

# UNLEASHING THE POWER

OF MY 20 YEARS OLD CAR



Stanislas Lejay (WhiteMotion)

Black Hat Europe 2019

# Who's that brat?



- \* Stanislas Lejay, french computer security engineer
- \* Love cars, and to fiddle with things
- \* Automotive vulnerability researcher at WhiteMotion (Tokyo, JP)

**WHITEMOTION**

Automotive Cyber Security

## DISCLAIMER

This work has been done in order to learn about ECUs and to use the full power of my car on racetracks and closed roads only.

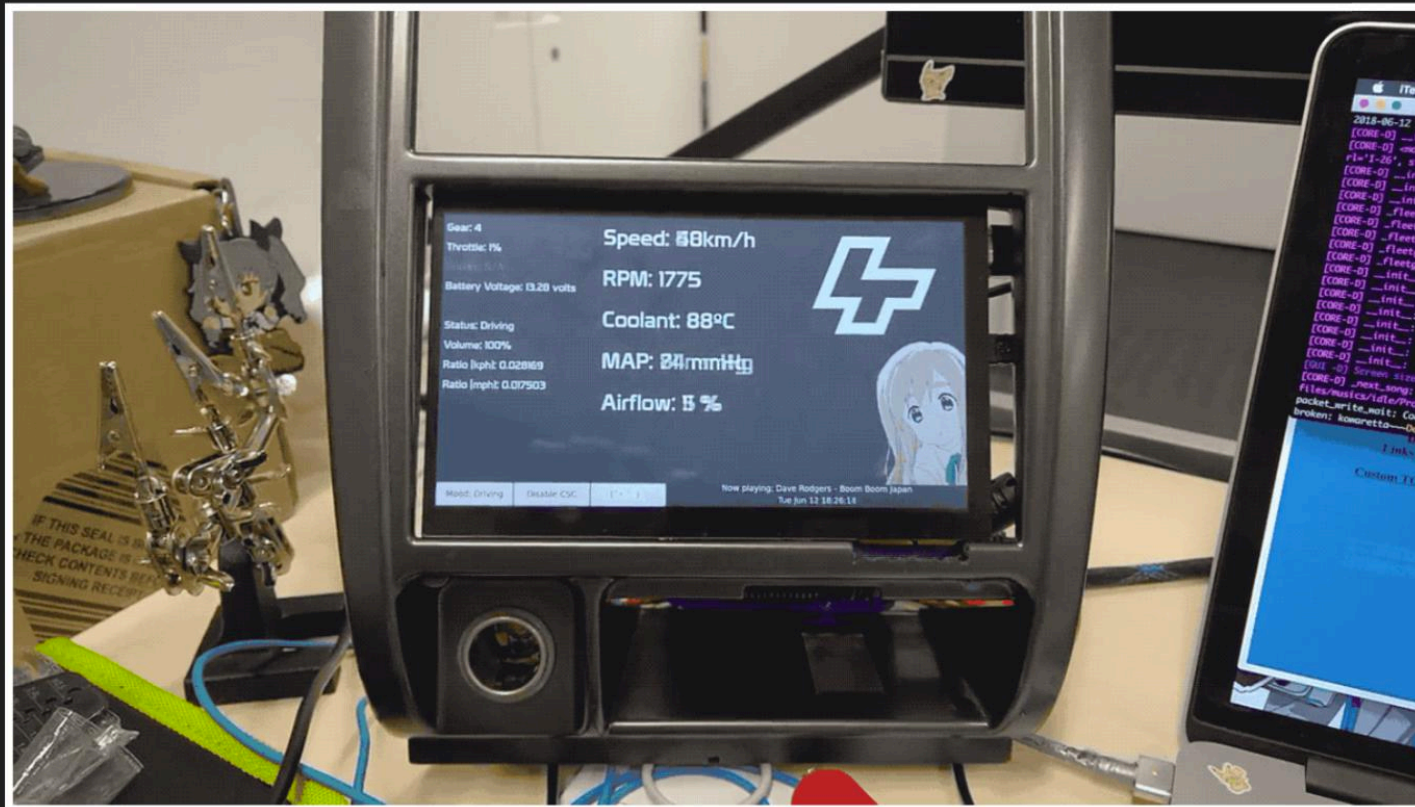
While this can be applied to any car from this era and is a common practice amongst enthusiasts who don't want to go aftermarket, I was asked to remove the car's precise information from this presentation.

