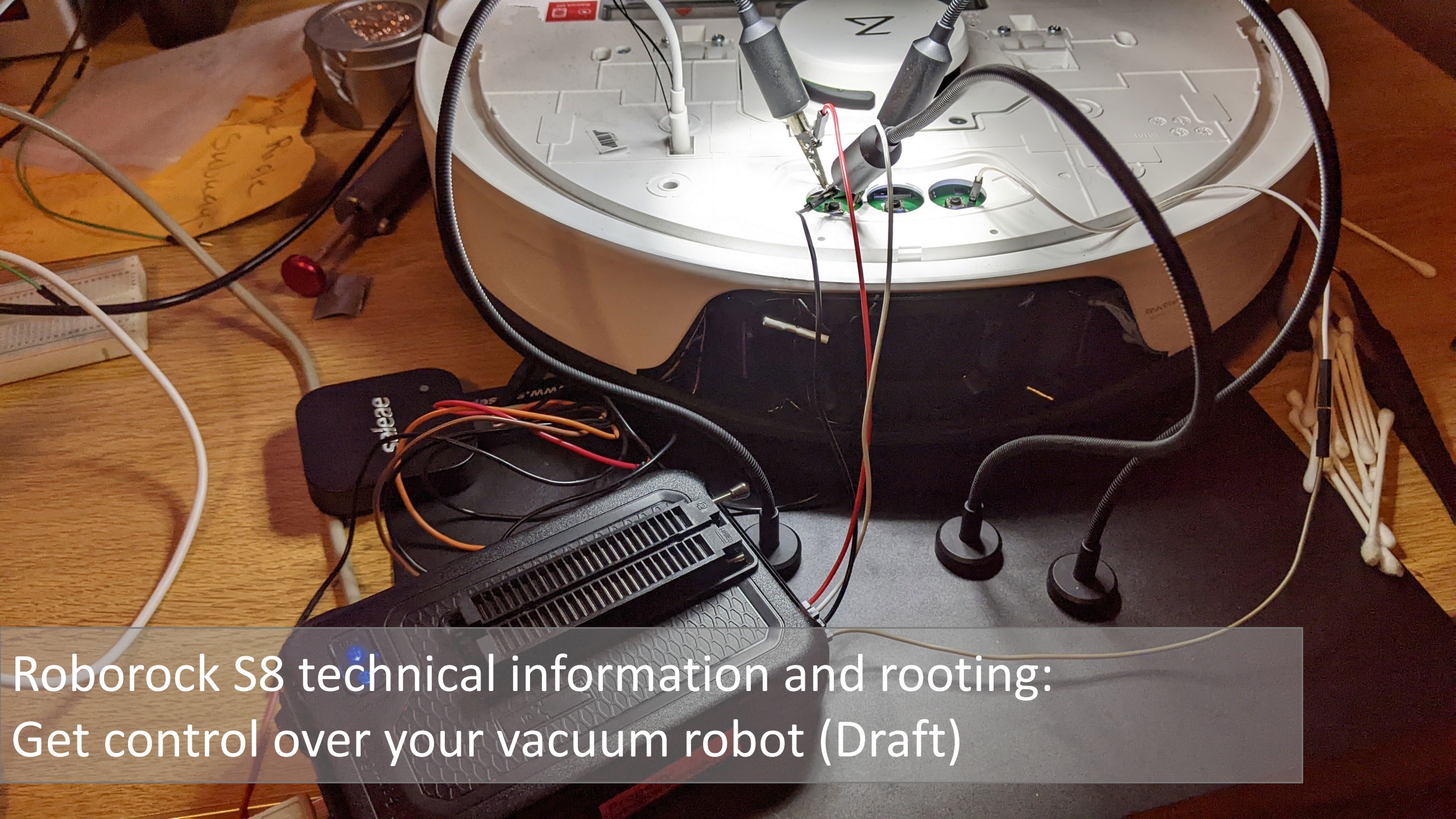


Roborock S8 technical information and rooting:
Get control over your vacuum robot (Draft)

Before you continue: Please look thru all slides and the FAQ before you start your adventure

You might want to join the Telegram channel:

https://t.me/dust_announce

Good news: No soldering or teardown required … if you dare and have a S8, S8 Pro Ultra

Why get root access?

- Use Valetudo (https://valetudo.cloud/)
  - Replace the cloud functionality with an open-source software
  - Integrate the device into your home automation
- Install your own soundfiles/voices

Currently tested devices (08.2024)

- Roborock S8 (a51)
- Roborock S8 Pro Ultra (a70)
- Roborock G10s (a46, requires teardown)

Risks

- A failed flash can leave the device in an undefined state
- Requires reflashing
- Problem:
  - All partitions need to be recreated
  - Device identity (Device ID, Cloud keys, MAC) gets lost
  - Calibration data lost
- Observations:
  - Device identity does not matter if you use Valetudo
  - Unknown consequences for lost calibration data

More risks

- Problem: Hardware differs even for the same „model“
- The root method cannot verify your exact model
- Flashing an incorrect firmware will perma-damage your device
  - Other, not obvious side effects might occur
  - Recovery might be tricky and can cause problems
- Important: Make sure that you have the correct firmware
  - Do not try to use the same generated firmware on multiple devices
  - If you are unsure, ask us first!

