S 14:13

Script Your Car

2013-12-28

Felix Domke

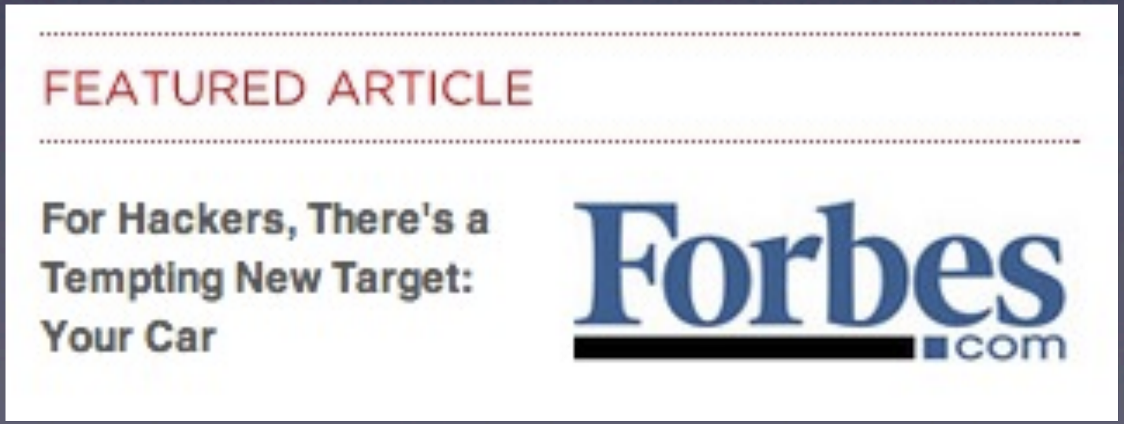
30C3

7.0 °C

km 76505 trip 579.1

Why Would Anyone Want to...?

- I love hacking. I love programming. I love Python.
- I don't love cars.
- But I'm spending way to much time in a car to not try to combine these.
- Also this:



FEATURED ARTICLE

For Hackers, There's a Tempting New Target: Your Car

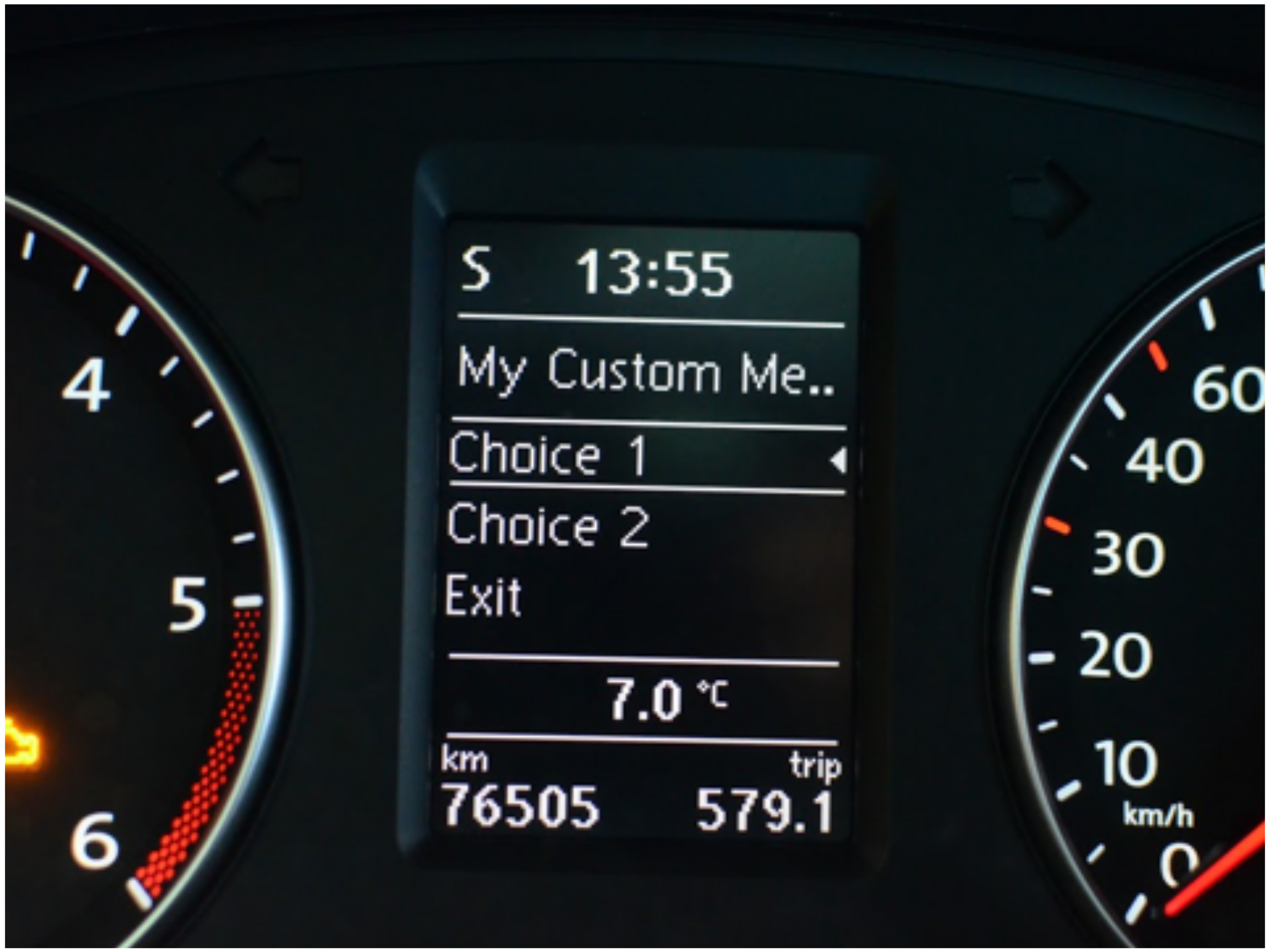
Forbes.com

Let’s Hack the Car...

- “If You Can’t Open It, You Don’t Own It.” ™
- Yeah, but I’m not a car mechanic. Plus, the car better still works after I’m done (or I’ll be done).
- What’s more interesting than an engine?
- ADDING PYTHON TO YOUR CAR.

A python in your car.

http://stuffaroundyou.blogspot.de/2012/05/picture-of-python-in-car-engine.html



Python in your car.

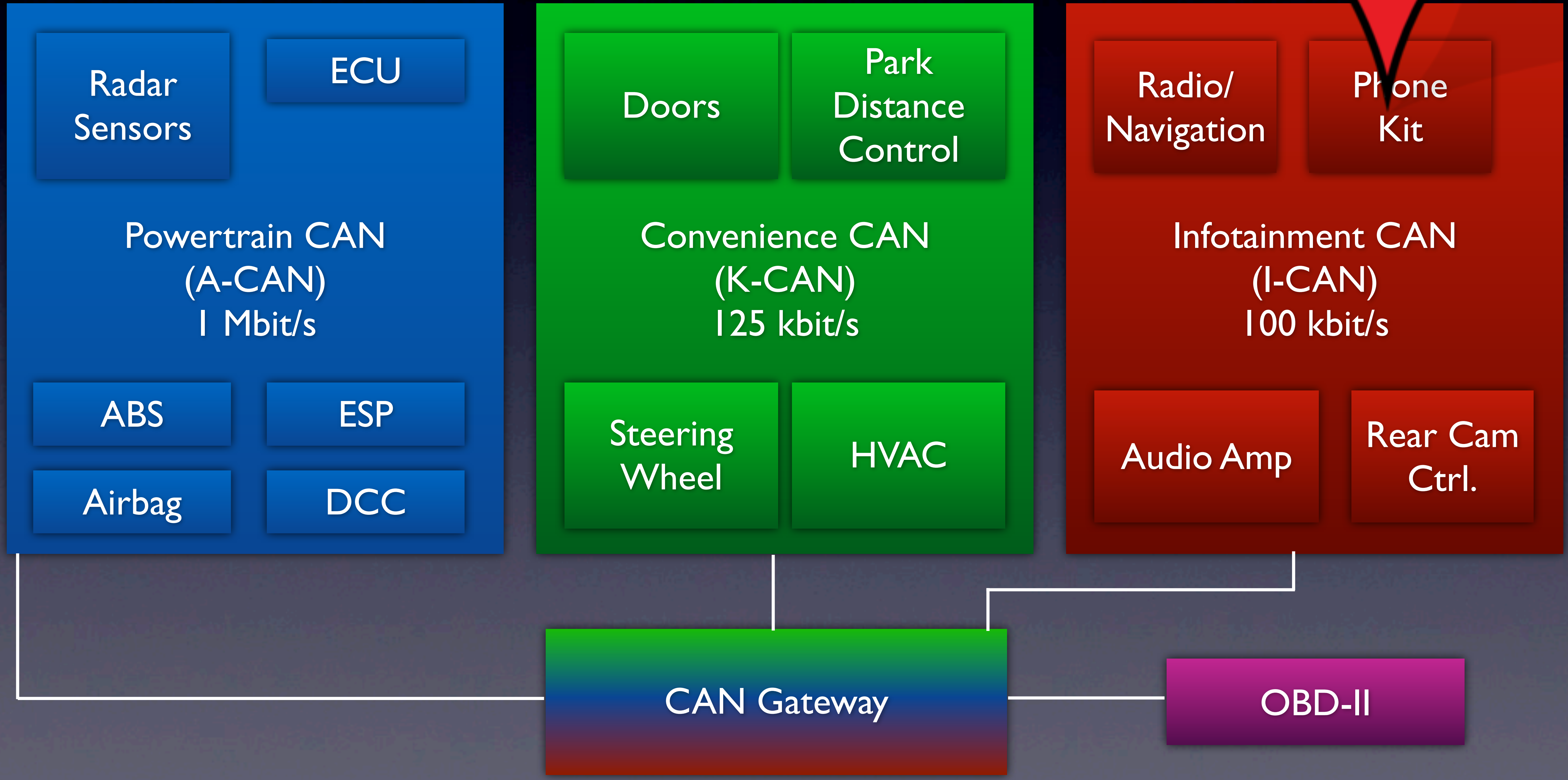
Entry Point: Bluetooth Kit?

- Why would hacking the bluetooth kit be interesting at all?
  - It shows up in the car's dashboard menu.
  - It supports Internet. (And Forbes said that Internet-enabled cars are the future!)
  - It can play audio.
  - If it breaks, I can still drive.

Where Are We?

- Most cars have “schematics” available.
  - Well, mostly wiring diagrams, but still.
- Volkswagen/Audi/Seat/Skoda has “erWin”
  - Allows downloading official documents for a small fee. No need to pirate them!