# The test subject



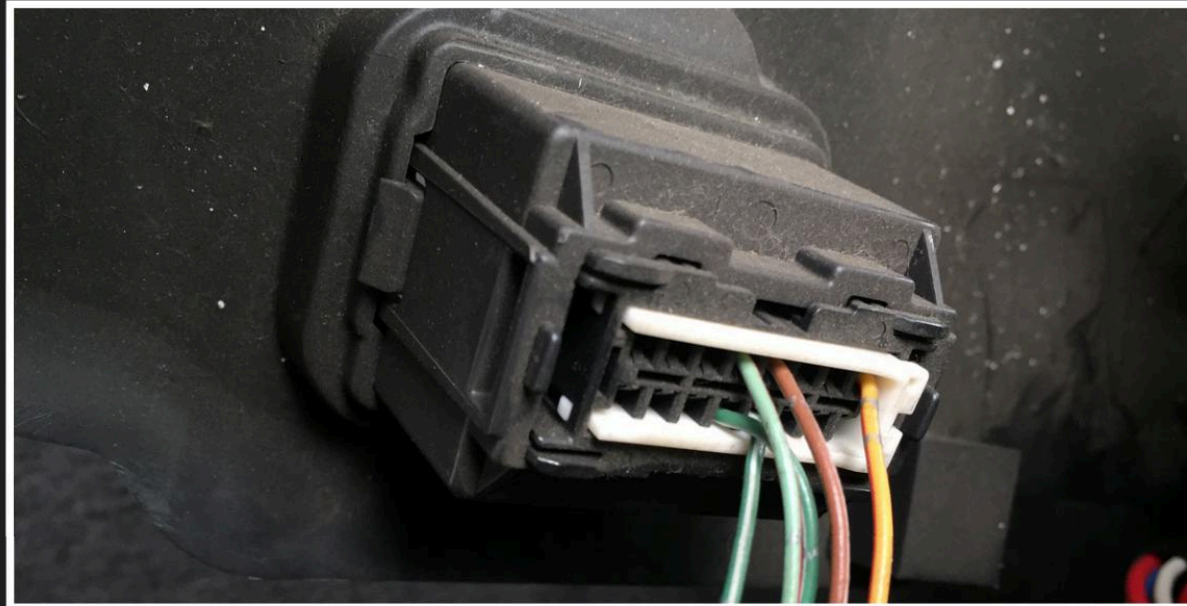
- \* 1997 [MANUFACTURER] [MODEL]
- \* ~300hp from factory, mostly stock but for suspensions
- \* Nicknamed `Little Beast`
- \* 50 000km driven since I bought it, still rips