Sidenotes

- Rooting will permanently change files on your device*
  - Cloud connection / App usage should be still possible
- By installing a custom firmware, you should still not update your device with official firmwares anymore
  - That is the price of rooting
  - You can update your device with custom firmwares
- Connect to your robot and write down the MAC address

* we needed to reconstruct parts from the firmware by extracting contents from flash and guessing unknown parts.

Important information

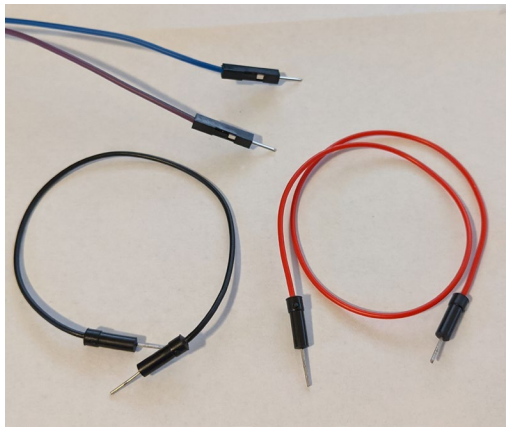
- DO NOT UPDATE THE FIRMWARE VIA THE APP if you ever plan to root your device
  - New firmwares have rooting methods patched
  - Root might be still possible, but is more complicated and risky
- Before rooting: test the device without using the app
  - Fully charge your device
  - Use the buttons to start the cleaning run
  - Let the robot clean multiple times
  - Background: Some devices come with defects from the factory. It is tricky to figure out if your robot behaves weirdly due to it being already broken or due to rooting

Current available rooting methods

- Livesuit
  - Requires: USB connection
  - Idea: creation of special factory image, triggering bootrom mode (FEL), flashing of all partitions via Livesuit software (requires Windows)

Tools required for root

- Breadboard Jumper Wires
- Or better: PCBites
- Something to scratch off soldermask
- MicroUSB cable (for Livesuit/FEL)



Opening the device

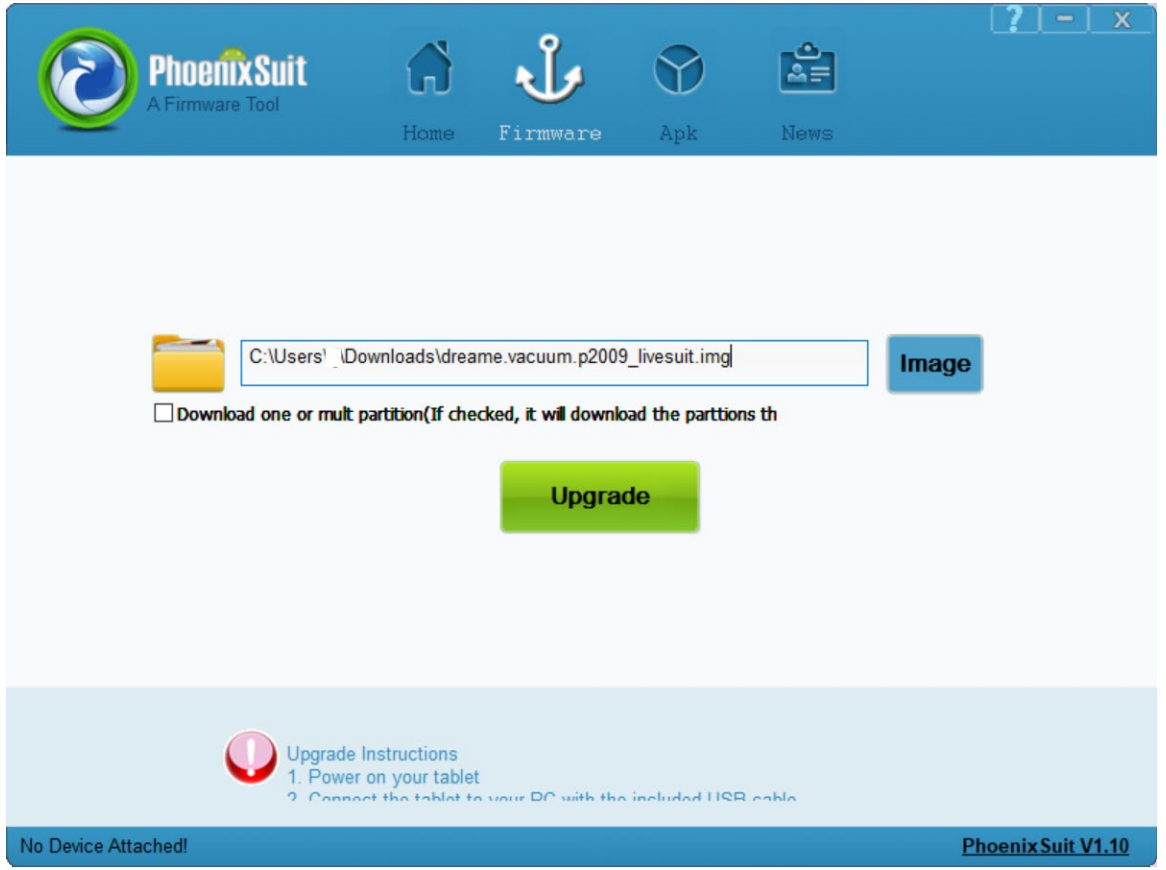
ROOTING

Root Preparations

- Requirements:
  - Windows (can be a VM)
  - (you can also use Linux, but it requires Kernel modules installation)
  - USB port
- Preparation:
  - Download and unpack Phoenixsuit from here: https://androidmtk.com/download-phoenixsuit
    - Alternative for Linux: https://github.com/linux-sunxi/sunxi-livesuite
    - Livesuit for Linux behaves the same as Phoenixsuit
  - Download and unpack ADB and Fastboot from here: https://developer.android.com/tools/releases/platform-tools
  - Download and unpack drivers from here: https://builder.dontvacuum.me/nextgen/usbdriver.zip
  - Download image from: https://builder.dontvacuum.me/nextgen/