A (Semi-)Modern Ca



Radar Sensors

ECU

Powertrain CAN
(A-CAN)
1 Mbit/s

ABS

ESP

Airbag

DCC

Doors

Park Distance Control

Convenience CAN
(K-CAN)
125 kbit/s

Steering Wheel

HVAC

Radio/ Navigation

Phone Kit

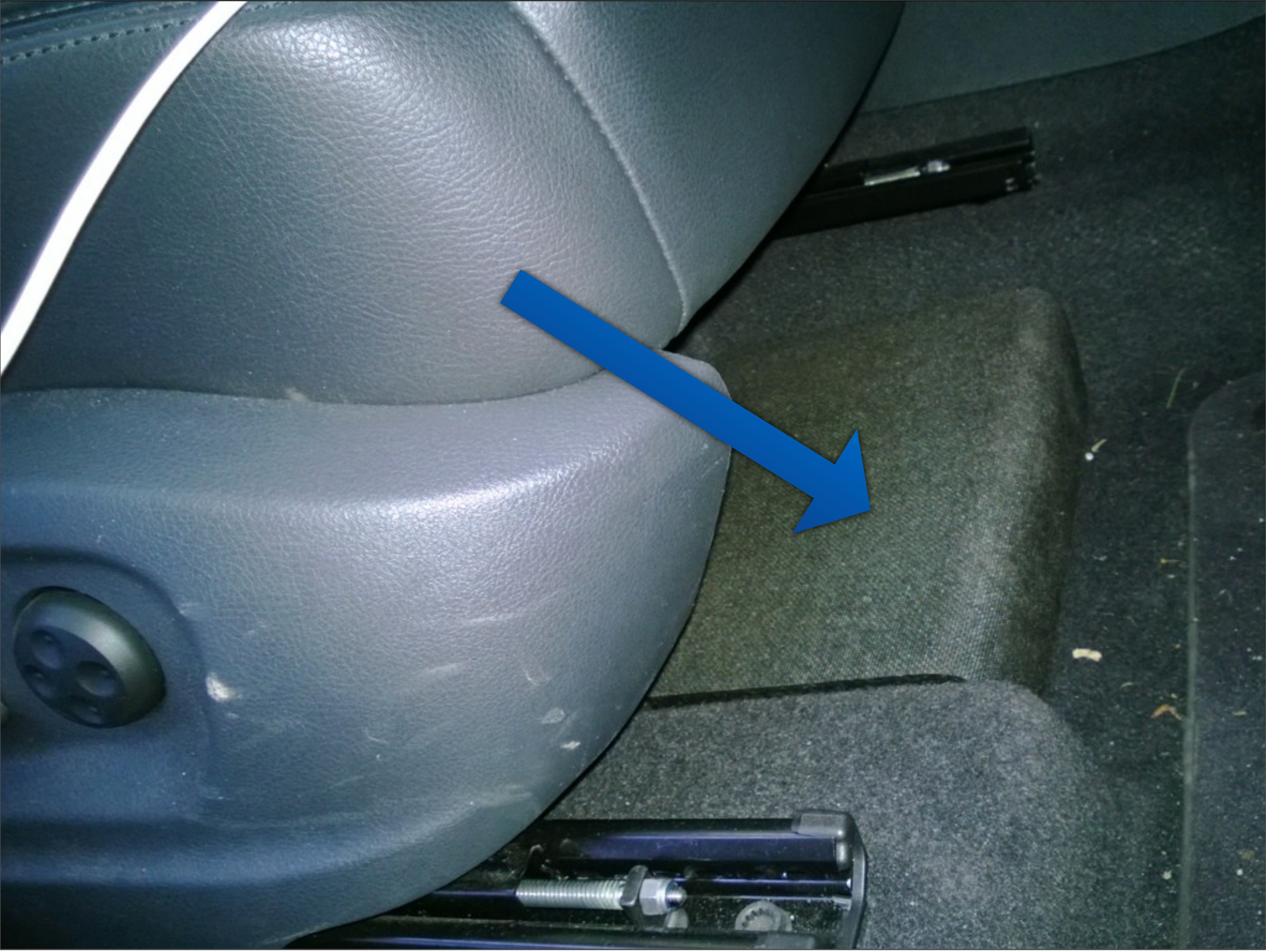
Infotainment CAN
(I-CAN)
100 kbit/s

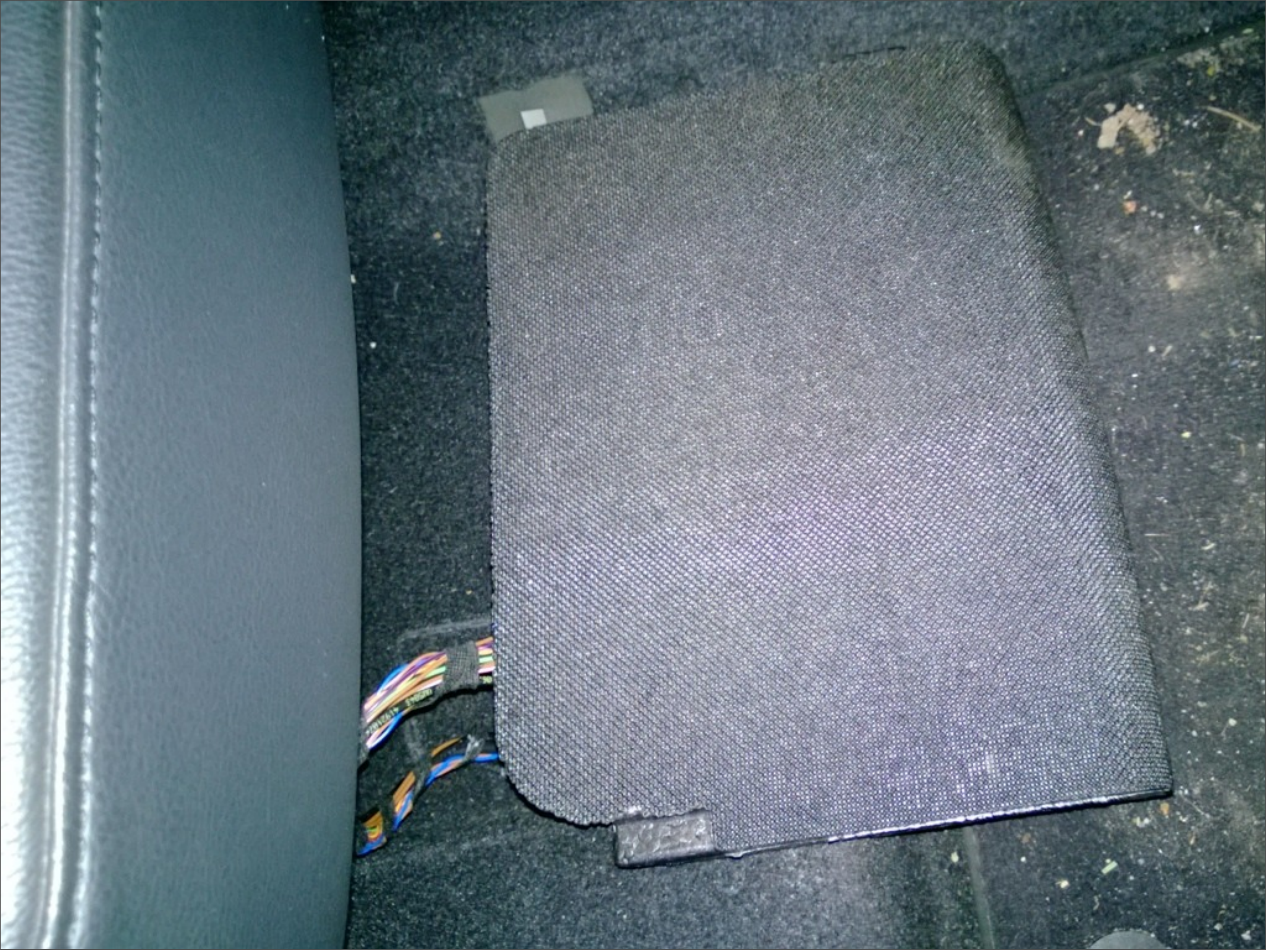
Audio Amp

Rear Cam Ctrl.