# Playing around



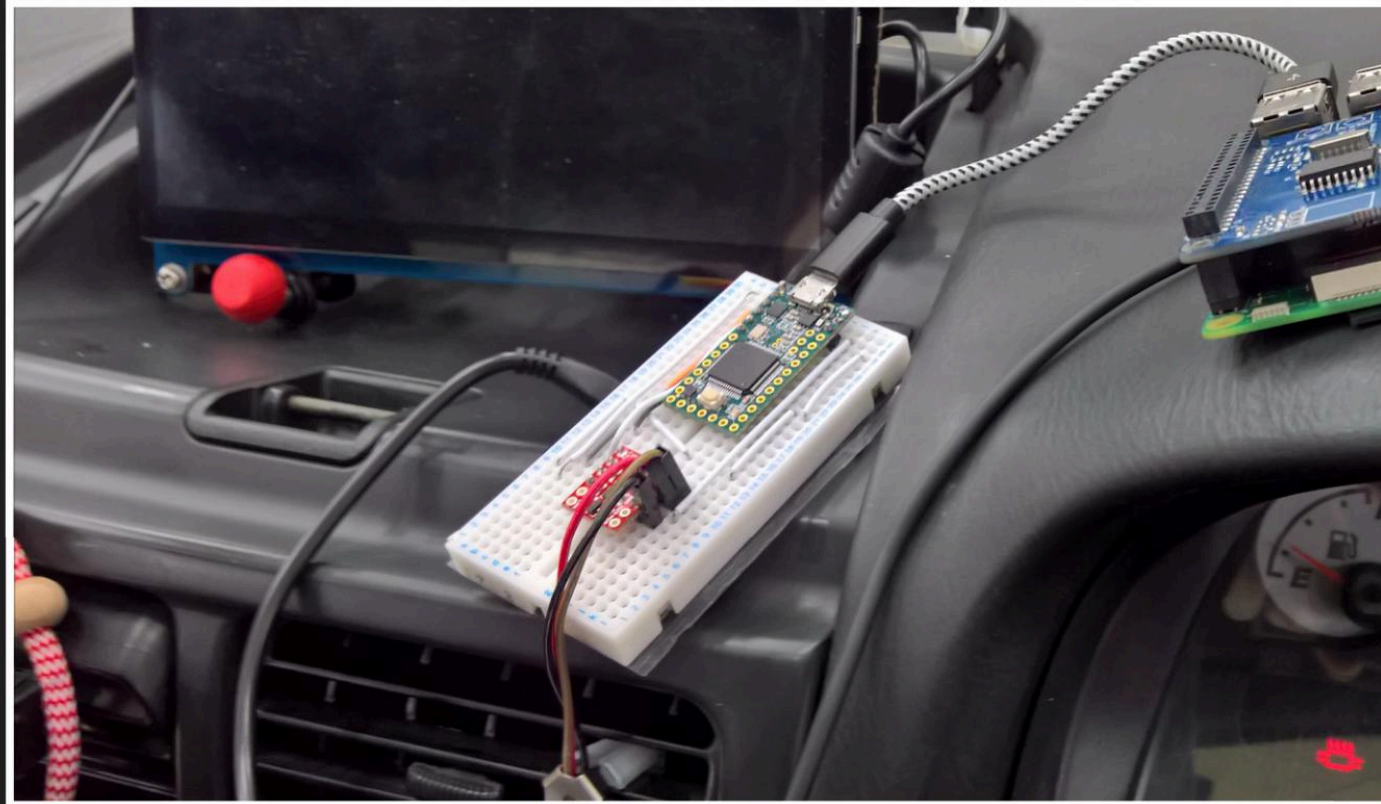
- \* I was developing my own IVI
- \* It has data logging options, alarms, music, etc
- \* Powerful tool when it can communicate with the ECU

# Reconnaissance



- \* How to communicate with the car ?
- \* Features an OBD-II like port, but no CAN
- \* xSM is a proprietary protocol from [THAT MANUFACTURER]
- \* Documented on `[alcyone.org.uk](http://alcyone.org.uk)`
- \* No K-Line connected, xSM1 it is

# xSM1



- \* Simple serial protocol running at 1953 bauds at 5V TTL.
- \* Teensy + level converter is all I need

# Reading from the ECU

```
byte simple_read_data_from_address(short addr) {  
  
    // 78 msb lsb 00  
    byte read_cmd[4] = {0x78, byte(addr >> 8), byte(addr & 0xff), 0x00};  
    byte answer[3] = {0};  
  
    HWSerial.clear();  
    for (int i = 0; i < 4; ++i) {  
        HWSerial.write(read_cmd[i]);  
    }  
  
    HWSerial.flush();  
    HWSerial.readBytes(answer, 3);  
    stop_read();  
    return answer[2];  
}
```



