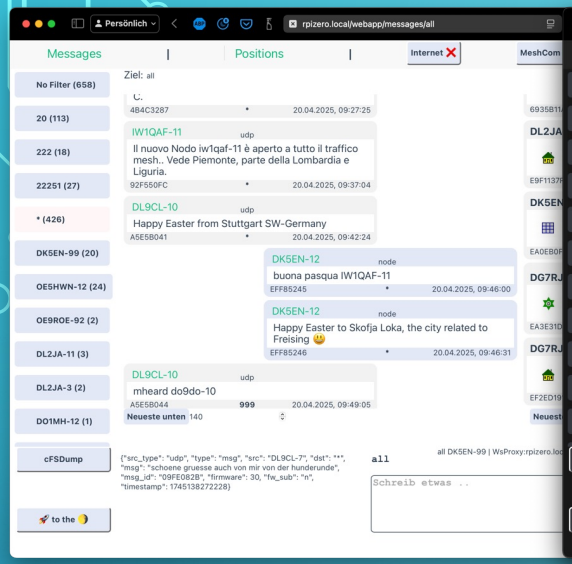
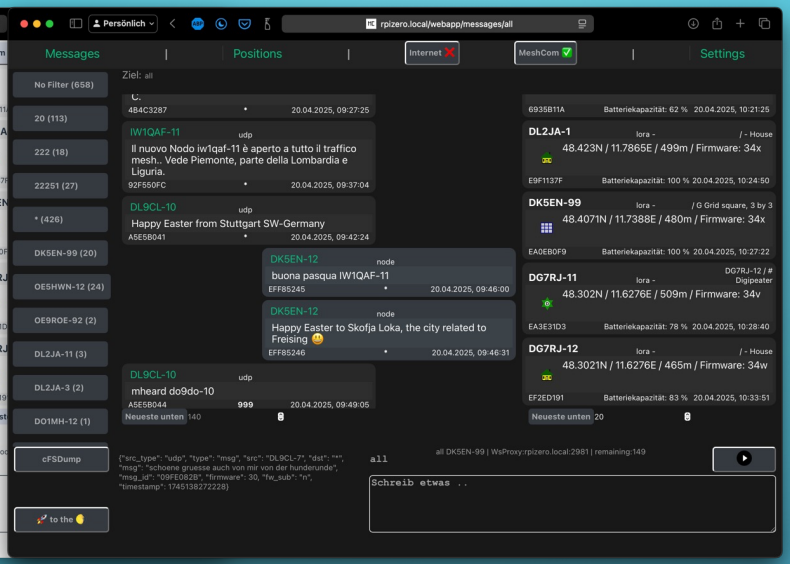
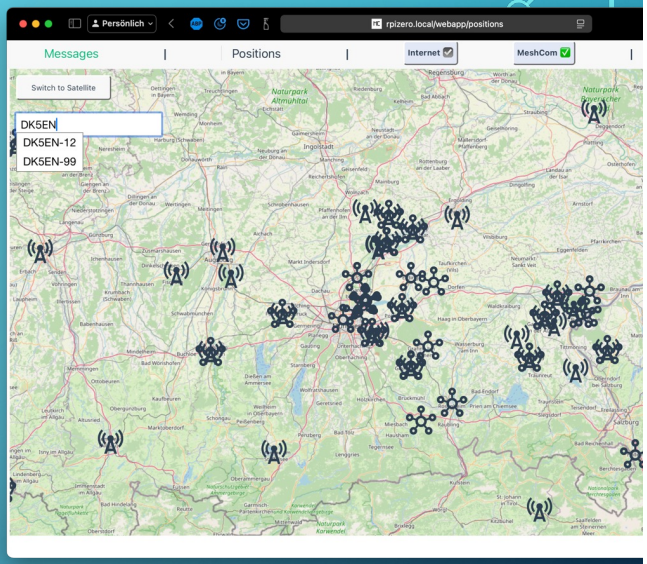
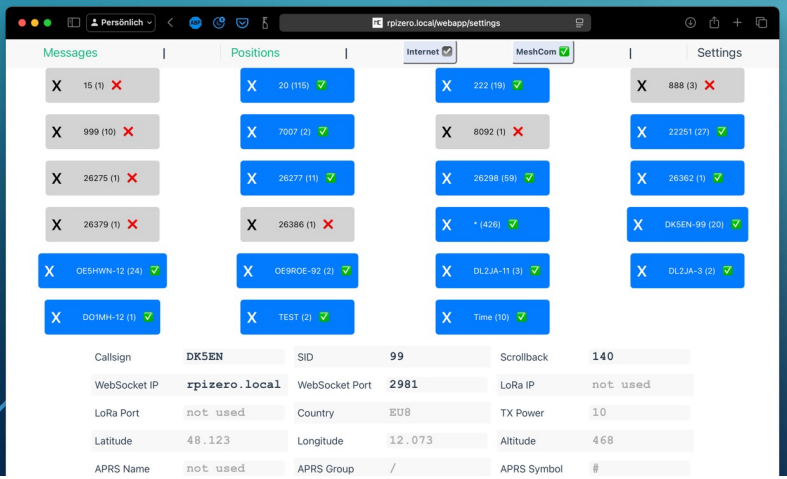
# MESHCOM MCAPP INSTALLATION

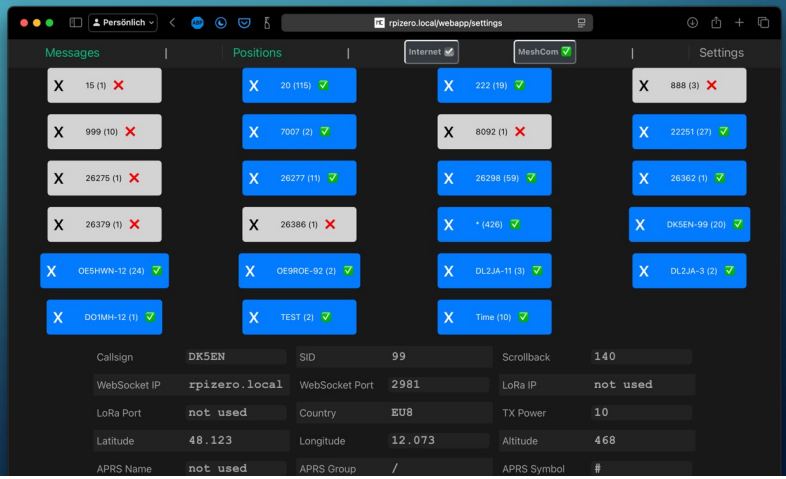
# SCREEN SHOTS – RESPONSIVE UI WITH DARK MODE



# VORWORT

- Die Kommunikation zwischen Webbrowser am PC und Server Komponente am Raspi Zero kann nur TLS-verschlüsselt erfolgen, weil die modernen Browser dies erzwingen. Dies ist kein unnötiger Luxus, sondern notwendig
- Offizielle Zertifikate werden von Let's Crypt nur für offizielle Domains ausgestellt. Wenn man mit mDNS „.local" (auf MacOS oder Windows mit iTunes installiert) oder Fritz!Box .fritz.box arbeitet, dann gibt es keine SSL-Zertifikate, die gegen ein getrustete Root-Zertifikat laufen, die im Browser und Betriebssystem vorinstalliert sind
- Jedoch ist die für den geübten Admin kein Problem, denn Caddy bringt eine PKI mit Zertifikatsrotation mit sich. x.509 Zertifikate sind trotzdem komplex
- Daher müssen das self-signed Root Zertifikat der Caddy PKI importieren