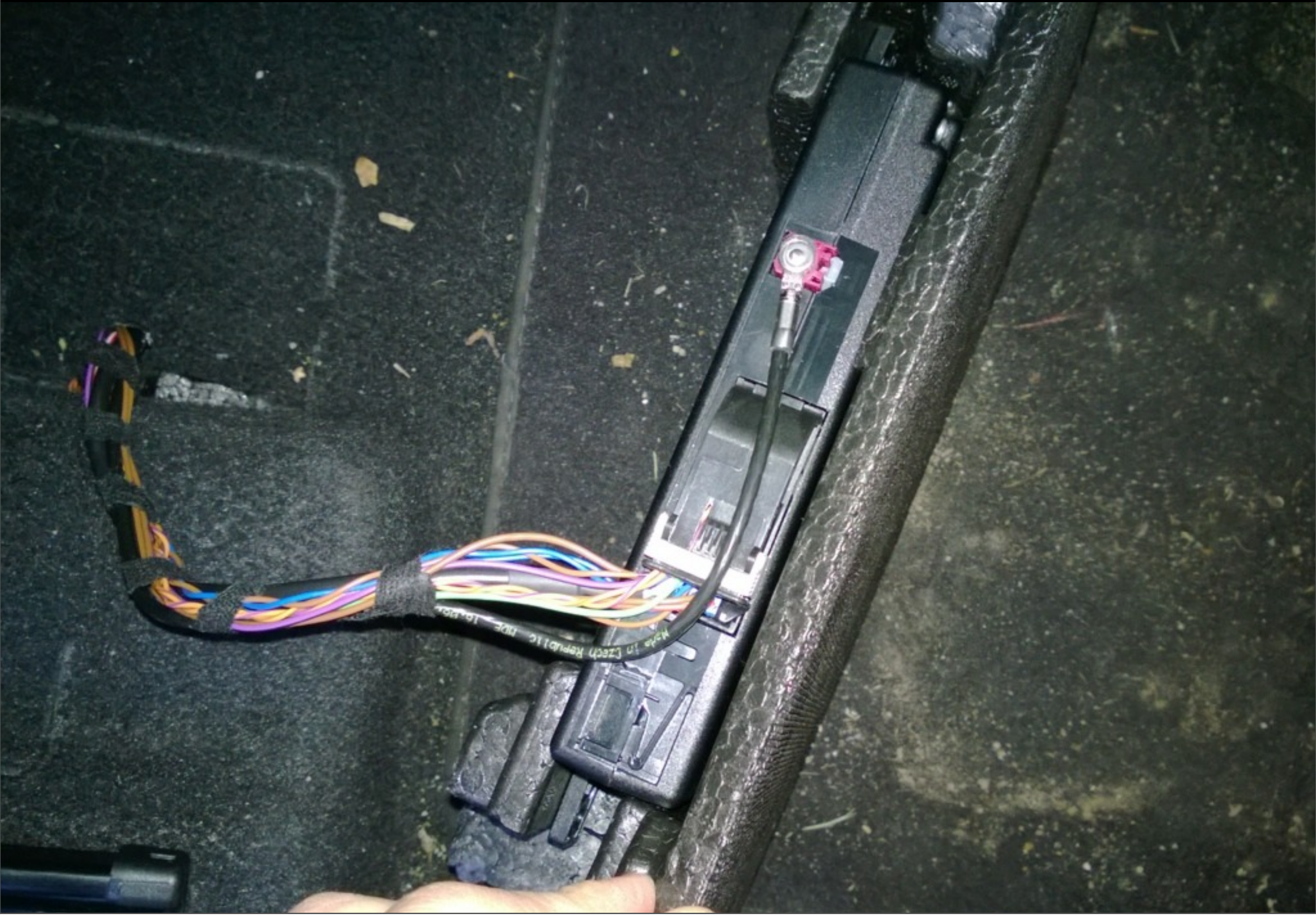
CAN Gateway

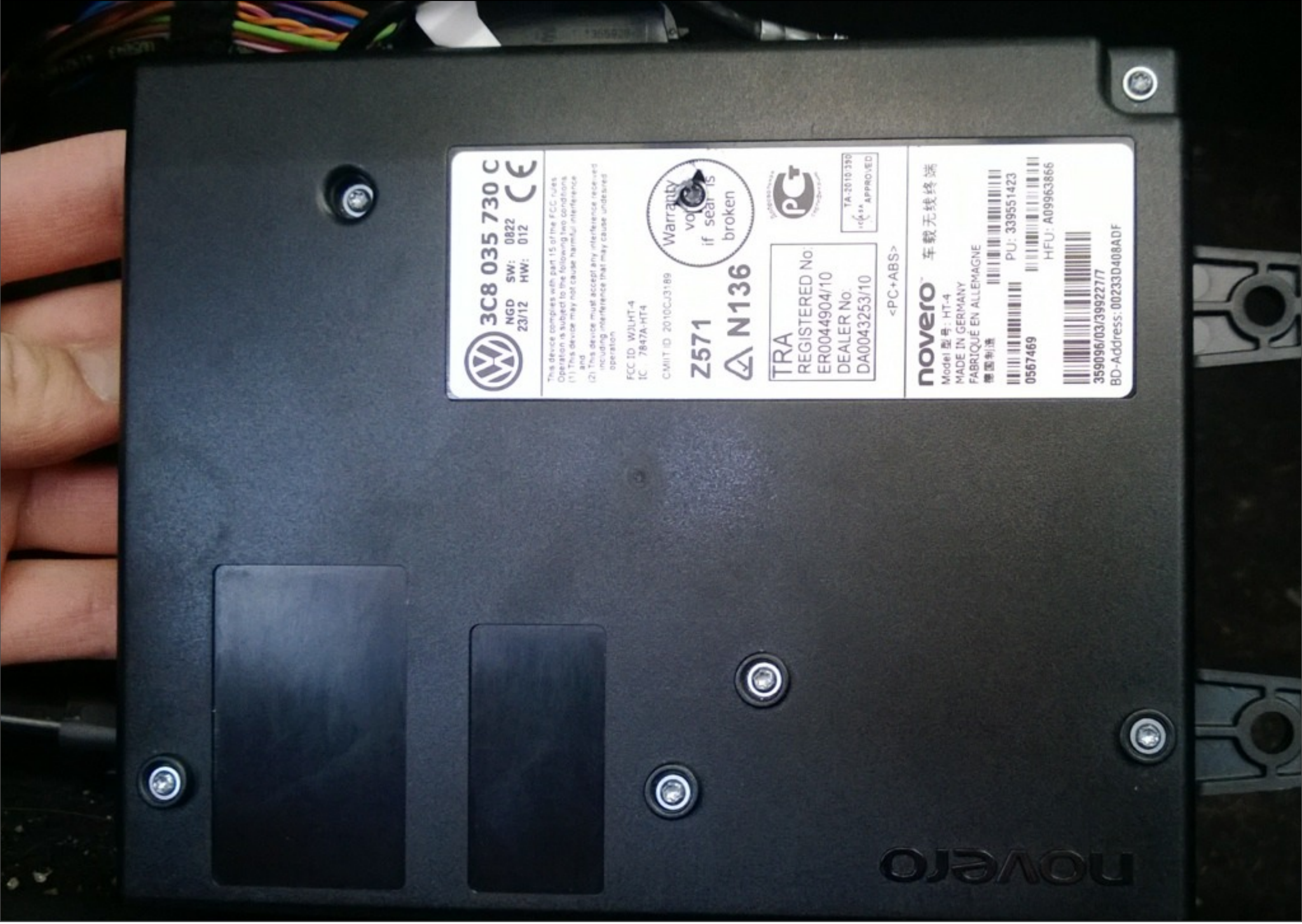
OBD-II





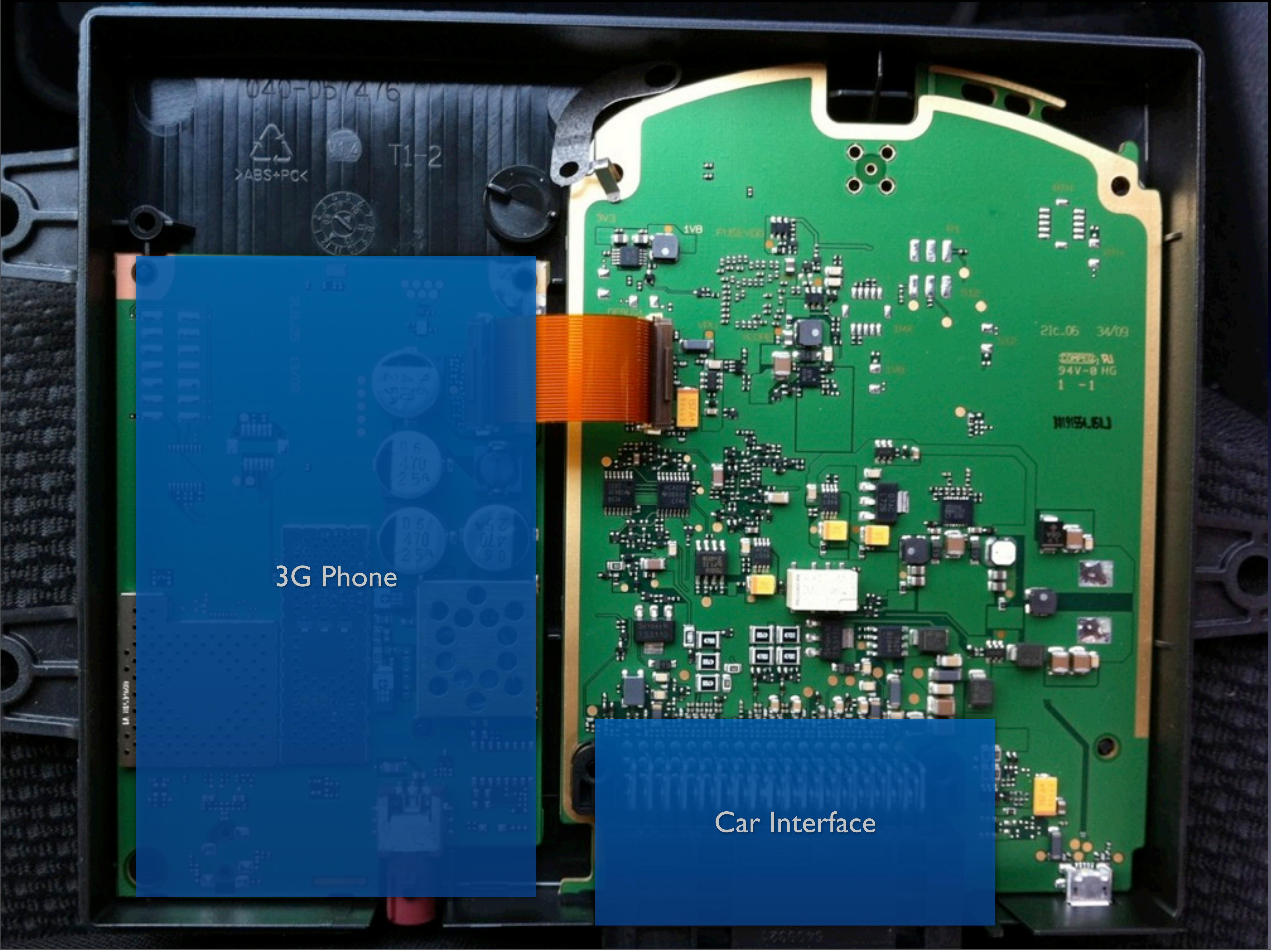
Saturday, December 28, 13

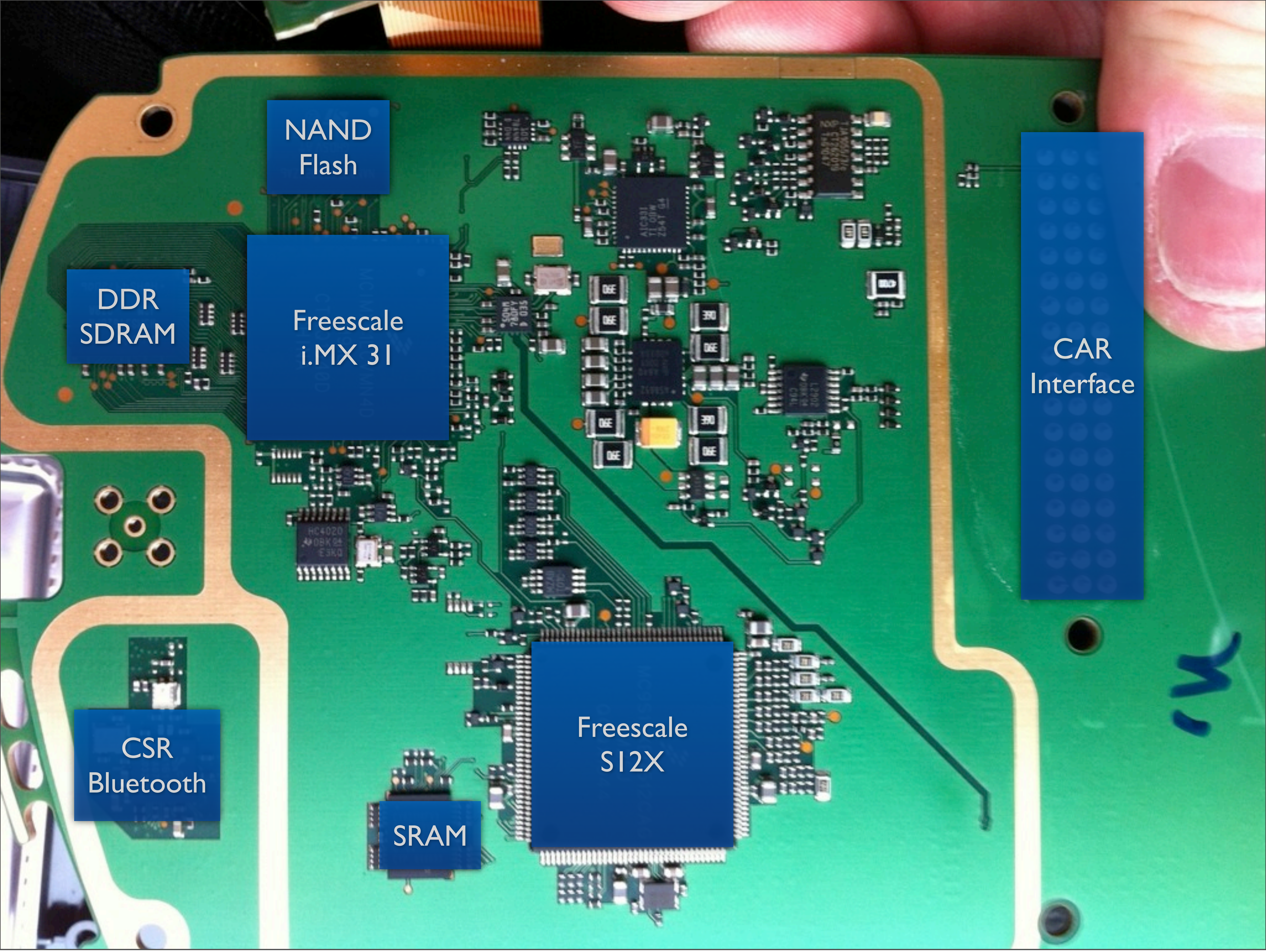

Made in Czech Republic


3C8 035 730 C
NGD 23/12
SW: 0822
HW: 012
This device complies with part 15 of the FCC rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference and (2) This device must accept any interference received including interference that may cause undesired operation.
FCC ID: W3LHT-4
IC: 7847A-HT4
CMIIT ID: 2010CJ3189
Z571
N136
Warranty void if seal is broken
TRA
REGISTERED No:
ER0044904/10
DEALER No:
DA0043253/10
TA-2010/390
APPROVED
<PC+ABS>
novero
车载无线终端
Model 型号: HT-4
MADE IN GERMANY
FABRIQUÉ EN ALLEMAGNE
德国制造
0567469
PU: 339551423
HFU: A09963866
359096/03/399227/7
BD-Address: 00233D408ADF
novero

novero GmbH

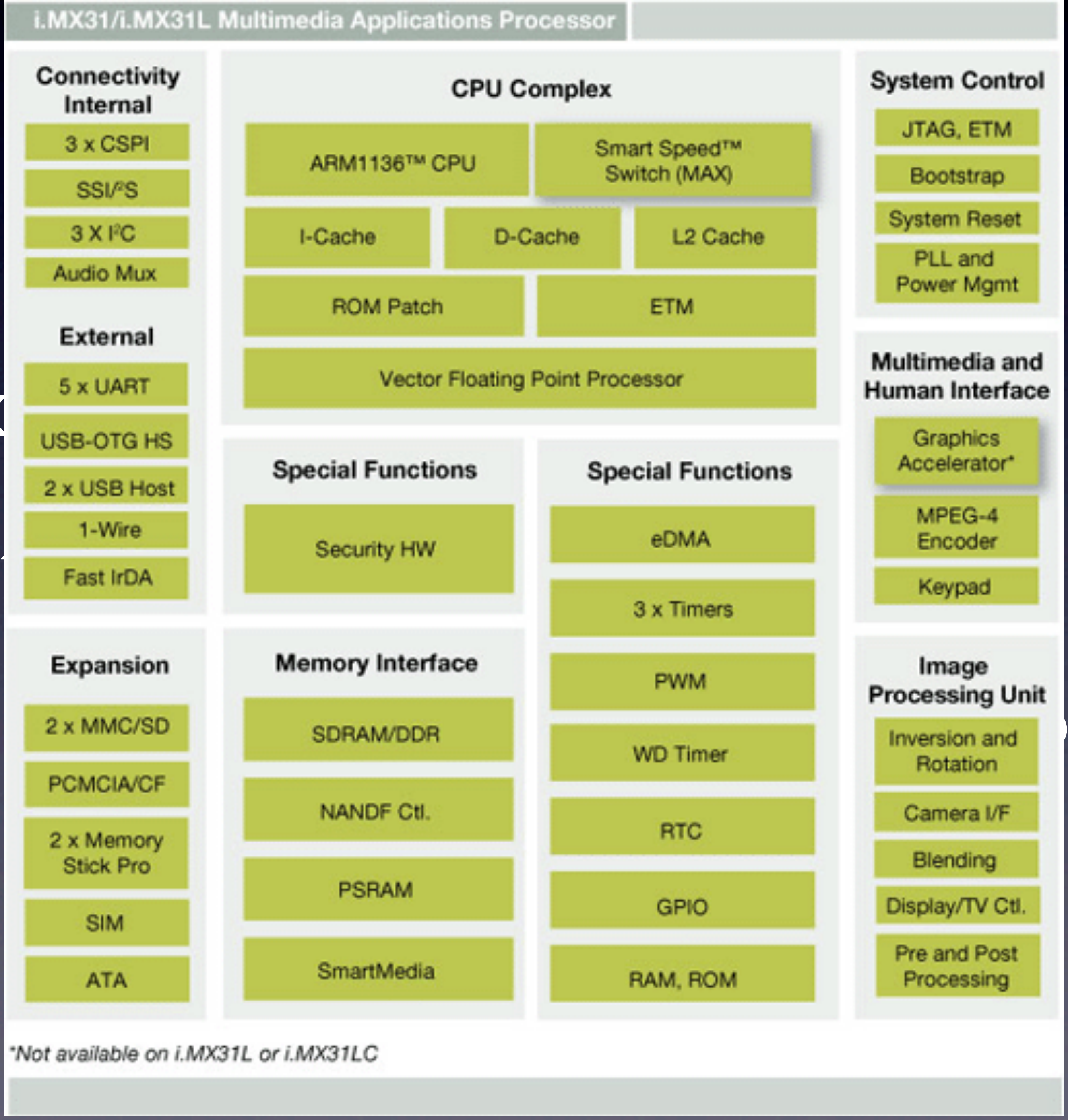
- For the Volkswagen-Group (VW/Audi/Seat/Skoda...), current Bluetooth Kits are built by Novero GmbH.
- Novero GmbH was Nokia's Automotive Group, but split out in 2008.


3G Phone
Car Interface


NAND
Flash
DDR
SDRAM
Freescale
i.MX 31
CAR
Interface
CSR
Bluetooth
SRAM
Freescale
S12X

i.MX 31

- ARM
- Linux
  - (Le
- Why wer?



i.MX31/i.MX31L Multimedia Applications Processor

Connectivity Internal
3 x CSPI
SSI/I^2S
3 X I^2C
Audio Mux

External
5 x UART
USB-OTG HS
2 x USB Host
1-Wire
Fast IrDA

Expansion
2 x MMC/SD
PCMCIA/CF
2 x Memory Stick Pro
SIM
ATA

CPU Complex
ARM1136™ CPU
Smart Speed™ Switch (MAX)
I-Cache
D-Cache
L2 Cache
ROM Patch
ETM
Vector Floating Point Processor

Special Functions
Security HW

Memory Interface
SDRAM/DDR
NANDF Ctl.
PSRAM
SmartMedia

Special Functions
eDMA
3 x Timers
PWM
WD Timer
RTC
GPIO
RAM, ROM

System Control
JTAG, ETM
Bootstrap
System Reset
PLL and Power Mgmt

Multimedia and Human Interface
Graphics Accelerator*
MPEG-4 Encoder
Keypad

Image Processing Unit
Inversion and Rotation
Camera I/F
Blending
Display/TV Ctl.
Pre and Post Processing

*Not available on i.MX31L or i.MX31LC

HT-4 Features (User)

- It bridges a remote SIM card to the mobile phone via rSAP.
- It allows a 3G DUN connection via Bluetooth.
- It does speech recognition and synth.
- It plays A2DP.

HT-4 Features (Hacker)

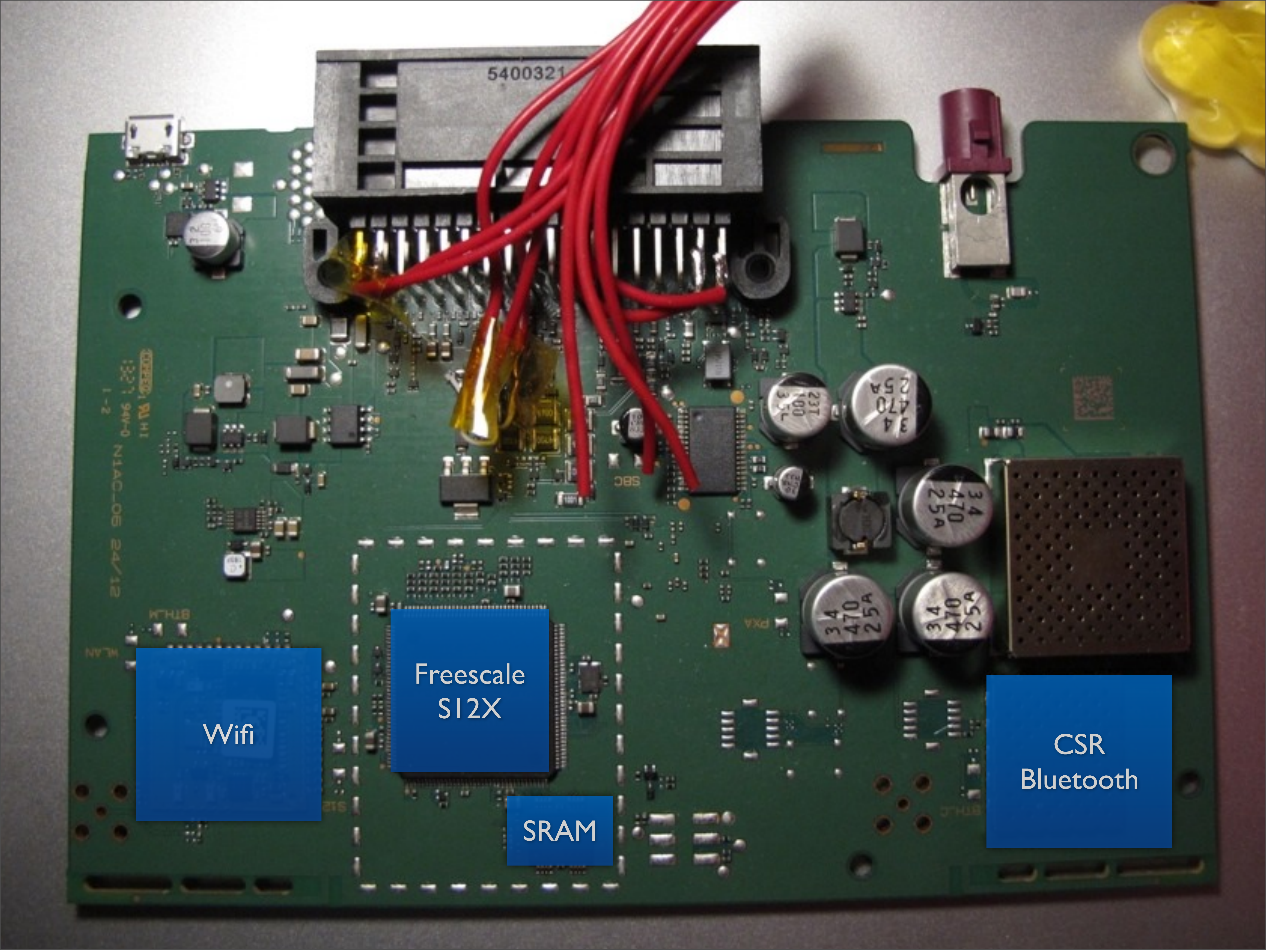
- RSAP: It has software control over the SIM card
- DUN: It has software control over the PPP session
- Voice Control: It has a lot of CPU power
- A2DP/HFP: It can play audio, and receive audio

