

Display Panel

Install

Wegen zu vielen Problemen mit Wayland wird hier **Trixie mit X11** genutzt.

Das Hauptproblem ist, dass das Display beim Stromsparen einen nervigen Bildschirm anzeigt ... "No Signal" ...

Einstellungen

- `sudo raspi-config`
 - I2C Aktivieren
 - umstellen von Wayland auf X11
 - Screen Blanking aus !

100 Mbit Fix

- Man muss dafür per SSH mit WLAN auf dem Pi sein
- `sudo nano /etc/systemd/system/eth0fix.service`

```
[Unit]
Description=100MBit Fix for eth0
After=network-online.target
Requires=network-online.target

[Service]
ExecStart=/sbin/ethtool -s eth0 speed 100 duplex full autoneg on
Type=oneshot

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

- `sudo systemctl enable eth0fix.service`
- `sudo systemctl restart eth0fix.service`

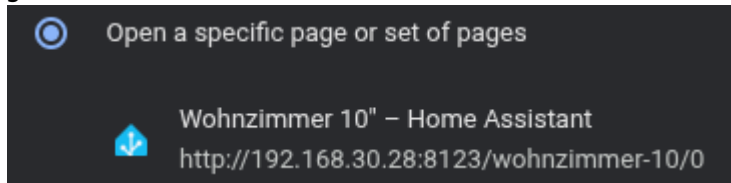
Updates

- Command Line
`sudo apt update && sudo apt upgrade -y && sudo apt install -y git git-lfs silversearcher-ag wavemon hexedit sudoku tcpdump iptraf mc htop dcfldd nano usbutils ranger ncd u can-utils multitail fd-find lsof x11vnc minicom joystick i2c-tools speedtest-cli iotop fio ir-keytable curl inxi lshw nala fzf duf`

- UI Tools
sudo apt update && sudo apt upgrade -y && sudo apt install -y flameshot terminator cutecom jstest-gtk xclip
- Nutzloses Zeug
sudo apt autoremove -y modem* cups* avahi* triggerhappy* rpi-connect openvpn xscreensaver

Browser

- Settings



<http://192.168.30.28:8123/wohzimmer-10/0>



Damit ist dann auch das "Restore Previous" weg beim Start!

- Appearance → Mode → Dark

... steuern

falls benötigt ...

```
# Chromium in Fullscreen
sleep 15
#WID=$(DISPLAY=:0 xdotool search --onlyvisible --class chromium|head -1)
#DISPLAY=:0 xdotool windowactivate ${WID}
#DISPLAY=:0 xdotool key F11

# An Display Position navigieren mit der Maus und Vollbild auslösen
DISPLAY=:0 /usr/bin/xdotool mousemove --sync 954 200 click 1
```

Autostart

- Alles in die Autostart packen ist keine gute Option. xset wird nicht sauber gestartet.
- nano /home/pi/start_kiosk.sh

[start_kiosk.sh](#)

```
#!/bin/bash

# Warte kurz, bis der Desktop vollständig geladen ist (WICHTIG!)
sleep 5
```

```
# Energieeinstellungen deaktivieren
xset s off
xset s noblank
xset -dpms

# Chromium Crash-Flags bereinigen
# 1. Sperrdateien löschen (WICHTIG!)
# Verhindert die Meldung "Profil wird bereits verwendet"
rm -f /home/pi/.config/chromium/SingletonLock
rm -f /home/pi/.config/chromium/Default/SingletonLock

# 2. 'Local State' säubern (Globaler Status)
sed -i 's/"exited_cleanly":false/"exited_cleanly":true/'
/home/pi/.config/chromium/'Local State'

# 3. 'Preferences' säubern (Profil Status)
# Dieser eine Befehl erschlägt "Crashed", "SessionCrashed" und
andere Zustände auf einmal
sed -i
's/"exit_type":"[^"]\+"/"exit_type":"Normal"/;s/"exited_cleanly":f
alse/"exited_cleanly":true/'
/home/pi/.config/chromium/Default/Preferences