- Da bei SSL-Zertifikaten immer der Hostname überstimmen muss, ist es nicht möglich mit IP-Adressen zwischen Webbrowser und Raspi im lokalen Netz zu arbeiten. Es muss alles zwingend über DNS-Namen laufen, die auch dem cn= Eintrag im Zertifikat entsprechen müssen
- Die Serverkomponente ist ein Python Script, das die Messages per UDP mit dem MeshCom Node austauscht und alles über einen websocket weiterleitet. Der WebSocket wird TLS verschlüsselt durch Caddy, unseren Reverse Proxy
- Der lighttpd Webserver wird ebenso durch Caddy TLS verschlüsselt. Die Webseite selbst ist statisch, es wird kein PHP benötigt
- Wer seinen MeshCom Knoten nur über IP-Adresse erreicht, kann dies im `C2-mc-ws.py` Skript entsprechend anpassen.
- Zum Abschluss nicht vergessen auf dem MeshCom Knoten `--extudp on` zu konfigurieren und einzuschalten

Happy Meshing in MeshCom de DK5EN

# UPDATING TO THE LATEST VERSION

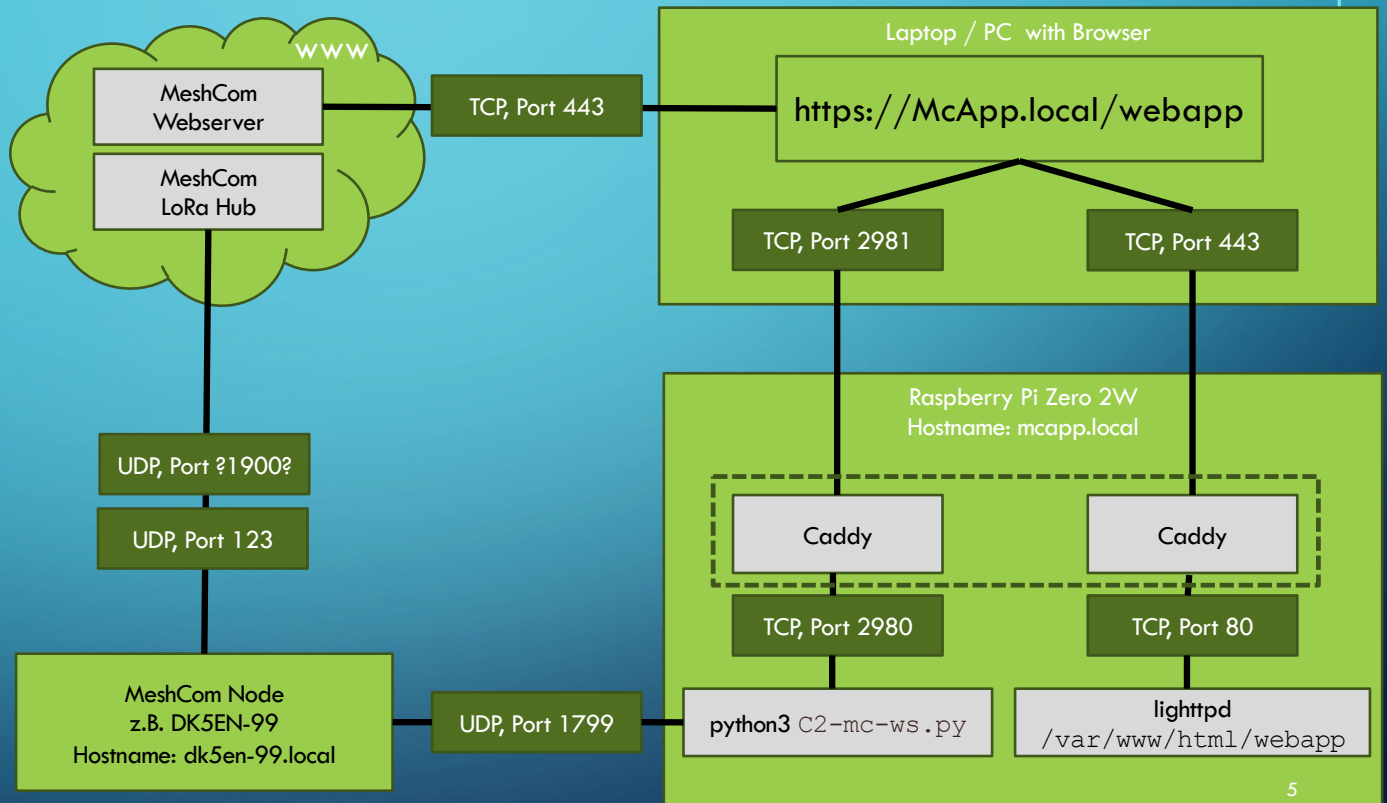
- To install or update everything, just use the script

```
martin@McApp:~ $ curl -fsSL https://raw.githubusercontent.com/DK5EN/McAdvChat/main/mc-install.sh | sudo bash

[INFO] Skript läuft unter Benutzer: martin
[INFO] Lokale WebApp-Version: v0.1.0
[INFO] Lokale Python-Skript-Version: v0.2.0
[INFO] Lokale Shell-Skript-Version: v0.2.0
[INFO] Install-Skript-Version: v0.1.0
[INFO] Remote WebApp-Version: v0.1.0
[INFO] Remote Python-Skript-Version: v0.2.0
[INFO] Remote Shell-Skript-Version: v0.2.0
[INFO] Reloade Webserver ...
[INFO] Prüfe WebApp unter https://rpiZero.local/webapp/version.txt
[INFO] WebApp erfolgreich aktualisiert auf Version v0.1.0
[INFO] Installations-Skript erfolgreich abgeschlossen.
```

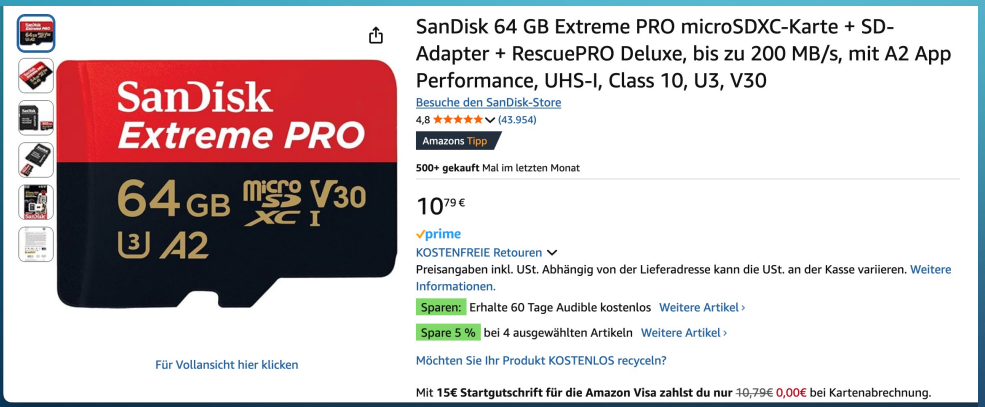
# HIGH LEVEL ARCHITECTURE

- High level overview of how everything is tied together



# SD CARD & RPI ZERO

- Insert your at least 32GB SD Card into your card reader
- Please only use SD Cards with 100MBit/s like SanDisk
- Have your Raspberry Pi Zero 2W at Hand