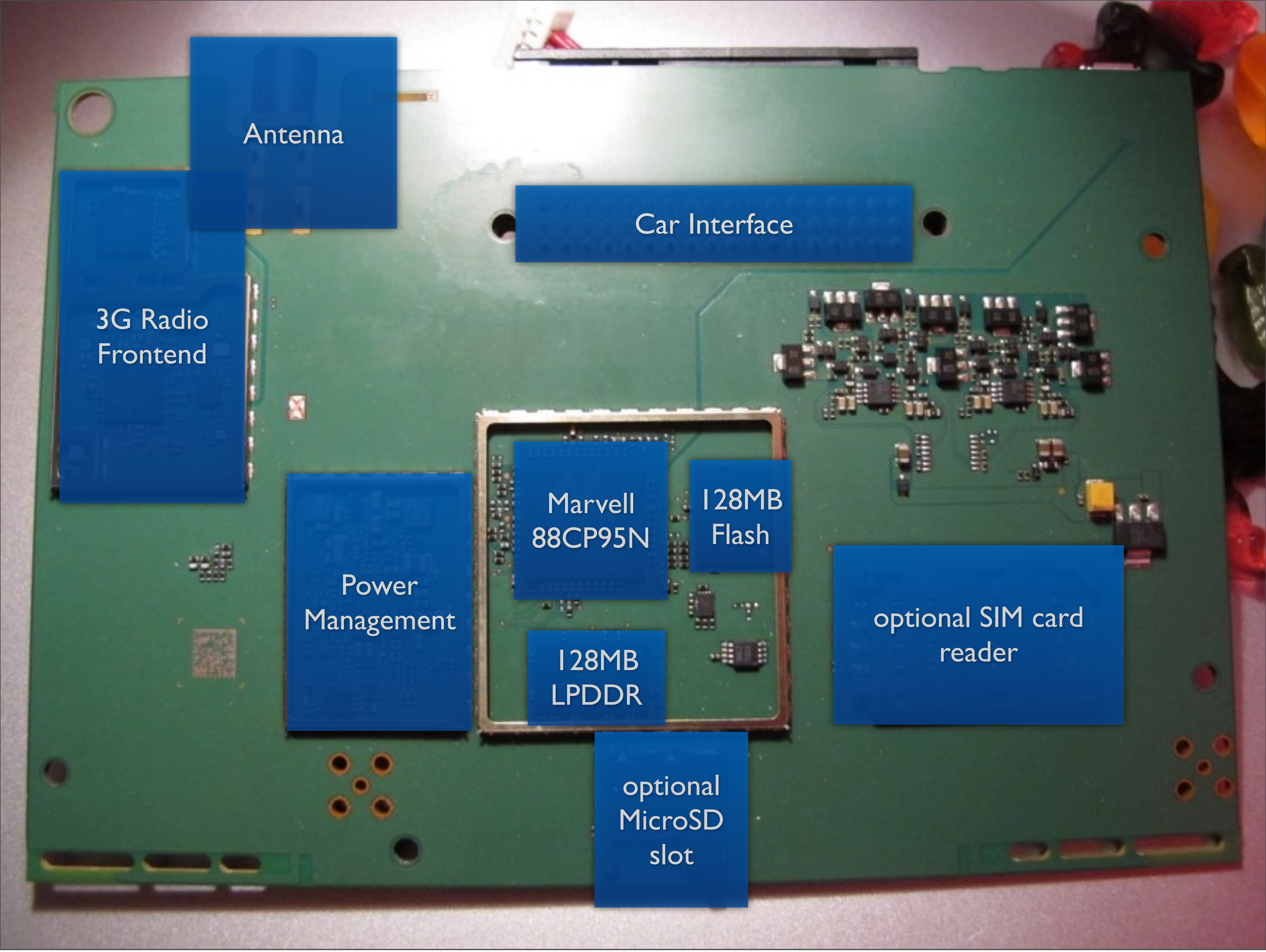
“Ha Car”

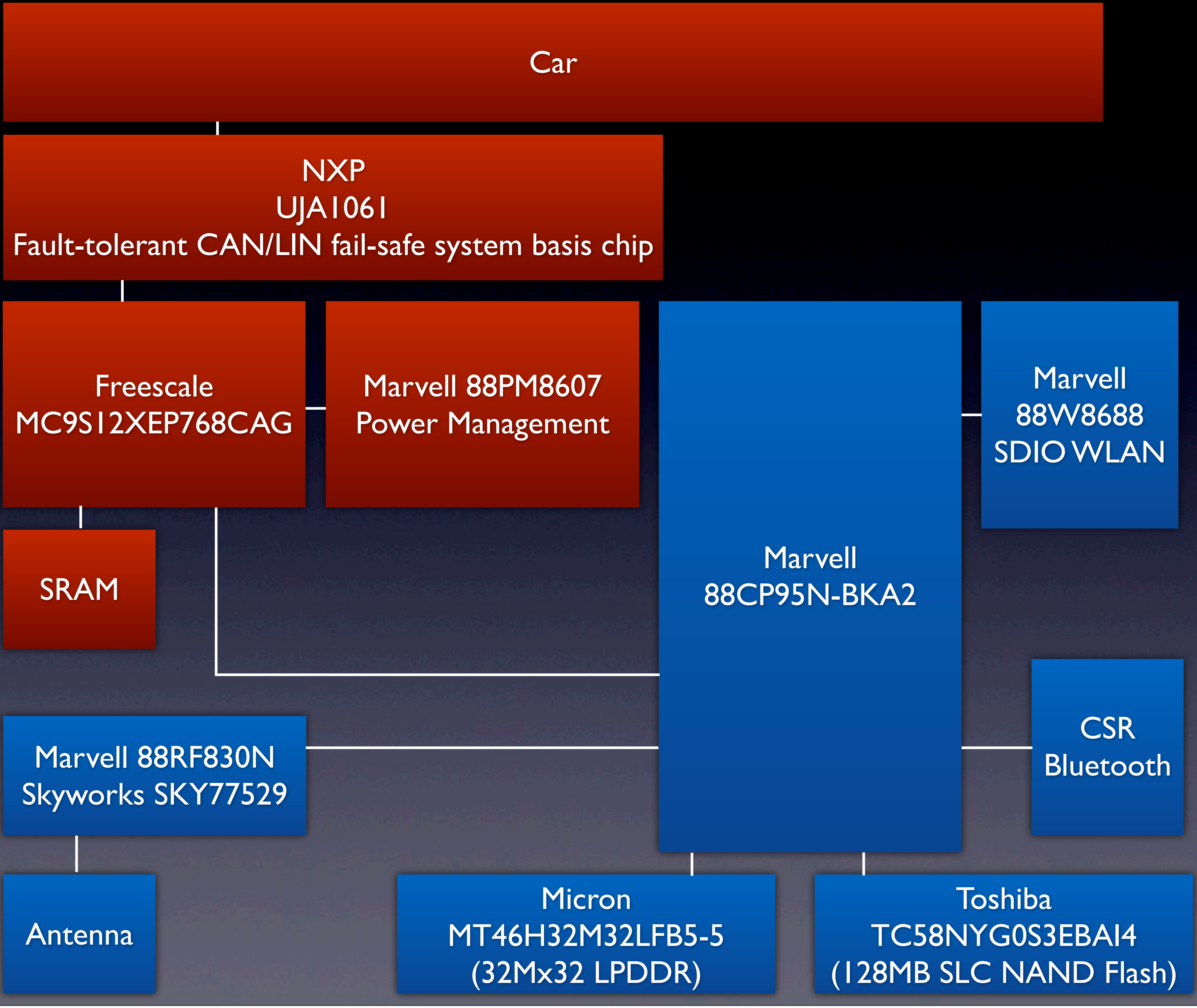
- “Let’s ju dhelp scope.” at?
  - It’s co
  - Trade ead battery
  - ... can
- Urg. Le

Novero HT-5

- Got a “spare” bluetooth kit, an HT-5
- HT-5 is used in MY’14 cars, can be retrofitted into any car with an HT-4
- Additional end-user features: WiFi sharing
- Additional hacker features: WiFi access point, builds up own PPP connection, Router+NAT (hence more likely to use real OS)
- Turns out - different hardware platform.

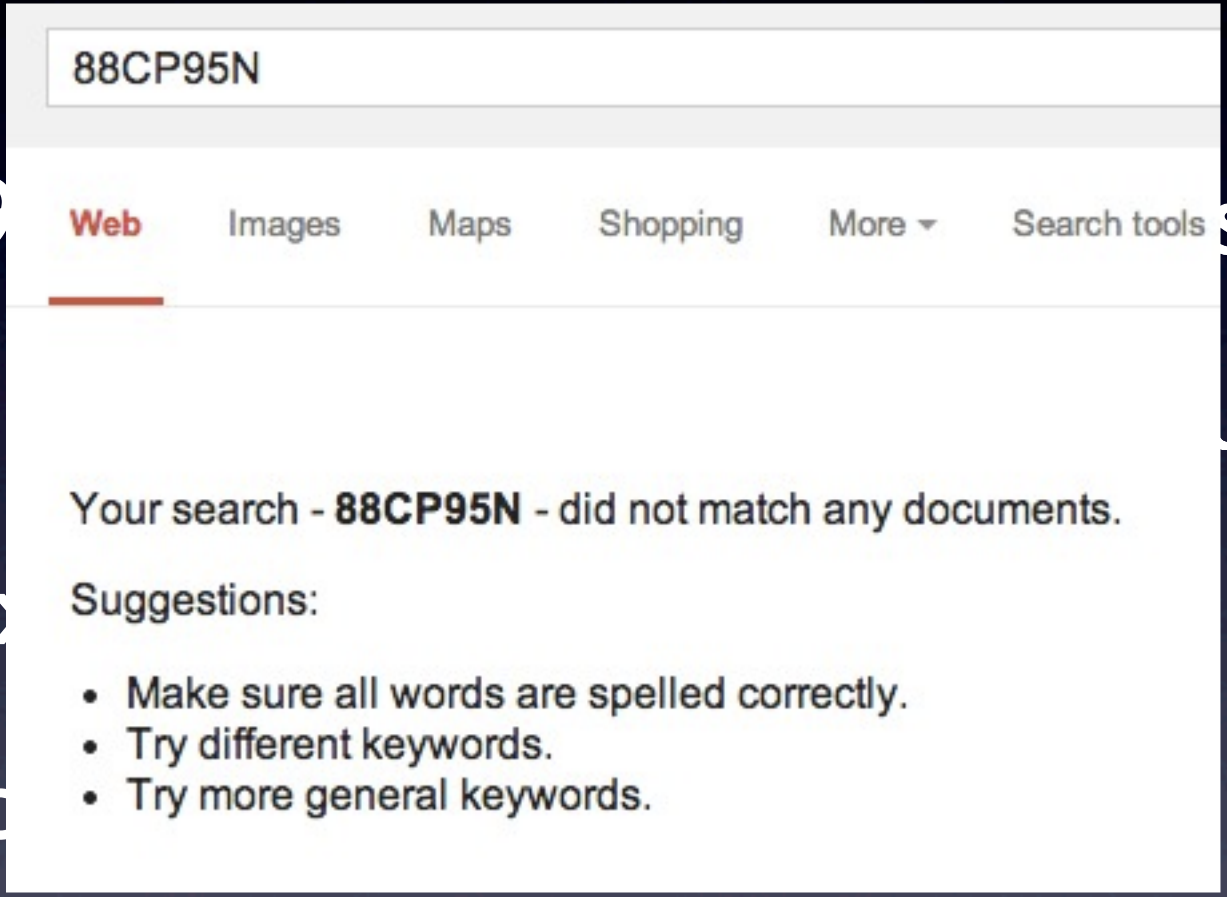

5400321
Wifi
Freescale
S12X
SRAM
CSR
Bluetooth


Antenna
Car Interface
3G Radio Frontend
Power Management
Marvell 88CP95N
128MB Flash
128MB LPDDR
optional SIM card reader
optional MicroSD slot


Car
NXP
UJA1061
Fault-tolerant CAN/LIN fail-safe system basis chip
Freescale
MC9S12XEP768CAG
Marvell 88PM8607
Power Management
Marvell
88W8688
SDIO WLAN
Marvell
88CP95N-BKA2
SRAM
CSR
Bluetooth
Marvell 88RF830N
Skyworks SKY77529
Antenna
Micron
MT46H32M32LFB5-5
(32Mx32 LPDDR)
Toshiba
TC58NYG0S3EBAI4
(128MB SLC NAND Flash)

Marvell 88CP95N

- 88CP9 s
- Seems unication
- Cortex
- 3G mo
- Found in some Android tablets



88CP95N

Your search - **88CP95N** - did not match any documents.

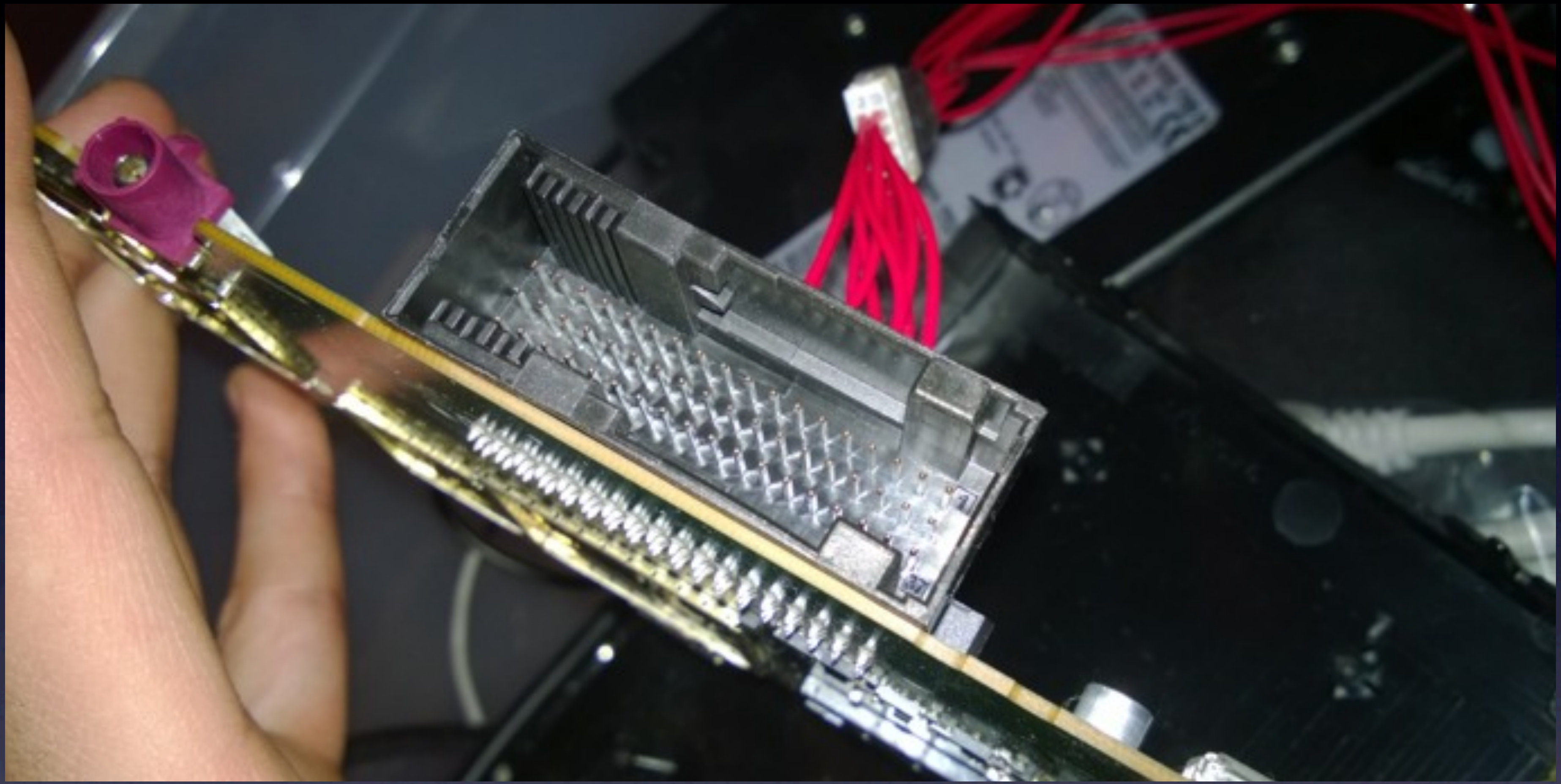
Suggestions:

- Make sure all words are spelled correctly.
- Try different keywords.
- Try more general keywords.

