Dynaframe Pro

- Buttons
  - Display on/off - shutdown (deutsch)
  - Display on/off - shutdown (englisch)

Buttons

Buttons

Display on/off - shutdown (deutsch)

https://github.com/Archonw/Dynaframe-Stuff/blob/main/scripts/display-on-off-shutdown.py

Von dieser Github Seite wird der Code verwendet, mit dessen Hilfe das HDMI-Sginal des Pi's ab- und angeschaltet werden kann, sowei bei langen Drücken der Pi herunter fährt.

Nachdem der Button am Pi angschlossen ist kann es mit der Software losgehen. Dafür wird ein Python-Script verwendet.

Im ersten Schritt erstellen wir auf dem Pi einen neuen Ordner "scripts" in dem wir dann eine Datei "button.py" anlegen werden welche wir dann im zweiten Teil über die "rc.local" beim Systemstart aufrufen lassen.

1. Teil

Der Raspberry Pi ist an und im Netwzerk angemeldet, so dass wir auf ihn über unser Netzwerk sehen und aufrufen können. Haben wir bei der Erst-Einrichtung des Pi den Namen nicht verändert, so finden wir ihn unter dem Namen DYNAFRAMEPRO.

Netzwerk.png

Schauen wir welche Ordner nun auf unserem Pi angezeigt werden so sehen wir unter anderem den Ordner "assets".

In diesen navigieren wir hinein.

assets.png

Hier sind einige Ordner enthalten. Wir erstellen hier einen neuen Ordner "scripts" und navigieren und den neunen Ordner hinein.

scripts.png

In dem noch leeren Ordner erstellen wir eine Datei mit dem Namen "button.txt" und öffnen diese.

button-txt.png

Hier kopieren wir folgenden Code hinein.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from gpiozero import Button
from subprocess import check_call
from signal import pause
import os
import requests

Button.was_held = False
button_pressed_time = None

def get_display_power_status():
    result = os.popen('vcgencmd display_power').read()
    return int(result.split('=')[1])

def toggle_display():
    current_status = get_display_power_status()
    if current_status == 1:
        response = requests.get("http://127.0.0.1/command/?COMMAND=SCREENSTATE&VALUE=false")
        print("Display off")
    else:
        response = requests.get("http://127.0.0.1/command/?COMMAND=SCREENSTATE&VALUE=true")
        print("Dislpay on")

def shutdown():
    print("Shutdown function called")
```

```
    os.system('sudo shutdown now')

def held(btn):
    btn.was_held = True
    print("Button was held, not just pressed")
    shutdown()

def released(btn):
    if not btn.was_held:
        toggle_display()
    btn.was_held = False

btn = Button(19, hold_time=4)  # Set your GPIO number and hold time here

btn.when_held = held
btn.when_released = released

# Warten
pause()
```

Wichtig: hier muss in Zeile 40 der Pin angegeben werden, an dem der Button am Pi angeschlossen wurde. In diesem Fall ist es der Pin 19.

Die fertige Datei sollte dann so aussehen:

button anpssen.png

Diese können wir nun speichern und schließen.

Jetzt müssen wir sie noch kurz umbenennen in "button.py"

button anlegen.png

Mit diesem Fenster sind wir fertig und können es nun auch schließen.

2. Teil

Wir haben nun die erforderliche Datei erstellt damit wir ein Python-Script beim start aufrufen und im Hintergrund laufen lassen können. Jetzt müssen wir noch dafür sorgen, dass dieses Script auch bei Systemstart aufgerufen wird. Dazu gibt es einige Wege; wir werden hier den Weg über eine

Datei die "rc.local" nehmen.

Dazu müssen wir uns entweder per ssh auf den Pi verbinden, rufen über eine angeschlossene Tastatur und Maus ein Terminal auf oder verbinden uns über VNC auf den Pi und rufen dann ein Terminal am Pi auf.

Haben wir das gemacht geben wir folgenden Befehl im Terminal ein, um einen Editor zu starten, der uns die Datei rc.local öffnet, damit wir sie bearbeiten können.