# Chromium starten (Kiosk Modus als Beispiel, ansonsten deine
Flags nutzen)
chromium --noerrdialogs --touch-events=enabled
```

- Unter Debian Trixie (X11) liegt die autostart dann hier :
sudo nano /etc/xdg/lxsession/rpd-x/autostart

autostart

```
@lxpanel-pi
@pcmanfm-pi
@bash /home/pi/start_kiosk.sh
```

Sonst liegt die Autostart hier /home/pi/.config/lxsession/LXDE-pi/autostart

- chmod +x /home/pi/start_kiosk.sh
- Nach einem Restart kann man xset -q prüfen ob Screenblinking und Screensaver Timeout deaktiviert sind:

```
pi@SHome-Display:~ $ xset -q
...
Screen Saver:
  prefer blanking: no    allow exposures: yes
  timeout: 0    cycle: 600
...
```

X11VNC

- `sudo apt install x11vnc`
- für einen ersten Test kann man das verwenden
`x11vnc -usepw -forever -display :0`
- **Einrichtung als Dienst**
- `sudo x11vnc -storepasswd /etc/x11vnc.pass`
- `sudo nano /lib/systemd/system/x11vnc.service`

```
[Unit]
Description=Start X11VNC
After=multi-user.target

[Service]
Type=simple
ExecStart=/usr/bin/x11vnc -display :0 -auth guess -forever -loop -
noxdamage -repeat -rfbauth /etc/x11vnc.pass -rfbport 5900 -shared

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

- `sudo systemctl enable x11vnc.service`

Fullscreen

- für Fullscreen Kontroller brauchen wir die HA Integration “Browser Mod” (Installieren über HACS)
https://github.com/thomasloven/hass-browser_mod
- auf dem Endgerät dann die Seite öffnen
<http://192.168.30.28:8123/browser-mod>
 - Register aktivieren
 - sinnvollen Namen in Browser ID eingeben `WZ_Display_10Zoll`
- Jetzt sollte der Browser als Device in HA auftauchen
- Wir brauchen BrowserMod um JavaScript auszuführen um Fullscreen Toggel hinzubekommen
- Das Ausblenden von HA Elementen passiert mit `kiosk-mode`
<https://github.com/NemesisRE/kiosk-mode>
- Konfig vom `kiosk-mode` passiert im YAML Modus des Dashboards!

```
kiosk_mode:
  user_settings:
    - users:
      - wohnzimmer
        hide_sidebar: true
        hide_header: false
        hide_assistant: true
```

Das muss am Anfang ergänzt werden. Damit wird für den User `wohnzimmer` die Sidebar und der Assistant abgeschaltet.

Display

- Venv einrichten
 - `sudo apt -y update && sudo apt-get install -y build-essential cmake make gcc pkg-config python3-dev python3-virtualenv libsystemd-dev libopenblas-dev`
 - `cd ~ && mkdir -p DisplayOn0ff && cd DisplayOn0ff`
 - `python -m venv . && source bin/activate`
 - `pip3 install --upgrade pip && pip3 install RPi.GPIO pynput smbus paho-mqtt psutil`
- Compile VL53 Lib ...
 - `sudo apt-get install build-essential python-dev`
 - `cd ~`
 - `git clone https://github.com/pimoroni/VL53L0X_rasp_python.git`
 - `cd VL53L0X-python`
 - `make`
 - VL53L0X.py in den Scriptordner kopieren ...
- Tests starten ...