Let’s Boot it on My Desk.

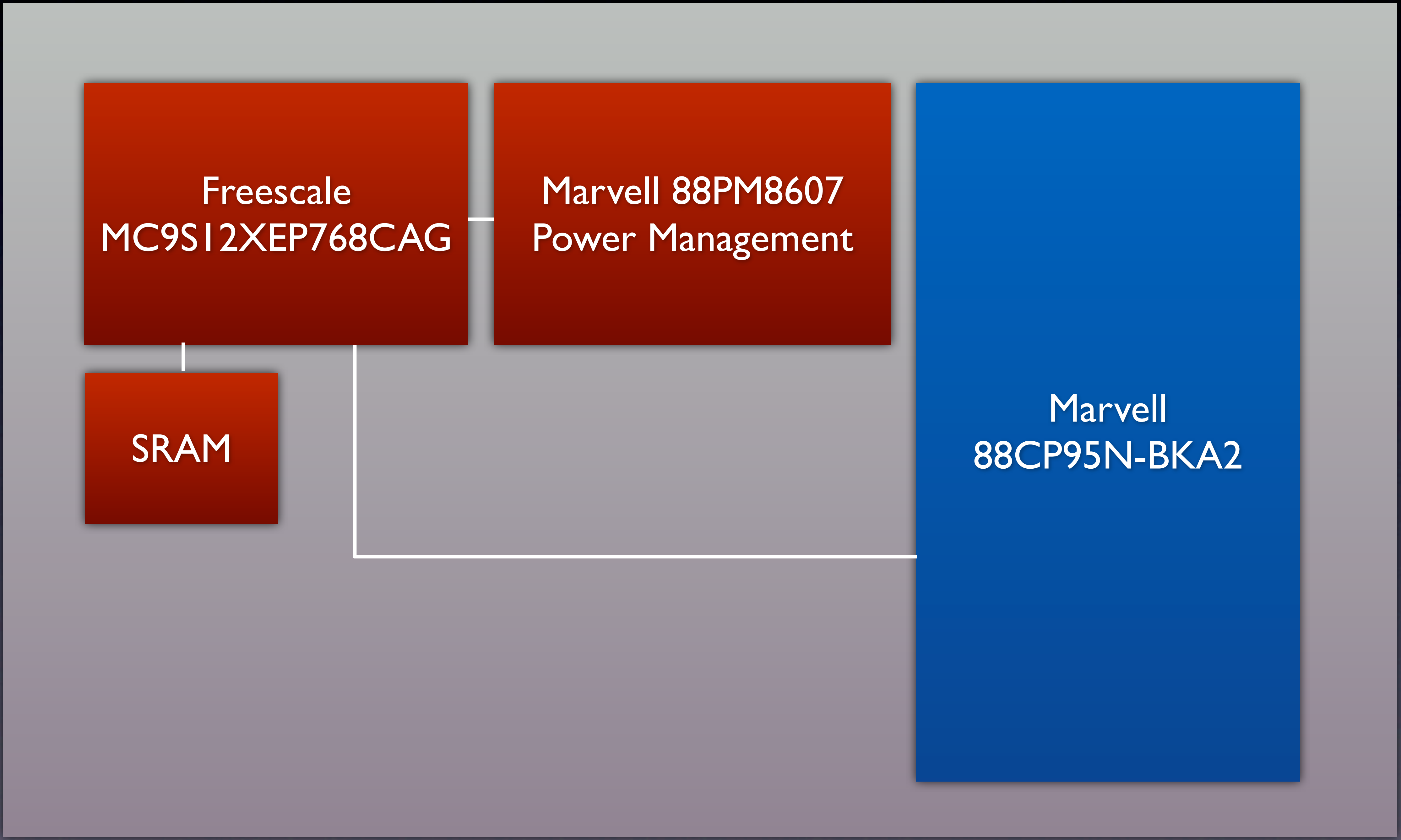
- Plan:
  - Connect 12V.
  - Hope it boots.
  - Find UART.
  - Hope it’s Linux.
  - Install Python.

Car Interface

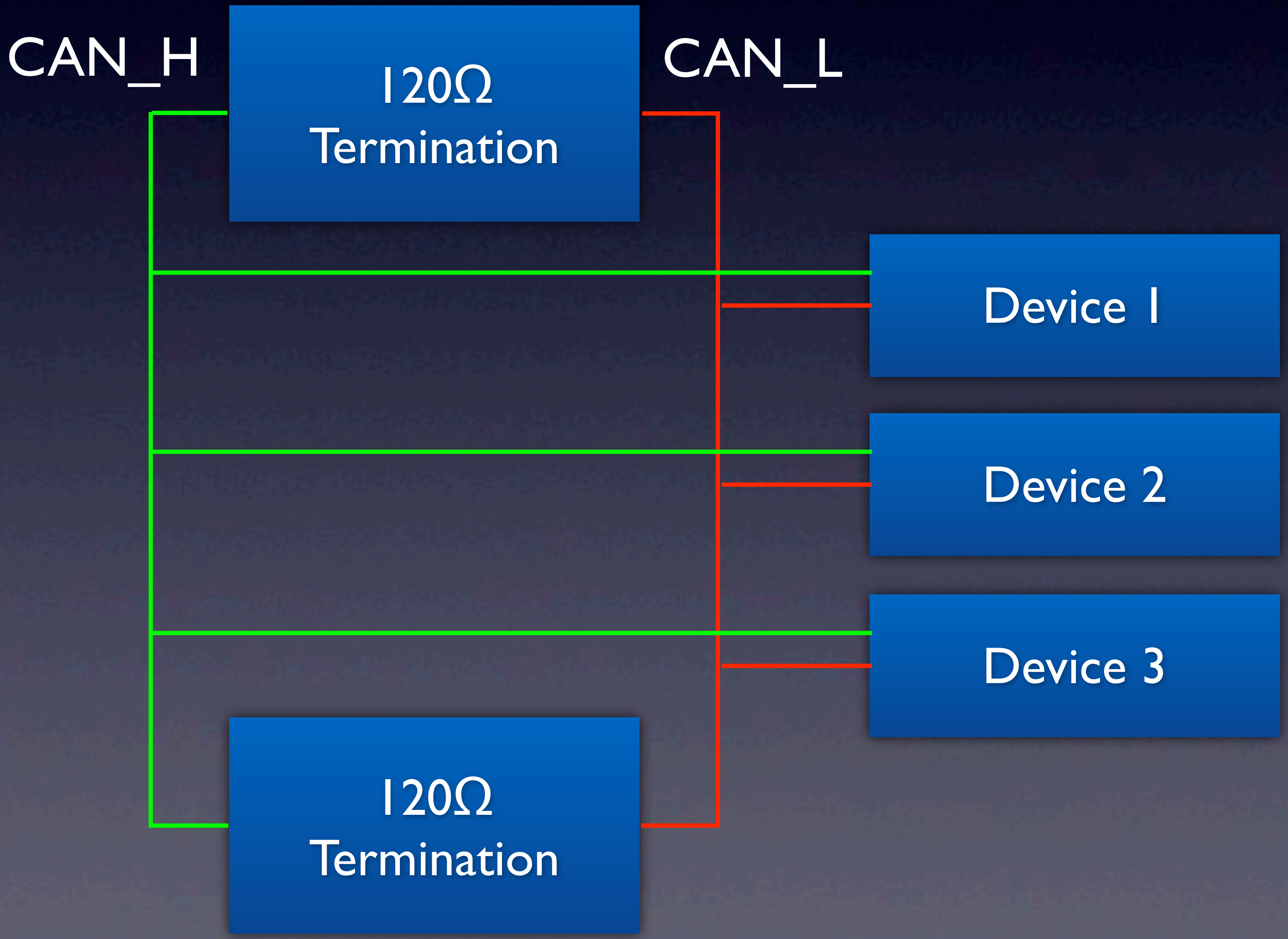


- 54 pin connector
- Pinout can be derived from car schematics
- GND, 12V, CAN_H/CAN_L available

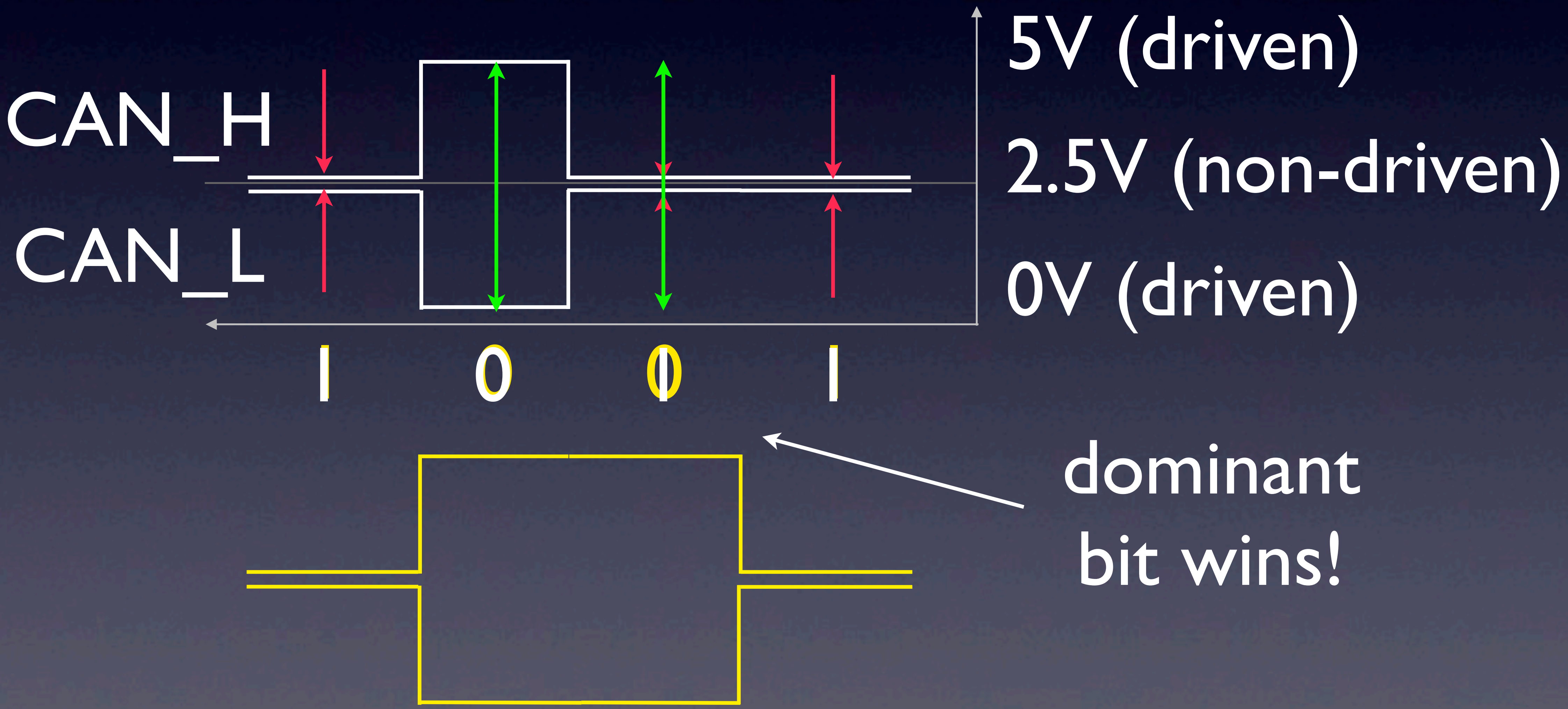
Powering Up


Freescale
MC9S12XEP768CAG
Marvell 88PM8607
Power Management
SRAM
Marvell
88CP95N-BKA2

CAN Bus Crash Course



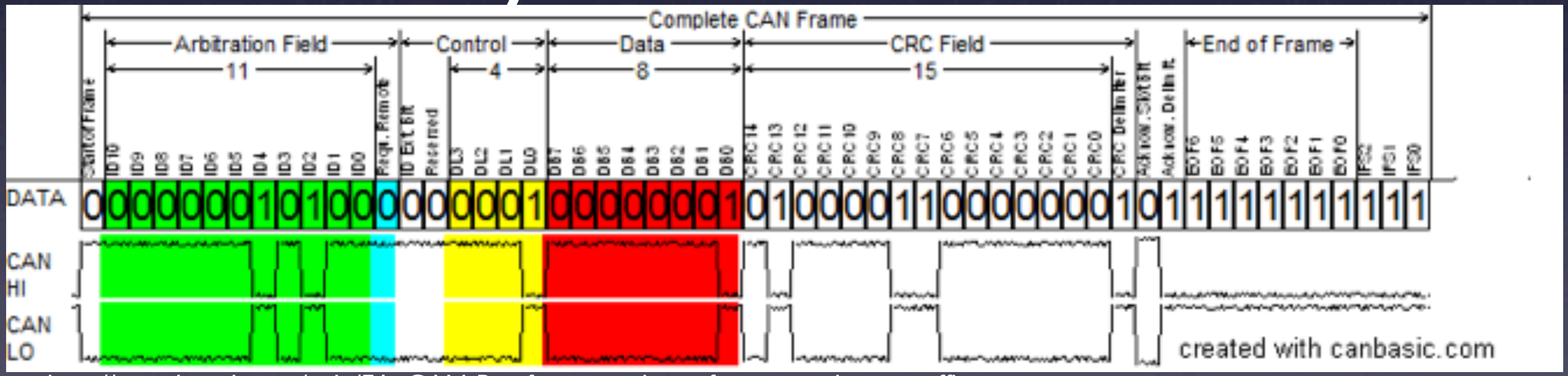
CAN Bus Crash Course



0 - Dominant   1 - Recessive

CAN Bus Crash Course

- Base frame format:
  - 11-bit Identifier (also used for priority)
  - 0 to 8 bytes of data



http://en.wikipedia.org/wiki/File:CAN-Bus-frame_in_base_format_without_stuffbits.png