# Dumping the ECU



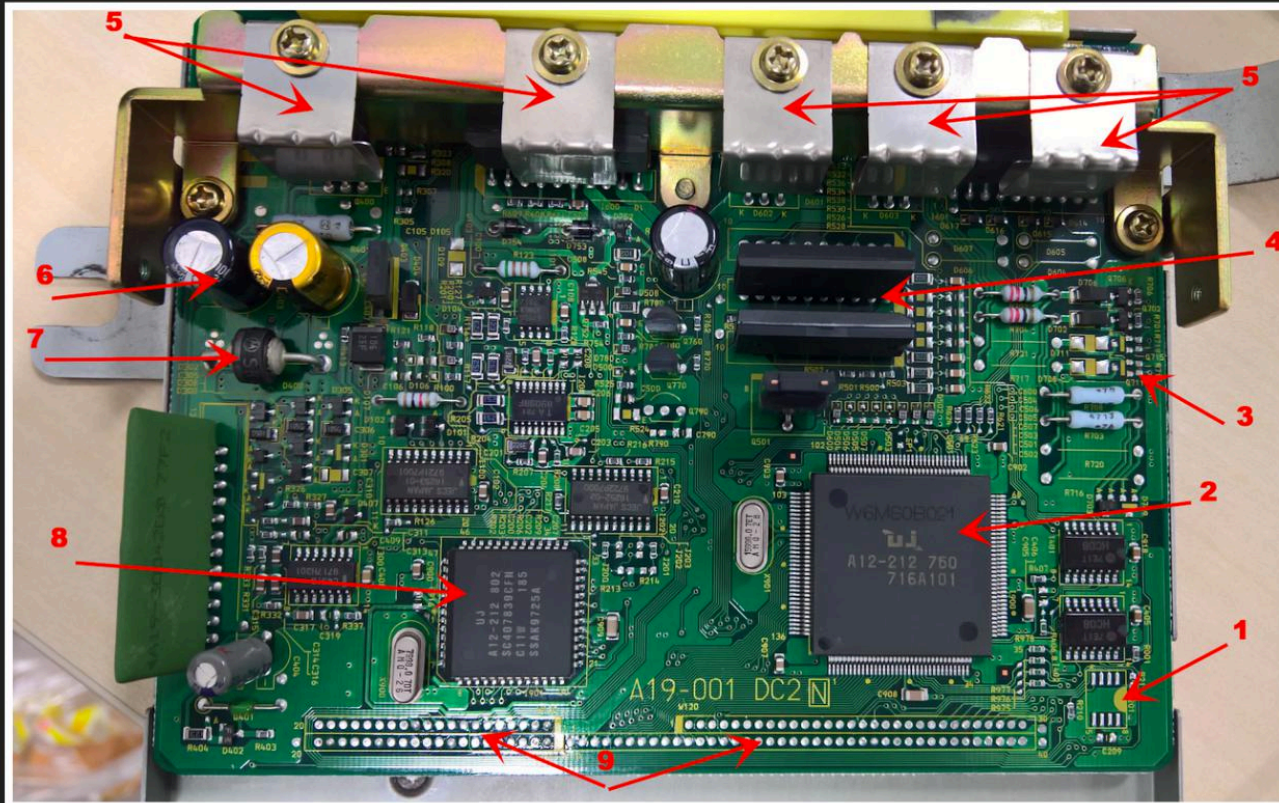
- \* About 5 queries per second
- \* Plug the car's battery to a charger
- \* for loop from 0 to 0xffff, with a few checks
- \* Dump the whole address space in 9ish hours

# Finding the ECU



- \* What architecture is it ? What ISA ?
- \* Need to check the CPU to determine that
- \* Take the ECU from under the passenger carpet (!!)

# ECU and Processor



- 1- Immobiliser Chip (none)
- 2- Main CPU + FW
- 3- Ignition circuitry
- 4- Low power transistors
- 5- High power transistors
- 6- Power management
- 7- Diode
- 8- Secondary custom IC
- 9- Contact strip

- \* IC based on the Mitsubishi M37791 CPU (7700 family)
- \* ``M flag`` changes the instruction decoding at runtime
- \* At least IDA handles it

# Reverse engineering

- \* 64kb blob, where to start ?
- \* Turn everything into code.
- \* Look for maps, find the Xrefs to those maps, and go from here
- \* Struggle until you have something ok-ish (normal RE)

The screenshot displays the IDA Pro interface with several windows open:

- Disassembly Window:** Shows assembly code for RAM:1110 to RAM:123E. The code consists of various instructions like `.BVTI`, `.BVTB`, and `.BVTI` with associated addresses and values.
- Hex View Window:** Shows the raw hex data corresponding to the assembly code, with a color-coded legend on the right.
- 3D Surface Plot:** Titled "advance\_map", it plots RPM (0 to 7000) on the x-axis and Engine Load (0 to 50) on the y-axis. The z-axis represents a value from 0 to 14. The plot shows a high RPM region at low engine loads, which then drops and levels off as engine load increases.
- Symbol Table:** Lists symbols from RAM:1110 to RAM:123E, including `RAM:1120 map2` and `RAM:1220`.