```
sudo nano /etc/rc.local
```

rc.local.png

Wir fügen diese Zeile kurz vor dem Ende ein.

```
/usr/bin/python3 /home/pi/Dynaframe/Assets/scripts/button.py
```

So sollte die Datei jetzt aussehen.

rc.local-assets.png

Um den Editor zu schließen und die Datei zu speichern drücken wir "Strg+X" und danach "Y"

Jetzt fehlt nur noch ein Neustart des Pi und der Button ist einsatzbereit.

Buttons

Display on/off - shutdown (englisch)

https://github.com/Archonw/Dynaframe-Stuff/blob/main/scripts/display-on-off-shutdown.py

From this Github page the code is used, with the help of which the HDMI signal of the Pi can be switched off and on, as well as the Pi shuts down during long presses.

After the button is connected to the Pi, you can start with the software. A Python script is used for this.

In the first step, we will create a new folder "scripts" on the Pi, in which we will then create a file "button.py", which we will then call up in the second part via the "rc.local" at system startup.

Part. 1

The Raspberry Pi is connected to and logged into the network, so we can view and access it via our network. If we didn't change the name when we first set up the Pi, we find it under the name DYNAFRAMEPRO.

Netzwerk.png

If we look at which folders are now displayed on our Pi, we can see, among other things, the folder "assets".

We navigate into it.

assets.png

Some folders are included here. We create a new folder "scripts" here and navigate and the new folder into it.

scripts.png

In the folder that is still empty, we create a file called "button.txt" and open it.

button-txt.png

Here we copy the following code.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from gpiozero import Button
from subprocess import check_call
from signal import pause
import os
import requests

Button.was_held = False
button_pressed_time = None

def get_display_power_status():
    result = os.popen('vcgencmd display_power').read()
    return int(result.split('=')[1])

def toggle_display():
    current_status = get_display_power_status()
    if current_status == 1:
        response = requests.get("http://127.0.0.1/command/?COMMAND=SCREENSTATE&VALUE=false")
        print("Display off")
    else:
        response = requests.get("http://127.0.0.1/command/?COMMAND=SCREENSTATE&VALUE=true")
        print("Dislpay on")

def shutdown():
    print("Shutdown function called")
    os.system('sudo shutdown now')

def held(btn):
```

```
    btn.was_held = True
    print("Button was held, not just pressed")
    shutdown()

def released(btn):
    if not btn.was_held:
        toggle_display()
    btn.was_held = False

btn = Button(19, hold_time=4)  # Set your GPIO number and hold time here

btn.when_held = held
btn.when_released = released

# Warten
pause()
```

Important: here the pin to which the button was connected to the Pi must be specified in line 40. In this case, it is pin 19.

The finished file should then look like this:

button anpssen.png

We can now save and close them.

Now we have to rename it to "button.py"

button anlegen.png

We are done with this window and can now close it.

Part 2

We have now created the necessary file so that we can call a Python script at startup and run it in the background. Now we have to make sure that this script is also called at system startup. There are a few ways to do this; we will take the path here via a file called "rc.local".

To do this, we either have to connect to the Pi via ssh, call up a terminal via a connected keyboard and mouse, or connect to the Pi via VNC and then call up a terminal on the Pi.

Once we have done this, we enter the following command in the terminal to start an editor that will open the rc.local file so that we can edit it.

```
sudo nano /etc/rc.local
```

rc.local.png

We insert this line just before the end.

```
/usr/bin/python3 /home/pi/Dynaframe/Assets/scripts/button.py
```

This is what the file should look like now.

rc.local-assets.png

To close the editor and save the file we press "Ctrl+X" and then "Y"

Now all that's missing is a restart of the Pi and the button is ready for use.