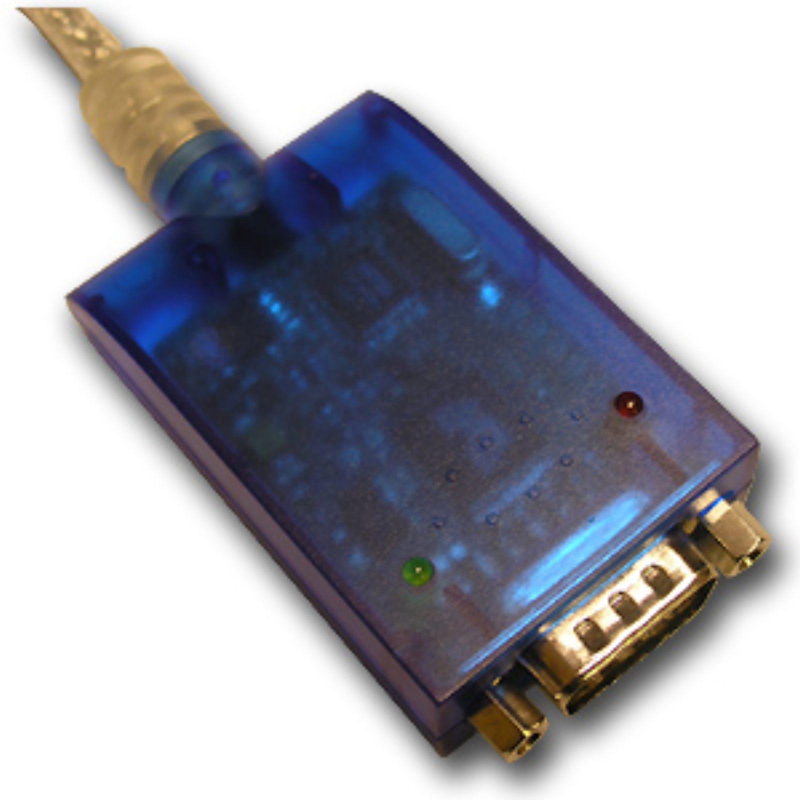
  - And lots of other stuff the Transfer Layer cares about (but we don't).

Make This Work kthx.

- Revised Plan:
  - Connect CAN while device in car
  - Capture CAN traffic
  - On desk, replay CAN traffic
  - Hope it boots.
  - UART, Linux, Python etc.

CAN PC Interface

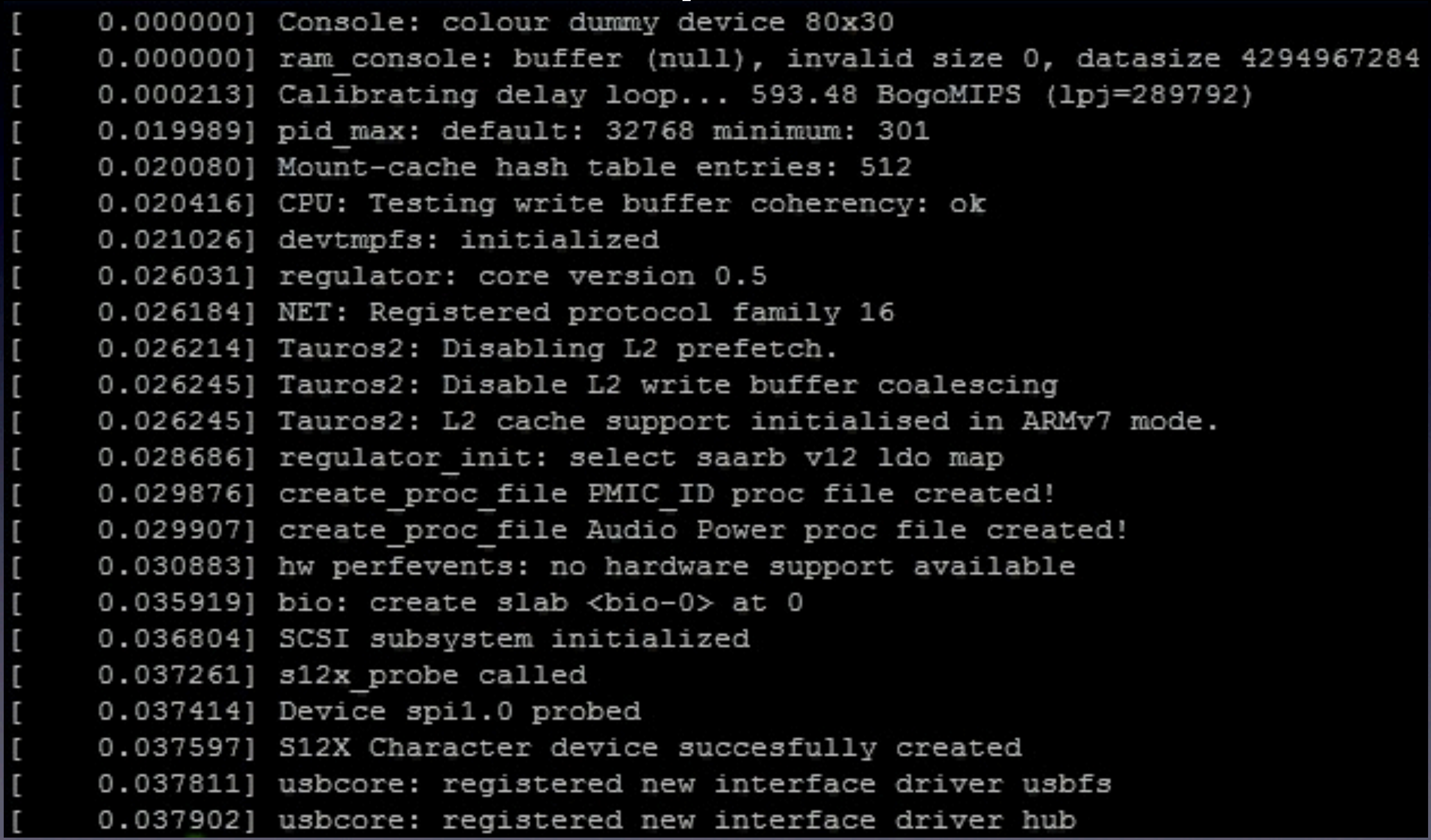
- “Any” CAN interface adapter will work.
- I used a LA , which is... okay.
- (Has a rath e limit, but OK for our pur
- Any microc tive CAN support will work, but please don’t try to bitbang.



Emulating Vehicle Power

- Also called “S-Kontakt” or “KL15” (Klemme 15, DIN 72552)
- Let’s try!
- Device draws reasonable power now!
- Isolated CAN message:
  - ID=661, “03 00 00 00 00 00 00 00”

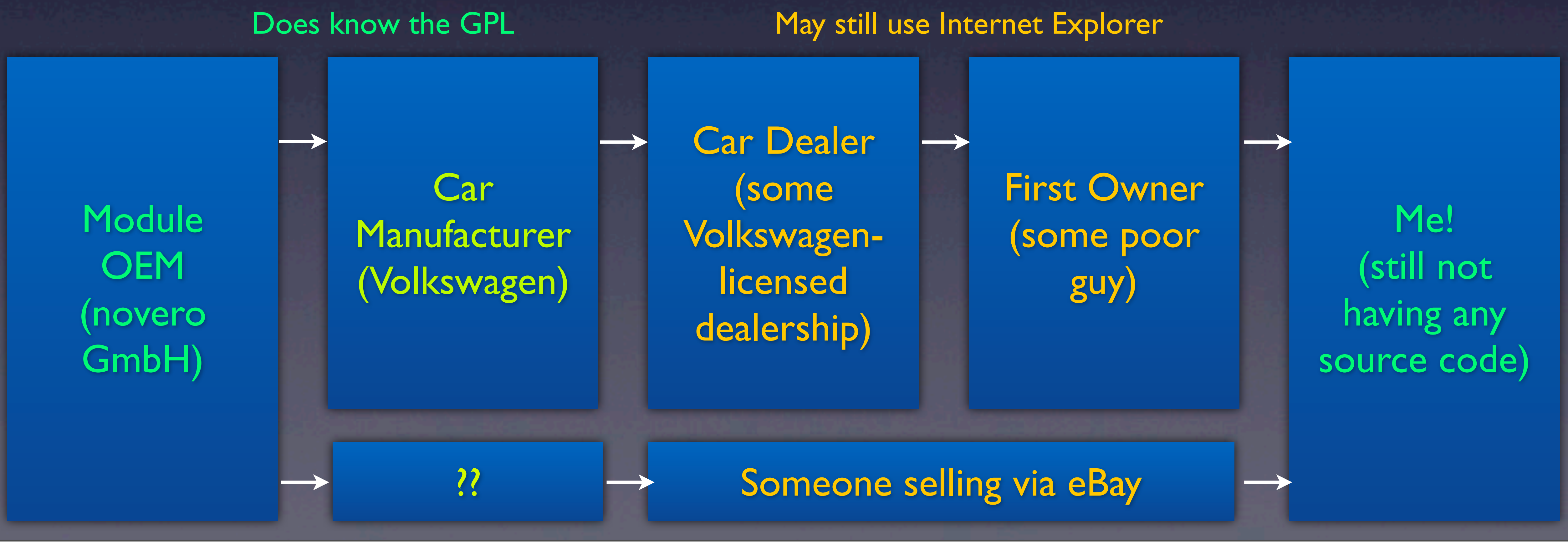
Now, WTF?



```
[    0.000000] Console: colour dummy device 80x30
[    0.000000] ram_console: buffer (null), invalid size 0, datasize 4294967284
[    0.000213] Calibrating delay loop... 593.48 BogoMIPS (lpj=289792)
[    0.019989] pid_max: default: 32768 minimum: 301
[    0.020080] Mount-cache hash table entries: 512
[    0.020416] CPU: Testing write buffer coherency: ok
[    0.021026] devtmpfs: initialized
[    0.026031] regulator: core version 0.5
[    0.026184] NET: Registered protocol family 16
[    0.026214] Tauros2: Disabling L2 prefetch.
[    0.026245] Tauros2: Disable L2 write buffer coalescing
[    0.026245] Tauros2: L2 cache support initialised in ARMv7 mode.
[    0.028686] regulator_init: select saarb v12 ldo map
[    0.029876] create_proc_file PMIC_ID proc file created!
[    0.029907] create_proc_file Audio Power proc file created!
[    0.030883] hw perfevents: no hardware support available
[    0.035919] bio: create slab <bio-0> at 0
[    0.036804] SCSI subsystem initialized
[    0.037261] s12x_probe called
[    0.037414] Device spi1.0 probed
[    0.037597] S12X Character device succesfully created
[    0.037811] usbcore: registered new interface driver usbfs
[    0.037902] usbcore: registered new interface driver hub
```

GPL

- Oh, why didn’t I see the GPL in the owner’s manual?
- More importantly: Who has to send me the source code?
- It’s complicated. But IANAL.



Does know the GPL

May still use Internet Explorer

Module OEM (novero GmbH) → Car Manufacturer (Volkswagen) → Car Dealer (some Volkswagen-licensed dealership) → First Owner (some poor guy) → Me! (still not having any source code)

Module OEM (novero GmbH) → ?? → Someone selling via eBay → Me! (still not having any source code)

Local Access == root?

- Boots into Linux - but we don’t know the root password. `$1$JN.iQytI$b1EbtEaRL2xSgZVri6dU/` if you have some spare time.
- Flash modification would help, but meh, BGA flash. Such an effort.
- U-Boot used in non-secure configuration (allows entering console by hammering ^C)
- Traditional “init=/bin/sh” trick should work?

Local Access == root!

- It uses initrd. So it's "rdinit=/bin/sh".
- Doesn't work. We don't know the FS layout. (And there's no `/dev/console`)
- U-Boot allows dumping memory, so we manually extract the initrd.
- Initscript, yay. Directly booting real rootfs, then `init=/bin/sh` works. *`mount -o remount,rw /; passwd`* and there we go.

Hardware Tricks

- HT-5: Micro-SD slot can be added and “just works”. Even easier!
- SIM slot (requires reconfiguration via Diag)
- USB OTG on ext. accessible Micro-USB
  - Firmware Upgrade, Diagnostics
  - HT-4 host-only, HT-5 is real OTG
  - serial by default, can be changed to usb-ethernet

Emulating The Display

- To hack on the desk, we need to emulate the car.
- Or at least:
  - Steering wheel buttons
  - Display
  - Everything to keep it alive

S12X

- I was hoping to find CAN messages arriving in Linux, and then being parsed by a binary with symbols and excessive debug spew. No such luck.
- Finding "other" end on device for CAN messages proved... interesting.
- The S12X abstracts all of that to a very high level.

BAP

- Very simplified:
  - BAP allows a control unit to provide “values” (like “screen content”), and a display unit to use these “values”.
  - BAP caches and synchronizes changes, and manages lifecycle (heartbeat, errors) in a well-defined way.