And my IVI works



# History: JDM speed limiters



- \* Cars sold in Japan used to have a ringing sound when going over 100km/h
- \* Nowadays, this became an ECU controlled speed limiter at around 180km/h
- \* Some cars have options for circuits

# My car is no exception

- \* Fuel cut is pretty brutal
- \* Can I get rid of it ?
- \* Need to understand how it works

# Speed limiter: Activation

```
speed_over_or_equal_186: ; Over 186km/h, check if speed limiter already activated  
bbs #2, nsl_bitvector_1, speed_previously_over_188
```

```
cmp A, fixed_value_0x5e ; Speed limiter not activated, compare speed with 188 / 2  
bcc speed_under_188
```

```
seb #2, nsl_bitvector_1 ; Speed over or equal to 188km/h, trigger speed limiter -> Fuel cutoff  
bra speed_over_or_equal_188
```

```
speed_previously_over_188: ; speed limiter already activated, check if  
cmp A, fixed_value_0x5d ; need to maintain it  
beq speed_under_188
```

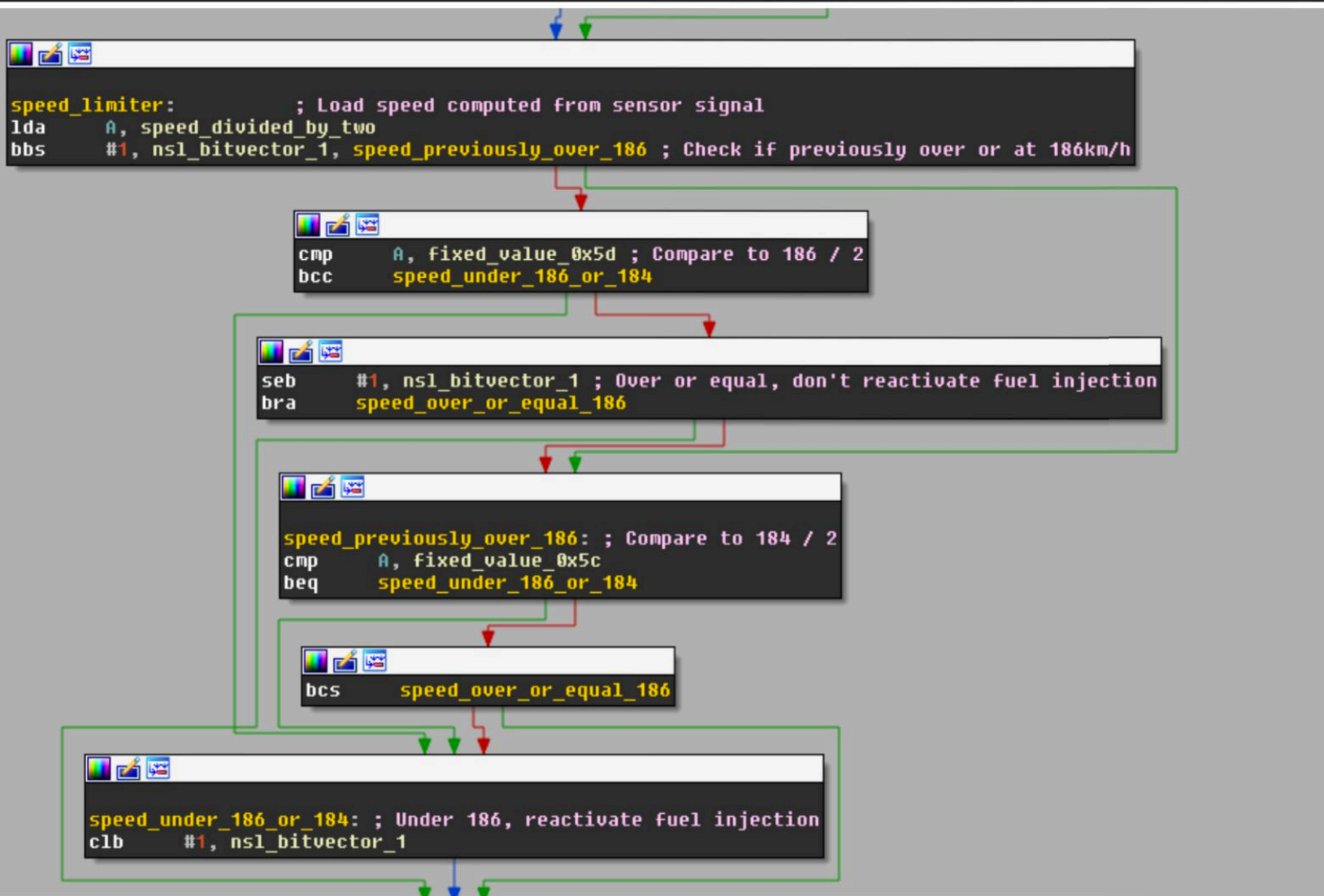
```
bcs speed_over_or_equal_188
```

```
speed_under_188: ; No need to trigger speed limiter  
clb #2, nsl_bitvector_1
```

```
speed_over_or_equal_188: ; Maintain speed limiter  
ldx word_000008DE  
bbs #10h, nsl_bitvector_2, loc_00008252
```



