'YES'.

The customized OS is now written to your SD-Card.

Put your SD-Card in your Raspberry Pi and power on your Pi device.

Login via SSH (e.g. use PuTTY) or connect a HDMI monitor and USB keyboard to your Raspberry device and continue with the setup as described below.

- 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/firmware to support I2C and SPI devices:

**CAUTION:** If you want to edit config.txt ,cmdline.txt and setup.txt under Windows in a Terminal Window, e.g. using Terminal Apps like PuTTY, the Linux folder /boot/firmware on

the SD-Card is mapped to a windows partition FAT32 USB Drive named bootfs and contains the above mentioned files.

```
# 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]
enable_uart=1
dtoverlay=miniuart-bt
# force_turbo=1
```

- Add in /boot/firmware/cmdline.txt at the end of line this to enable USB camera with fswebcam :  
`dwc_otg.fiq_fsm_mask=0x3`
- Reboot system
- Edit /etc/modules to load i2c-dev at boot, add this line :  
`i2c-dev`
- Add file :  
`sudo touch /etc/modprobe.d/raspi-blacklist.conf`
- Install git: `sudo apt install git`
- Get a copy from Pi-Ager repository to your local system:  
`git clone -b master 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/firmware/`  
Create a symbolic link in /boot to /boot/firmware/setup.txt  
`cd /boot`  
`sudo ln -s /boot/firmware/setup.txt setup.txt`
- 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`

- Install php 8:  
`sudo apt install php-fpm php-cli`  
`sudo apt install php`  
`sudo apt install php-common php-sqlite3`
- Install additional modules for php:  
`sudo apt install php-mbstring php-zip php-curl`
- 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

To enable PHP8 in Lighttpd, we must modify `/etc/php/8.2/fpm/php.ini` and uncomment the line `cgi.fix_pathinfo=1`. In my `php.ini` file, I found it on line 807. To uncomment, just remove the semicolon in the beginning.

`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>
```

Change owner: `sudo chown www-data:www-data /var/www/test.html`

Enter your IP Address into the browser followed by `/test.html`

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

For 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 authentication 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
```

Add "broken-scriptfilename" and "x-sendfile" :

```
fastcgi.server += ( ".php" =>
    (
        "socket" => "/run/php/php-fpm.sock",
        "broken-scriptfilename" => "enable",
        "x-sendfile" => "enable"
    )
)
```

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 smbush  

```
sudo apt-get install python3-smbush
```
- Install sqlite3:  

```
sudo apt install sqlite3
```
- Install pip3  

```
sudo apt install python3-pip
```

**CAUTION :**

```
sudo pip3 install <module>
```

 no longer supported.

If you see an error message, try to install the module with  

```
sudo apt install python3-<module name>
```

. If this does not work, try install modules with  

```
sudo pip3 install <module name> --break-system-packages
```

```
error: externally-managed-environment
× This environment is externally managed
└> To install Python packages system-wide, try apt install
    python3-xyz, where xyz is the package you are trying to
    install.
```

If you wish to install a non-Debian-packaged Python package,  
create a virtual environment using `python3 -m venv path/to/venv`.  
Then use `path/to/venv/bin/python` and `path/to/venv/bin/pip`. Make  
sure you have `python3-full` installed.

If you wish to install a non-Debian packaged Python application,  
it may be easiest to use `pipx install xyz`, which will manage a  
virtual environment for you. Make sure you have `pipx` installed.

See `/usr/share/doc/python3.11/README.venv` for more information.  
note: If you believe this is a mistake, please contact your Python  
installation or OS distribution provider. You can override this, at the

risk of breaking your Python installation or OS, by passing `--break-system-packages`.

hint: See PEP 668 for the detailed specification.

- 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 cryptography  
`sudo apt install python3-cryptography`
- Install uuidgen  
`sudo apt install uuid-runtime`
- 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 Display 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`
- Install schedule  
`sudo pip3 install schedule`
- Install MQTT client  
`sudo apt install python3-paho-mqtt`
- Workaround for Adafruit\_DHT for Pi4:  
In `"/usr/local/lib/python3.11/dist-packages/Adafruit_DHT/platform_detect.py"`, you can add/modify some lines, so it should workaround the issue, that GPIO Control does not work.

```
79
80 def pi_version():
81     """Detect the version of the Raspberry Pi. Returns either 1, 2, 3, 4, 5 or
82     None depending on if it's a Raspberry Pi 1 (model A, B, A+, B+),
83     Raspberry Pi 2 (model B+), Raspberry Pi 3, Raspberry Pi 3 (model B+), Raspberry Pi 4,
84     Raspberry Pi 5 or not a Raspberry Pi.
85     """
86     # Check /proc/cpuinfo for the Hardware field value.
87     # 2708 is pi 1
88     # 2709 is pi 2
89     # 2835 is pi 3
90     # 2837 is pi 3b+
91     # 2711 is pi 4
92     # 2712 is pi 5
93     # Anything else is not a pi.
94     with open('/proc/cpuinfo', 'r') as infile:
95         cpuinfo = infile.read()
96         # Match a line like 'Hardware      : BCM2709'
97         match = re.search('^Hardware\s+:\s+(\w+)\$', cpuinfo,
98                           flags=re.MULTILINE | re.IGNORECASE)
99     if not match:
100         # Couldn't find the hardware, assume it isn't a pi.
101         return None
102     if match.group(1) == 'BCM2708':
103         # Pi 1
104         return 1
105     elif match.group(1) == 'BCM2709':
106         # Pi 2
107         return 2
108     elif match.group(1) == 'BCM2835':
109         # Pi 3
110         return 3
111     elif match.group(1) == 'BCM2837':
112         # Pi 3b+
113         return 3
114     elif match.group(1) == 'BCM2711':
115         # Pi 4 works same as 3b+
116         return 3
117     elif match.group(1) == 'BCM2712':
118         # Pi 5 not working
119         return 5
120     else:
121         # Something else, not a pi.
122         return None
123
```