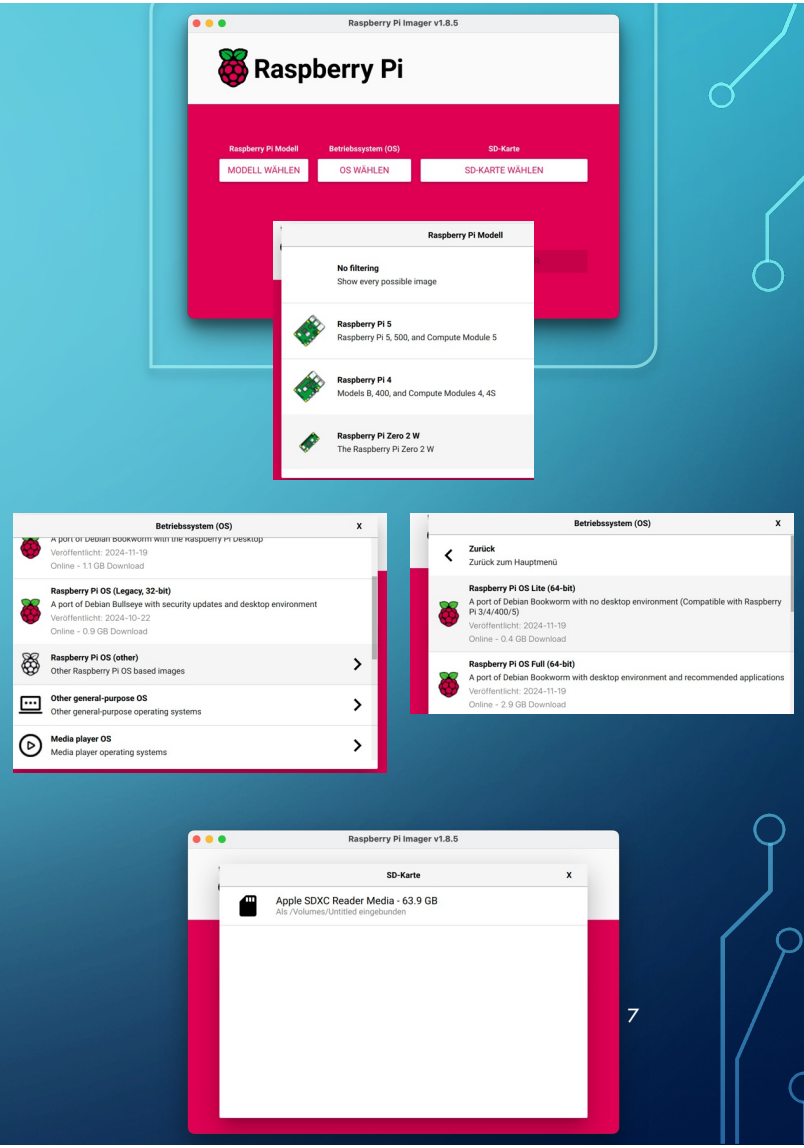



# CUSTOM OS INSTALLATION

We want to install a headless, 64Bit Debian Bookwork

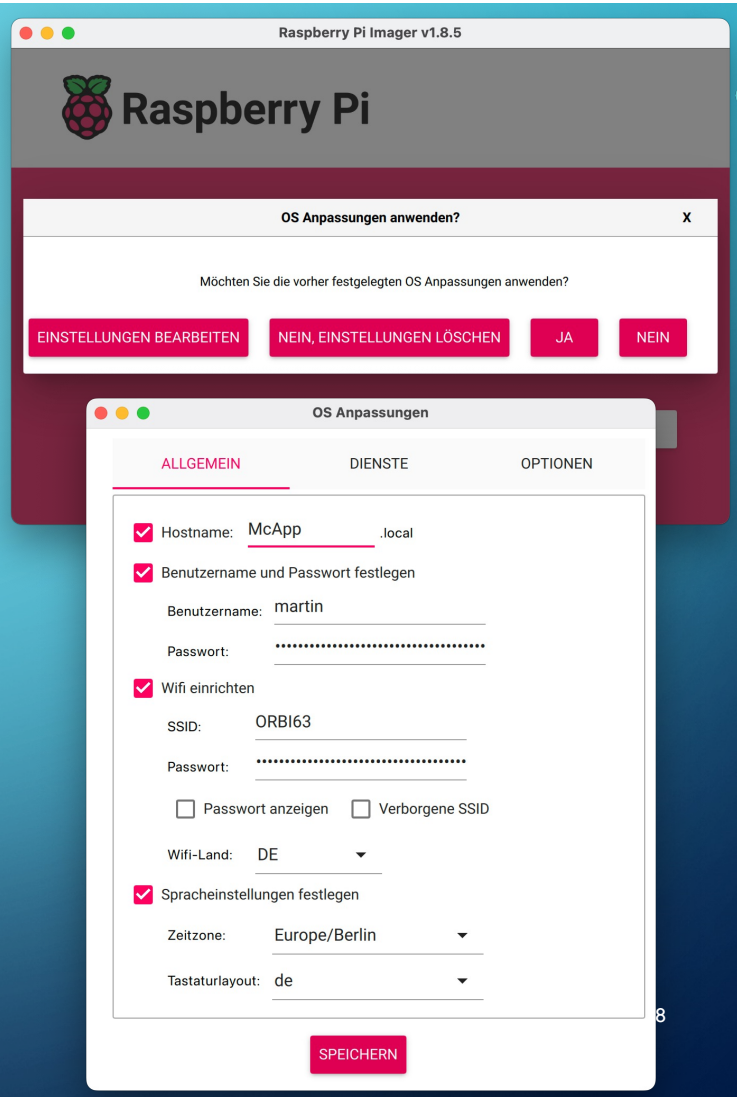
- I do recommend Raspberry PI Imager, which can be found here https://www.raspberrypi.org/downloads/
- Select your Model: Raspberry Pi Zero 2 W
- For OS, select “other“ – „Raspberry Pi OS Lite (64-bit), with no desktop, approx. 0,4GB
- Select your SD Card