BAP

- “Bedien- und Anzeigeprotokoll” - German Engineering is in da house!
- Hard to find anything more technical than THIS, but it has OSI layers, so it must be good:

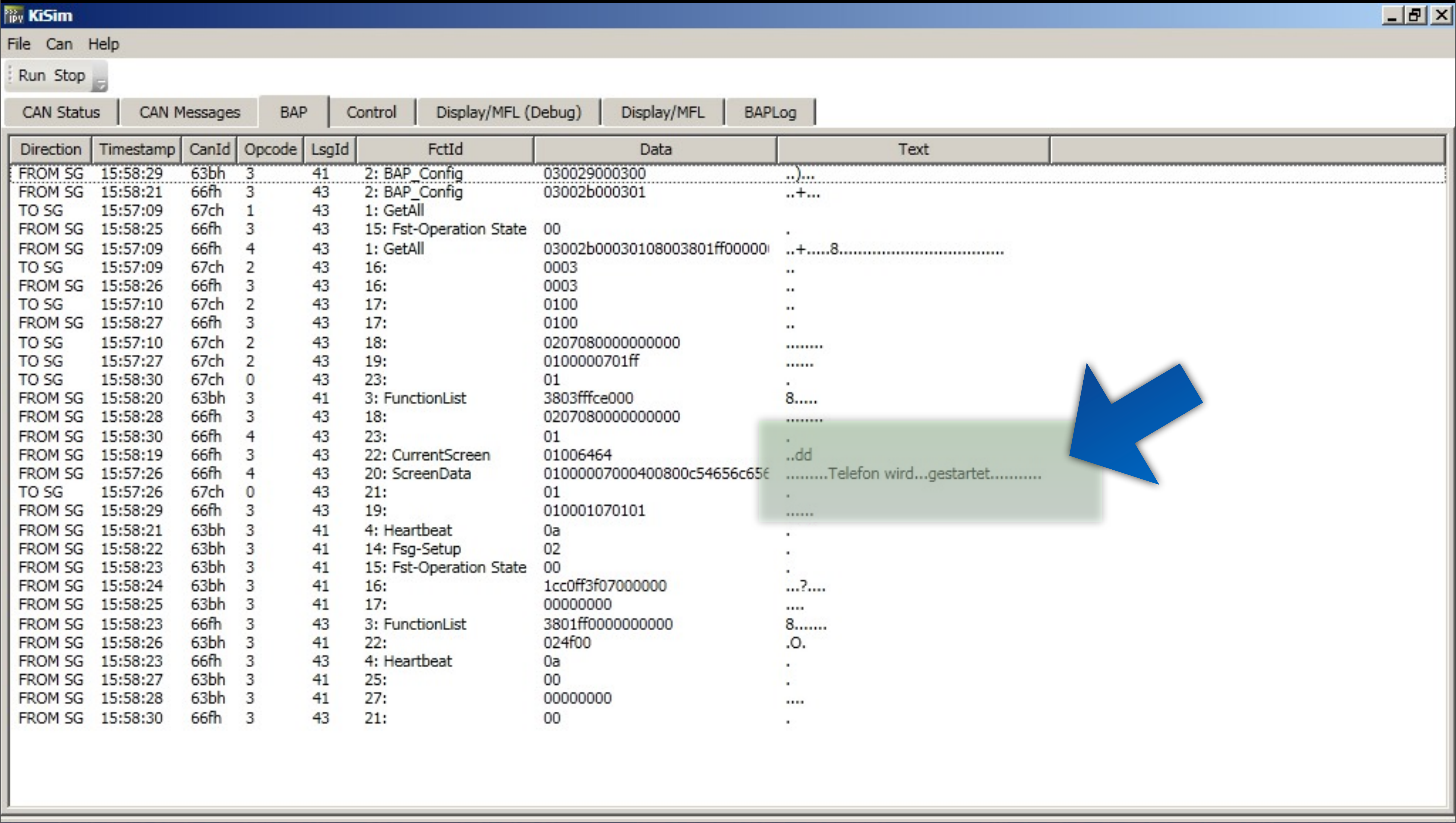
OSI Layer	BAP Name	Example
Application Layer	BAP Application Layer	“Value X is Y”
Presentation Layer		
Session Layer	BAP Protocol Layer	“Update Value X”
Transport Layer	BAP Communication Layer	Single BAP messages
Network Layer		
Data Link Layer		
Physical Layer	CAN or LIN	CAN messages

Reversing BAP

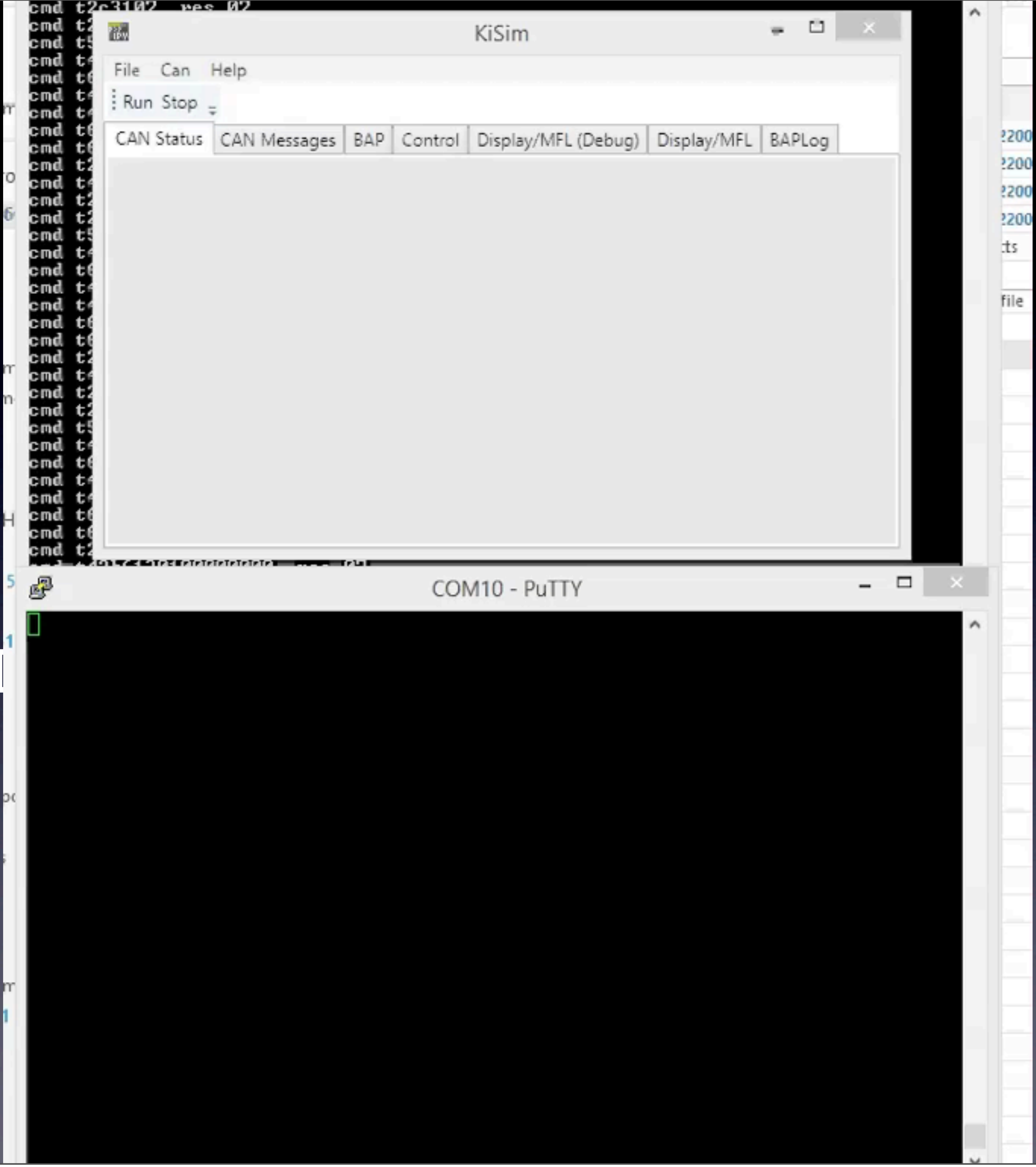
ID	Data
62c	80374c0103002f00
62c	c0030108003803cf
62c	c1ff000000000a02
62c	c20008001cc00b00
62c	c300000000030001
62c	c40000ffffffffff
62c	c5ffff00000000ff
62c	c6ffff0100000200
62c	c70000
...	...
62c	0c0203002f000301

```
Opcode=4
LsgId=48
FctId=1

03002f0003010800
3803cfff00000000
0a020008001cc00b
0000000000030001
0000ffffffffffff
ff00000000ffffff
01000002000000
```

```
Opcode=0
LsgId=48
FctId=2

03002f000301
```



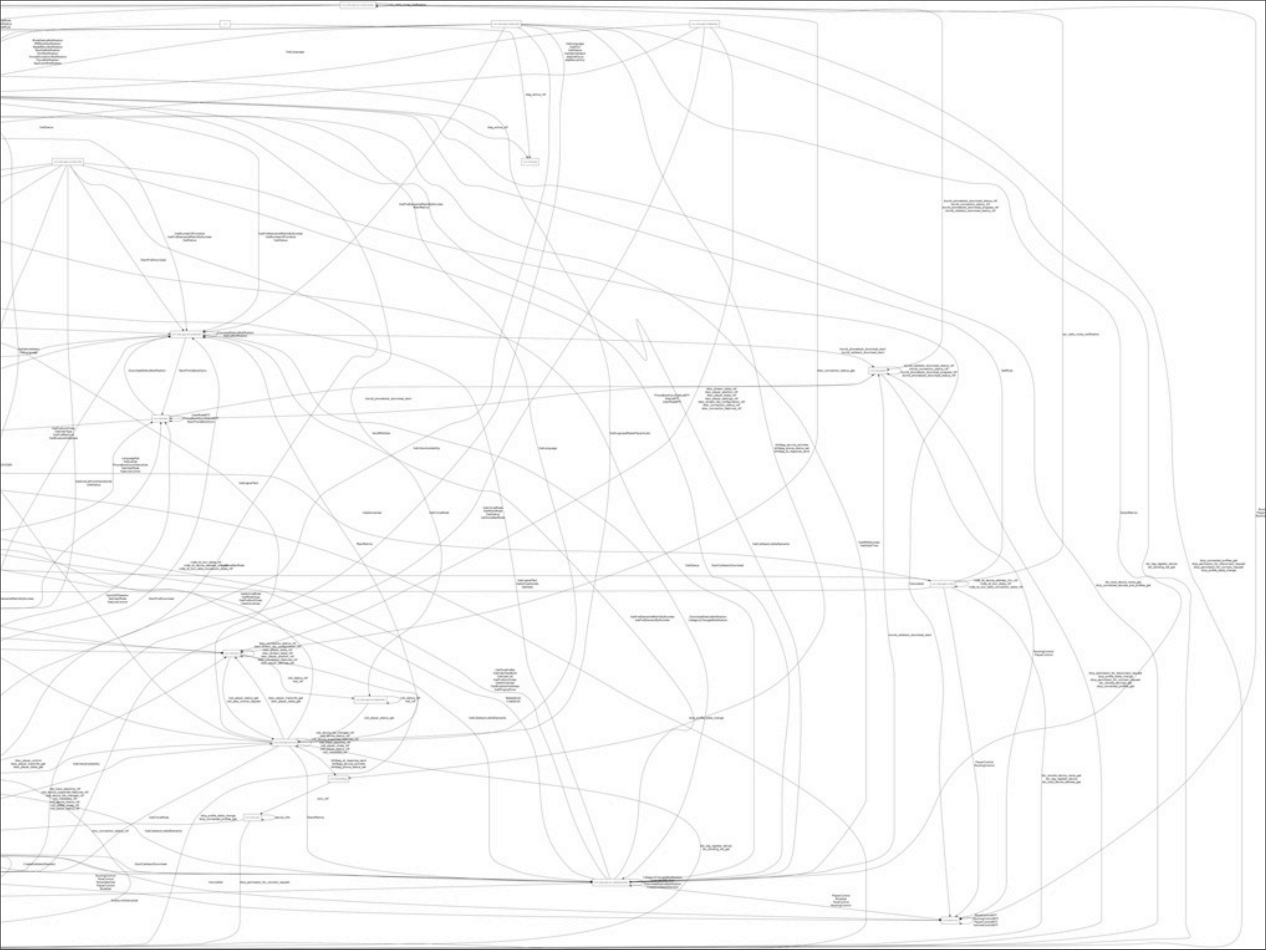
Direction	Timestamp	CanId	Opcode	LsgId	FctId	Data	Text
FROM SG	15:58:29	63bh	3	41	2: BAP_Config	030029000300	..)...
FROM SG	15:58:21	66fh	3	43	2: BAP_Config	03002b000301	..+...
TO SG	15:57:09	67ch	1	43	1: GetAll		
FROM SG	15:58:25	66fh	3	43	15: Fst-Operation State	00	.
FROM SG	15:57:09	66fh	4	43	1: GetAll	03002b00030108003801ff00000	..+.....8.................................
TO SG	15:57:09	67ch	2	43	16:	0003	..
FROM SG	15:58:26	66fh	3	43	16:	0003	..
TO SG	15:57:10	67ch	2	43	17:	0100	..
FROM SG	15:58:27	66fh	3	43	17:	0100	..
TO SG	15:57:10	67ch	2	43	18:	0207080000000000	....
TO SG	15:57:27	67ch	2	43	19:	0100000701ff	......
TO SG	15:58:30	67ch	0	43	23:	01	.
FROM SG	15:58:20	63bh	3	41	3: FunctionList	3803fffce000	8.....
FROM SG	15:58:28	66fh	3	43	18:	0207080000000000	....
FROM SG	15:58:30	66fh	4	43	23:	01	.
FROM SG	15:58:19	66fh	3	43	22: CurrentScreen	01006464	..dd
FROM SG	15:57:26	66fh	4	43	20: ScreenData	01000007000400800c54656c656	.........Telefon wird...gestartet...........
TO SG	15:57:26	67ch	0	43	21:	01	.
FROM SG	15:58:29	66fh	3	43	19:	010001070101	......
FROM SG	15:58:21	63bh	3	41	4: Heartbeat	0a	.
FROM SG	15:58:22	63bh	3	41	14: Fsg-Setup	02	.
FROM SG	15:58:23	63bh	3	41	15: Fst-Operation State	00	.
FROM SG	15:58:24	63bh	3	41	16:	1cc0ff3f07000000	...?....
FROM SG	15:58:25	63bh	3	41	17:	00000000	....
FROM SG	15:58:23	66fh	3	43	3: FunctionList	3801ff0000000000	8.......
FROM SG	15:58:26	63bh	3	41	22:	024f00	.O.
FROM SG	15:58:23	66fh	3	43	4: Heartbeat	0a	.
FROM SG	15:58:27	63bh	3	41	25:	00	.
FROM SG	15:58:28	63bh	3	41	27:	00000000	....
FROM SG	15:58:30	66fh	3	43	21:	00	.