Root Step 1

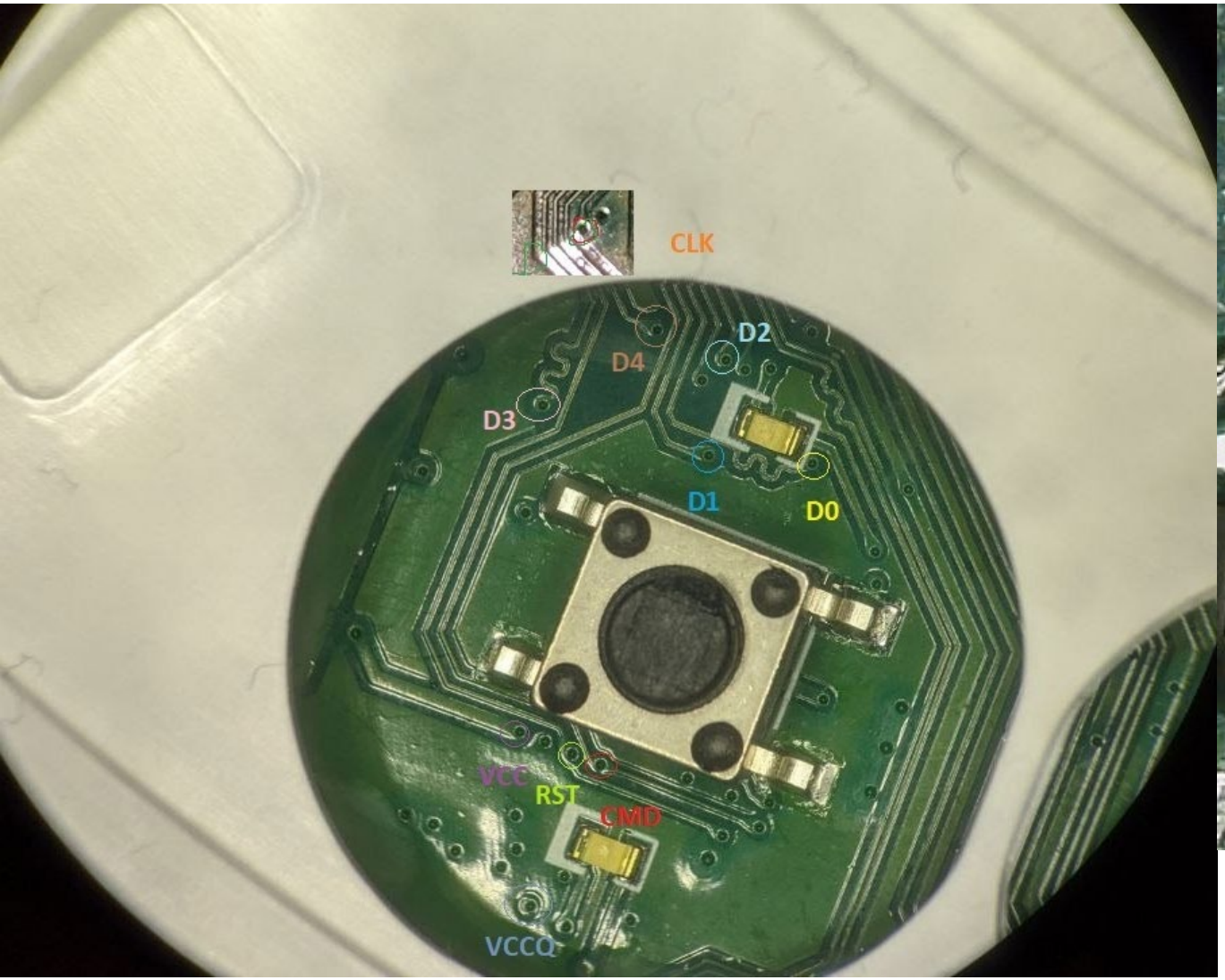
- Make sure that the robot is not connected over USB
- Open Phoenixsuit and select image “dust-RR-livesuit.img”