.. And click next



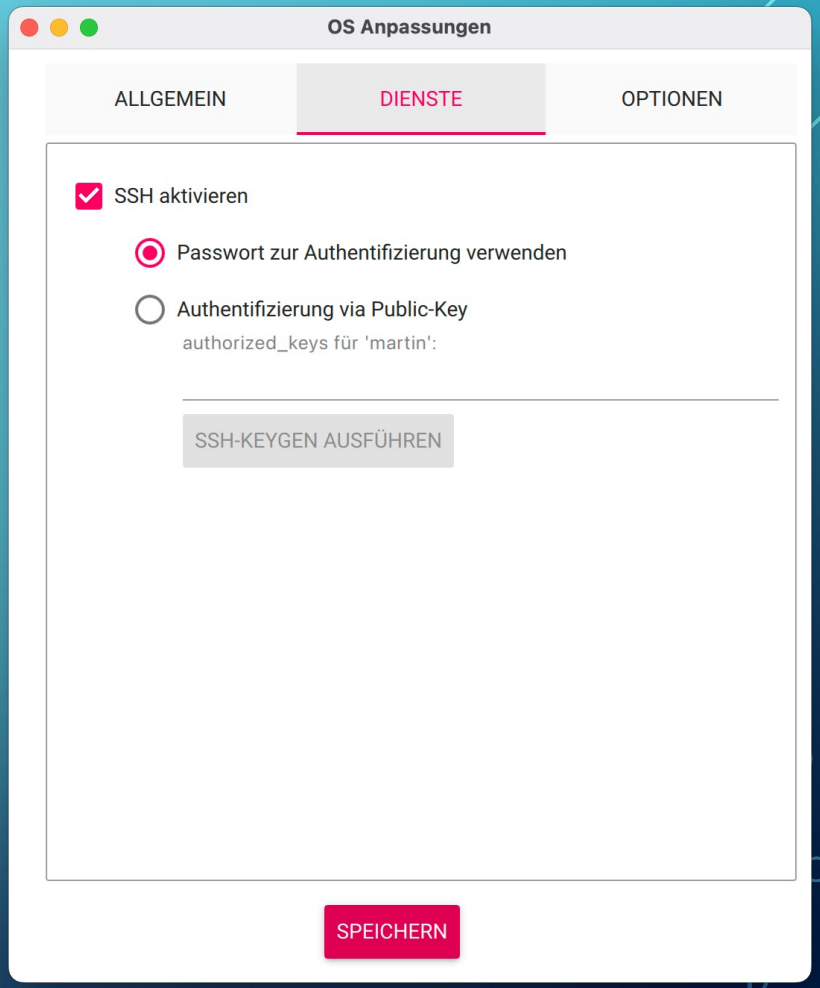
# IMPORTANT CUSTOMIZATIONS

- Choose your hostname: McApp in our example
- Choose what ever username, you want. I do not recommend to setup a standard „pi“ user, as this is a security risk
- Choose your login password, which should later be changed to a pre-shared ssh key
- Make sure you have your WiFi Settings correct, because otherwise you will not be able to access your headless system



# ACTIVATE SSH

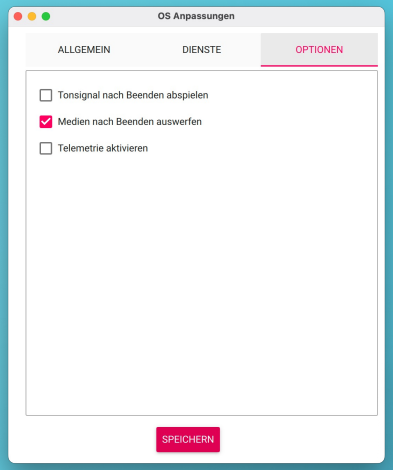
- Make sure that ssh is activated.
- For the initial setup, we start with password Authentication.
- If you are experienced, you can also set a pre-shared key.



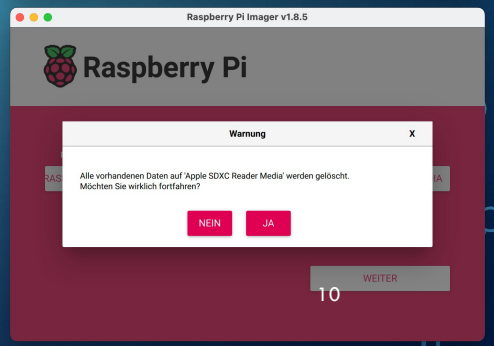
# OPTIONS

- Nothing to change here, everything standard.

- Now click save
- Then click yes to apply custom settings
- Now agree to erase everything on the SD card.

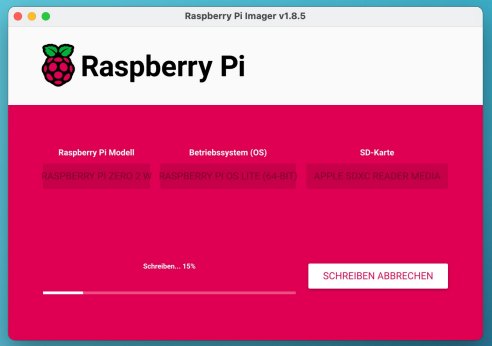






# FLASHING THE SD CARD

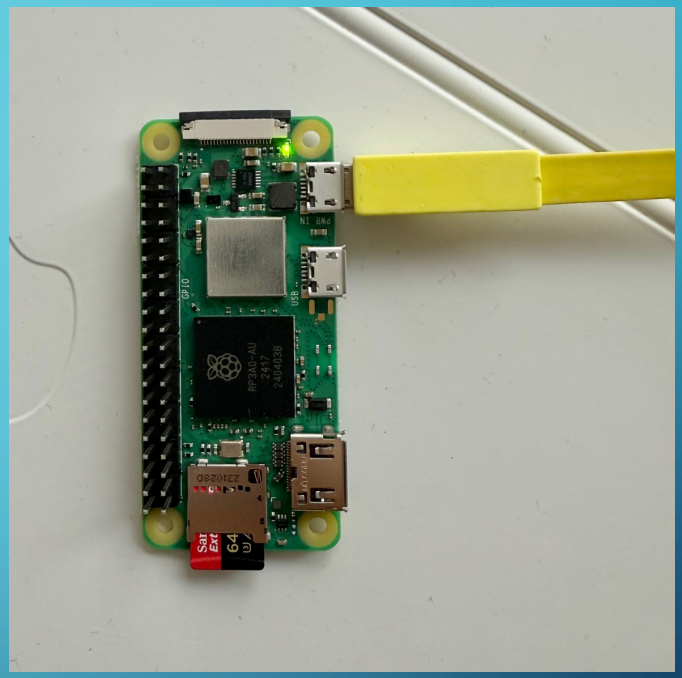
- Now wait for the flashing to be finished
- On MacOS you get asked about your Admin password, as this is a low level write, that needs more privileges




- After a short while you should see the success message
- Close Raspberry Pi Imager, eject your SD card

# FIRST TIME BOOT UP

- Insert your SD card into Raspberry Pi 2 Zero
- Attach 5V via Mini USB Jack
- The greenlight starts flashing



Raspberry Pi is booting up and expanding the filesystem. Depending on your SD card, this takes at least 2 Minutes. Grab yourself a coffee and wait

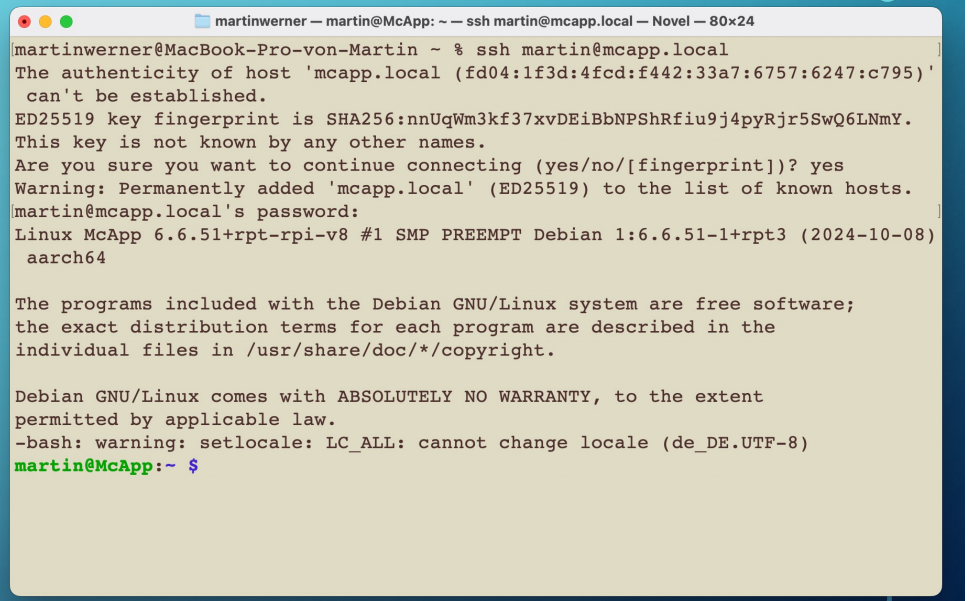
- If you have mDNS, then you can try to ping your Raspberry Pi
- Otherwise check your WiFi Router for the IP of the new device

```
martinwerner@MacBook-Pro-von-Martin .ssh % ping mcapp.local
ping: cannot resolve mcapp.local: Unknown host
martinwerner@MacBook-Pro-von-Martin .ssh % ping mcapp.local
PING mcapp.local (192.168.68.70): 56 data bytes
64 bytes from 192.168.68.70: icmp_seq=0 ttl=64 time=121.193 ms
64 bytes from 192.168.68.70: icmp_seq=1 ttl=64 time=11.150 ms
64 bytes from 192.168.68.70: icmp_seq=2 ttl=64 time=8.156 ms
64 bytes from 192.168.68.70: icmp_seq=3 ttl=64 time=3.976 ms
64 bytes from 192.168.68.70: icmp_seq=4 ttl=64 time=8.038 ms
64 bytes from 192.168.68.70: icmp_seq=5 ttl=64 time=15.590 ms
```