- Unblock wifi for Pi4, add rfkill unblock wifi and disable power management for wlan0:  
`cd /etc`



sudo nano rc.local

```
GNU nano 7.2 rc.local
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

# disable power management for wlan0 to increase WLAN reliability
rfkill unblock wifi
sleep 10
iwconfig wlan0 power off

# enable AP-STA mode
iw dev wlan0 interface add wlan1 type __ap
sleep 2

# enable captive portal
nodosplash

exit 0
```

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

```
pi@pi-ager:~$ sudo crontab -l
#-----
# Shell variable for cron
SHELL=/bin/bash
# PATH variable for cron
PATH=/usr/local/bin:/usr/local/sbin:/sbin:/usr/sbin:/bin:/usr/bin:/usr/bin/X11
#M S T M W Befehl
#-----
#* */* * * * /usr/bin/flock -w 0 /var/run/pi-ager_backup.pid /usr/local/bin/pi-ager_backup.sh >> /var/log/pi-ager_backup
.log 2>&1
#*/2 * * * * logger "greetings from the crontab" > /dev/null 2>&1
```

- 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,/usr/bin/raspi-config,/usr/bin/nmcli`

Save and exit.

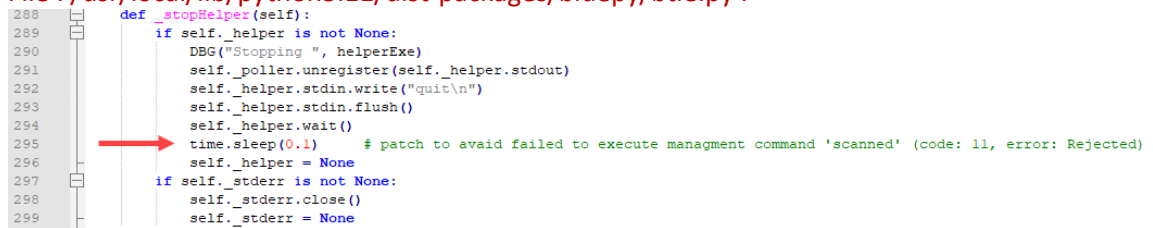
- Install access point with network manager:  
 Copy from repository /usr/local/bin/setup\_pi-ager-ap.sh to destination.  
 Create virtual interface wlan1:  
`sudo iw dev wlan0 interface add wlan1 type __ap`  
 then run script :  
`cd /usr/local/bin`  
`sudo ./setup_pi-ager-ap.sh`
- Install nodogsplash captive portal  
`sudo apt install iptables`  
`sudo apt install libmicrohttpd-dev`  
`cd ~`  
`git clone https://github.com/nodogsplash/nodogsplash.git`  
`cd ./nodogsplash`  
`make`  
`sudo make install`  
 copy all from repository /etc/nodogsplash to /etc/nodogsplash  
`sudo chmod +x /usr/bin/nodogsplash`  
`sudo chmod +x /usr/bin/ndsctl`  
`sudo nano /etc/rc.local`  
 Before exit at the end of file insert the following lines :  
  
`# start pi-ager`  
`systemctl start pi-ager_main.service`  
  
`# disable power management for wlan0 to increase WLAN reliability`  
`rkill unblock wifi`  
`sleep 1`  
`# iwconfig wlan0 power off`  
`iw wlan0 set power_save off`  
  
`# enable AP-STA mode`  
`iw dev wlan0 interface add wlan1 type __ap`  
`sleep 1`  
  
`iw wlan1 set power_save off`  
  
`# enable captive portal`

nodogsplash

exit 0

- Install Bluetooth modules to support Bluetooth Temp./Hum Sensor from Xiaomi
- `sudo apt install libglib2.0-dev`
- `sudo pip3 install bluepy requests`
- `sudo apt install bluetooth libbluetooth-dev`
- `# sudo pip3 install pybluez pycryptodomex`
- Patch btle.py to stop intermittend errors:

File : /usr/local/lib/python3.11/dist-packages/bluepy/btle.py :



```
288 def _stopHelper(self):
289     if self._helper is not None:
290         DBG("Stopping ", helperExe)
291         self._poller.unregister(self._helper.stdout)
292         self._helper.stdin.write("quit\n")
293         self._helper.stdin.flush()
294         self._helper.wait()
295         time.sleep(0.1) # patch to avoid failed to execute management command 'scanned' (code: 11, error: Rejected)
296         self._helper = None
297     if self._stderr is not None:
298         self._stderr.close()
299         self._stderr = None
```

- 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 /var/updatessid.sh /var/show_wifi_connections.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`  
Then activate these services:  
`sudo systemctl daemon-reload`
- 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