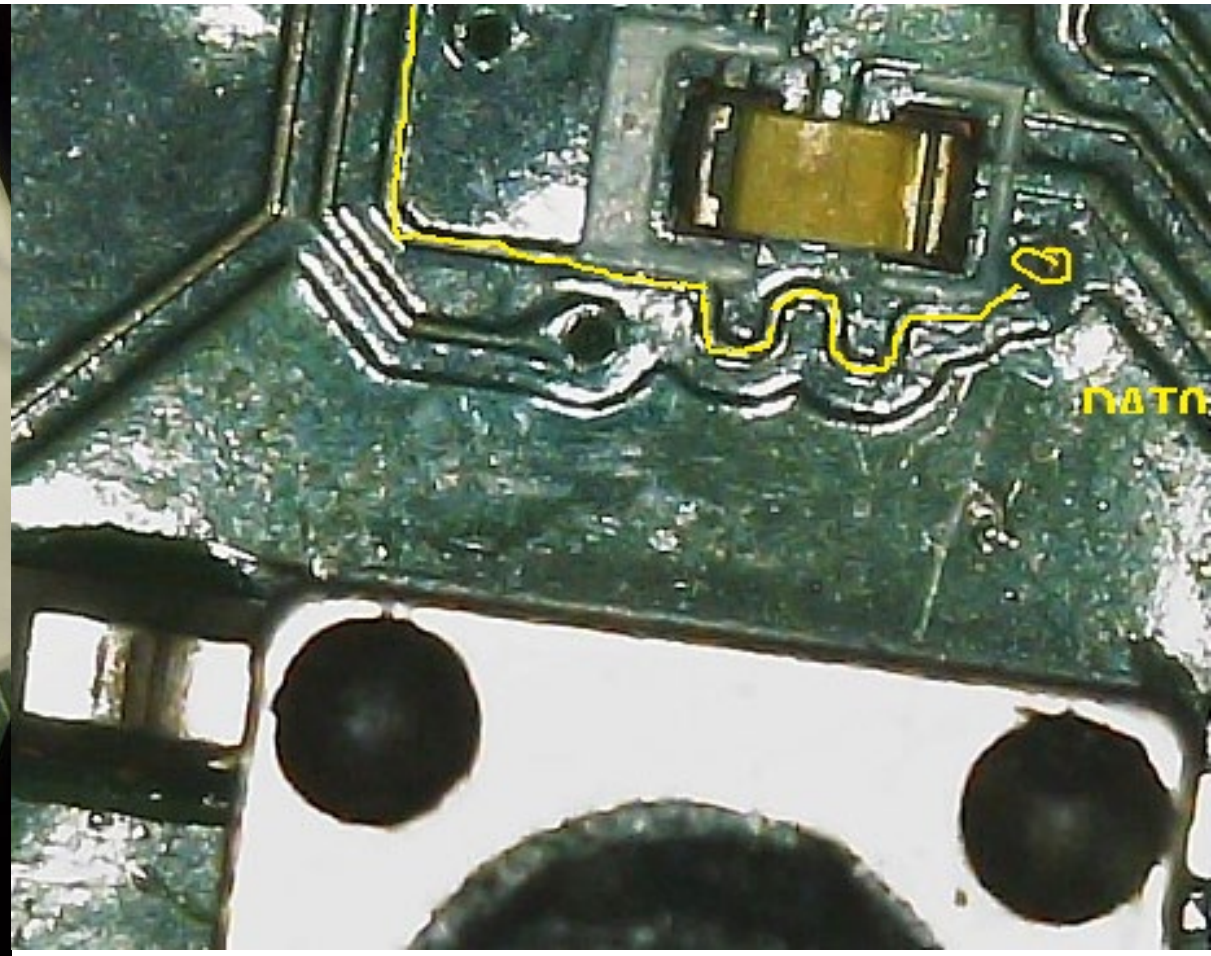


Root Step 2

- Boot the Robot in FEL mode
  - Connect MicroUSB to your computer (no USB hub, preferred USB 2.0 port)
  - S8, S8 Pro: Short DAT0 trace to GND (e.g. USB shield)
  - Alternative if you do a tear down: GND TPA41 on the back side
  - G10s: GND TPA41 on the back side, or use UART and press “2” while boot
  - Press the power button for 3 seconds
    - USB device should show up on your computer
    - You might need to install the Phoenixsuit drivers located in “Drivers/AW_Driver”
      - (via Device Manager -> Unknown Device -> Update drivers)