# TIME TO ACCESS YOUR RASPI

Use putty on Windows or term on MacOS

- Make sure to use the correct username for your Pi Zero
- Accept the new ssh fingerprint
- Enter your password
- You should now have ssh access to your Raspi



# TIME TO DO SOME CHECKS

- uname -a as we want to see, that we have installed the right Linux
- uname -m as we want to see that we are in a 64-bit environment

```
martin@McApp:~ $ uname -a
Linux McApp 6.6.51+rpt-rpi-v8 #1 SMP PREEMPT Debian
1:6.6.51-1+rpt3 (2024-10-08) aarch64 GNU/Linux
martin@McApp:~ $ uname -m
aarch64
```

# TIME TO UPDATE YOUR APT CACHE

`sudo apt update`

No need for a full blown upgrade yet! We do it at the end.

```
martin@McApp:~ $ sudo apt update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
..
Get:25 http://deb.debian.org/debian bookworm-updates/main Translation-en [360 B]
Fetched 25.6 MB in 17s (1550 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
104 packages can be upgraded. Run 'apt list --upgradable' to see them.
N: Repository 'http://deb.debian.org/debian bookworm InRelease' changed its 'Version' value from '12.8' to
'12.10'
```

# CADDY INSTALLATION

Caddy is our TLS reverse proxy, and PKI. It does automatic certificate rotation for us.

```
sudo apt install -y debian-keyring debian-archive-keyring curl apt-transport-https
```

```
martin@McApp:~ $ sudo apt install -y debian-keyring debian-archive-keyring curl apt-transport-https
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
..
```

# NOW ADD CADDY REPO

```
martin@McApp:~ $ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --
dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg

martin@McApp:~ $ echo "deb [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg] \

https://dl.cloudsmith.io/public/caddy/stable/deb/debian all main" | \

sudo tee /etc/apt/sources.list.d/caddy-stable.list

deb [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg]
https://dl.cloudsmith.io/public/caddy/stable/deb/debian all main
```

# NOW RETRIEVE CADDY UPDATES

Update the apt cache again to have caddy included

We ignore the Repo Error, it works anyway

```
martin@McApp:~ $ sudo apt update
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://archive.raspberrypi.com/debian bookworm InRelease
Hit:3 http://deb.debian.org/debian-security bookworm-security InRelease
Hit:4 http://deb.debian.org/debian bookworm-updates InRelease
Ign:5 https://dl.cloudsmith.io/public/caddy/stable/deb/debian all InRelease
Err:6 https://dl.cloudsmith.io/public/caddy/stable/deb/debian all Release
  404  Not Found [IP: 108.138.36.64 443]
Reading package lists... Done
E: The repository 'https://dl.cloudsmith.io/public/caddy/stable/deb/debian all Release' does not have a
Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

# INSTALLING OUR TLS REVERSE PROXY, THE WEBSERVER AND SCREEN

```
martin@McApp:~ $ sudo apt install caddy lighttpd screen
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
After this operation, 43.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Click Y to install, then check caddy for installation success

```
martin@McApp:~ $ caddy version
2.6.2
```

# EDITION CADDY CONFIG

```
martin@McApp:~ $ hostname

McApp

martin@McApp:~ $ sudo vi /etc/caddy/Caddyfile
```

You can use nano as well as your text editor of choice.

- Delete everything in Caddyfile and replace it
- Make sure that you use your hostname
- Make sure to use your domain
- .local for mDNS enabled environments (MacOS / Apple)
- .fritz.box for Fritz!Box WLAN Router (Windows w/o iTunes)
- Remark: you have a mDNS responder on Windows, if you install Itunes

```
{
        auto_https disable_redirects
        log {
                #level DEBUG
                level INFO
                format console
        }
}
mcapp.local {
        tls internal
        reverse_proxy 127.0.0.1:80
        encode gzip
}


mcapp.local:2981 {
        tls internal
        reverse_proxy 127.0.0.1:2980
}
```

# MAKING SURE THAT CADDY LIKES OUR CONFIGURATION

Caddy is a bit hyper critical, when it comes to configuration files. But Caddy knows what it likes and also supports us here.

- Change to caddy config directory
- Let caddy format the input
- Then validate the caddy file

- Most important „Valid configuration“
- All the rest is for us more a debug output

```
martin@McApp:~ $ cd /etc/caddy/
martin@McApp:/etc/caddy $ sudo caddy fmt --overwrite
martin@McApp:/etc/caddy $ sudo caddy fmt
martin@McApp:/etc/caddy $ sudo caddy validate
2025/04/17 07:54:48.715 INFO using adjacent Caddyfile
2025/04/17 07:54:48.728 INFO tls.cache.maintenance started
background certificate maintenance {"cache": "0x400035a3f0"}
2025/04/17 07:54:48.730 WARN http automatic HTTP->HTTPS
redirects are disabled {"server_name": "srv0"}
2025/04/17 07:54:48.730 INFO http server is listening only on
the HTTPS port but has no TLS connection policies; adding one
to enable TLS {"server_name": "srv1", "https_port": 443}
2025/04/17 07:54:48.731 WARN http automatic HTTP->HTTPS
redirects are disabled {"server_name": "srv1"}
2025/04/17 07:54:48.732 INFO tls.cache.maintenance stopped
background certificate maintenance {"cache": "0x400035a3f0"}
Valid configuration
```

# START CADDY, CHECK LIGHTTPD

```
martin@McApp:/etc/caddy $ sudo systemctl restart caddy

martin@McApp:/etc/caddy $ sudo systemctl enable --now caddy

martin@McApp:/etc/caddy $ ps uax|grep caddy

caddy       2348  0.5  8.6 1415176 36736 ?       Ssl  10:06   0:00
/usr/bin/caddy run --environ --config /etc/caddy/Caddyfile

martin@McApp:/etc/caddy $ ps uax |grep lighttpd

www-data    2131  0.0  0.5   4116  2560 ?        Ss   09:43   0:00
/usr/sbin/lighttpd -D -f /etc/lighttpd/lighttpd.conf
```

# PROVISION ROOT CERTIFICATE
# CHECK ACCESS VIA BROWSER

Copy the self sign root certificate, that is valid for 10 years, to your web browser, so that you can import it on our Client Machine.

```
martin@McApp:~ $ sudo cp /var/lib/caddy/.local/share/caddy/pki/authorities/local/root.crt /var/www/html/
martin@McApp:~ $ sudo chmod a+r /var/www/html/root.crt
martin@McApp:~ $ ls -l /var/www/html
total 8
-rw-r--r-- 1 root root 3388 Apr 17 09:43 index.lighttpd.html
-rw-r--r-- 1 root root  627 Apr 17 10:22 root.crt
```

# CHECK THE WEBSERVER DOWNLOAD SSL CERTIFICATE

- First we access our lighttpd via http – the unencrypted version.