Tests

VL

Motion Detect

- Sensor : VL53L0X
- Altes Script

```
import sys
import time
import VL53L0X
import os
import subprocess
import RPi.GPIO as GPIO
from pynput import mouse

def on_move(x, y):
    global Display_Last_Action
    print('Pointer moved to {0}'.format((x, y)))
    sys.stdout.flush()
    Display_Last_Action = time.time()
    print("Time %d" % (Display_Last_Action))

def on_click(x, y, button, pressed):
    global Display_Last_Action
    print('{0} at {1}'.format('Pressed' if pressed else 'Released', (x, y)))
```

```
sys.stdout.flush()
Display_Last_Action = time.time()
print("Time %d" % (Display_Last_Action))

def on_scroll(x, y, dx, dy):
    global Display_Last_Action
    print('Scrolled {0} at {1}'.format('down' if dy < 0 else 'up', (x,
y)))
    sys.stdout.flush()
    Display_Last_Action = time.time()
    print("Time %d" % (Display_Last_Action))

def toggleOnOff():
    GPIO.output(36, GPIO.HIGH)
    time.sleep(0.4)
    GPIO.output(36, GPIO.LOW)
    print("Toggle Switch ....")
    sys.stdout.flush()

listener = mouse.Listener(
    on_move=on_move,
    on_click=on_click,
    on_scroll=on_scroll)
listener.start()

# Last Time Event happend on Display
Display_Last_Action = time.time()
Display_State = True
Display_Timeout_Sec = 180

# RPi.GPIO Layout verwenden (wie Pin-Nummern)
GPIO.setmode(GPIO.BOARD)
# RuntimeWarning: This channel is already in use, continuing anyway.
# Use GPIO.setwarnings(False) to disable warnings.
GPIO.setwarnings(False)

# Pin 36 (GPIO 16) auf Output setzen -> Display ON / OFF
GPIO.setup(36, GPIO.OUT)
# Pin 38 (GPIO 20) auf Output setzen -> Brightness +
GPIO.setup(38, GPIO.OUT)
# Pin 40 (GPIO 21) auf Output setzen -> Brightness -
GPIO.setup(40, GPIO.OUT)

# Create a VL53L0X object
tof = VL53L0X.VL53L0X(i2c_bus=1,i2c_address=0x29)
tof.open()
tof.start_ranging(VL53L0X.VL53L0XAccuracyMode.BETTER)

while 1:
    # Distanz vom VL53L0X Sensor messen
```

```

distance = tof.get_distance()
#print("%d mm, %d cm" % (distance, (distance/10)))

# Display ggf. abschalten
if time.time()-Display_Last_Action >= Display_Timeout_Sec:
    if Display_State == True:
        #subprocess.call("DISPLAY=:0 xset dpms force off",
shell=True)
        toggleOnOff()
        Display_State = False
        print("Display wird abgeschaltet ...")
        sys.stdout.flush()

# Abstand vom Sensor pruefen
if distance < 900:
    print("DETECTED, %d cm , %d sec" % (distance/10, time.time()-
Display_Last_Action ))
    sys.stdout.flush()
    Display_Last_Action = time.time()
    print("Time %d" % (Display_Last_Action))

# Display ggf. wieder einschalten
if time.time()-Display_Last_Action < Display_Timeout_Sec:
    if Display_State == False:
        #subprocess.call("DISPLAY=:0 xset dpms force on",
shell=True)
        toggleOnOff()
        Display_State = True
        print("Display wird eingeschaltet ...")
        sys.stdout.flush()

# Display Einschalten
#os.system('DISPLAY=:0 xdotool key "shift"')
###subprocess.call("DISPLAY=:0 xset s reset", shell=True)
#subprocess.call("DISPLAY=:0 xset dpms force on", shell=True)
#subprocess.call("DISPLAY=:0 xset dpms 120 120 120",
shell=True)

time.sleep(0.1)

tof.stop_ranging()
tof.close()

```

- Service

```

pi@SHome-Display-VPN-DNS:/ $ cat lib/systemd/system/display.service
[Unit]
Description=Display Controller

[Service]
Environment=DISPLAY=:0

```

```
Environment=XAUTHORITY=/home/pi/.Xauthority
ExecStart=/usr/bin/python /home/pi/DisplayOn/MotionDetect.py
Restart=always
RestartSec=30s
KillMode=process
TimeoutSec=infinity
```

[Install]

```
WantedBy=graphical.target
```

From:
<https://www.drklipper.de/> - **Dr. Klipper Wiki**

Permanent link:
https://www.drklipper.de/doku.php?id=haussteuerung:display_panel&rev=1766814211

Last update: **2025/12/27 06:43**