S8, S8 Pro Ultra DAT0 position

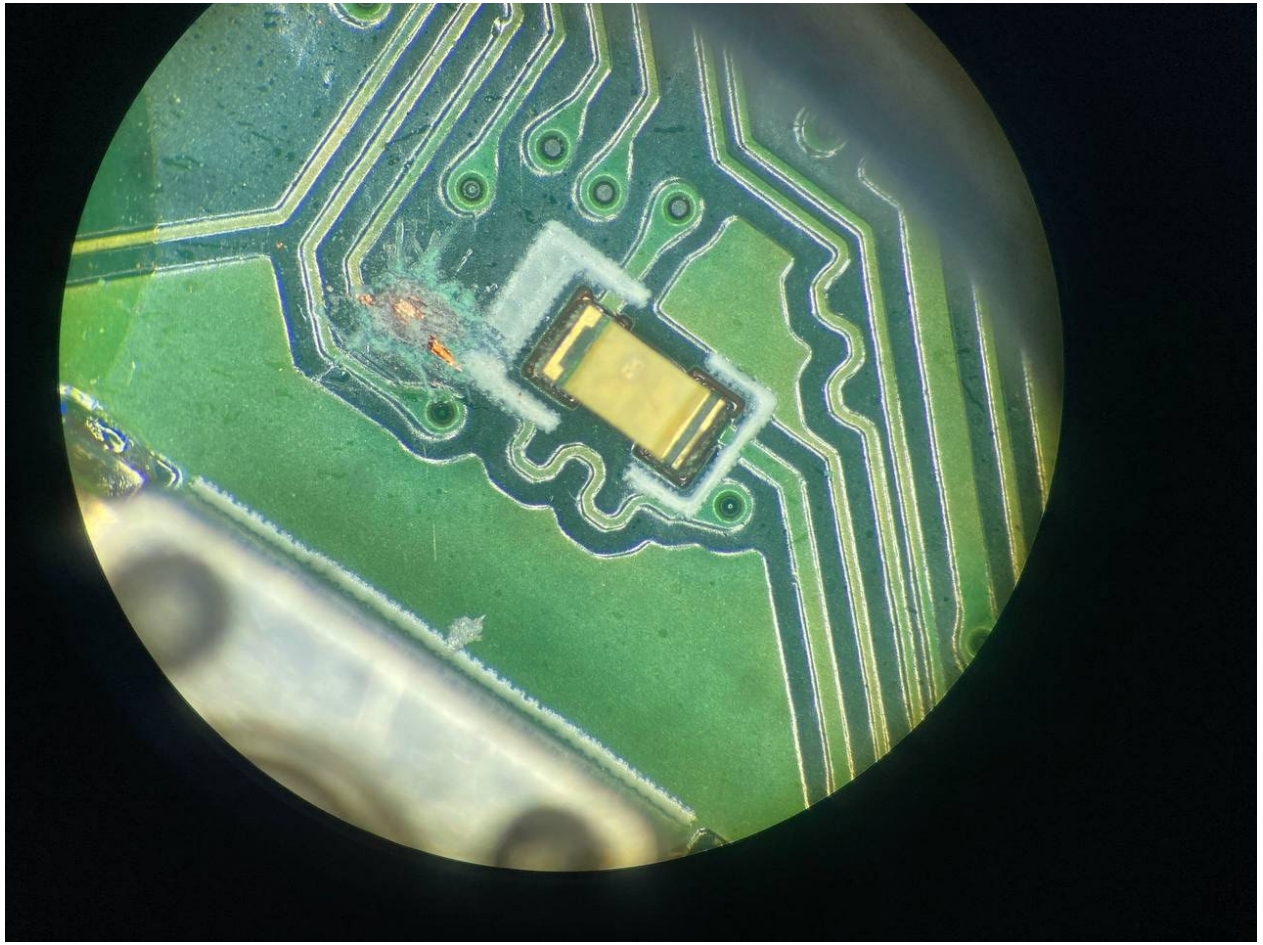


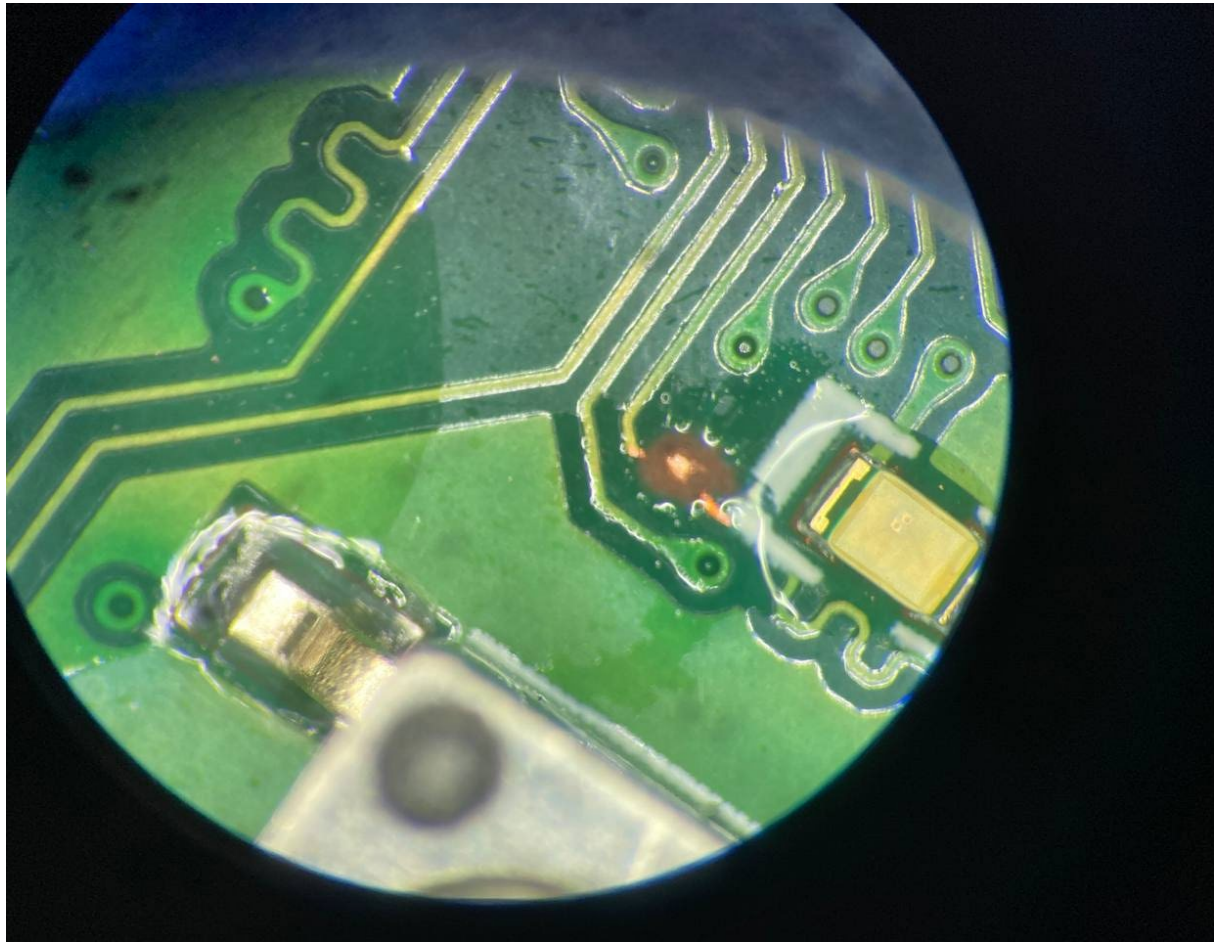


I have a quick demonstration here:
https://media.ccc.de/v/camp2024-57158-vacuum_robot_security_and_privacy#t=3487