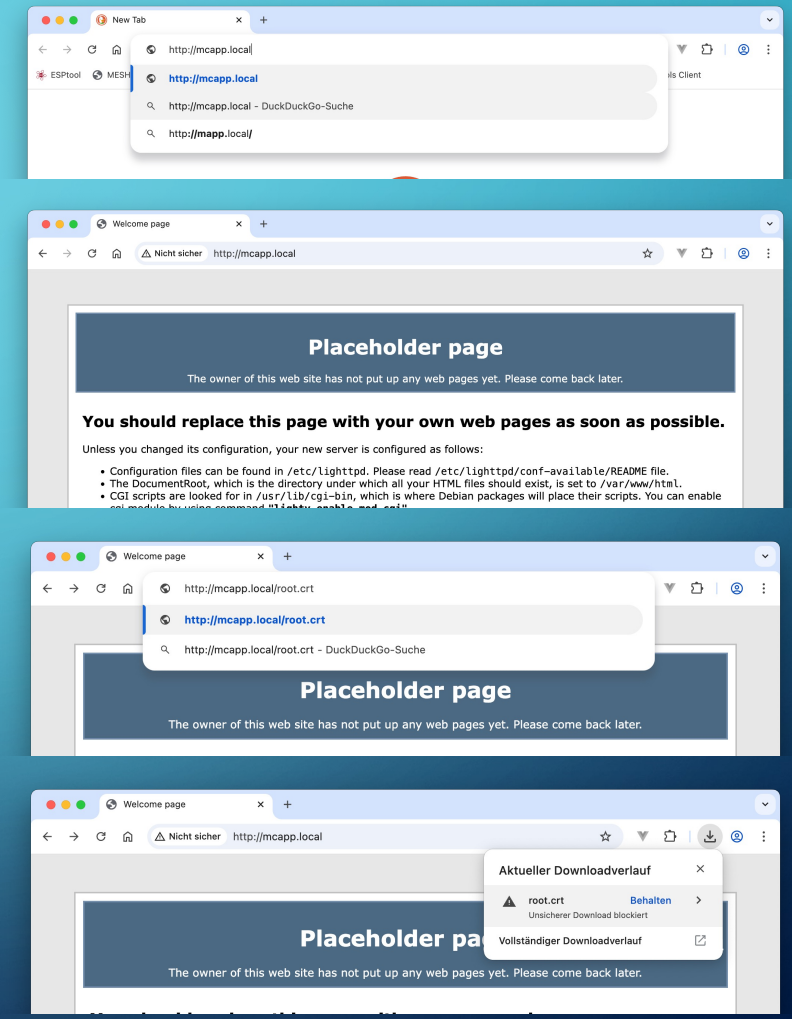
http://mcapp.local

- You should see the placeholder page

- Now we download the self-signed root certificate

http://mcapp.local/root.crt

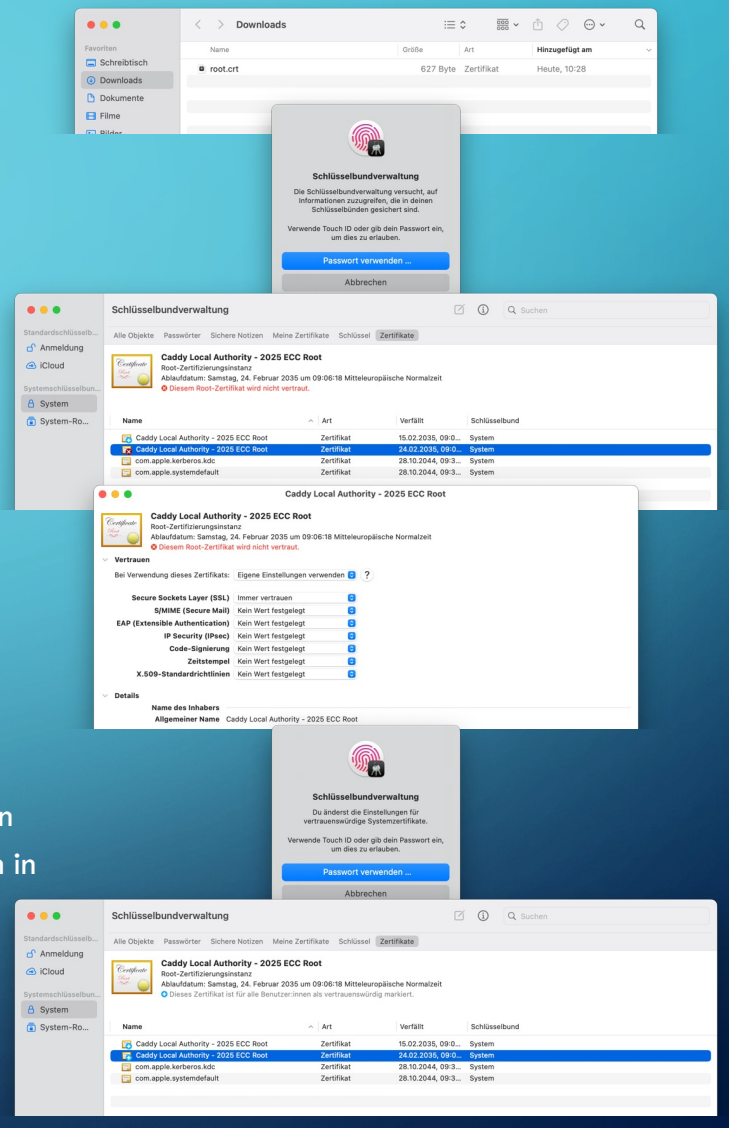
- Make sure to accept the blocked download



# INSTALL THE SSL CERTIFICATE

- Locate your root.crt in your download folder
- Double click root.crt
- Enter your Admin password
- Now locate the newly installed certificate
- Trust the certificate for TLS encryption

- Same is true for the iPhone. After import, you have to accept it in General – Profiles AND then you need to trust it, which is hidden in the settings menu