# Speed limiter: Deactivation



The different options

# Bypassing the limiter: Aftermarket ECU



入札件数 残り時間  
0 入札履歴 11 時間 詳細 ウォッチ

現在価格  
50,000円 (税 0 円)

送料 落札者負担 配送方法の詳細

**入札する**

出品者情報 **落札率が高い**  
zztmiaさん フォロー

総合評価: **442** | **良い評価 99.8%**  
出品者のその他のオークションを見る  
出品地域: 富山県

新着出品のお知らせ登録

出品者へ質問 回答済み 1 件

更新情報

- \* Easiest and most customizable solution but
  - Not cheap
  - Needs a retune
  - I'm losing all the work done so far

# Bypassing the limiter: Daughterboard

Board 97/8 (Optional discounted cable and software available).

Category: [Store](#)



SKU 00002

**£315.00**

**Installation service**

- Add Install (+£30.00)

**Cable and software**

- SSM (+£95.00)
- OBD (+£95.00)

**Motorsports Pack**

- Motorsports Pack (Anti Lag, Launch Control, Flat Shift and Traction Control) (+£100.00)

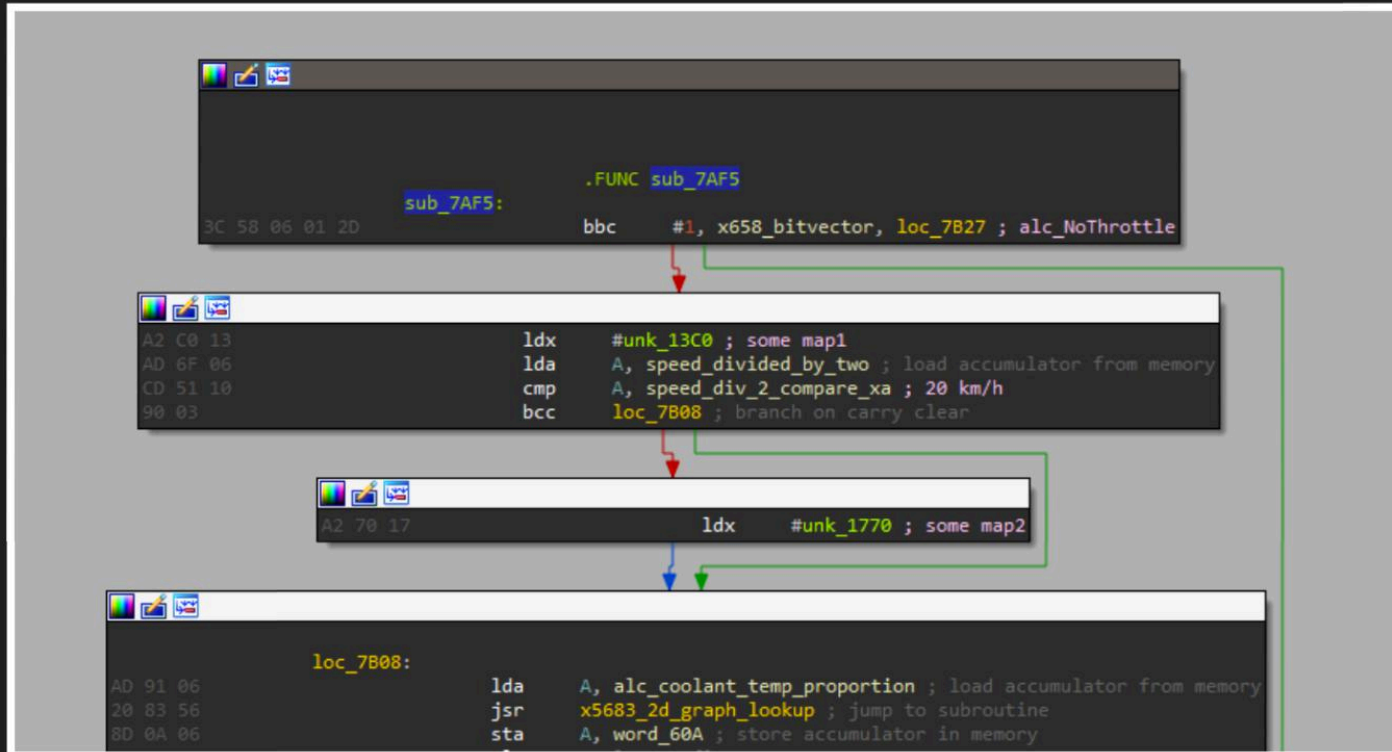
Please type the large 2 digit ECU code here (e.g. 8S, 6S, 1S, 7S, 9S, 4W, 5W, 4G, 5G, etc).

Qty (3 available)

[Add to Bag](#)

- \* Piggybacks on the stock ECU but
  - Not cheap
  - Needs a retune

# Bypassing the limiter: Cut VSS



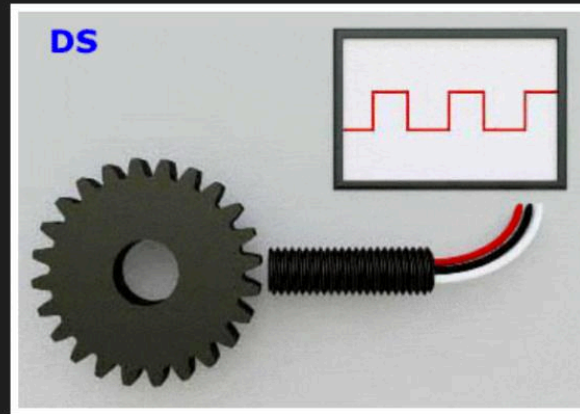
- \* Easiest, cheapest, easiest to understand solution but...
- \* Transition Maps
- \* Verification in code leading to limp mode

# Bypassing the limiter: Faking the VSS signal



- \* HKS sells the "Speed Limit Defencer" (SLD)
- \* Not cheap (100-200 euros)

# Faking the VSS signal for cheap\*



- \* SLD's principle is fairly easy
- \* Intercept the signal and send a dummy one if needed
- \* The ECU still receives a high speed signal, but not too high

GIF SOURCE: [HTTPS://SENSORSO.COM/GEAR-DETECTION-SENSORS.HTML](https://sensorso.com/gear-detection-sensors.html)

# Teensy for the win

```
// Initial version of the of the limiter bypass
while (speed >= 180) {
    if (previous_signal_value) {
        signal_value = 0;
    } else {
        signal_value = 1023;
    }
    analogWrite(VSS_TX, signal_value);

    // Remove the delay to get 354km/h and stall the engine
    delay(5); // Speed recorded is around 140km/h

    previous_signal_value = signal_value;
}
```

and then...



# Results



# Black Hat Sound Bytes

- \* Most aftermarket tools are not witchcraft
- \* ECUs are getting complicated, but the basics stay the same
- \* Go simple, but go safer

Thank you!



Twitter: 0xP1kachu