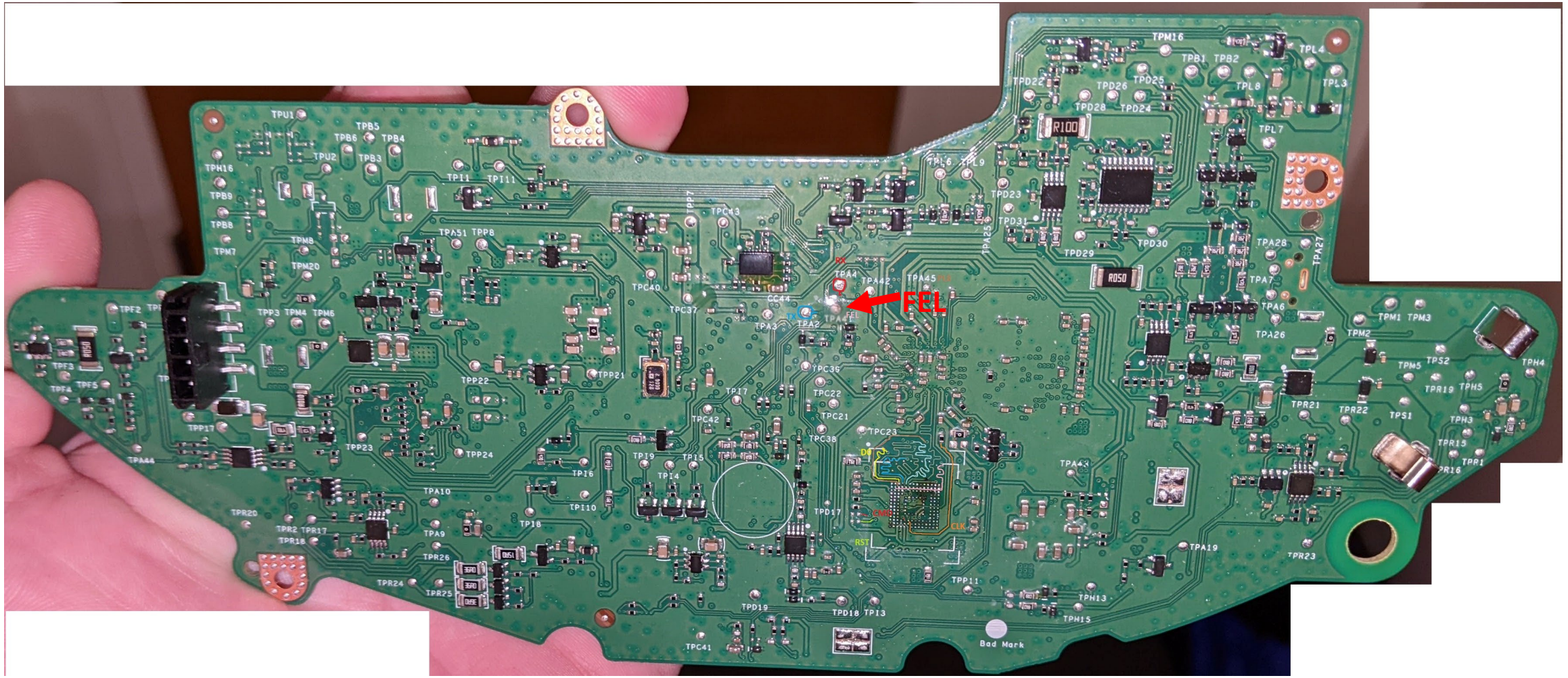
S8, S8 Pro Ultra DAT0 position

- Do not scratch too deep!!