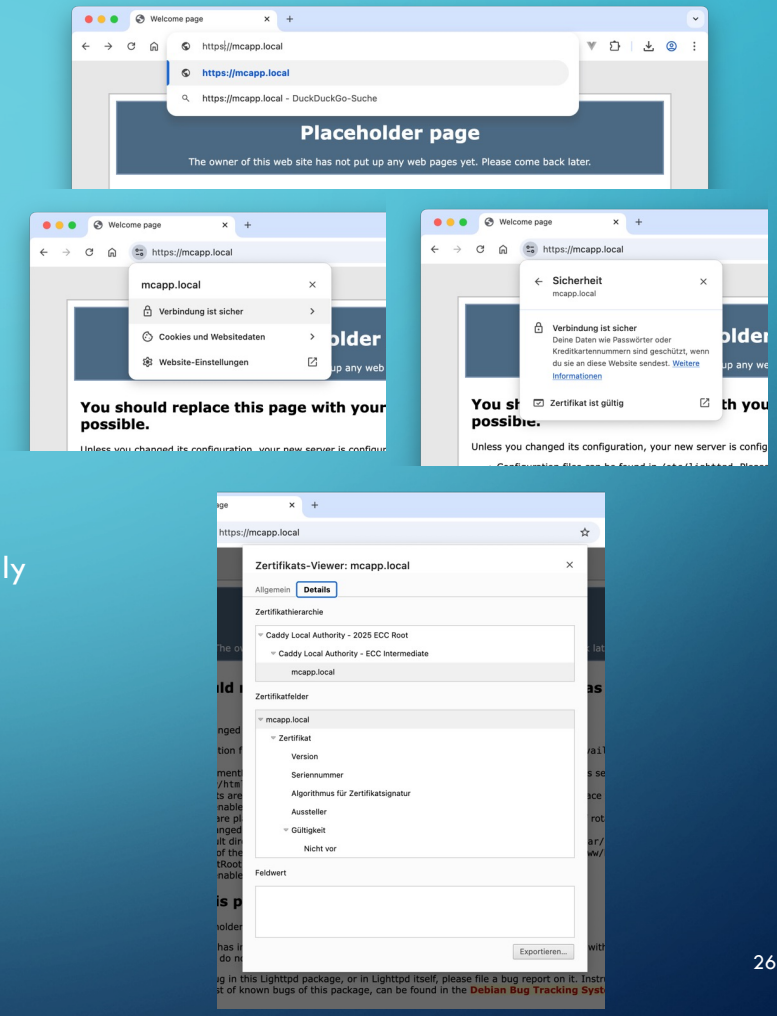


# TESTING SSL ACCESS

- Go to your web browser, now we access https

https://mcapp.local

- If everything worked out, as expected, you should see a fully trusted root chain.



# ON GITHUB YOU HAVE TWO SCRIPTS, THAT ARE OUR SERVER COMPONENT, COPY IT TO YOUR NEW RASPBERRY PI

`mc-screen`

`C2-mc-ws.py`

- Make sure our wrapper script is executeable: `chmod +x mc-screen`
- vi or nano into C2-mc-ws.py as you have to adjust the Hostname or IP of your MeshCom Node

- mc-screen is a wrapper for setting up python virtual environment and executing C2-mc-ws.py

# NOW INSTALL THE WEBAPP

```
martin@McApp:~ $ sudo mkdir /var/www/html/webapp
martin@McApp:~ $ sudo chown martin:www-data /var/www/html/webapp
martin@McApp:~ $ ls -l /var/www/html/
total 12
-rw-r--r-- 1 root   root     3388 Apr 17 09:43 index.lighttpd.html
-rw-r--r-- 1 root   root      627 Apr 17 10:22 root.crt
drwxr-xr-x 2 martin www-data 4096 Apr 17 12:48 webapp
```

Now copy the webarchive over and unzip it

```
martin@McApp:/var/www/html $ ls -l
insgesamt 12
-rw-r--r-- 1 root   root     3388 17. Apr 09:43 index.lighttpd.html
-rw-r--r-- 1 root   root      627 17. Apr 10:22 root.crt
drwxr-xr-x 4 martin www-data 4096 17. Apr 13:12 webapp
martin@McApp:/var/www/html $ ls -l webapp/
insgesamt 28
drwxr-xr-x 2 martin martin 4096 17. Apr 13:12 assets
-rwxr-xr-x 1 martin martin 2580 17. Apr 13:12 favicon.ico
-rw-r--r-- 1 martin martin 4286 17. Apr 13:12 favicon-ren.ico
drwxr-xr-x 2 martin martin 4096 17. Apr 13:12 img
-rwxr-xr-x 1 martin martin  699 17. Apr 13:12 index.html
-rwxr-xr-x 1 martin martin  284 17. Apr 13:12 manifest.json
martin@McApp:/var/www/html $
```

# ACTIVATE REDIRECT IN LIGHTTPD

```
martin@rpiZero:/etc/lighttpd $ sudo vi /etc/lighttpd/lighttpd.conf
```

Go to the end of the file and add this.

We have a webpage, that is rendered inside our browser and the path is faked. So we need to make lighttpd aware.

```
$HTTP["url"] =~ "^/webapp/" {
  url.rewrite-if-not-file = (
    "^/webapp/(.*)" => "/webapp/index.html"
  )
}
```

# CREATE PYTHON ENVIRONMENT
# INSTALL WEBSOCKETS
# FIRE UP UDP PROXY

```
martin@McApp:~ $ python3 -m venv venv
martin@McApp:~ $ source venv/bin/activate
(venv) martin@McApp:~ $ pip install websockets


(venv) martin@McApp:~ $ python3 C2-mc-ws.py
Dump geladen: 0 Nachrichten (0.00 KB)
Nach dem Prune verbleiben 0 Nachrichten
WebSocket-Server läuft auf ws://0.0.0.0:2980
UDP-Proxy läuft auf Port 1799, Weiterleitung an ('dk5en-99.local', 1799)
Drücke 'q' + Enter zum Beenden und Speichern
```

Now press q and enter to quit.

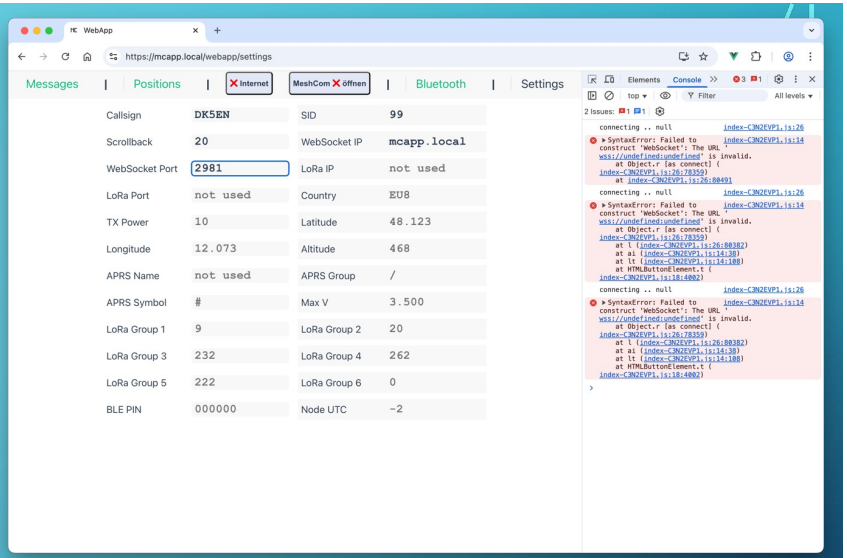
We have a startup script that runs the python3 UDP proxy in a screen session

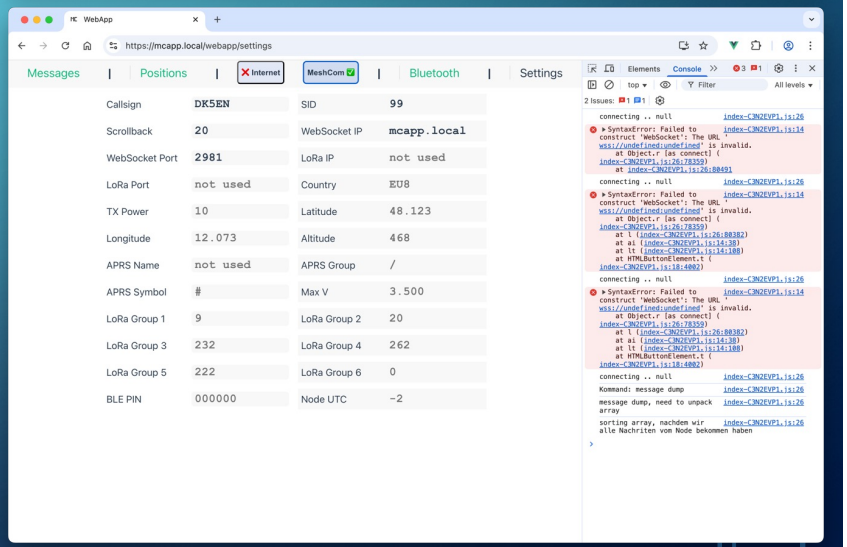
```
martin@McApp:~ $ chmod +x mc-screen
martin@McApp:~ $ ./mc-screen
```