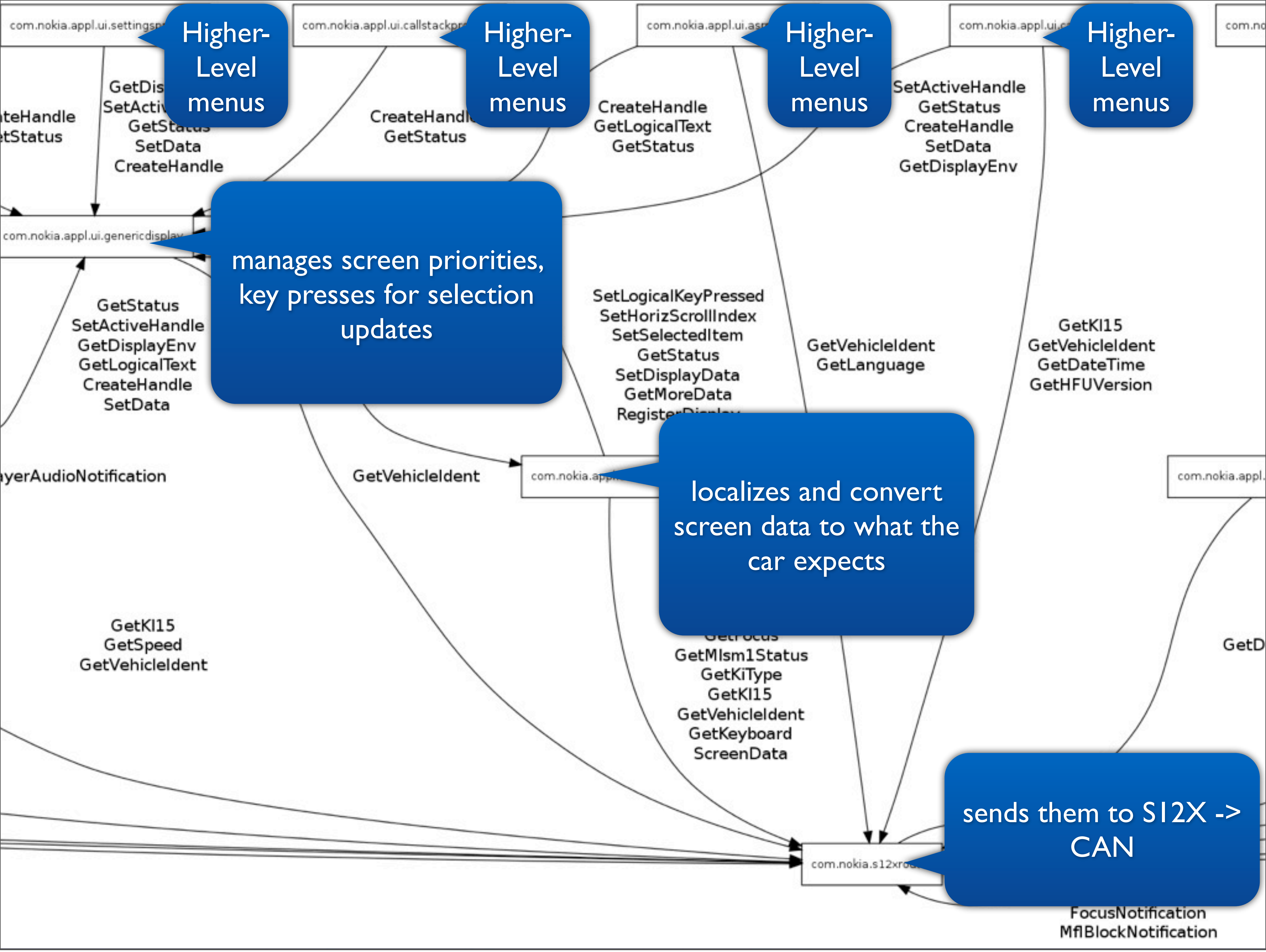


```
cmd t2c3102  res 02
cmd t2
cmd t5
cmd t4
cmd t6
cmd t4
cmd t4
cmd t6
cmd t6
cmd t2
cmd t4
cmd t2
cmd t2
cmd t5
cmd t4
cmd t6
cmd t4
cmd t4
cmd t6
cmd t6
cmd t2
cmd t4
cmd t2
cmd t2
cmd t5
cmd t4
cmd t6
cmd t4
cmd t4
cmd t6
cmd t6
cmd t2
```

D-Bus

- D-Bus used very extensively
- S12X messages are posted on D-Bus
- We see screen updates, key presses, bluetooth events...
- Only very high level view of CAN, though.
- NAVPOS, Speed signals are visible.





Higher-Level menus

Higher-Level menus

Higher-Level menus

Higher-Level menus

manages screen priorities, key presses for selection updates

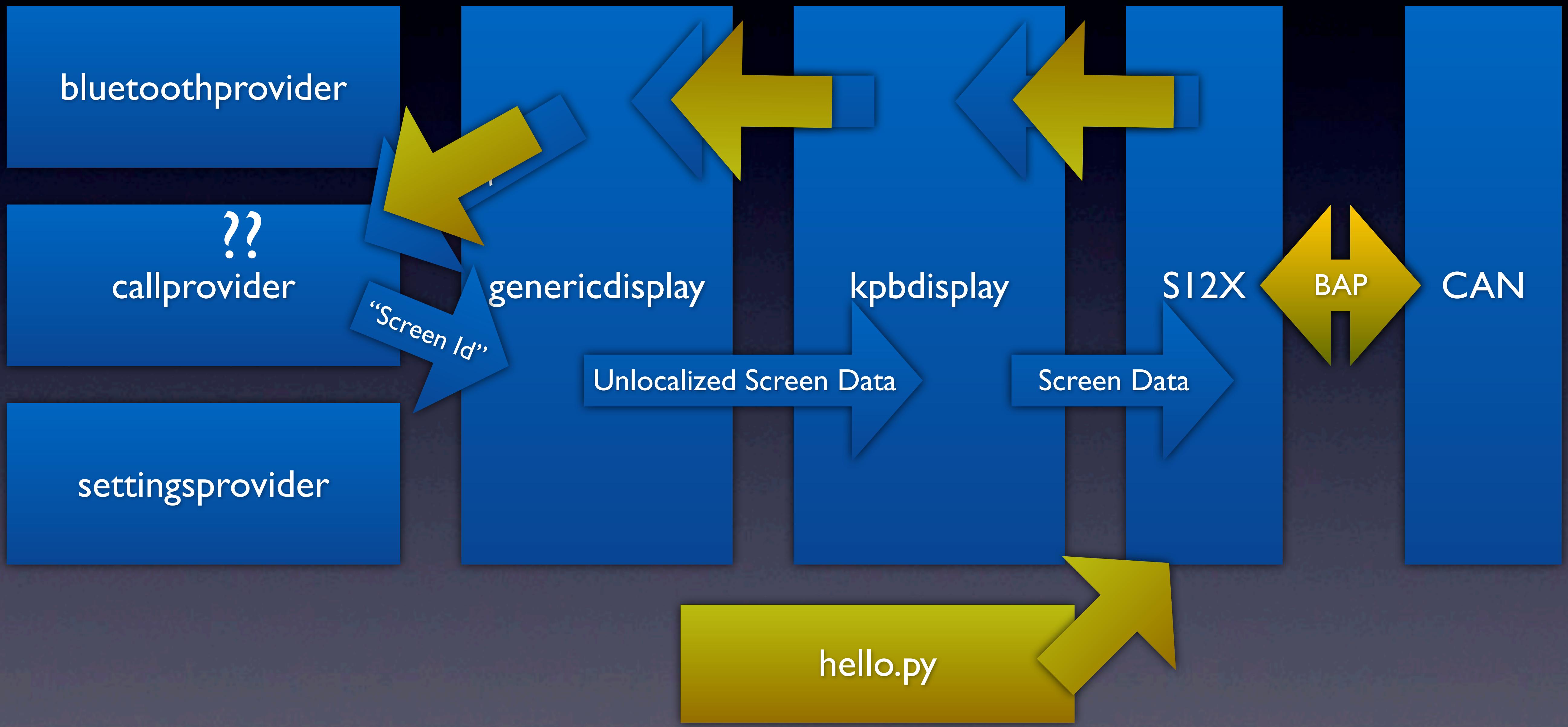
localizes and convert screen data to what the car expects

sends them to S12X -> CAN

“Hello World” in Python

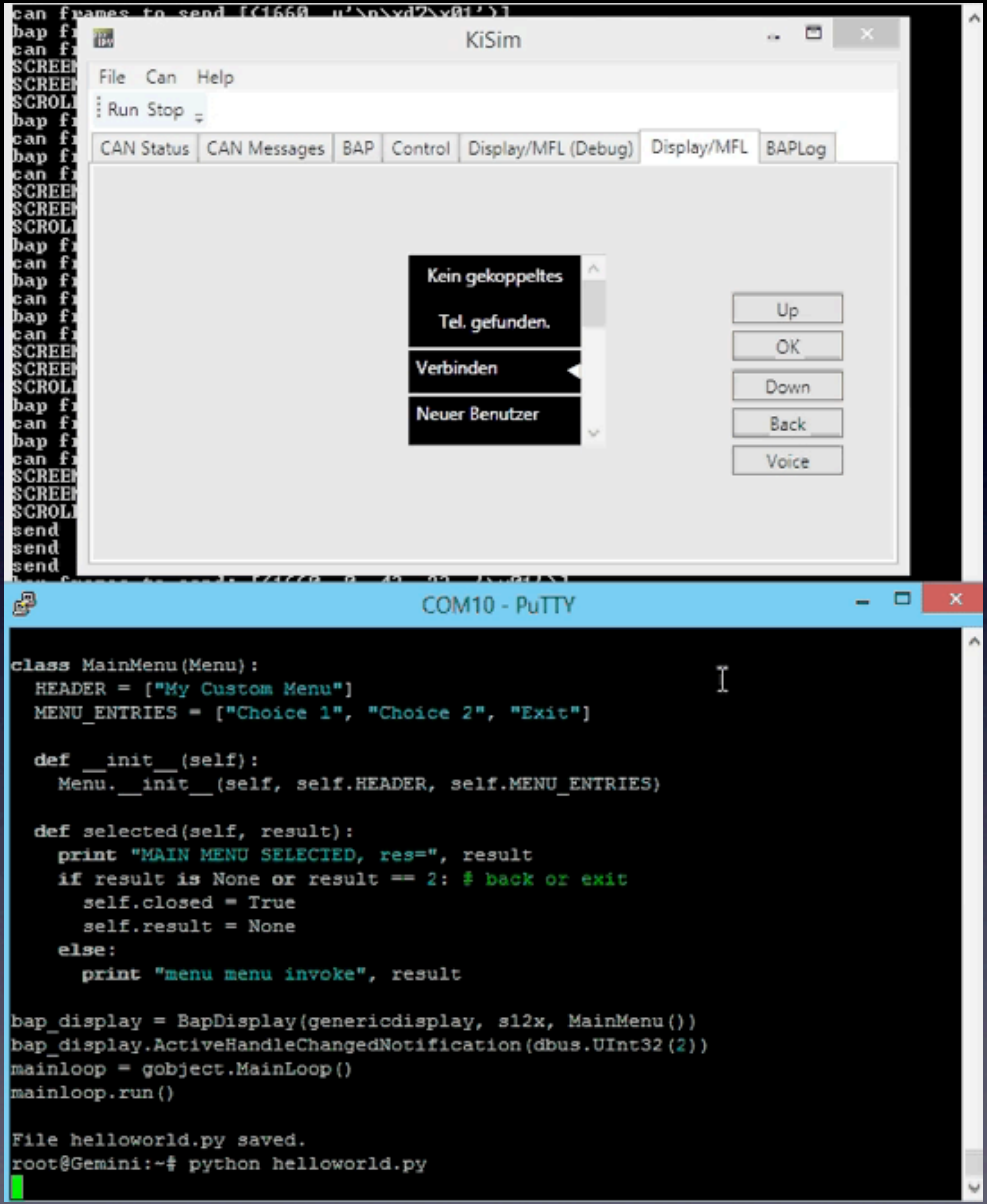
```
import dbus

session_bus = dbus.SessionBus()

s12x = session_bus.get_object(
   "com.nokia.s12xrouter", "/com/nokia/s12xrouter")

s12x.ScreenData(dbus.ByteArray(
   "008001000007000400000004043300120000800b48656c6
   c6f20576f726c64330007010080003300070200800033000
   703008000".decode('hex')))
```

Hijacked Data Flow



Ok, but...

- Conflicts with original Menu
- a few more lines required...
  - create logic screen and set as active
  - All key presses will be forwarded to us
  - no more screen updated from original logic



```
can frames to send [(1660, u'\n\xd2\x01')]
bap fr
can fr
SCREE
SCREE
SCROL
bap fr
can fr
bap fr
can fr
SCREE
SCREE
SCROL
bap fr
can fr
bap fr
can fr
bap fr
can fr
SCREE
SCREE
SCROL
bap fr
can fr
bap fr
can fr
SCREE
SCREE
SCROL
send
send
send
```

```
class MainMenu(Menu):
  HEADER = ["My Custom Menu"]
  MENU_ENTRIES = ["Choice 1", "Choice 2", "Exit"]

  def __init__(self):
    Menu.__init__(self, self.HEADER, self.MENU_ENTRIES)

  def selected(self, result):
    print "MAIN MENU SELECTED, res=", result
    if result is None or result == 2: # back or exit
      self.closed = True
      self.result = None
    else:
      print "menu menu invoke", result

bap_display = BapDisplay(genericdisplay, s12x, MainMenu())
bap_display.ActiveHandleChangedNotification(dbus.UInt32(2))
mainloop = gobject.MainLoop()
mainloop.run()

File helloworld.py saved.
root@Gemini:~# python helloworld.py
```

Why This? (and not a car PC)

- Affordable, automotive-proven hardware
- Lifecycle management (no battery draining!), clean startup and -shutdown
- Well-behaving lower layers implemented - misbehavior on the Linux side should never do any damage due to abstractions

Can This Brick My Car?

- You won’t hear me saying “no”.
- Cars are designed for safety, not security.
- The S12X has fairly extensive sanity checking - sending too many or too few messages will fail “gracefully”.
- However, if you intentionally tunnel the right messages through the gateway, ...

“[...] Ich hatte nach einer Anleitung im Internet einen Adapter gebaut, der während der Fahrt Daten aus den div. Steuergeräten ausliest und die dann in einem Datenlogger auf dem PDA ablegt. [...]
Irgendwann auf dem Weg von Frankfurt nach Cuxhaven ist dann [...] erst die Intrumententafel ausgefallen. [...] nachdem die die Instrumente wieder Werte gezeigt haben [...] ging der Motor aus, die Knöpfe an den Türen hoch und die vorderen Airbags gingen alle auf. (Fahrer- / Beifahrerairbag, Sitzairbag und Gurtstraffer)
[...]
Der Wagen war Schrott (Beide B-Säulen von Gurtstraffern verzogen [...]) und irgendwie kam der Gutachter von der Versicherung dahinter, das der gespeicherte Unfall mit Seitenaufprall nicht über Beschleunigungssensoren im Airbag-Steuergerät kam. Dann natürlich noch die ganze Elektronik mit den Datenkabeln zum Diagnosestecker hinter der Instrumententafel.

War also ein teures Experiment, weil die Versicherung keinen Cent bezahlt hat und der Wagen noch einen Zeitwert von 15.000 Euro hatte.”

http://www.mikrocontroller.net/topic/33809#247235

For The Bad Guys...

- Access to the microphone
- Access to the Internet
- Access to GPS

- Ideal hardware for a tracking bug!
- Short physical access to unit is enough. (But so is for deploying a dedicated bug.)

For The Good Guys...

- All sorts of geofencing applications:
  - home automation
  - electronic parking tickets (which may be a bad idea by itself)
- Weather
- Traffic (Google/Bing traffic to TMC FM modulator, anyone?)

Thanks!

- tmbinc@elitedvb.net
- github.com/tmbinc/car