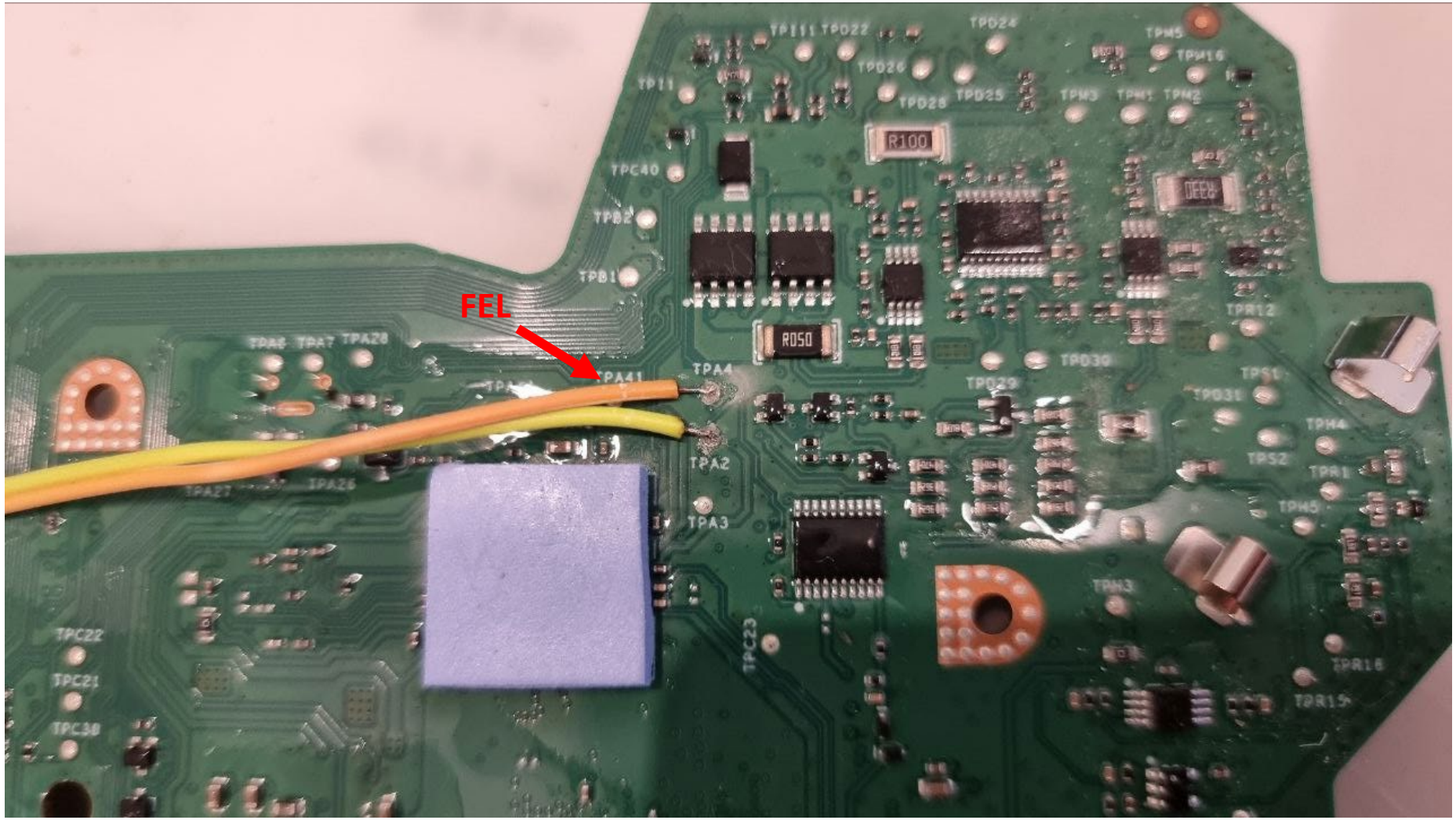




S8, S8 Pro Ultra FEL pin position

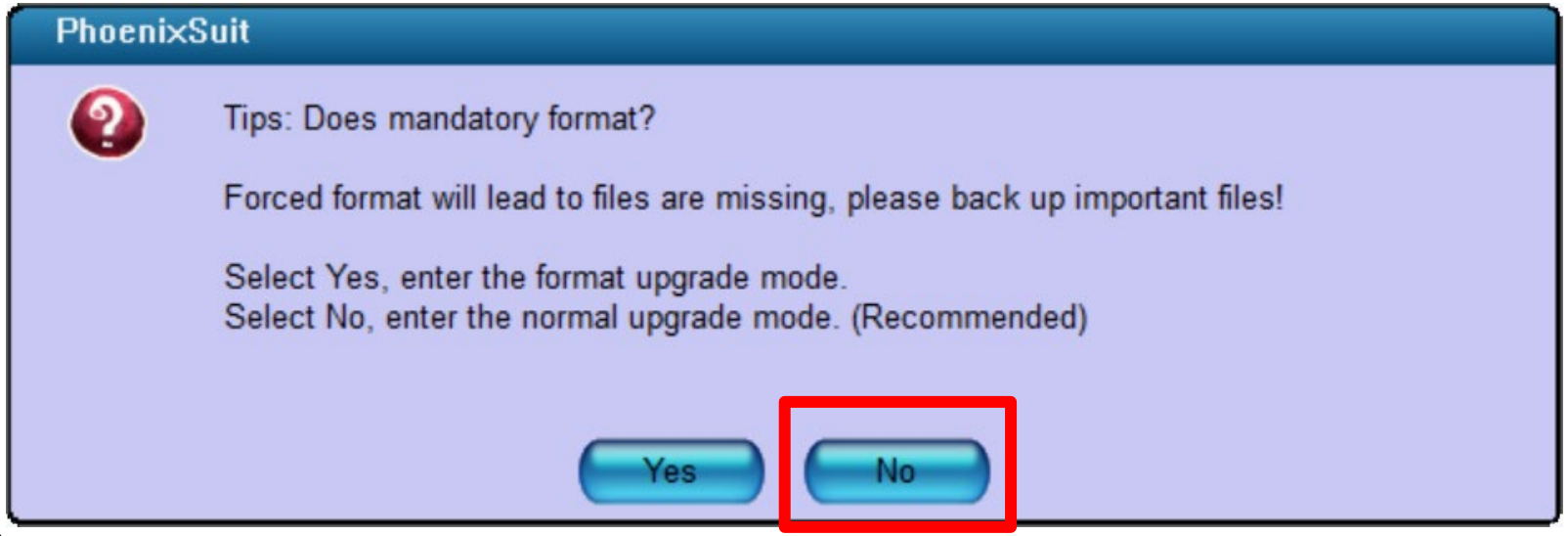


G10s pinout


FEL
TPA4
TPA2
TPA3
TPC22
TPC21
TPC38
TPC23
TPB2
TPB1
TPC40
R100
R050
TPR12
TPH4
TPR1
TPH5
TPH3
TPR16
TPR15

Root Step 3

- Disconnect FEL or DAT0 before proceeding
- In Phoenixsuit
  - Click on Update
  - If asked about mandatory format, select “No” !
  - The flashing process will get stuck/fail -> that’s okay and expected!



Root Step 3 Trouble shooting

- Do not use USB hubs
- Use only USB 2.0 connections
- If the device is detected initially, but disappears:
  - Try to use a different USB port
  - If you use VMs: Make sure that the USB filter is correct

Root Step 4

- Device should reappear as an ADB device.
- Open command line (cmd) and run:

  # fastboot devices

  – should return "Android Fastboot fastboot"

  # fastboot getvar config

  – should return "config: aabbccddeeff1122334455667788990O“
- Go to https://builder.dontvacuum.me
  – generate a FEL firmware
  – supplying the string from above as the config value (aabbccddeeff1122334455667788990O)
- safe this string, as you will need it again in the future

Don’t panic if dustbuilder does not accept your config value. Please send us a PM via Telegram and we will fix that.

Root Step 4a (for debugging)

- Run additionally these commands and save them:

  # fastboot getvar dustversion
  - should return “dustversion: 2024.xx.xx“

  # fastboot getvar ramsize
  - should return “ramsize: XXX0000000000000ZXXXXXXXXXXXXXX “
  - If Z=4 -> 1024MByte RAM, if Z=2 -> 512MByte RAM

  # fastboot getvar toc0hash
  - should return “toc0hash: aabbccddeeff1122334455667788990 0“

  # fastboot getvar toc1hash
  - should return “toc1hash: aabbccddeeff1122334455667788990 0“

  # fastboot getvar toc1version
  - should return “toc1version: U-Boot xxxx“

If Dustbuilder does not accept your config value, please provide all these information to us. This helps us to figure out what kind of settings we need to configure.

Upload this information and your generated files in 4b) to https://check.builder.dontvacuum.me (only if config value is unknown!)

Root Step 4b (for fallback)

- In case something goes wrong, this step helps to fix your device:
  - Linux/Windows:
    - fastboot get_staged dustx100.bin
    - fastboot oem stage1
    - fastboot get_staged dustx101.bin
  - Expected outputs:

Uploading 'dustx100.bin' *OKAY [38.853s]*