From now on, every time you reboot your server, make sure to fire up your UDP proxy

# OPEN WEBBROWSER

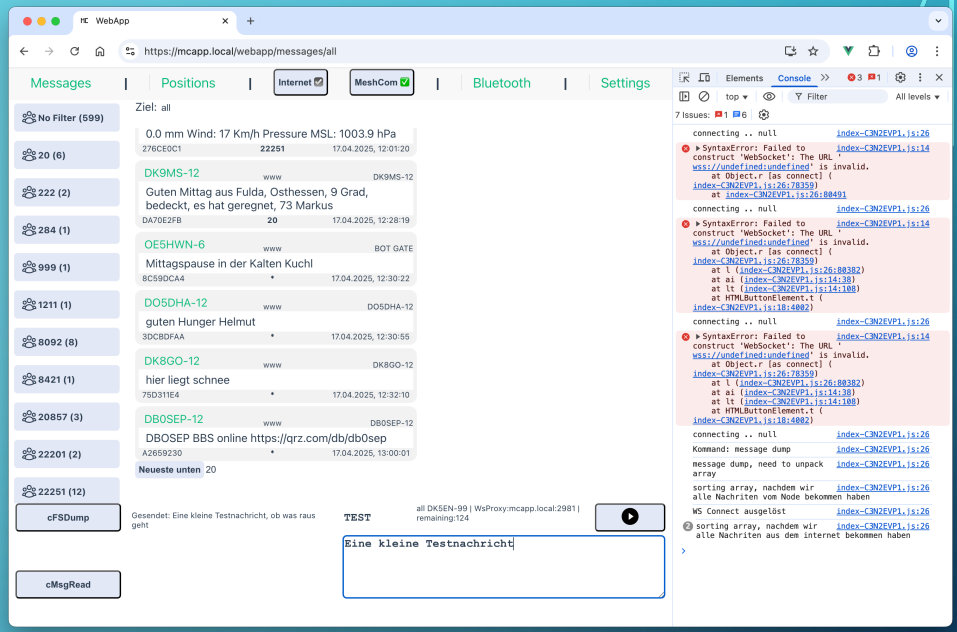
- Click option command J for Debug Output
- Go to url: https://mcapp.local/webapp/
- Click on Settings
- Enter your Callsign
- Your SID
- 20 for scroll back buffer
- mcapp.local for your UDP Proxy server
- 2981 for the secure socket of the UDP Proxy that runs in python
- Click Connect MeshCom

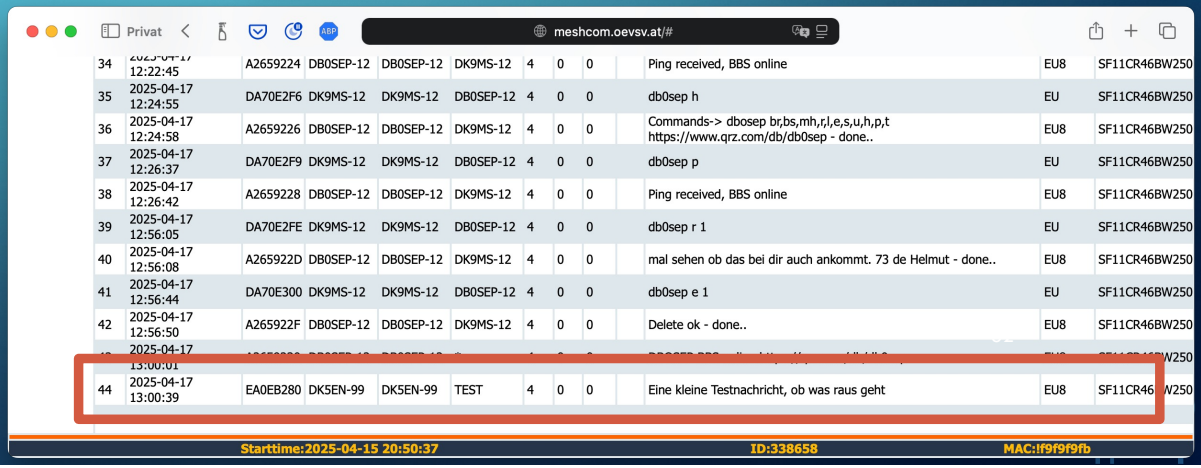


# TIME TO TEST EVERYTHING

- Go to https://meshcom.oevsv.at/#
- Click on Test
- Now enter a test Message
- Check on MeshCom Test page




| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 34 | 12:22:45 | A2659224 | DB0SEP-12 | DB0SEP-12 | DK9MS-12 | 4 | 0 | 0 | Ping received, BBS online | EU8 | SF11CR46BW250 |
| 35 | 2025-04-17 12:24:55 | DA70E2F6 | DK9MS-12 | DK9MS-12 | DB0SEP-12 | 4 | 0 | 0 | db0sep h | EU | SF11CR46BW250 |
| 36 | 2025-04-17 12:24:58 | A2659226 | DB0SEP-12 | DB0SEP-12 | DK9MS-12 | 4 | 0 | 0 | Commands-> dbosep br,bs,mh,r,l,e,s,u,h,p,t https://www.qrz.com/db/db0sep - done.. | EU8 | SF11CR46BW250 |
| 37 | 2025-04-17 12:26:37 | DA70E2F9 | DK9MS-12 | DK9MS-12 | DB0SEP-12 | 4 | 0 | 0 | db0sep p | EU | SF11CR46BW250 |
| 38 | 2025-04-17 12:26:42 | A2659228 | DB0SEP-12 | DB0SEP-12 | DK9MS-12 | 4 | 0 | 0 | Ping received, BBS online | EU8 | SF11CR46BW250 |
| 39 | 2025-04-17 12:56:05 | DA70E2FE | DK9MS-12 | DK9MS-12 | DB0SEP-12 | 4 | 0 | 0 | db0sep r 1 | EU | SF11CR46BW250 |
| 40 | 2025-04-17 12:56:08 | A265922D | DB0SEP-12 | DB0SEP-12 | DK9MS-12 | 4 | 0 | 0 | mal sehen ob das bei dir auch ankommt. 73 de Helmut - done.. | EU8 | SF11CR46BW250 |
| 41 | 2025-04-17 12:56:44 | DA70E300 | DK9MS-12 | DK9MS-12 | DB0SEP-12 | 4 | 0 | 0 | db0sep e 1 | EU | SF11CR46BW250 |
| 42 | 2025-04-17 12:56:50 | A265922F | DB0SEP-12 | DB0SEP-12 | DK9MS-12 | 4 | 0 | 0 | Delete ok - done.. | EU8 | SF11CR46BW250 |
| 44 | 2025-04-17 13:00:39 | EA0EB280 | DK5EN-99 | DK5EN-99 | TEST | 4 | 0 | 0 | Eine kleine Testnachricht, ob was raus geht | EU8 | SF11CR46 W250 |

# ANNOYANCES WITH LOCALES AND A SYSTEM UPDATE

```
martin@McApp:~ $ sudo raspi-config
```

- 5 Localization -> L1 locale -> select de_AT.UTF-8 -> OK -> C.UTF-8 -> OK
- Finish

```
martin@McApp:~ $ sudo apt-get update
martin@McApp:~ $ sudo apt-get dist-upgrade
martin@McApp:~ $ sudo reboot
```