Finished. Total time: 38.854s

You should get ~400MBytes per file!

If your robot reboots in stage1, just skip the first step and continue with “fastboot oem stage1”

Provide us the files in case your robot is “soft-bricked”

Root Step 5

- After you get the firmware package, unpack the archive
  - Select the file “_rr.XXX_phoenixsuit.img” as a firmware file in PhoenixSuit
  - Power off the device
  - Repeat Steps 2 and 3 (with the new file)

Root Step 6

- copy 8 character string from check.txt and run command:

  # fastboot oem dust aabbccdd

  – this should return OKAY, if it does not, DO NOT PROCEED!

- Now we can start the rooting process:

  # fastboot oem prep

  – this should return OKAY, if it does not, DO NOT PROCEED!

Root Step 7

- copy 8 character string from check.txt and run command:

  # fastboot flash boot_a boot.img

  # fastboot flash boot_b boot.img

  # fastboot flash system_a rootfs.img

  # fastboot flash system_b rootfs.img

  – this should return OKAY. It might take 5-10 minutes for the commands to complete. Do not worry. Do not power off the device!

Root Step 7

- We should be done now and need to restart the device:

  # fastboot reboot
- The device should now reboot (the jingle should play)
- Press the outside 2 buttons to put the device in provisioning mode
- Connect to the robots WiFi, ssh to 192.168.8.1 (user:root)
- Proceed with Valetudo installation

FIRMWARE CUSTOMIZATION

You have now installed a custom firmware ;)

Customization of firmware

- The firmware builder adds to hooks to the start process
  - /mnt/reserve/_post_root.sh (gets executed after boot)
- At each boot we check for the presence of the custom files
  - Safety measurement: In case something goes wrong, a factory reset will delete this files
- An example file can be found in /root

Valetudo installation

- Download valetudo binary to /data:
  - https://github.com/Hypfer/Valetudo/releases
  - Device dependent (check our website):
    - S8, S8 pro, G10s: valetudo-aarch64
  - “wget https://github.com/Hypfer/Valetudo/releases/latest/download/valetudo-aarch64-O /mnt/data/valetudo”
- Make valetudo executable
  - “chmod +x /data/valetudo”
- Enable boot script
  - “cp /root/_post_root.sh.tpl /mnt/reserve/_post_root.sh”
  - “chmod +x /mnt/reserve/_post_root.sh”
- Reboot

You might need to add “--no-check-certificate” if wget fails


Thank you for watching!
@dgi_DE
Website: dontvacuum.me